Determining the Stego Algorithm for JPEG Images

Tomáš Pevný[a], Jessica Fridrich[b]

June 12, 2006

[a]Department of Computer Science, [b]Department of Electrical and Computer Engineering, SUNY Binghamton, Binghamton, NY 13902–6000

Abstract

The goal of Forensic Steganalysis is to detect the presence of embedded data and eventually extract the secret message. A necessary step toward extracting the data is determining the steganographic algorithm used to embed the data. In this paper, we construct blind classifiers capable of detecting steganography in JPEG images and assigning stego images to 6 popular JPEG embedding algorithms. The classifiers are support vector machines that use 23 calibrated DCT feature calculated from the luminance component.

1 Introduction

The goal of Steganography is to hide the very presence of communication by embedding messages in innocuous looking objects, such as digital images. To embed the data, the original (cover) image is slightly modified and becomes the stego image. The embedding process is usually controlled using a secret stego key shared between the communicating parties. The most important requirement of any steganographic system is statistical undetectability of the hidden data given the complete knowledge of the embedding mechanism and the source of cover objects but not the stego key (so called Kerckhoffs' principle). Attempts to formalize the concept of steganographic security include [2, 6, 29, 19].

The goal of Steganalysis is discovering the presence of hidden messages and determining their attributes. In practice, a steganographic scheme is considered secure if no existing steganalytic attack can be used to distinguish between cover and stego objects with success significantly better than random guessing [7]. There are two major classes of steganalytic methods—targeted attacks and blind steganalysis. Targeted attacks use the knowledge of the embedding algorithm [11], while blind approaches are not tailored to any specific embedding paradigm [9, 8, 4, 5, 21, 28, 15]. Blind approaches can be thought of as practical embodiments of Cachin's [6] definition of steganographic security. It is assumed that "natural images" can be characterized using a small set of numerical features. The distribution of features for natural cover images is then mapped out by computing the features for a large database of images. Using methods of artificial intelligence or pattern recognition, a classifier is then built to distinguish in the feature space between natural images and stego images.

Avcibas et al. [4] were the first who proposed the idea to use a trained classifier to detect and classify robust data hiding algorithms. The authors used image quality metrics as features and tested their method on three watermarking algorithms. Avcibas et al. [5, 3] later proposed a different set of features based on binary similarity measures between the lowest bit-planes. Farid [8, 21] constructed the features from higher-order moments of distribution of wavelet coefficients from several high-frequency sub-bands and their local linear prediction errors. Harmsen et al. [15] proposed to use the center of gravity of the Histogram Characteristic Function to detect additive noise steganography in the spatial domain. Inspired by this work, Xuan et al. [28] used absolute moments of the histogram characteristic function constructed in the wavelet

domain for blind steganalysis. Goljan et al. [14] calculate the features as higher-order absolute moments of the noise residual in the wavelet domain.

Compared to targeted schemes, blind approaches have certain important advantages. They are potentially capable of detecting previously unseen steganographic methods and they can assign stego images to a known steganographic algorithm based on the location of the feature vector in the feature space. This is because different steganographic algorithms introduce different artifacts into images and thus leave a specific fingerprint. For example, the shrinkage in the F5 algorithm [27] leaves a characteristic artifact in the histogram, which consists of a larger number of zeros and slightly decreased number of all non-zero DCT coefficients. In contrast, other programs, such as OutGuess [23] or Model Based Steganography [24], preserve the histogram. By using a large number of features, rather than just the histogram, we increase the chances that two different embedding programs will indeed produce images located in different parts of the feature space.

Knowing the program used to embed the secret data, a forensic examiner can continue the steganalysis with brute-force searches for the stego/encryption key and eventually extract the secret message. Since JPEG images are by far the most common image format in current use, we narrow the study in this paper to the JPEG format. Our goal is to construct blind steganalyzers for JPEG images capable of stego algorithm identification that would give reliable results for JPEG images of arbitrary quality factor and that would correctly handle double-compressed images (which is an issue so far largely avoided in previous works on steganalysis of JPEGs). Construction of such a classifier requires large training image databases and extensive computational and storage resources. In this paper, we construct two steganalyzers—a bank of classifiers for single-compressed images that can reliably assign stego images to 6 popular JPEG steganographic techniques and a classifier that assigns double-compressed stego images to either OutGuess or F5, which are the only two programs that we tested that can produce double compressed images. This paper can be considered as an extension of our previous work on this subject [13, 9].

In the next section, we describe the construction of calibrated DCT features that will be used to construct our classifiers. In Section 3, we give some implementation details of the support vector machines (SVM) used in this paper and discuss training and testing procedures. We also describe the database of test images for training and testing. In Section 4, we construct a seven-class SVM multi-classifier that assigns single-compressed JPEG images with a wide range of quality factors to either the class of cover images or six JPEG steganographic algorithms (F5 [27], OutGuess [23], Steghide [16], JP Hide&Seek [1], Model based steganography with and without deblocking [24, 25, 26]). Section 5 is entirely devoted to steganalysis of double-compressed images. We first explain how the process of calibration must be modified to correctly account for double compression. Then, we build a three-class SVM that can assign a double-compressed JPEG image to either the class of cover images or double-compressed images embedded using F5 or OutGuess. The experimental results from both classifiers are interpreted in their corresponding sections. The paper is concluded in Section 6.

2 DCT Features

Our choice of the features for blind JPEG steganalysis is determined by our highly positive previous experience with DCT features [9] and the comparisons reported in [13, 20]. Both studies report the superiority of JPEG classifiers that use calibrated DCT features. We note that the results presented here pertain to the selected feature set and different results might be obtained for a different feature set.

In this section, we briefly describe the features (see Figure 1), referring to [9] for more details. For now, let us assume that the stego image has *not* been double compressed. The process of calculating the features starts with a vector functional $\mathbf{F}$ that is applied to the stego JPEG image J_1. For instance, $\mathbf{F}$ could be the histogram of all luminance coefficients. The stego image J_1 is decompressed to the spatial domain, cropped by a few pixels in each direction, and recompressed with the quantization table from J_1 to obtain J_2. The same vector functional $\mathbf{F}$ is then applied to J_2. The *calibrated* scalar feature f is obtained as a difference, if

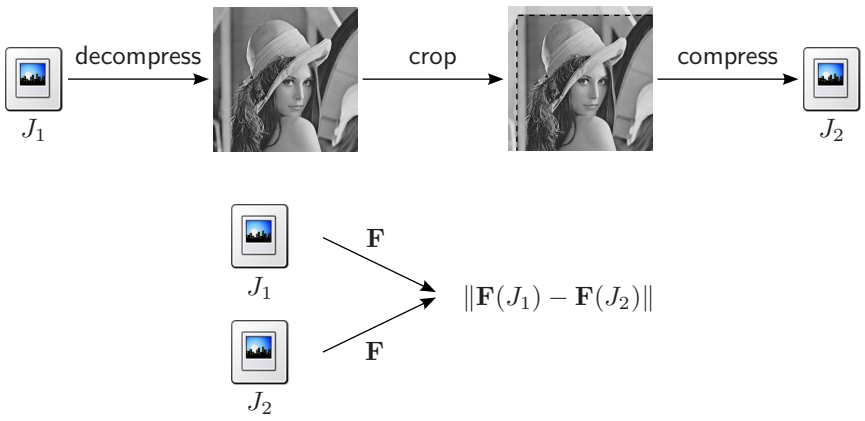


Figure 1: Calibrated features f are obtained from functionals $\mathbf{F}$.

