Performance study of common image steganography and steganalysis techniques

Mehdi Kharrazi
Polytechnic University
Department of Electrical and Computer Engineering
Brooklyn, New York 11201

Husrev T. Sencar
Nasir Memon
Polytechnic University
Department of Computer and Information Science
Brooklyn, New York 11201

Abstract. *We investigate the performance of state of the art universal steganalyzers proposed in the literature. These universal steganalyzers are tested against a number of well-known steganographic embedding techniques that operate in both the spatial and transform domains. Our experiments are performed using a large data set of JPEG images obtained by randomly crawling a set of publicly available websites. The image data set is categorized with respect to size, quality, and texture to determine their potential impact on steganalysis performance. To establish a comparative evaluation of techniques, undetectability results are obtained at various embedding rates. In addition to variation in cover image properties, our comparison also takes into consideration different message length definitions and computational complexity issues. Our results indicate that the performance of steganalysis techniques is affected by the JPEG quality factor, and JPEG recompression artifacts serve as a source of confusion for almost all steganalysis techniques.* © 2006 SPIE and IS&T. [DOI: 10.1117/1.2400672]

1 Introduction

A range of image-based steganographic embedding techniques have been proposed in the literature, which in turn have led to the development of a large number of steganalysis techniques. The reader is referred to Ref. 1 for a review of the field. These techniques could be grouped into two broad categories, namely, specific and universal steganalysis. The specific steganalysis techniques, as the name suggests, are designed for a targeted embedding technique. These types of techniques are developed by first analyzing the embedding operation and then (based on the gained knowledge) determining certain image features that become modified as a result of the embedding process. The design of specific steganalysis techniques requires detailed knowledge of the steganographic embedding process. Consequently, specific steganalysis techniques yield very accurate decisions when they are used against the particular steganographic technique.

The second group of steganalyzers, universal techniques, were proposed to alleviate the deficiency of specific steganalyzers by removing their dependency on the behavior of individual embedding techniques. To achieve this, a set of distinguishing statistics that are sensitive to wide variety of embedding operations are determined and collected. These statistics, obtained from both the cover and stego images, are then used to train a classifier, which is subsequently used to distinguish between cover and stego images. Hence, the dependency on a specific embedder is removed at the cost of finding statistics that distinguish between stego and cover images accurately and classification techniques that are able to utilize these statistics.

Much research has been done on finding statistics that are able to distinguish between cover and stego images obtained through different embedding techniques.[2–5] Although previous studies report reasonable success on controlled data sets, there is a lack of assessment on how various proposed techniques compare to each other. This is mainly because previous work is limited either in the number of embedding techniques studied or the quality of the data set used in addition to the classification technique employed.

For example, Ref. 5 uses a data set of images consisting of only 1800 images. These images were compressed at the same rate and were of the same size. In Ref. 2, two steganalysis techniques are studied using the same data set of 1800 images. A larger study was done in Refs. 4 and 6, employing 40,000 images with constant size and compression rate, where only one steganalysis technique was investigated. Thus, there is a lack of a study that provides comparative results among a number of universal steganalysis techniques over data sets of images with varying properties, e.g., source, nature, compression level, size, etc. Our goal in this work is twofold: first, to evaluate a range of embedding techniques against the state of the art universal steganalysis techniques, and second, to investigate the effect of

Paper 06110SSR received Jun. 25, 2006; revised manuscript received Sep. 10, 2006; accepted for publication Sep. 12, 2006; published online Dec. 18, 2006. This paper is a revision of a paper presented at the SPIE/IS&T conference on Security, Steganography, and Watermarking of Multimedia Contents, Jan. 2005, San Jose. The paper presented there appears (unrefereed) in SPIE Proceedings Vol. 5681.

image properties on the performance of steganalysis techniques. In this regard, we are interested in answering questions such as

1. What are the impacts of the factors such as size, texture, or source on steganography and steganalysis?
2. How do compression and recompression operations affect the steganalysis performance?
3. Does the image domain used for steganographic embedding have to match with the domain of steganalysis?
4. What are the required computational resources for deploying a steganalyzer?

Some of these questions are inherently hard to answer and are subjects of ongoing research. For example, techniques aimed at reliably determining the source of an image (e.g., digital camera, scanner, computer graphics, etc.) are just emerging and have certain shortcomings.[7,8]

The rest of this paper is organized as follows. We begin by introducing the data set used in our experiments in Sec. 2. Section 3 discusses our experimental setup. Section 4 evaluates a number of discrete cosine transform (DCT)-based embedding techniques. Section 5 discusses the effect of recompression on the performance of steganalyzers. The performances of spatial- and wavelet-based embedding techniques are evaluated in Secs. 6 and 7, respectively. Section 8 discusses the effects of JPEG compression artifacts on spatial and wavelet domain embedding technique. In Sec. 9, we investigate the effect of image texture on the performance of steganalyzers. Issues concerning the poor performance of a wavelet-based steganalyzer,[4] the maximum embedding rate achievable by each embedding technique, and the required computational resources are addressed along with our discussion in Sec. 10.

2 Description of Data Set

One of the important aspects of any performance evaluation work is the data set employed in the experiments. Our goal was to use a data set of images that would include a variety of textures, qualities, and sizes. At the same time, we wanted to have a set that would represent the type of images found in the public domain. Obtaining images by crawling Internet sites would provide us with such data set. Thus, we obtained a list of 2 million JPEG image links from a web crawl. We chose the JPEG image format due to its wide popularity. From this list, we were able to access and download only a total number of 1.5 million images, out of which 1.1 million unique and readable images were extracted. Image uniqueness was verified by comparing SHA1 (secure hash algorithm 1) hashes of all available images. A histogram of total number of pixels in the images is given in Fig. 1(a).

JPEG images are compressed using a variety of quality factors. But since one has a freedom in selecting the quantization table when compressing an image using the JPEG algorithm, there is no standard definition of a quality factor. Therefore, we approximated the quality factor of the images in our data set by deploying the publicly available Jpegdump program.[9] Essentially, Jpegdump estimates the quality factor of the image by comparing its quantization table to the suggested quantization table in the JPEG standard. A histogram of estimated JPEG quality factors is given in Fig. 1(b).

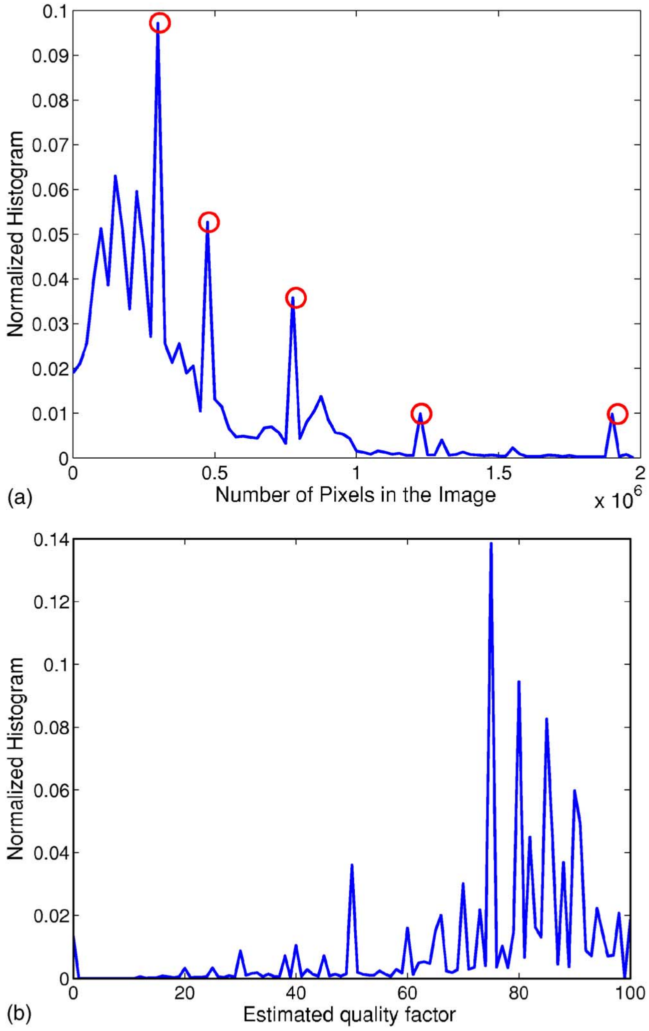


Fig. 1 (a) Normalized histogram of number of pixels in each image, with a bin size of 25,000 pixels. The five main peaks (denoted by circles) correspond to images of size 480 × 640, 600 × 800, 768 × 1024, 1280 × 960, and 1200 × 1600 respectively. (b) Normalized histogram of estimated JPEG quality factors.

Given the variety in size as well as the quality of the images obtained, we decided to break up our data set into a number of categories. Table 1 provides the number of images in each category. We restricted our experiments to the medium-size images with high, medium, and low qualities, where only 100K randomly selected images from among the medium-quality images were used in the experiments. Furthermore, since some of the studied techniques were designed to operate only on gray-scale images (and their color image extensions are the subjects of further study), all images are converted to gray scale by having their color information stripped off. The image size histograms (in number of pixels), as well as the estimated JPEG quality factors are given in Fig. 2.

3 Experimental Setup

Universal steganalyses are composed of two important components. These are feature extraction and feature clas-

Table 1 Cover image data set.

