



Annual ADFSL Conference on Digital Forensics, Security and Law

2012
Proceedings

May 30th, 10:30 AM

Double-Compressed JPEG Detection in a Steganalysis System


Jennifer L. Davidson

Department of Mathematics, Iowa State University, davidson@iastate.edu

Pooja Parajape

Department of Electrical and Computer Engineering, Iowa State University, poojap@iastate.edu

Follow this and additional works at: <https://commons.erau.edu/adfsl>

 Part of the [Computer Engineering Commons](#), [Computer Law Commons](#), [Electrical and Computer Engineering Commons](#), [Forensic Science and Technology Commons](#), and the [Information Security Commons](#)

Scholarly Commons Citation

Davidson, Jennifer L. and Parajape, Pooja, "Double-Compressed JPEG Detection in a Steganalysis System" (2012). *Annual ADFSL Conference on Digital Forensics, Security and Law. 2.*
<https://commons.erau.edu/adfsl/2012/wednesday/2>

This Peer Reviewed Paper is brought to you for free and open access by the Conferences at Scholarly Commons. It has been accepted for inclusion in Annual ADFSL Conference on Digital Forensics, Security and Law by an authorized administrator of Scholarly Commons. For more information, please contact commons@erau.edu.

EMBRY-RIDDLE
Aeronautical University™
SCHOLARLY COMMONS

(c)ADFSL



DOUBLE-COMPRESSED JPEG DETECTION IN A STEGANALYSIS SYSTEM

Jennifer L. Davidson

Department of Mathematics
Iowa State University, Ames, IA 50011
Phone: (515) 294-0302
Fax: (515) 294-5454
davidson@iastate.edu

Pooja Parajape

Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011
poojap@iastate.edu

ABSTRACT

The detection of hidden messages in JPEG images is a growing concern. Current detection of JPEG stego images must include detection of *double compression*: a JPEG image is *double compressed* if it has been compressed with one quality factor, uncompressed, and then re-compressed with a different quality factor. When detection of double compression is not included, erroneous detection rates are very high. The main contribution of this paper is to present an efficient double-compression detection algorithm that has relatively lower dimensionality of features and relatively lower computational time for the detection part, than current comparative classifiers. We use a model-based approach for creating features, using a subclass of Markov random fields called *partially ordered Markov models* (POMMs) to modeling the phenomenon of the bit changes that occur in an image after an application of steganography. We model as noise the embedding process, and create features to capture this noise characteristic. We show that the nonparametric conditional probabilities that are modeled using a POMM can work very well to distinguish between an image that has been double compressed and one that has not, with lower overall computational cost. After double compression detection, we analyze histogram patterns that identify the primary quality compression factor to classify the image as stego or cover. The latter is an analytic approach that requires no classifier training. We compare our results with another state-of-the-art double compression detector.

Keywords: steganalysis; steganography; JPEG; double compression; digital image forensics.

1. INTRODUCTION

Steganography is the practice of hiding a secret message or a payload in innocent objects such that the very existence of the message is undetectable. The goal of steganography is to embed a secret payload in a cover object in such a way that nobody apart from the sender and the receiver can detect the presence of the payload. Steganalysis, on the other hand, deals with finding the presence of such hidden message. Steganalysis can be categorized as passive or active. Passive steganalysis deals with detecting the presence of embedded message. Active steganalysis seeks further information about the secret message such as length, embedding algorithm used and actual content. Steganalysis identifies the presence of such hidden messages in an image. Since JPEG is a compressed file format, it requires lower bandwidth for transmission and lesser space for storage. Many embedding software for JPEG such as Outguess, F5, and Jsteg are freely available on the Internet [1]. This makes JPEG a good medium for steganography. Hence, we focus our attention on the steganalysis of JPEG images.

If a JPEG image is compressed twice, each time using a different compression factor or quantization matrix, then the image is said to be *double-compressed*. Steganographic algorithms such as F5 [2] and

Outguess [3] can produce such double-compressed images during the process of embedding the payload. Existence of double-compression thus suggests manipulation of the original image. Blind steganalyzers built on the assumption of single-compressed images give misleading results for the double-compressed images. In Table 1, we present a state-of-the-art stego classifier that has a very accurate stego vs. cover detection rate [4], but applied to JPEG data that has been double compressed. The quality factor for the JPEG images is 75%, and the amount of data embedded in the JPEG coefficients are described in terms of *bits per non zero (JPEG) coefficient*, or bpnz. 0.05 designate that roughly 5% of the available bits for embedding were used, a significantly small amount. 0.4 bpnz is typically the maximum capacity available for embedding. Thus, Table 1 shows that the accuracies for Canvass on JPEG images with no double compression, which are typically in the range of 40% to 100%, here are far less when double compressed images are fed into the Canvass stego detector that was designed to detect images with no double compression, and in the range of 28%-75%. Thus, it is clearly important to detect the presence of double-compression for satisfactory performance of stego images. Detection of double-compression is a binary classification problem where a given image can be classified as either single-compressed or double-compressed. A double-compression detector can be thought of as a pre-classifier to multi-classifiers that detects a number of different steganographic methods used for embedding the payload.

Table 1. Detection accuracy of the GUI software package Canvass applied to double compressed JPEG data.

	Percentage of Detection Accuracy of Canvass			
SQF = 75	0.05 bpnz	0.1 bpnz	0.4 bpnz	Average
Cover	45.39			45.39
F5	28.84	34.54	58.29	40.56
Outguess	42.04	57.73	74.84	58.02

2. JPEG COMPRESSION, DOUBLE COMPRESSION, AND MODE HISTOGRAMS

JPEG stands for Joint Photographic Experts Group. It is a lossy compression algorithm that stores the data in the form of quantized DCT coefficients. Figure 1 shows the process involved in JPEG compression. Each 8x8 block in the original image has the Discrete Cosine Transform (DCT) applied, then those real values are quantized into integer values using the quantization matrix; those integers are scanned in a zig-zag order and the resulting string is entropy encoded using the Huffman coding. The resulting file is then stored as a .jpg file.

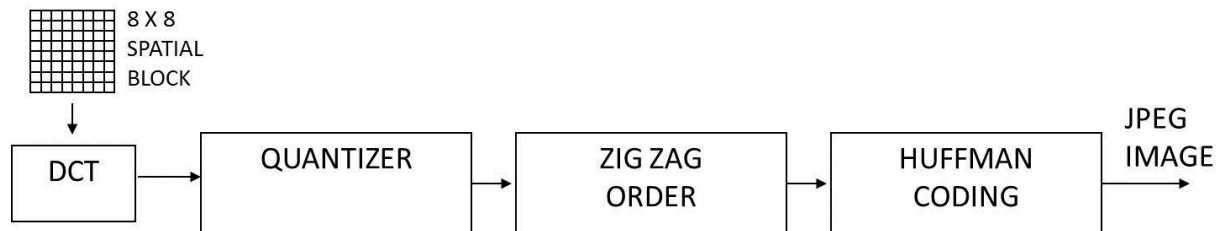


Figure 1. JPEG Compression Process.

The DCT coefficients are given by b_{pq} :

$$b_{pq} = \alpha_p \alpha_q \sum_{x,y=0}^7 a_{xy} \cos\left(\frac{(2x+1)\pi q}{16}\right) \cos\left(\frac{(2y+1)\pi p}{16}\right)$$

where

$$0 \leq p, q \leq 7$$

and

$$\alpha_p = \alpha_q = \begin{cases} \frac{1}{\sqrt{8}}, & \text{if } p, q = 0 \\ \frac{1}{2}, & \text{otherwise} \end{cases}$$

The DCT coefficients b_{pq} are then quantized by dividing each value point-wise using a 8x8 matrix Q followed by rounding to the nearest integer. Q is called the *quantization matrix* (QM). The quantized DCT coefficients are given by

$$B_{pq} = \left[\frac{b_{pq}}{Q_{pq}} \right]$$

where $[]$ denotes rounding to the nearest integer and $0 \leq p, q \leq 7$. Each location (p,q) in the quantization matrix refers to a unique frequency and is called a *mode*. The coefficients at mode $(0,0)$ are called DC coefficients and the coefficients at all the other modes are called AC coefficients. Quantization is an irreversible lossy step. As we can see, division by larger quantization steps results in more compressed data. JPEG standard allows 100 different *quality factors*. A quality factor is an integer between 1 and 100, inclusive. The value 100 is for $Q_{pq} = 1$ for $0 \leq p, q \leq 7$ and it thus corresponds to an uncompressed and highest quality image. Each quality factor is associated with a unique quantization matrix that computes the quality factor at that compression rate. The quantized DCT coefficients are then Huffman encoded for further compression of data. This step is lossless.

Next, we look at the effect of double-compression on the DCT coefficients of the JPEG image. A JPEG image is double-compressed if it is compressed twice with different quantization matrices Q^1 and Q^2 . In that case we have

$$B_{rs} = \left[\left[\frac{b_{rs}}{Q_{rs}^1} \right] * \frac{Q_{rs}^2}{Q_{rs}^1} \right] \quad \text{Eq. 1}$$

Here, Q^1 is called the *primary quantization matrix* (PQM) and Q^2 is called the *secondary quantization matrix* (SQM). Figure 2 shows the double-compression process for a JPEG image. In this example, the image is first compressed at a lower quality factor $QF = 50$, uncompressed and then recompressed at a higher quality factor $QF = 75$. Note that a larger quality factor results in a quantization matrix with smaller values. We observe that, depending on the values of the quantization steps in primary and secondary quantization matrices, the histograms of the DCT coefficients exhibit different characteristic patterns. These patterns are discussed later and are key to our efficient detectors.

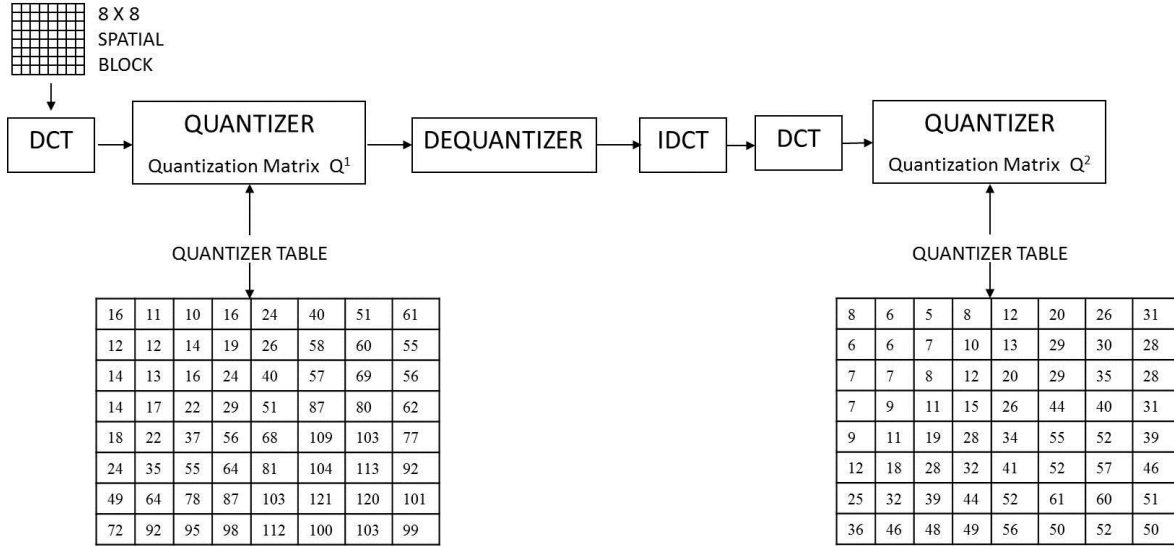


Figure 2. JPEG Double Compression Process. First compression is at QF=50% and second compression is at QF=75%.

