The Steganographer is the Outlier: Realistic Large-Scale Steganalysis

Andrew D. Ker, *Member, IEEE,* and Tomáš Pevný

***Abstract*—We present a method for a completely new kind of steganalysis to determine who, out of a large number of actors each transmitting a large number of objects, is hiding payload inside some of them. It has significant challenges, including unknown embedding parameters and natural deviation between innocent cover sources, which are usually avoided in steganalysis tested under "laboratory conditions". Our method uses standard steganalysis features, the Maximum Mean Discrepancy measure of distance, and ranks the actors by their degree of deviation from the rest: we show that it works reliably, completely unsupervised, when tested against some of the standard steganography methods available to non-experts. We also determine good parameters for the detector and show that it creates a two-player game between the guilty actor and the steganalyst.**

I. Introduction

STEGANALYSIS aims to detect the presence of hidden payload inside apparently-innocent covers. Although a refined discipline, particularly when the covers are still images, no research has yet considered how to detect payload when monitoring an entire network. In such a case the detector will see vast numbers of objects, transmitted by a variety of users each of whom uses slightly differing sources, the embedding methods used by "guilty" users may be unknown, and the amount of payload almost certainly is unknown. Such a situation is completely different to the "laboratory conditions" found in most steganalysis experiments, and the challenges are different from classifying an individual object as cover or stego. We address them in this paper.

After briefly surveying the state of art in steganalysis of individual objects (Subsection I-A), we explore the requirements of large-scale steganalysis (Section II). We then propose a new steganalysis paradigm (Section III), which differs from conventional steganalysis in two main ways: it takes as its unit the *actor* (network or social network user, or cover object source) rather than the single object, and it performs anomaly detection rather than classification, calibrating its expection of actors by the behaviour of the majority. This means that the method is entirely unsupervised, and robust to the challenges of large-scale steganalysis.

We then perform large-scale experiments, using a real-world social networking image set and steganography accessible to the non-expert (Section IV), to demonstrate that the method works robustly on a number of embedding algorithms (Section V), and to tune some of its parameters (Section VI). The experiments are performed using well-established embedding methods with available implementations, and use a well-established feature set: neither of these represents the academic state-of-art, but we stress that the contribution of this paper is the framework for large-scale steganalysis, rather than a detector using particular features. By testing in well-understood circumstances, we eliminate potential complications from the latest embedding methods and detection features. Finally, we conclude with discussion of many future directions for this line of research (Section VII).

This work extends our two previous conference papers on this subject [17], [18]: as well as new metrics for accuracy, we additionally evaluate a number of different parameters for the large-scale detector, uncover a game between the embedder's strategy and the detector's optimal behaviour, and compare to the (scarce) relevant prior art.

A. State of the Art in Binary Steganalysis

The contemporary approach to steganalysis involves three components. It extracts, from each object under examination, steganalytic *features* of high dimension; it supplies training sets of cover and stego objects; and it runs a machine learning algorithm on the training data. This creates a decision function for novel objects, classifying them as cover or stego, occasionally with some sort of associated level of confidence.

Recent feature sets for the domain of still images, where the literature is most advanced, comprise thousands of relatively weak features [7], [8], [19]. It seems that there may be a linear relationship between number (or rate) of embedding changes and the position of feature in the feature space, since linear classifiers [21], [24] are sufficient to devise very accurate detectors.

Some drawbacks of this approach are that detectors are targeted towards a given steganographic algorithm and payload

A. Ker is with the Department of Computer Science, Oxford University, Oxford OX3 0AU UK e-mail: adk@cs.ox.ac.uk.

T. Pevný is with the Agent Technology Center, Czech Technical University in Prague, Karlovo náměstí 13, 121 35 Prague 2, Czech Republic e-mail: pevnak@gmail.com.

The work on this paper was supported by European Office of Aerospace Research and Development under the research grant numbers FA8655-11-3035 and FA8655-13-1-3020. 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.

Manuscript received January 9, 2014; revised April 22, 2014; accepted June 6, 2014

size [26] (used to create the training stego set), and to a particular source of covers. If the objects under scrutiny come from a different source (e.g. camera or settings) from the training data, the accuracy of detectors decreases, often dramatically [1]. This phenomenon, a cover-source mismatch, is unavoidable in reality unless the suspected steganographer is considerate enough to supply their enemy with their cover source.

Another drawback is that the design is for a decision function for single objects, which would only be applicable in the case when a steganalyst is presented with a small amount of data to classify individually. It does not address the challenges of large-scale steganalysis.

II. Large-Scale steganalysis

Suppose that a steganalyst is monitoring a large network, with multiple users and many potentially suspect communications. For example, they might be scanning all the images on a social media site for hidden content, or acting as a corporate firewall to prevent data exfiltration. We identify four requirements for a steganalysis system:

Universality. The steganalyst may not know what steganography algorithm is being used on the network. As much as possible, their detector should be able to identify unknown or new embedding methods, with unknown sizes of payload. Most existing steganalysis methods do not have this property (see Subsection III-D for a survey of prior art).

Robustness. If the steganalyst has some training data, they cannot ensure that it comes from an identical source to that used by the actors they are monitoring. As much as possible, their detector should not suffer unpredictably from cover-source mismatch. Again, existing steganalysis methods generally fail this condition [27].

Multiple actor. The network has multiple users, some (most) of them innocent of steganographic embedding, but each with a slightly different cover source. The detector needs to determine *who* is guilty, not necessarily which of their objects specifically contain payload. No previous steganalysis methods have considered this case. Exactly what output is required depends on the situation: it might be known that at least one guilty actor exists, or not, and it might be required to obtain a probability of guilt for each or simply a ranking. In this paper we assume that it is sufficient to rank the actors in order of likeliness of guilt.

Multiple object. Each actor emits many objects. For innocent actors, all of their objects are plain covers. For guilty actors, some (not necessarily all) of them contain payload. The detector must aggregate the evidence in the objects, and this is the *pooled steganalysis* problem from [14], which has not yet been addressed successfully.

We also require a certain computational efficiency, ideally linear in the number of objects captured from the network. Large-scale monitoring, in real-world scenarios, may have to cope with vast amounts of intercepted data.

Naturally, we expect some penalty for universality and robustness: a hypothetical detector for single images with these properties would likely be inferior to existing binary classification steganalysis when tested under laboratory conditions (known algorithm and payload size, no mismatch). However, we are able to turn the large-scale situation, with multiple actors and objects, to our advantage. More evidence is available: of individual actor's guilt, of the behaviour of innocent actors, and (crucially) of how much innocent actors sources tend to differ from each other. As with the original pooled steganalysis problem, the difficulty is how to aggregate the evidence.

