Batch Steganography in the Real World

Andrew D. Ker
Department of Computer Science
University of Oxford
Parks Road
Oxford OX1 3QD, UK
adk@cs.ox.ac.uk

Tomáš Pevný
Agent Technology Center
Czech Technical University in Prague
Karlovo namesti 13
121 35 Prague 2, Czech Republic
pevnak@gmail.com

ABSTRACT

We examine the universal pooled steganalyzer of [15] in two respects. First, we confirm that the method is applicable to a number of different steganographic embedding methods. Second, we consider the converse problem of how to spread payload between multiple covers, by testing different payload allocation strategies against the universal steganalyzer. We focus on practical options which can be implemented without new software or expert knowledge, and we test on real-world data. Concentration of payload into the minimal number of covers is consistently the least detectable option. We present additional investigations which explain this phenomenon, uncovering a nonlinear relationship between embedding distortion and payload. We conjecture that this is an unavoidable consequence of blind steganalysis. This is significant for both batch steganography and pooled steganalysis.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*information hiding*

General Terms

Security, Algorithms

Keywords

Batch Steganography, Anomaly Detection, Steganalysis, Embedding Strategies

1. INTRODUCTION

Steganography and steganalysis have been mainly concerned with simple abstractions of the data hiding problem, focusing on embedding of payload in *one* cover, or detection of payload in *one* object belonging to *one* user. The problems of *batch steganography* – how best to spread payload between multiple covers – and *pooled steganalysis* – how to pool evidence from multiple objects of suspicion – were posed in 2006 [11]. At the time it was proposed that an essentially game-theoretic situation would occur, with the optimal choice of payload spreading depending on the opponent's method for amalgamating evidence, and vice versa. Until recently there was little published research on these topics, but early analyses [11, 12] indicated that, for the embedder, the optimal behaviour is likely to be extreme concentration of payload into as few covers as possible, or the opposite in which payload is spread as thinly as possible. However, these theoretical results could not be confirmed without practical pooled steganalyzers to test against.

In 2011-12 [14, 15] we demonstrated a method for pooled steganalysis which treats actors (users who transmit cover or stego objects) as the unit of classification, measures distance between actors as a distributional difference between the feature clouds of their transmitted objects, and then applies *outlier analysis* to detect guilty (steganography-using) actors. Now that the literature is finally equipped with at least one practical method for pooled steganalysis, we can look for practical methods for batch steganography. We briefly discuss theoretical approaches to this problem in subsection 1.1.

The work reported in this paper was motivated by two aims. First, to extend the experimental evidence base for the pooled steganalyzer of [15], which so far has only been tested against one steganographic algorithm (nsF5), by testing against additional steganographic embedding algorithms, focusing on those with existing implementations which make them usable by a non-expert. Second, to test different *embedding strategies*[1], again with a focus on embedding strategies which could be used by a non-expert. After briefly summarising the blind pooled steganalyzer (section 2) we present the design of our experiments in section 3 and the results in section 4.

The authors found the consistency of the results, which favour concentration of payload in as few covers as possible, surprising. We performed further investigations to understand why they arise, which highlighted the significance of an embedding distortion which, for individual images measured by a popular steganalytic feature set, is *nonlinear* with respect to payload. Furthermore, feature preprocessing such as normalization or whitening makes a significant difference to this effect. These results are reported in section 5. Fi-

MM&Sec'12, September 6–7, 2012, Coventry, United Kingdom.

[1]To avoid confusion, the terminology *embedding algorithm* is used for the steganographic method for embedding in individual objects, and *embedding strategy* for the batch problem of how payload is allocated amongst multiple covers.

nally, section 6 concludes the paper with a discussion of these insights, and their importance for the design of both embedding strategies and future pooled steganalysis.

1.1 Optimal Embedding Strategy

When the batch steganography problem was presented, some theoretical approaches were tried. In [11] the detector is modelled as a single real output for each image (e.g. quantitative steganalyzer), combined into a pooled steganalysis using a number of approaches. In a rather restricted framework it is shown that the optimal embedding strategy is one of the two extremes: concentration of payload into the fewest number of covers, or spreading paying equally amongst all covers. A similar result is obtained in [12], for additional pooling methods, and another related result appears in [13]. All of these are limited in scope, and in particular it is assumed that the covers are homogeneous (equal capacity and sensitivity to embedding). This is not the case for the real world.

We can make use of more recent literature to determine an embedding strategy which is either optimal (if the definition of optimality is in terms of a distortion function) or at least near-optimal (if defined in terms of detectability) using adaptive embedding [4, 18]. Designed for embedding in a single image (or potentially a cover of any medium), adaptive embedding aims to concentrate the changes caused by embedding in those areas where they will be least detectable: in practice, in noisy and edge regions of images. Given a function which measures distortion, embedding which minimizes distortion must be located randomly following a Gibbs distribution, and the close approximation to this Gibbs embedding can be obtained using Trellis coding [2, 3, 5].

Given a set of cover images, in principle it should be possible to extend the distortion function to measure distortion in multiple images (perhaps by simple summation, perhaps by a more sophisticated definition) and apply Gibbs embedding to the entire set of images, performing at a stroke both batch allocation and embedding using a strategy which optimizes distortion. However, this is not straightforward: the definition of distortion should not necessarily be simple summation, particularly if the cover images are of different sizes or very heterogeneous; there is the underlying limitation that the optimality of the embedding depends on a close relationship between distortion and detectability; finally, there is as-yet no available implementation of Gibbs embedding in multiple images.

For this paper, it is the final limitation which causes us to exclude Gibbs embedding. We have chosen to focus on real-world steganography, such as would be possible by a non-expert using tools available today, addressing the following question: how would we advise a non-expert to hide data in multiple objects? This restricts us to embedding software which works on single images at a time (because no batch embedding has yet been implemented), to embedding algorithms that have available implementations, and to a set of cover images obtained from a real-world source. We assume that the embedder can use existing tools to determine the maximum capacity of individual images (see subsection 3.1) and manually split the payload into segments of their chosen size[2]. We can benchmark simple options against the only existing pooled steganalysis method, which is described in the following section.

