Multi-class Blind Steganalysis for JPEG Images

Tomáš Pevný[a], Jessica Fridrich[b]

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

ABSTRACT

In this paper, we construct blind steganalyzers for JPEG images capable of assigning stego images to known steganographic programs. Each JPEG image is characterized using 23 calibrated features calculated from the luminance component of the JPEG file. Most of these features are calculated directly from the quantized DCT coefficients as their first order and higher-order statistics. The features for cover images and stego images embedded with three different relative message lengths are then used for supervised training. We use a support vector machine (SVM) with Gaussian kernel to construct a set of binary classifiers. The binary classifiers are then joined into a multi-class SVM using the Max-Win algorithm. We report results for six popular JPEG steganographic schemes (F5, OutGuess, Model based steganography, Model based steganography with deblocking, JP Hide&Seek, and Steghide). Although the main bulk of results is for single compressed stego images, we also report some preliminary results for double-compressed images created using F5 and OutGuess. This paper demonstrates that it is possible to reliably classify stego images to their embedding techniques. Moreover, this approach shows promising results for tackling the difficult case of double compressed images.

1. INTRODUCTION

Steganography is the art of undetectable communication in which messages are embedded in innocuous looking objects, such as digital images. In the process of embedding, the original (cover) object is slightly modified to embed the data. The modified cover object is called the stego object. The embedding process usually depends on a secret stego key shared between both communicating parties. The main requirement of steganographic systems is statistical undetectability of the hidden data given the 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, 18, 25]

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 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,[10] while blind approaches are not tailored to any specific embedding paradigm.[4, 5, 8, 9] Blind approaches can be thought of as practical embodiments of Cachin's definition of steganographic security. It is assumed that "natural images" can be characterized using a small set of numerical features. The distribution of feature vectors 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.

The idea to use a trained classifier to detect data hiding was first introduced in a paper by Avcibas et al.,[4] where image quality metrics were proposed as features and the method was tested on several robust watermarking algorithms as well as LSB embedding. Avcibas et al.[3, 5] later proposed a different set of features based on binary similarity measures between the LSB plane and the second LSB plane capitalizing on the fact that most steganographic schemes use the LSB of image elements as the information-carrying entity. Farid[8, 20] constructed the features from higher-order moments of distribution of wavelet coefficients and their linear prediction errors from several high-frequency sub-bands. Other authors have investigated the problem of blind steganalysis using trained classifiers.[14, 24]

An important advantage of blind methods is that they are potentially capable of detecting previously unseen steganographic methods and they can classify the embedding algorithm based on the location of the feature

vector in the feature space. This classification is an important first step toward extracting the secret message because knowing the program used to embed the secret data, the forensic examiner can continue with brute-force searches for the stego key.[11]

In this paper, we focus on steganalysis of JPEG images. Blind steganalysis of JPEG images in the most general setting (arbitrary quality factors and double compressed images) requires very extensive computational resources and thus substantial time, as well. In this paper, we outline a strategy for constructing such a steganalyzer and some preliminary results that show promise that a reliable multi-class steganalyzer for JPEG images can be built. In particular, we construct a blind classifier that can reliably assign stego images to 6 known JPEG steganographic techniques (F5, OutGuess, Steghide, JP Hide&Seek, Model based steganography with and without deblocking) for 18 quality factors. We also address the difficult issue of double compressed stego images,[10] which was so far largely avoided in previous works on steganalysis of JPEG images. This paper is an extension of our previous work on this subject.[9, 13]

In the next section, we describe the process of calibration used to construct DCT-based features for all our classifiers. In Section 3, we give the implementation details of the SVMs used in this paper. We also describe the database of test images and discuss training and testing procedures. In Section 4, we construct a seven-class SVM for detecting steganographic algorithms for single-compressed JPEG images embedded with six popular JPEG steganographic algorithms and various quality factors. Section 5 is devoted to steganalysis of double-compressed images. We first explain how the process of calibration must be modified to account for double compression by estimating the primary quantization matrix from the double-compressed stego image. Then, we use this estimator to construct a three-class SVM capable of distinguishing double-compressed cover images from double-compressed (and embedded) images using F5 and OutGuess (these are the only programs in our tests that can produce double compressed stego images). The experimental results from both classifiers are interpreted in view of our ultimate goal to construct a blind JPEG steganalyzer capable of handling JPEG images of arbitrary quality factors and both single and double compressed images. The paper is concluded in Section 6.