III. Detecting Anomalous Actors

Our proposed detector identifies actors that significantly deviate from the majority. We assume the scenario of multiple actors each emitting multiple objects, all of which are seen by the detector, who also knows which actor sent what. In the discussion below, we assume that the objects are digital images, but the same system could be used for any domain with good steganalytic features.

The detector works in three steps: first, extracting standard steganalytic features from all objects; second, calculating distances between each pair of *actors* based on the cloud of feature points that they have emitted; third, identifying actors deviating from the majority using an anomaly measure computed from the distances. If steganalytic features are sensitive to hidden payload, and relatively insensitive to other characteristics of the objects, then an actor's deviation is evidence of their guilt: the steganographer is the outlier. These three steps are now described in detail, with a discussion of the design choices to be made by the steganalyst.

A. Features

The detector extracts features from every image transmitted by every actor. The steganalyst's first design decision is to select a suitable feature set: in theory, a detector should work with any steganalytic features sensitive to embedding changes and relatively insensitive to image content.

In experiments performed in this paper, which use JPEG images, we have chosen so-called PF274 features [28], because they reliably detect the steganographic algorithms used (described in Subsection IV-B), their extraction is fast, and they have good signal to noise ratio. In other work we have shown that the detector works with high dimensional features as well [20], but due to their sensitivity to image content, they have to be made robust with respect to it [31].

Once features are extracted from all images, the steganalyst must pre-process them, to make the contribution of each feature equal and hence the distance (below) meaningful. We determined (see Section VI-A) that a global whitening works best. The whitening projects features into a new space, of slightly lower dimension, where features are uncorrelated and they have unit variance in each direction. The base of the projection space is found by eigenvalue decomposition of the features' covariance matrix (the same operation is used in principal component analysis) calculated from all images. For numerical stability, projections with corresponding eigenvalues smaller than 0.01 (see Section VI-A) are discarded.

B. Distance Between Actors

We propose to measure distance between actors using an empirical Maximum Mean Discrepancy (MMD) [9], which is a measure of similarity between probability distributions. It has the useful property that it can be estimated robustly, even for high dimensional probability distributions, from relatively little data. MMD corresponds to an L_2 distance in some Hilbert space implicitly defined through a positive definite kernel function $\kappa(x, y) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ (if pre-processed features are real and of dimension d). Popular kernels include the linear kernel $\kappa(x, y) = x^{\mathrm{T}} y$ and the Gaussian kernel $\kappa(x, y) = \exp(-\gamma \|x - y\|^2)$, where γ is the inverse kernel width. Assuming n samples $\{x_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$, pre-processed feature vectors from actors X and Y, a sample estimate of the MMD distance has the following simple form

$$MMD(X, Y) = \frac{1}{\binom{n}{2}} \sum_{1 \le i < j \le n} \kappa(x_i, x_j) - \kappa(x_j, y_i) - \kappa(x_i, y_j) + \kappa(y_i, y_j). \tag{1}$$

An adjustment can be made when the number of samples is different between X and Y, for which we refer to the original publication [9].

The above formula requires $O(n^2)$ computations, where n is the number of images emitted by each actor. This is undesirable for large-scale application, but a simple approximation is available: in case of a the linear kernel, MMD converges with $n \to \infty$ to $\|\bar{x} - \bar{y}\|_2^2$, the L_2 norm between the actors' centroids $\bar{x}$ and $\bar{y}$ in the feature space (see Appendix B). We call this the centroid 'kernel' (it is not really a kernel, but an asymptotic approximation) and use it extensively in our experiments because it can be computed in $O(n)$ time. The influence of the MMD kernel on the quality of detection is studied in greater detail in Section VI-B.

C. Anomaly Detector

Once distances between actors are calculated, we identify outlying actors. From the plethora of outlier detection methods [3], we have chosen the local outlier factor (LOF) method [2], as it has several desirable features: (a) it detects outliers in probability distributions with clusters of different densities; (b) the provided anomaly score is interpretable, as values around one corresponds to true 'inliers' and values greater than two correspond to outliers. Given a set P of points (actors) with a metric $d : P \times P \to [0, \infty)$ and an integer parameter $1 < k < |P|$, the LOF is calculated as follows.[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 with 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 in P. The **local reachability density of** p is defined as an inverse of the average reachability distance of point p from all points $q \in P_k$,

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

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 means that it does not depend on absolute values of distances $d(p, q)$.

The original publication recommends $k = 10$, and we have used this value throughout all experiments except in Section VI-C. The results will show that the optimal value of k depends on the number of guilty actors and their embedding strategies. The LOF calculation is quadratic in the number of actors (as it must compute and rank all pairwise distances) which is slightly undesirable, but the number of actors is likely to be orders of magntitude smaller than the number of images.

By design, this detector works (only) for a multi-actor, multi-image scenario. And because it is completely unsupervised, it cannot suffer from mismatch between training and testing data. Thus our requirement of robustness is automatically met. It remains to demonstrate that it works and has good universality.

D. Relation to Prior Art

The vast majority of published work on steganalysis attacks a different problem: analysis of one image at a time. The first work proposing to investigate multiple images from a one actor was [14]: it describes different strategies of aggregating detection results from individual images to find whether one actor is guilty or not. Although the work assumes a targeted detector for a single image, it could be used with universal steganalyzer as well. It does not consider the scenario of multiple actors.

Universal steganalysis, where the steganalyst does not know the embedding algorithm, is a largely neglected field of research. Probably the first work in this field was [25], which modelled distribution of cover images by one-class SVM and classified deviations as stego images. The problem was further studied in [30], where it is shown that universal steganalyzers are sensitive to what is now called cover-source mismatch.

To the best of our knowledge, the large-scale steganalysis scenario described here has not been attacked at all, except in our prior work [16]–[18]. Combining the universal single-image steganalyzer [30] with aggregation methods published in [14] is the only prior art we can find. We compare our detector to it in Section V-B.

IV. Simulating the Real World

We wish to validate this new detection paradigm, in a situation which mimics as much as possible a real-world network scanning problem. We therefore selected covers, steganographic embedding methods, and strategies for guilty actors

[1] For this exposition we assume no exact duplicates in P or exactly tied distances between members of P, which simplifies the description considerably. For full details, see the original publication [2].

to allocate payload between covers, to mimic a hypothetical steganographer inserting payload into social media images.

