JPEG-Compatibility Steganalysis Using Block-Histogram of Recompression Artifacts

Jan Kodovský and Jessica Fridrich

Department of ECE, Binghamton University, NY, USA
{fridrich,jan.kodovsky}@binghamton.edu

Abstract. JPEG-compatibility steganalysis detects the presence of embedding changes using the fact that the stego image was previously JPEG compressed. Following the previous art, we work with the difference between the stego image and an estimate of the cover image obtained by recompression with a JPEG quantization table estimated from the stego image. To better distinguish recompression artifacts from embedding changes, the difference image is represented using a feature vector in the form of a histogram of the number of mismatched pixels in 8×8 blocks. Three types of classifiers are built to assess the detection accuracy and compare the performance to prior art: a clairvoyant detector trained for a fixed embedding change rate, a constant false-alarm rate detector for an unknown change rate, and a quantitative detector. The proposed approach offers significantly more accurate detection across a wide range of quality factors and embedding operations, especially for very small change rates. The technique requires an accurate estimate of the JPEG compression parameters.

1 Introduction

When a JPEG image is decompressed to the spatial domain, the pixel values in each 8×8 block must be obtainable by decompressing an 8×8 block of quantized DCT coefficients. However, most steganographic algorithms change the pixels in a way that makes each block almost surely incompatible with the compression in the sense that no DCT coefficient block can decompress to such a modified block of pixels. This JPEG-compatibility attack was described for the first time in 2001 [6]. The assumption that the cover was originally stored as JPEG is not that unreasonable as the vast majority of images are stored as JPEGs and casual steganographers might hide data in the spatial domain in order to hide larger payloads or simply because their data hiding program cannot handle the JPEG format. In fact, while there are almost eight hundred publicly available applications that hide messages in raster formats, fewer than two hundred can hide data in JPEGs.[1]

The original JPEG-compatibility detection algorithm [6] strived to provide a mathematical guarantee that a given block was incompatible with a certain

[1] Statistics taken from a data hiding software depository of WetStone Tech.

JPEG quantization matrix, which required a brute-force search. With an increasing quality factor (decreasing value of the quantization steps), however, the complexity of this search rapidly increases making it impractically time consuming to use in practice. This prompted researchers to seek alternatives.

In 2008, a quantitative LSB replacement detector was proposed [1,2] as a version of the weighted stego-image (WS) analysis [5,7] equipped with uniform weights and a pixel predictor based on recompressing the stego image with a quantization table estimated from the stego image. This detector proved remarkably accurate and also fairly robust w.r.t. errors in the estimated quantization table as well as different JPEG compressors. Luo et al. [12] used the same recompression predictor but based their decision on the number of pixels in which the stego image and its recompressed version differed. This allowed detection of embedding operations other than LSB replacement.

The cover-image prediction based on recompression is fairly accurate for low quality factors. With decreasing size of the quantization steps, the quantization noise in the DCT domain becomes comparable to the quantization noise in the spatial domain and the recompression predictor becomes increasingly poor, preventing thus the detection of (or quantifying) the embedding changes. However, the recompression artifacts due to quantization in both domains cannot be completely arbitrary. In particular, it is highly unlikely that such artifacts would manifest as a single changed pixel or, in general, a small number of changed pixels. This motivated us in Section 4 to form a feature vector as the histogram of the number of mismatched pixels in 8×8 blocks after recompression. This 65-dimensional feature vector better distinguishes embedding changes from recompression artifacts and significantly improves the detection accuracy especially for low embedding rates. In Section 5, we report the detection accuracy of three types of detectors, interpret the results, and compare them to previous art. The paper is summarized in Section 7.

2 Notation and preliminaries

We use the boldface font for matrices and vectors and the corresponding lowercase symbols for their elements. In particular, $\mathbf{X} = (x_{ij}) \in \mathcal{X} = \mathcal{I}^{n_1 \times n_2}$, $\mathcal{I} = \{0, \dots, 255\}$, and $\mathbf{Y} = (y_{ij}) \in \mathcal{X}$ will represent the pixel values of grayscale cover and stego images with $n = n_1 \times n_2$ pixels. For simplicity, we assume that both n_1 and n_2 are multiples of 8 and limit our exposition to grayscale images. This also allows us to use publicly available image datasets, such as the grayscale BOSSbase [4], which gives our results a useful context.

For convenience, images will also be represented by blocks, $\mathbf{X} = (\mathbf{X}^{(k)})$, $\mathbf{X}^{(k)} = (x_{ij}^{(k)})$, where now $i, j \in \{0, \dots, 7\}$ index the pixels in the kth block, $k \in \{1, \dots, n/64\}$, assuming, for example, that the blocks are indexed in a row-by-row fashion. For the purpose of this paper, we define the operator of JPEG compression on an 8×8 pixel block, $\mathbf{X}^{(k)}$, as $\mathrm{JPEG}_\theta(\mathbf{X}^{(k)}) = \mathbf{D}^{(k)} \in \mathcal{J}^{8\times 8}$, where $\mathcal{J} = \{-1023, \dots, 1024\}$ and $\mathbf{D}^{(k)}$ is the kth block of quantized Discrete Cosine Transform (DCT) coefficients. Here, θ stands for a vector parameter defining the

compressor, such as the quantization table(s), the type of the JPEG compressor (e.g., Matlab `imwrite` or ImageMagick `convert`), and the implementation of the DCT, such as 'float', 'fast', 'slow'. The parameters related to the lossless compression in JPEG, such as the Huffmann tables, are not important for our problem.