$\mathbf{F}$ is a scalar, or an L_1 norm if $\mathbf{F}$ is a vector or matrix

$$f = ||\mathbf{F}(J_1) - \mathbf{F}(J_2)||_{L_1}. \tag{1}$$

The cropping and recompression produce a "calibrated" JPEG image with most macroscopic features similar to the original cover image. This is because the cropped stego image is perceptually similar to the cover image and thus its DCT coefficients should have approximately the same statistical properties as the cover image. The cropping is important because the 8×8 grid of recompression is "out of sync" with the previous JPEG compression, which suppresses the influence of the previous compression (and embedding) on the coefficients of J_2. The operation of cropping can obviously be replaced with slight rescaling, rotation, or even non-linear warping (Stirmark). Because the cropped and recompressed image is an approximation to the cover JPEG image, the net effect of calibration is a decrease in image-to-image variations. We now define all 23 functionals used for steganalysis.

We will represent the luminance of a stego JPEG file with a DCT coefficient array $d_{ij}(k)$, $i, j = 1, \ldots, 8, k = 1, \ldots, n_B$, where $d_{ij}(k)$ denotes the (i, j)-th quantized DCT coefficient in the k-th block (there are total of n_B blocks).

The first vector functional is the histogram $\mathbf{H}$ of all $64 \times n_B$ luminance DCT coefficients

$$\mathbf{H} = (H_L, \ldots, H_R), \tag{2}$$

where $L = \min_{i,j,k} d_{ij}(k)$, $R = \max_{i,j,k} d_{ij}(k)$.

The next 5 vector functionals are histograms

$$\mathbf{h}^{ij} = (h_L^{ij}, \ldots, h_R^{ij}), (i, j) \in \{(1, 2), (2, 1), (3, 1), (2, 2), (1, 3)\} \tag{3}$$

of DCT coefficients of the 5 lowest frequency AC modes $(i, j) \in \mathcal{L} \triangleq \{(1, 2), (2, 1), (3, 1), (2, 2), (1, 3)\}$.

The next 11 functionals are 8×8 matrices $\mathbf{g}_{ij}^d$, $i, j = 1, \ldots, 8$, called "dual" histograms

$$\mathbf{g}_{ij}^d = \sum_{k=1}^{n_B} \delta(d, d_{ij}(k)), d = -5, \ldots, 5, \tag{4}$$

where $\delta(x, y) = 1$ if $x = y$ and 0 otherwise.

The next 6 functionals capture inter-block dependency among DCT coefficients. The first scalar functional is the variation V

$$V = \frac{\sum_{i,j=1}^{8} \sum_{k=1}^{|\mathbf{I}_r|-1} |d_{ij}(\mathbf{I}_r(k)) - d_{ij}(\mathbf{I}_r(k+1))| + \sum_{i,j=1}^{8} \sum_{k=1}^{|\mathbf{I}_c|-1} |d_{ij}(\mathbf{I}_c(k)) - d_{ij}(\mathbf{I}_c(k+1))|}{|\mathbf{I}_r| + |\mathbf{I}_c|}, \tag{5}$$

where $\mathbf{I}_r$ and $\mathbf{I}_c$ stand for the vectors of block indices $1, \ldots, n_B$ while scanning the image by rows and by columns, respectively.

The next two blockiness functionals are scalars calculated from the decompressed JPEG image representing an integral measure of inter-block dependency over the whole image

$$B_\alpha = \frac{\sum_{i=1}^{\lfloor (M-1)/8 \rfloor} \sum_{j=1}^{N} |\mathbf{c}_{8i,j} - \mathbf{c}_{8i+1,j}|^\alpha + \sum_{j=1}^{\lfloor (N-1)/8 \rfloor} \sum_{i=1}^{M} |\mathbf{c}_{i,8j} - \mathbf{c}_{i,8j+1}|^\alpha}{N \lfloor (M-1)/8 \rfloor + M \lfloor (N-1)/8 \rfloor}. \tag{6}$$

In (6), M and N are image height and width in pixels and $\mathbf{c}_{i,j}$ are greyscale values of the decompressed JPEG image.

The remaining three *functionals* are calculated from the co-occurrence matrix of DCT coefficients from neighboring blocks

$$\begin{aligned} N_{00} =& \mathbf{C}_{0,0}(J_1) - \mathbf{C}_{0,0}(J_2) \\ N_{01} =& \mathbf{C}_{0,1}(J_1) - \mathbf{C}_{0,1}(J_2) + \mathbf{C}_{1,0}(J_1) - \mathbf{C}_{1,0}(J_2) + \mathbf{C}_{-1,0}(J_1) - \mathbf{C}_{-1,0}(J_2) + \mathbf{C}_{0,-1}(J_1) - \mathbf{C}_{0,-1}(J_2) \\ N_{11} =& \mathbf{C}_{1,1}(J_1) - \mathbf{C}_{1,1}(J_2) + \mathbf{C}_{1,-1}(J_1) - \mathbf{C}_{1,-1}(J_2) + \mathbf{C}_{-1,1}(J_1) - \mathbf{C}_{-1,1}(J_2) + \mathbf{C}_{-1,-1}(J_1) - \mathbf{C}_{-1,-1}(J_2), \end{aligned} \tag{7}$$

where

$$\mathbf{C}_{st} = \frac{\sum_{i,j=1}^{8} \sum_{k=1}^{|\mathbf{I}_r|-1} \delta\left(s, d_{ij}(\mathbf{I}_r(k))\right) \delta\left(t, d_{ij}(\mathbf{I}_r(k+1))\right) + \sum_{i,j=1}^{8} \sum_{k=1}^{|\mathbf{I}_c|-1} \delta\left(s, d_{ij}(\mathbf{I}_c(k))\right) \delta\left(t, d_{ij}(\mathbf{I}_c(k+1))\right)}{|\mathbf{I}_r| + |\mathbf{I}_c|}. \tag{8}$$

The argument J_1 or J_2 in (7) denotes the image to which the coefficient array $d_{ij}(k)$ corresponds.

3 Classifier Construction

In our work, we used soft-margin SVMs [8, 20] with Gaussian kernel $\exp(-\gamma \|x - y\|^2)$. Soft-margin SVMs can be trained on non-separable data by penalizing incorrectly classified images with a factor $C \cdot d$, where d is the distance from the separating hyperplane and C is a constant. The role of the parameter C is to force the number of incorrectly classified images during training to be small. If incorrectly classified images from both classes have the same cost, C is the same for both cover and stego images. In steganography, however, false positives (cover images classified as stego) have associated a much higher cost than missed detection (stego images classified as cover). This is because images labeled as stego must be further analyzed using brute force searches for the secret stego key. To train an SVM with uneven cost of incorrect classification, we penalize incorrectly classified images using two different penalty parameters C_{FP} and C_{FN}, where the subscripts FP and FN stand for false positives and false negatives, respectively.