	High (90 to 100)	Medium (75 to 90)	Low (50 to 75)	Poor (50 to 0)
Large (75 K to 2000 K)	74,848	60,060	22,307	10,932
Medium (300 K to 750 K)	54,415	207,774	83,676	31,340
Small (10 K to 300 K)	77,120	301,685	102,770	44,329

sification. In feature extraction, a set of distinguishing statistics are obtained from a data set of images. There is no well-defined approach to obtaining these statistics, but often they are proposed by observing general image features that exhibit strong variation under embedding. The second component, feature classification, operates in two modes. First, the obtained distinguishing statistics from both cover and stego images are used to train a classifier. Second, the trained classifier is used to classify an input image as either being clean (cover image) or carrying a hidden message (stego image). In this context, the three universal techniques studied in this work take three distinct approaches in obtaining distinguishing statistics from images (i.e., feature extraction). These techniques are:

1. *BSM*: Avcibas *et al.*[2,10] considers binary similarity measures (BSMs), where distinguishing features are obtained from the spatial domain representation of the image. The authors conjecture that correlation between the contiguous bit planes decreases after a message is embedded in the image. More specifically, the method looks at seventh and eight bit planes of an image and calculates three types of features, which include computed similarity differences, histogram and entropy related features, and a set of measures based on a neighborhood-weighting mask.
2. *WBS* (wavelet-based steganalysis): A different approach is taken by Lyu and Farid[3,4] for feature extraction from images. The authors argue that most of the specific steganalysis techniques concentrate on first-order statistics, i.e., histogram of DCT coefficients, but simple countermeasures could keep the first-order statistics intact, thus making the steganalysis technique useless. So they propose building a model for natural images by using higher order statistics and then show that images with messages embedded in them deviate from this model. Quadratic mirror filters (QMFs) are used to decompose the image into wavelet domain, after which statistics such as mean, variance, skewness, and kurtosis are calculated for each subband. Additionally the same statistics are calculated for the error obtained from a linear predictor of coefficient magnitudes of each subband, as the second part of the feature set. More recently, in Ref. 6, Lyu and Farid expand their feature set to include a set of phase statistics. As noted in their work, these additional features have little effect on the performance of the steganalyzer. Therefore, we employed only the original set of features as proposed in Ref. 3
3. *FBS* (feature-based steganalysis): Fridrich[5] obtains a set of distinguishing features from DCT and spatial domains. As the the main component of the proposed approach, a simple technique is used to estimate statistics of the original image, before embedding. Estimation is simply done by decompressing the JPEG image, and then cropping its spatial representation by four lines of pixels in both horizontal and vertical directions. Afterward, the image is JPEG recompressed with the original quantization table. The difference between statistics obtained from the given

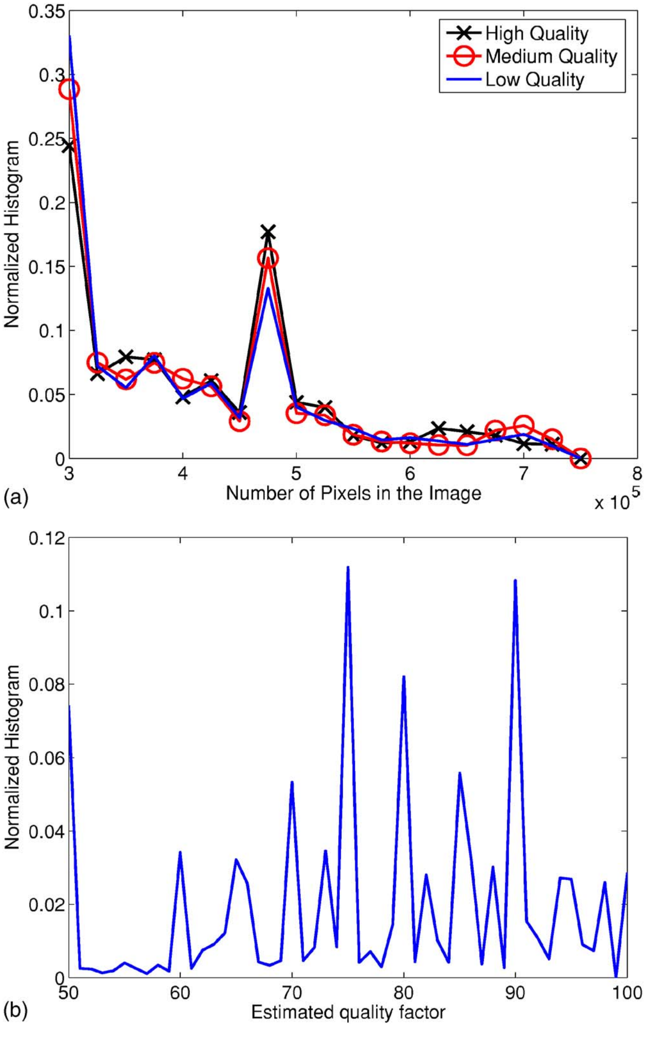


Fig. 2 (a) Normalized histogram of number of pixels in each image, with a bin size of 25,000 pixels, for images in the medium-size categories with high, medium, and low quality factors, and (b) normalized histogram of their estimated JPEG quality factor.

JPEG image and its original estimated version are obtained through a set of functions that operate on both spatial and DCT domains.

All three steganalysis techniques were implemented in the C programming language and verified by comparing test results against those reported by the authors. In the following, we discuss our experimental setup including issues related to embedded message length and the type of classifier used.

Note that the BSM and WBS techniques operate in spatial domain; therefore in the case of JPEG and JPEG2000 images, the images are first decompressed before being fed into the steganalyzer. In the case of the FBS technique, which operates on JPEG images, non-JPEG images are compressed with a quality factor of 100 and then fed in to the steganalyzer, to avoid the steganalyzer detecting different image formats rather than embedding artifacts.

3.1 *Message Size*

When creating the stego data set, we had a number of options in defining the length of the message to be embedded. In essence there are three possible approaches in defining the messages length:

1. Setting message size relative to the number of coefficients that the embedder operates on (i.e., changeable coefficients). This approach guarantees an equal percentage of changes over all images.
2. Setting constant message size. In such an approach, message sizes are fixed irrespective of the image size. As a down side, the data set created with such an approach could contain a set of images that have very few relative changes with respect to their size and images that have maximal changes incurred during the embedding process.
3. Set message size relative to image size. Similar to the preceding, we could have two images of the same size, but with a different number of changeable coefficients.

In creating our data set, we use the first approach in setting the message size as it also takes into account the image (content) itself, unlike the latter two. Note that the number of changeable coefficients in an image does not necessarily indicate the embedding rate achievable by a particular steganographic technique (as discussed in Sec. 10.2). In the following sections, we discuss in more detail the number of changeable coefficients with respect to the image type and the embedding technique.

3.2 *Classifier*

As noted earlier, the calculated features vectors obtained from each universal steganalysis technique are used to train a classifier, which in turn is used to classify between cover and stego images. A number of different classifiers could be employed for this purpose. Two of the techniques more widely used by researchers for universal steganalysis are Fisher's linear discriminate (FLD) and support vector machines (SVMs). SVMs are more powerful, but on the down side, require more computational power, especially if a nonlinear kernel is employed. To avoid high computational cost and to obtain a reasonable success, we have employed a linear SVM (Ref. 11) in our experiments.

To train and test a classifier, the following steps were performed:

1. A random subset of images, 10%, was used to train the classifier. Here, if the two sets of images (i.e., cover and stego) are nonequal, 10% of the smaller set is chosen as the size of the design set.
2. The rest of images (i.e., cover and stego), 90%, were tested against the designed classifier, and decision values were collected for each.
3. Given the decision values, the receiver operating curves (ROCs) curves are obtained.[12]
4. The area under the ROC curve, also known as AUR, was calculated as the accuracy of the designed classifier against previously unseen images.

4 DCT-Based Embedders

DCT domain embedding techniques are very popular due to the fact that DCT-based image format, JPEG, is widely used in the public domain in addition to being the most common output format of digital cameras. Although modifications of properly selected DCT coefficients during embedding will not cause noticeable visual artifacts, they will nevertheless cause detectable statistical changes. Various steganographic embedding methods are proposed, with the purpose of minimizing the statistical artifacts introduced to DCT coefficients. We studied four of these methods, namely Outguess,[13] F5 (Ref. 14), model based,[15] and perturbed quantization[16] (PQ) embedding techniques.

Note that since these techniques modify only nonzero DCT coefficients, message lengths are defined with respect to the number of nonzero DCT coefficients in the images. More specifically we have used embedding rates of 0.05, 0.1, 0.2, 0.4, and 0.6 BPNZ-DCT. In the rest of this section we introduce the results obtained for each of the mentioned embedding techniques.

4.1 *Outguess*

Outguess, proposed by Provos[13] realizes the embedding process in two separate steps. First, it identifies the redundant DCT coefficients that have minimal effect on the cover image, and then depending on the information obtained in the first step, chooses bits in which it would embed the message. Note that at the time Outguess was proposed, one of its goals was to overcome steganalysis attacks that look at changes in the DCT histograms after embedding. Provos, proposed a solution in which some of the DCT coefficients are left unchanged in the embedding process so that following the embedding, the remaining coefficients are modified to preserve the original histogram of the DCT coefficients.