When the Primary Quality Factor (PQF) is smaller than the Secondary Quality Factor (SQF), the image is first coarsely quantized and then finely quantized such as in Figure 2. The quantized DCT coefficients can thus take values only from the set $0, n, 2n, 3n, \dots$ where n is determined from Eq. 1. The histogram exhibits zeros at remaining points. The histogram in Figure 3(a) is from a single compressed image, and in Figure 3(b) the histogram shows the characteristic zero pattern for a double-compressed image. Here the primary quality factor is 63 and secondary quality factor is 75. The primary quantization step for mode (2,2) is 9 whereas the secondary quantization step is 6. After the first compression step, the de-quantized DCT coefficients can thus take values that are multiples of 9. After the second compression that includes re-quantization with step size of 6 and rounding, the quantized coefficients can take values only from the set $0, 2, 3, 4, 6, 8, 9, 11, \dots$. We notice that these values are the rounded integer multiples of $n = 9/6 = 1.5$. As a result, zeros occur at locations 1, 4, 7, 10 and so on. This is summarized in Figure 4. We exploit this characteristic to develop features to classify single versus double-compressed images as well as cover versus stego images.

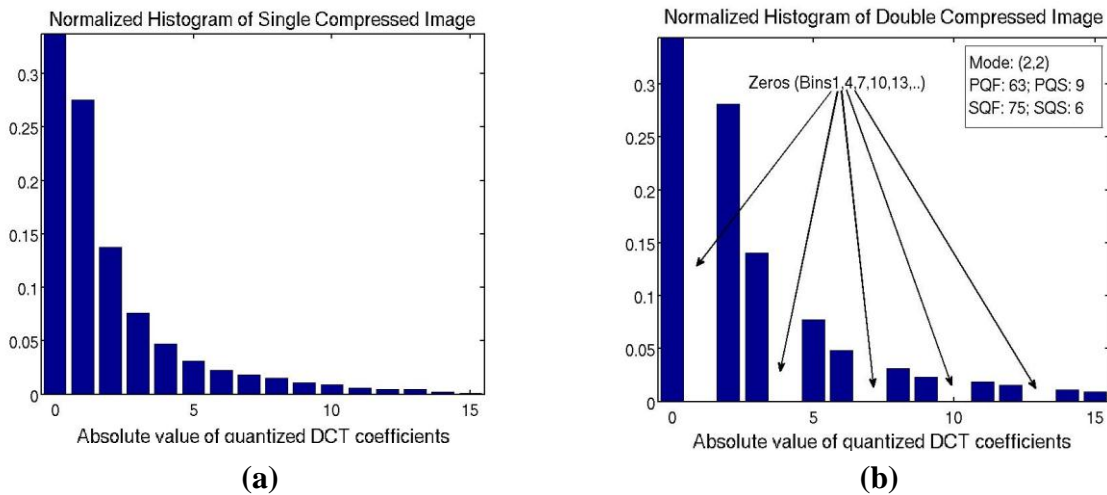


Figure 3. Histograms of quantized DCT coefficients for mode (2, 2) in (a) single-compressed image, (b) double-compressed image with PQF < SQF

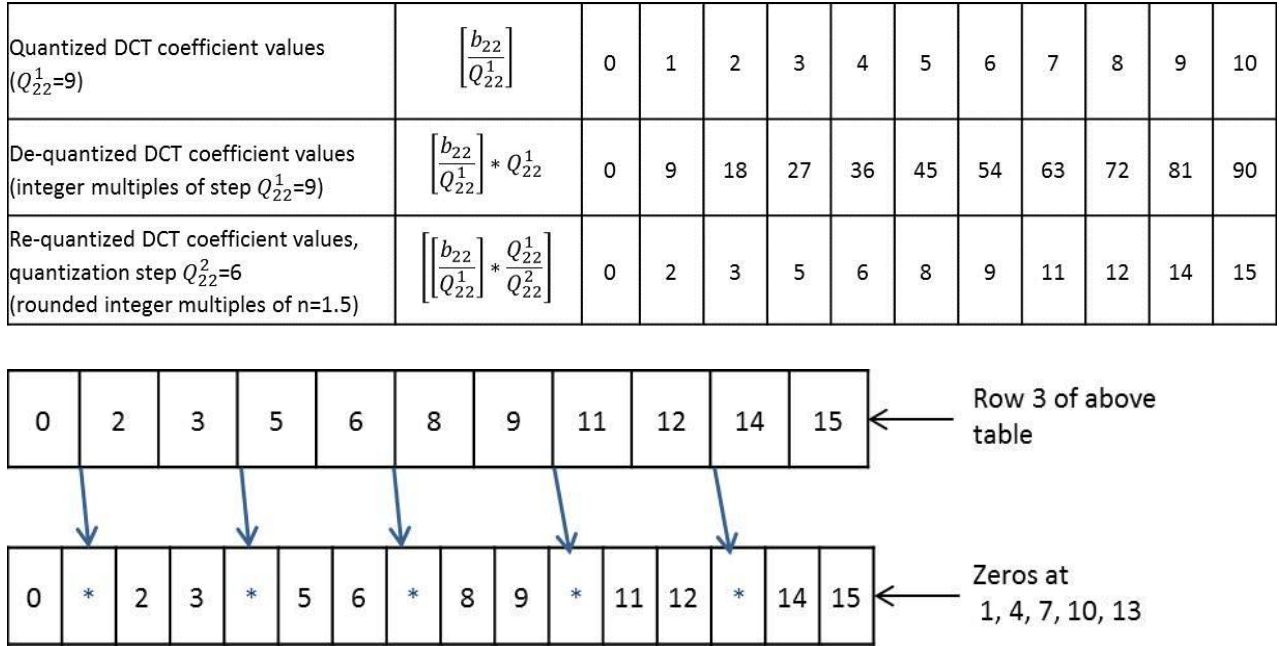


Figure 4. (a) Example of double-compressed coefficient values and (b) resulting zero patterns at mode (2,2).

When the primary quality factor is greater than the secondary quality factor, the mode histograms of the DCT coefficients exhibit peaks at certain places. The locations of these peaks vary according to the combinations of primary and secondary quantization steps in a way similar to the previous case. By experimentation on the image data, we determine that if there is 20% increase in the histogram values (bin heights) followed by a drop in the values, it indicates a peak. The value 20% is chosen after a series of trials and errors. Equation 1.4 also models this phenomenon. In Figure 5(a) and 5(b), we present histograms to demonstrate this phenomenon. In this case, the Primary Quality Factor is 80 and the Secondary Quality Factor is 75. The primary quantization step for mode (2,2) is 5 and the secondary quantization step is 6. The dequantized coefficients after the first compression are thus integer multiples of 5. After requantization with step 6, peaks occur at locations 3, 8 and 13 and so on. These are indicated by the arrows in Figure 5(b). Figure 6 explains this phenomenon in detail. A peak value occurs at bin value 3 after second compression because bin value 3 and 4 from the first compression end up in bin values 3 after the second compression.

3. PATTERN CLASSIFICATION

Detection of double-compression is a binary classification problem where an input image is classified as single-compressed or double-compressed. We propose to use a blind steganalyzer to solve this classification problem. Generally, blind steganalysis is carried out by converting the input image to a lower dimensional *feature space*. These features are then used to train a pattern classifier. Among various pattern classification techniques, Support Vector Machines (SVMs) have proven to be very powerful for the binary classification. SVMs use either linear or kernel-based supervised machine learning algorithms. Linear SVMs are seldom suitable for real world data that is hard to separate without error by a hyperplane. We next discuss the basic workings of an SVM.

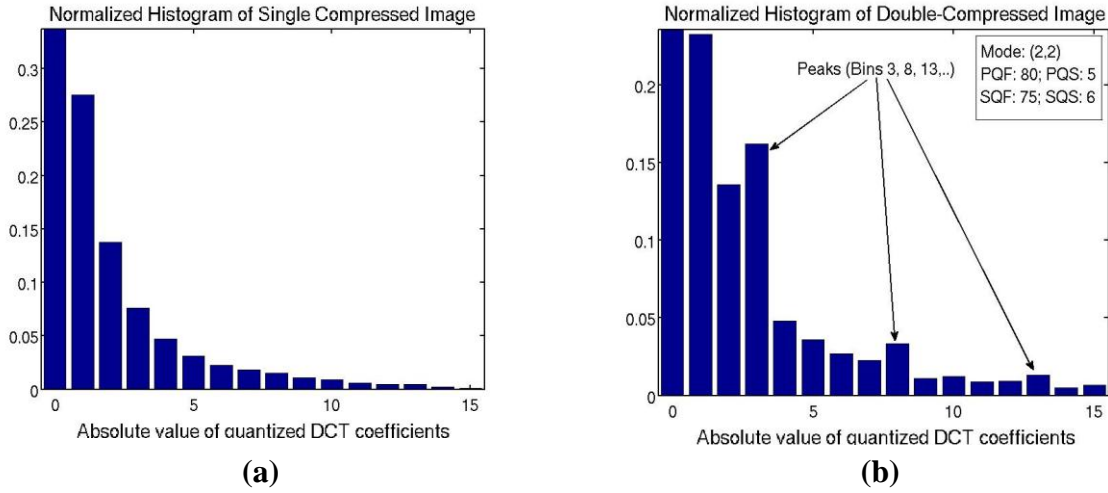


Figure 5. Histograms of quantized DCT coefficients for mode (2,2) in (a) single-compressed image (b) double-compressed image with PQF > SQF.