The parameters must be determined prior to training an SVM. For binary SVMs that classify between two classes of stego images, e.g., between F5 and OutGuess, we assign equal cost to both classes. Thus, we only need to determine two parameters—(γ, C). For SVMs that classify between cover and stego images, we need to determine three parameters (γ, C_{FP}, C_{FN}). Following the advice in [17], we calculated the parameters through a search on a multiplicative grid with n_{cv}-fold cross-validation. After dividing the training set into n_{cv} distinct subsets (e.g., $n_{cv} = 5$), $n_{cv} - 1$ of them were used for training and the remaining n_{cv}-th subset was used to calculate the validation error, false positive, and missed detection rates. This process was repeated n_{cv} times for each subset and the results were averaged to obtain the final parameter values. The averages are essentially estimates of the performance on unknown data. For SVMs classifying into two stego classes, the final values of the parameters were determined by the smallest estimated classification error. For SVMs classifying between the cover and stego classes, the parameters were determined by the smallest estimated missed detection rate under the condition that the estimated false positive rate is below 1%. If none of the

estimated false positive rates was below 1%, the parameters were determined by the smallest estimated false positive rate. The multiplicative grids for each SVM are described in the corresponding sections.

Prior to training, all elements of the feature vector were scaled to the interval $[-1, +1]$. The scaling coefficients were always derived from the training set. For the n_{cv}-fold cross-validation, the scaling coefficients were calculated from the $n_{cv} - 1$ subsets.

There exist several extensions of SVMs to enable them to handle more than two classes. The approaches can be roughly divided into two groups—"all-together" methods and methods based on binary (two-class) classifiers. A good survey with comparisons is the paper by Hsu [18], where the authors conclude that methods based on binary classifiers are typically better for practical applications. We tested the "max-wins" and Directed Acyclic Graph SVMs [22]. Since both approaches had very similar performance in our tests, we decided to use the "max-wins" classifier in all our tests. This method employs $\binom{n}{2}$ binary classifiers for every pair of classes (n is the number of classes into which we wish to classify). During classification, the feature vector is presented to all $\binom{n}{2}$ binary classifiers and the histogram of their answers is created. The class corresponding to the maximum value of the histogram is selected as the final target class. If there are two or more classes with the same number of votes, one of the classes is randomly chosen.

3.1 Image Database

Our image database contained approximately 6000 images of natural scenes taken under varying conditions (outside and inside images, images taken with and without flash and at various ambient temperatures) with the following digital cameras: Nikon D100, Canon G2, Olympus Camedia 765, Kodak DC 290, Canon PowerShot S40, images from Nikon D100 downsampled by a factor of 2.9 and 3.76, Sigma SD9, Canon EOS D30, Canon EOS D60, Canon PowerShot G3, Canon PowerShot G5, Canon PowerShot Pro 90IS, Canon PowerShot S100, Canon PowerShot S50, Nikon CoolPix 5700, Nikon CoolPix 990, Nikon CoolPix SQ, Nikon D10, Nikon D1X, Sony CyberShot DSC F505V, Sony CyberShot DSC F707V, Sony CyberShot DSC S75, and Sony CyberShot DSC S85. All images were taken either in the raw raster format TIFF or in proprietary manufacturer raw data formats, such as NEF (Nikon) or CRW (Canon). The proprietary raw formats were converted to BMP using the software provided by the manufacturer. The image resolution ranged from 800×631 for the scaled images to 3008×2000. We have included scaled images into the database, because images are usually resized when shared via e-mail.

For experiments with single-compressed images (Section 4), we divided all images into two disjoint groups. The first group was used for training and consisted of 3500 images from the first 7 cameras in the list (including the downsampled images). The second group contained the remaining 2500 images and was used for testing. Thus, no image or its different forms were used simultaneously for testing and training. This strict division of images also enabled us to estimate the performance on never seen images from a completely different source.

The database for the second experiment on double-compressed images (Section 5) was a subset of the larger database consisting of only 4500 images. This measure was taken to decrease the total computational time.

4 Multi-Classifier for Single-Compressed Images

In this section, we build a steganalyzer that can assign stego JPEG images to 6 known JPEG steganographic programs. We also require this steganalyzer to be able to handle single-compressed JPEG images with a wide range of quality factors. Instead of adding the JPEG quality factor as an additional 24-th feature, we opted for training a separate multi-classifier for each quality factor. This classifier bank performed better than one classifier with an additional feature. Also, the training can be done faster this way, because the complexity of training SVMs is $O(n_{im}^3)$, where n_{im} is the number of training images. In order to cover a wide range of quality factors with feasible computational and storage requirements, we selected the following grid of 18 quality factors $\mathcal{Q}_{18} = \{63, 67, 69, 71, 73, 75, 77, 78, 80, 82, 83, 85, 88, 90, 92, 94, 96\}$.

We prepared the stego images by embedding a random bit-stream of different relative lengths using the following six algorithms—F5 [27], Model based steganography without (MB1) and with deblocking (MB2) [24, 25, 26], JP Hide&Seek [1], OutGuess ver. 0.2 [23], and Steghide [16]. For F5, MB1, JP Hide&Seek, OutGuess, and Steghide, we embedded messages of three different lengths—100%, 50%, 25% of the maximal image embedding capacity. By maximal embedding capacity, we understand the length of the largest message that is possible to embed in a particular image by a particular program. In compliance with the directions provided by its author, for JP Hide&Seek we assumed that the embedding capacity of the image is equal to 10% of the image file size. For MB2, we only embedded messages of one length equivalent to 30% of the embedding capacity of MB1 to minimize the cases when the deblocking algorithm fails.

F5 and OutGuess are the only two programs that always decompress the cover image before embedding and embed data during recompression. Both algorithms, however, also accept raster lossless formats ("png" for F5 and "ppm" for OutGuess), in which case the stego image is not double-compressed. We also note that OutGuess had to be modified to allow saving the stego image at quality factors lower than 75.

4.1 Training

The max-wins multi-classifier trained for each quality factor employs $\binom{n}{2} = 21$ binary classifiers for every pair out of $n = 7$ classes. For each classifier, the training set consisted of 3400 cover and 3400 stego images. If, for a given class, more than one message length was available (all algorithms except MB2 and cover), the training set had an equal number of stego images embedded with message lengths corresponding to 100%, 50%, and 25% of the algorithm embedding capacity. The total number of images used for training of all 18 multi-classifiers combined was thus approximately $18 \times 17 \times 3500 = 1,071,000$ (there are 18 quality factors and 3 message lengths for 5 stego programs, plus one for MB2 and cover). For SVMs classifying into two stego classes, the parameters (γ, C) were determined by a grid-search on the multiplicative grid

$$(\gamma, C) \in \left\{(2^i, 2^j) | i \in \{-5, \ldots, 3\},\ j \in \{-2, \ldots, 9\}\right\}. \tag{9}$$

