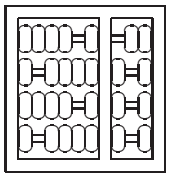

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Progressive Randomization for Steganalysis

Anderson Rocha *Siome Goldenstein*

Technical Report - IC-06-07 - Relatório Técnico

April - 2006 - Abril

Progressive Randomization for Steganalysis*

Anderson Rocha Siome Goldenstein

April 7, 2006

Abstract

In this paper, we describe a new methodology to detect the presence of hidden digital content in the Least Significant Bits (LSB) of images. We introduce the Progressive Randomization (PR) technique that captures statistical artifacts inserted during the hiding process. Our technique is a progressive application of LSB modifying transformations that receives an image as input, and returns n images that only differ in the LSB from the initial image. Each step of the progressive randomization approach represents a possible content-hiding scenario with increasing size, and increasing LSB entropy. We validate our method with 20,000 real, non-synthetic images. Using only statistical descriptors of LSB occurrences, our method already performs better than comparable techniques in the literature.

1 Introduction

Steganography is the art of secret communication. Its purpose is to hide the presence of communication, as opposed to cryptography, which aims to make communication unintelligible to whom do not possess the correct keys [1]. Applications of steganography can include feature location (identification of subcomponents within a data set), captioning, time-stamping, and tamper-proofing (demonstration that original contents have not been altered). However, there are indications that steganography have been used to spread child-pornography pictures in the internet [2, 3].

In this way, it is important to develop algorithms to detect the existence of hidden messages. In this context appears the *digital steganalysis*, that refers to the body of techniques that are devised to distinguish between *non-stego* or *cover-objects*, whose do not contain a hidden message, and *stego-objects* whose that contain a hidden message.

Digital pictures of natural scenes have distinct statistical behavior. With proper statistical analysis, we can determine whether or not an image has been altered, making forgeries mathematically detectable [4]. In this case, the general purpose of steganalysis is to collect sufficient statistical evidence about the presence of hidden messages in images, and use them to classify [5] whether or not a given image contains a hidden message.

*Research supported by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), grants 04/02384-1, and 05/58103-3, CNPq, grant 301278/2004, and FAEPEX-UNICAMP grant 1679.

In general, it is enough to detect whether a message is hidden in a digital content. For example, law enforcement agencies can track access logs of hidden contents to build a network graph of suspects. Later, using other techniques, such as physical inspection of apprehended material, they can uncover the actual contents and apprehend the guilty parties [6].

Among all message embedding techniques, the *Least Significant Bit* (LSB) insertion/modification is considered the most difficult one to detect [7, 8].

Johnson and Jajodia [9] have presented a careful analysis of fingerprints introduced by current steganographic software packages. However, their study was applied only for color indexed images. Westfeld and Pfitzmann [10] have introduced a powerful chi-square steganalytic technique that can detect images with secret messages that are embedded in consecutive pixels. Although, their technique is not effective for raw high-color images and for messages that are randomly scattered in the image. Fridrich et al. [11] have developed a detection method based on close pairs of color created by the process of embedding. However, this approach only works when the number of colors in the images is less than 30 percent of the number of pixels. Fridrich et al [12] have analyzed the capacity for lossless data embedding in the least significant bits and how this capacity is altered when a message is embedded. It is not clear how this approach is sensible to different images given that no training stage was applied.

Compared to previous approaches, Lyu and Farid [13, 14] have designed a more reliable classification technique. Their technique decomposes the image into quadrature mirror filters (QMFs) [15] and hierarchically analyzes the effect of the embedding process.

In this paper, we describe a new methodology to detect the presence of hidden digital content in the LSB fields of images. Our methodology reliably detects messages that are randomly scattered in the image. We introduce the Progressive Randomization (PR) technique to capture the statistical artifacts inserted during the hiding process.

Although in this paper we focused on steganalysis, we have strong experimental evidence that PR approach is appropriate to: (i) detect digitally retouched images; (ii) classify images into categories (indoors, outdoors, people, etc); (iii) perform content based image retrieval; and (iv) detect art forgery.

In the steganalysis context, our technique is a progressive application of LSB modifying transformations that receives an image as input, and returns n images that only differ in the LSB from the initial image. Each step of the progressive randomization approach, represents a possible content-hiding scenario with increasing size, and increasing LSB entropy.

We validate our methodology in a database with 20,000 real, non-synthetic images. Using only statistical descriptors of LSB values occurrences, our method already performs better than all comparable existing techniques in the literature [9–14].

In Section 2, we show how we can use the LSBs of an image to embed a message. In Section 3, we present the chosen statistical descriptors, the feature regions selection, and our progressive randomization detection framework. In Section 5, we present experimental results. Finally, we discuss the conclusions, future work, and extensions in Section 6.