2. 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, 19] Both studies report the superiority of JPEG classifiers that use DCT features. Here, we only briefly describe the features (see Figure 1), referring the reader to[9] for more details.

Let us assume for now that the stego image has *not* been double compressed. A vector functional $\mathbf{F}$ is applied to the stego JPEG image J_1. For example, this functional could be the histogram of all DCT coefficients. The stego image J_1 is decompressed to the spatial domain, cropped by a few pixels in each direction, and recompressed with the same quantization table as 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 $\mathbf{F}(J_1) - \mathbf{F}(J_2)$, if $\mathbf{F}$ is a scalar, or an L_1 norm for vector or matrix functionals

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

The cropped and recompressed image is an approximation to the cover JPEG image. Thus, the net effect of calibration is to decrease image-to-image variations and increase the features' sensitivity to embedding.

We now define all 23 functionals used for steganalysis.

Let the luminance of a stego JPEG file be represented 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

$$\mathbf{h}^{ij} = (h_L^{ij}, \ldots, h_R^{ij}), \tag{3}$$

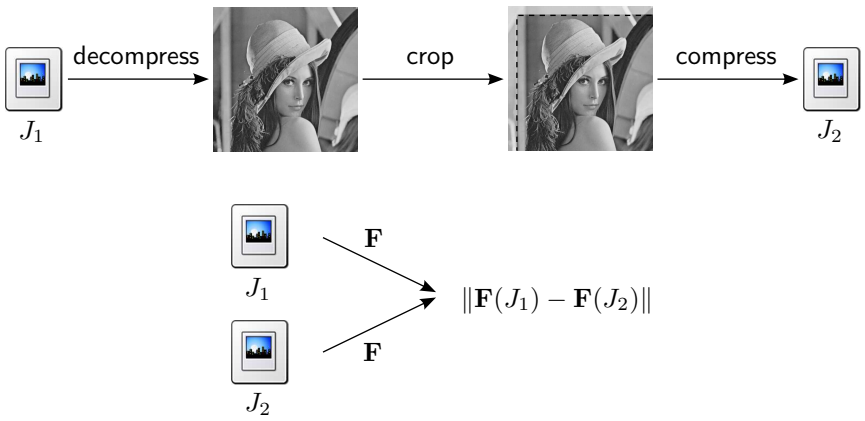

Figure 1. Calibrated features f are obtained from functionals $\mathbf{F}$.

of coefficients of 5 individual DCT modes $(i,j) \in \{(1,2),(2,1),(3,1),(2,2),(1,3)\} \triangleq \mathcal{L}$. The next 11 functionals are dual histograms represented with 8×8 matrices $\mathbf{g}_{ij}^d$, $i,j = 1,\dots,8$, $d = -5,\dots,5$

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

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

The next 6 functionals capture inter-block dependency among DCT coefficients. The first 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 $\mathbf{I}_c$ denote the vectors of block indices $1,\dots,n_B$ while scanning the image by rows and by columns, respectively.

Two next two blockiness functionals are scalars calculated from the decompressed JPEG image representing an integral measure of inter-block dependency over all DCT modes 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 grayscale values of the decompressed JPEG image.

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

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

where

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

3. SVM CLASSIFIER

SVMs are naturally able to classify only to 2 classes. There exist various extensions to enable SVMs to handle more then two classes. They 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,[17] where the authors conclude that methods based on binary classifiers are better for practical applications. In this paper, we used the "max-wins" method. 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 target class. If there are two or more classes with the same number of votes, one of the classes is randomly chosen.

In our work, we used soft-margin SVMs[8, 19] with Gaussian kernel $\exp(-\gamma\|x-y\|^2)$. Soft-margin SVMs could be trained on non-separable data (unlike hard-margin SVMs) 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 parameter C forces the number of incorrectly classified images during training to be minimized. In most applications, C is the same for both cover and stego images. This implicitly implies that incorrectly classified images from both classes have the same cost. However in steganography, false positives (cover image classified as stego) have associated a much higher cost than missed detection (stego images classified as cover). To train SVMs with uneven cost of incorrect classification, we penalize incorrectly classified images using two different penalty parameters C_{FP} and C_{FN}, where FP and FN stand for false positives and false negatives, respectively.