2. BLIND UNIVERSAL POOLED STEGANALYSIS

Suppose that multiple actors each transmit multiple objects, all of which have been intercepted by the steganalyst. The steganalyst is assumed to know which actor sent which object. We assume that each actor used a source of cover objects, different from sources used by other actors. We do not require the steganalyst to have access to these sources, which makes it difficult to train traditional steganalysis methods. The steganalyst's aim is to identify a *guilty* actor or actors, who use steganography in some (not necessarily all) of their transmitted objects.

The first published method [14] for this scenario relied on hierarchical clustering. Its effectiveness was tested in laboratory conditions by simulating up to 13 actors by different digital cameras. Subsequent work [15] simulated realistic conditions by using a social network dataset with thousands of actors. The same work also proposed to replace hierarchical clustering by an outlier detection algorithm (*local outlier factor* [1]), as it argued that guilty actors represent outliers rather then tight clusters in the feature space. The latter method, used in this paper, works as follows.

First, we extract features from all available images. In principle, any steganalytic feature set might work, if sensitive to the steganographic algorithm used by guilty actors. (More precisely, it should be more sensitive to steganographic alteration than to innocent image processing operations.) In experiments presented in this paper, we have used the PF-274 feature set [19], since it offers good detection accuracy against the tested steganographic algorithms; it also has relatively low dimension. After extraction, the features are normalized to zero mean, unit variance, and optionally whitened and renormalized.

Second, we group the extracted features by actor, and calculate distances between all pairs of actors by using maximum mean discrepancy (MMD) [8]. In experiments presented here, we use the linear kernel for MMD, which corresponds to the L_2-distance between the mean feature vectors of individual actors. This choice is based on the experiments presented in [14]. The advantage of pooling objects from each actor is that the method improves the signal-to-noise ratio compared with working on individual objects. This leads to better accuracy in the identification of the guilty actor.

Third, we rank actors based on their pairwise distances calculated in the previous step. To do so, we use the local outlier factor method (LOF), because it has favourable properties for our application: it requires only pairwise distances between points (actors); it is application agnostic; it provides a measure of how much an outlier each point (actor) is; it relies on single hyper-parameter, to which the method is not particularly sensitive.

The LOF method provides a measure of being outlier for every actor, but does not provide a threshold, above which the actor should be considered to be an outlier. We simply use the LOF values to rank the actors according to their

[2]We will not consider how the sender informs the recipient of the lengths of each segment, or which image corresponds to which segment. It could be part of the shared secret key or a small header.

guiltiness. The evaluation criterion measures how often is guilty actor among the top $x\%$ most suspicious actors.

Some details of the LOF method are briefly described in Appendix A.

3. EXPERIMENTAL DESIGN

In this paper we evaluate the blind universal pooled steganalyzer in a variety of situations, testing different combinations of the following parameters: total number of actors, number of images per actor, embedding algorithm, embedding strategy, and total payload. For all the experiments the true number of guilty actors was fixed at one, for which the anomaly detector of [15] is best suited. The options for the other parameters are described in the following subsections.

3.1 Embedding Algorithms

Searching the internet for steganographic algorithms for JPEG images that can be readily used by ordinary users, we have found just five: F5 [24, 25], JPHide&Seek [17], Steghide [9, 10], OutGuess [21, 22], and JSteg [23].

OutGuess [21] is an adaptation of LSB replacement for JPEG images. It modifies DCT coefficients and skips coefficients equal to zero or one to avoid visible distortions due to changing zeros to ones. To evade detection based on first-order statistics, OutGuess saves half of the DCT coefficients for a statistical restoration phase, during which it tries to reconstruct the histogram of DCT coefficients of the original cover image. It is similar to **JSteg** [23], which we did not test because it is very weak and even more detectable than OutGuess [20].

F5 [24] tries to preserve the shape of the histogram of DCT coefficients. It encodes the message into LSBs of DCT coefficients, but instead of replacing the LSBs, it decreases the absolute values of DCT coefficients if their LSBs does not match the message. The F5 algorithm does not use zeros for embedding, and if a DCT coefficient is changed to zero during embedding then the message bit is re-embedded again utilizing another coefficient. This feature increases number of zeros in the stego image, which is called *shrinkage*. F5 also uses matrix embedding, a coding scheme that increases the embedding efficiency (the number of bits embedded per embedding change), reducing embedding distortion for small payloads.

Steghide [10] also tries to preserve first-order statistics like OutGuess, but without resorting to a statistical restoration phase. Steghide creates a graph-like structure with vertices representing (groups of) coefficients that need to be changed. Edges between vertices represents modifications in both groups, such that if performed, both vertices code the message. The edges are also assigned weights based on distortion of visual similarity caused by an embedding change. During embedding, the algorithm tries to find vertex matching in the graph, minimizing the distortion while coding the message.

Although the implementation in C of the **JPHide&Seek** algorithms are available, a higher-level description of its function is not known to the authors.

Besides the above four algorithms, we have included the **nsF5** algorithm, for which only a simulator has been released [16]. Although this is not in accord with our goal to investigate the security of practically available algorithms, the algorithm was included due to its popularity in the research community. The nsF5 algorithm uses the same embedding operation as F5, but replaces matrix embedding based on Hamming codes with wet paper codes with improved efficiency [6] to remove the shrinkage effect. Note that we used the 2008 version of the nsF5 simulator, which is not the same as the most recent version currently published by the author: the current version simulates a higher embedding efficiency.

3.2 Embedding Strategies

Recall the goal of batch steganography: the steganographer wants to spread a message of total length M among images $(X_1, \ldots, X_n)$ with capacities $(c_1, \ldots, c_n)$, using a steganographic embedding algorithm for individual images. (We will discuss how $(c_1, \ldots, c_n)$ are determined in the following subsection.) We have identified five strategies to determine the message fragment lengths $(m_1, \ldots, m_n)$, with $M = \sum_{i=1}^{n} m_i$, to be embedded into the images.