A. Cover Images

The images were obtained from a leading social network site, which is popular for sharing pictures. Such a service could provide an ideal steganographic channel, because uploading and downloading many images should not raise suspicion. We used a web crawler to download all *public* images from users who identified themselves as members of Oxford University. We stopped after downloading more than 4 million images from more than 70 000 users. All personally identifiable information was removed, and the files anonymized except for grouping images uploaded by the same user. The *actors* in our experiments are the uploaders, which mimic well the behaviour of real-world actors: sometimes a single actor uses two or three cameras. In the experiments described here, we used a randomly selected subset of 4000 actors and 200 images for each actor, for a total of 800 000 images.

At the time of crawling, the social networking site automatically resized large images, to approximately 1Mpix, and then JPEG compressed them with quality factor 85. This simplifies steganalysis, since it is known that steganalytic features are very sensitive to different quantisation matrices [29], but does introduce a second compression (if the files were originally uploaded as JPEGs); double compression is usually considered a difficult nuisance parameter in steganalysis [29].

Apart from a fairly uniform size and completely uniform quality factor, the images in the database are very diverse, as they (a) come from different sources (cameras, flatbed scanners), (b) are of different types (indoor party pictures, cities, outdoor nature scenes, etc.), and (c) underwent different image processing from acquisition to download. Some of them are not natural images at all, but synthetic images or mosaics. Most steganalysis literature would perform experiments on images with such "impurities" removed, but we did not remove them: the impurities are there in practice and these images are a good prototype for what might be expected when monitoring a real network. Their proportion in the database should reflect the proportion we can expect in the wild, since it was crawled from a real-world source.

B. Embedding Algorithms

In our experiments, we have used the following five steganographic algorithms: F5 [38], [39], F5 with shrinkage removed by wet paper codes and matrix embedding turned off (nsF5), JPHide&Seek [23], OutGuess [32], [33], and Steghide [10], [11]. These algorithms have diverse embedding mechanisms, software implementations are all publicly available (except for nsF5 where only a simulator exists), and they do not utilise side information in the form of the raw image. Thus they could be applied by a non-expert. Furthermore, there is copious evidence that these embedding methods can be detected by the chosen steganalytic feature set. Below, the ideas behind each algorithm are briefly described. For further details we refer to the original publications.

OutGuess [32] is an improved version of **JSteg** [37]. OutGuess inserts the message by using standard LSB replacement, while it avoids changing zeros and ones. Since this embedding operation changes the first-order histogram of DCT coefficients, OutGuess reserves some DCT coefficients to restore, approximately, the first order histogram. By doing so, OutGuess performs approximately twice as many changes as JSteg, which makes the algorithm more detectable by methods (features) modelling higher order dependencies.

Unlike OutGuess, **F5** [38] does not try to preserve the first-order histogram of DCT coefficients. Instead, it preserves the shape of the histogram, making it similar to that of the cover image. The message is embedded by changing the absolute values of DCT coefficients toward zero. DCT coefficients equal to zero are skipped, and if the coefficient is changed to zero during embedding, it is skipped as well and a new one is utilised for re-embedding. The F5 algorithm was also the first algorithm to use matrix embedding, a coding scheme that increases embedding efficiency, here measured as the number of bits embedded per embedding change.

Steghide [11] tries to preserve first-order statistics, but without making additional embedding changes like OutGuess. The algorithm starts by constructing a graph, where each vertex corresponds to a group of pixels that need to be changed. The weight of an edge between two vertices is proportional to the distortion caused by modification of both vertices such that they code the message. During the embedding, the algorithm finds the partition of the graph minimising the cost, subject to the chosen message being coded.

Despite the C source code for **JPHide&Seek** being available, its method of operation has not been described. To our knowledge, the algorithm has not been published in any scientific or other paper.

The **nsF5** algorithm uses the same type of embedding changes as the F5 algorithm. To avoid introducing more zeros (the shrinkage effect), nsF5 uses wet paper codes with improved efficiency [6]. The experiments in this paper used the version of the algorithm from 2008, which simulates the embedding efficiency of particular wet paper codes; this differs from the version currently published by the author, which simulates the theoretically-optimal efficiency.

C. Embedding Strategies

Embedding in multiple images poses new problems, originally described in [14]. The steganographer must choose how to spread a message of total length M bits into n covers $(X_1, \ldots, X_n)$ with capacities $(c_1, \ldots, c_n)$ by using the chosen steganographic algorithm. We distinguish the *embedding strategy*, which allocates payload amongst objects, from the *embedding algorithm* which inserts the payload steganographically.

In [17], we have identified five simple strategies to break the message into fragments of lengths $(m_1, \ldots, m_n)$ such that $M = \sum_{i=1}^{n} m_i$. Since one of the strategies had little practical value, we omit it here. None of the strategies is theoretically optimal, and indeed the batch steganography problem has not been solved. We have chosen strategies that could be

applied by a non-expert, similarly to our choice of embedding algorithms.

The **greedy** strategy tries to use as few images as possible. The steganographer chooses the cover with highest capacity, and embeds part of his message up to maximum capacity. If more message remains, he repeats with the cover of next highest capacity, until the whole message is embedded.

If the images are ordered by capacity so that $c_1 \geq c_2 \geq \ldots \geq c_n$, this leads to the following 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 **maximum** strategy is a variation of the greedy strategy, where the images for embedding are selected in random order and used to full capacity. This simulates a case of a steganographer who is not able to estimate capacity until they embed.

The **linear** strategy distributes the message 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 of their capacity. Thus

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

For relatively large payloads and covers of uneven capacity, sometimes m_i exceeds c_i. In this cases, we set $m_i = c_i$ and recalculate an even message length for the remaining images.

We do not consider, in this work, how the receiver is to reconstruct the original message. The allocation of payload might be part of a shared secret key, or stored in the first few bits of payload in a fixed position.

V. Main Experimental Results

We simulated large-scale steganalysis using tens of thousands of experiments. In each experiment, we randomly selected N_A actors out of the 4000 in our data set, and N_I images from each actor. Exactly one guilty actor is simulated, by using the chosen embedding algorithm (from Subsection IV-B) and embedding strategy (Subsection IV-C) to insert of payload size np, where $0 \leq p < 0.25$ and n is the total number of nonzero coefficients in their images. Thus p is the number of bits per nonzero coefficient (bpnc). It is important to measure the payload size relative to a fixed quantity, not to the capacity of an individual embedding algorithm, otherwise the results are incomparable. We then calculate features from each of the $N_A N_I$ images, MMD between each pair of actors, and LOF scores for each actor.