Prior to training an SVM, we have to determine its parameters. For binary SVMs that classify between two classes of stego images, e.g., between F5 and OutGuess, we consider the cost associated with both classes as equal. Thus, we need to determine two parameters—(γ, C). For SVMs that classify between cover and stego images, we have to determine three parameters (γ, C_{FP}, C_{FN}). Following the advice in,[16] we determined 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 was repeated n_{cv} times for each subset and the values calculated from each subset were averaged. These averages were used as estimates of the performance on unknown data. For SVMs classifying only into stego classes, the final values of the parameters were determined by the least estimated validation error on the multiplicative grid. For SVMs classifying between the cover and stego classes, the parameters were determined by the least estimated missed detection rate for the estimated false positive rate below 1%. If none of the estimated false positive rates was below 1%, than the smallest estimated false positive rate determined the parameter values. The multiplicative grids are described in the corresponding sections.

After determining the parameters, we used the whole training set to train the SVM. Before training, we scaled all elements of the feature vector to the interval $[-1, +1]$. The scaling coefficients were always derived from the training set. For the cross-validation, the scaling coefficients were calculated on the $n_{cv} - 1$ subsets.

3.1. Image Database

For the first experiment for single-compressed images (Section 4), we used approximately 6000 images of natural scenes taken under varying conditions (close-ups, landscapes, outside and inside shots with and without flash, and pictures taken 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 in raw formats.

Before performing the experiments, all images were divided into two disjoint groups. The first group contained 3500 images from the first 7 cameras in the list (including the resized images) and was used for training. The second group, containing about 2500 images was used for testing. Thus, no image or its different forms were used simultaneously for testing and training. This strict division of images also enables us to estimate the performance on never seen images of a completely different origin.

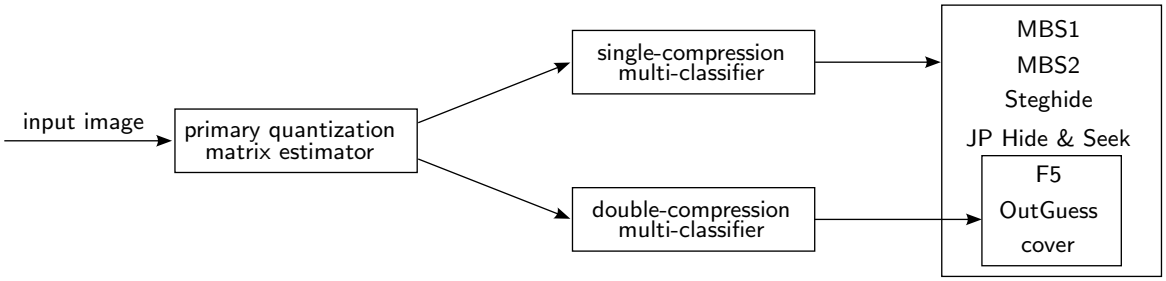

Figure 2. Classifier structure. Abbreviations are explained in the text.

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. SINGLE-COMPRESSION MULTI-CLASSIFIER

As already stated in the introduction, the goal of this paper is to pave our way toward 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 is to construct two classifiers—one that takes single-compressed images as its input and assigns them to their classes and the second one that only accepts images deemed to be double-compressed and assigns them to either a class of double compressed cover images or to stego programs that can produce such images—F5 and OutGuess. Both classifiers are preceded by an estimator of the primary (cover image) quantization matrix, which is an SVM that can tell if the image was double compressed and also estimate the primary quantization matrix (see Figure 2). This estimator then makes a decision if the image under investigation is a single-compressed image or double-compressed image and then sends it, together with an estimate of the primary quantization matrix, to the appropriate classifier.

In this section, we build the first classifier that only deals with single-compressed images. Instead of adding the JPEG quality factor as an additional feature, we opted for training a separate multi-classifier for each quality factor. The reason for this is that this set of separate classifiers 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 preselected the following 18 quality factors $\mathcal{Q}_{18} = \{63, 67, 69, 71, 73, 75, 77, 78, 80, 82, 83, 85, 88, 90, 92, 94, 96\}$.