In the **max-greedy** strategy, the steganographer wants to embed the message into the fewest possible number of covers. During embedding, he iteratively chooses the covers with highest capacity yet to be used, and embeds a portion of the message equal to the capacity of the image. Assuming that the images are ordered by capacity $c_1 \geq c_2 \geq \ldots \geq c_n$, this leads to the following for the message lengths

$$\begin{aligned} m_i &= c_i, \ \forall i \in \{1, \ldots, I-1\}, \\ m_I &= M - \sum_{i=1}^{I-1} m_i, \\ m_i &= 0, \ \forall i \in \{I+1, \ldots, n\}, \end{aligned}$$

where I denotes the smallest possible number of images with sufficient capacity, i.e.

$$I = \arg\min_i M \leq \sum_{j=1}^{i} c_j.$$

The **max-random** strategy is the same as max-greedy, except that the covers used for embedding are chosen in a random order. Consequently, the number of utilized images can be higher then in the max-greedy strategy. This simulates a steganographer who is using a strategy of concentrating payload, but is not taking individual capacity into account in advance.

In the **linear** strategy, the message is distributed into all available covers proportionately to their capacity. This means that

$$m_i = \frac{c_i M}{\sum_{j=1}^{n} c_j}.$$

(Fractional bits are ignored in this study.)

In the **even** strategy, the message is distributed evenly into all available covers regardless to their capacity. Thus

$$m_i = \frac{M}{n}.$$

Executing this strategy it can happen that for some images, the message length m_i exceeds their capacity, c_i. In these cases, we set $m_i = c_i$ and recalculate an even message length for the remaining images.

In the **sqroot** strategy, the message is again spread among all images with the length of the fragments being propor-

	Mean estimated capacity	Mean relative capacity	Number of zero capacity covers
nsF5	53192	0.80	1265
F5	38427	0.57	0
JPHide&Seek	22133	0.33	0
Steghide	25249	0.38	3161
OutGuess	25654	0.38	121

Table 1: Average capacity of images in the social network image database, for the steganographic algorithms used in this paper. Capacities (bits) were estimated by the algorithm described in Section 3.3. The relative capacity (bpnc) is defined as the estimated capacity divided by the number of non-zero DCT coefficients in the image. Number of zero capacity images is out of 800 000.

tional to the square root of their capacities, i.e

$$m_i = \frac{\sqrt{c_i}M}{\sum_{i=j}^{n}\sqrt{c_j}}.$$

In our preliminary experiments we found that this so frequently exceeded the capacities that we discarded the strategy. (It is, in any case, based on a misreading of the square root law of capacity.)

3.3 Estimation of capacities

The embedding strategies described above assures the steganographer to know the maximum length (capacity c_i) he can embed into a particular image with the chosen embedding algorithm. In fact such a maximum is not always well-defined, since capacity can depend on content. Following our aim of simulating real-world practical steganography, we estimate the maximum message length for each embedding algorithm, and each cover image, as follows.

First, we query the implementation of the algorithm to provide an initial estimate of the maximum message length. This is done either by embedding a very short message into the given image (implementations of F5, JPHide&Seek, and OutGuess print the estimated capacity to the console upon embedding), or asking for information about a given image (the implementation of the Steghide). The implementation of nsF5 does not provide a capacity estimate, so we set it to $0.8 \cdot \mathrm{nc}$, where nc stands for the number of non-zero DCT coefficients.[3]

Once we have an initial estimate of capacity, we try to embed a randomly generated message of this length. If the embedding fails, the estimate of the capacity is decreased by 10 bytes and the procedure is repeated. Otherwise, we deem the current estimate of the capacity as final. We allow a maximum of 100 repetitions, after which the image is deemed as not suitable for the embedding and its capacity is set to zero.

The purpose of the last step is threefold: (a) it refines the initial estimates, (b) it verifies that the message can be actually embedded, and (c) it discards singular images (such as night pictures, blue skies, etc.).

[3]This capacity estimate of nsF5 is based on discussions with the author and our experiences from previous work; it does not seem to have been published.

In practice, the actual message length may also depend on contents of the the message itself, and on the steganographic key, because of slightly varying correlation between the cover and the message. To circumvent unpredictable behaviour, we decreased all capacity estimates to 90% of the original. Our capacity estimates are thus strictly conservative, but this should affect the results only slightly, as there should be little difference between embedding at total capacity and near-total capacity. The mean capacities, relative capacities, and number of zero capacity images (out of 800 000 in our image set, see below) are displayed in Table 1.

3.4 Real-World Images

Our experiments were performed on a highly realistic data set, obtained from a leading social networking site. Since the process of creating this data set has already been described in [15], we recapitulate it here briefly. All images uploaded to a popular social network and made publicly visible, by users who identified themselves with membership of Oxford University. After downloading by following public links, the files were anonymized and no personally-identifiable information was retained. All images have the same format (JPEG), with the same quality factor (85), and approximately the same size (1Mpix). Nonetheless they are very hard to steganalyze, because of an unknown and probably wild processing history: the social network automatically resizes and re-compresses uploaded images, following potential image processing operations between the camera and upload.

The different users will be using different cameras (potentially more than one each), which makes the set heterogeneous. Some images are not even natural photographs: they include montages, images with captions, or entirely synthetic advertisements. In a realistic scenario, we must deal with this type of difficult data. Pooled steganalysis can help to amplify a stego signal, but we must expect a certain variation between innocent actors.

For the experiments, we have used subset of this database, selecting exactly 200 photos from each of 4000 users ("actors"), who after anonymization are known only by an integer 1–4000. These 800 000 images form the data set used in this paper. It is a highly realistic image set because it is exactly the sort of media used on the internet in large social networks.