The parameters (γ, C_{FP}, C_{FN}) for SVMs classifying between the cover class and a stego class were determined on the grid

$$(\gamma, C_{FP}, C_{FN}) \in \left\{(2^i, 10 \cdot 2^j, 2^j), (2^i, 100 \cdot 2^j, 2^j) | i \in \{-5, \ldots, 3\},\ j \in \{-2, \ldots, 9\}\right\}. \tag{10}$$

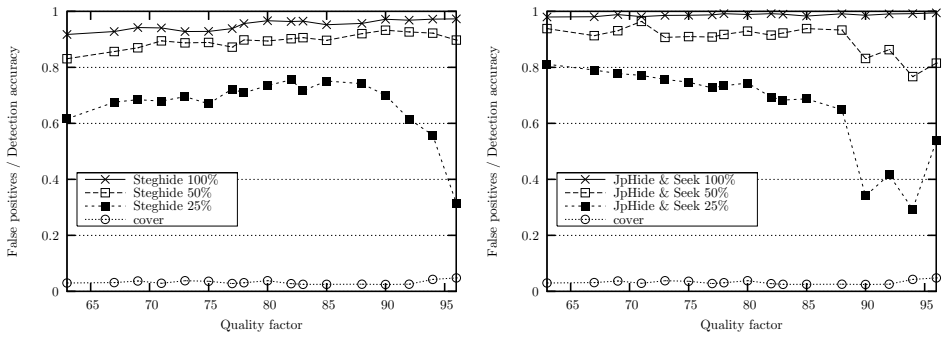
In both cases, 5-fold cross-validation was used to estimate the performance. Due to the large number of parameters for each classifier $(18 \times (6 \times 3 + 15 \times 2))$, we do not include them in this paper.

4.2 Testing and discussion

The testing database consisted of 2500 source images never seen by the classifier and their embedded versions prepared in the same manner as the training set. Out of the 2500 images, 1000 were taken by cameras used for producing the training set, while the remaining 1500 all came from cameras not used for the training set. The whole testing set for all quality factors contained approximately $18 \times 17 \times 2500 = 765,000$ images.

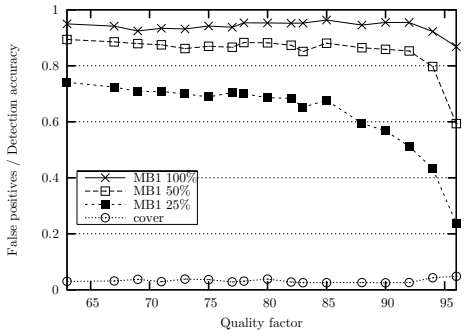
In Table 1, we show, as an example, the confusion matrix for the multi-classifier trained for the quality factor 75 and tested on images of the same quality factor. The multi-classifier reliably detects stego images for message lengths 50% or larger. For fully embedded images, the classification accuracy is 93–99% with false negative rate of 0.44%. JP Hide&Seek and OutGuess are consistently the easiest to detect for all message lengths, while MB2 and MB1 are the least detectable methods. At low embedding rates, the detection of F5 is also lower when compared to other methods, which is likely due to matrix embedding that decreases the number of embedding changes.

With decreasing message length, the results of the classification naturally become progressively worse. At this point, we would like to point out that there are certain fundamental limitations that cannot be overcome. In particular, it is not possible to distinguish between two algorithms that employ the same embedding mechanism by inspecting the statistics of DCT coefficients. For example, two algorithms that use LSB

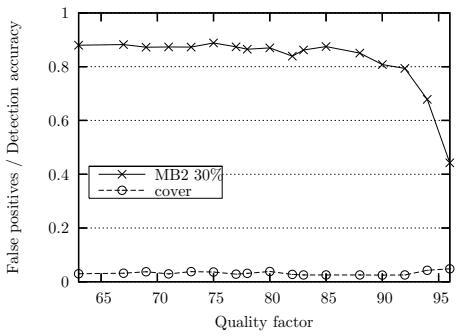


(a) Steghide

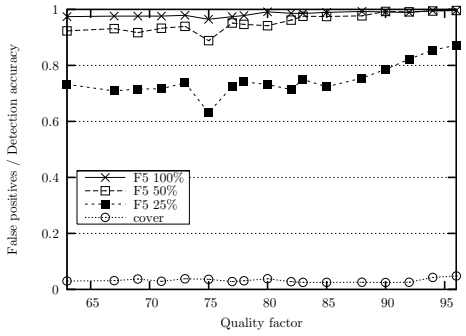
(b) Jp Hide&Seek



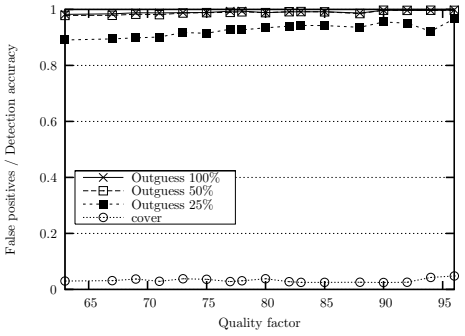
(c) MB1



(d) MB2



(e) F5



(f) OutGuess

Figure 2: Percentage of correctly classified images embedded with a given stego algorithm and false positives (percentage of cover images detected as stego) for all 18 multi-classifiers. Each figure corresponds to one stego algorithm and each curve to one relative payload.

	Classified as						
Embedding algorithm	Cover	F5	JP Hide&Seek	MB1	MB2	OutGuess	Steghide
F5 100%	0.32%	97.40%	1.04%	0.60%	0.00%	0.12%	0.52%
JP Hide&Seek 100%	0.00%	0.52%	98.32%	0.56%	0.00%	0.12%	0.48%
MB1 100%	0.08%	0.16%	0.72%	94.44%	0.32%	1.56%	2.72%
OutGuess 100%	0.00%	0.04%	0.52%	0.08%	0.04%	99.08%	0.24%
Steghide 100%	0.04%	0.04%	1.68%	2.96%	0.24%	1.52%	93.53%
F5 50%	0.96%	91.65%	0.92%	4.12%	0.28%	0.76%	1.32%
JP Hide&Seek 50%	0.32%	0.88%	90.46%	5.23%	0.04%	0.40%	2.68%
MB1 50%	0.80%	0.52%	0.16%	87.57%	2.20%	1.92%	6.83%
OutGuess 50%	0.08%	0.16%	0.20%	0.48%	0.08%	98.64%	0.36%
Steghide 50%	0.28%	0.44%	0.16%	3.99%	3.47%	2.84%	88.82%
MB2 30%	6.75%	0.40%	0.36%	1.76%	88.46%	0.56%	1.72%
F5 25%	10.99%	63.60%	1.04%	16.98%	2.56%	0.68%	4.16%
JP Hide&Seek 25%	6.15%	1.28%	74.96%	12.74%	0.92%	0.24%	3.71%
MB1 25%	11.02%	1.68%	0.56%	69.17%	6.63%	1.12%	9.82%
OutGuess 25%	1.32%	0.76%	0.24%	2.80%	3.23%	89.14%	2.52%
Steghide 25%	7.07%	1.36%	0.24%	12.42%	11.14%	1.96%	65.81%
Cover	96.45%	0.12%	0.20%	1.44%	0.40%	0.08%	1.32%

Table 1: Confusion matrix for the multi-classifier trained for quality factor 75 tested on single-compressed 75-quality JPEG images. The first column contains the embedding algorithm and the relative message length. The remaining columns show the results of classification.

embedding along a pseudo-random path will be indistinguishable in the feature space. This phenomenon might be responsible for "merging" of the MB1, MB2, and Steghide classes.