Typically, the JPEG operator will be applied to the entire image in a block-by-block fashion to obtain an array of DCT coefficients of the same dimension, $\mathbf{D} \in \mathcal{J}^{n_1 \times n_2}$, as the original uncompressed image: $\text{JPEG}_\theta(\mathbf{X}) = \mathbf{D} = (\mathbf{D}^{(k)})$, $\text{JPEG}_\theta(\mathbf{X}^{(k)}) = \mathbf{D}^{(k)}$ for all k. We also define the JPEG decompression operator as $\text{JPEG}_\theta^{-1} : \mathcal{J}^{8\times 8} \to \mathcal{I}^{8\times 8}$. In short, $\text{JPEG}_\theta^{-1}(\mathbf{D}^{(k)})$ is the kth pixel block in the decompressed JPEG image $\text{JPEG}_\theta^{-1}(\mathbf{D})$. The decompression involves multiplying the quantized DCT coefficients by the quantization matrix, applying the inverse DCT to the resulting 8×8 array of integers, and quantizing all pixel values to $\mathcal{I}$. Note that JPEG_θ^{-1} is not the inverse of JPEG_θ, which is many-to-one. In fact, in general $\text{JPEG}_\theta^{-1}(\text{JPEG}_\theta(\mathbf{X})) \neq \mathbf{X}$; the difference between them will be called the recompression artifacts.

All experiments are carried out on the BOSSbase image database ver. 0.92 [4] compressed with Matlab JPEG compressor `imwrite` with different quality factors. The original database contains $9,074$ images acquired by seven digital cameras in their RAW format (CR2 or DNG) and subsequently processed by converting to grayscale, resizing, and cropping to the size of 512×512 pixels using the script available from [4].

3 Prior art

In this paper, we compare to the WS detector adapted for decompressed JPEGs [1] and the method of Luo *et al.* [12]. Both methods output an estimate of the embedding change rate, β, defined as the ratio between the number of embedding changes and the number of all pixels.

3.1 WS adapted for JPEG

Böhme's change-rate estimator of LSB replacement in decompressed JPEGs (WSJPG) is a version of the WS estimator:

$$\hat{\beta}_{\text{WSJPG}} = \frac{1}{n} \sum_{i,j=1}^{n_1,n_2} (y_{ij} - \bar{y}_{ij})(y_{ij} - \hat{y}_{ij}), \tag{1}$$

where $\bar{y} = y + 1 - 2 \bmod (y, 2)$ is y with its LSB "flipped,"

$$\hat{\mathbf{Y}} = (\hat{y}_{ij}) = \text{JPEG}_\theta^{-1}\left(\text{JPEG}_\theta(\mathbf{Y})\right), \tag{2}$$

is the recompression pixel predictor, and $\mathbf{R} = (r_{ij})$, $r_{ij} = y_{ij} - \hat{y}_{ij}$ is the residual. Note that both $\hat{\mathbf{Y}}$ and $\mathbf{R}$ depend on θ but we do not make this dependence explicit for better readability. The WSJPG estimator is limited to LSB replacement and will not work for other embedding operations, such as LSB matching.

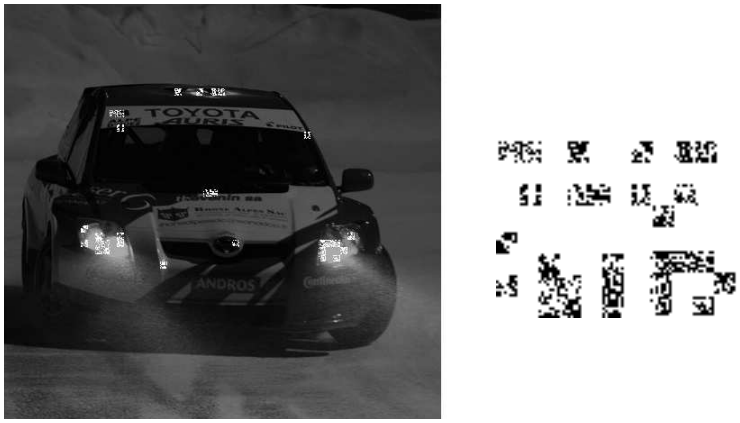

Fig. 1. Left: cover image '101.pgm' from BOSSbase compressed with quality factor 80. Right: close up of the recompression artifacts (grouped into a smaller region) with the same quality factor. The image contrast was decreased to better show the artifacts.

3.2 Detector by Luo *et al.*

The detector by Luo *et al.* [12] (which we abbreviate LUO) is also quantitative – it returns an estimate of the change rate as the detection statistic. It is computed from the relative number of differences between $\mathbf{Y}$ and $\hat{\mathbf{Y}}$:

$$\triangle_\theta = \frac{1}{n} \left| \{(i,j) | r_{ij} \neq 0\} \right| . \tag{3}$$

In general, both the embedding changes as well as the recompression artifacts contribute to $\triangle_\theta$. Since the artifacts depend on θ, the authors further transform $\triangle_\theta$ to obtain an unbiased estimate of the change rate:

$$\hat{\beta}_{\mathrm{LUO}} = p_\theta(\triangle_\theta), \tag{4}$$

where $p_\theta(x)$ is a polynomial. The authors show that it is sufficient to consider a third degree polynomial, $p_\theta(x) = a_\theta + b_\theta x + c_\theta x^2 + d_\theta x^3$. Note that as long as the polynomial is monotone (as it seems to always be in [12]), $\triangle_\theta$ is an equivalent detection statistic, which is why we use it here for performance evaluation.