For each combination of parameters, each experiment is repeated 500 times with a different selection of actors and guilty actor. We need a benchmark to reflect how well the guilty actor is identified, and we have chosen the *average rank of the guilty actor*. An average rank of one corresponds to perfect detection — the guilty actor is always ranked most suspicious — and an average rank of $\frac{N_A+1}{2}$ corresponds to random guessing. We would not expect a universal, unsupervised detector to achieve perfect accuracy, but instead hope that it provides intelligence by ranking a truly guilty actor amongst the top 5-10%, say, of all actors.

A. Detecting Different Algorithms

We first demonstrate that the proposed detector is capable of detecting a wide range of algorithms (it has good universality). We tested $N_A \in \{100, 400, 1600\}$ actors with $N_I = 100$ images each, and every combination of embedding algorithm and strategy from subsects. IV-B and IV-C. The steganalyser used the centroid 'kernel' for the MMD distance measure, whitening of raw features (more on this in Subsection VI-A), and the LOF parameter $k = 10$.

The average ranks of the true guilty actor, when hiding payloads $p \in \{0.025, 0.05, \ldots, 0.25\}$, are shown in Figure 1. Our first observation is that the method works, when the total payload is large enough. There are differences between embedding algorithms (it confirms the known relative security of the embedding algorithms [28], that nsF5 is most secure and OutGuess/Steghide least secure) and between embedding strategies, but the overall pattern is consistent. With weaker embedding algorithms, perfect identification of the guilty actor is achievable around 0.1–0.2 bpnc payload sizes; perfect detection is not observed with nsF5 and F5, but the guilty actor is consistantly ranked as one of the 2–6 most suspicious out of 100. Monitoring larger number of actors does not substantially change the results, except that the average rank of the guilty actor apparently scales slightly sublinearly with the total number of actors N_A. A similar phenomenon was observed in [17], using a different metric for the anomaly detector, and this finding would seem favourable for large-scale steganalysis, but it has not yet been explained.

Second, we observe that the greedy strategy is consistently the most secure for the embedder: the average rank of the guilty actor is higher, in all algorithms and payloads. We will later show that this is only true when the steganalyst uses the centroid 'kernel'. The second most secure strategy is maximum, except in the case of F5 where the matrix embedding induces a nonlinear relationship between payload size and steganographic distortion. The linear strategy is next, and even is most insecure. The reason that the greedy/maximum strategy is more secure than linear/even has been explained in detail in our prior work [17].[2] In brief, this effect is caused by whitening the features in the pre-processing stage. But, as will

[2]In [17] the greedy and maximum strategies are called max-greedy and max-random, respectively.

be shown in Subsection VI-A, such preprocessing is needed to achieve good accuracy. We discuss this further in Section VII.

Because the greedy strategy dominates maximum, and linear dominates even, in subsequent experiments we will discard the maximum and even strategies.

B. Comparison with Prior Art

As mentioned above, no other literature addresses the large-scale steganalysis problem, so there is no direct prior art that we can compare to. The only method we can identify, for ranking guilty actors without knowledge of the embedding algorithm, is a combination of the universal detector proposed in [30] and *pooling strategies* described in [14] (which aggregate the scores for each actor).

The universal steganalyzer, here implemented as a one-class support machine (1-SVM) [34], assigns to each image a score $f(x) = \langle w, x\rangle_{\mathcal{H}} - b$, where x is a feature vector of a given image and (w, b) defines a hyperplane in a Hilbert space $\mathcal{H}$, which is determined from the training data. Following [14], we implemented two pooling strategies to rank the guiltiness of each actor from the scores of their images: (i) we calculated the average score

$$\frac{1}{n}\sum_{i=1}^{n} f(x_i),$$

or (ii) we calculated the number of positive scores (number of images classified as outliers)

$$\#\{x_i \,|\, f(x_i) > 0\}.$$

There may well be better methods for aggregation, but the literature does not yet contain them.

The 1-SVMs were trained on 6000 cover images from 60 actors; the pool of actors used for this training was disjoint from, but from the same social media source as, all the other experiments in the paper. To avoid bias from picking a particularly good or poor set of training images, we trained 20 different 1-SVMs, on different cover examples, and picked one of the machines at random for each experiment. The 1-SVM hyperparameters were $\nu = 0.01$ (proportion of outliers) and γ (width of Gaussian kernel) using the following common heuristic: inverse median of squared distances between cover images (features) in the training set. The features were normalised to have zero mean and unit variance. Note the distinction between the method presented in this paper, which is completely untrained, and the use of 1-SVMs, which do require cover training data.

The average rank of the guilty actor, hiding payloads $p \in \{0.025, 0.05, \ldots, 0.25\}$ using only the nsF5 algorithm and greedy/linear strategies, is shown in Figure 2. The steganalyst used the two aggregations of 1-SVM scores, or the proposed detector with the same settings as in the previous subsection. Here $N_A = 100$ and $N_I = 100$ (similar results are observed, but not included here, with other combinations of parameter). The graphs clearly show that the proposed solution is substantially more accurate than prior art (except for tiny payloads when both are guessing randomly), which is not aggregating effectively the evidence from the multiple images.

VI. Supplemental Experiments

The detector described in Section III has several hyperparameters, which influence its performance. In the previous section, we used parameters based on our previous experiments published in [16], [18]. Here, we re-examine the choices one-by-one.

Unless otherwise indicated, all experiments in this section share the same setting of one guilty actor emitting a payload of 0.1 bpnc, using nsF5 algorithm and the greedy or linear strategy. The steganalyst uses the proposed detector with whitened features, the centroid 'kernel', and LOF parameter $k = 10$. $N_I = 100$ and $N_A \in \{100, 400, 1600\}$.

A. Pre-processing

The pre-processing of features has a significant impact on the accuracy of detection. In steganalysis literature a common pre-processing is normalisation, where each feature is individually scaled to have zero mean and unit variance (usually across the cover training set). The goal is to prevent features with high variance from dominating other, perhaps more informative, features of low variance. In this application it is essential, so that the MMD distances are meaningful and not dominated by noisy components.

For additional stabilization, one can also apply whitening (principal component transform), to decorrelate the features (again, usually on the cover training set); in this application, because equally-scaled components are essential, we further apply normalization after whitening.

Figure 3 shows the performance of the detector, when the steganalyst uses unprocessed (raw), normalised, and whitened features. In all situations (different embedding strategies of the guilty actor, different number of actors) the raw features are near-useless, confirming the importance of equal scaling of features in the anomaly detector: a different situation may hold in supervised classifiers, where the training phase can learn to ignore noisy features. Whitened features are consistently the best option. Surprisingly, with an increasing number of actors, the advantage of whitening over normalization decreases.