To present the results for all quality factors in a concise and compact manner, in Figure 2 we show the false positives and the detection accuracy for each steganographic algorithm separately. For each graph, on the x axis is the quality factor $q \in \mathcal{Q}_{18}$ and each curve corresponds to one relative message length. The detection accuracy is the percentage of stego images embedded with a given algorithm that are correctly classified as embedded by that algorithm. The false positive rate is the percentage of cover images classified as stego and is also shown in each graph (it is the same in each graph).

The false positive rate and detection accuracy for fully embedded images vary only little across the whole range of quality factors. For less than fully embedded images, the classification accuracy decreases with increasing quality factor. The situation becomes progressively worse with shorter relative message lengths. This phenomenon can be attributed to the fact that with higher quality factor the quantization steps become smaller and thus the embedding changes are more subtle and do not impact the features as much.

Next, we examined the images that were misclassified. In particular, we inspected all misclassified cover images and all stego images containing a message larger than 50% of the image capacity. We noticed that some of these images were very noisy (images taken at night using a 30 second exposure), while others did not give us any visual clues as to why they were misclassified. We note, though, that the embedding capacity of these images was usually below the average embedding capacity of images of the same size.

As the calibration used in calculating the DCT features subjects an image to compression twice, the calibrated image has a lower noise content than the original JPEG image. Thus, we hypothesize that very noisy images are more likely to be misclassified. To test this hypothesis, we blurred the noisy cover images that were classified as stego using a blurring filter with Gaussian kernel with diameter 1 and reclassified them. After this slight blurring, all of them were properly classified as cover images.

Most of the misclassified images from the remaining cameras were "flat" images, such as blue sky shots or completely dark images taken with a covered lens. Flat images do not provide sufficient statistics for steganalysis. Because these images have a very low capacity (in tens of bytes) for most stego schemes, they

are not suitable for steganography anyway.

Since we only trained the classifiers on a selected subset of 18 quality factors, a natural question to ask is if this set is "dense enough" to allow reliable detection for JPEG images with *all* quality factors in the range 63–96. To address this issue, we compared the performance of several pairs of classifiers trained for two different but close quality factors (e.g., q and $q+1$) on images with a single quality factor q. For example, we used one classifier trained for quality factor 66 and the other trained for 67 and compared their performance on images with quality factor 67.

Generally, the increase in false positives between both classifiers was about 0.3%. The exception was the classifier trained for the quality factor 77, where the false positive rate was by 1.5% higher on cover JPEG images with quality factor 78 in comparison with the classifier trained for quality factor 78. We point out that the quantization tables for these two quality factors differ in 3 out of 5 lowest frequency AC-coefficients. This indicates that for best results, a dedicated multi-classifier needs to be built for each quality factor. This is especially true for those quality factors around which the quantization matrices go through rapid changes.

Finally, an interesting and important question is what does the classifier do when presented with a stego algorithm that it was not trained on. Referring to our previous work [13, Table 5 in Section 4.4], we trained the same multi-classifier for single-compressed images embedded with only F5, OutGuess, MB1, and MB2, and then presented the classifier with images embedded with JP Hide&Seek. It correctly recognized the images as stego images (only 1.5% were classified as cover images) and assigned most of the images (62.5%) to F5 and 29.6% to MB1. In general, it is difficult to predict the result of the classification because it depends on how the SVM partitions the feature space.

5 Multi-Classifier for Double-Compressed Images

In this section, we construct a steganalyzer that can classify double-compressed JPEG images into three classes—F5, OutGuess, and cover, because F5 and OutGuess are the only stego programs in our set capable of producing double-compressed images. We constrained ourselves to just one secondary quality factor of 75 (the default factor for OutGuess). The classifier has an additional module that first estimates the primary quality factor from a given stego image. This estimated quality factor is then appended as an additional feature to the feature vector. We decided to create one large classifier for all primary quality factors instead of a set of specialized classifiers for each combination of primary / secondary quality factor, as we did in the single-compression classifier case. We opted for this solution, because one big multi-classifier can better deal with inaccuracies in the detection of the primary quality factor. The results reported here are the first attempt to address the issue of double compression, which has been largely avoided in the research literature due to its difficulty.

5.1 DCT Features for Double Compressed Images

As explained in [10], the process of calibration must be modified for images that went through multiple JPEG compression. In this section, we explain the modified calibration process.

Double compression occurs when a JPEG image, originally compressed with a primary quantization matrix Q^{pri}, is decompressed and compressed again with a different secondary quantization matrix Q^{sec}. For example, both F5 and OutGuess always decompress the cover JPEG image to the spatial domain and then embed the secret message while recompressing the image with a user-specified quality factor. If the second factor is different than the first one, the stego image experiences what we call double JPEG compression.

The purpose of calibration when calculating the DCT features is the estimation of the cover image. When calibrating a double-compressed image, the calibration *must* mimic what happens during embedding. In other words, the decompressed stego image after cropping should be first compressed with the primary (cover) quantization matrix Q^{pri}, decompressed, and finally compressed again with the secondary quantization matrix Q^{sec}. Because the primary quantization matrix is not stored in the stego JPEG image, it has to be estimated. Without incorporating this step, the results of steganalysis that uses DCT features might be completely misleading [11].

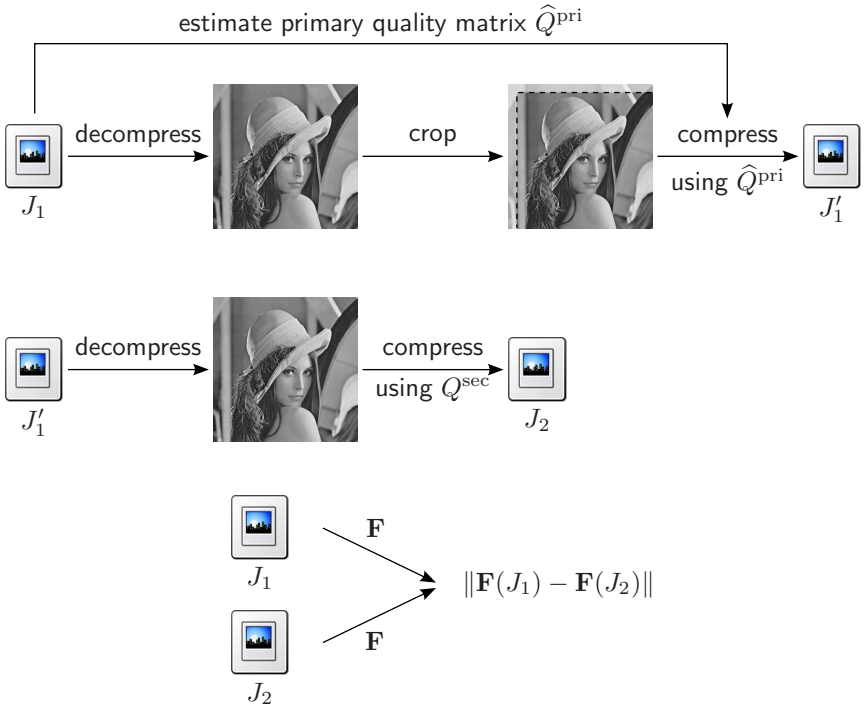


Figure 3: Calibrated features for double-compressed JPEG images.