4 The histogram feature

Recompression artifacts manifest quite differently in the residual $\mathbf{R} = \hat{\mathbf{Y}} - \mathbf{Y}$ than the embedding changes. Figure 1 shows the cover image '101.pgm' from BOSSbase originally compressed with quality factor 80 together with the recompression artifacts. Although the artifacts typically occur in saturated areas, such as the overexposed headlights, they can show up in other regions with no saturated pixels (the car's hood and roof). The artifacts usually show up as a whole

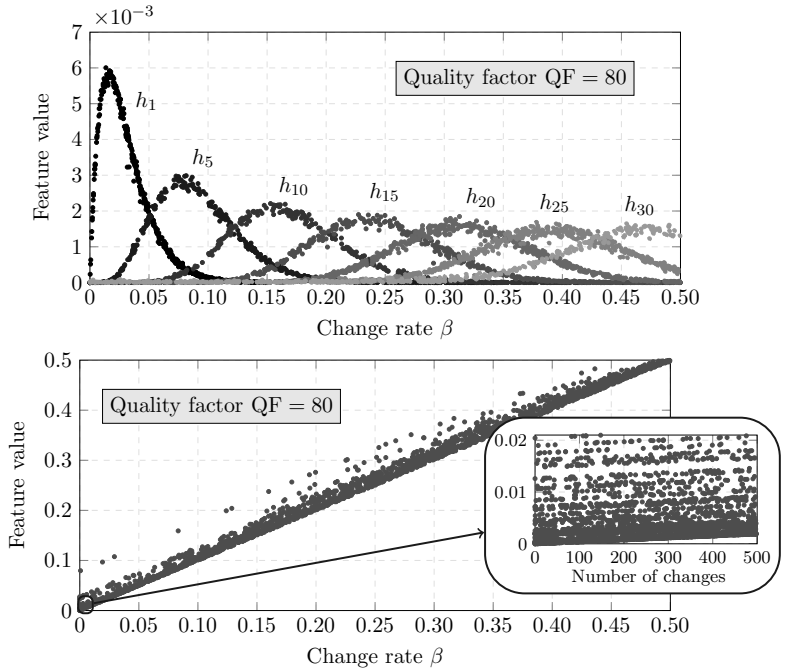

Fig. 2. Values of selected features h_i (top) and $\triangle_\theta$ (bottom) across 100 images and randomly selected change rates.

pattern and almost never as individual pixels. Classifying them, however, would be infeasible as there are simply too many possible patterns and their number quickly increases with the quality factor. In fact, this is why the search in [6] is computationally intractable.

In this paper, we delegate the difficult task of distinguishing "legitimate" recompression artifacts from those corrupted by embedding changes to machine learning. To this end, each block, $\mathbf{R}^{(k)}$, of the residual is represented using a scalar – the number of pixels in $\mathbf{R}^{(k)}$ for which $r_{ij}^{(k)} \neq 0$. Denoting this number as $0 \leq \rho^{(k)} \leq 64$, $k = 1, \ldots, n/64$, each image will be mapped to a feature vector $\mathbf{h} = (h_m)$ obtained as the histogram of $\rho^{(k)}$:

$$h_m = \frac{64}{n} \left| \{k | \rho^{(k)} = m\} \right|, \quad m = 0, \ldots, 64. \tag{5}$$

This feature vector can be considered as a generalization of (3) because $\triangle_\theta = \frac{1}{64} \sum_{m=0}^{64} m h_m$ is a projection of $\mathbf{h}$ onto a fixed direction.

Using 100 randomly selected images and a large number of change rates, in Figure 2 (top) we show how the individual features h_m react to increasing change rate. Together, the features capture the effects of embedding much better than the scalar $\triangle_\theta$. For example, a small number of embedding changes affect primarily h_1 while the recompression artifacts typically disturb h_m with a much larger

m. In contrast, $\triangle_\theta$ cannot distinguish embedding changes from recompression artifacts. Zooming in Figure 2 (bottom) around $\beta = 0$ reveals individual "lines" of dots corresponding to the 100 tested images. The vertical offset of the lines is due to recompression artifacts that introduce undesirable noise into $\triangle_\theta$, which prevents reliable detection (and estimation) of small change rates.

We close this section with one more remark. Detecting steganography using a binary classifier with a higher-dimensional feature is usually considered as less convenient or practical than alternative detectors that, for example, provide an estimate of the change rate. This is mainly because one needs to train the classifier on examples of cover (and stego) images from a given source. However, when images from a different source are tested, one may experience a loss of detection accuracy due to lack of robustness of today's classifiers to model mismatch (when one trains on one source but tests on another). In our case, however, the effect of the model mismatch is largely mitigated due to the fact that *all* JPEG-compatibility attacks require the knowledge of the JPEG parameter θ to apply in the first place. The source of JPEG images compressed with one quality factor is much more homogeneous than images in their uncompressed format because the compression suppresses the noise and thus evens out the source, making the issue with model mismatch less serious.

5 Experiments

This section contains all experiments and their interpretation. First, we measure the detection reliability of a clairvoyant detector (built for a specific change rate) across a wide spectrum of JPEG quality factors while comparing the results with WSJPG and LUO. Then, a single constant false-alarm rate (CFAR) detector is built to detect all change rates. Finally, we construct and test a quantitative version of the detector. All experiments are carried out under the assumption that the JPEG compressor parameter θ is correctly estimated, postponing the discussion of detector robustness to Section 6.

5.1 Classifier

The clairvoyant detector and the CFAR detector are instances of the ensemble [9,8] available from `http://dde.binghamton.edu/download/ensemble`. The ensemble reaches its decision using majority voting by fusing decisions of L individual base learners implemented as Fisher linear discriminants trained on random d_{sub}-dimensional subspaces of the feature space. The random subspace dimensionality, d_{sub}, and the number of base learners, L, are determined automatically by measuring the out-of-bag estimate of the testing error on bootstrap samples of the training set as described in [9].