We embedded messages of length 0.05, 0.1, and 0.2 BPNZ-DCT in our cover data set using the Outguess[13] embedding technique. The code for Outguess is publicly available and implemented quite efficiently[17] in C. The performance of the universal steganalysis techniques, in terms of AUR, are given in Fig. 3. As part of the embedding process, the Outguess program, first recompresses the image, with a quality factor defined by the user, and then it uses

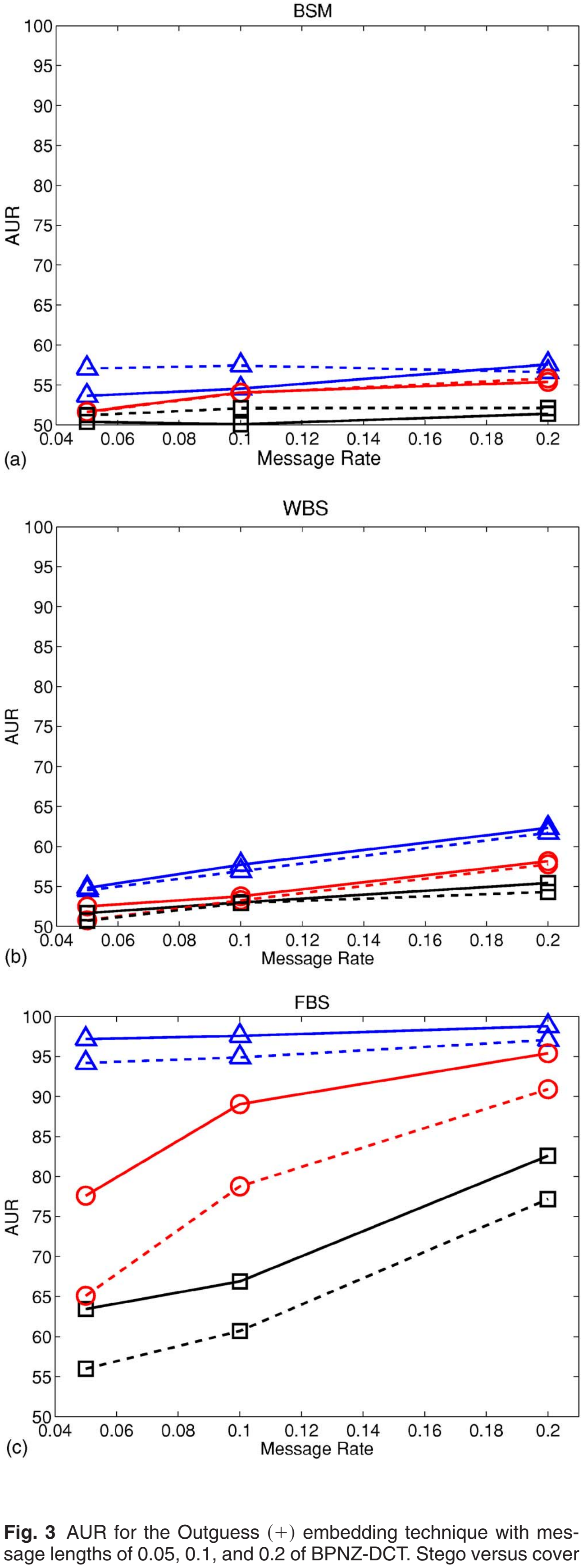


Fig. 3 AUR for the Outguess $(+)$ embedding technique with message lengths of 0.05, 0.1, and 0.2 of BPNZ-DCT. Stego versus cover images are indicated by solid lines, and stego versus recomp-cover are shown with the dashed lines. Actual values are provided in Sec. 12. The symbols □, ○, and △ correspond to high-, medium-, and low-quality images, respectively.

the obtained DCT coefficient to embed the message. To minimize recompression artifacts, we communicated the estimated quality factor of the image to the Outguess program. But a question that comes to mind is whether the steganalyzer is distinguishing between cover and stego images or cover and recompressed cover images. To investigate this question, we also looked at how the steganalysis technique performs when it is asked to distinguish between the set of stego images and recompressed cover images (where the latter is obtained by recompressing the original images using their estimated quality factor). The results obtained are given in Fig. 3.

4.2 *F5*

F5 (Ref. 14) was proposed by Westfeld and embeds messages by modifying the DCT coefficients. (For a review of jsteg, F3, and F4 algorithms that F5 is built on, please refer to Ref. 14.) The most important operation done by F5 is matrix embedding with the goal of minimizing the amount of changes made to the DCT coefficients. Westfeld[14] takes n DCT coefficients and hashes them to k bits, where k and n are computed based on the original images as well as the secret message length. If the hash value equals the message bits, then the next n coefficients are chosen, and so on. Otherwise one of the n coefficients is modified and the hash is recalculated. The modifications are constrained by the fact that the resulting n DCT coefficients should not have a hamming distance of more than $d_{\max}$ from the original n DCT coefficients. This process is repeated until the hash value matches the message bits.

A JAVA implemented version of the F5 code is publicly available. Similar to Outguess, the available implementation of F5 first recompresses the image, with a quality factor input by the user, after which the DCT coefficients are used for embedding the message. We used the quality factor estimated for each image as an input to the F5 code when embedding a message. Messages of length 0.05, 0.1, 0.2, and 0.4 BPNZ-DCT were used to create the stego data set. We have also obtained AUR values on how well the techniques could distinguish between the stego and recompressed images. The results obtained are provided in Fig. 4.

4.3 *Model-Based Embedding Technique*

Unlike techniques discussed in the two previous subsections, the model-based technique, proposed by Sallee,[15] tries to model statistical properties of an image and preserves them during embedding process. Sallee breaks down transformed image coefficients into two parts and replaces the perceptually insignificant component with the coded message bits. Initially, the marginal statistics of quantized (nonzero) ac DCT coefficients are modeled with a parametric density function. For this, a low-precision histogram of each frequency channel is obtained, and the model is fit to each histogram by determining the corresponding model parameters. Sallee defines the offset value of a coefficient within a histogram bin as a symbol and computes the corresponding symbol probabilities from the relative frequencies of symbols (offset values of coefficients in all histogram bins).

At the heart of the embedding operation is a nonadaptive arithmetic decoder that takes as input the message signal and decodes it with respect to measured symbol probabili-

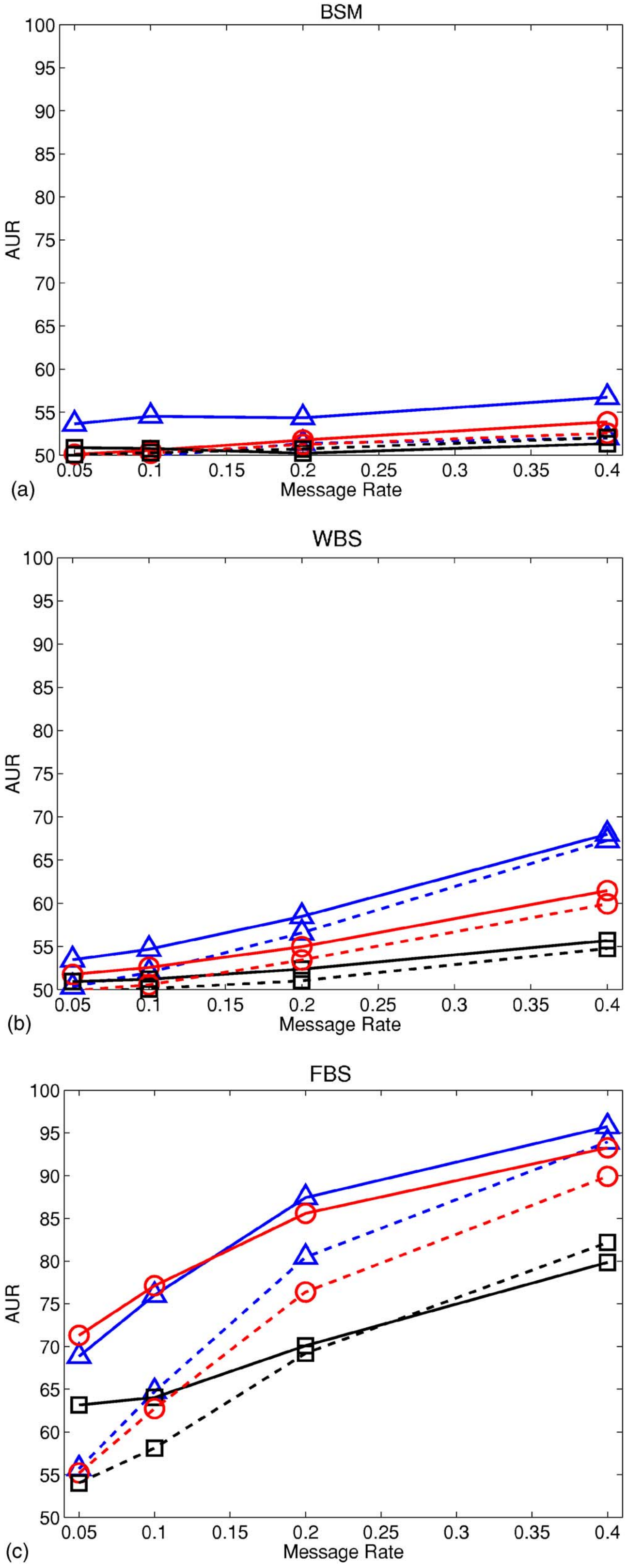


Fig. 4 AUR for the F5 embedding technique with message lengths of 0.05, 0.1, 0.2, and 0.4 of BPNZ-DCT. Stego versus cover images are indicated by solid lines, and stego versus recomp-cover are shown with the dashed lines. Actual values are provided in Sec. 12. The symbols □, ○, and △ correspond to high-, medium-, and low-quality images, respectively.

ties. Then the entropy decoded message is embedded by specifying new bin offsets for each coefficient. In other words, the coefficients in each histogram bin are modified with respect to embedding rule, while the global histogram and symbol probabilities are preserved. Extraction, on the other hand, is similar to embedding. That is, model parameters are determined to measure symbol probabilities and to obtain the embedded symbol sequence (decoded message). (Note that the obtained model parameters and the symbol probabilities are the same both at the embedder and detector.) The embedded message is extracted by entropy encoding the symbol sequence.

Unlike the previous two techniques, the model-based technique does not recompress the image before embedding. Therefore, a comparison of recompressed and stego images does not apply in this case. Although Matlab code is publicly available for this technique, we implemented this technique in C since given our large data set, embedding speed was an important factor. We used message lengths of 0.05, 0.1, 0.2, 0.4, and 0.6 BPNZ-DCT to create our data set. The obtained results are given in Fig. 5.

4.4 *PQ Technique*

Taking a different approach from the previous embedding techniques, Fridrich *et al.*[16] propose the PQ embedding technique in which the message is embedded while the cover image undergoes compression. That is, a JPEG image is recompressed with a lower quality factor, where only selected set of DCT coefficients that could be quantized to an alternative bin with an error smaller than some preset value are modified. The crux of the method lies in determining which coefficients are to be used for embedding so that the detector can also determine the coefficients carrying the payload. For this, the embedder and the detector agree on a random matrix as side information. Essentially, the embedding operation requires solving a set of equations in GF(2) (Galois Fields 2) arithmetic. Finding the solution to the system requires finding the rank of a $k \times n$ matrix, which is computationally intensive. Therefore, to speed up the embedding process, the image is broken into blocks of smaller sizes, and the system is solved independently for each block. This incurs an additional overhead, which must be embedded in each block for successful message extraction.

