Benchmarking steganographic and steganalysis techniques

Mehdi Kharrazi[a], Husrev T. Sencar[b] Nasir Memon[b]

[a]Department of Electrical and Computer Engineering
[b]Department of Computer and Information Science
Polytechnic University, Brooklyn, NY 11201, USA

ABSTRACT

There have been a number of steganography embedding techniques proposed over the past few years. In turn the development of these techniques have led to an increased interest in steganalysis techniques. More specifically Universal steganalysis techniques have become more attractive since they work independently of the embedding technique. In this work, our goal is to compare a number of universal steganalysis techniques proposed in the literature which include techniques based on binary similarity measures, wavelet coefficients' statistics, and DCT based image features. These universal steganalysis techniques are tested against a number of well know embedding techniques, including Outguess ,[1] F5 ,[2] Model based ,[3] and perturbed quantization .[4] Our experiments are done using a large dataset of JPEG images, obtained by randomly crawling a set of publicly available websites. The image dataset is categorized with respect to the size and quality. We benchmark embedding rate versus detectability performances of several widely used embedding as well as universal steganalysis techniques. Furthermore, we provide a framework for benchmarking future techniques.

1. INTRODUCTION

Steganography refers to the science of "invisible" communication. Unlike cryptography, where the goal is to secure communications from an eavesdropper, steganographic techniques strive to hide the very presence of the message itself from an observer. Although steganography is an ancient subject, the modern formulation of it is often given in terms of the *prisoner's problem*[5] where Alice and Bob are two inmates who wish to communicate a secret message m in order to hatch an escape plan. However, all communication between them is examined by the warden, Wendy, who will put them in solitary confinement at the slightest suspicion of covert communication. In order to communicate the secret message Alice uses a steganographic algorithm to "embed" m into a *cover-object* c, thus obtaining the *stego-object* s. The stego-object s is then sent through the public channel. Wendy will inspects every object past through with the help of steganalysis techniques in order to see of there are any secret messages hidden in the object being transferred. With the wide availability of digital images, and the high degree of redundancy present in them despite compression, there has been an increased interest in using digital images as cover-objects for the purpose of steganography. In turn the development of these techniques and the widespread availability of tools for them have led to an increased interest in steganalysis techniques.

Although all embedding techniques utilize redundancies in the cover image for the embedding process, they differ on their approach, and the image type they operate on. For example more recent techniques such as F5,[2] Outguess,[1] Model based embedding,[3] and Perturbed quantization embedding[4] operate on JPEG images by modifying the DCT coefficients. Although changing the DCT coefficients will cause unnoticeable visual artifices, they do cause detectable statistical changes. These statistical changes are used by steganalysis techniques to detect any embedded messages. Thus each of these techniques tries to minimize these statistical changes with different approaches.

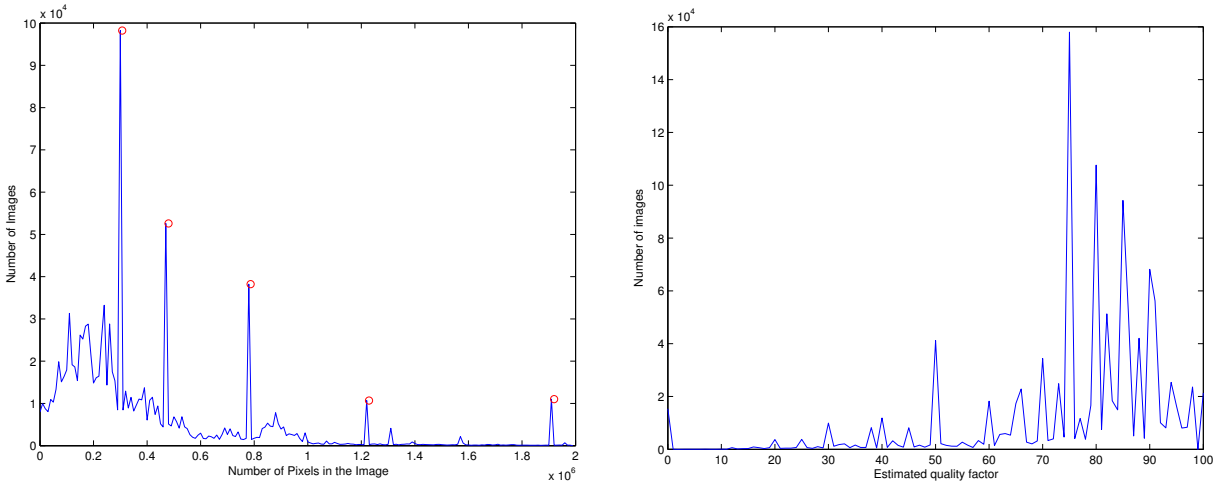
Essentially there are two approaches to the problem of steganalysis, one is to come up with a steganalysis method specific to a particular steganographic algorithm. The other is developing universal steganalysis techniques which are independent of the steganographic algorithm being analyzed. Each of the two approaches has it's own advantages and disadvantages. A steganalysis technique specific to an embedding method would give very good results when tested only on that embedding method, and might fail on all other steganographic algorithms. On the other hand, a steganalysis method which is independent of the embedding algorithm might preform less accurately overall but still provide acceptable results on new and unseen embedding algorithms. For a good survey of image steganography techniques, the reader is referred to.[6]

Given the variety of techniques available for embedding as well as steganalysis, we will compare these methods in the following study. The rest of this paper is organized as follows. In section 2 we introduce the dataset we used in our benchmarking experiments. In section 3 we introduce a number of Universal steganalysis techniques, and discuss their properties. In section 4 we review state of the are embedding techniques, and show their performance against the discussed universal steganalysis techniques. In section 5 we discuss the obtained results and conclude.