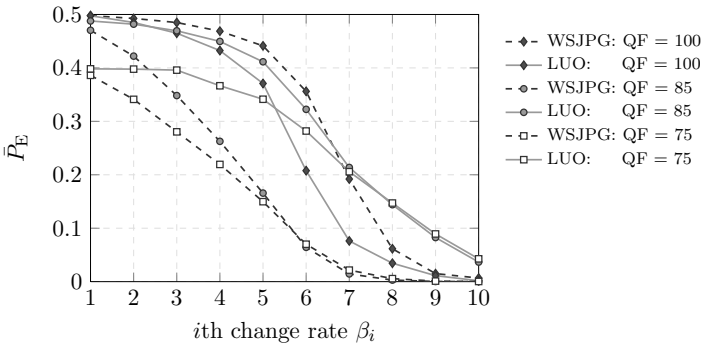

Fig. 3. Detection error $\bar{P}_E$ for WSJPG (dashed lines) and LUO (solid lines) for all ten change rates $\beta_1, \ldots, \beta_{10}$ and three selected quality factors 75, 85, and 100. Steganographic algorithm: LSB replacement.

5.2 Clairvoyant detector

In this section, detection accuracy will be measured using the minimal total error under equal priors on the testing set:

$$P_E = \min_{P_{FA}} \frac{P_{FA} + P_{MD}(P_{FA})}{2}, \tag{6}$$

where P_{FA} and P_{MD} are the false-alarm and missed-detection rates. We always report the mean value of P_E, denoted as $\bar{P}_E$, over ten random splits of BOSSbase into equally-sized training and testing sets. Since the spread of the error over the splits, which includes the effects of randomness in the ensemble construction (e.g., formation of random subspaces and bootstrap samples), is typically very small, we do not show it in tables and graphs. We note that a separate classifier was trained for each β, which is why we call it clairvoyant.

First, we work with LSB replacement to be able to compare to the WSJPG detector. The focus is on detection of very small change rates:

$$\beta_i = \begin{cases} \frac{1}{n}(1, 10, 25, 50, 100) & \text{for } i = 1, \ldots, 5, \\ 0.001, 0.0025, 0.005, 0.01, 0.02 & \text{for } i = 6, \ldots, 10. \end{cases} \tag{7}$$

as this is where we see the biggest challenge in steganalysis in general. The actual embedding changes were always made pseudo-randomly and different for each image. The first five change rates correspond to making 1, 10, 25, 50, and 100 pseudo-randomly placed embedding changes. Note that the change rate $\beta_6 = 0.001$ corresponds to 261 embedding changes for BOSSbase images, continuing thus the approximately geometric sequence of $\beta_1, \ldots, \beta_5$. Furthermore, β is the expected change rate when embedding 2β bits per pixel (bpp) if no matrix embedding is employed or the payload of $H^{-1}(\beta)$ bpp if the optimal binary coder is used ($H^{-1}(x)$ is the inverse of the binary entropy function on $x \in [0, 0.5]$).

QF	Number of changed pixels					Change rate (cpp)				
	1	10	25	50	100	0.001	0.0025	0.005	0.01	0.02
70	0	0	0	0	0	0	0	0	0	0
	0.3873	0.3468	0.2922	0.2295	0.1568	0.0763	0.0230	0.0057	0.0009	0.0003
75	0	0	0	0	0	0	0	0	0	0
	0.3861	0.3412	0.2804	0.2194	0.1497	0.0701	0.0216	0.0057	0.0010	0.0003
80	0	0	0	0	0	0	0	0	0	0
	0.4248	0.3761	0.3014	0.2295	0.1471	0.0625	0.0167	0.0037	0.0005	0.0003
85	0.0101	0	0	0	0	0	0	0	0	0
	0.4704	0.4220	0.3483	0.2626	0.1657	0.0642	0.0145	0.0029	0.0003	0.0002
90	0.0852	0.0046	0.0007	0.0010	0	0	0	0	0	0
	0.4899	0.4534	0.3950	0.3155	0.2197	0.0882	0.0183	0.0034	0.0005	0.0002
91	0.0798	0.0019	0.0001	0	0	0	0	0	0	0
	0.4913	0.4513	0.3882	0.3080	0.2076	0.0808	0.0167	0.0031	0.0004	0.0001
92	0.0893	0.0010	0	0	0	0	0	0	0	0
	0.4907	0.4505	0.3852	0.2981	0.1968	0.0722	0.0157	0.0032	0.0003	0.0001
93	0.4499	0.1017	0.0023	0	0	0	0	0	0	0
	0.4949	0.4727	0.4313	0.3673	0.2583	0.0936	0.0196	0.0040	0.0005	0.0001
94	0.4888	0.3885	0.2448	0.0906	0.0124	0.0003	0	0	0.0000	0
	0.4966	0.4802	0.4527	0.4094	0.3291	0.1482	0.0314	0.0081	0.0016	0.0003
95	0.4948	0.4472	0.3680	0.2538	0.0977	0.0025	0	0	0	0
	0.4972	0.4841	0.4611	0.4285	0.3589	0.1854	0.0372	0.0092	0.0028	0.0003
96	0.4973	0.4728	0.4320	0.3675	0.2509	0.0488	0.0018	0.0002	0.0001	0
	0.4975	0.4868	0.4680	0.4386	0.3797	0.2151	0.0499	0.0104	0.0028	0.0005
97	0.4983	0.4842	0.4595	0.4208	0.3438	0.1512	0.0178	0.0024	0.0003	0.0001
	0.4975	0.4877	0.4723	0.4433	0.3890	0.2316	0.0557	0.0108	0.0030	0.0007
98	0.4982	0.4795	0.4475	0.3936	0.3009	0.1744	0.0272	0.0034	0.0003	0.0001
	0.4980	0.4892	0.4725	0.4462	0.3911	0.2446	0.0587	0.0121	0.0024	0.0005
99	0.4988	0.4843	0.4602	0.4195	0.3398	0.1525	0.0161	0.0007	0	0
	0.4979	0.4899	0.4766	0.4588	0.4169	0.3016	0.1110	0.0226	0.0036	0.0007
100	0.4986	0.4855	0.4611	0.4251	0.3540	0.0942	0.0048	0.0006	0.0001	0.0001
	0.4978	0.4926	0.4849	0.4688	0.4413	0.3561	0.1920	0.0616	0.0151	0.0068

Table 1. Mean detection error $\bar{P}_{\mathrm{E}}$ for the proposed method (shaded) versus WSJPG.