The PQ technique was the last DCT-based embedding technique we studied. We implemented the code for this technique in C and had a stego data set created with message lengths of 0.05, 0.1, 0.2 and 0.4 BPNZ-DCT. The corresponding steganalysis results are provided in Fig. 6. Similar to previously studied techniques, we determined how the universal steganalyzers perform in distinguishing between recompressed (with quantization steps doubled) and PQ stego images, as given in Fig. 6.

5 Recompression Effect

A good classification-based technique must have a high detection rate, and at the same time, a small false alarm rate. As we illustrated in the previous section, some of the JPEG-based steganographic embedding techniques recompress the JPEG image before embedding the message in them, which may be the cause of false alarms (i.e., classifier misclassifying images because of the recompression ar-

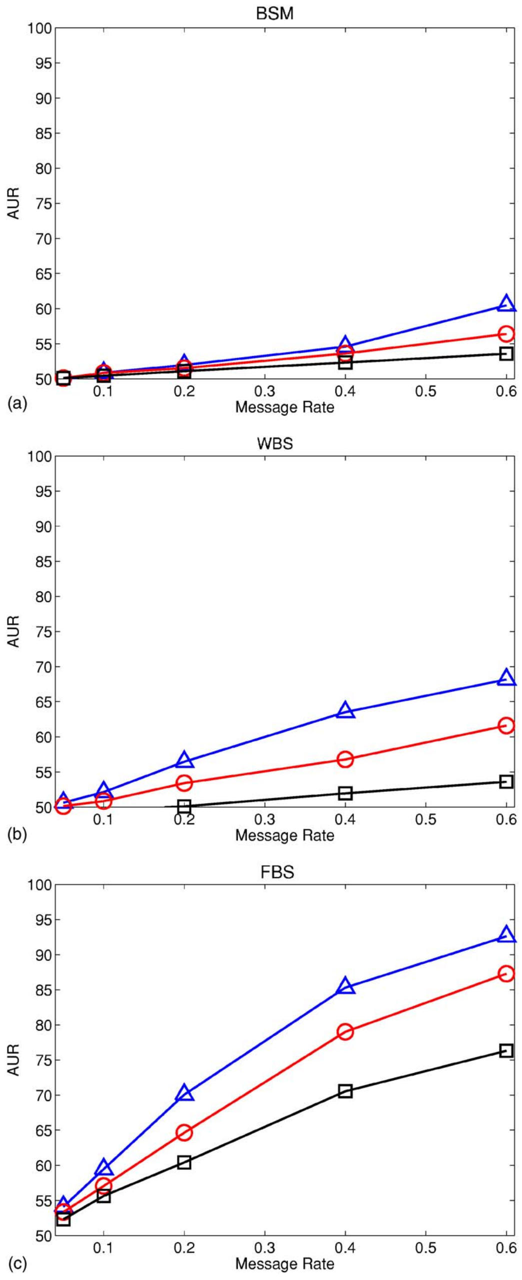


Fig. 5 AUR for the model-based embedding technique with message lengths of 0.05, 0.1, 0.2, 0.4, and 0.6 of BPNZ-DCT. Stego versus cover images are indicted by solid lines, and stego versus recomp-cover are shown with the dashed lines. Actual values are provided in Sec. 12. The The symbols □, ○, and △ correspond to high-, medium-, and low-quality images, respectively.

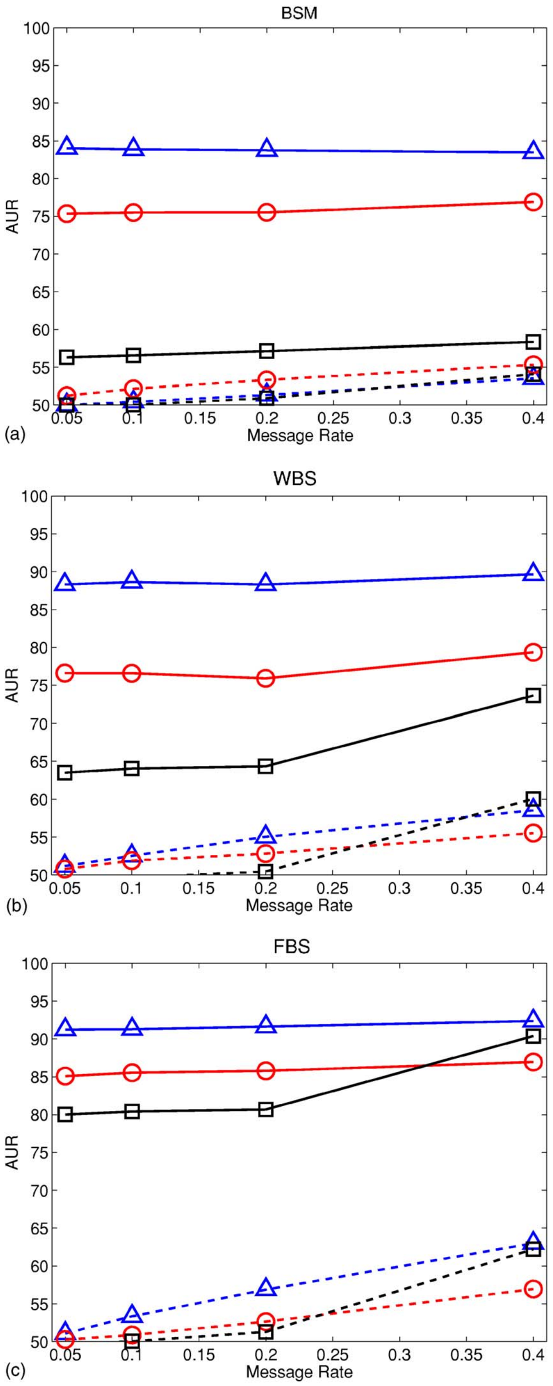


Fig. 6 AUR for the PQ embedding technique with message of lengths of 0.05, 0.1, 0.2, and 0.4 of BPNZ-DCT. Stego versus cover images are indicated by solid lines, and stego versus recomp-cover are shown with the dashed lines. Actual values are provided in Sec. 12. The symbols □, ○, and △ correspond to high-, medium-, and low-quality images, respectively.

tifacts). Thus, we are interested in how the discussed universal steganalysis techniques perform when asked to classify between a set of original cover images and their recompressed versions. We call this procedure the universal steganalysis confusion test. Based on the results in the previous section, there are two cases of interest:

1. Recompressing images with the quality factor estimated from the original image. As evident from Table 2, unlike FBS which confuses recompressed images as stego, BSM and WBS are not able to distinguish between cover and recompressed cover images. This type of recompression was seen with Outguess and F5 embedding techniques.
2. Recompressing images with a quality factor smaller than the original quality factor. More specifically the quantization steps were doubled. In this case, the

Table 2 Effect of the recompression on steganalysis techniques for case 1 and case 2.

	Case 1			Case 2		
	HQ	MQ	LQ	HQ	MQ	LQ
BSM	51.13	50.04	53.17	56.76	74.84	83.93
WBS	51.02	50.55	52.78	63.79	73.56	88.54
FBS	64.54	69.39	64.88	79.93	84.90	91.07

HQ, MQ, and LQ refer to high-, medium-, and low-quality image sets, respectively.

FBS technique is affected most. Note that such a recompression is deployed by the PQ embedding technique.

6 Spatial Domain Embedders

Spatial domain embedding techniques were the first to be proposed in the literature. Their popularity is derived from their simple algorithmic nature, and ease of mathematical analysis. We have studied two least significant bit techniques, LSB and LSB±. In the LSB technique, the LSB of the pixels is replaced by the message bits to be sent. Usually the message bits are scattered around the image. This has the effect of distributing the bits evenly; thus, on average, only half of the LSBs are modified. Popular steganographic tools based on LSB embedding[18–20] vary in their approach for hiding information. Some algorithms change LSB of pixels visited in a random walk, others modify pixels in certain areas of images. Another approach, called LSB±, operates by incrementing or decrementing the last bit instead of replacing it; an example of such approach is used in Ref. 20.

The set of BMP (bitmap) images is obtained by decompressing the images from the three image sets being studied to BMP format. Since all pixels in the image are modifiable, the number of changeable coefficients is equal to the number of pixels in the images. Thus, message lengths of 0.05, 0.1, 0.2, 0.4, and 0.6 bits/pixel were used to create the stego data set, where we had implemented the LSB embedder in C. The obtained results for the LSB technique are in Fig. 7.

The second studied technique was LSB± with which the pixel values are either incremented or decremented by one instead of flipping the pixel's least significant bit. Again using a C implementation, and message lengths as in the LSB case the stego data set was created. Results are shown in Fig. 8. The superior performance of FBS with the LSB and LSB± techniques will be discussed in Section 8.

7 Wavelet Domain Embedding

Wavelet-domain-based embedding is quite new, and not as well developed or analyzed as DCT-based or spatial domain techniques. But such techniques will gain popularity as JPEG2000 compression becomes more widely used. Therefore, we studied a wavelet-based embedding technique called StegoJasper[21] as part of our work. In JPEG2000 compression algorithm, wavelet coefficients are bit plane coded in a number of passes, where, depending on the pass and the importance of the bit value, the bit is either coded or discarded. Using information available to both the encoder and decoder, Su and Kuo first identify a subset of the preserved bits that are used for embedding the secret message. Then, bits are modified while keeping in mind the amount of contribution they make to the reconstructed image at the decoder side. In other words, bits with least level of contributions are modified first, this backward embedding approach minimizes the embedding artifact on the resulting stego image.