2. DATASET

In benchmarking image steganographic techniques, one of the important components is the image dataset employed. Our goal was to use a dataset of images which would include a variety of textures, qualities, and sizes, at the same time we wanted to have a set which would represent the type of images found on public domains. Obtaining images by crawling Internet sites, would provide us with such dataset. Thus we obtained a list of 2 million image links from a web crawl. From this list, a total number of 1.5 million images were downloaded, out of which 1.1 million unique and readable images were extracted. Image uniqueness was verified by comparing SHA1 hashes of all available images. Examining the JPEG headers, about 100 thousand images were found containing EXIF headers, thus being original images captured from a digital camera. The histogram of total number of pixels in the images could be seen in figure 1.

JPEG images also could be compressed using a variety of quality factors. But there is no standard definition for a one to one mapping of quality factor to the quantization table used when compressing an image using the JPEG algorithm. Thus we estimated the quality factor of the images in our dataset using the publicly available Jpegdump program, which estimates the quality factor of the image by comparing it's quantization table to the suggested quantization table in the JPEG standard. A histogram of estimated JPEG quality factors could be seen in figure 1.



(a) (b)

Figure 1. (a) Histogram of number of pixels in each image, with bin size of 10000 pixels. The 5 main peaks (denoted by circles) correspond to images of size 480x640, 600x800, 768x1024, 1280x960 , and 1200x1600 respectively. **(b)** Histogram of estimated JPEG quality factors.

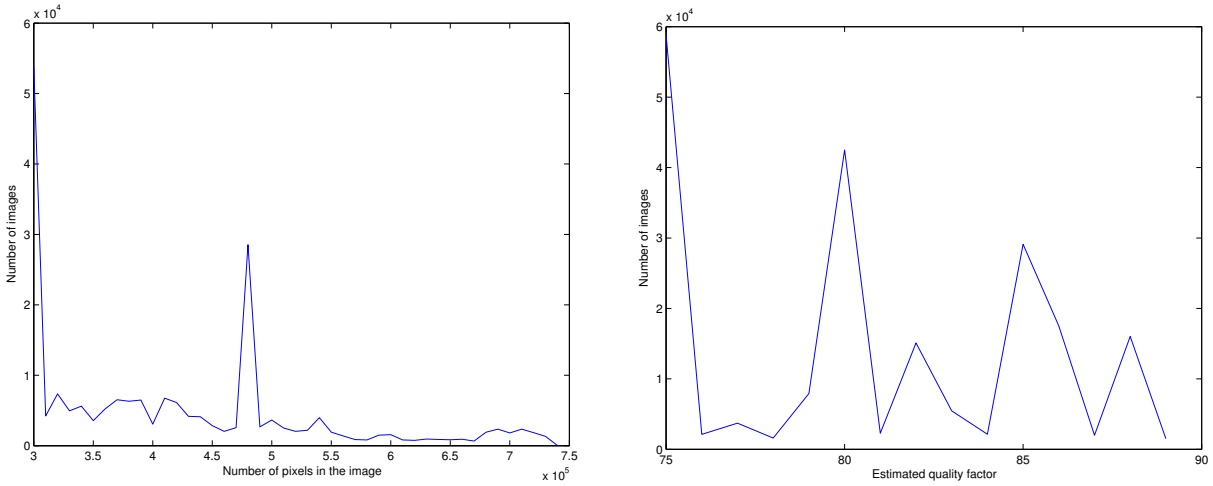
Given the variety in size as well as the quality of the images obtained, we decided to break up our dataset into a number of categories based on these two factors. Table 1 shown the number of images, in each category. For the rest of this paper we have concentrate our efforts in obtaining the benchmarking results for the medium size medium quality images.

Table 1. Cover image dataset.

	High (90-100)	Medium (75-90)	Low (50-75)	poor (50-0)
Large (750K-2000k)	74848	60060	22307	10932
Medium (300K-750K)	58009	207774	83711	31340
Small (10K-300K)	77120	301685	102770	44329

2.1. Medium size, Medium quality images

We started our experiments using 100K images, randomly selected from the images available in this category, with their color information striped off. The image size histograms, in number of pixels, as well as the estimates JPEG quality factors for this category of images could be seen in figure 2.



(a) (b)

Figure 2. (a) Histogram of number of pixels in each image, with bin size of 10000 pixels, for images in the medium size, medium quality category and **(b)** histogram of their estimated JPEG quality factor.