For such small β, the WSJPG method performed better than LUO with the exception of quality factor 100 (see Figure 3). Thus, in Table 1 we contrast the proposed method with WSJPG. The improvement is apparent across all quality factors and change rates and is especially large for the five smallest change rates. Remarkably, the clairvoyant detector allows reliable detection of a single embedding change for quality factors up to 92. Then the error abruptly increases. This is related to the first occurrence of '1' in the quantization table. With this quantization step, the rounding error in the spatial domain becomes comparable to the rounding error in the DCT domain and the recompression predictor no longer provides an accurate estimate of the cover. Despite this limitation, reliable detection of change rates $\beta_6, \ldots, \beta_{10}$ is still possible even for high quality factors. It appears that the least favorable quality factor is not 100 but 98 (for change rates $\beta_i, i > 5$). The detection error is not monotone w.r.t. the quality factor and one can observe "ripples" even at lower quality factors (e.g., from 90 to 91).

	Number of changed pixels					Change rate (cpp)				
QF	1	10	25	50	100	0.001	0.0025	0.005	0.01	0.02
80	.0213	.0017	.0022	.0016	.0018	.0017	.0013	.0007	.0006	.0004
90	.1235	.0160	.0065	.0035	.0049	.0035	.0023	.0024	.0024	.0012
95	.4953	.4627	.3974	.3306	.2415	.0859	.0286	.0191	.0076	.0023

Table 2. Average detection error $\bar{P}_{\mathrm{E}}$ for HUGO.

We note that our feature vector $\mathbf{h}$ (5) as well as Luo's $\triangle_\theta$ work well for other steganographic methods than LSB replacement. Repeating the above experiment with LSB matching, we obtained identical values of $\bar{P}_{\mathrm{E}}$ well within its statistical spread. Interestingly, content-adaptive embedding appears to be slightly less detectable, which is most likely due the fact that recompression artifacts weakly correlate with texture/edges. The results for the content-adaptive HUGO [14] displayed in Table 2 should be contrasted with the corresponding rows of Table 1.[2]

5.3 CFAR detector

In the previous experiment, a separate classifier was trained for each change rate and quality factor. However, in practice, the steganalyst will likely have no or little prior information about the payload and will face the more difficult one-sided hypothesis testing problem of deciding whether $\beta = 0$ or $\beta > 0$. For this purpose, we now construct a single CFAR classifier and report its performance for LSB replacement.

Following the recipe in [13], we first tried training on a uniform mixture of change rates from a certain range. This, however, caused the detector to be undesirably inaccurate for small change rates. There appears to be an interesting interplay between the design false-alarm rate, the ability to detect small change rates, and the detection rate. Through a series of experiments, we determined that the best results were obtained when training on a *fixed small* change rate for which the clairvoyant detector's P_{E} was neither too small or too big (a value in the range $P_{\mathrm{E}} \approx 0.2 - 0.3$ seemed to work the best). This makes an intuitive sense as $P_{\mathrm{E}} \approx 0.5$ would not allow accurate determination of the direction into which the features move with embedding, while easy detectability, $P_{\mathrm{E}} \approx 0$, is also bad as there exist many decision boundaries that are equally good but only some of them are useful for smaller change rates.

The performance of the detector for three quality factors is displayed in Figure 4. Three graphs show the detection rate $P_{\mathrm{D}}(\beta)$ for selected design P_{FA}. Overall, the false-alarm rates on the testing set agreed rather well with the design rates, which we show only for the quality factor 100 just as an example. For quality factor 90, even as few as six embedding change can be detected

[2] To obtain the desired change rate β_i, we searched for the payload iteratively using the authors' embedding script.

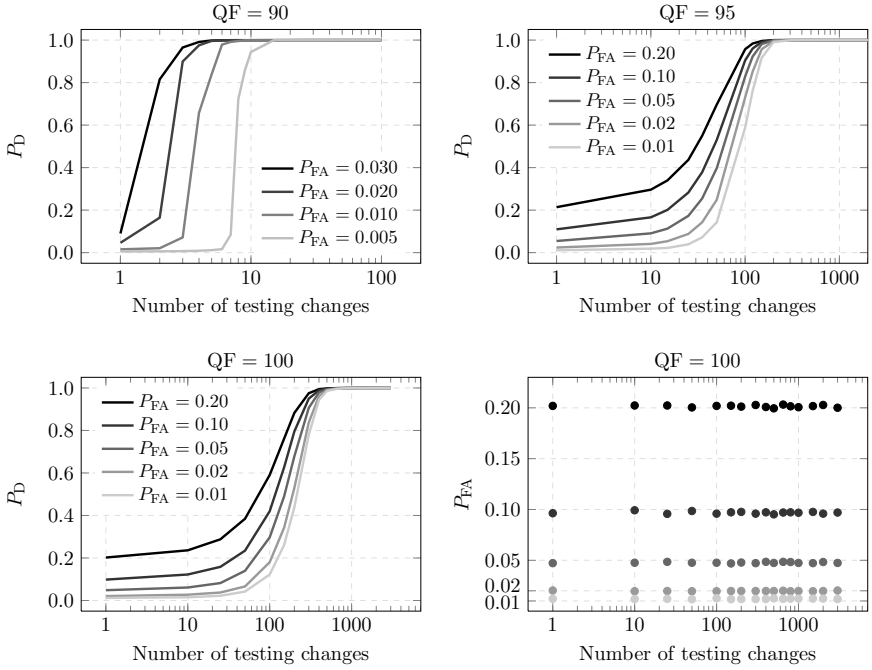

Fig. 4. Probability of detection P_{D} on the test set as a function of β for several design false alarm rates P_{FA} and three quality factors. For the highest quality factor, we also report the false alarm rate on test images. The CFAR classifier for quality factors 90, 95, and 100 was trained on 10, 25, and 50 changes, respectively.

reliably with $P_{\mathrm{FA}} = 0.01$. For quality factors 95 and 100, P_{D} experiences a sharp increase around 100 changes.