3.5 Experimental protocol

By using the social network data set, we simulate the scenario of monitoring a network and identifying the guilty users. We vary the number of actors, $n_a \in \{100, 400, 1600\}$, and number of images per actor, $n_i \in \{10, 20, 50, 100\}$. The actors and images were always selected randomly. All experiments discussed in Section 4 follow the protocol described here.

First, we must determine the size of the entire message M. Although its maximum depends on the embedding algorithm, in order to make like-for-like comparisons between different algorithms we need a fixed reference point. Since the *information* in a JPEG image is the nonzero coefficients, we measure payload relative to the total number of nonzero DCT coefficients. These are commonly referred to by the acronym "nc", hence we will measure total payload as the number of bits per nonzero coefficient, bpnc. Bearing in mind that algorithms such as OutGuess and Steghide can-

not reliably embed more than about 0.25 bpnc in a single image, we will choose payloads only up to this level. This also ensures that the message length does not exceed the total capacity, $M \leq \sum_{i=1}^{n} c_i$.

Then we can perform what we call "one experiment":

1. Randomly select n_a actors and n_i images per actor.
2. Randomly choose a guilty actor and embed random payload measuring p bpnc into his images using the chosen embedding strategy and embedding algorithm. The message length embedded into the images is equal to pN, where N is the total number of all non-zero DCT coefficients in the actors' images.
3. Extract features from images of all actors.
4. Normalize each feature to zero mean and unit variance.
5. Whiten the set of features to decorrelate them, discarding trivial components corresponding to eigenvalues of less than 0.01. This is achieved using the Principal Component Transform (PCT), after which the features are renormalized to unit variance. (The whitening step may be omitted.)
6. Group extracted features by actor.
7. Calculate distances between actors (linear MMD, i.e. the distances between the mean feature vector of each actor).
8. Calculate LOF values of every actor. The number of nearest neighbours, k, was set to 10, as in the publication [15].

For every combination of parameters, we have repeated the experiment 500 times (except 250 times for F5, which has a slow Java implementation). We must decide on a metric for success of the steganalyzer: we take the proportion of experiments in which the guilty actor was ranked in the top 5% most suspicious.

There are many combinations of parameters making up one experiment, each requiring the examination of hundreds of thousands of images. Performing all the experiments took approximately 230 core days, distributed over a small cluster.

4. RESULTS

Figure 1 displays the experimental results for $n_i = 100$ images per actor. The array of charts varies the number of actors (100, 400, 1600, horizontally) and embedding algorithm (vertically). Within each chart, the x-axis gives the payload in bpnc, and the y-axis the rate of success for the steganalyzer, when the guilty actor was ranked in the top 5% by LOF. The different embedding strategies are denoted by different point types.

There are some ways in which these charts show variation: larger payloads are unsurprisingly more detectable but the different embedding algorithms are of different security (of which more shortly), and it is somewhat easier to find a guilty actor in the top 5% of 1600 than the top 5% of 100 (a result replicating some in [15]). But most striking is the consistency between all these experiments: for any given payload size, where detection is not around random or perfect, the even embedding strategy is most detectable, the linear strategy next most detectable, and the max strategies least detectable. Max-greedy is less detectable than max-random. Similar consistency is seen with $n_i = 50, 20$, and even $n_i = 10$ although detection rates are much lower when there is so much less evidence available, and if the performance metric is changed to count the guilty actor in the top n most suspicious rather than top x%. We will not bore the reader with displays of such charts.

This represents an important lesson for a batch steganographer: it is substantially more secure to allocate payload into the smallest total number of covers, picking largest capacity covers and filling them completely. Depending on the detector and desired security, up to approximately twice as large a payload can be embedded using the max strategies than the linear or even strategies. Another way of looking at this data is to conclude that the blind universal steganalyzer has a weakness, which can be exploited by the embedder choosing the max-greedy strategy.

One other item we highlight from Figure 1 is the curious behaviour of JPHide&Seek: there is above-random detection even for the smallest payloads when even or linear strategies are used for embedding. It seems likely that JPHide&Seek embeds either a minimal message length or includes some sort of header, making it even more desirable to use as few images as possible for stego payload.

We make a comparison of the security of the different embedding algorithms, assuming that the best strategy (max-greedy) is used, in Figure 2. This confirms results seen in steganalysis of individual images, for example as in [7], where nsF5 is found to be the most secure choice, followed by F5 and JPHide&Seek, with Steghide substantially less secure and OutGuess the worst.

5. FURTHER INVESTIGATIONS

We want to understand *why* the results of section 4 occur. We focus on three particular questions:

(1) Why do the strategies which concentrate payload (max-greedy and max-random) evade detection substantially better than those which spread payload (linear and even)?

(2) Why do the strategies which use image capacity in the allocation (max-greedy and linear) evade detection better than those which do not (max-random and even)?

(3) Do the results represent a weakness of the blind universal pooled steganalyzer, which can be fixed?

First, we can examine the problem by recalling that the steganalyzer depends only on the L_2-distance between the centroids of each actor's feature cloud (the PF-274 features extracted from the entire set of images they transmit). What matters to detectability, apart from the inner workings of the LOF calculation, is how the the guilty actor's centroid is affected by embedding.

Let us fix a guilty actor, and write $v_1, \ldots, v_n$ for the feature vectors which are extracted from the *cover* images they use. Correspondingly, let us write $v'_1, \ldots, v'_n$ for the feature vectors extracted from their transmitted objects. We continue to write $m_1, \ldots, m_n$ for the payload allocated into each image by the guilty actor's strategy. Thus if $m_i = 0$ then $v'_i = v_i$, otherwise very likely $v'_i \neq v_i$. Write $\bar{v}$ and $\bar{v}'$ for the guilty actor's centroid of cover and stego images respectively.