Quantized DCT coefficient values ($Q_{22}^1=5$)	$\left[\begin{matrix} b_{22} \\ Q_{22}^1 \end{matrix} \right]$	0	1	2	3	4	5	6	7	8	9	10
De-quantized DCT coefficient values (integer multiples of step $Q_{22}^1=5$)	$\left[\begin{matrix} b_{22} \\ Q_{22}^1 \end{matrix} \right] * Q_{22}^1$	0	5	10	15	20	25	30	35	40	45	50
Re-quantized DCT coefficient values, quantization step $Q_{22}^2=6$	$\left[\begin{matrix} b_{22} \\ Q_{22}^1 \end{matrix} \right] * \frac{Q_{22}^1}{Q_{22}^2}$	0	1	2	3	3	4	5	6	7	8	8

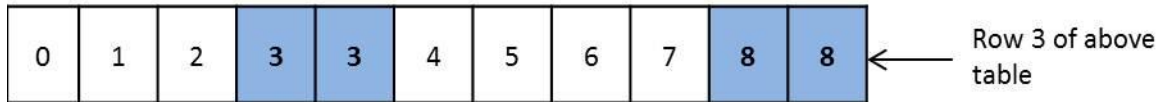
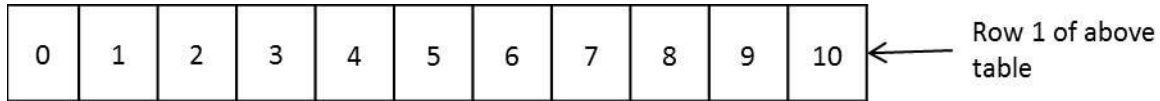


Figure 6. Example of double-compressed coefficient values and resulting peak patterns at mode (2,2). Bins 3 and 4 (row 1) combine to produce bin 3 (row 3).

Consider a given set of training pairs $(x_i, y_i), i=1,2,\dots,k$ where $x_i \in R^n$, n represents the number of training data or features, and $y \in (-1, 1)^k$ represents the class. The SVMs require solution in the form of a weight vector w_i to the following optimization problem

$$\begin{aligned} \min_{\vec{w}, b, \xi} & \left(\frac{1}{2} \vec{w}^T \vec{w} + C \sum_{i=1}^l \xi_i \right) \\ \text{subject to } & y_i (\vec{w}^T \Phi(\vec{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \end{aligned} \tag{Eq. 2}$$

The training data are mapped into a higher dimensional space by function Φ . Then SVM finds a linear separating hyperplane with the maximal margin of space between the hyperplane and data in the higher dimensional space. In this case,

$$K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i)^T \Phi(\vec{x}_j)$$

is called the kernel function whereas $C > 0$ is the penalty parameter of the error term ξ .

There are two distinct phases involved while using a SVM for classification. In the first phase, the SVM is trained on the features extracted from a large number of images, that is, a weight vector \vec{w}_i is found satisfying Eq. 2. In the second phase, the SVM is tested on the data which is previously unseen, that is, a vector of data with unknown class is passed through the SVM and its class is output.

We next discuss the different steps involved in training the SVM. First, the input data is linearly scaled so that all the elements are in the range $[-1, 1]$ or $[0, 1]$. The same scaling is used for both the training and the testing data. This step is necessary to prevent the large feature values from dominating the small values. Then, we choose a kernel function. We have various options for the kernel functions, such as Gaussian, Radial Basis Function (RBF), Polynomial, Sigmoid etc. We use RBF kernel for all the experiments. RBF kernel is the most suitable when the number of features is small and the data needs to be mapped to higher dimensional space using non linear kernel. Also, the RBF kernel is numerically less complex than polynomial and sigmoid kernels. It is given by:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0$$

where γ is the kernel parameter and the norm used is the Euclidean distance.

To run the code to implement the SVM, we used a standard SVM library in [5], which determines the optimum values for the kernel parameters C and γ . Selection of the optimum values is done by performing an exhaustive grid search on the predefined points. Cross validation is required to ensure good performance of the detector on the unknown data. In *v-fold cross-validation*, the training data is divided into v subsets of equal sample size and $v-1$ SVMs are created. Each SVM uses $v-1$ subsets for training and the remaining subset is used as unknown test data. This gives the estimate of prediction accuracy for the unknown data by averaging the accuracy values. Cross validation also prevents the over-fitting problem resulting in better test accuracy [5]. In our experiments we use $v = 5$.

4. LITERATURE REVIEW

Double-compression in JPEG image almost always indicates image manipulation. An innocent image with a certain quality factor can sometimes be saved at a different default quality factor, resulting in double-compression of innocent image. In image forensics, double-compression is used to detect image forgery. A doctored image may be created by pasting a small part of one image into another image. If the quality factors of these two images are different, it results in double-compression of the pasted region.

The embedding algorithms we used were F5, Jsteg-jpeg, JPHide and Seek, Outguess and StegHide. These algorithms are easily accessible to many non-technical users, and available and easy to install from code accessible on the internet. Other more recent algorithms including nsF5, MB, and YASS are not considered here due to their more recent introduction and unfamiliarity outside the steganalysis

community.

In [6] Fridrich et al. proposed a targeted attack for JPEG images embedded using F5 steganographic algorithm. The method mainly considered single-compressed images and was based on estimating the cover histogram from the stego image and determining the relative number of modifications in the histogram values. The authors also suggested a method to estimate the primary quantization steps based on simulating the double-compression using all the candidate quantization matrices. However, in [6], the authors assumed that the images are already known to be double-compressed. No method was implemented for the double-compression detection. Lukas and Fridrich extended this idea in [7] and proposed three methods for detection of primary quantization steps in double-compressed JPEG images. Two of these methods were based on simulated double-compression as in [6]. The methods were computationally intensive and did not lead to reliable results. The third method was based on neural networks and it outperformed the first two methods. However, in the latter paper the authors assumed that the double-compressed images were cover images only.

He et al. [8] proposed a scheme to detect doctored JPEG images based on double quantization effects seen in the DCT coefficient histograms. The method used Bayesian approach for detecting doctored blocks in the images. A support vector machine was trained on the four dimensional features that are similar to Fisher discriminator in pattern recognition. This method is not efficient when a high quality image is recompressed using a lower quality factor. Also, for more accurate formulation of the feature vectors, the method needs the primary quality factor that cannot be estimated from the proposed algorithm. In [9], M. Sorell developed a method to detect the primary quantization coefficients of double-compressed images for forensic purposes. The method was used to detect re-quantization in innocent JPEG images to identify the source of the images. Methods for detecting doctoring of JPEG images, where a region undergoes double-compression after being cut and pasted into a larger image, have been investigated in [8,9,10]. Although some of these methods lead to accurate detection of double-compression, the application was for forensic purposes, not steganalysis. Since the embedding process for steganography can change the image statistics significantly, application of these methods for steganalysis might involve significant algorithmic modifications.

In [11], T. Pevny and J. Fridrich proposed a stego detector for both single and double-compressed images. They used a neural network-based primary quantization step detector to detect double-compression. Depending on the result of this detector, the image was sent to one of the two banks of multi-classifiers designed for single and double-compressed images. The detection of double-compression was based on the naive approach of comparing the primary and secondary quantization steps and it led to inaccuracies in the detection. To overcome this, a separate double-compression detector was designed in [12] and it was used as a pre-classifier to the stego detector. Instead of neural networks, the authors used Support Vector Machines. The SVMs were trained on features based on first order statistics of the mode histograms of quantized DCT coefficients. The primary quantization step detection followed the double-compression detection. For each possible combination of primary and secondary quantization steps, the authors use a separate SVM which led to a total of 159 SVMs. This approach, while increasing accuracies, is very compute-intensive.

C. Chen et al. proposed similar machine learning based scheme for JPEG double-compression detection [7] similar to [12]. The proposed 324 dimensional features were modeled using a Markov process and transition probability matrices of 2D difference arrays of quantized DCT coefficients were used to create the feature vectors. Then a support vector machine was trained on these features. The method is discussed in detail in Section 6 and the results are compared with our detector. Chen's model is highly compute-intensive and we show that accuracies are lower than those resulting from our approach. Pevny et. al created a complete multi-classifier for single and double-compressed images [2] by combining all the modules from [10,12]. The double-compression detector was based on histogram mode statistics. The primary quality step estimation was done by a bank of SVMs designed for all the possible combinations of primary and secondary quality steps. Then two banks of multi-

classifiers were created for single-and double-compressed images. This model is also highly compute-intensive.

On a related topic, in [17], the authors perform an extensive analysis of the errors introduced by initial JPEG compression and double compression. The main difference is that they do not apply their analysis to stegoimages, only to cover or innocent images. They investigate three different types of forensics, one of which is determination of the primary standard JPEG quantization table in a double-compressed JPEG image. Their detection schemes are based on sound and novel analysis of the characteristics of the error that can occur during JPEG compression. The authors use a simple formula and have relatively high detection accuracy, ranging from roughly 70 percent to 100 percent. The method we implement gives ranges of roughly 50-100% accuracy, but for cover and stego images, the latter embedded with a range of embedding rates, from 0.05 to 0.4 bpnz. See Figure 16 below. The two detection schemes cannot be compared directly as different databases and image types (cover and stego) were used in the two experiments. Other methods such as in [18], use models to characterize the effects of rounding and recompression, but again do not use stego images in the analysis. Our methods are applied to both cover and stego image data.

5. OUR APPROACH

The model proposed by Chen in [7] detected double compressed images, as did the model in [12]. The main disadvantage we see is very heavy computational resource requirements. In [12], 15 SVMs are required for steg vs. cover detection of single-compressed images, 159 additional SVMs are required for primary quality steps estimation whereas 3 SVMs are required for steg vs. cover detection of double-compressed images. Considering the time required for training and testing of each of these SVMs and the computational resources at hand, we decided to create a different approach that would minimize the number of SVMs required while maintaining accuracies, thus saving a lot of computational time.

A major contribution of our approach towards time saving is achieved in the primary quality factor estimation step. Instead of using SVM for each possible combination of primary and secondary quality step as in [2], we use an analytical approach based on the signature patterns in the histograms. The analytical method does not involve training and testing of SVM and is significantly less computationally intensive.

We created a steganalyzer that (refer to Figure 7):

1. Classifies an image as single-compressed or double-compressed (Box 1).
2. Classifies a double-compressed image as cover or stego and estimate the primary quality factor from the set of standard JPEG quality factors for further analysis of the image using targeted attacks. [6,13]. (Box 2).
3. Classifies a single-compressed image as cover or stego (Box 3). (Note: Only QF = 75 case is implemented in the current software.)

We assume three classes for double-compressed images: cover, F5 and Outguess. In practice, double-compressed images can have any quality factor but we restrict the scope to secondary quality factors 75 and 80 that are the default quality factor for Outguess and F5 respectively.

In section 3.1, we propose the overall scheme for steganalysis of single and double-compressed images. In section 3.2, we describe the image database used for training and testing various SVMs. Section 3.3 gives detailed description of the features used for steganalysis whereas section 3.4 contains a brief discussion about features used by the state-of-the-art double- compression detectors.