In image recognition applications, principal component analysis (PCA) is frequently used as a denoising filter, discarding components corresponding to small eigenvalues in the correlation matrix. We use the same in our application, applying it to all feature vectors pooled across all actors in each experiment, and discarding the components with low eigenvalue after whitening the features. In the previous experiments we discarded such components with corresponding eigenvalues lower than 0.01, as we expected them to carry noise. But is this really a sensible choice? Figure 4 shows the average rank of the guilty user, as we vary the number of components retained after whitening: the components were sorted from those highest eigenvalue (variance) to lowest, and we kept only the highest. The results show that better accuracy is acheived when most components are used; for large number of actors the improvement is very negligible and it looks like that the optimum is near, but not quite at, the maximum of retaining all 274 components. Due to the difficulty of finding

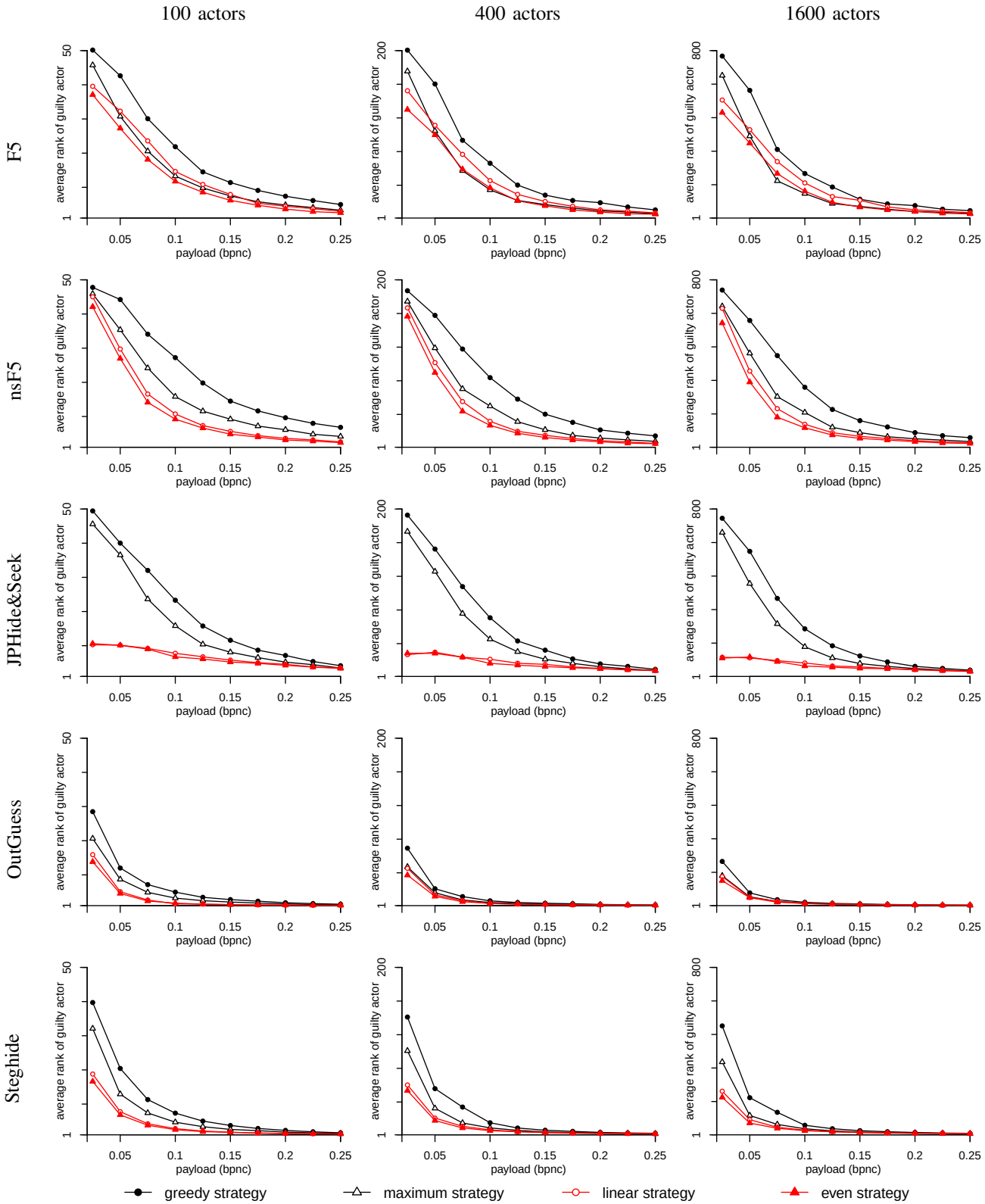

Fig. 1. Performance of the proposed detector: from top to bottom, five different embedding algorithms; from left to right, different numbers of actors N_A ($N_I = 100$ in each case); lines in each chart denote different embedding strategies; each x-axis represents total payload (bpnc) and each y-axis represents the average rank of the truly guilty actor. The detector parameters are: centroid 'kernel', whitened features, $k = 10$.

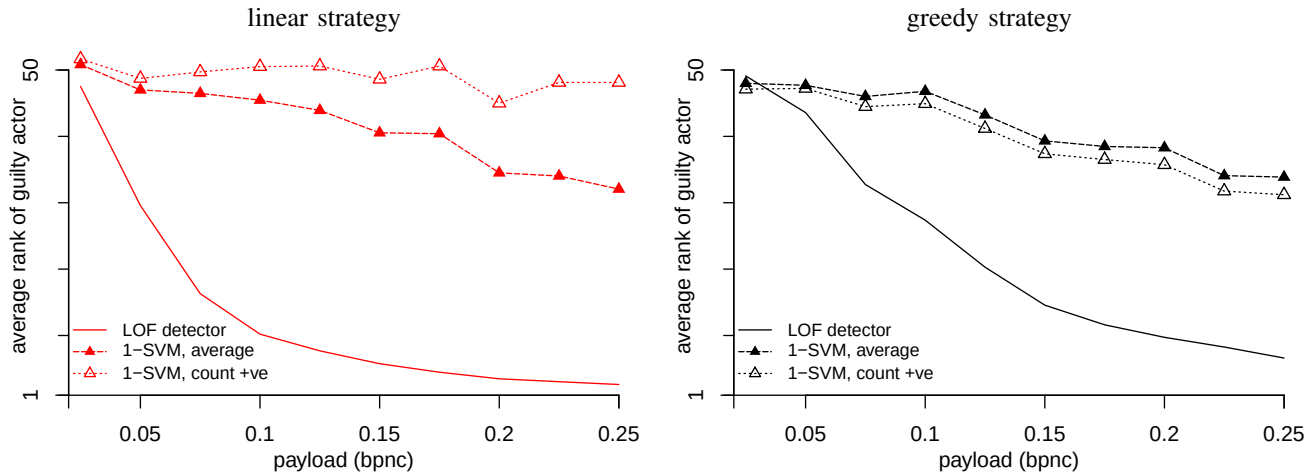

Fig. 2. Comparison of proposed detector with prior art. Each x-axis represents total payload (bpnc) and each y-axis represents the average rank of the truly guilty actor; left, using the linear strategy; right, using the greedy strategy. In both cases $N_A = N_I = 100$, and the detector parameters are: centroid 'kernel', whitened features, $k = 10$.