In our work, we use the algorithm [12] for estimation of the primary quantization matrix. It employs a set of neural networks that estimate from the histogram of individual DCT modes the quantization steps Q_{ij}^{pri} for the 5 lowest frequency AC coefficients $(i,j) \in \mathcal{L} = \{(2,1),(1,2),(3,1),(2,2),(1,3)\}$. Constraining to the lowest frequency steps is necessary because the estimates of the higher-frequency quantization steps become progressively less reliable due to insufficient statistics for these coefficients. From the 5 lowest-frequency quantization steps, we determine the whole primary quantization matrix Q^{pri} as the closest standard quantization table using the following empirically constructed algorithm.

1. Apply the estimator [12] to the stego image and find the estimates $\hat{Q}_{ij}^{pri}, (i,j) \in \mathcal{L}$.
2. Find all standard quantization tables Q for which $Q_{ij} = \hat{Q}_{ij}^{pri}$ for at least one $(i,j) \in \mathcal{L}$.
3. Assign a matching score to all quantization tables Q found in Step 2. Each quantization table receives two points for each quantization step $(i,j) \in \mathcal{L}$ for which $Q_{ij} = \hat{Q}_{ij}^{pri}$ and one point for the quantization step that is a multiple of 2 or $\frac{1}{2}$ of the detected step.
4. The quantization table with the highest score is returned as the estimated primary quantization table.

Note, that for certain combinations of the primary and secondary quantization steps it is in principle very hard to determine the primary step (e.g., deciding whether $\hat{Q}_{ij}^{pri} = 1$ or $\hat{Q}_{ij}^{pri} = Q_{ij}$). In such cases, the estimator returns $\hat{Q}_{ij}^{pri} = Q_{ij}$ and the image is detected as single compressed. Fortunately, in these cases, the impact of incorrect estimation of the primary quantization table is not significant for steganalysis because the double compressed image does not exhibit strong traces of double compression anyway. The modified calibration process that incorporates the estimated primary quantization matrix is described in Figure 3.

Classifier	Cover 0%	F5 100% / 50% / 25%	OutGuess 100% / 50% / 25%
Cover vs. F5	3400	1133 / 1133 / 1133	—
Cover vs OutGuess	3400	—	1133 / 1133 / 1133
F5 vs. OutGuess	—	1133 / 1133 / 1133	1133 / 1133 / 1133

Table 2: Structure of the set of training images for one primary quality factor for training the three binary SVMs for double-compression multi-classifier. We have created one multi-classifier for all primary quality factors in $\mathcal{Q}_{12}$, thus the whole training set contained images double-compressed with primary quality factors in $\mathcal{Q}_{12}$ and the secondary quality factor 75.

5.2 Training

The training database of stego images was constrained to images embedded with three relative message lengths using F5 and OutGuess. The secondary (stego) quality factor was fixed to 75, since this is the default quality factor for OutGuess. To decrease the computational and storage requirements, we used a smaller image database and trained the classifier again on a preselected subset of quality factors. The training set was prepared from 3400 raw images and the testing set from additional 1050 images. The training set contained JPEG images with primary quality factors in the range from 63–100. The primary quality factors used for training were selected so that for every quality factor $q \in \{63, \ldots, 100\}$, there is a quality factor q', such that for the corresponding quantization matrices $\sum_{(i,j)\in\mathcal{L}} |Q_{ij} - Q'_{ij}| \leq 2$. This leads to the following set of 12 primary quality factors $\mathcal{Q}_{12} = \{63, 66, 69, 73, 77, 78, 82, 85, 88, 90, 94, 98\}$. Each raw image was JPEG compressed with the appropriate primary quality factor before embedding and then a random bit-stream of relative length 100%, 50%, and 25% of the image embedding capacity was embedded using F5 and OutGuess with the stego quality factor set to 75. The cover images were also JPEG compressed with the secondary quality factor 75.

To summarize, for training each raw image was processed in 7 different ways (OutGuess 100%, OutGuess 50%, OutGuess 25%, F5 100%, F5 50%, F5 25%, and cover JPEG) and for 12 different primary quality factors selected from $\mathcal{Q}_{12}$. The total number of images used for training was $12 \times 7 \times 3400 = 285,600$. Table 2 shows the distribution of images in the training set for one primary quality factor and all three binary SVMs (cover vs. F5, cover vs. OutGuess, and F5 vs. OutGuess). The training set for each machine consisted of approximately $12 \times 3400 = 40,800$ cover and the same amount of stego images. The stego images were randomly chosen to uniformly cover all message lengths for each algorithm.

The parameters γ and C were determined by a grid-search on the multiplicative grid

$$(\gamma, C) \in \left\{(2^i, 2^j) | i \in \{-5, \ldots, 3\},\ j \in \{0, \ldots, 10\}\right\}$$

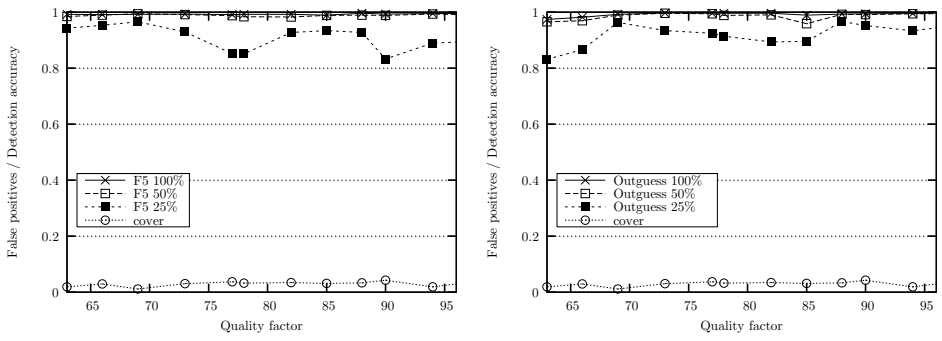
combined with 5-fold cross-validation, as described in Section 3. In particular, we used $\gamma = 4$, $C = 128$ for the cover vs. F5 SVM, $\gamma = 4$, $C = 64$ for the cover vs. OutGuess SVM, and $\gamma = 4$, $C = 32$ for the F5 vs. OutGuess machine. For all three classifiers, the best validation error on the grid was achieved for a fairly narrow kernel, which suggests that the separation boundaries between different classes are rather thin.