3. UNIVERSAL STEGANALYSIS

Universal steganalysis techniques work by designing a classifier based on a training set of cover-objects and stego-objects obtained from a variety of different embedding algorithms. Classification is done based on some inherent "features" of typical natural images which can get violated when an image undergoes some embedding process. Hence, designing a feature classification based universal steganalysis technique consists of tackling two independent problems. The first is to find and calculate features which are able to capture statistical changes introduced in the image after the embedding process. The second is coming up with a strong classification algorithm which is able to maximize the distinction captured by the features and achieve high classification accuracy. There have been a number Universal steganalysis techniques proposed in the literature, these techniques differ in the features sets they propose for capturing the natural image statistics. For example Avcibas et al.[7] look at seventh and eight bit planes of an image and calculates several binary similarity measures. Farid et al.,[8,9] calculate a number of statistics from the wavelet domain representation of the image after decomposing it with quadratic mirror filters (QMF). Fridrich,[10] looks at statistics directly from the JPEG domain, such as DCT coefficient histograms, as well as spatial domain statistics.

Typically, good features should be accurate, monotonic, and consistent in capturing statistical signatures left by the embedding process. Detection accuracy can be interpreted as the ability of the features to detect the presence of a hidden message with minimum error on average. Similarly, detection monotonicity signifies that the features should ideally be monotonic in their relationship to the embedded message size. Finally,

detection consistency relates to the feature's ability to provide consistently accurate detection for a large set of steganography techniques and image types. This implies that the feature should be independent of the type and variety of images supplied to it. Below we will review 3 techniques which we have studied, we should note that there have been other techniques proposed in the literature ,[11,12] but we chose these 3 techniques, which would highlight different approaches in obtaining the features which will be used in steganalysis.

3.1. BSM

The first technique which we have considered is binary similarity measure (BSM), in which features are obtained from the spatial domain representation of the image. More specifically Avcibas et al.[11] look at seventh and eight bit planes of an image and calculates several features. The authors conjecture that correlation between the contiguous bit planes decreases after a message is embedded in the image. In the rest of this paper we refer to this technique as BSM.

3.2. Wavelet Based

A different approach is taken by Farid et. al[8,9] 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 counter measures 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 form this model. Quadratic mirror filters (QMF) are used to decompose the image into wavelet domain, after which higher order statistics such as mean, variance, skewness, and kurtosis are calculated for each sub-band. Additionally the same statistics are calculated for the error obtained from an optimal linear predictor of coefficient magnitudes of each sub-band, as the second part of the feature set. This technique will be denoted as WBS for the rest of this paper.

3.3. Feature Based

In a more recent work by Fridrich ,[10] a set of distinguishing features are obtained from DCT and spatial domain. But the the main component of her work, is a simple technique which is used to estimate statistics of the original image. Estimation is simply done by decompressing the JPEG image, and then cropping it's spatial representation by 4 pixels. After words the image is re-compressed to JPEG using the original quantization table. The obtained image will have statistical properties very much similar to the original image, before any introduced distortions. The difference between statistics obtained from the given JPEG image, and it's original estimated version are obtained trough a set of functions which operate on both spatial and DCT domain. For the rest of this work we will refer to this technique as FBS.

3.4. Classifier

In all of the above methods, the calculated features are used to train a classifier, which in turn is used to classify cover and stego images. A number of different classifiers could be employed for this purpose. Two of the more widely techniques used by researches for universal steganalysis are fisher linear discriminate (FLD), and support vector machines (SVM). Support vector machines are more powerful, but on the down side, require more computational power, specially if a non-linear kernel is employed. To avoid high computational cost, We have employed a linear SVM [13] in our experiment's.

In order to train and test a classifier, the following steps were preformed:

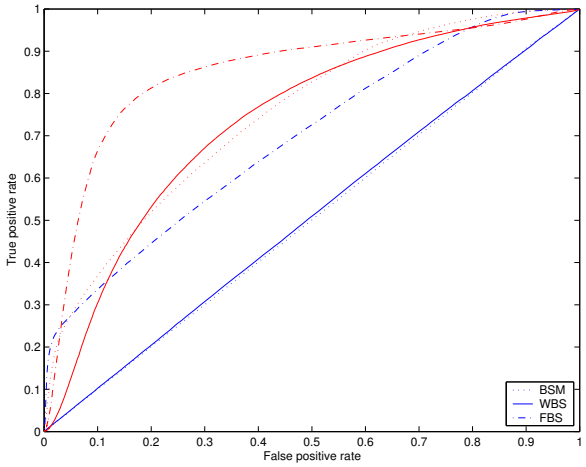
- A random subset of images, 10%, was used to train the classifier. Here if the two sets are non-equal, 10% of the smaller set is chosen as the size of the design set
- The rest of images, 90%, was used against the designed classifier, and decision values were collected for each
- Given the decision values, the ROC curve was obtained. For more on ROC the reader is referred to .[14]
- The area under the ROC curve, also known as AUR, was calculated as the accuracy of the designed classifier against previously unseen images