To create the JPEG2000 stego data set from our original JPEG data set, we first estimated the bit-rate of each JPEG image (by dividing its file size by the image dimensions in pixels). Then the JPEG images were compressed with a JPEG2000 compressor using the calculated bit rate in order to obtain the cover set. Similarly, JPEG images were fed into a modified JPEG2000 compressor,* to obtain the stego data set. Note that since the least significant bits of selected wavelet coefficients are modified, we define the number of changeable coefficients in this case equal to the number of selectable coefficients. Obtained accuracy results are given in Fig. 9.

8 JPEG Artifacts

In the experimental results, we observed that FBS is able to obtain high accuracy with spatial domain embedding techniques as well, although it was designed exclusively for DCT-based (i.e., JPEG) images. Such results can be explained by considering the fact that the BMP images used in the experiments were obtained from JPEG images, thus baring JPEG compression artifacts. That is, if the BMP image is compressed back to JPEG domain with a quality factor of 100, as we have done in our experiments when feeding non-JPEG images to the FBS technique, the individual DCT histograms will contain peaks centered at the quantization step sizes of the original JPEG image. But if the same BMP image is compressed to a JPEG image, with a quality factor of 100, after LSB or LSB± embedding then the added noise will cause the sharp peaks to leak to neighboring histogram bins. Such a difference is the source of the high accuracy results by the FBS technique.

In fact, a close inspection of the results shows that the performance of the steganalysis techniques varies by the quality factor of the original JPEG images. Thus, we obtained 13,000 gray-scale images, which were downsampled to a size of 640×480 to minimize any JPEG com-

*The StegoJasper code was provided by Dr. Po-Chyi Su and Dr. C.-C. Jay Kuo.

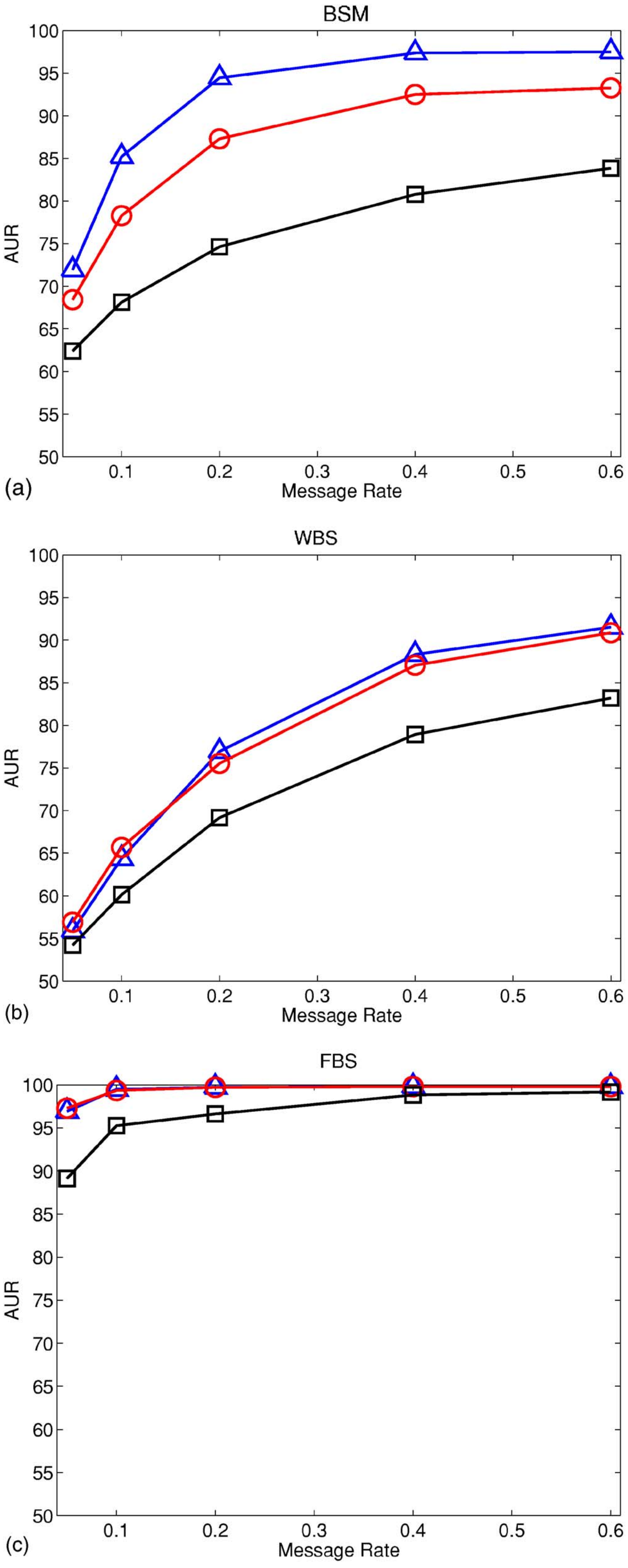


Fig. 7 AUR for the LSB embedding technique, with message lengths of 0.05, 0.1, 0.2, 0.4, and 0.6 of bits/pixels. Actual values are provided in Sec. 12. The symbols □, ○, and △ correspond to high-, medium-, and low-quality images, respectively.

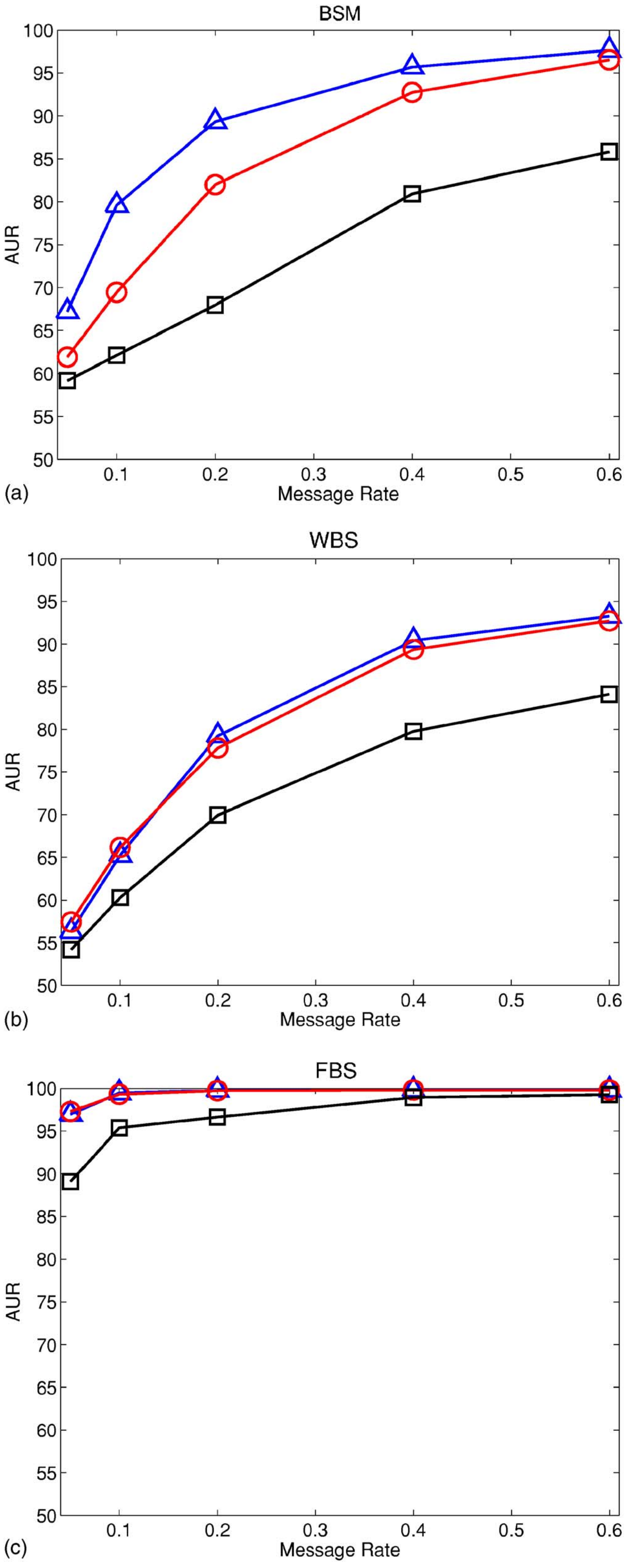


Fig. 8 AUR for the LSB±embedding technique, with message lengths of 0.05, 0.1, 0.2, 0.4, and 0.6 of bits/pixels. Actual values are provided in Sec. 12. The symbols □, ○, and △ correspond to high-, medium-, and low-quality images, respectively.