5.4 Quantitative detector

Since WSJPG and LUO are both quantitative detectors, in this section we built a quantitative version of our detector using Support Vector Regression (SVR) and compare to previous art (tests carried out for LSB replacement).

Following the methodology described in [15], the BOSSbase was divided into two halves, one used to train the quantitative detector and the other used for testing. We used ν-SVR [16] with a Gaussian kernel whose hyper-parameters (kernel width, γ, cost, C, and the parameter ν which bounds the number of support vectors) were determined using five-fold cross-validation on $\mathcal{G}_\gamma \times \mathcal{G}_C \times \mathcal{G}_\nu$, where $\mathcal{G}_\gamma = \{2^k | k = -5, \ldots, 3\}$, $\mathcal{G}_C = \{10^k | k = -3, \ldots, 4\}$, and $\mathcal{G}_\nu = \{\frac{1}{10}k | k = 1, \ldots, 9\}$. We used a public SVM package libSVM [3].

The regressor was trained on images embedded with change rates chosen uniformly and pseudo-randomly from $[0, b]$. Its accuracy was measured on stego images from the testing set embedded with a fixed change rate β using relative bias, $B_{\mathrm{r}}(\beta)$, and relative median absolute deviation (MAD) $M_{\mathrm{r}}(\beta)$:

β	Proposed scheme				Cascade
	$b = 0.0005$	$b = 0.005$	$b = 0.05$	$b = 0.5$	
$10/n$	-2.78 ± 4.84	$\times$	$\times$	$\times$	-2.78 ± 4.84
$50/n$	$+0.64 \pm 2.34$	-9.04 ± 8.06	$\times$	$\times$	$+0.65 \pm 2.35$
$100/n$	-0.22 ± 2.00	-3.36 ± 4.13	-15.6 ± 28.5	$\times$	-0.10 ± 2.02
0.001	-3.83 ± 1.72	-0.19 ± 1.75	-5.326 ± 10.9	$\times$	-0.19 ± 1.75
0.0035	-16.4 ± 1.37	$+0.11 \pm 0.71$	-0.47 ± 3.06	$\times$	$+0.13 \pm 0.71$
0.01	-43.7 ± 1.07	-0.90 ± 0.80	-0.00 ± 1.06	-16.3 ± 17.2	-0.00 ± 1.06
0.035	$\times$	$\times$	$+0.05 \pm 0.40$	-3.74 ± 4.68	$+0.07 \pm 0.40$
0.1	$\times$	$\times$	-21.1 ± 1.17	-1.17 ± 1.74	-1.27 ± 1.67
0.2	$\times$	$\times$	$\times$	-0.57 ± 0.94	-0.57 ± 0.94
0.3	$\times$	$\times$	$\times$	-0.26 ± 0.79	-0.24 ± 0.74
0.4	$\times$	$\times$	$\times$	$+0.02 \pm 0.51$	$+0.04 \pm 0.47$
0.5	$\times$	$\times$	$\times$	-0.90 ± 1.52	-0.96 ± 1.49

Table 3. Relative bias and median absolute deviation, $B_r(\beta) \pm M_r(\beta)$, as a function of β. Crosses correspond to failures (either B_r or M_r is larger than 50%). The best performance per change rate is highlighted. JPEG quality factor is 90.

$$B_{\mathrm{r}}(\beta) = \frac{1}{\beta}(\mathrm{med}(\hat{\beta}) - \beta) \times 100\%, \tag{8}$$

$$M_{\mathrm{r}}(\beta) = \frac{1}{\beta}\mathrm{med}(|\hat{\beta} - \mathrm{med}(\hat{\beta})|) \times 100\%, \tag{9}$$

where $\hat{\beta}$ is the estimated change rate and the median $\mathrm{med}(\cdot)$ is always taken over all stego images in the testing set. Note that $B_{\mathrm{r}}(\beta)$ is the *percentual* inaccuracy in estimating β, while $M_{\mathrm{r}}(\beta)$ captures the statistical spread in the same units. These relative quantities are more informative when detecting change rates of very different magnitudes.

Table 3 shows $B_{\mathrm{r}}(\beta) \pm M_{\mathrm{r}}(\beta)$ when training on stego images embedded with change rates from $[0, b]$ for four values of b for JPEG quality factor 90. The detection was declared unsuccessful, and marked by a cross, when either $B_{\mathrm{r}}(\beta)$ or $M_{\mathrm{r}}(\beta)$ was larger than 50%. The table reveals that for small β, significantly better results could be obtained by training the regressor on a smaller range $[0, b]$, provided $\beta < b$. This is because a smaller interval yields a higher density of training change rates and allows the regressor to locally adjust its hyper-parameters.

This insight inspired us to construct the quantitative detector by cascading SVR detectors D_i trained on progressively smaller ranges $[0, b_i]$, $b_i > b_{i+1}$, $b_i \in [0, 0.5]$:

1. Set $\mathbf{b} = (b_1, \ldots, b_k)$, initialize $i = 1$.
2. Compute $\hat{\beta}_i$ using D_i. If $i = k$, terminate and output $\hat{\beta}_i$.
3. If $\hat{\beta}_i \leq b_{i+1}$, increment $i = i + 1$, go to Step 2.
4. Output $\hat{\beta}_i$.