3.5. Confusion Test

A good classification based technique needs to have high detection rate, at the same time a small false alarm rate. As we will discuss in the next section, some of the steganographic embedding techniques, re-compress the JPEG image, before embedding the message in them. Thus false alarm could be caused by the classifier misclassifying images because of the re-compression artifacts. Thus we are interested in how the discussed Universal steganalysis techniques, preform when asked to classify between a set of original cover images, and their re-compressed versions. We call this procedure the Universal steganalysis confusion test. As it will become more evident in the proceeding section, we are interested in 2 cases of re-compression:

1. Re-compressing image with the quality factor estimated from the original image. As evident from figure 3, unlike FBS which confuses re-compressed images as stego, BSM and WBS are not able to distinguish between cover and re-compressed cover images. This type of re-compression is seen with Outguess and F5 embedding technique which we will discuss in the next section.

2. Re-compressing images with a quality factor smaller than the original quality factor. More specifically the quantization steps were doubled. Here all the techniques are effect although FBS is effected the most. Such re-compression is seen with the PQ technique.



(a)

	1	2
BSM	50.04	74.84
WBS	50.55	73.56
FBS	69.39	84.90

(b)

Figure 3. (a) Effect of the re-compression on steganalysis techniques. Case 1, is denoted by color blue, and case 2, by red. **(b)** The AUR for each case.

4. EMBEDDING TECHNIQUES

As mentioned earlier, we have concentrated our work only on techniques which operate on JPEG images. Most of the work in this category has been concentrated on making use of redundancies in the DCT (discrete cosine transform) domain, which is used in JPEG compression

Although changing the DCT coefficients will cause unnoticeable visual artifices, they do cause detectable statistical changes. In order to minimize statistical artifacts left after the embedding process, different methods for altering the DCT coefficients have been proposed, we will discuss four of these methods, namely Outguess,[1] F5,[2] Model based embedding,[3] and Perturbed quantization embedding.[4]

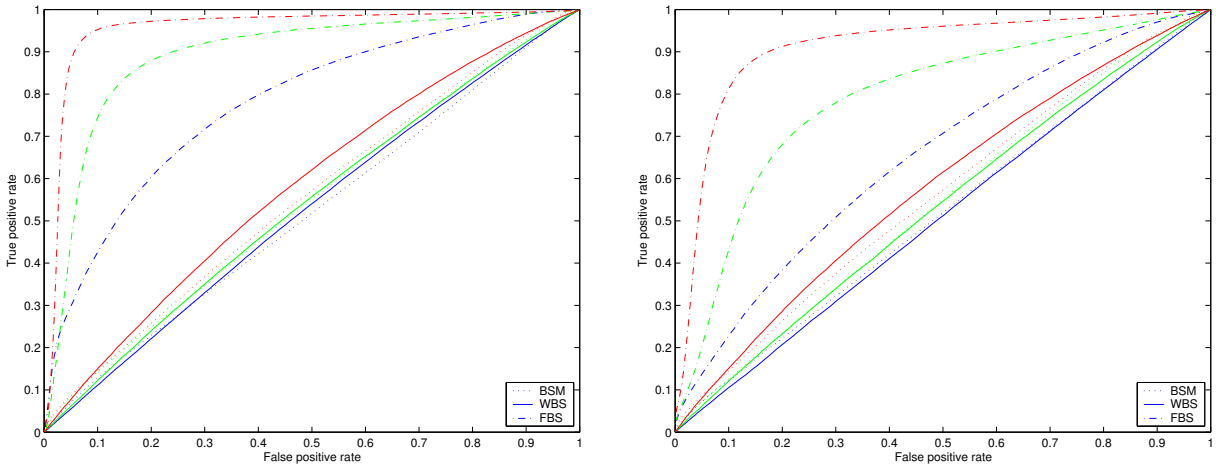
But first we should mention that as part of creating the stego dataset, we had to decide on how to set the message size used in the embedding process. There have been 3 approaches in setting the message size when creating stego datasets:

- Set message size relative to the number of non zero DCT coefficients .[10] Using this approach with embedding techniques, which only utilize non zero DCT coefficients, guarantees a set number of modified coefficients in the data set.
- Set constant message size. In such approach message sizes are fixed irrelative of the image size. As a down side, the dataset created with such approach could contain a set of image which have very few relative changes with respect to their size, and images which have maximal changes incurred during the embedding process.
- Set message size relative to image size. Again here, we could have two images of the same size, but with different number of changeable coefficients.

In creating our data set we have employed the first approach in setting the message size. More specifically we have used embedding rates .05, .1, .2, .4, and .6 BPNZ-DCT (Bits Per Non Zero DCT coefficients) , were there were enough images to embed in. In the rest of this section we will introduce the embedding techniques considered in our experiments.

4.1. Outguess

Outguess,[1] which was proposed by Provos, is an embedding algorithm which embeds messages in the DCT domain. Outguess goes about the embedding process in two separate steps. First it identifies the redundant DCT coefficients which 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. We should note that at the time Outguess was proposed, one of its goals was to overcome steganalysis attacks which look at changes in the DCT histograms after embedding. So Provos, proposed a solution in which some of the DCT coefficients are left unchanged in the embedding process, afterwords these remaining coefficients are adjusted in order preserve the original histogram of DCT coefficients.