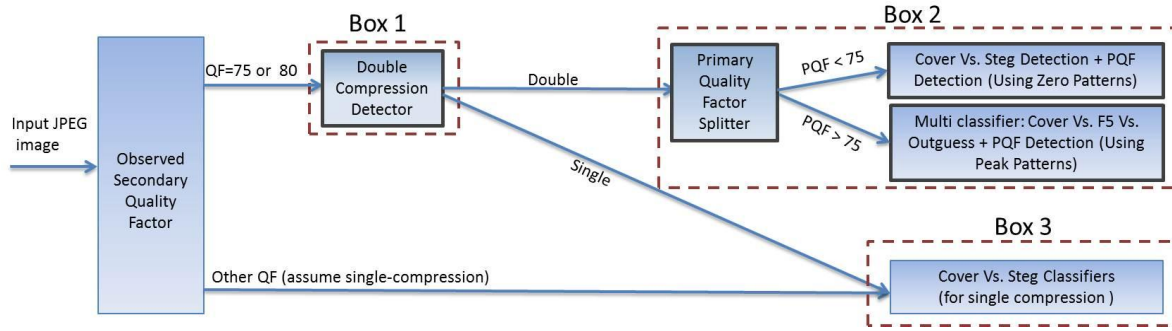


Figure 7. Our steganalyzer.

Figure 7 shows the complete flow of our steganalyzer. If the observed quality factor of the input image is 75 or 80, there is a high probability of double-compression. This image is passed to the double-compression detector. If the image is single-compressed at $QF = 75$, it is passed to the existing single-compression steganalyzer software. If the image has $QF \neq 75$ or $QF \neq 80$, then we do not process it at this time.

If the image is found to be double-compressed, it is further given to PQF splitter module. The PQF splitter determines if the PQF of the image is less than or greater than its SQF and accordingly classifies the image into two classes. This step helps in the further processing because it is observed that the signature patterns in the histograms are different based on the relation between the PQF and the SQF. Depending on this relation, two different approaches are followed. When the PQF is less than the SQF, we use the analytical approach based on the histogram signature patterns to achieve stego vs. cover classification. When the PQF is greater than the SQF, we use a multi-class SVM-based pattern classifier.

6. IMAGE DATA

In this section, we describe the database used for all the experiments. This description is necessary because the performance of a SVM-based classifier depends on the database used for training.

We use images from BOWS-2 database available from [7]. This database was created for the BOWS-2 contest. It consists of approximately 10000 grey-scale images in pgm format. The images are of uniform size of 512x512 pixels. We divide the BOWS-2 database into two mutually exclusive sets of equal size. One set is used for creating the training database and other set is used for creating the testing database. This division allows use of test images that are previously not seen by the SVM. For all the experiments, a total of 30000 images are used for training, 15000 of each class.

For single-compressed images, five steganographic algorithms are considered: F5, Jsteg, JP Hide and Seek, Outguess and StegHide. For double-compressed images, only F5 and Outguess are considered because we assume that these are the only algorithms that are likely to produce double-compressed images. For all the embedding algorithms except Outguess, three embedding levels are used: 0.05 bpnz, 0.1 bpnz and 0.4 bpnz. Outguess fails to embed the payload at 0.4 bpnz. Hence the 0.2 bpnz embedding rate is used in this case only. In this case, Bits Per Non-Zero ac coefficient (bpnz) is used to describe the payload length. Each steganographic algorithm embeds a binary payload consisting of a randomly generated bitstream into the quantized DCT coefficients array of the JPEG image.

In our experiments, we consider 33 primary quality factors in the set

$$S = \{63, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94\}.$$

Secondary quality factors are limited to 75 and 80. We ignore primary quality factors 64, 65, 66 and 67 because F5 and Outguess fail to embed the payload in the images with these quality factors.

7. FEATURES

There are a wide variety of features that have been used for pattern classification. Farid [14] used features based on the linear prediction error of wavelet sub-band coefficients. In [15], features based on quantized DCT coefficients were used for blind steganalysis. Shi et al. [16] used a Markov process to model four directional differences between the neighboring quantized DCT coefficients to create features. In all the methods, the idea is to use features that are sensitive to the changes introduced by the embedding process. Such features contain the information that can effectively separate the classes under consideration. We used the conditional probabilities that are from Partially Ordered Markov Models (POMMs) for double-compression detection and steganalysis of single and double-compressed images. POMMs exploit neighborhood dependencies amongst the quantized DCT coefficients and have been successfully used for the forward and inverse texture modeling problem. In order to improve the performance of the POMMs, we additionally use features based on the histogram signature patterns. We next present the details of these features.

Markov random field models (MRFs) are a well-known stochastic model applied to many different problems. In imaging problems, a MRF is defined in terms of local neighborhoods of pixels called cliques that exhibit the probabilistic dependency in the imaging problem. There is an implicit underlying graph when using a MRF. The neighborhoods of pixels for POMMs, however, have an implicit underlying *acyclic directed graph*, which in turn is related mathematically to a *partially ordered set*. We omit the details of the relation to partial orders, but now describe the basic POMM model in terms of an acyclic directed graph.

Let a finite acyclic digraph (V, E) with corresponding poset $(V, <)$. Here $V = (V_1, V_2, \dots, V_k)$ is the set of vertices and E is the set of directed edges given by $E = \{(i, j) : V_i, V_j \in V \text{ and } (i, j) \text{ is an edge with tail on } i \text{ and head on } j\}$. (V, E) is called *acyclic* when there does not exist a set of edges $(i_1, j_1), (i_2, j_2), \dots, (i_k, j_k)$ where $j_n = i_{n+1}, n=1, 2, \dots, k$ and $j_k = i_1$. Figure 8 shows an acyclic digraph. We notice that there is no path of directed edges that start and end at the same vertex.

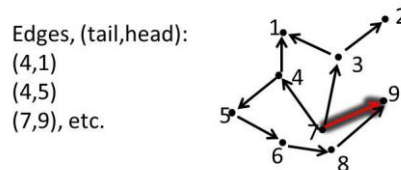


Figure 8. Example of an acyclic directed graph.

Definition 1: A set of elements V with a binary relation $<$ is said to have a partial order with respect to $<$ if:

1. $a < a, \forall a \in V$ (reflexivity)
2. $a < b, b < c \Rightarrow a < c$ (transitivity)
3. If $a < b$ and $b < a$, then $a = b$ (anti-symmetry).

In this case, $(V, <)$ is called a *partially ordered set* or a *poset*.

Definition 2: For any $B \in V$, the cone of B is the set $\text{cone } B = \{C \in V : C < B, C \neq B\}$.

Definition 3: For any $B \in V$, the *adjacent lower neighbors* of B are elements in C such that (C, B) is a directed edge in (V, E) . $\text{adj}_{<} B = \{C : (C, B) \text{ is a directed edge in } (V, E)\}$.

Definition 4: Any $B \in V$ is a minimal element if there is no element $C \in V$ such that $C < B$, that is, there is no directed edge (C, B) . Let L_0 be the set of minimal elements in the poset.

Figure 9 explains these definitions pictorially.

With this background, we now proceed towards defining POMMs. Let $P(A)$ be the discrete probability for r.v. A and $P(A|B)$ be the conditional probability of r.v. A given another r.v. B . L_0 is the set of minimal elements in the poset.

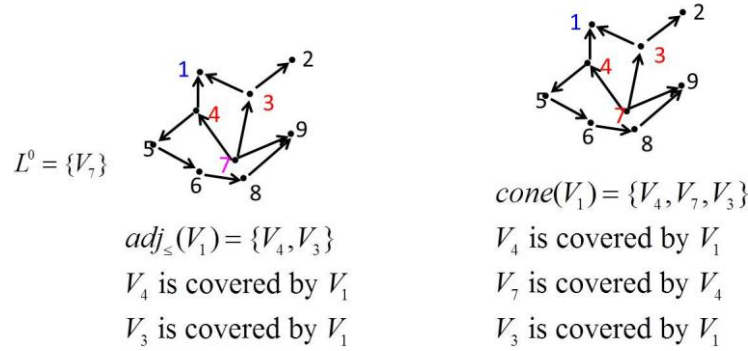


Figure 9. Adjacent lower neighborhoods, cone sets, minimal set.

With this background, we now proceed towards defining POMMs. Let $P(A)$ be the discrete probability for r.v. A and $P(A|B)$ be the conditional probability of r.v. A given another r.v. B . L_0 is the set of minimal elements in the poset.

Definition 5: Consider a finite acyclic digraph (V, E) of r.v.s with the corresponding poset $(V, <)$. For $B \in V$, consider a set Y_B of r.v.s not related to $B, Y_B = \{C : B \text{ and } C \text{ are not related}\}$. Then $(V, <)$ is called a *partially ordered Markov model* (POMM) if for any $B \in V \setminus L_0$ and any subset $U_B \subset Y_B$ we have

$$P(B \mid \text{cone } B, U_B) = P(B \mid \text{adj}_{<} B)$$

In this case, the lower adjacent neighbors in B describe the Markovian property of the model.

Next, we are interested in using POMMs for modeling steganographic changes to image. For steganalysis applications, we create features that use directly the quantized DCT coefficient values that are modeled by a POMM. We describe a general approach that lets the steganalyst use her expertise to construct such a POMM. First, subsets of pixels are chosen that contain relative information that may have changed after embedding a payload. Next, a function f from the set of subsets to the real numbers is found that is used to quantify the change in values that occur after embedding, and applied to each subset under consideration. Then, an acyclic directed graph is created from the set of subsets and values in the range of f (vertices) and the binary relation describing the function and its range (edges). The induced poset is constructed from this acyclic digraph, and a POMM is created using this poset. This gives rise to the conditional probabilities $P(B \mid \text{adj}_{<} B)$ which are used as features for steganalysis.

Let A be a $M \times N$ array of quantized DCT coefficients: $A = \{A_{i,j} : 1 \leq i \leq M, 1 \leq j \leq N\}$. Let $Y = \{Y_1, Y_2, \dots, Y_t\}$ be a collection of subsets of r.v.s in A . Here, every Y_i is a shift invariant ordered set. For example, consider a case where $Y_1^h = \{A_{1,1}, A_{1,2}\}$, $Y_2^h = \{A_{1,2}, A_{1,3}\}$, and so on. Each Y_i^h is a set containing two pixels that are adjacent in the horizontal direction. Let $Y^h = \{Y_1^h, Y_2^h, \dots, Y_k^h\}$ contain all such sets of r.v.s in the array A . Let f be the function $f : Y \rightarrow R$ where R is a set of real numbers defined by

$$f(y_1, y_2) = y_1 - y_2$$

$$f(Y_i^h) = f(A_{j,k}, A_{j,k+1}) = A_{j,k} - A_{j,k+1}, \text{ for some indices } i, j \text{ and } k.$$

In this case, $f(Y_i)$ is the image of Y_i under f and Y_i is the pre-image of $f(Y_i)$. An acyclic digraph (V, E) is created from the set of vertices $V = Y \cup f(Y)$ and the set of edges $E = \{E_i\}$ between an element of the range from f and an element of Y . Thus, edge $E_i = (f(Y_i), Y_i)$ has tail on image $f(Y_i)$ and head on pre-image Y_i . Thus (V, E) forms a *function-subset* or *f-Y acyclic digraph*. This acyclic digraph is used to create a sequence of POMMs whose conditional probabilities are used as features. If f is a function that exploits the dependency among the quantized DCT coefficients, then it is considered useful for steganalysis. For such function f , $P(Y_i | f(Y_i))$, which is a frequency of occurrence of pre-image of $f(Y_i)$, can be used to distinguish between cover and stego images. This is the motivation for using *f-Y acyclic digraphs*.