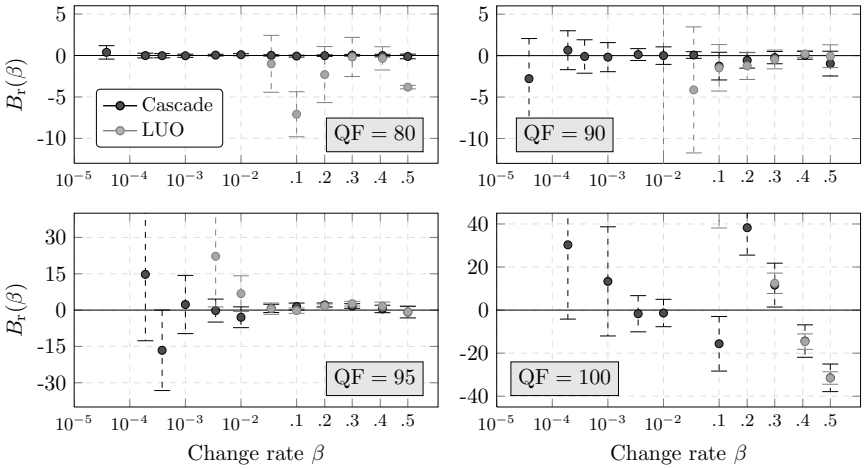

Fig. 5. Quantitative steganalysis of LSB replacement for 'Cascade' and LUO for different JPEG quality factors in terms of the relative median bias B_r; error bars depict M_r. Note the different ranges on y-axis.

The performance of this cascading regressor is reported in the last column of Table 3. As expected, it strongly benefits from its individual sub-detectors and consequently delivers superior performance across all change rates. To complete the picture, in Figure 5 we compare LUO with 'Cascade' for JPEG quality factors, 80, 90, 95, and 100. While both estimators become progressively inaccurate with increasing JPEG quality factor, 'Cascade' clearly outperforms LUO for small β in all cases while both estimators become comparable for larger β. We note that cascading the regressor for $\triangle_\theta$ by training on smaller intervals $[0, b]$ did not improve its performance. This is due to the low distinguishing power of $\triangle_\theta$ on smaller change rates (see Figure 2 bottom).

For quality factor 100 and $\beta \gtrsim 0.2$, neither of the two detectors can estimate the change rate reliably, and both begin outputting an estimate of $\hat{\beta} \approx 0.35$ (on average). This is because in this range the features are very noisy due to recompression artifacts – the quantization table consists solely of ones. Consequently, the regression learns the output that yields the smallest error on average.

5.5 Error analysis

We now decompose the compound error of the proposed quantitative detector trained on $[0, 0.5]$ into the within-image error, E_W, and the between-image error, E_B, using the procedure described in [2].

The tails of the E_W distribution are analyzed by randomly selecting a single image from the testing set followed by 200 independent realizations of LSB embedding at a fixed change rate. Our experiments confirm that this error follows the Gaussian distribution. To estimate the between-image error, we compute

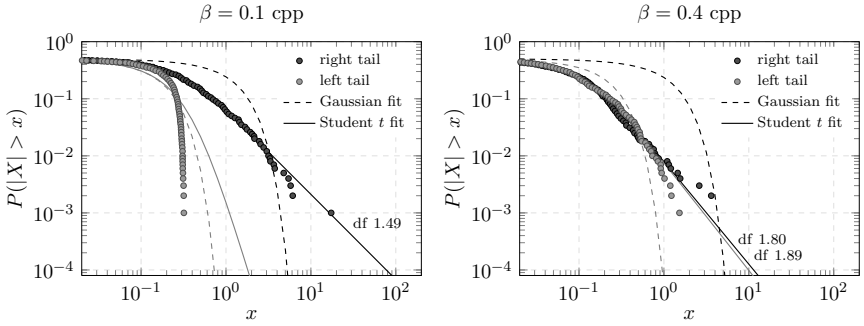

Fig. 6. Tail probability for the between-image error E_{B} for $\beta = 0.1$ and 0.4 with the Gaussian and the Student's t maximum likelihood fits. JPEG quality factor 90.

the change rate estimate for 1000 testing images by averaging estimates over 20 embedding realizations (for every image). The log-log empirical cdf plot of the resulting estimates is shown in Figure 6 for two selected values of β. While the the Student's t-distribution was generally a good fit for the right tail, we observed great variations in the distribution of the left tail based on the value of β. The tail could be extremely thin for some β, while for others it did follow the thick-tailed Student's t-distribution. We attribute these variations to the highly non-linear dependence of the feature vector on β seen in Figure 2.

6 Robustness to JPEG compressor parameters

The WSJPG detector appears to be quite resistant to incorrectly estimated quantization table or the JPEG compressor [2]. This is because stronger recompression artifacts due to improperly estimated compression parameter θ are not likely to manifest as flipped LSBs. In contrast, our feature vector, as well as LUO, are rather sensitive to θ because they *count* the mismatched pixels instead of utilizing their *parity*. While this allows them to detect embedding operations other than LSB flipping, this generality lowers their robustness.