2 LSB steganography background

Among all message embedding techniques, the *least significant bit* (LSB) insertion/modification is the most difficult one to detect [7, 8], and it is imperceptible to humans [7]. A typical color image has three channels: red, green and blue (R,G,B); each one offers one possible bit per pixel to the hiding process. In Figure 1, we show an example on how we can possibly hide information in the LSB fields. Suppose that we want to embed the bits

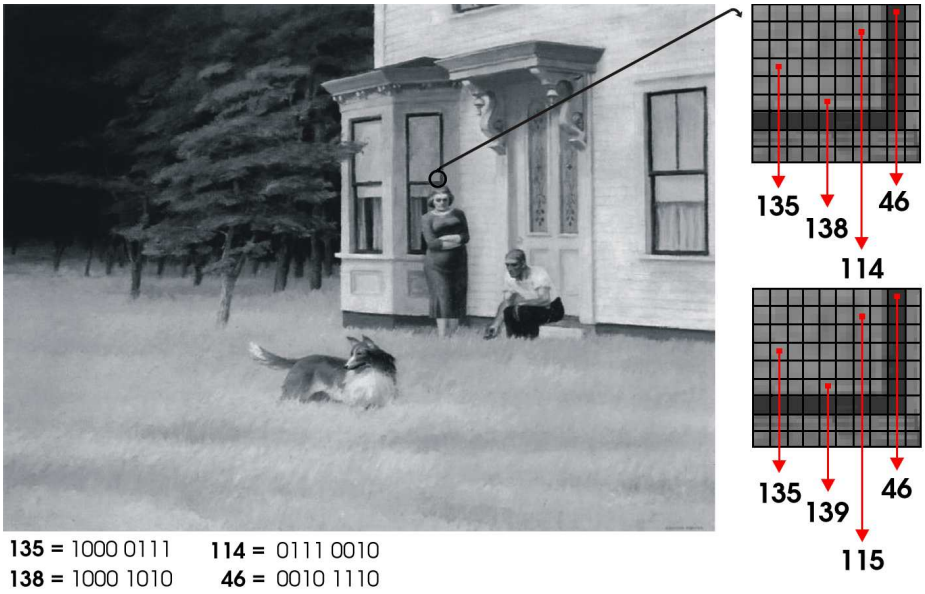

Figure 1: An example of LSB embedding of the bits **1110**.

1110 into the selected area. We have one bit available at each pixel and we need four pixels to hide our bit sequence. In other words, the embedding process consists in the proper modulation of the LSB of selected pixels.

3 Proposed method

In this section, we describe our detection framework. It is a combination of statistical descriptors χ^2 and U_T, feature region selection, and the progressive randomization stage.

3.1 Statistical descriptors

Any possible LSB information hiding procedure will change the contents of a selected number of pixels. This implies in a change of pixel values statistics in a local neighborhood.

An L-bit color channel can represent 2^L possible values. If we split these values into 2^{L-1} pairs which only differs in the LSBs, we are considering all possible patterns of neighboring bits for the LSBs. Each of these pairs are called *pair of value* (PoV) in the sequence [10].

When we use all the available LSB fields to hide a message in an image, the distribution of odd and even values of a PoV will be the same as the 0/1 distribution of the message bits. The idea of the statistical analysis is to compare the theoretically expected frequency distribution of the PoVs with the real observed ones [10]. However, we do not have the original image and thus the expected frequency. In the original image, the theoretically expected frequency is the arithmetical mean of the two frequencies in a PoV. As we know, the embedding function only affects the LSBs, so it does not affect the PoVs distribution after an embedding. Given that, the arithmetical mean remains the same in each PoV, and we can derive the expected frequency through the arithmetic mean between the two frequencies in each PoV.

As presented in [10, 16], we can apply the χ^2 (chi squared-test) and U_T (Ueli Maurer Universal Test) over these PoVs to detect hidden messages. In our detection framework, we extend these two descriptors. The χ^2 test general formula is

$$\chi^2 = \sum_{i=1}^{\nu+1} \frac{(f_{obs} - f_{exp})^2}{f_{exp}}, \tag{1}$$

where ν is the number of analyzed PoVs, f_{obs} and f_{exp} are the observed frequencies and the expected frequencies respectively. For the sake of brevity, we do not present the U_T test here [17].

Previous approaches that use these descriptors can only detect sequential messages hidden in the first available pixels' LSB. Those approaches only consider the descriptors' value, and do not take in account that, for different images, the threshold value for detection may be quite distinct.

Moreover, simply measuring the descriptors constitute a low-order statistic measurement. This approach can be defeated by techniques that maintain basic statistical profiles in the hiding process. We address the low-order statistics problem by looking at the descriptors' behavior along selected regions (feature regions).

3.2 Feature regions selection

Given an image I, we want r regions with size $l \times l$ pixels that have enough information to produce good descriptors[1].

As we are aiming *blind detection* (detection not based in specific steganography tools), our framework must be able to detect random, sequential and point-specific messages embedded in the images.