Figure 10 shows the example of *f-Y* directed graphs we use for our features. There are two subsets Y_{112} and Y_{249} shown which represent a portion of the quantized DCT coefficients array. Function f , when applied to these subsets produces value of -1. So the probability $P(3, 4 | (3 - 4))$ measures the frequency of occurrence of this pattern. A collection of all such conditional probabilities are captured by the POMMs.

As mentioned above, function $f = y_1 - y_2$, where y_1 and y_2 are adjacent pixels, is a very useful feature for steganalysis. The directional difference arrays created in this manner have been successfully used in [7], [16]. We use this function f to create a series of POMMs, one in each of the horizontal, vertical, diagonal, and minor diagonal (the reverse diagonal) directions. The conditional probabilities thus obtained are averaged over a collection of POMMs.

For a rectangular array A of r.v.s we consider four directional subsets in horizontal, vertical, diagonal and minor diagonal directions given by $Y_{i,k}^h = \{A_{i,k}, A_{i,k+1}\}$, $Y_{i,k}^v = \{A_{i,k}, A_{i+1,k}\}$, $Y_{i,k}^d = \{A_{i,k}, A_{i+1,k+1}\}$, and $Y_{i,k}^m = \{A_{i+1,k}, A_{i,k+1}\}$. From the four sets Y^h, Y^v, Y^d, Y^m thus obtained, we create four acyclic digraphs (V^*, E^*) where $V^* = (Y^* \cup f(Y^*))$ and $E^* = \{E_i^* : E_i^* = (f(Y_i^*), Y_i^*)\}$, and $*$ $\in \{h, v, d, m\}$. For each digraph, a POMM is defined by its conditional probability given by:

$$P^*(Y^* | f(Y^*)) = P^*(Y^*, f(Y^*)) / P^*(f(Y^*))$$

These probabilities are calculated from the histogram bins. The histograms are clipped between $[-T, T]$. This not only avoids the sparse probability density function at the histogram ends, but also reduces the number of computations and size of feature vector. Thus, we have $-T \leq A_{i,j} \leq T$. This limits the values of P^* to $(2T + 1)^2$ in each direction. In our experiments, we use $T = 3$. We now define a matrix F^* of size $(2T + 1) \times (2T + 1)$:

$$F^*(w, z) = P^*(Y^* | f(Y^*)) = P^*(w, z | f(w - z)),$$

From these, a set of $(2T + 1)^2$ features can then be defined as

$$F(w, z) = \frac{1}{4} \sum_{* \in \{h, v, d, m\}} F^*(w, z) = \frac{1}{4} \sum_{* \in \{h, v, d, m\}} P^*(Y^* | f(Y^*)) = \frac{1}{4} \sum_{* \in \{h, v, d, m\}} P^*(w, z | f(w - z)) \quad \text{Eq. 3}$$

The POMMs are applied on the global array to capture intrablock dependencies and on the mode arrays to capture interblock dependencies. These conditional probabilities are used to define our features.

The intrablock features are defined on the set of global Q-DCT coefficients, on the entire array itself.

Eq. 3 is used to calculate the average values of the four directions. This produces $(2T + 1)^2$ features.

Interblock features are defined on the mode arrays. 64 mode arrays are created from the 64 modes by collecting all the Q-DCT values at each mode. Eq. 3 is then applied to each of the 64 arrays. Those values are averaged over all 64 arrays, giving an additional $(2T + 1)^2$ feature values.

We apply *calibration* to the image before extracting features. An image is calibrated by decompressing to the spatial domain and then compressing it back after cropping by few pixels in both directions in the spatial domain. This helps to create the features that are dependent on the data embedded in the image rather than the image content itself [16]. For an image I^o , we obtain a calibrated image I^c . The intra and inter block POMMs are calculated for both I^o and I^c . The final feature vector F is obtained by taking the difference between F^o and F^c , $F = (F^o - F^c)(w, z)$. We get a total of $2(2T + 1)^2$ POMM features. For our experiments, we use threshold value $T = 3$. This results in 49 intrablock and 49 interblock features giving a total of 98 POMM features.

The second set of features is derived from the histogram patterns noted in the above section. As shown above, when the primary quality factor is smaller than secondary quality factor, the mode histograms of Q-DCT coefficients have zeros at some places. These zeros can be captured by looking at the forward differences between histogram bin values. For a single-compressed image, the distribution of DCT coefficients is ideally Gaussian. The forward differences are thus almost always small and positive. For double-compressed images, the zeros can take the form of local minimum instead of being exact zeros. So instead of finding absolute zeros, if the drop in the successive histogram values is more than 75%, it is considered to indicate presence of a zero. For images with primary quality factors less than the secondary quality factor, this approach is followed. For secondary quality factor of 75, the primary quality factors under consideration are given by:

$$s \in S_{75} = \{63,68,69,70,71,72,73\}$$

The quality factor 74 is not taken into account because the quantization steps for the 9 lower modes we use are identical to the quantization steps for quality factor 75. Thus images having primary quality factor 74 and secondary quality factor 75 are considered as single-compressed for classification purposes.

When the secondary quality factor is 80, there are 13 primary quality factors under consideration which are given by:

$$s \in S_{80} = \{63,68,69,70,71,72,73,74,75,76,77,78,79\}$$

For any given image, we first inspect the first 22 bin values in the histograms of the nine low frequency DCT coefficients. The nine low frequency modes are:

$$H = \{(0, 1), (0, 2), (0, 3), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (3, 1)\}$$

The remaining modes are not considered because most of the quantized DCT coefficients have value zero or close to zero. The histogram location k for mode m is represented by $h_m(k)$ where $0 < (m = i+j) \leq 3$; i and j are the coordinates of the location in each 8×8 block. Using the values from $h_m(k)$, we create a 9 dimensional matrix \hat{M} for the candidate image.

$$\hat{M}(m, k) = \chi_{\geq 0} \left(\left(\frac{h_m(k) - h_m(k+1)}{h_m(k)} \right) - 0.75 \right)$$

where $1 \leq m \leq 9$ and $1 \leq k \leq 22$, and χ is the indicator function. Each value $\hat{M}(m, k)$ is a zero feature. The matrix \hat{M} is called the *zero feature matrix*. We use zero features for double-compression detection, Primary quality factor estimation, and cover vs. stego detection when the primary quality factor is smaller than the secondary quality factor.

We achieve highly accurate double-compression detection rate by combining the POMM features and zero features. We observed that the accuracy of the double-compression detector improves significantly when the zero features are added to the 98 POMM features. The zero features are particularly effective when the primary quality factor of an image is lower than the secondary quality factor. For a given input image, a 9×21 sized matrix \hat{M} of the zero locations is created, and then each matrix row of values are summed to produce 9 features. These features are appended to the 98 POMM features and used to train the SVMs.

We also observed that the signature patterns in the histograms are generally distinctive for a given combination of primary and secondary quality factors. We quantized the effect of this phenomenon for estimating the primary quality factor of the image. For a given combination of primary and secondary quality factors, we create standard matrices M_s^t , where t is the secondary quality factor. These matrices capture the expected locations of zero-valued histogram bins. Thus, for primary quality factor s and secondary quality factor t , $s < t$ we define

$$M_s^t(m, k) = \begin{cases} 1, & \text{if bin } k \text{ of mode } m \text{ is zero} \\ 0, & \text{if bin } k \text{ of mode } m \text{ is not zero} \end{cases}$$

All of the matrices M_s^t can be calculated beforehand and stored. To use these matrices to estimate the primary quality factor of an unknown image that has already been detected as double compressed whose secondary quality factor t is less than 75 (or 80), we calculate the matrix \hat{M} for the unknown image. Then we find the matrix M_s^t that is the “closest” to \hat{M} from the standard set of matrices already precalculated. The closest estimated standard matrix M_{est} is formed by minimizing the absolute difference between each M_s^t and \hat{M} :

$$M_{est}^t = \arg \min_{M_s^t \in S_{std}^t} \sum |M_s^t - \hat{M}|$$

We also observed that the dips in the histograms assume absolute zero value whereas for stego images, the dips can be characterized by drop in the histogram values. We used this fact in the successful detection of stego images.

Cover Vs. stego detection for PQF < SQF. The histogram zero patterns can also be used for stego detection in images with $PQF < SQF$. It was stated earlier that the 75% drop in the histogram bin values characterizes a zero for a double-compressed image. Ideally a zero bin should have value exactly equal to zero but the statistics of the DCT coefficients changes due to the process of embedding. In order to account for these changes, a 75% drop is considered to indicate a zero. For double compressed cover images, the drop is 100% and the zero bins actually take zero value. For double compressed stego images, we consider even a small drop in the bin value as a zero. This acts as a distinguishing feature between cover and stego images. For all the standard matrices M_s^t from the standard set S_{std} , we count the number of absolute zero bins that are expected to occur for each combination of PQF and SQF.

For an input image, first the PQF is estimated. For the estimated PQF, the expected count of zero bins is obtained. From matrix \hat{M} derived from an input image, we count the total number of absolute zero values. If this count matches exactly with the expected standard count, the image is a cover image;

otherwise it is a stego image.

In this case, classification is done using an analytical method as opposed to SVM-based detection. For a given detector, set S_{std} of standard matrices is created and stored beforehand. This method saves a lot of time required in feature extraction and SVM training, and is also very accurate.

8. RESULTS

We compare our double compression detector with a state-of-the-art double-compression detector by Chen et al [7]. In this method, nearest-neighbour differences in the vertical, horizontal, diagonal and minor diagonal directions are calculated, resulting in a set of four 2-dimensional difference arrays. In order to reduce the computations, these arrays are thresholded to values in between -4 and +4. The difference arrays are modeled using a one-step Markov process characterized by a transition probability matrix (TPM). This results in a 9 TPM matrix which leads to 324 features. These features are used to train a support vector machine. See [7] for more details.

The authors assume that the double-compressed images are cover images only and the primary quality factors considered are in the range 50 to 95 in steps of 5. The detection accuracies obtained are very high. However, when the method is used for double-compressed stego images, the detection accuracies drop. We compare this detector and our POMM based detector.

POMM detector. An unknown image is input to the steganalyzer. If the observed quality factor of the image is 75 or 80, it is likely to be double-compressed. This image is passed on to the double-compression detector (referred to as DCDetector). Two separate detectors are created for secondary quality factors 75 and 80. Each detector is a binary classifier trained on 15000 single-compressed and 15000 double-compressed image data. For the first detector, the images with PQF 74 and 75 are excluded whereas for the second detector, images with PQF 80 are excluded from the training database. This is because the statistics of the images with PQF equal to SQF is the same as single-compressed images. The stego images are further divided amongst the number of classes and subdivided amongst three embedding levels. Double-compressed images are also divided equally amongst the different primary quality factors.

For testing, we start with 1500 unique cover images at each PQF in the range 63 to 94. In case of F5 and outguess, we use 500 images at each of the three embedding levels, resulting in 1500 images at each PQF.