(a) (b)

Figure 4. (a) ROC curves for Outguess (+) stego images, against cover images. **(b)** ROC curves for Outguess (+) stego images, against cover but re-compressed images. Blue corresponds to .05, green to .1, red to .2 BPNZ-DCT as message length.

The code for Outguess is publicly available. The code is implemented quite efficiently in C. We have embedded messages of length .05, .1, and .2 BPNZ-DCT in our cover data set. The performance of the universal steganalysis techniques could be seen in figure 4. As part of the embedding process, the Outguess program, first re-compresses the image, to quality factor defined by the user, and then uses the obtained DCT coefficient to embed the message. Here in order to minimize the re-compression artifact we have passed in the estimated quality factor of the image to the Outguess program. Given the confusion test results in the previous section, we were in interested in knowing how well the steganalysis techniques would preform if it was asked to distinguish between

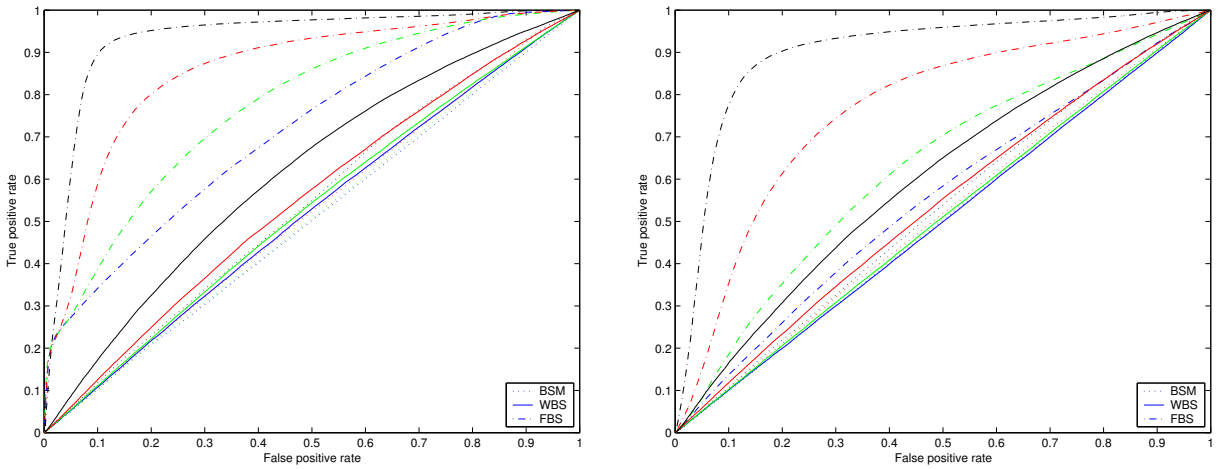
the set of stego images and cover but re-compressed images. Where the re-compressed images are obtained by re-compressing them using the their estimated quality factor. The obtained ROC curves could be seen in figure 4.

4.2. F5

F5[2] embedding algorithm was proposed by Westfeld as the latest in a series of algorithms, which embed messages by modifying the DCT coefficients. For a review of Jsteg, F3 and F4 algorithms that F5 is built on, please refer to.[2]

F5 employs matrix embedding with which it minimizes the amount of change made to the DCT coefficients. Westfeld,[2] takes n DCT coefficients and hashes them to k bits. If the hash value equals to 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. So then given an image, the optimal values for k and n could be selected.

F5 code is also publicly available, the code is written in JAVA. Similar to Outguess, the available implementation of F5, first re-compresses the image, with a quality factor inputed by the user, after which the DCT coefficient are used for embedding the message. Again we have used the quality factor estimated for each image as an input to the F5 code, when embedding a message. Messages of length .05, .1, .2, and .4 BPNZ-DCT were used to create the stego data set. The ROC curves obtained from the steganalysis techniques being studied could be seen in figure 5. We have also obtained ROC curves to see how well the techniques could distinguish between the stego and re-compressed image. Where the re-compressed images are obtained by re-compressing them using the their estimated quality factor. The ROC curves obtained could be seen in figure 5.



(a) (b)

Figure 5. (a) ROC curves for F5 stego images, against cover images. **(b)** ROC curves for F5 stego images, against cover but re-compressed images. Blue corresponds to .05, green to .1, red to .2, black to .4 BPNZ-DCT as message length.

4.3. Model based

The third technique we have studied models some of the statistical properties of the image and tries to preserve these models in the embedding process. Sallee[3] proposes a method which breaks down transformed image coefficients into two parts, and replaces the perceptually insignificant component with the coded message signal. Initially, the marginal statistics of quantized (non-zero) 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

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).

In the heart of the embedding operation is a non-adaptive arithmetic decoder which takes as input the message signal and decodes it with respect to measured symbol probabilities. 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). (It should be noted 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.