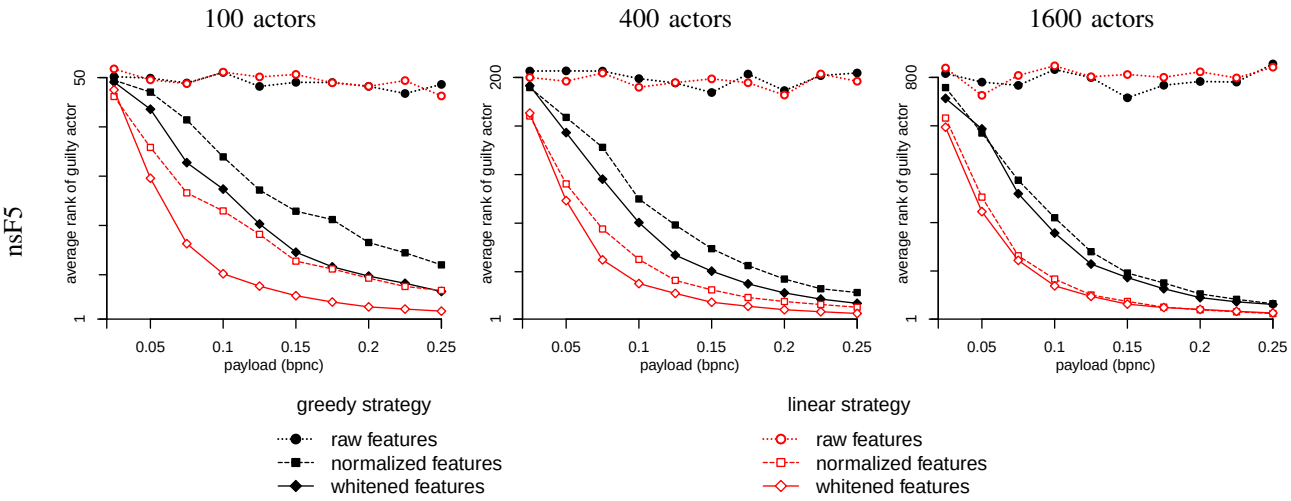

Fig. 3. The effect of different feature pre-processing. From left to right, different numbers of actors N_A ($N_I = 100$ in each case); lines in each chart denote different pre-processing options and embedding strategy; each x-axis represents total payload (bpnc) and each y-axis represents the average rank of the truly guilty actor. The embedder uses nsF5 embedding. The other detector parameters are: centroid 'kernel', $k = 10$.

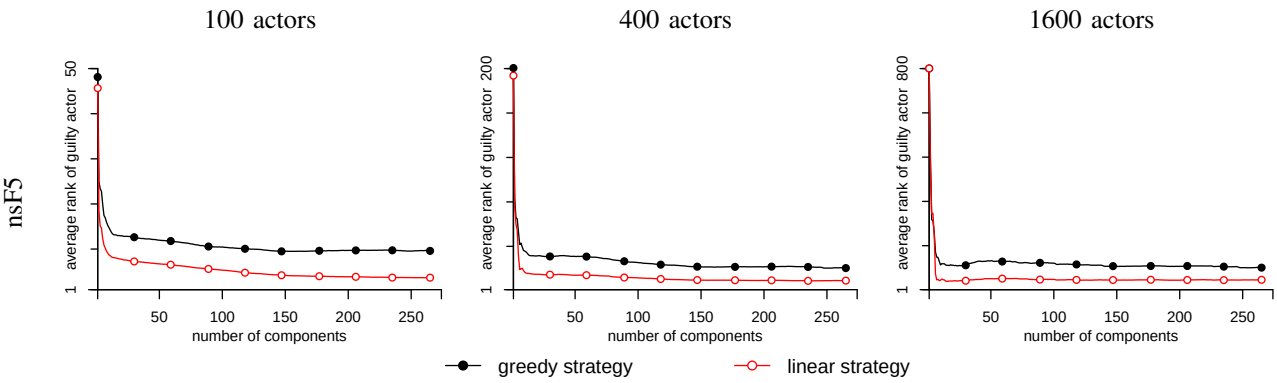

Fig. 4. The effect of the number of components retained after whitening. From left to right, different numbers of actors N_A ($N_I = 100$ in each case); lines in each chart denote different embedding strategy; each x-axis denotes the number of components kept for the LOF analysis, and each y-axis represents the average rank of the truly guilty actor. The embedder uses nsF5 embedding with total payload 0.2 bpnc. The other detector parameters are: centroid 'kernel', $k = 10$.

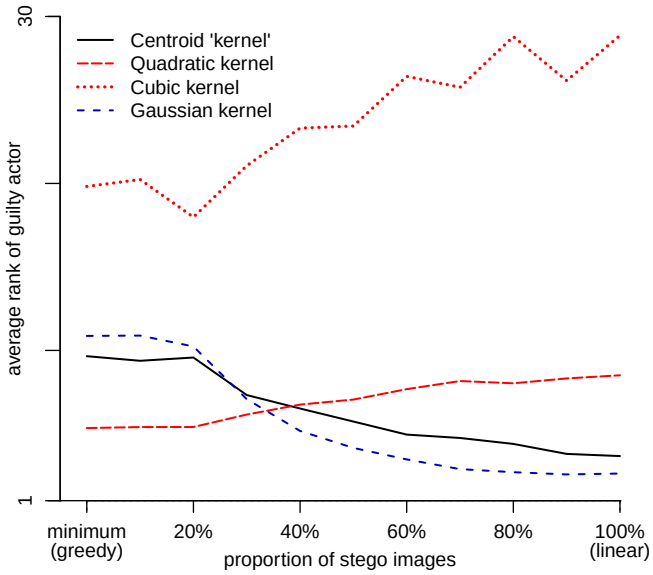

Fig. 5. The effect of the MMD kernel. The x-axis denotes strategies interpolating smoothly between greedy (left) and linear (right), the y-axis the average rank of the truly guilty actor. The different lines indicate diffeerent kernels. Here $N_A = N_I = 100$, the payload is 0.2 bpnc and the embedding is by nsF5, the features were whitened, and the LOF parameter $k = 10$.

this maximum in advance, and for good numerical stability, we suggest continuing to use the eigenvalue threshold 0.01.