Our double-compression detector uses combination of 98 POMM features and 9 zero features giving a total of 107 features. A double-compression detector is required to have very low false positive rate. False positive rate is the percentage of single-compressed test images that get classified as double-compressed. If a single-compressed image is classified as double-compressed, it can be assigned to only cover, F5 or outguess instead of six classes: cover, F5, jp hide & seek, jsteg, outguess and steghide. This leads to classification errors. In Figure 11, we present the accuracy plots.

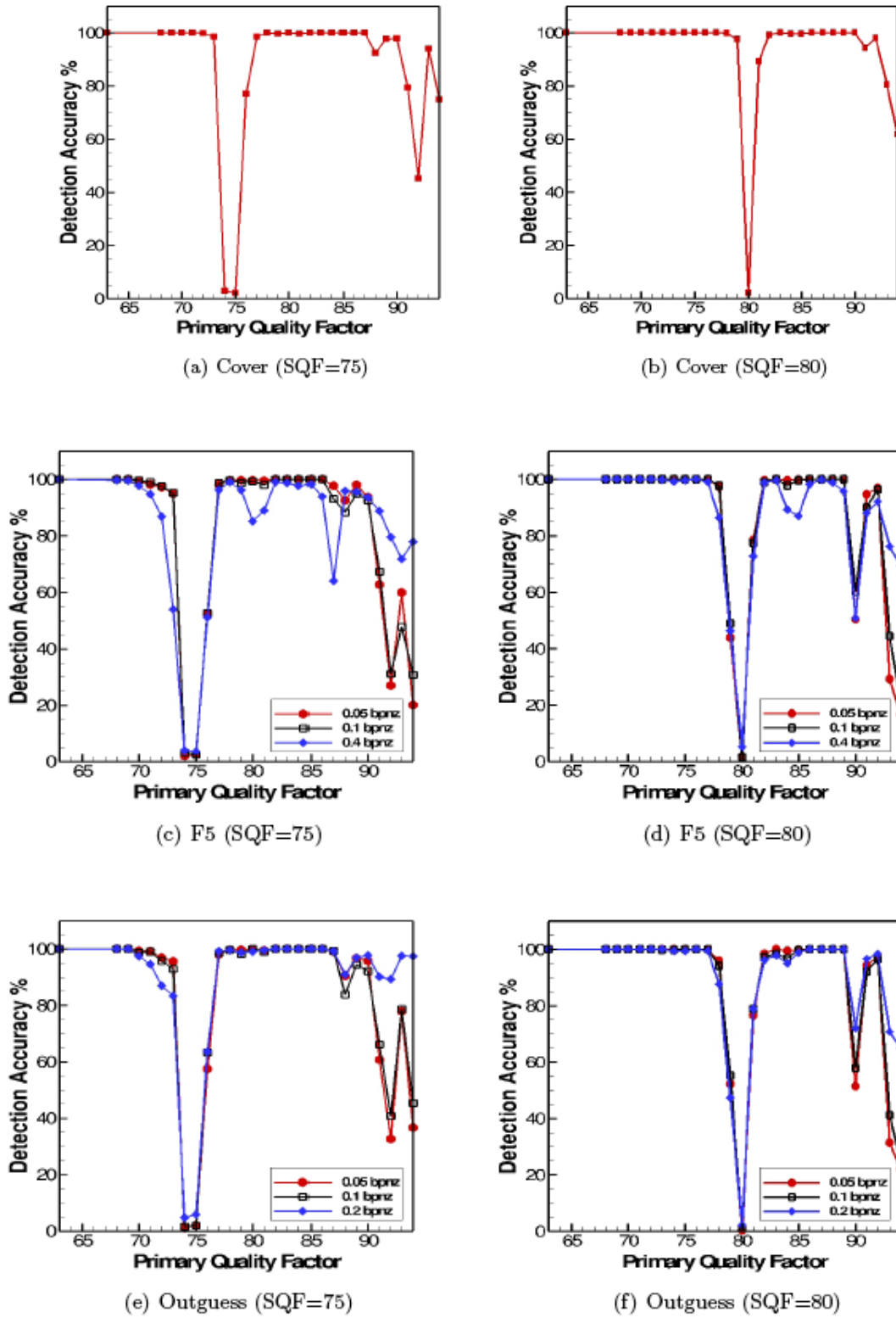


Figure 11. Accuracy of double-compression detectors on double-compressed cover [(a),(b)], F5 [(c),(d)] and outguess [(e),(f)] images.

The DCDetector gets very high accuracy for many of the values. In Figure 11, it can be seen that the DCDetector can classify cover images accurately except for two cases. First case is when the PQF is equal to or close to the SQF (74, 75 and 76 in the left column plots and 80 and 81 in the right column plots) and second case is when PQF is greater than 90. PQF = 74 and SQF = 75 is a special case because

the nine low frequency locations that we consider are identical for quality factors 74 and 75. This makes it impossible to detect images with this combination of quality factors using only this information. This is because when the PQF is very close to the SQF, the statistics of double-compressed image is close to the corresponding single-compressed image. Also, for PQF greater than 90, most of the quantization steps are equal to one due to which the effect of double-compression is not obvious. This justifies the drop in

the detection accuracies in these two cases. A similar trend is observed in case of F5 and ouguess with respect to the PQF. Also, the detection accuracies drop when the embedding level increases. For the cases mentioned above where the PQF is close to the SQF, if the detector classifies the image as single-compressed, it is considered as a correct decision. In order to determine the overall detection accuracy, we consider the average of true positive rate for double-compressed test images and true negative rate for the single-compressed test images.

In Table 2, we present the false positive rates. Except for F5 at 0.4 bpnz and ouguess at 0.2 bpnz, the false positive rates are lower than 8% for SQF = 75. For SQF = 80, the false positive rates are lower than corresponding cases of SQF = 75 by at least 0.5.

(a)					(b)				
% False Positive Rate					% False Positive Rate				
SQF = 75	0.05 bpnz	0.1 bpnz	0.4 bpnz	Average	SQF = 80	0.05 bpnz	0.1 bpnz	0.4 bpnz	Average
Cover	3			3	Cover	1.8			1.8
F5	2.8	3.8	8.6	5.07	F5	1	2.6	4.4	2.67
JPHS	3.4	2.6	4.6	3.53	JPHS	0.4	0.6	0.8	0.6
Jsteg	1.2	0.8	3	1.67	Jsteg	1	0.2	0.4	0.53
Ouguess	3	6.8	8	5.93	Ouguess	0.2	0.6	1.8	0.87
Steghide	3	2.4	2.2	2.53	Steghide	0.4	0.6	0.4	0.47

Table 2. False positive rates for cover and stego for the double compression detector.

We also implemented Chen’s double compression detector. In Figure 12, we give the percent false positive rate for the three levels of embedding for the 5 embedding algorithms for both Chen’s detector and the POMM detector. In Figure 13, we give the overall average detector accuracy rates. These two figures show that Chen’s detector has accuracies comparable to our POMM-based detector for a limited number of PQFs. For most of the PQFs, the accuracies are lower than our detector.

	% FPR (Chen SQF=80)				% FPR (POMMSQF=80)			
	0.05 bpnz	0.1 bpnz	0.4 bpnz	Average	0.05 bpnz	0.1 bpnz	0.4 bpnz	Average
Cover	5.2			5.2	1.8			1.8
F5:	5.8	4.4	8.4	6.2	1	2.6	4.4	2.67
JPHS	6	5.6	5.4	5.67	0.4	0.6	0.8	0.6
Jsteg	4.4	3.4	3.2	3.67	1	0.2	0.4	0.53
Ouguess	4.6	3.6	5.2	4.47	0.2	0.6	1.8	0.87
Steghide	5.6	5	3.8	4.8	0.4	0.6	0.4	0.47

Figure 12. False positive rates for Chen’s detector and POMM detector.

	Average Accuracy %	
	Chen	POMM
SQF = 80		
Cover	88.68	97.08
F5	76.53	90.23
Outguess	75.93	91.08

Figure 13. Overall average detector for Chen and POMM detectors.

To use the histogram information to its best use, we created an SVM that could distinguish between the primary quantization matrix of double compressed images into two groups: greater than 75 (or 80) and less than 75 (or 80). We call this the primary quantization factor splitter, PQF splitter. The PQF splitter is accurate because the histogram signature patterns in the histograms of the quantized DCT coefficients are different when the PQF is smaller than the SQF and when the PQF is greater than the SQF. In order to decide which approach to take, it is necessary to find the relation between PQF and SQF for a particular image. The PQF Splitter classifies an image based on whether the PQF is smaller than or greater than the SQF. Two separate binary classifiers are created for SQF of 75 and 80. For SQF = 75, there are 7 PQFs in the range below 75 (63, 68, ..., 73) and 19 PQFs above 75 (76, 77, ..., 94). We use 15000 training images for each class. The images are further divided into cover, F5 and outguess categories and then subdivided based on the PQF and embedding levels as well. In the same way, a database is created for SQF = 80 case. In this case, there are 13 PQFs below 80 and 14 PQFs above 80. So the division of images changes accordingly. The classification accuracies of the PQF Splitters are shown in Figure 14. The plots on the left show accuracies for SQF = 75 and the plots on the right show accuracies for SQF = 80 case.

Note that when the SQF = 75, Figure 14 shows the detection accuracies are almost always close to 100% over the entire range of PQFs for cover, F5 and outguess. When SQF = 80, the detection accuracies for cover images are close to 100% over the entire range of PQFs. When SQF = 80, the detection accuracies for F5 and outguess images drop when PQF is 79. In this case, the double-compressed image is close to its single-compressed version. For PQFSplitter, the detection accuracies do not vary with the embedding levels.

Cover vs. stego detector. Once the range of PQF is determined by the PQF splitter, we perform cover Vs. stego detection. Depending on whether the PQF of an image is smaller or larger than SQF, we use different methods for classification.

Cover Vs. stego detector for Primary Quality Factor < Secondary Quality Factor. When PQF of an image is less than SQF, the analytical method is used. The method is based on the histogram signature patterns. This approach saves time and computational resources required for the feature extraction process and intensive training of support vector machines. Figure 15 shows the detection accuracy for when the observed quality factor (SQF) is 75 (left column) and for when the observed quality factor is 80 (right column). We observe that the detection accuracies are almost always close to 100% for F5 and outguess. For cover, the detection accuracies are close to 100%. When the PQF is very close to SQF, there is a drop in the detection accuracy.

Cover Vs. stego classifier for Primary Quality Factor > Secondary Quality Factor. We tried three different approaches for this multi-classification problem: Cover Vs. stego binary classifier, multiclassifier, and majority vote classifier. Our results showed that the majority vote multiclassifier worked the best. We summarize the results in Table 4.