5.3 Testing and discussion

Table 3 shows the confusion matrix calculated for images from the testing set that was prepared in exactly the same manner as the training set from additional 1050 images never seen by the classifier (i.e., the number of test images was $12 \times 7 \times 1050 = 88,200$). In Figure 4, we depict the results for each quality factor separately and for each steganographic algorithm. The figure shows the percentage of correctly classified stego images and the percentage of cover images classified as stego (false positives) as a function of the quality factor. We see that when the message is longer than 50% of the embedding capacity, the classification accuracy is very good. The classification accuracy for short message lengths (25% of embedding capacity) is above 90%. The false alarm rate is about 3%.

	Classified as		
Algorithm	Cover	F5	OutGuess
F5 100%	0.56%	99.26%	0.18%
OutGuess 100%	0.52%	0.11%	99.36%
F5 50%	0.87%	98.87%	0.25%
OutGuess 50%	0.80%	0.28%	98.93%
F5 25%	8.30%	90.86%	0.84%
OutGuess 25%	5.23%	1.30%	93.47%
Cover	96.99%	1.99%	1.02%

Table 3: Confusion matrix showing the classification accuracy of the multi-classifier for double-compressed images (results are merged over all primary quality factors). The primary quality factors of all test images are the same as those used for training. For each primary quality factor, algorithm, and message length, there are approximately 1050 images.



(a) F5 (b) OutGuess

Figure 4: Percentage of correctly classified images embedded with F5 (left) and OutGuess (right) and false positives for all 12 multi-classifiers. Each curve corresponds to one relative payload.

Algorithm	Cover	F5	OutGuess
F5 100%	8.1%	91.24%	0.66%
OutGuess 100%	0.76%	0.21%	99.01%
F5 50%	9.21%	90.00%	0.79%
OutGuess 50%	1.78%	0.52%	97.70%
F5 25%	14.92%	82.89%	1.19
OutGuess 25%	13.96%	1.99%	84.05%
Cover	94.34%	3.33%	2.33%

Table 4: Classification accuracy on a test set of double-compressed images with 20 different primary quality factors from $\mathcal{Q}_{20}$, 8 of which were not used for training (compare to Table 3).

While the false positive rate stays approximately the same for all primary quality factors, the missed detection rate for images with short messages varies much more. For example, for F5 stego images with message length 25%, the highest missed detection rate is 15.36% for the primary quality factor 90, while for the primary quality factor 69 the rate is only 2.77%. A similar pattern was observed for OutGuess. In general, the missed detection rate is better for images with lower primary quality factors.

To obtain further insight into this phenomenon, we examined the accuracy of the estimator of the primary quality factor. As one can expect, the embedding changes themselves worsen the estimates of the primary quantization table. This effect is more pronounced for images with primary quality factor above 88, which are often detected as single-compressed images. Therefore, further improvement is expected with more accurate estimators of the primary quality factor. In particular, the estimator should be trained not only on double-compressed cover images, but also on examples of double-compressed stego images.

Similar to Section 4, we next decided to test the performance of the classifier for double-compressed images on images with primary quality factors that were not among those that the classifier was trained on. We added to the testing database double-compressed and embedded images with 8 more quality factors, obtaining the following expanded set of $8 + 12 = 20$ primary quality factors $\mathcal{Q}_{20} = \{63, 67, 69, 70, 71, 73, 75, 77, 78, 80, 82, 83, 85, 87, 88, 90, 92, 94, 96, 98\}$. Table 4 shows the confusion table. Although the false alarm percentage increased by about 1% for each class, the misclassification among different classes increased by almost 10%. This indicates that reliable classification for double-compressed images requires training on a denser set of quality factors.

6 Conclusions

In this paper, we construct a classifier bank capable of assigning JPEG images to 6 known JPEG steganographic algorithms. We also address the difficult issue of double-compressed images by building a separate classifier for images that were recompressed during embedding with a different quantization matrix. Because the classifiers described in this paper can identify the embedding algorithm, they form an important first step in forensic steganalysis whose goal is to not only detect the secret message presence but eventually extract the message itself. As such, this tool is expected to be useful for law enforcement and forensic examiners.

The classifiers are built from 23 features calculated from the luminance component of DCT coefficients using the process of calibration. The first classifier bank is designed for single-compressed images. For each quality factor, a set of $\binom{n}{2}$ binary support vector machines is constructed that can distinguish between pairs from $n = 7$ classes (6 stego programs + cover images). Each classifier is built from cover and the same number of stego images embedded with messages of relative length 25%, 50%, and 100% of the embedding capacity. The max-wins multi-classifier is then used to evaluate the individual decisions of 21 binary classifiers to assign an image to a specific class. The performance is evaluated using confusion matrices and graphs that show the classification accuracy for each algorithm as a function of the quality factor for separate message lengths.

The second classifier is designed to assign double-compressed images to three classes—images embedded

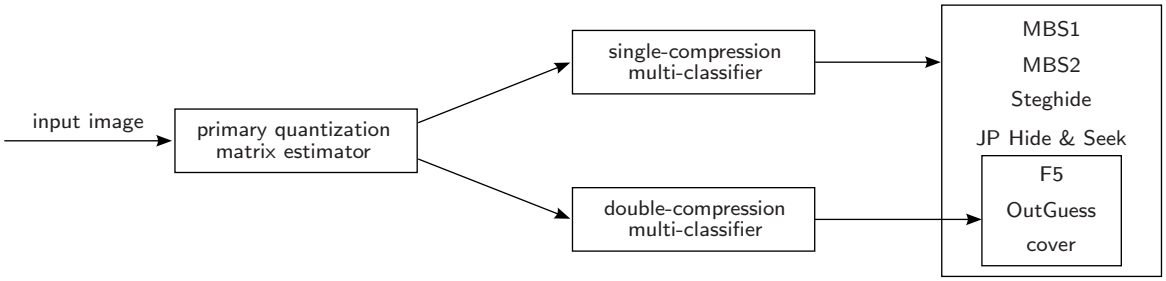


Figure 5: The proposed steganalyzer structure. Abbreviations are explained in the text.

with F5, OutGuess, and non-embedded cover images. We classify into these classes because F5 and OutGuess are the only stego programs that can produce double compressed images during embedding (when the cover image quality factor is not the same as the stego image quality factor). Double compression must be corrected for in the calibration process. This requires estimation of the primary (cover) quality factor. We trained a classifier for test images of 12 different quality factors. This classifier gave satisfactory performance on a testing set of double-compressed stego images with the same 12 quality factors produced by F5 and OutGuess. It also performed reasonably well when tested on JPEG images with quality factors that were not included in the training set. More accurate results are expected after expanding the training set of quality factors.

As already stated in the introduction, the goal of this paper is to build a solid foundation for constructing a multi-class steganalyzer capable of assigning images to known steganographic programs and able to handle images of arbitrary quality factor and both single and double-compressed images. Our plan for the future is to further refine both classifier banks constructed in this paper and merge them into one complex steganalyzer. This steganalyzer will be preceded by an SVM estimator of the primary (cover image) quantization matrix. This estimator first makes a decision if the image under investigation is a single-compressed image or a double-compressed image and then sends it, together with an estimate of the primary quantization matrix, to the appropriate classifier (see Figure 5).