B. Kernel

The kernel function used in the calculation of MMD distance underpins the entire system. Thus far we have used the centroid 'kernel', which approximates the linear kernel $\kappa(x, y) = x^{\mathrm{T}}y$, because it has linear time complexity. Experiments in [18] report its superiority with respect to other kernels, but only one embedding strategy was examined.

There is reason to believe that alternative kernels should have advantages against certain embedding strategies. To summarise the argument in [17]: because of noisy components, steganalysis features per object are distorted by an amount *sublinear* in the payload, and the linear kernel only agglomerates this distortion linearly. Hence the apparent superiority of the greedy strategy. But consider some of the theoretical steganography work on small payloads: in principle, distortion relating to statistical detectability (KL divergence) is locally quadratic in the payload size [5]. Thus a kernel which captures such distortion would be more powerful against the greedy strategy. This motivates us to examine polynomial kernels $k(x, y) = (x^{\mathrm{T}}y)^2$ (quadratic, which should be powerful against the greedy strategy) and $k(x, y) = (x^{\mathrm{T}}y)^3$ (cubic), as well as Gaussian $k(x, y) = \exp(-\gamma\|x - y\|^2)$. The problem of setting γ is treated in more detail in Appendix A. We also investigated higher-order polynomials, but the results were weak and are not included here.

For a more fine-grained analysis of embedding strategy, we used methods which apply linear embedding to the greatest-capacity proportion P of the guilty actor's images. For small P, this is equivalent to the greedy strategy (maximum payload in fewest covers); for $P = 100\%$ it is the linear strategy, and in between it exchanges size of payload-per-image for number of images used. The average rank of the guilty actor, when testing each kernel against strategies for various P, is shown in Figure 5. The results validate our theory that the embedding strategies have a detection counter-strategy: for low P (greedy) the quadratic kernel is indeed most accurate, while for high P it is the Gaussian kernel with best performance. Our use of the centroid 'kernel' was slightly sub-optimal, in that it is dominated by the Gaussian or quadratic kernels, but the difference between centroid and Gaussian is not so great as to outweigh the benefits of its linear time complexity.

As predicted in [14] and [15], we find a two-player game between the embedder and detector. We could define the zero-sum payoff to be the average rank of the guilty actor. In that case we can even, purely for illustration, use standard linear-programming techniques to compute the equilibrium strategies from our empirical data. It turns out that both players should use *mixed* (randomized) strategies to avoid being exploited by their opponent: the embedder should pick the greedy or linear strategy at random, with probability approximately 0.52 and 0.48, respectively (intermediate options are dominated), while the detector should pick the Gaussian kernel with probability 0.27, and quadratic with probability 0.73. The average rank is then approximately 6.9. Of course, these strategies are only optimal within the narrow confines of this game, in particular for this embedding algorithm, payload size, number of images, and number of actors, and other possible kernels or embedding strategies may change the results. The game theory of steganography is still in its infancy [36], and has not reached the batch steganography case.

C. Multiple Guilty Actors, Effect of k

Thus far, our experiments have simulated only one guilty actor, and the number of nearest neighbours in LOF method was set to $k = 10$, as recommended in [2]. If there are multiple guilty actors in a tight cluster then the optimal choice of k depends on the size of that cluster: too small a value k causes the cluster to be deemed 'normal', too large will smooth out its anomaly level. To briefly investigate this, we vary the number of guilty actors from $\{1, 2, 4, 8\}$ out of a total of $N_A = 100$ actors, and tested $k \in \{2, 4, 6, \ldots, 20\}$. We compute the same metric: average rank of the guilty actor.

The results in Table I reveal that smaller values of k are slightly more accurate for detecting a single guilty actor, and larger values better for detecting more actors. This suggests that the guilty actors indeed form some sort of cluster, which would probably not happen if the guilty actors used different embedding methods or strategies (we postpone such further experiments to future work). The default value of $k = 10$, which we have used in this paper, seems to be a good compromise, but the optimal k depends also on the embedding strategy used by the guilty actors, introducing another potential game between embedder and detector.

VII. Conclusion

We have presented a first method for addressing a new and realistic problem in steganalysis: detecting the guilty

	linear strategy #guilty actors				greedy strategy #guilty actors			
k	1	2	4	8	1	2	4	8
2	2.7	7.0	63.1	62.2	6.9	14.5	29.3	42.7
4	2.9	5.0	16.3	62.5	7.8	11.3	19.1	31.6
6	3.1	4.9	10.0	43.0	8.4	11.8	17.9	28.2
8	3.4	4.8	9.0	24.1	9.2	12.4	17.6	26.5
10	3.5	5.1	8.7	17.8	9.7	12.1	17.7	26.5
12	3.7	5.1	8.5	15.4	9.8	12.5	17.8	26.1
14	3.8	5.1	8.2	15.1	10.3	13.0	18.3	27.4
16	3.9	5.2	8.2	14.3	10.7	13.0	18.9	27.3
18	3.9	5.5	8.2	13.8	10.7	13.4	19.6	28.2
20	4.1	5.4	8.0	13.6	10.9	14.0	19.7	28.2
Perfect	1	1.5	2.5	4.5	1	1.5	2.5	4.5

TABLE I
THE AVERAGE RANK OF 1,2,4, OR 8 GUILTY ACTORS USING EITHER LINEAR OR GREEDY STRATEGY. THE DETECTOR VARIES THE NEAREST NEIGHBOUR PARAMETER k IN THE LOF METHOD. HERE $N_A = N_I = 100$, THE PAYLOAD IS 0.2 BPNC, THE EMBEDDING IS BY NSF5, AND THE FEATURES WERE WHITENED. THE LAST ROW CAPTIONED "PERFECT" SHOWS THE AVERAGE RANK OF A PERFECT DETECTOR.

actor, rather than individual image, in the setting of network monitoring when there are many actors and images to consider. To our knowledge this is the first work to address this problem or the pooled steganalysis problem [14], and the first to use MMD in steganalysis detection. A key element is to turn some of the difficulties in the problem – the large number of users and images – to our advantage by calibrating the behaviour of outliers against that of the majority, allowing completely unsupervised detection.

Our experiments, which simulated embedding and detection in $100 - 1600$ actors, each transmitting 100 images[3], repeated thousands of times with different combinations of embedding parameters, payload size, and detection parameters.