We embedded a random binary stream of different lengths using six different algorithms—F5,[23] Model based steganography without (MB1) and with deblocking (MB2),[22] JP Hide&Seek,[1] OutGuess ver 0.2,[21] and Steghide.[15] For F5, MB1, JP Hide&Seek, OutGuess, and Steghide we embedded messages of three different length—100%, 50%, 25% of the embedding capacity for a given image. For JP Hide&Seek, in compliance with the directions provided by its author, 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 lossless formats (F5 accepts "png" and OutGuess accepts "ppm"), in which case the stego image is not double-compressed. We also note that we had to slightly modify OutGuess to allow saving the stego image at quality factors below 75.

The max-wins multi-classifier employs $\binom{n}{2}$ binary classifiers for every pair out of $n = 7$ classes, which in this case results in 21 SVMs. For each machine, 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 contained an equal number of examples of all three message lengths corresponding to 100%, 50%, and 25% of the algorithm embedding capacity. The total number of images used for training was approximately $18 \times 17 \times 3500 = 1,071,000$ (there are 3 message lengths for 5 stego programs, one for MB2, and cover).

	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 of the multi-classifier for quality factor 75 calculated on single-compressed JPEG images from the testing set with quality factor 75. The left most column contains the embedding algorithm and the embedded message length. The remaining columns show the results of classification. The quality factors of all test images are the same as the quality factors used for training.

The parameters (γ, C) for SVMs classifying only into stego classes 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 into the cover class were determined on the multiplicative 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. We do not include in this paper the parameters for all SVMs, because there are too many of them ($18 \times (6 \times 3 + 15 \times 2)$). However, the parameter values are available upon request by e-mail.

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 taking the training set, while the remaining 1500 were all taken by cameras not used to produce 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 an example of the performance of the multi-classifier for the quality factor 75. The multi-classifier reliably detects stego images for message lengths 50% or larger. For fully embedded images, the classification accuracy is 92–99% with false alarms of about 0.5%. JP Hide&Seek and OutGuess are consistently the easiest to detect across all three message lengths. On the other hand, MB2 and MB1 are the least detectable methods. At low embedding rates, the detection of F5 is also lower compared to other methods. This is likely due to matrix embedding, which decreases the number of embedding changes.

With decreasing message length, the results of the classification become progressively worse, which is to be expected. At this point, we point out that there is a fundamental limitation that cannot be overcome. Namely, it is not possible to distinguish between two algorithms that employ the same embedding mechanism just by inspecting the statistics of DCT coefficients. For example, two algorithms that use LSB embedding in the DCT domain along a pseudo-random path will appear indistinguishable to any classifier unless brute searches for stego

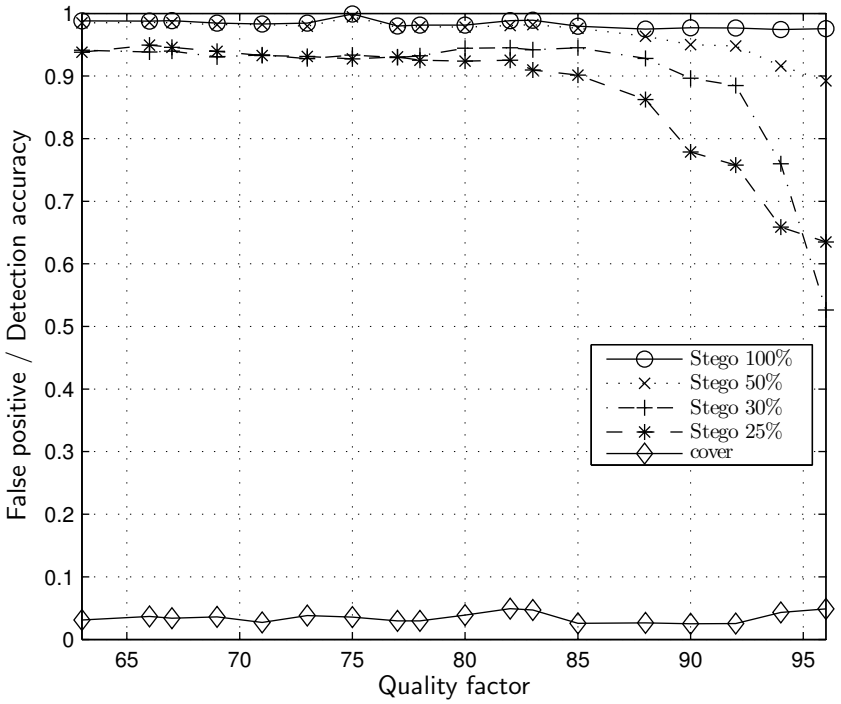

Figure 3. Average false positive rate $\overline{FA}$ (solid line with diamonds) together with average detection accuracy $\overline{D}(p)$ of 18 multi-classifiers for quality factors from set $\mathcal{Q}_{18}$. The quality factors of all test images are the same as the quality factors used for training.

key[11] are employed. This phenomenon might be responsible for "merging" of results among MB1, MB2, and Steghide. The correct algorithm assignment rate for these algorithms is lower than for the remaining algorithms.