First, we select four regions that cover the entire image without overlap, Q_{rs} or *Quads regions*[2]. Then, we identify four regions in the image that are rich in detail, H_{rs} or *Harris-regions*. All of them are graphically depicted for a real example in Figure 2. To find the H_{rs} regions, we use a filter as defined by Harris and Stephens [18],

$$H_p = \det(G) - \alpha \ \mathrm{tr}(G)^2, \tag{2}$$

[1] There are hidings that look for regions in the image that are rich in details to reduce the inserted artifacts in the LSB channel [7, 9].

[2] If we use the whole image instead of regions we may not identify messages embedded in specific locations.

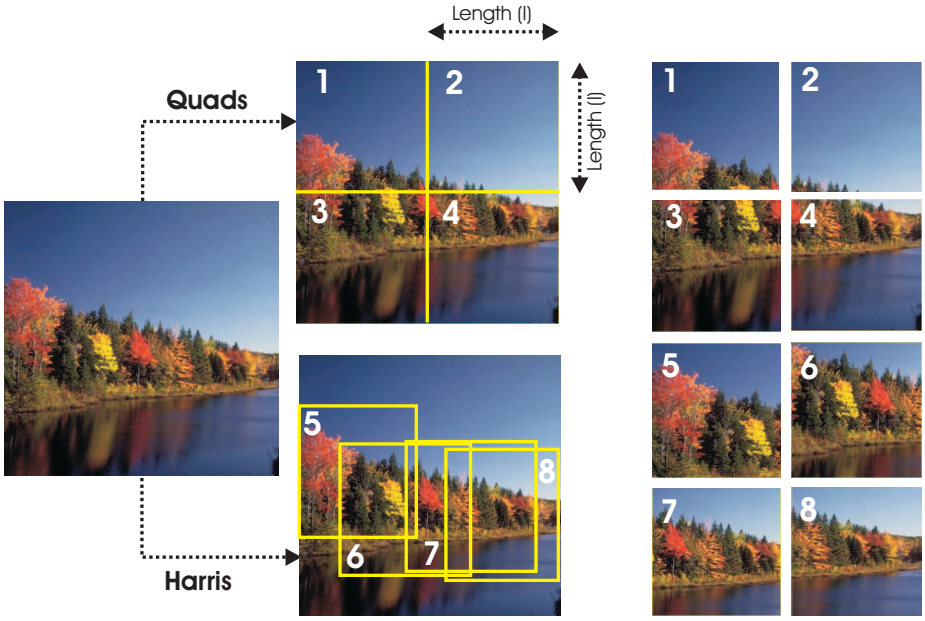

Figure 2: Q_{rs} and H_{rs} feature regions extraction.

where det(.) is the determinant operation, tr(.) is the matrix trace, α is a scale factor, and G is a symmetric 2×2 matrix

$$G = \begin{bmatrix} \sum \nabla_x^2 & \sum \nabla_x \nabla_y \\ \sum \nabla_x \nabla_y & \sum \nabla_y^2 \end{bmatrix}. \tag{3}$$

3.3 Progressive randomization

In this paper, we introduce the Progressive Randomization (PR) methodology that captures statistical artifacts inserted during the hiding process.

The progressive randomization framework is a progressive application of LSB modifying transformations. It receives the original image I as input, and returns n images, which only differ in the LSB from the original image. The output is a direct transformation of the input $O_i = T_i(I)$.

The T_i transformations represent possible hiding processes of messages with different sizes. In our experiments (Section 5), we use $n = 6$ with message sizes[3] 01%, 05%, 10%, 25%, 50%, and 75%. Given that we are detecting contents that are randomly scattered in the image, the greater the message embedding, the greater the value of the LSB entropy.

For the original image and for each generated image, we compute the chosen statistical descriptors values in the selected regions. In our experiments, we have selected eight feature regions and two descriptors (χ^2 and U_T).

[3]A message with size m% is a block of information that uses m percent of the available LSBs.

We need to consider variations in the context of different images. Our descriptors must be robust to these variations. We are interested in the variation rate of our descriptors rather than in their direct values. If we normalize all the measurements taken from the transformations and the original image, we handle this problem. In this way, we get the relative relationship of each step of the progressive randomization and the original image,

$$Norm(O_i) = d_j(O_i)/d_j(I), \tag{4}$$

where d denotes a descriptor of an image taken in a region $1 \leq j \leq k$. In our approach, the descriptor d can be χ^2 or U_T. Figures 3(a-b) show the behavior of our descriptors along the progressive randomization approach of an image. The embedding process we consider is based on a random position key that tells us where to embed each bit. Also, if you are hiding a message, you either encrypt or compress it beforehand, which lends the message pseudo-random with high entropy. In this case, the greater the embedded message (x-axis) the greater the normalized measured descriptor (y-axis) value. The modified bits change a local region even in the cases when we use extra methods to globally preserve statistics.