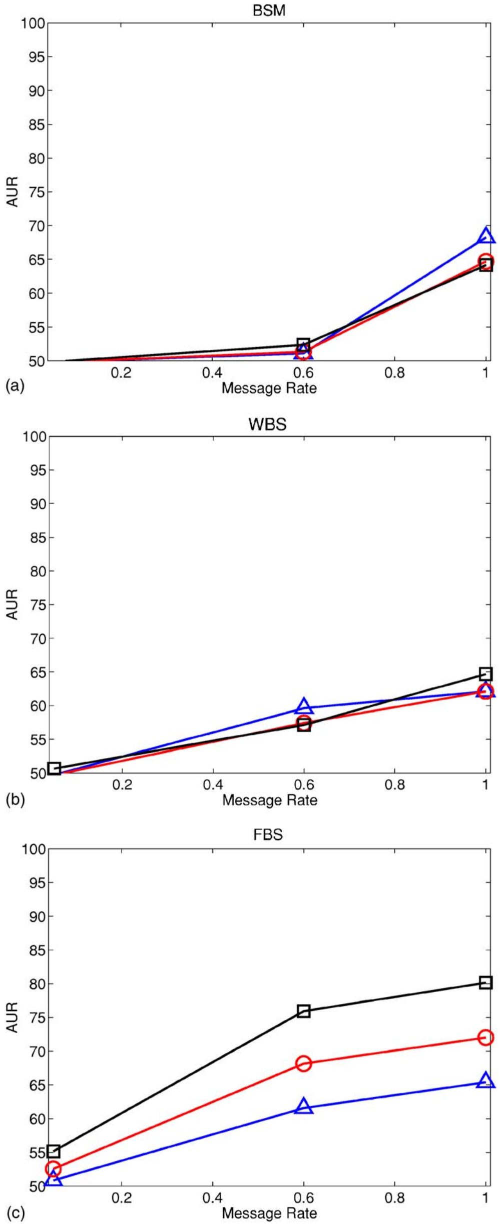


Fig. 9 AUR for the StegoJasper embedding technique with messages lengths of 0.05, 0.6, and 1 of bits/changeable coefficients. Actual values are provided in Sec. 12. The symbols □, ○, and △ correspond to high-, medium-, and low-quality images, respectively.

pression artifacts. Using the LSB embedding technique a stego data set was created using a message length equal to 0.6 bits/pixel. Classifiers were trained for each steganalysis technique using 15% of the data set, and the remaining images were used to test the trained classifier. Interestingly, using a linear classifier, none of the steganalysis techniques were able to obtain acceptable accuracy results. But after using a nonlinear classifier, we were able to obtain good performance results only for the BSM technique. The obtained results are shown in Fig. 10.

Another JPEG-artifact-related phenomenon we observed is that, unlike other techniques studied, in the case of the JPEG2000 embedding technique as the quality of images is decreased, the accuracy of steganalyzer decreases. This could be explained by observing that as the JPEG2000 images are compressed with a lower quality factor, the original JPEG artifacts are minimized making steganalyzers less effective in detecting such stego images. In Fig. 9, we see that in the case of FBS, this effect is maximized.

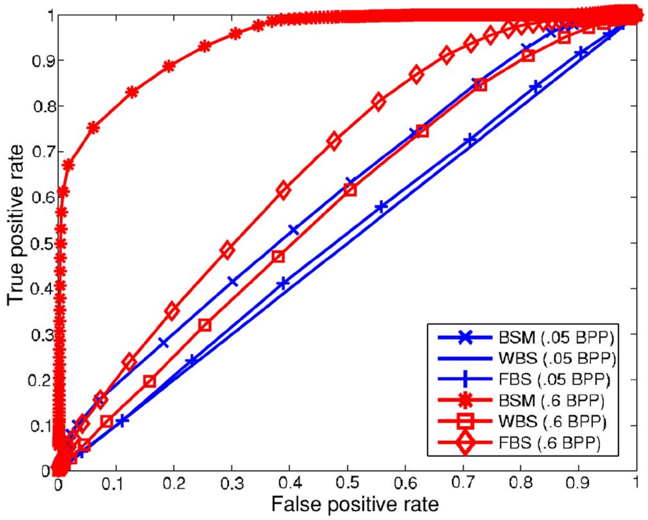


Fig. 10 ROC curves obtained from the studied steganalysis technique against the LSB technique. In this case, the image data set was modified to minimize the JPEG artifacts.

9 Image Texture

In the preceding sections we categorized images with respect to their JPEG quality factor, and observed the effect on the performance of the steganalyzers. But other than the JPEG quality factor, image properties such as image texture could be used to categorize the images. There are many approaches to quantify the texture of an image. A crude measure of image texture would be the mean variance of JPEG blocks. This measure is simple and can be efficiently computed, even with our large data set.

To examine the effect of image texture on steganalysis, we calculate the mean block variance of all the images in our dataset. (The variance is observed to change from 0 to 11,600). Using the mean of the available range, the cover image set was divided into two categories—of high and low variance. Each cover image set was then used to obtain a stego data set, using the model based embedding technique, with message lengths of 0.05, 0.1, 0.2, 0.4 and 0.6 BPNZ-DCT coefficients. The obtained AUR values are displayed in Fig. 11. From the figure we could observe that the performance of the classifier is affected by the variance of the images being used. More specifically, the classifier performs less accurately when confronted with high-variance images (i.e., highly textured or noisy) as expected.

10 Discussion

In this section, we first explain the poor performance of WBS over DCT-based embedding techniques. Then we compare the maximum embedding rate as well as the message lengths over different embedding domains. Last, we note the required computational resources for our experiments.

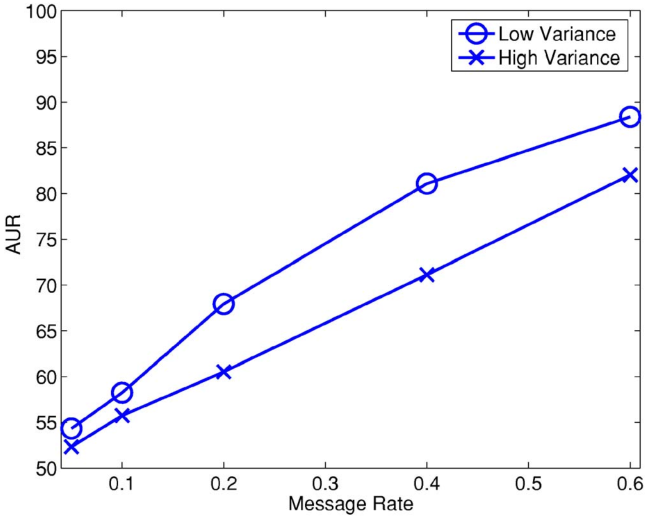


Fig. 11 AUR values obtained for the FBS steganalysis technique against the model-based technique.

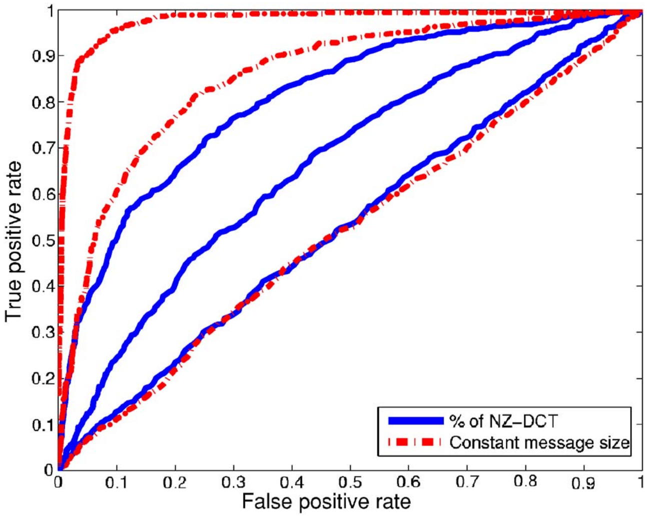


Fig. 12 Effects of message lengths definition on the WBS technique.

10.1 *WBS's Poor Performance*

In the experimental results we have obtained for the WBS technique, we were unable to achieve performance numbers in the same range as reported by Lyu and Farid.[4] We believe that the difference in the performance is due to the following factors:

1. We used a linear SVM as opposed to a nonlinear SVM.
2. Our data set includes images with variety of qualities as well as sizes as opposed to constant quality and size.
3. There are different message length definitions.

It is our understanding that the last point in the preceding list has the largest effect on the results. We did a small experiment to verify this point. As discussed earlier, there are a number of ways to create the stego data set. In Ref. 4 constant message sizes are used to create the stego data set. In accordance with that study, we selected 2000 gray-scale images of size 800×600 with quality of 85 as cover and created a stego data set with Outguess (+) technique.

We defined three message lengths as 1, 5, and 10% of maximum rate, which we defined as 1 bit/pixel. Thus, since all images have constant size in our data set the message lengths used were 600, 3000, and 6000 bytes. Out of 2000 images, we were able to embed into 1954, 1450, and 585 images using messages of size 1, 5, and 10%. Then for each message length a linear SVM classifier was trained using the set of cover images and stego images with that message length, using an equal number of images in the design set. The design set size was set to 40% of the smaller of the two cover and stego data sets. The designed classifier was tested against the remaining images. The resulting ROC curves are given in Fig. 12.

Next we created a stego data set with the message length definition we used in our work, where the message length ranges from 0.05, 0.1, and 0.2 BPNZ-DCT. The number of images in which we were able to embed a message was, respectively, 1948, 1893, and 1786. Note that the difference in message length definition may lead to considerable differences in embedded message lengths, as indicated by the two sets of numbers. For example in Ref. 3, Lyu and Farid report that they were able to embed only into approximately 300 out of 1800 images with the highest embedding rate used in their experiments. Whereas in our experiments, at highest embedding rates (0.2 BPNZ-DCT) we were able to embed into 1786 out of 2000 of the images. Again using the same setup as in the previous case, classifiers were designed and tested. The resulting ROC curves are seen in Fig. 12. As is evident from the obtained results, the classifiers performance changes considerably depending on the message length definition used.

10.2 *Maximum Embedding Rate*

Earlier we stated that our definition of message length is relative to the number of changeable coefficients in image, which is dependent on the embedding technique and the coefficients it used in the process. But in the experiments, we observed that the DCT-based embedding techniques were not able to fully utilize the changeable coefficients available in the images (where changeable coefficients in this case were non-zero DCT coefficients). Thus, we experimentally obtained the maximum embedding rate for each of the four techniques. The corresponding results are given in Fig. 13, where the values obtained for each technique are sorted independently for better visualization. Note that maximum embedding rates obtained are only estimates, and in some cases optimistic. For example, with the PQ technique, we are showing the ratio of changeable coefficient (i.e., coefficients that fall in a small range around the quantization values) over the total number of NZ-DCT coefficients. Actual embedding rate will be lower due to the embedding overhead incurred when splitting the image into smaller blocks to speed up the embedding process. As observed in Fig. 13, the model-based embedding technique is able to best utilize the changeable coefficients in the embedding process over different image quality values, and Outguess comes in as the worst technique in utilizing the changeable coefficients.