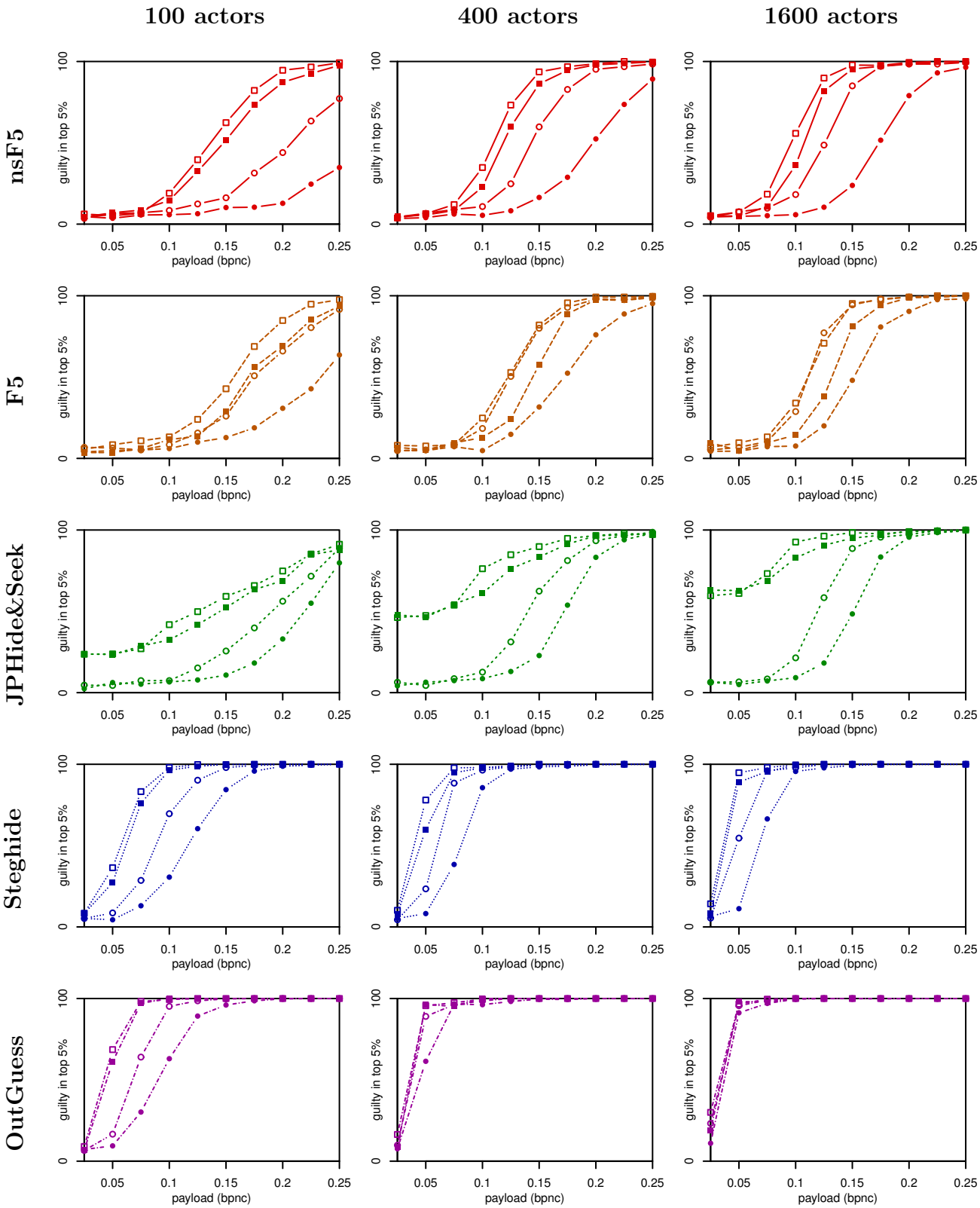

Figure 1: Accuracy of the blind pooled steganalyzer, for five different embedding algorithms (charts vertically), number of actors (charts horizontally), total payloads in bpnc (x-axis of each chart), and embedding strategy: max-greedy (●), max-random (○), linear (■), even (□). The y-axis is the proportion of experiments in which the true guilty actor was ranked in the top 5% most suspicious actors. In all these charts, the number of images per actor $n_i = 100$.

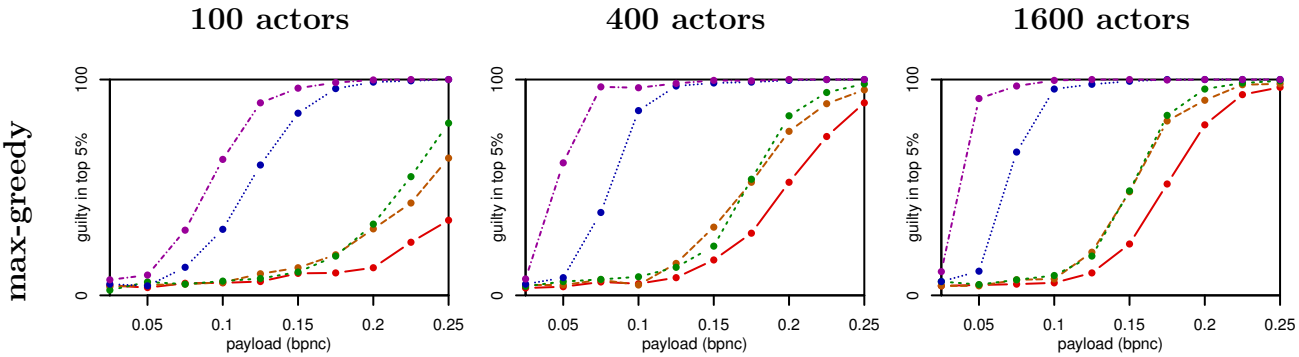

Figure 2: Comparison of different embedding algorithms, which follow the same colours and line style as Figure 1.

Despite copious literature making use of these (and other) features for classification of payload, we know little about how payload size effects features. We know that it *does* affect features, because steganography is detectable, and from [20] we know that it is possible to construct an estimator for the (relative) payload length from the stego features. In [20] it is shown that even an ordinary least-squares estimator has quite good performance, which would suggest a broadly linear relationship between features and payload size.