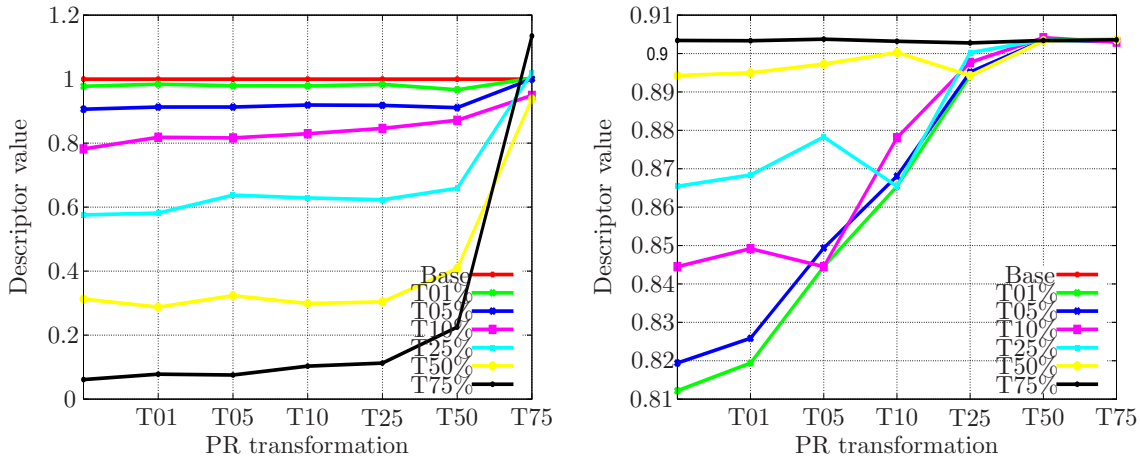

Figure 3: Normalized descriptor's behavior Q_1 region of Figure 2 along the progressive randomization. (a) χ^2. (b) U_T.

With the normalization, the descriptors' behavior becomes independent of the particular image.

4 Classification

We use a labeled set of images to learn the behavior of our statistical descriptors and train different classifiers (supervised learning). The goal is to determine whether a new incoming image contains a hidden message. We test and validate our framework using two

classifiers: *Linear Discriminant Analysis* (LDA) and a two-class *Support Vector Machine* (SVM) [5, 19, 20]. We also perform our tests and validation using *Bagging ensemble* [5].

4.1 Linear Discriminant Analysis (LDA)

In the LDA model, we want to select the components of a linear transformation that maximize the differences between the analyzed classes while minimize the intra-classes variability. Denote x_i^0 as the set of images that are in the training set and do not have an embedded message and x_i^1 otherwise. N is the number of samples in a given set. The within-class mean are defined as

$$\mu_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} x_i^0, \quad \text{and} \quad \mu_1 = \frac{1}{N_1}\sum_{j=1}^{N_1} x_j^1. \tag{5}$$

From the within-class means, we can define the general mean

$$\mu = \frac{1}{N_0+N_1}\left(\sum_{i=1}^{N_0} x_i^0 + \frac{1}{N_1}\sum_{j=1}^{N_1} x_j^1\right), \tag{6}$$

and the within-class scatter matrix

$$S_w = M_0 M_0^T + M_1 M_1^T, \tag{7}$$

where the i^{th} column of the matrix M_0 contains the zero-meaned i^{th} exemplar given by $x_0^i - \mu_0$. Similarly, the j^{th}-column of matrix M_1 contains $x_1^i - \mu_1$. The between-class scatter matrix is defined as

$$S_b = N_0(\mu_0 - \mu)(\mu_0 - \mu)^T + N_1(\mu_1 - \mu)(\mu_1 - \mu)^T. \tag{8}$$

To maximize the differences between the analyzed classes and minimize the intra-classes variability in a single dimension, it is enough to calculate the maximal generalized eigenvalue-eigenvector $\vec{e}$ of S_b and S_w, that is, $S_b\vec{e} = \lambda S_w\vec{e}$. With the maximal generalized eigenvector, we can project the samples into one-dimensional linear subspace and threshold the new samples to perform the classification.

4.2 Support Vector Machine (SVM)

In the SVM model, we are looking for an optimal separating hyperplane between two classes. This is done maximizing the *margin* (minimal distance of an example to the decision surface). When it is not possible to find a linear separator, we project the data into a higher space using techniques known as kernels [20]. SVMs can be linear separable, linear non-separable and non-linear.