Input \ Result	Cover	F5	Outguess	Unknown
Cover	96.76 %	3.05 %	0.14 %	0.05 %
F5	9.48 %	90.34 %	0.09 %	0.09 %
Outguess	0.7 %	0.3 %	98.86 %	0.14 %

Input \ Result	Cover	F5	Outguess	Unknown
Cover	98.35 %	1.55 %	0.08 %	0.02 %
F5	5.6 %	93.93 %	0.3 %	0.17 %
Outguess	0.3 %	0.24 %	99.34 %	0.12 %

Table 4. Confusion matrices for cover Vs. stego detection for $PQF > SQF$ and (a) $SQF = 75$ and (b) $SQF = 80$.

We summarize the results of cover Vs. stego detection for $PQF > SQF$ case in the confusion matrix given in Table 4. We pass test images of known class to the detector and calculate the probabilities of both true and false detection. We observe that the detection accuracies for cover and outguess are above 96%. For F5, 9.5% of the images get misclassified as cover when SQF is 75 and this corresponds to the lowest detection accuracy. The low accuracies for F5 are due to the fact that F5 does not preserve the global statistics of the image. The artifacts of double-compression are thus lost during the process of embedding.

At this point, we have cover Vs. stego detectors for both $PQF < SQF$ and $PQF > SQF$ case. We determine the overall accuracy of the detector by averaging the accuracies of these two detectors over all $PQFs$ and all embedding levels. This gives us the cover Vs. stego detection accuracy of our detector for double-compressed images. In Table 5, we compare these results with those obtained from previous Canvass software. We can clearly see significant rise in the detection accuracies.

	% Detection Accuracy of Canvass				
$SQF = 75$	0.05 bpnz	0.1 bpnz	0.4 bpnz*	Average	FPR
Cover	45.39			45.39	54.61
F5	28.84	34.54	58.29	40.56	
Outguess	42.04	57.73	74.84	58.02	

	% Detection Accuracy of new detectors			
$SQF = 75$	0.05 bpnz	0.1 bpnz	0.4 bpnz	Average
Cover	97.45			97.45
F5	82.28	90.88	95.36	89.51
Outguess	93.91	96.29	90.77	93.67

Table 5. Overall detection accuracy of (a) previous Canvass (b) our detector tested on double-compressed images.

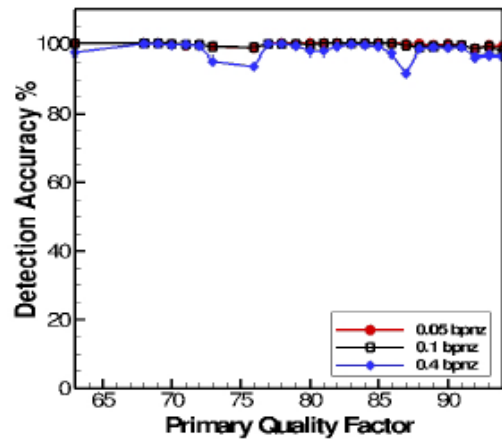
Primary Quality Factor Detector. The last step in the blind steganalysis is estimation of the primary quality factor. It can be used to extract further information regarding the secret payload; although we do not use it for further processing in this work. The signature zero patterns in the histograms are used

for the PQF detection. Separate detectors are created for $SQF = 75$ and $SQF = 80$ case. Figure 16 shows the detection accuracy plots. Again, the plots on the left represent $SQF = 75$ case and plots on the right represent $SQF = 80$ case.

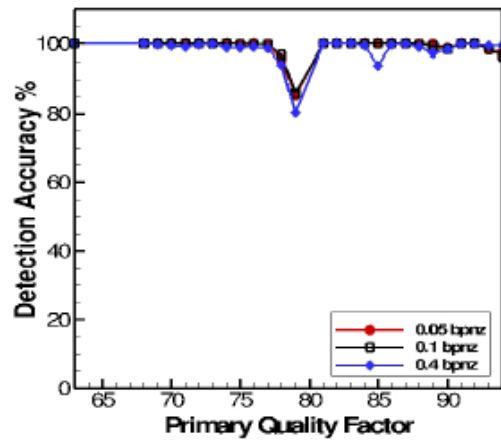
We first discuss the results when $SQF = 75$. These are shown in the left column of Figure 16. The PQF detection accuracies for double-compressed cover images are almost always 100% when $PQF < SQF$. In general, when PQF is less than the SQF, the zero patterns in the histograms are prominent. This leads to high detection accuracies. Double-compressed F5 images with PQF 70 get misclassified as PQF = 71. Out of the nine low-frequency modes considered for the detection, the quantization matrices for quality factors 70 and 71 vary for 2 modes. We do not detect PQF 89. This is because the histogram patterns arising from the

combination of PQFs 88 and 89 with SQF 75 are identical. The quality factor 89 gets detected as 88. Therefore we exclude it from the detection algorithm. In general, the detection accuracies are low when $PQF > SQF$ because the histogram patterns are not very prominent.

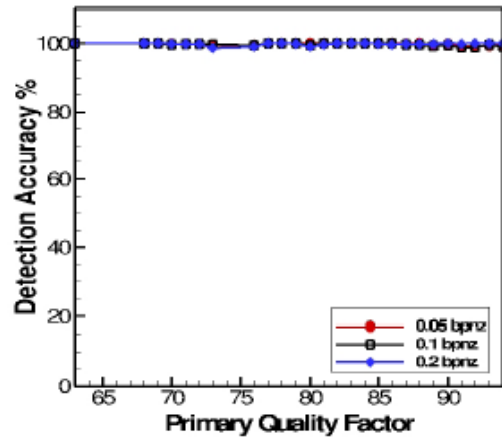
We now discuss the results for $SQF = 80$ case. These are shown in the right column of Figure 16. Similar to the $SQF = 75$ case, the detection accuracies are close to 100% when $PQF < SQF$. PQF 75 gets detected as 74. This is because the values at nine-low frequency modes under consideration are identical for these two quality factors. Therefore we exclude the quality factor 75. Similarly, for F5 and outguess, PQF 85 almost always gets detected as 84 and PQF 88 gets detected as 89. This explains the drops in the detection accuracies for these PQFs. We do not detect PQFs 92 and 93. This is because the histogram patterns arising from the combinations of PQFs 91, 92 and 93 with SQF 80 are identical. The quality factors 92 and 93 get detected as 91. Therefore we exclude these quality factors from the detection algorithm.



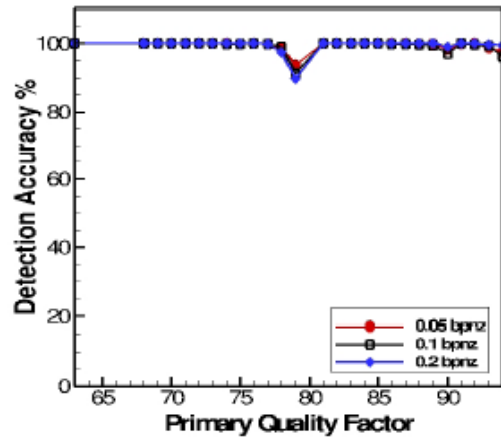
(c) F5 (SQF=75)



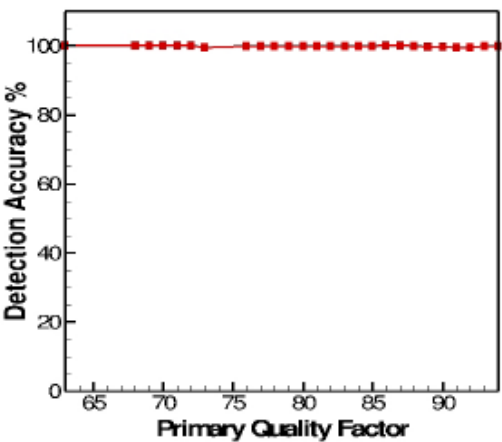
(d) F5 (SQF=80)



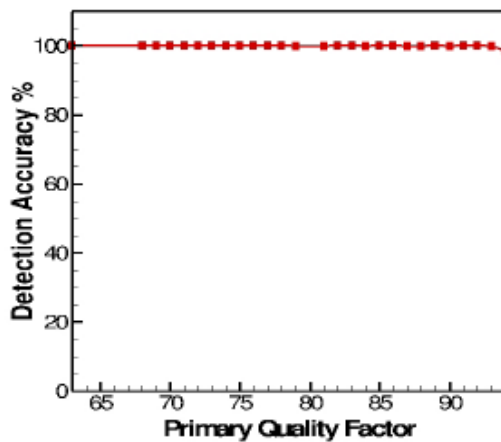
(a) Cover (SQF=75)



(b) Cover (SQF=80)



(a) Cover (SQF=75)



(b) Cover (SQF=80)

Figure 14. Accuracy of PQF Splitters on double-compressed Cover, F5 and Outguess images.

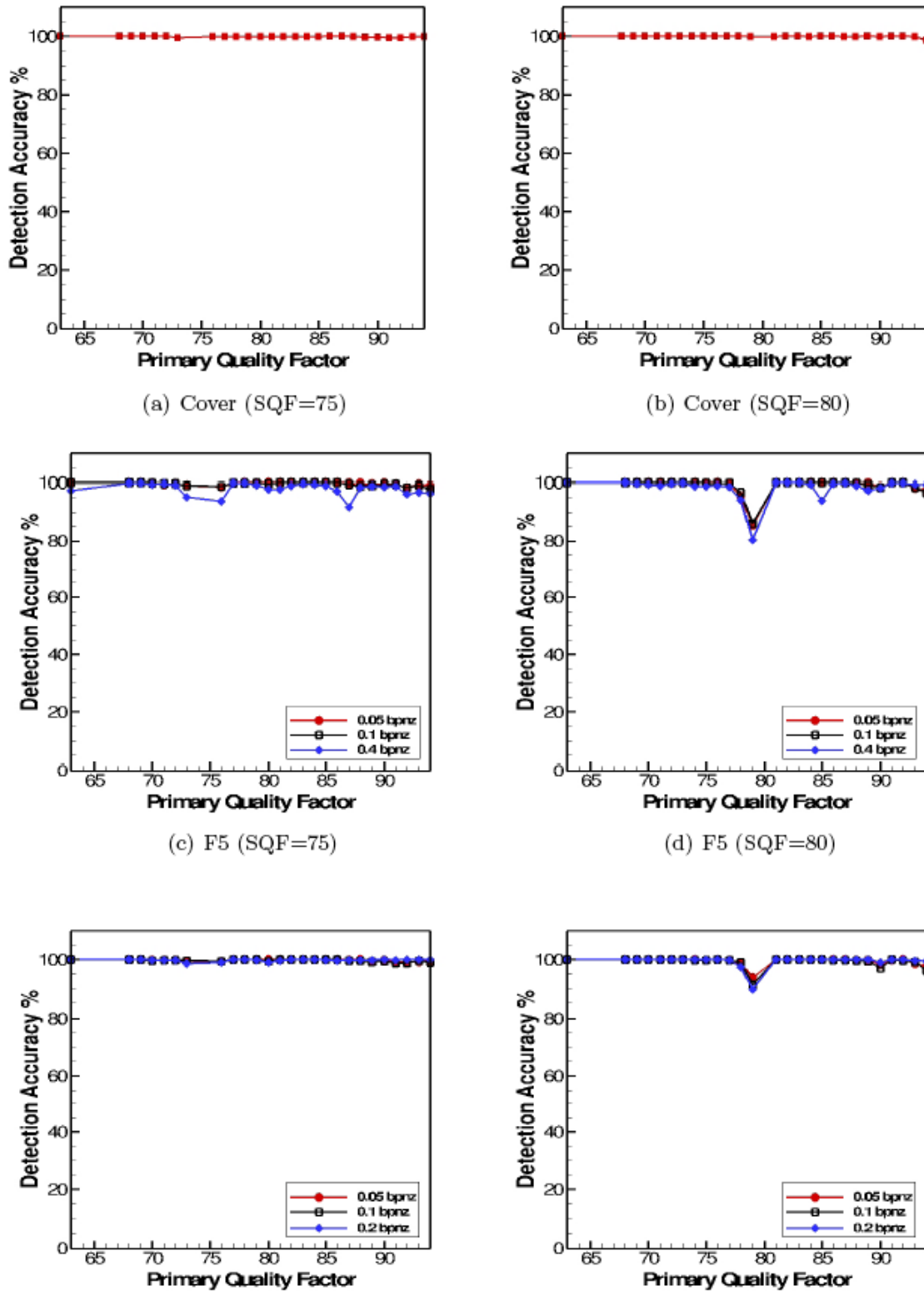


Figure 15. Accuracy of PQF Splitters on double-compressed Cover, F5 and Outguess images.