Although Matlab code is publicly available for this technique, we have implemented this technique in C. This was needed since, given our large dataset embedding speed was an important factor. Unlike the previous two techniques, Model Based technique, does not re-compress the image, when embedding. We have used message lengths of .05, .1, .2, .4, and .6 BPNZ-DCT create our dataset. The obtained ROC curves could be seen in figure 6.

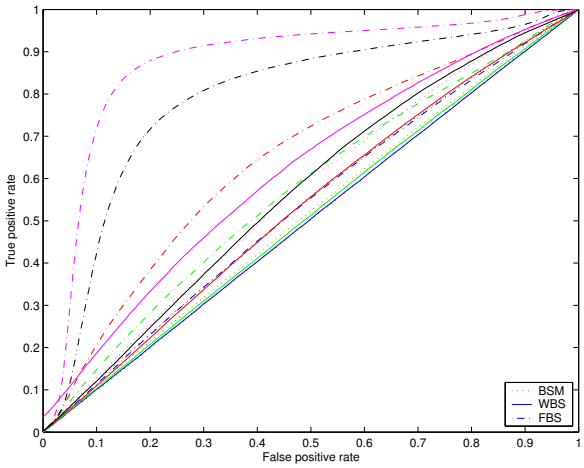


Figure 6. ROC curves for Model Based stego images, vs cover images. Blue corresponds to .05, green to .1, red to .2, black to .4, and magenta to .6 BPNZ-DCT as message length.

4.4. PQ

Fridrich ,[4] proposes an embedding technique called perturbed quantization, in which the cover image goes through an information reducing operation and the message is embedded in this process. For example, a JPEG image is re-compressed with a lower quality factor, but at the quantization step in the compression process, DCT coefficients which could be quantized to an alternate bin with an error smaller than some set value are modified. At the detector, in order to determine which DCT coefficients carry the payload, Fridrich[4] proposes a solution by posing the problem in terms of *writing into memory with defective cells.*

The embedding operation requires solving a set of equations in GF(2) arithmetic. Finding the solution to the system, required finding the rank of a kxn matrix, which is computationally intensive. Thus as proposed by the authors, the image is broken into blocks of smaller size, and the system is solved on a smaller dataset, in order to speed up the embedding process, at the cost of an additional overhead which needs to be embedded for each block.

We implemented the code for this technique in C, and had a stego dataset created with messages lengths of .05, .1, .2 and .4 BPNZ-DCT. The obtained ROC curves obtained when comparing the cover images against the PQ embedded images could be seen in figure 7. Again as with the previous techniques, we were interested in knowing how the universal steganalysis techniques will preform if asked to distinguish between re-compressed (with quantization steps doubled) and stego images, the obtained ROC curves could be seen in figure 7.

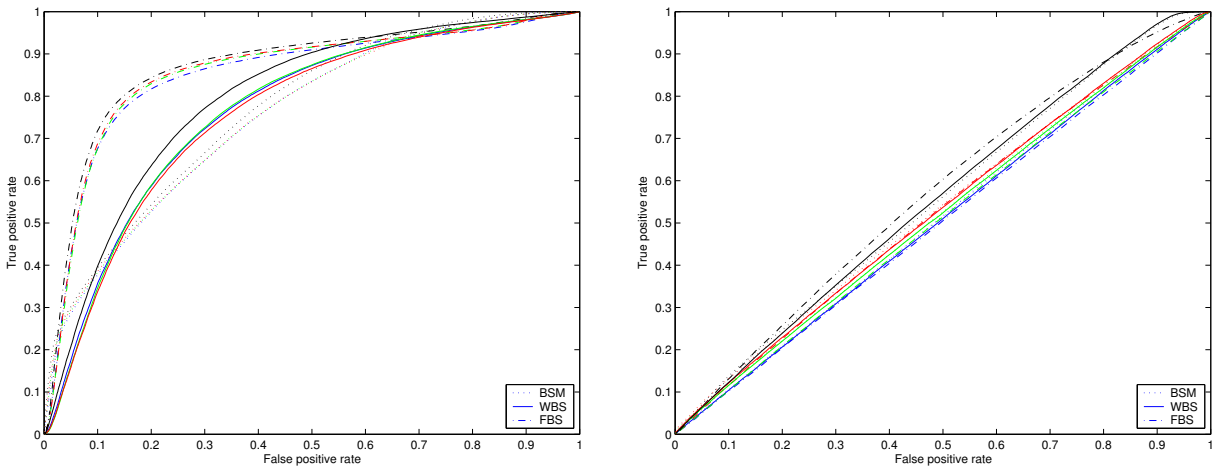


Figure 7. (a) ROC curves for PQ stego images, against cover images. **(b)** ROC curves for PQ stego images, against cover but re-compressed images. Blue corresponds to .05, green to .1, red to .2 BPNZ-DCT as message length.