So suppose that, abusing mathematical notation to reason roughly about average distortion, the effect of payload on features is approximately linear, i.e.

$$v_i' \approx v_i + m_i \delta_v$$

where δ_v is a direction in which stego objects tend to move, then

$$\bar{v}' \approx \bar{v} + \bar{\delta_v} \sum m_i.$$

This of course neglects many details, including if stego objects move in different directions or at different rates depending on the cover or message (though this should all wash out on average). However, it at least leads us to expect that *all embedding strategies should be equally detectable*, as they cause equal distortion to the guilty actor's centroid when the total payload is fixed.

Since this is manifestly not the case, our first investigation into the cause of (1) and (2) is to test the assumption of a linear relationship between payload and distortion in individual images. We picked an embedding algorithm and a random cover image, and computed three feature vectors:

- v_0, the cover features;
- $v_{0.1}$, features from the image with a fixed payload of 10% of its capacity;
- v_p, features from the image with a random-length payload, proportion p of its capacity.

On the left of Figure 3 we display scatterplots of

$$\frac{\|v_p - v_0\|}{\|v_{0.1} - v_0\|} \tag{1}$$

against p. If the relationship between feature change and (relative) payload is linear then we would expect to see a straight line. The use of a fixed payload in the denominator may create an artificial "pinch" near 0.1, but means that the results can be displayed with a comparable y-axis.

We observe some sort of linear fit, and no sign of a significantly nonlinear fit, although the extent of the relationship depends on the embedding algorithm. (It shows that the least-squares quantitative estimator is doing some work, weighting towards relevant features and away from noisy features, in order to have such good performance displayed in [20].) We do not see a nonlinear relationship.

The authors found this puzzling, because it does not fit with the observed behaviour of the different batch embedding strategies. But then we remembered that the LOF detector works on features which, in order to equalize their weight and remove correlation, have been normalized, whitened, and then renormalized (we call this "normwhite space"). So we performed an identical experiment but computing the norms of (1) on features which are first subject to the same operations. These are displayed in the middle of Figure 3.

This time the results are markedly different. There is a strong nonlinearity, with larger payloads causing proportionately less distortion than smaller ones. Observing the values on the y-axes, we see that payloads of near-full capacity cause distortion only about 3–4 times as large as payload at 10% of capacity. This phenomenon directly explains why concentration of payload in few covers is the best type of batch steganography against our universal pooled detector, since the total distortion "cost" is much lower thanks to the sublinear relationship.

How is it possible for the left and middle columns of Figure 3 to be so different, and how is a nonlinear relationship compatible with the workings of linear least squares regression in [20]? To answer the first, consider that normalization is a shear transform geometrically and (unlike PCT, which is orthonormal) can magnify or reduce the angles between vectors; vectors which are nearly parallel (such as stego distortion in these images, a fact which we verified but do not include here for reasons of space) can become much less parallel when subject to shear, and that is exactly what happens to stego features. To answer the second, we must simply expect that the regression manages to find a linear transform which does the opposite, making stego distortion nearly parallel and thus reducing the nonlinear effect of payload.

We performed a number of investigations to try to explain the nonlinear relationship between (whitened, normalized) features and payload. They revealed that, perhaps