In the linear separable SVMs, the points that lie on the hyperplane satisfy the constraint

$$w^t x_i + b = 0, \tag{9}$$

where w is the normal to the hyperplane, b is the bias of the hyperplane, $|b|/||w||$ is the perpendicular distance from the origin to the hyperplane and $||.||$ is the Euclidean norm. To find out w and maximize the margin, we can use Lagrangian multipliers

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j x_i^t x_j y_i y_j, \tag{10}$$

where α is the positive Lagrangian multipliers, x are the samples of the X input vector and y are the expected outputs. A solution to the linear separable classifier, if it exists, yields values of α_i. Using the α_i values, we can calculate the normal to the hyperplane w

$$w = \sum_{i=1}^{N} \alpha_i x_i y_i. \tag{11}$$

From w we can calculate the bias b of the separating hyperplane applying the Karush-Kuhn-Tucker [19] condition

$$b = \frac{1}{N} \sum_{i=1}^{N} y_i - w^t x_i. \tag{12}$$

Sometimes, a linear separable SVM may not find a solution. Such a situation can be handled introducing *slack* variables ϵ_i that relax the constraint in the Equation 9 [20].

When it is not possible to find a linear hyperplane that optimally separates the data, we can apply a non-linear SVM. In this model, we map the training examples into a higher-dimensional Euclidean space in which a linear SVM can be applied. This mapping can be done using a Kernel K such a polynomial or a radial basis function (RBF) [19].

Denoting by $\phi = \mathcal{L} \rightarrow \mathcal{H}$, a mapping from the lower to the higher space and $K(x_i, x_j) = \phi(x_i)^t \phi(x_j)$ an on-the-fly kernel, we can test a new incoming exemplar z simply solving the equation

$$w^t \phi(z) + b = \sum_{i=1}^{N} (\alpha_i K(x_i, z) y_i + b). \tag{13}$$

4.3 Boostrap Aggregation (Bagging)

In Bagging ensemble, we evaluate the predictions over a collection of bootstrap samples to reduce its variance and consequently the prediction error. We create an ensemble by training individual classifiers on bootstrap samples[4] of the training set.

For each bootstrap sample Z^b, $b = 1, 2, \ldots, B$, we apply the selected classifier and get the prediction $\vec{f}^b(X)$. The bagging estimate is defined by

$$\vec{f}_{bag} = \frac{1}{B} \sum_{b=1}^{B} \vec{f}^b(X) \tag{14}$$

[4]Bootstrap samples are samples made with replacement and same size from a dataset.

In each step, we construct a training set doing sampling-with-replacement from the original training set. As a result of this operation, each classifier is trained on the average of 63.2% of the training examples [5].

In our experiments (Section 5), we have performed tests using classifiers of the same type and ran experiments with the LDA and SVM classifiers. For each selected ensemble classifier, we have combined the results applying a simple majority vote across the ensemble.

5 Experiments and results

In this section, we describe how we train, test and validate our framework. We show the accuracy of our approach with the selected classifiers and compare our results with previous work in the literature [10, 13, 14, 16].

We validate our methodology in a database with 20,000 real, non-synthetic images. All images have a resolution of 512×512 pixels and are stored in the PNG format. These images come from personal and from copyright-free image databases in the internet.

Here, we define a classifier *accuracy* as the ratio between the number of correctly classified images in the *testing set* (set of images not used in the training phase) and the total number of images. *False positive rate* (FPR) is the ratio between the number of non-stego images that are misclassified as stego and the total number of non-stego images. Finally, we define *false negative rate* (FNR), as the ratio between the number of stego images that are misclassified as non-stego and the total number of stego images.

5.1 Training and testing

We assume that all our 20,000 images are non-stego (images that do not contain any hidden message).

In order to train a classifier, we need both stego and non-stego examples. To obtain stego examples, we embed messages in a subset of our database. One stego image can contain an embedded message of variable size. We have selected $n = 6$ possible content-hiding scenarios with message sizes 1%, 5%, 10%, 25%, 50%, and 75% of the LSB channel available space to simulate stego images.

We have created a version of our image database for each one of our selected content-hiding scenarios. We apply the Progressive Randomization over each image, analyze it looking for its feature regions, measure the descriptors on these regions and normalize the analyzed values (Section 3.3). Later, we train a two-class (non-stego/stego) classifier for each group of stego-images and perform the test phase over a set of non-trained images.

Figures 4(a-b) show the U_T descriptor separability measured on the image showed in Figure 2. Note the differences in the descriptor values when we apply the progressive randomization operation over a stego image. The greater the embedded message, the lower the ratio between subsequent iterations of the progressive randomization operation.