Figure 3 shows the average false positive rate and detection accuracy for 18 multi-classifiers corresponding to 18 quality factors from $\mathcal{Q}_{18}$. The average rates were calculated in the following manner. Let $D(\text{Alg}, p)$ denote the number of all test stego images embedded with p% message with algorithm Alg$\in$\{F5, OutGuess, Steghide, JPHS, MB1, MB2\} that were correctly assigned to their embedding algorithm, and $N_{st}(\text{Alg}, p)$ is the number of stego test images for each algorithm Alg and message length p. For p=25%, 50%, 100%, the average detection rate shown in Figure 3 is obtained as

$$\overline{D}(p) = \frac{1}{5}\left(\frac{D(\text{F5},p)}{N_{st}(\text{F5},p)} + \frac{D(\text{OutGuess},p)}{N_{st}(\text{OutGuess},p)} + \frac{D(\text{Steghide},p)}{N_{st}(\text{Steghide},p)} + \frac{D(\text{JPHS},p)}{N_{st}(\text{JPHS},p)} + \frac{D(\text{MB1},p)}{N_{st}(\text{MB1},p)}\right). \tag{11}$$

For $p = 30\%$, the detection rate is calculated as $\frac{1}{N_{st}}D(\text{MB2}, 30)$, where now N_{st} denotes the number of stego images embedded using MB2.

The average false positive rate was obtained as

$$\overline{FA} = \frac{1}{6N_{cov}}\left(FA(\text{F5}) + FA(\text{OG}) + FA(\text{Steghide}) + FA(\text{JPHS}) + FA(\text{MB1}) + FA(\text{MB2})\right), \tag{12}$$

where $FA(\text{Alg})$ is the number of test cover images classified as embedded with algorithm Alg and N_{cov} is the total number of test cover images.

The average false positive rate $\overline{FA}$ and detection accuracy $\overline{D}(100)$ vary only little across the range of quality factors. For less than fully embedded images, the average detection accuracy decreases with increasing quality

factor. The situation becomes progressively worse with shorter relative message length. We attribute this phenomenon to the fact that with higher quality factor the quantization steps become smaller and thus the embedding changes are more subtle and become more difficult to detect.

Since we only trained the classifiers on a 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 two 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 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, whose false positive rate was 1.5% higher on cover JPEG images with quality factor 78 in comparison with the classifier trained for quality factor 78. We note 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 should be built for each quality factor.

5. MULTI-CLASSIFIER FOR DOUBLE-COMPRESSED IMAGES

5.1. Features for Double Compressed Images

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 than embed the secret message into the JPEG file 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—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.[10]

In our work, we use the algorithm[12] for estimation of the primary quantization matrix. This algorithm 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 $(i,j) \in \mathcal{L} = \{(2,1),(1,2),(3,1),(2,2),(1,3)\}$. This constraint to the lowest frequency steps is necessary because estimating the higher-frequency quantization steps becomes 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 estimation of double compression is described in Figure 4.