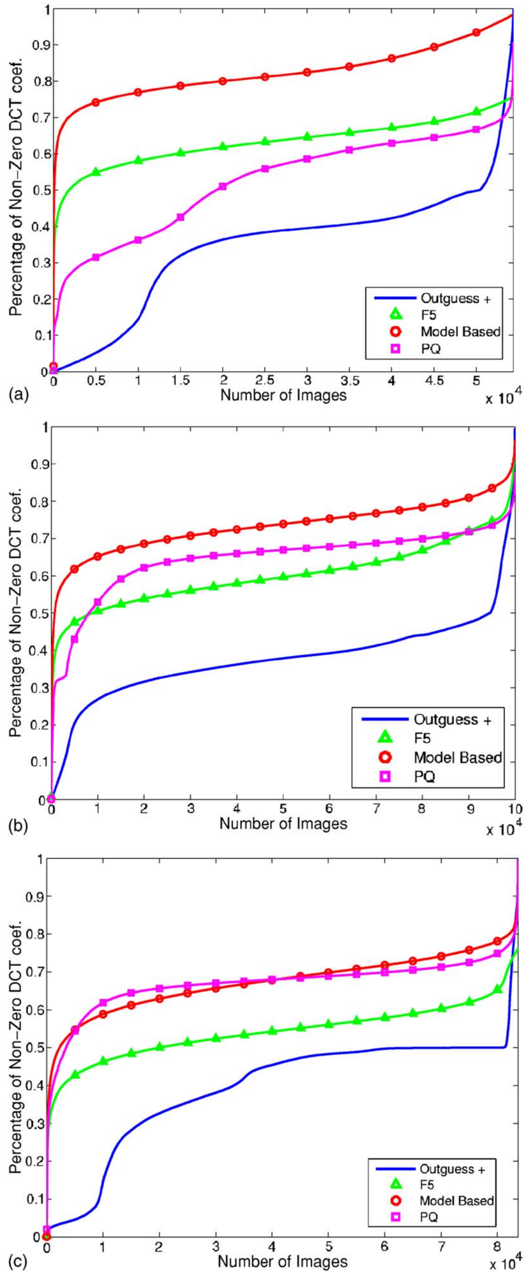


Fig. 13 Maximum embedding rates for DCT-based embedding techniques for (a) high-quality, (b) medium-quality, and (c) low-quality images.

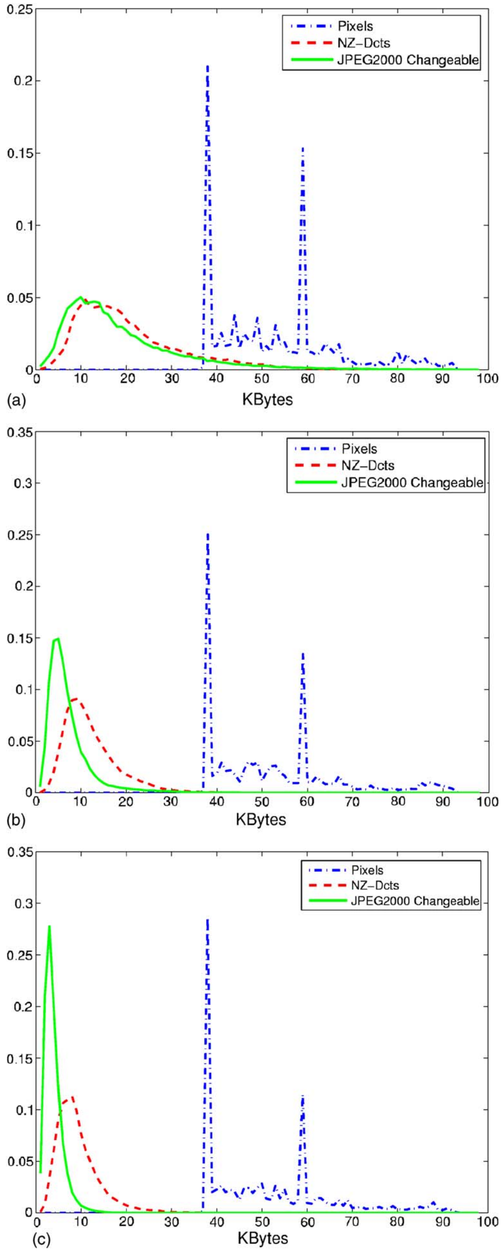


Fig. 14 Histogram of changeable coefficients divided by 8 to get embeddable byte values for (a) high-quality, (b) medium-quality, and (c) low-quality images.

To compare the message lengths that can be embedded by all studied techniques, we first calculated the three different types of changeable coefficients, assuming 1 bit embedding per changeable coefficient, the obtained values are divided by 8 to obtain byte values. The resulting histogram of such values is shown in Fig. 14. We should note that as shown earlier with the DCT based embedding techniques not all changeable coefficients are utilized. For example, with the model based technique on average only 60% of changeable coefficients are utilized. As we see in Fig. 14, spatial domain techniques could carry the largest messages. Also, we observe that StegoJasper is able to carry messages even larger than the DCT-based embedding techniques. We note that we are not considering any detectability constraints here, but merely investigating how well the set of changeable coefficients are utilized by each embedding technique.

10.3 *Computational Resources*

Working with such a huge data set required much processing time. The cover images took about 7 Gbytes of space, and our stego data set had an overall size of 2 Tbytes. Our experiments were done on a Linux box with four Xeon 2.8-GHz processors. In embedding techniques, we found PQ to be the slowest code, taking a few days to embed in the cover data set at the largest embedding rate studied. On the other hand, Outguess was the fastest code, completing the embedding process in about 4 h at the largest message length studied.

With steganalysis techniques we found BSM to be the fastest technique, roughly taking about 3 h to process 100K images. FBS took about 4 h and WBS was the slowest of all taking about 12 h. Note that the processing times we obtained are quite implementation specific, and better performance could potentially be obtained by further optimization of the codes.

11 Conclusion

We investigated the performance of universal steganalysis techniques against a number of stegonagraphic embedding techniques using a large data set of images. Through our work we made a number of observations. The most important are

1. The FBS technique outperforms other studied techniques in this study. Although as we illustrated in Sec. 8, FBS results on spatial domain embedders are affected by the fact that the image sets used in the experiments were originally JPEG compressed. Hence, if true BMP images (i.e., no compression artifacts) are employed then the BSM technique obtains superior performance with spatial domain embedding techniques.
2. The PQ embedding technique is found to be the least detectable technique among the considered techniques in our experiments.
3. JPEG image quality factor affects the steganalyzers performance. Cover and stego images with high-quality factors are less distinguishable than cover and stego image with lower quality.
4. JPEG recompression artifacts confuse all steganalyzers to varying extent. Furthermore, such artifacts also carry over with format conversion (e.g., FBS results with StegoJasper showed dependency on the JPEG quality factor).

This work aimed at answering a number of questions raised in the introduction. However, some of the raised questions are inherently difficult to answer. For example, it is usually argued that images obtained from a scanner or generated through computer graphics will behave differently from high resolution images obtained from a digital camera. However, accurate categorization of images based on their origin (e.g., digital camera, scanned, computer graphics) remains a difficult task. Another question we were not able to resolve was the dependency of the steganalyzer's performance on the size of images. This can be attributed to our data set in which the variation in the image sizes was not significant. However, the detection performance is likely to suffer for smaller images, as the distinctiveness of the collected statistics will reduce. These issues are the subject of further study.

12 Appendix

AUR values obtained from experiments in Secs. 4, 6, and 7 are presented in this Appendix in Tables 3–11.

Table 3 AUR of high-quality images.

	Outguess	F5	Model Based	PQ	
0.05	50.38	50.86	50.11	56.34	BSM
0.05	51.66	50.95	49.61	63.50	WBS
0.05	63.44	63.16	52.31	80.03	FBS
0.1	50.08	50.78	50.44	56.58	BSM
0.1	53.00	51.21	49.64	60.05	WBS
0.1	66.90	64.04	55.65	80.42	FBS
0.2	51.41	50.22	51.10	57.14	BSM
0.2	55.43	52.39	50.10	64.35	WBS
0.2	82.59	70.11	60.42	80.69	FBS
0.4	NA	51.34	52.23	58.35	BSM
0.4	NA	55.68	51.96	73.64	WBS
0.4	NA	79.86	70.54	90.39	FBS
0.6	NA	NA	53.58	NA	BSM
0.6	NA	NA	53.61	NA	WBS
0.6	NA	NA	76.32	NA	FBS

Table 4 AUR for all embedding techniques when compared against cover but recompressed high-quality images.

	Outguess	F5	PQ	
0.05	51.21	50.06	50.00	BSM
0.05	50.72	49.76	49.45	WBS
0.05	55.99	54.04	49.70	FBS
0.1	52.11	50.29	50.03	BSM
0.1	52.91	50.12	49.66	WBS
0.1	60.71	58.12	50.06	FBS
0.2	52.12	50.73	50.91	BSM
0.2	54.32	51.04	50.46	WBS
0.2	77.18	69.22	51.29	FBS
0.4	NA	52.06	54.08	BSM
0.4	NA	54.78	60.05	WBS
0.4	NA	82.19	62.22	FBS

Table 5 AUR for medium-quality images.

	Outguess	F5	Model Based	PQ	
0.05	51.66	50.12	50.11	75.36	BSM
0.05	52.50	51.76	50.14	76.61	WBS
0.05	77.61	71.32	53.35	85.09	FBS
0.1	54.06	50.56	50.85	75.50	BSM
0.1	53.77	52.58	50.85	76.59	WBS
0.1	89.05	77.12	57.06	85.55	FBS
0.2	55.39	51.76	51.53	75.53	BSM
0.2	58.16	54.97	53.41	75.92	WBS
0.2	95.41	85.59	64.65	85.79	FBS
0.4	NA	53.86	53.62	76.90	BSM
0.4	NA	61.46	56.79	79.36	WBS
0.4	NA	93.27	79.01	86.96	FBS
0.6	NA	NA	56.40	NA	BSM
0.6	NA	NA	61.61	NA	WBS
0.6	NA	NA	87.29	NA	FBS

Table 6 AUR for all embedding techniques when compared against cover but recompressed medium-quality images.