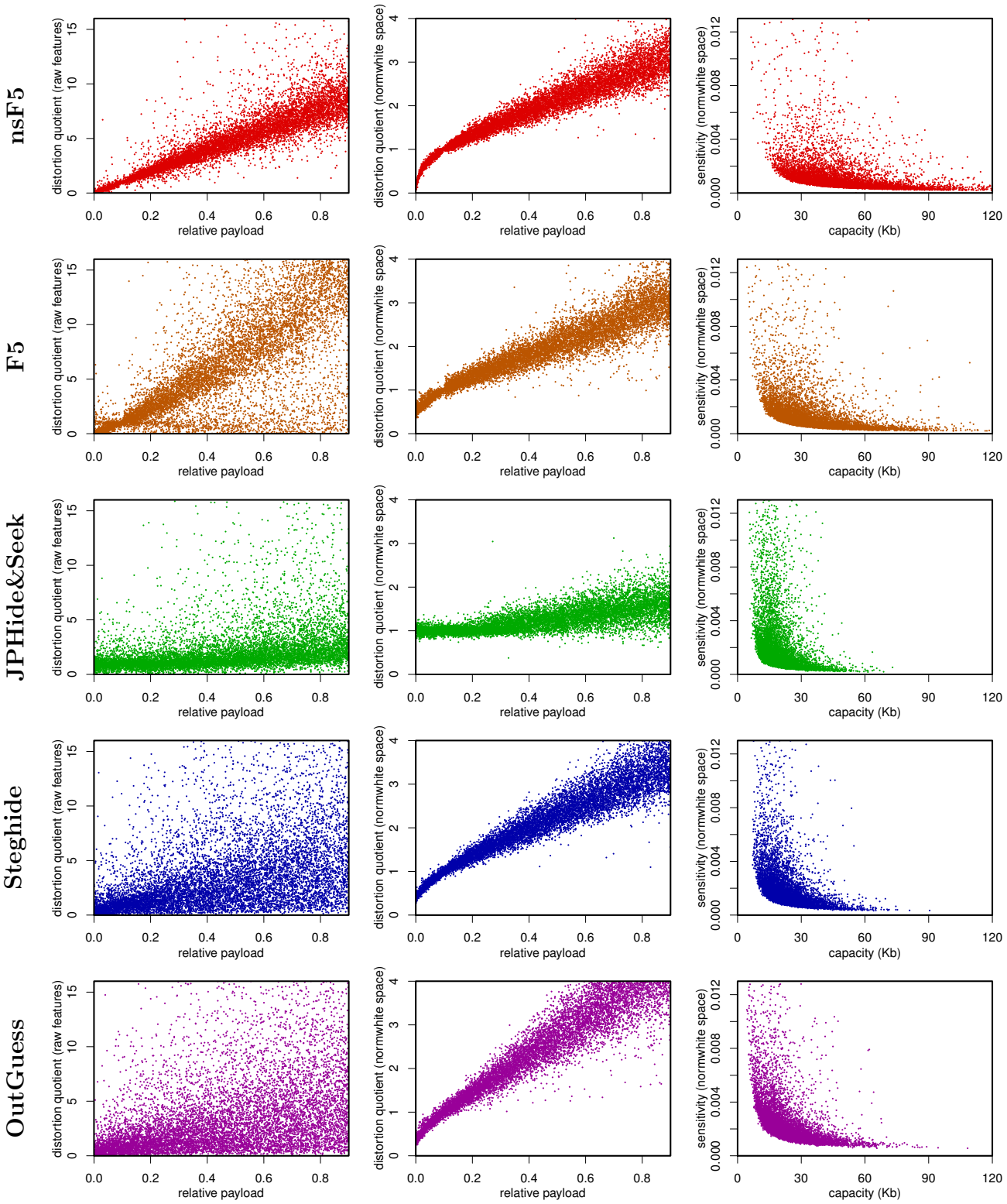

Figure 3: Further experiments to understand the optimality of concentrated payload. Five different embedding algorithms, vertically. Left: relative payload versus feature distortion (relative to distortion at 0.1 bpnc) in individual images. Middle: the same, but distortion calculated for normalized whitened features. Right: capacity versus sensitivity to payload, in individual images. Each plot is from 10 000 images taken at random from the entire database of 800 000.

unsurprisingly, not all features contain useful information about payload. After whitening we should talk of "components" rather than "features", since each component is a linear combination of features. For components with much information, for example those with the largest eigenvalues from PCT, the addition of payload causes apparently-linear movement in one direction or the other. But particularly for components with small eigenvalues from PCT, the addition of payload simply causes noise, with the value moving in no particular direction. Now consider that

$$\|v_p - v_0\| = \sqrt{(c_p^1 - c_0^1)^2 + \cdots + (c_p^m - c_0^m)^2} \qquad (2)$$

where c_p^i is the i-th component of features from an image with payload p. If some of these differences $c_p^i - c_0^i$ consist only of noise, (2) boils down to $\sqrt{cp^2 + d}$, where c and d have positive expectation. The explains the shapes seen in Figure 3. Essentially it stems from the inclusion of some noisy features/components which are not informative about steganography. However, it seems impossible for a *blind* detector to avoid this, since it cannot remove non-informative features without some stego information.

Although the nonlinear phenomenon explains why the max strategies are more secure than linear or even, we also want to answer question (2), above. We can explain this quite simply, by looking at the rate at which $\|v_p - v_0\|$ grows with p. In the third column of Figure 3, we plotted the capacity of each image against $\|v_p - v_0\|/p$. The latter quantity we call *sensitivity*. Clearly, images with larger capacity have less sensitivity (the statistical significance of the relationship is extremely strong for each embedding algorithm). Thus there is an additional advantage, as well as concentration in as few covers as possible, of the max-greedy strategy because it picks the least sensitive covers[4].

Finally, we turn to question (3): can the detector be improved in light of our new understanding? The authors think not: there is no solution to restoring a linear payload relationship by avoiding whitening/normalization, because this ruins the scales of the features and makes the LOF an incorrect measure of outliers. Although it makes the performance of the different embedding strategies more equal, it does so by lowering detectability of all of them (charts not included for reasons of space).

6. CONCLUSIONS

We have tested the LOF-based anomaly detector from [15] more widely, to show that it works against different embedding algorithms and strategies, that it is quite sensitive to payloads of around 0.1bpnc or lower, and hence real-world steganographic algorithms (which excludes nsF5) are rather insecure. The consistent lesson is that a greedy embedding strategy, which concentrates payloads in as few covers of largest possible capacity, is able to exploit a property of the detector. The property is due to a nonlinear relationship between (unavoidably normalized) features and payload size, which is an important insight for both embedders and detectors.

We should verify that this is not merely a result for the PF-274 feature set we have used in this work and it would be helpful to develop a hypothesis test for the phenomenon. This is a matter for future work.

We have also exposed a gap in the research on steganalysis features, to understand their geometric structure and the effects of steganographic payload beyond merely the existence of quantitative estimators. Further work is needed to determine whether near-linearity of stego distortion can be increased and exploited.

We must stress that much of the quasi-analysis of section 5 arises because of the use of linear MMD in the LOF distances. However, the only examination of an alternative (Gaussian kernel, [14]) showed weaker detection performance. In theory, a nonlinear detector can have power proportional to the square of the payload size, at least locally (essentially the reason for the square root law of steganographic capacity), so further researching into alternative MMD kernels is important. Another line of research would be to replace LOF with a method more robust to noisy features.

Finally, we might conjecture that all blind steganalysis will have a similar weakness: features with pure noise cannot be removed without knowledge of some stego data, leaving a distortion which must always be nonlinear. It would be interesting to formalize this and perhaps prove the general optimality of greedy embedding. From the detector's side, we suggest that blind detection could be augmented by non-blind feature selection, in which the feature set is tuned in the direction of known stego algorithms.

[4]It would not be in keeping with the philosophy of this paper to allow an embedder directly to measure sensitivity of each image in their embedding strategy, since it would require them to know the feature set used by their opponent.

7. ACKNOWLEDGMENTS

The work on this paper was supported by European Office of Aerospace Research and Development under the research grant number FA8655-11-3035. 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 EOARD or the U.S. Government.

The work of T. Pevný was also supported by the Grant Agency of Czech Republic under the project P103/12/P514.

8. REFERENCES

[1] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *Proc. 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD, pages 93–104. ACM, 2000.

[2] T. Filler and J. Fridrich. Gibbs construction in steganography. *IEEE Transactions on Information Forensics and Security*, 5(4):705–720, 2010.

[3] T. Filler and J. Fridrich. Minimizing additive distortion functions with non-binary embedding operation in steganography. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6, 2010.