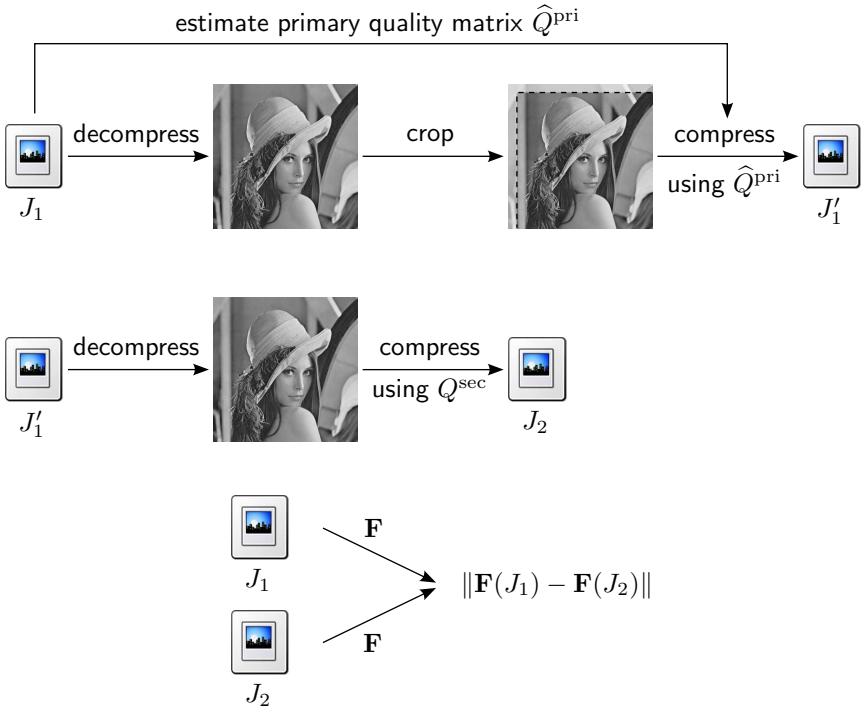

Figure 4. Calibrated features for double-compressed JPEG images.

5.2. Training

The database used to train the multi-classifier for double-compressed images was narrowed down to algorithms that are capable of producing such images—F5 and OutGuess. Thus, the multi-classifier consisted of $n = 3$ classes (F5, OutGuess, and cover). The secondary quality factor was fixed to 75, since this is the default quality factor for OutGuess. To decrease the computational and storage requirements, we also used a smaller image database. 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 to ensure 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 $\mathcal{Q}_{12} = \{63, 66, 69, 73, 77, 78, 82, 85, 88, 90, 94, 98\}$. This choice was forced upon us to decrease the tremendous computational and storage burden associated with creating the training database of images. Each raw image was converted to the JPEG image with the appropriate primary quality factor before embedding and then a random bit-stream of length 100%, 50%, and 25% of the capacity of the image for a given algorithm 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 from $\mathcal{Q}_{12}$. Thus, 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

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. Subset of the training set for one primary quality factor for the training of binary SVMs for double-compression multi-classifier. The whole training set contained images double-compressed with primary quality factors in $\mathcal{Q}_{12}$ and secondary quality factor 75.

	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. Classification accuracy of the multi-classifier for double-compressed images. All images are from the testing set only. The primary quality factors of all test images are the same as the primary quality factors used for training. For each primary quality factor, algorithm, and message length there are approximately 1050 images.

machine. For all three classifiers, the best validation error on the grid was achieved for a narrow kernel, which suggests that the separation boundaries between different classes are rather thin.

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 but from additional 1050 images never seen by the classifier (i.e., the number of test images was $12 \times 7 \times 1050 = 88,200$). We see that when the message is longer than 50% of the embedding capacity, the detection accuracy is very good. The classification accuracy for short message lengths (25% of embedding capacity) is above 90% with about 3% false alarms (cover images detected as stego).

We also inspected the distribution of errors as a function of the primary quality factor. We observed that while the false positive rate stays approximately the same for all primary quality factors, the missed detection rate for images with short messages varied 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 same images with primary quality factor 69, the rate is only 2.77%. Similar pattern was observed for OutGuess. Generally, the missed detection rate is better for images with lower primary quality factor.

To obtain further insight and explain this phenomenon, we examined the accuracy of the estimator of the primary quality factor. For primary quality factor below 85, the estimator gives correct results in more than 90% of cases. The accuracy of the estimation decreases with increasing primary quality factor. For example, for cover images with primary quality factor 94, almost all images are detected as single-compressed images. This is not surprising, because it is hard to correctly estimate the primary quality factor when the primary compression is very fine (the second compression makes it harder to distinguish the quantization step 1 from no quantization at all). Additionally, as one can expect, the embedding changes themselves further confuse the estimator of the primary quantization table, especially for images with primary quality factor above 88. These images are detected as single-compressed images. Both of these components are the likely causes for the worsened classification for higher-quality primary quantization factors. Therefore, further improvement may be possible 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

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).

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 denser set of quality factors.