5. DISCUSSION

5.1. Maximum embedding rate

Although the maximum embedding rate of an steganographic technique is of little importance without a measure of detectability, it does provide us with a sense of how well the embedder utilizes dependencies in the JPEG image. The maximum embedding rate for the embedding techniques studied could be seen in figure 8, where the values obtained for each technique are sorted independently for better visualization. We should note that these values are estimated, and in some cases optimistic. For example with PQ, we are showing the ratio of changeable coefficient, which fall in an ϵ range. Actual embedding rate will be lower do to the embedding overhead incurred when splitting the image into smaller blocks in order to speed up the embedding process.

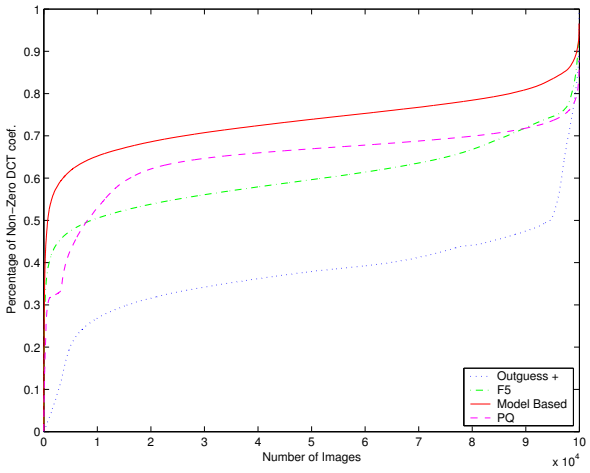


Figure 8. Maximum embedding rate

5.2. Detection rate

As mentioned in the previous sections, we have opted to use the area under the ROC curve, to represent the accuracy for each technique. These numbers are tabulated in table 9. It is evident from the results that FBS

has the best performance among the 3 techniques studied. However, it should be noted that the embedding technique considered in this work operate in DCT domain, and mostly they tend to violate similar regularities in DCT domain (e.g., joint block-wise dependencies between 8×8 blocks, global and marginal histograms). Consequently techniques (such as FBS) that rely on DCT based statistical features are expected to perform better. The performance of FBS on spatial and other transform domain based embedding techniques is the subject of the ongoing research. We should also note that one should expect a better performance from WBS when the classifiers are designed using image sets which are compressed at similar quality factors.

Another steganalyzer design issue raised in our experiments is the identification of the re-compressed images as stego images. All the steganalysis techniques tend to confuse re-compressed images as stego images to different extents. This requires a special handling of images that have undergone multiple compression operations (at different quality factors) prior to steganalysis.

	Outguess	F5	Model Based	PQ	
.05	51.66	50.12	50.11	75.36	BSM
.05	52.50	51.76	50.14	76.61	WBS
.05	77.61	71.32	53.35	85.09	FBS
.1	54.06	50.56	50.85	75.50	BSM
.1	53.77	52.58	50.85	76.59	WBS
.1	89.05	77.12	57.06	85.55	FBS
.2	55.39	51.76	51.53	75.53	BSM
.2	58.16	54.97	53.41	75.92	WBS
.2	95.41	85.59	64.65	85.79	FBS
.4	NA	53.86	53.62	76.90	BSM
.4	NA	61.46	56.79	79.36	WBS
.4	NA	93.27	79.01	86.96	FBS
.6	NA	NA	56.40	NA	BSM
.6	NA	NA	61.61	NA	WBS
.6	NA	NA	87.29	NA	FBS

(a)

	Outguess	F5	PQ	
.05	51.61	49.94	51.23	BSM
.05	50.76	49.87	50.79	WBS
.05	65.10	55.20	50.27	FBS
.1	53.98	50.23	52.16	BSM
.1	53.27	50.58	51.90	WBS
.1	78.77	62.74	50.87	FBS
.2	55.82	51.25	53.33	BSM
.2	57.77	53.44	52.82	WBS
.2	90.91	76.39	52.64	FBS
.4	NA	52.55	55.34	BSM
.4	NA	59.94	55.54	WBS
.4	NA	89.93	56.95	FBS

(b)

Figure 9. (a) AUR for all embedding techniques. **(b)** AUR for all embedding techniques when compared against cover but re-compressed images.

5.3. Computational Resources

Working with such huge dataset, required much processing time. The cover images took about 7GB of space, and our stego dataset had an overall size of 104GB. Our experiments were done on a Linux box with 4 Xeon 2.8GHz processors. In embedding techniques we found PQ to be the slowest code, taking a few days to embed in the cover dataset at the largest embedding rate studied. On the other hand outguess was the fastest code, completing the embedding process in about 4 hours at the largest message length studied.

With steganalysis techniques we found BSM to be the fastest technique, roughly taking about 3 hours to process 100K images. FBS took about 4 hours and WBS was the slowest of all taking about 12 hours. We should note that the processing times we obtained are quite implementation specific, and better performance could be obtained, with a more careful and optimized implementation of the techniques.

5.4. Future plans

We plan to continue our benchmarking experiments on the rest of the available images. The goal is to study the effects, if any, of difference image size and quality clusters on the performance of universal steganalysis techniques. Further more we are looking at ways to combine the steganalysis techniques discussed in this paper in order to obtain better performance. Such task would require us to select a subset of features from each technique which best distinguish between stego and cover images and then merge the selected features.