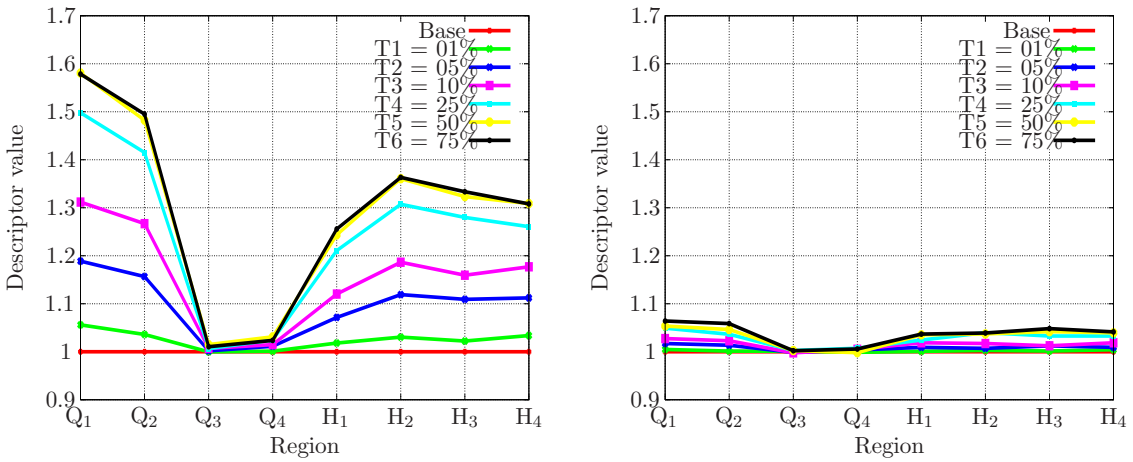

Figure 4: Behavior of U_T over the image showed in Figure 2, along the progressive randomization. (a) Non-stego image. (b) Stego version of the same image with an embedded message of size $|M| = 25\%$ of the LSB channel available space.

5.2 Validation

We select eight regions (Section 3.3), four regions (Quads) that are spatially constant, and four regions (Harris) that are instance specific. For each region, we calculate two statistical descriptors (χ^2 and U_T). randomization. With $n = 6$ possible transformations, we have 112 descriptor values. After normalization (Section 3.3), our classification procedures operate on a 96-dimensional space.

Our C++ implementation, running on an AMD 64 bits 3,200+ with 2 GB of RAM, generates the 96-dimensional descriptor vector of an image with resolution of 512×512 pixels in 30 seconds.

After training the selected classifiers (one for each relative message size), we test our framework as we have described in Section 5.1. In our experiments, we have used the R software package [21] to train and evaluate the selected classifiers (Section 3.3).

We perform a k-fold cross-validation. We partition the 20,000 images into $k = 10$ subsets of size 2,000 images. Of the k subsets, we retain one as the validation data for testing the model, and the remaining $k - 1$ subsets we use as training data. We repeat the cross-validation process k times (the folds), with each of the k subsets used exactly once as the validation data. We average the k results (μ) from the folds and calculate the standard deviation, σ. In all subsequent tables, we report the results of a 10-fold cross-validation (all using the same partition of the image collection).

5.3 Progressive randomization

Table 1 presents the results for stand-alone (white background) and for Bagging (gray background) classifiers for each chosen relative-size message embedding. We have chosen 100 iterations for Bagging. When we have testing cases with small relative-size embeddings

	LDA		**SVM-RBF**		**Type**
	μ	σ	μ	σ	
05%	65.2%	1.0%	70.7%	0.9%	Stand-alone
	69.4%	0.8%	69.0%	0.8%	Bagging
10%	75.5%	0.7%	80.2%	0.5%	Stand-alone
	78.1%	0.8%	78.2%	0.7%	Bagging
25%	85.6%	0.8%	89.3%	0.6%	Stand-alone
	88.7%	0.4%	88.9%	0.6%	Bagging
50%	89.0%	0.6%	94.0%	0.5%	Stand-alone
	93.7%	0.5%	93.7%	0.5%	Bagging
75%	92.0%	0.6%	96.3%	0.3%	Stand-alone
	96.3%	0.4%	96.2%	0.4%	Bagging

Table 1: Pairwise classification with four Q_{rs} and four H_{rs}.

(e.g. $|M| = 01\%$ and $|M| = 05\%$), LDA produces better results than SVM-RBF. In the remaining cases, SVM-RBF produces better accuracies than LDA. Although SVM-RBF produces better results, it is computationally expensive. In this case, we can use Bagging ensemble with a weaker classifier like LDA. With this approach, we have Bagging and LDA as the best tradeoff between accuracy and time complexity. Using Bagging and LDA, we detect stego images with messages of size $|M| = 25\%$ with an accuracy of $\mu = 88.7\%$ and $\sigma = 0.4\%$ that is statistically the same result (e.g. $\mu = 89.3$ and $\sigma = 0.6\%$) of the high-computationally time spending SVM-RBF. Our results clearly indicate that SVM does not benefit from bagging.

5.3.1 Influence of Harris regions

We use Harris regions in a subjective/anecdotal attempt to find localized embeddings that an individual can do in the image areas with high-degree of details[5].

It is important to understand both positive and negative impacts of using Harris regions when the message is evenly distributed along all the image, like the ones we use to classify and validate in this paper.

Using four Harris and four Quads regions is slightly inferior than using eight constant regions. However, the greater the embedded message, the lower the difference in the accuracies of the methods.