6. CONCLUSIONS

The task of blind steganalysis of JPEG images in the most general setting (arbitrary quality factors and double compressed images) requires extensive computational resources and thus substantial time. In this paper, we outline a strategy for constructing such a steganalyzer and some preliminary results that show promise that a reliable multi-class steganalyzer for JPEG images can be built.

We describe a steganalytic classifier capable of assigning JPEG images to 6 known JPEG steganographic algorithms. The classifier is built from 23 features calculated from DCT coefficients using the process of calibration. A set of $\binom{n}{2}$ binary support vector machines are 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.

We also investigate the difficult issue of double-compressed images. F5 and OutGuess may produce double compressed images as a result of embedding if the cover image quality factor is not the same as the stego image quality factor. The double compression must be corrected for in the calibration process. This requires estimation of the primary (cover) quality factor. We trained a three-class classifier (F5, OutGuess, and cover) for 12 different cover quality factors. This classifier gave satisfactory performance on a testing set of double-compressed stego images 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.

Our future effort will include merging the single-compression and the double-compression multi-classifiers into one, thus obtaining a general JPEG steganalyzer capable of handling JPEG images of arbitrary quality factor and both single and double compressed images. Given the required number of different combinations of quality factors, steganographic techniques, and payload types, this effort requires tremendous computational power and storage. The classifier will start with the estimator of double compression. If the image is determined as double-compressed, it will be sent to the classifier that deals with double compressed images (an expanded three-class machine from Section 5 distinguishing F5, OutGuess, and cover images). If the image is deemed to be single-compressed, the image will be forwarded to an expanded version of the classifier from Section 4. The double-compression estimator would have to be set to produce a very low rate of false positives (decides double-compressed when the image is not) because once the image deemed double-compressed is sent to the three-class machine, the outcome can only be F5, OutGuess, or cover.

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 *Proceeedings of SPIE Electronic Imaging, Security and Watermarking of Multimedia Contents III*, volume 4314, pages 523–531, San Jose, CA, 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. Springer-Verlag, 1998.
7. R. Chandramouli, M. Kharrazi, and N. Memon. Image steganography and steganalysis. In T. Kalker, I. Cox, and Yong Man Ro, editors, *International Workshop on Digital Watermarking*, volume 2939 of *Lecture Notes in Computer Science*, pages 25–49, 2002.
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. Springer-Verlag, 2003.
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. Springer-Verlag, 2005.
10. 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.
11. J. Fridrich, M. Goljan, and D. Soukal. Searching for the stego key. In *Proceedings of SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI*, San Jose, CA, January 2004.
12. J. Fridrich and J. Lukas. Estimation of primary quantization matrix in double compressed JPEG images. In *International Conference on Image Processing*, Rochester, New York, September 22-25 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, Siana, Italy, 2005. Springer-Verlag.
14. J.J. Harmsen and W.A. Pearlman. Steganalysis of additive noise modelable information hiding. In *Proceedings of SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents V*, pages 131–142, Santa Clara, CA, 2003.
15. 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, CMS 2005*, volume 3677 of *LNCS*, pages 119–128, Salzburg, Austria, September 19–21 2005.
16. 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.

17. 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.
18. S. Katzenbeisser and F.A.P. Petitcolas. Security in steganographic systems. In *Proceedings of SPIE Electronic Imaging, Security and Watermarking of Multimedia Contents IV*, volume 4675, pages 50–56, San Jose, CA, 2002.
19. M. Kharrazi, H. T. Sencar, and N. Memon. Benchmarking steganographic and steganalytic techniques. In *Proceedings of SPIE Electronic Imaging, Security, Steganography and Watermarking of Multimedia Contents VII*, volume 5681, pages 252–263, San Jose, CA, January 2005.
20. S. Lyu and H. Farid. Steganalysis using color wavelet statistics and one-class support vector machines. In *Proceedings of SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI*, pages 35–45, San Jose, CA, January 2004.
21. N. Provos. Defending against statistical steganalysis. In *10th USENIX Security Symposium*, Washington DC, 2001.
22. P. Sallee. Model based steganography. In Kalker, I.J. Cox, and Yong Man Ro, editors, *Digital Watermarking. 2nd International Workshop*, volume 2939 of *Lecture Notes in Computer Science*, pages 154–167. Springer-Verlag, 2004.
23. 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. Springer-Verlag, 2001.
24. 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 *LNCS*, pages 262–277. Springer-Verlag, Berlin, 2005.
25. 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. Springer-Verlag, 1998.