ACKNOWLEDGMENTS

This work was supported by AFRL Grant No. F30602-03-C-0091. We would like to thank Emir Dirik and Nishant Mehta for coding some of the techniques used, and Prof. Suel and Yen-Yu Chen for providing us with a list of crawled image links

REFERENCES

1. N. Provos, "Defending against statistical steganalysis," *10th USENIX Security Symposium* , 2001.
2. A. Westfeld, "F5a steganographic algorithm: High capacity despite better steganalysis," *4th International Workshop on Information Hiding.* , 2001.
3. P. Sallee, "Model-based steganography," *International Workshop on Digital Watermarking, Seoul, Korea.* , 2003.
4. J. Fridrich, M. Goljan, and D. Soukal, "Perturbed quantization steganography with wet paper codes," *ACM Multimedia Workshop, Magdeburg, Germany, September 20-21* , 2004.
5. G. Simmons, "The prisoners problem and the subliminal channel," *CRYPTO* , pp. 51–67, 1983.
6. M. Kharrazi, H. T. Sencar, and N. Memon, "Image steganography: Concepts and practice," *to appear in Lecture Note Series, Institute for Mathematical Sciences, National University of Singapore* , 2004.
7. I. Avcibas, N. Memon, and B. sankur, "Image steganalysis with binary similarity measures.," *IEEE International Conference on Image Processing, Rochester, New York.* , September 2002.
8. S. Lyu and H. Farid, "Detecting hidden messages using higher-order statistics and support vector machines," *5th International Workshop on Information Hiding.* , 2002.
9. S. Lyu and H. Farid, "Steganalysis using color wavelet statistics and one-class support vector machines," *SPIE Symposium on Electronic Imaging, San Jose, CA,* , 2004.
10. J. Fridrich, "Feature-based steganalysis for jpeg images and its implications for future design of steganographic schemes," *Proc. 6th Information Hiding Workshop, Toronto, Canada, May 23-25* , 2004.
11. I. Avcibas, N. Memon, and B. sankur, "Steganalysis using image quality metrics.," *Security and Watermarking of Multimedia Contents, San Jose, Ca.* , Feruary 2001.
12. W. P. Jeremiah Harmsen, "Kernel fisher discriminant for steganalysis of jpeg hiding methods," *SPIE Electronic Imaging 2004, San Jose Jan. 19-23* , 2004.
13. 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`.
14. T. Fawcett, "Roc graphs: Notes and practical considerations for researchers," `http://www.hpl.hp.com/personal/Tom_Fawcett/papers/ROC101.pdf`.

6. APPENDIX

In the experimental results we have obtained for the WBS technique, we were unable to achieve performance numbers in the same range as suggested by Farid et. al.[9] We believe the performance difference may be due to the following factors:

- We have used a linear SVM as opposed to a non-linear SVM
- Our dataset includes images with variety of qualities as well as sizes as opposed to constant quality and size
- Different message length definitions

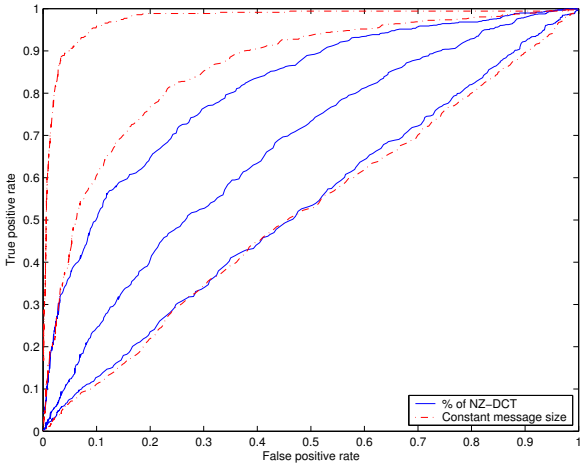


Figure 10.

It is our understanding that the last point in the above list, has the largest effect on the results. We did a small experiment to show this. As pointed out earlier, there are a number of ways to create the stego dataset. In [9] constant message sizes are used to create the stego dataset. Thus using 2000 gray scale images of size 800x600 with quality of 85 as cover, we created a stego dataset with Outguess (+).

We defined 3 message lengths as 1, 5, and 10 percent of maximum rate which we defined as 1 bit per pixel. Thus since all images have constant size in our dataset 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%. Afterward given a message length a linear SVM classifier was trained using the set of cover images and stego images with that message length, using equal number of images in the design set. Design set size was set to 40% of the smaller of the two cover and stego datasets. The designed classifier was tested against the remaining images. The resulting ROC curve could be seen in figure 10.

Next we created a stego dataset with the message length definition we used in our work, were the message length ranges from .05, .1, and .2 BPNZ-DCT. The number of images in which we were able to embed a message in was respectively 1948, 1893, and 1786. It should be noted 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,[8] Farid et al. report that they were able to embed only in to 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 in to 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 could be seen in figure 10. As evident from the obtained results the classifiers performance changes considerably depending on the message length definition used.