	Outguess	F5	PQ	
0.05	51.61	49.94	51.23	BSM
0.05	50.76	49.87	50.79	WBS
0.05	65.10	55.20	50.27	FBS
0.1	53.98	50.23	52.16	BSM
0.1	53.27	50.58	51.90	WBS
0.1	78.77	62.74	50.87	FBS
0.2	55.82	51.25	53.33	BSM
0.2	57.77	53.44	52.82	WBS
0.2	90.91	76.39	52.64	FBS
0.4	NA	52.55	55.34	BSM
0.4	NA	59.94	55.54	WBS
0.4	NA	89.93	56.95	FBS

Table 7 AUR for low-quality images.

	Outguess	F5	Model Based	PQ	
0.05	53.63	53.63	49.87	84.05	BSM
0.05	54.81	53.46	50.63	88.30	WBS
0.05	97.16	68.86	54.11	91.24	FBS
0.1	54.53	54.52	50.87	83.90	BSM
0.1	57.72	54.68	52.14	88.65	WBS
0.1	97.58	76.03	59.46	91.29	FBS
0.2	57.59	54.35	51.97	83.78	BSM
0.2	62.33	58.47	56.46	88.30	WBS
0.2	98.78	87.44	70.07	91.63	FBS
0.4	NA	56.72	54.59	83.48	BSM
0.4	NA	67.99	63.53	89.65	WBS
04	NA	95.75	85.31	92.38	FBS
0.6	NA	NA	60.48	NA	BSM
0.6	NA	NA	68.18	NA	WBS
0.6	NA	NA	92.62	NA	FBS

Table 8 AUR for all embedding techniques when compared against cover but recompressed Low-quality images.

	Outguess	F5	PQ	
0.05	57.08	49.89	50.00	BSM
0.05	54.52	50.33	51.18	WBS
0.05	94.19	55.70	51.08	FBS
0.1	57.45	49.85	50.41	BSM
0.1	56.91	51.99	52.53	WBS
0.1	94.89	64.74	53.35	FBS
0.2	56.61	51.38	51.33	BSM
0.2	61.72	56.59	55.04	WBS
0.2	97.07	80.47	56.88	FBS
0.4	NA	52.00	53.52	BSM
0.4	NA	67.28	58.54	WBS
0.4	NA	93.95	63.00	FBS

Table 9 AUR for LSB embedded images.

	LSB (H)	LSB (M)	LSB (L)	
0.05	62.39	68.42	71.94	BSM
0.05	54.22	56.91	55.90	WBS
0.05	89.13	97.30	96.92	FBS
0.1	68.13	78.28	85.21	BSM
0.1	60.14	65.69	64.40	WBS
0.1	95.26	99.35	99.48	FBS
0.2	74.63	87.30	94.45	BSM
0.2	69.18	75.54	76.94	WBS
0.2	96.62	99.71	99.74	FBS
0.4	80.78	92.50	97.37	BSM
0.4	78.94	87.06	88.33	WBS
0.4	98.33	99.80	99.80	FBS
0.6	83.85	93.27	97.52	BSM
0.6	83.20	90.86	91.52	WBS
0.6	99.18	99.80	99.80	FBS

Here H is high-, M is medium-, and L is low-quality images.

Table 10 AUR for LSB±embedded images.

	LSBP (H)	LSBP (M)	LSBP (L)	
0.05	59.16	61.91	67.21	BSM
0.05	54.17	57.14	56.30	WBS
0.05	89.07	97.30	96.96	FBS
0.1	62.11	69.46	79.60	BSM
0.1	60.29	66.18	65.26	WBS
0.1	95.38	99.31	99.47	FBS
0.2	67.97	81.99	89.34	BSM
0.2	69.95	77.82	79.24	WBS
0.2	96.62	99.73	99.76	FBS
0.4	80.92	92.74	95.68	BSM
0.4	79.77	89.36	90.42	WBS
0.4	98.94	99.80	99.80	FBS
0.6	85.82	96.52	97.64	BSM
0.6	84.10	92.73	93.28	WBS
0.6	99.27	99.80	99.81	FBS

Here H is high-, M is medium-, and L is low-quality images.

Table 11 AUR for StegoJapser embedded images.

	SJ (H)	SJ (M)	SJ (L)	
0.05	49.86	49.80	49.83	BSM
0.05	50.67	49.71	49.74	WBS
0.05	55.14	52.54	50.83	FBS
0.6	52.36	51.32	51.10	BSM
0.6	57.14	57.44	59.62	WBS
0.6	75.93	68.15	61.56	FBS
1	64.15	64.70	68.24	BSM
1	64.70	62.10	62.11	WBS
1	80.15	72.02	65.39	FBS

Here H is high-, M is medium-, and L is low-quality images.

Acknowledgment

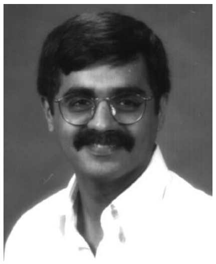
This work was supported by Air Force Research Lab (AFRL) Grant No. F30602-03-C-0091. We would like to thank Ismail Avcibas, Emir Dirik, and Nishant Mehta for coding some of the techniques used, Torsten Suel and Yen-Yu Chen for providing us with a list of crawled image links, and Po-Chyi Su and C.-C. Jay Kuo for providing us with their implementation of StegoJasper.

References

1. M. Kharrazi, H. T. Sencar, and N. Memon, *Image Steganography: Concepts and Practice*, Lecture Notes Series, Institute for Mathematical Sciences, National University of Singapore, Singapore (2004).
2. I. Avcibas, M. Kharrazi, N. Memon, and B. Sankur, "Image steganalysis with binary similarity measures," *EURASIP J. Appl. Signal Process.* **2005**(17), 2749–2757 (2005).
3. S. Lyu and H. Farid, "Detecting hidden messages using higher-order statistics and support vector machines," in *Proc. 5th Int. Workshop on Information Hiding* (2002).
4. S. Lyu and H. Farid, "Steganalysis using color wavelet statistics and one-class support vector machines," *Proc. SPIE* **5306**, 35–45 (2004).
5. J. Fridrich, "Feature-based steganalysis for jpeg images and its implications for future design of steganographic schemes," in *Proc. 6th Information Hiding Workshop*, Toronto (2004).
6. S. Lyu and H. Farid, "Steganalysis using higher order image statistics," *IEEE Trans. Inf. Forens. Secur.* **1**(1), 111–119 (2006).
7. S. Dehnie, H. T. Sencar, and N. Memon, "Digital image forensics for identifying computer generated and digital camera images," in *Proc. Int. Conf. on Image Processing* (2006).
8. S. Lyu and H. Farid, "How realistic is photorealistic?" *IEEE Trans. Signal Process.* **53**(2), 845–850 (2005).
9. http://www.programmersheaven.com/zone10/cat453/15260.htm.
10. I. Avcibas, N. Memon, and B. Sankur, "Steganalysis using image quality metrics," in *Proc. Security and Watermarking of Multimedia Contents*, San Jose, CA (2001).
11. C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," (2001). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
12. T. Fawcett, "Roc graphs: notes and practical considerations for researchers," http://www.hpl.hp.com/personal/Tom_Fawcett/papers/ROC101.pdf.
13. N. Provos, "Defending against statistical steganalysis," in *Proc. 10th USENIX Security Symp.* (2001).
14. A. Westfeld, "F5-a steganographic algorithm: high capacity despite better steganalysis," in *Proc. 4th Int. Workshop on Information Hiding* (2001).
15. P. Sallee, "Model-based steganography," in *Proc. Int. Workshop on Digital Watermarking*, Seoul, Korea (2003).
16. J. Fridrich, M. Goljan, and D. Soukal, "Perturbed quantization steganography with wet paper codes," in *Proc. ACM Multimedia Workshop*, Magdeburg, Germany (2004).
17. B. W. Kernighan and D. M. Ritchie, *The C programming language*, 2nd ed., Prentice Hall, Englewood Cliffs, NJ (1988).
18. F. Collin, Encryptpic, http://www.winsite.com/bin/Info?500000033023.
19. G. Pulcini, Stegotif, http://www.geocities.com/SiliconValley/9210/gfree.html.
20. Toby Sharp, "Hide 2.1," http://www.sharpthoughts.org (2001).
21. P.-C. Su and C.-C. J. Kuo, "Steganography in JPEG 2000 compressed images," *IEEE Trans. Consum. Electron.* **49**(4), 824–832 (2003).

Mehdi Kharrazi received his BE degree in electrical engineering from the City College of New York and his MS and PhD degrees in electrical engineering from the Department of Electrical and Computer Engineering, Polytechnic University, Brooklyn, New York, in 2002 and 2006 respectively. His current research interests include network and multimedia security.

Husrev T. Sencar received his PhD degree in electrical engineering from New Jersey Institute of Technology in 2004. He is currently a postdoctoral researcher with ISIS Laboratory of Polytechnic University, Brooklyn, New York. His research focuses on the use of signal processing approaches to address emerging problems in the field of security with an emphasis on multimedia, networking, and communication applications.



Nasir Memon is a professor in the Computer Science Department at Polytechnic University, New York. His research interests include data compression, computer and network security, multimedia communication, and digital forensics. He has published more than 200 papers in journals and conference proceedings on these topics. He was an associate editor for *IEEE Transactions on Image Processing*, the *Journal of Electronic Imaging*, and the *ACM Multimedia Systems Journal*. He is currently an associate editor for the *IEEE Transactions on Information Security and Forensics*, the *LNCS Transaction on Data Hiding, IEEE Security and Privacy Magazine, IEEE Signal Processing Magazine*, and the *International Journal on Network Security*.