[4] T. Filler and J. Fridrich. Design of adaptive steganographic schemes for digital images. In N. D. Memon, J. Dittmann, A. M. Alattar, and E. J. Delp III, editors, *Media Watermarking, Security, and Forensics XIV*, volume 7880 of *Proc. SPIE*, page 78800F. SPIE, 2011.

[5] T. Filler, J. Judas, and J. Fridrich. Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6(3):920–935, 2011.

[6] J. Fridrich, M. Goljan, and D. Soukal. Wet paper codes with improved embedding efficiency. *IEEE Transactions on Information Forensics and Security*, 1(1):102–110, 2006.

[7] J. Fridrich, T. Pevný, and J. Kodovský. Statistically undetectable JPEG steganography: Dead ends challenges, and opportunities. In *Proc. 9th ACM Workshop on Multimedia and Security*, MM&Sec, pages 3–14. ACM, 2007.

[8] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample problem. pages 513–520, 2007.

[9] S. Hetzl. Implementation of the Steghide algorithm ver. 0.5.1 (released October 2003). `http://steghide.sourceforge.net/`, last accessed April 2012.

[10] S. Hetzl and P. Mutzel. A graph-theoretic approach to steganography. In *Proc. 9th International Conference on Communications and Multimedia Security*, CMS, pages 119–128. Springer, 2005.

[11] A. D. Ker. Batch steganography and pooled steganalysis. In J. Camenisch, C. Collberg, N. Johnson, and P. Sallee, editors, *Proc. 8th Information Hiding Workshop*, volume 4437 of *LNCS*, pages 265–281. Springer, 2006.

[12] A. D. Ker. Batch steganography and the threshold game. In E. Delp III and P. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505 of *Proc. SPIE*, pages 0401–0413. SPIE, 2007.

[13] A. D. Ker. Perturbation hiding and the batch steganography problem. In K. Solanki, K. Sullivan, and U. Madhow, editors, *Proc. 10th Information Hiding Workshop*, volume 5284 of *LNCS*, pages 45–59. Springer, 2008.

[14] A. D. Ker and T. Pevný. A new paradigm for steganalysis via clustering. In N. Memon, J. Dittmann, A. Alattar, and E. Delp III, editors, *Media Watermarking, Security, and Forensics XIII*, volume 7880 of *Proc. SPIE*, pages 0U01–0U13. SPIE, 2011.

[15] A. D. Ker and T. Pevný. Identifying a steganographer in realistic and heterogeneous data sets. In N. Memon, A. Alattar, and E. Delp III, editors, *Media Watermarking, Security, and Forensics XIV*, volume 8303 of *Proc. SPIE*, pages 0N01–0N13. SPIE, 2012.

[16] J. Kodovský. simulator of the nsF5 algorithm with wet paper codes (released 2008). `http://dde.binghamton.edu/download/nsf5simulator/`, last accessed April 2012.

[17] A. Latham. Implementation of the JPHide and JPSeek algorithms ver 0.3 (released August 1999). `http://linux01.gwdg.de/~alatham/stego.html`, last accessed April 2012.

[18] T. Pevný, T. Filler, and P. Bas. Using high-dimensional image models to perform highly undetectable steganography. In P. W. L. Fong, R. Böhme, and R. Safavi-Naini, editors, *Proc. 12th Information Hiding Workshop*, volume 6387 of *LNCS*, pages 161–177. Springer, 2010.

[19] T. Pevný and J. Fridrich. Merging Markov and DCT features for multi-class JPEG steganalysis. In E. J. Delp III and P. W. Wong, editors, *Media Watermarking, Security, and Forensics IX*, volume 6505, pages 03–14. SPIE, 2007.

[20] T. Pevný, J. Fridrich, and A. D. Ker. From blind to quantitative steganalysis. *Information Forensics and Security, IEEE Transactions on*, 7(2):445–454, 2012.

[21] N. Provos. Defending against statistical steganalysis. In *Proc. 10th Conference on USENIX Security Symposium - Volume 10*, SSYM, pages 323–335. USENIX Association, 2001.

[22] N. Provos. Implementation of the OutGuess algorithm ver. 2.0 (released October 2001). `http://www.outguess.org/`, last accessed April 2012.

[23] D. Upham. Implementation of the JSteg steganographic algorithm. `http://zooid.org/~paul/crypto/jsteg/`, last accessed on April 2012.

[24] A. Westfeld. F5-a steganographic algorithm. In I. Moskowitz, editor, *Proc. 4th Information Hiding Workshop*, volume 2137 of *LNCS*, pages 289–302. Springer, 2001.

[25] A. Westfeld. Implementation of the F5 steganographic algorithm (released May 2011). `http://code.google.com/p/f5-steganography/`, last accessed April 2012.

APPENDIX

A. LOCAL OUTLIER FACTOR

Suppose that we are given a set P of points, with a metric $d : P \times P \to \mathbb{R}$ and an integer parameter $1 < k < |P|$.

For this exposition we assume no exact duplicates in P or exactly tied distances between members of P, which simplifies the description. For full details, see the original publication [1].

The **reachability distance of point p from q**, $r_k(p, q)$, is the greater of $d(p, q)$ and $d(q, q')$, where q' is q's k-nearest neighbour. Compared to the metric d, the reachability distance reduces statistical fluctuations for close objects, with smoothing controlled by the parameter k.

Fix a point p, and write P_k for the k-nearest neighbourhood of p. The **local reachability density of p** is defined as an inverse of the average reachability distance of point p from all points $q \in P_k$,

$$lrd_k(p) = k\left(\sum_{q \in P_k} r_k(p, q)\right)^{-1},$$

and the **local outlier factor (LOF) of p** is

$$lof_k(p) = \frac{1}{k} \sum_{q \in P_k} \frac{lrd_k(q)}{lrd_k(p)}.$$

Thus $lof_k(p)$ captures the degree to which p is further from its k-nearest neighbours than they are from theirs. Defining it as a relative number makes the method adaptive in the sense that (a) it does not depend on absolute values of distances $d(p, q)$ and (b) outliers can be detected in dense as well as in sparse regions of P.