The overall detection performance of any JPEG-compatibility detector will necessarily strongly depend on the accuracy of the estimator of θ as well as the prior distribution of θ in the testing set. Despite some encouraging work, such as [11], we consider the problem of estimating θ as an open and quite difficult problem for the following reasons. Most JPEG images today originate in digital cameras, which, unfortunately, almost exclusively use quantization tables customized for the image content, the imaging sensor, the manufacturer's color space, and the image size [17].[3] For color images, one may have to estimate up

[3] `http://www.hackerfactor.com/blog/index.php?/archives/244-Image-Ballistics-and-Photo-Fingerprinting.html`
`http://www.impulseadventure.com/photo/jpeg-quantization.html`

to three quantization tables, one for the luminance and one for each chrominance component, as well as the chrominance subsampling. The quantization tables may even be different between different cameras of the same model as manufacturers continue to upgrade the firmware. Multiple JPEG compressions further complicate the matter. Thus, the search space may be quite large even when one considers estimating only the quantization tables themselves. Methods that estimate the individual quantization steps, such as [6,11,10], may fail for high compression ratios as there may be little or no data in the JPEG file to estimate the quantization steps for sparsely populated medium–high frequency DCT modes.

The only meaningful evaluation of the robustness requires the steganalyzer to be tested as a whole system, which includes the compression estimator, and testing on non-standard quantization tables as well as multiply compressed images. The authors feel that the problem of robust compression parameter estimation is a separate issue that is beyond the scope of this paper.

7 Conclusions

This paper describes a new implementation of JPEG-compatibility steganalysis capable of detecting a wide range of embedding operations at very low change rates. As proposed previously, the image under investigation is first recompressed with a JPEG compressor estimated from the test image. The recompression artifacts are described using a 65-dimensional feature vector formed as the histogram of blocks with a certain number of mismatched pixels. This feature vector can better distinguish between recompression artifacts and embedding changes than the scalar proposed by Luo *et al.* [12]. In particular, it allows accurate detection of fewer than ten embedding changes for quality factors up to 92. For higher quality factors, the detection error sharply increases due to the onset of quantization steps equal to one. Nevertheless, very reliable detection of change rates as low as 0.005 remains possible for quality factors up to 100 (in 512×512 grayscale images).

Three types of detectors are constructed for a fixed quality factor – a family of clairvoyant detectors trained for a specific change rate, a constant false-alarm rate detector for unknown change rate for practical applications, and a quantitative detector.

The proposed method, as well as all JPEG-compatibility detectors, need to be supplied with an estimator of the JPEG compressor parameters (quantization table(s), DCT implementation, etc.). Future research will focus on tests with real-life datasets, including images compressed with non-standard quantization tables and multiply-compressed images, and on extension of this work to color images. The latter would require estimation of chrominance quantization table(s) as well as chrominance subsampling.

8 Acknowledgements

The work on this paper was partially supported by Air Force Office of Scientific Research under the research grants number FA9550-08-1-0084 and FA9950-12-1-0124. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of AFOSR or the U.S. Government. The authors would like to thank Vojtěch Holub and Miroslav Goljan for useful discussions and Rainer Böhme for help with correctly implementing the WS attack.

References

1. R. Böhme. Weighted stego-image steganalysis for JPEG covers. In *Information Hiding, 10th International Workshop*, volume 5284 of LNCS, pages 178–194, Santa Barbara, CA, June 19–21, 2007. Springer-Verlag, New York.
2. R. Böhme. *Advanced Statistical Steganalysis*. Springer-Verlag, Berlin Heidelberg, 2010.
3. Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
4. T. Filler, T. Pevný, and P. Bas. BOSS (Break Our Steganography System). `http://www.agents.cz/boss/`, July 2010.
5. J. Fridrich and M. Goljan. On estimation of secret message length in LSB steganography in spatial domain. In *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 23–34, San Jose, CA, January 19–22, 2004.
6. J. Fridrich, M. Goljan, and R. Du. Steganalysis based on JPEG compatibility. In A. G. Tescher, editor, *Special Session on Theoretical and Practical Issues in Digital Watermarking and Data Hiding, SPIE Multimedia Systems and Applications IV*, volume 4518, pages 275–280, Denver, CO, August 20–24, 2001.
7. A. D. Ker and R. Böhme. Revisiting weighted stego-image steganalysis. In *Proceedings SPIE, Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, volume 6819, pages 5 1–5 17, San Jose, CA, January 27–31, 2008.
8. J. Kodovský and J. Fridrich. Steganalysis in high dimensions: Fusing classifiers built on random subspaces. In *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security and Forensics of Multimedia XIII*, volume 7880, pages OL 1–13, San Francisco, CA, January 23–26, 2011.
9. J. Kodovský, J. Fridrich, and V. Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, 7(2):432–444, April 2012.
10. A. B. Lewis and M. G. Kuhn. Exact JPEG recompression. In N. D. Memon, E. J. Delp, P. W. Wong, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Security and Forensics of Multimedia XII*, volume 7543, page 75430V, San Jose, CA, January 17–21, 2010.
11. W. Luo, F. Huang, and J. Huang. JPEG error analysis and its applications to digital image forensics. *IEEE Transactions on Information Forensics and Security*, 5(3):480–491, September 2010.

12. W. Luo, Y. Wang, and J. Huang. Security analysis on spatial ±1 steganography for JPEG decompressed images. *IEEE Signal Processing Letters*, 18(1):39–42, 2011.
13. T. Pevný. Detecting messages of unknown length. In N. D. Memon, E. J. Delp, P. W. Wong, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security and Forensics of Multimedia XIII*, volume 7880, pages OT 1–12, San Francisco, CA, January 23–26, 2011.
14. T. Pevný, T. Filler, and P. Bas. Using high-dimensional image models to perform highly undetectable steganography. In R. Böhme and R. Safavi-Naini, editors, *Information Hiding, 12th International Workshop*, volume 6387 of LNCS, pages 161–177, Calgary, Canada, June 28–30, 2010. Springer-Verlag, New York.
15. T. Pevný, J. Fridrich, and A. D. Ker. From blind to quantitative steganalysis. *IEEE Transactions on Information Forensics and Security*, 7(2):445–454, April 2012.
16. B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, 2001.
17. Y. Taro. Image coding apparatus and method, 2005. US Patent 6968090.