[5]In such areas, the inserted artifacts are less noticeable [7, 9]

5.3.2 Final considerations

Looking at our results (Table 1, we see that the smaller the message the worse is the classifier performance. When we have images with embedded messages of relative size less than 1% of the LSB channel available space, the detection is very hard, and still an open problem – it is almost impossible to detect them. In practice, when pornographers use images to sell their child-porn images, they usually use a reasonable portion of the LSB channel available space. A typical cover image of 1024×768 pixels in resolution has 294 KB of available space in the LSB channel. A typical JPEG image of 800×600 pixels in resolution has a size of about 75 KB. In the case where a pornographer wants to distribute such an image embedded in a typical cover image, he will use about 25% of the LSB available space. In this class of problem, our approach detects such activities with accuracy just under 90% (i.e. $\mu = 89.3\%$ and $\sigma = 0.6\%$) using SVM-RBF.

5.4 Westfeld-Pfitzmann's approach

Westfeld and Pfitzmann [10, 16] have devised an approach that only detects sequential hidden messages embedded from the first available LSB. This approach is not is not robust to image contexts variability. Also, it is not robust to detect messages altered from some embedding message procedure that keep some basic statistics such as mean, variance, and standard deviation about the cover image.

Our framework overcomes these problems and increases the classification accuracy in about 12 percentile points. To compare Westfeld-Pfitzmann's results with ours, we have modified their approach to use a training stage and to consider the same feature regions as in our approach. Table 2 compares both approaches. Our results (gray background) are about 10 percentile points better than Westfeld-Pfitzmann's results for small relative-size message embeddings (e.g. $|M| = 05\%$) and are about eight percentile

	LDA		**SVM-RBF**		**Type**
	μ	σ	μ	σ	
05%	60.4%	0.8%	52.6%	0.1%	WP
	65.2%	1.0%	70.7%	0.9%	PR
10%	68.6%	0.9%	54.6%	4.1%	WP
	75.5%	0.7%	80.2%	0.5%	PR
25%	81.6%	0.6%	72.9%	1.9%	WP
	85.6%	0.8%	89.3%	0.6%	PR
50%	90.7%	0.5%	83.0%	0.6%	WP
	89.0%	0.6%	94.0%	0.5%	PR
75%	95.0%	0.4%	84.8%	0.9%	WP
	92.0%	0.6%	96.3%	0.3%	PR

Table 2: Westfeld-Pfitzmann's (WP) vs. Progressive Randomization (PR) approach.

points better than Westfeld-Pfitzmann's results for medium relative-size message embeddings (e.g. $|M| = 10\%$ and $|M| = 25\%$).

5.5 Lyu-Farid's approach

Lyu and Farid [13,14] have designed a technique that decomposes the image into quadrature mirror filters (QMFs) [15] to analyze the effect of the embedding process. They have used a database of about 40,000 images.

The authors tuned their classifiers parameters to have a false positive rate of only 1%. They prefer to miss some images with embedded messages than to misclassify an image with no message. Table 3 compares Lyu and Farid's results (white background) and ours (gray background). The accuracy showed there, for comparison, is the percentage of the stego-

	LDA		**SVM-RBF**		**Type**
	μ	σ	μ	σ	
01%	1.3%	–	1.9%	–	LF
	3.2%	0.5%	3.6%	1.0%	PR
10%	2.8%	–	6.2%	–	LF
	7.0%	0.8%	15.8%	1.1%	PR
50%	16.8%	–	44.7%	–	LF
	24.2%	1.5%	53.1%	1.6%	PR
99%	42.3%	–	78.0%	–	LF
	95.8%	0.5%	97.0%	0.6%	PR

Table 3: Lyu and Farid's (LF) vs. Progressive Randomization (PR) approach considering FPR = 1%. LF's results from [13, 14].

images correctly classified. It does not make sense to compute the accuracy as the number of non-stego images correctly classified, given that we configured the classifier parameters to result in a FPR = 1%.

Looking at the table, we can see that our results (gray background) are more reliable than Lyu and Farid's results. Our progressive randomization approach detects small message embeddings (e.g. $|M| = 01\%$ and $|M| = 10\%$) with an accuracy of about two through nine percentile points better than Lyu and Farid's approach considering both LDA and SVM-RBF. For medium message embeddings (e.g. $|M| = 50\%$), our Progressive Randomization has an improvement of about eight percentile points over Lyu and Farid's (about eight standard deviations) approach considering LDA and two percentile points considering SVM-RBF. When we consider large relative-size message embeddings, our approach is 53 (about 106 standard deviations) percentile points more reliable than Lyu and Farid's approach considering LDA and about 19 percentile points (about 31 standard deviations) better considering SVM.

When we have a fixed FPR = 1%, the results using SVM-RBF, although computationally more expensive, are better than using simply linear discriminant analysis (LDA). In both approaches, however, the smaller the message, the worse the classifier accuracy.

Although our technique still does not take advantage of spatial coherence, it already performs better than the existing comparable techniques [9–14].