The estimator of double-compression will have a significant impact on the overall accuracy of the classification because once an image is deemed double-compressed, it can only be a cover image or embedded using F5 or OutGuess. Thus, this estimator should be tuned to have a very low false positive rate (incorrectly detecting double-compression when the image is single-compressed). As pointed out in Section 5, we currently use a neural-network based estimator from [12] trained on double-compressed images. However, the steganalyzer might be presented with images that were jointly double-compressed *and* embedded. The act of embedding might change the distribution of DCT coefficients and thus might confuse the double-compression estimator. Obviously, it is necessary to train the estimator on both double-compressed images and double-compressed and embedded images. Since this topic deserves a paper of its own, we plan to first carefully design the estimator and then incorporate it into steganalysis as discussed above.

7 Acknowledgements

The work on this paper was supported by Air Force Research Laboratory, Air Force Material Command, USAF, under the research grant number FA8750–04–1–0112. 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] JP Hide&Seek. http://linux01.gwdg.de/~alatham/stego.html.

[2] R.J. Anderson and F.A.P. Petitcolas. On the limits of steganography. *IEEE Journal of Selected Areas in Communications, Special Issue on Copyright and Privacy Protection*, 16(4):474–481, 1998.

[3] I. Avcibas, M. Kharrazi, N. Memon, and B. Sankur. Image steganalysis with binary similarity measures. *EURASIP Journal on Applied Signal Processing*, 17:2749–2757, 2005.

[4] I. Avcibas, N. Memon, and B. Sankur. Steganalysis using image quality metrics. In E. Delp and P. W. Wong, editors, *Proceeedings of SPIE Electronic Imaging, Security and Watermarking of Multimedia Contents III*, volume 4314, pages 523–531, 2001.

[5] I. Avcibas, B. Sankur, and N. Memon. Image steganalysis with binary similarity measures. In *Proceedings of International Conference on Image Processing*, volume 3, pages 645–648, 2002.

[6] C. Cachin. An information-theoretic model for steganography. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop*, volume 1525 of *Lecture Notes in Computer Science*, pages 306–318, 1998.

[7] R. Chandramouli, M. Kharrazi, and N. Memon. Image steganography and steganalysis: Concepts and practice. In T. Kalker, I. Cox, and Yong Man Ro, editors, *Digital Watermarking, 2nd International Workshop*, volume 2939 of *Lecture Notes in Computer Science*, pages 35–49, 2004.

[8] H. Farid and L. Siwei. Detecting hidden messages using higher-order statistics and support vector machines. In F.A.P. Petitcolas, editor, *Information Hiding, 5th International Workshop*, volume 2578 of *Lecture Notes in Computer Science*, pages 340–354, 2002.

[9] J. Fridrich. Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. In J. Fridrich, editor, *Information Hiding, 6th International Workshop*, volume 3200 of *Lecture Notes in Computer Science*, pages 67–81, 2005.

[10] J. Fridrich, M. Goljan, and D. Hogea. Steganalysis of JPEG images: Breaking the F5 algorithm. In F. A. P. Petitcolas, editor, *Information Hiding, 5th International Workshop*, volume 2578 of *Lecture Notes in Computer Science*, 2002.

[11] J. Fridrich, M. Goljan, D. Hogea, and D. Soukal. Quantitative steganalysis: Estimating secret message length. *ACM Multimedia Systems Journal. Special issue on Multimedia Security*, 9(3):288–302, 2003.

[12] J. Fridrich and J. Lukas. Estimation of primary quantization matrix in double compressed JPEG images. In *Proceedings of International Conference on Image Processing*, volume 3, 2002.

[13] J. Fridrich and T. Pevný. Towards multi–class blind steganalyzer for JPEG images. In M. Barni, I. Cox, T. Kalker, and H. J. Kim, editors, *4th International Data Hiding Workshop*, volume 3710 of *Lecture Notes in Computer Science*, pages 39–53, 2005.

[14] M. Goljan, J. Fridrich, and T. Holotyak. New blind steganalysis and its implications. In E. Delp and P. W. Wong, editors, *Proceedings of SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, pages 1–15, 2006.

[15] J.J. Harmsen and W.A. Pearlman. Steganalysis of additive noise modelable information hiding. In E. Delp and P. W. Wong, editors, *Proceedings of SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents V*, volume 5020, pages 131–142, 2003.

[16] S. Hetzl and P. Mutzel. A graph-theoretic approach to steganography. In J. Dittmann et al., editor, *Communications and Multimedia Security. 9th IFIP TC-6 TC-11 International Conference*, volume 3677 of *Lecture Notes in Computer Science*, pages 119–128, 2005.

[17] C. Hsu, C. Chang, and C. Lin. *A practical guide to support vector classification.* Department of Computer Science and Information Engineering, National Taiwan University, Taiwan. http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[18] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001. http://citeseer.ist.psu.edu/hsu01comparison.html.

[19] S. Katzenbeisser and F.A.P. Petitcolas. Security in steganographic systems. In E. Delp and P. W. Wong, editors, *Proceedings of SPIE Electronic Imaging, Security and Watermarking of Multimedia Contents IV*, volume 4675, pages 50–56, 2002.

[20] M. Kharrazi, H. T. Sencar, and N. Memon. Benchmarking steganographic and steganalytic techniques. In E. Delp and P. W. Wong, editors, *Proceedings of SPIE Electronic Imaging, Security, Steganography and Watermarking of Multimedia Contents VII*, volume 5681, pages 252–263, 2005.

[21] S. Lyu and H. Farid. Steganalysis using color wavelet statistics and one-class support vector machines. In E. Delp and P. W. Wong, editors, *Proceedings of SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 35–45, 2004.

[22] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In S.A. Solla, T.K. Leen, and K.-R. Mueller, editors, *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.

[23] N. Provos. Defending against statistical steganalysis. In *10th USENIX Security Symposium*, 2001.

[24] P. Sallee. Model based steganography. In Kalker, I.J. Cox, and Yong Man Ro, editors, *International Workshop on Digital Watermarking*, volume 2939 of *Lecture Notes in Computer Science*, pages 154–167, 2004.

[25] P. Sallee. *Statistical Methods for Image and Signal Processing.* PhD thesis, University of California Davis, 2004.

[26] Phil Sallee. Model-based methods for steganography and steganalysis. *Int. J. Image Graphics*, 5(1):167–190, 2005.

[27] A. Westfeld. High capacity despite better steganalysis (F5 a steganographic algorithm). In I.S. Moskowitz, editor, *Information Hiding, 4th International Workshop*, volume 2137 of *Lecture Notes in Computer Science*, pages 289–302, 2001.

[28] G. Xuan, Y.Q. Shi, J. Gao, D. Zou, C. Yang, Z. Zhang, P. Chai, C. Chen, and W. Chen. Steganalysis based on multiple features formed by statistical moments of wavelet characteristic function. In M. Barni, editor, *Information Hiding. 7th International Workshop*, volume 3727 of *Lecture Notes in Computer Science*, pages 262–277, 2005.

[29] J. Zöllner, H. Federrath, H. Klimant, A. Pfitzmann, R. Piotraschke, A. Westfeld, G. Wicke, and G. Wolf. Modeling the security of steganographic systems. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop*, volume 1525 of *Lecture Notes in Computer Science*, pages 344–354, 1998.