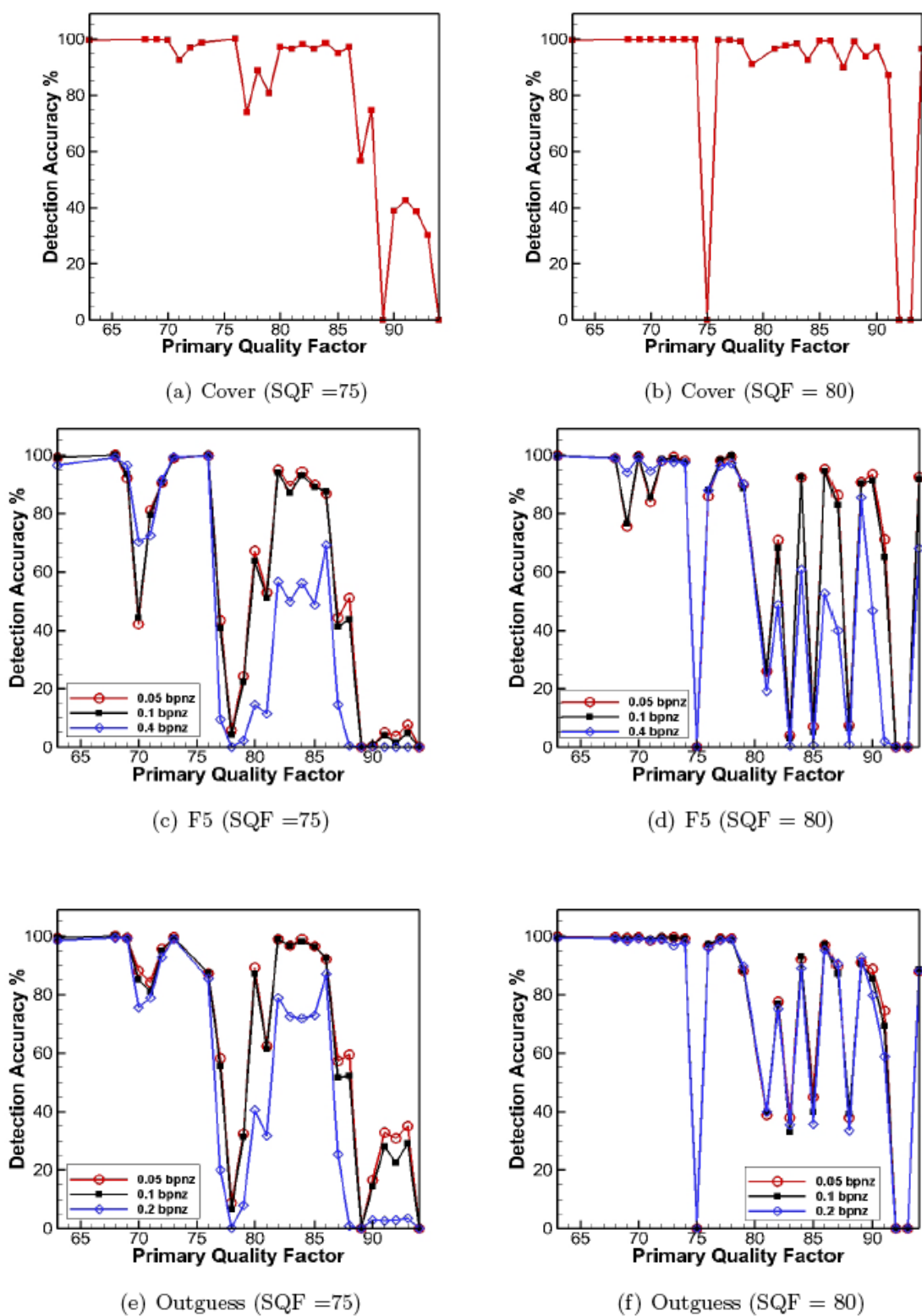


Figure 16. Accuracy of PQF detector.

9. CONCLUSIONS.

In this work, we created a complete steganalyzer system for single as well as double-compressed images. We introduced a new statistical modeling tool to measure the changes caused by various steganographic algorithms as well as by double-compression. We showed that the POMM features perform better than the state-of-the-art double-compression detectors. We also proved the utility of POMMs for solving variety of classification problems such as double-compression detection, PQF splitting and cover Vs. stego detection.

We introduced analytical methods for cover Vs. stego detection and primary quality factor detection. The methods are based on the signature patterns in the histograms of quantized DCT coefficients, as opposed to the other SVM-based classification methods. Each SVM in our experiments is trained on 30000 image data and tested on approximately 1,35,000 image data for different cases. Creating each SVM involves 1 day to extract the training features from 30000 images, 1.5 days of actual SVM training and 2 days for testing. The SVMs-based approach [6] requires 1 SVM for double-compression detector, 159 SVMs for primary quality

step estimation, 3 binary classifiers for stego detection of double-compressed images and 15 SVMs for stego detection of single-compressed images. Thus a total of 178 SVMs are required. Our approach on the other hand uses 1 SVM for double-compression detection, 1 for PQF splitting, 15 SVMs for single-compressed stego detection and 3 SVMs for double-compressed stego detection, which gives a total of 20 SVMs. These novel analytical methods thus save a large amount of time required for feature extraction and intensive SVM training. The histogram pattern features were also used in addition to POMMs to improve the detection accuracies of SVM-based classifiers.

We show that the detection scheme works better if a double-compression detector is used as a pre-classifier. The detection accuracies for double-compressed images improve significantly compared to the previous software. We also compare the performance of the various modules with those presented in [12]. The POMM-based PQF detector has high detection accuracies for $PQF < SQF$. The conditional probabilities given by POMMs describe the relations between inter and intrablock pixels in a JPEG image. In the future, other functions could be investigated to describe other pixel dependencies. Also, currently we limit the steganalysis of single-compressed images to quality factor 75 and that of double-compressed images to secondary quality factors of 75 and 80. But this novel method for steganalysis of double-compressed data looks promising and could be generalized for any combination of primary and secondary quality factors.

10. REFERENCES

1. Stegoarchive. Available online at <http://www.stegoarchive.com>
2. A. Westfeld. F5 A Steganographic Algorithm: High Capacity Despite Better Steganalysis. Information Hiding, 4th International Workshop, I. S. Moskowitz, Ed. Pittsburgh, PA, USA, April 2001, pages 289-302.
3. N. Provos. Outguess: Universal Steganography. <http://www.Outguess.org>, August 1998.
4. Davidson, J., Jalan, J. Canvass - A Steganalysis forensic tool for JPEG images, 2010 ADFSL Conference on Digital Forensics, Security and Law, St. Paul, MN, May 2010.
5. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, 2001.
6. J. Fridrich, M. Goljan and D. Hoge. Steganalysis of JPEG Images: Breaking the F5 Algorithm. 5th Information Hiding Workshop, Noordwijkerhout, The Netherlands, October 2002, pages 310-323.
7. C. Chen, Y. Q. Shi, and W. Su. A machine learning based scheme for double JPEG compression detection. Proceedings IEEE Int. Conf. Pattern Recognition, Tampa, FL. pages 8-11, 2008.

8. J. He, Z. Lin, L. Wang, X. Tang. Detecting doctored JPEG images via DCT coefficient analysis. Proceedings of the 9th European Conference on Computer Vision, Lecture Notes in Computer Sciences, vol. 3953, Springer, Berlin, pages 423-435, 2006.
9. M. Sorell. Conditions for effective detection and identification of primary quantization of re-quantized JPEG images. E-FORENSICS. 1st International ICST Conference on Forensic Applications and Techniques in Telecommunications, Information and Multimedia. ICST, 2008.
10. Z. Lin, J. He, X. Tang, C. Tang. Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis. PR(42), No. 11, pages 2492-2501, November 2009.
11. J. Fridrich, T. Pevny. Determining the Stego Algorithm for JPEG Images, Special Issue of IEE Proceedings-Information Security, 153(3), pp. 75-139, 2006.
12. T. Pevny and J. Fridrich. Detection of double-compression in JPEG images for applications in steganography. IEEE Transactions on Information Forensics and Security, vol. 3, no. 2, pages 247-258, June 2008.
13. J. Fridrich, M. Goljan and D. Hoge. Attacking the outguess. Proc. of the ACM Workshop on Multimedia and Security, Juan-les-Pins, France, December 2002.
14. Siwei Lyu and Hany Farid. Detecting hidden messages using higher-order statistics and support vector machines. Information Hiding, 5th International Workshop, IW 2002, volume 2578 of Lecture Notes in Computer Science, pages 340-354. Springer-Verlag, New York.
15. T. Pevny and J. Fridrich. Merging Markov and DCT features for multi-class JPEG steganalysis. In Proceedings of SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX, San Jose, CA, volume 6505, pages 0304, January 2007
16. Y. Q. Shi, C. Chen, and W. Chen. A Markov process based approach to effective attacking jpeg steganography. In Proceedings of the 8th Information Hiding Workshop, volume 4437 of LNCS, pages 249-264. Springer, 2006.
17. W. Luo, J. Huang, G. Qui. JPEG error analysis and its applications to digital image forensics. IEEE Transactions on Information Forensics and Security, Vol. 5, No. 3, pages 480-491, Sep. 2010.
18. T. Gloe. Demystifying histograms of multi-quantised DCT coefficients. IEEE Conf. on Multimedia and Expo (ICME), pages 1-6, 2011.