6 Conclusions and future work

We have presented a new methodology that allows us to detect hidden messages randomly scattered in the LSB field of an image.

Our progressive randomization approach only uses statistics of the LSB fields to capture the artifacts inserted by the embedding process. Our results have a better accuracy than previous approaches in the literature, and indicate that our method is an effective approach for embedding message detection.

The smaller the relative-size-message embedding, the worse is the classifier performance. The detection of very small relative-sized contents is still an open problem.

The detection of very small relative-size contents is very hard, and still an open problem – it is almost impossible to detect them. However, in practical situations, like when pornographers use images to sell their child-porn images, they usually use a reasonable portion of the LSB channel available space (e.g. 25%). In this class of problem, our approach detects such activities with accuracy just under 90% (i.e. $\mu = 89.3\%$ and $\sigma = 0.6\%$) using SVM-RBF.

In this paper, we focused on steganalysis. However, we have already strong experimental evidence that we can apply PR approach in other contexts such as the detection of digitally retouched images and art forgery, classification of images and content based image retrieval.

Our future work includes a theoretical analysis of the relationship among progressive randomization, entropy, and information theory to obtain proofs of correctness, and limitations of our method. We are also interested in a multi scale analysis of our approach to take into account the advantage of spatial coherence.

Finally, we plan to apply our technique to the detection of different types of steganography methods, and to other types of image classification problems.

Acknowledgments

The authors would like to thank the financial support of FAPESP (Procs. 04/02384-1, and 05/58103-3), CNPq (Proc. 301278/2004), and FAEPEX-UNICAMP (Proc. 1679).

References

[1] R. Anderson and F. Petitcolas, "On the limits of steganography," *IEEE Journal of Selected Areas in Communications*, vol. 16, pp. 474–481, may 1998. [Online]. Available: citeseer.ist.psu.edu/article/anderson98limits.html

[2] S. V. Hart, "Forensic examination of digital evidence: a guide for law enforcement," *NIJ-US*, 2004.

[3] S. Morris, "The future of netcrime now (1) – threats and challenges," Home Office Crime and Policing Group, Tech. Rep. 62/04, 2004.

[4] R. T. Mercuri, "The many colors of multimedia security," *Communications of the ACM*, vol. 47, pp. 25–29, 2004.

[5] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer Verlag, 2001.

[6] N. Johnson and S. Jajodia, "Steganalysis of images created using current steganography software," in *Proc. of the 2nd Intl. Workshop on Information Hiding*. Springer-Verlag, 1998, pp. 273–289.

[7] P. Wayner, *Disappearing cryptography*. Morgan Kaufmann Publishers, 2002.

[8] F. Petitcolas, R. Anderson, and M. Kuhn, "Information hiding — A survey," *Proc. of the IEEE*, vol. 87, pp. 1062–1078, 1999.

[9] N. Johnson and S. Jajodia, "Exploring steganography: Seeing the unseen," *IEEE Computer*, vol. 31, no. 2, pp. 26–34, 1998. [Online]. Available: citeseer.ist.psu.edu/johnson98exploring.html

[10] A. Westfeld and A. Pfitzmann, "Attacks on steganographic systems," in *Proc. of the 3rd Intl. Workshop on Information Hiding*, 1999, pp. 61–76.

[11] J. Fridrich, R. Du, and M. Long, "Steganalysis of LSB enconding in color images," in *ICME – Intl. Conf. Multimedia Expo*, vol. 3, Aug. 2000, pp. 1279–1282.

[12] J. Fridrich, M. Goljan, and R. Du, "Reliable detection of LSB steganography in color and grayscale images," in *Proc. of the ACM Workshop on Multimedia and Security*, 2001, pp. 27–30.

[13] S. Lyu and H. Farid, "Detecting hidden messages using higher-order statistics and support vector machines," in *Proc. of the 5th Intl. Workshop on Information Hiding*, 2002, pp. 340–354.

[14] H. Farid, "Detecting hidden messages using higher-order statistical models," in *Proc. of the 5th Intl. Conf. on Image Processing*, vol. 2, 2002, pp. 905–908.

[15] P. Vaidyanathan, "Quadrature mirror filter banks, m-band extensions and perfect reconstruction techniques," *IEEE ASSP Magazine*, pp. 4–20, jul 1987.

[16] N. Provos and P. Honeyman, "Detecting steganographic content on the internet," University of Michigan, Tech. Rep. CITI 01-1a, 2001.

[17] U. Maurer, "A universal statistical test for random bit generators," *Journal of Cryptology*, vol. 5, pp. 89–105, 1992.

[18] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of The 4th Alvey Vision Conf.*, 1988, pp. 147–151.

[19] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge U. Press, 2000.

[20] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: citeseer.ist.psu.edu/cortes95supportvector.html

[21] W. N. Venables and D. M. Smith, *An introduction to R: a programming environment for data analysis and graphics*. R Development Core Team, 2005.