Essential to every modern steganalysis method are the features that drive it. We used a well-established feature set, which is not one of the recent "rich model" [7], [13], [20] feature sets that have been published recently, for two reasons. First, for experiments on such a scale it is essential that the features be calculated quickly and the many-thousand dimensional rich model features are not quick to extract. Second, and a subject of our current work, is that larger feature sets perform *worse* when plugged into the same anomaly detection framework despite being considerably more effective in traditional binary steganalysis. We have examined this phenomenon and it is an unavoidable consequence of unsupervised learning: there is no way to discard or weight down the weaker features in a training phase, so the anomaly detector is overwhelmed by noise. It illustrates that the design of good features for the standard binary-classification steganalysis, where extra features are essentially free of cost, does not translate well into an unsupervised case. Our current work involves partially-supervised dimensionality reduction to focus the power of large feature sets into a smaller set which is suitable for unsupervised anomaly detection [31].

[3]Further experiments with as few as 20 and as many as 200 images were performed, but the results were so similar to those presented here that we do not include them.

We also tested only well-understood embedding algorithms which have long been known to be detectable by statistical analysis, and naive embedding strategies with little or no adaptive allocation of payload between images. Again this is required for experimental efficiency, but we were also motivated by simulating a "real-world" steganographer who uses tools currently easily available on the internet. For the same reason we used a large image set downloaded from social media. Our method, however, is applicable to any embedding method where the features are more sensitive to stego content than payload content, and it would be very surprising if even the most recent adaptive embedding methods [4], [12] were not detectable, albeit at higher payloads because of their lower distortion, than their simpler ancestors tested here.

Although we have investigated good parameters (feature preprocessing, kernel, LOF parameter) for the detector, this demonstrated the game at the heart of steganography and steganalysis: fixed (known) choices by the embedder allows the detector to tune parameters to enhance accuracy, whereas fixed (known) choices by the detector allow the embedding to tune their parameters to reduce detection accuracy. We cannot solve such games until all the options, for embedder and detector, are properly understood, but in the future we might hope to find both equilibria and, more practically usefully, conservative minimax strategies for each player.

Other future work should be to examine more closely the cases of multiple guilty actors, as well as the case of zero guilty actors: throughout this paper we assumed that at least one actor was guilty, which may not be true in application. Ideally we would like to estimates the probability of guilt for each actor, which is a known and difficult problem in anomaly detection. Another direction might be to use multiple anomaly detectors instead of a single LOF, hoping that diversity amongst outlier-detection methods will lead to better results, but this runs into the known, nontrivial, problem of aggregating scores [22].

REFERENCES

[1] P. Bas, T. Filler, and T. Pevný. "Break our steganographic system": The ins and outs of organizing BOSS. In *Proc. 13th Information Hiding Conference*, pages 59–70. Springer, 2011.

[2] 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.

[3] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: a survey. *ACM Computing Surveys*, 41:15:1–15:58, 2009.

[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, A. D. Ker, and J. Fridrich. The square root law of steganographic capacity for Markov covers. In Edward J. Delp III, J.a Dittmann, Nasir D. Memon, and Ping Wah Wong, editors, *Media Forensics and Security XI*, volume 7254 of *Proc. SPIE*, pages 0801–0811. SPIE, 2009.

[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 and J. Kodovský. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3):868–882, 2012.

[8] J. Fridrich, J. Kodovský, V. Holub, and M. Goljan. Steganalysis of content-adaptive steganography in spatial domain. In *Proc. 13th Information Hiding Conference*, pages 102–117. Springer, 2011.

[9] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample problem. In *Advances in Neural Information Processing Systems 19*, pages 513–520. MIT Press, 2007.

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

[11] 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.

[12] V. Holub and J. Fridrich. Digital image steganography using universal distortion. In *Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '13, pages 59–68, New York, NY, USA, 2013. ACM.

[13] V. Holub and J. Fridrich. Random projections of residuals for digital image steganalysis. *Information Forensics and Security, IEEE Transactions on*, 8(12):1996–2006, 2013.

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

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

[16] A. D. Ker and T. Pevný. A new paradigm for steganalysis via clustering. In Nasir D. Memon, J.a Dittmann, Adnan M. Alattar, and Edward J. Delp III, editors, *Media Watermarking, Security, and Forensics III*, volume 7880, pages 0U01–OU13. SPIE, 2011.

[17] A. D. Ker and T. Pevný. Batch steganography in the real world. In *Proceedings of the 14th ACM Workshop on Multimedia and Security*, MM&Sec '12, pages 1–10, New York, NY, USA, 2012. ACM.

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

[19] J. Kodovský and J Fridrich. Steganalysis in high dimensions: Fusing classifiers built on random subspaces. In *IS&T/SPIE Electronic Imaging*, pages 78800L–78800L. International Society for Optics and Photonics, 2011.

[20] J. Kodovský and J. Fridrich. Steganalysis of JPEG images using rich models. In N.D. Memon, A.M. Alattar, and E.J. Delp III, editors, *Media Watermarking, Security, and Forensics XIV*, volume 8303 of *Proc. SPIE*, 2012.

[21] J. Kodovský, J. Fridrich, and V. Holub. Ensemble classifiers for steganalysis of digital media. *Information Forensics and Security, IEEE Transactions on*, 7(2):432–444, 2012.

[22] H. P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Interpreting and unifying outlier scores. In *Proceedings of the Eleventh SIAM International Conference on Data Mining*, pages 13–24, 2011.

[23] 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.

[24] I. Lubenko and A. D. Ker. Going from small to large data in steganalysis. In *IS&T/SPIE Electronic Imaging*, pages 83030M–83030M. International Society for Optics and Photonics, 2012.

[25] S. Lyu and H. Farid. Steganalysis using higher-order image statistics. *IEEE Transactions on Information Forensics and Security*, 1(1):111–119, 2006.

[26] T. Pevný. Detecting messages of unknown length. *Electronic Imaging, Media Watermarking, Security and Forensics of Multimedia XIII, San Francisco, CA. Proceedings SPIE, J.uary*, pages 23–26, 2011.

[27] T. Pevný, P. Bas, and J. Fridrich. Steganalysis by subtractive pixel adjacency matrix. *Information Forensics and Security, IEEE Transactions on*, 5(2):215–224, 2010.

[28] 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.

[29] T. Pevný and J. Fridrich. Multiclass detector of current steganographic methods for JPEG format. *IEEE Transactions on Information Forensics and Security*, 3(4):635–650, December 2008.

[30] T. Pevný and J. Fridrich. Novelty detection in blind steganalysis. In *Proceedings of the 10th workshop on Multimedia & security*. ACM, New York, USA, 2008.

[31] T. Pevný and A. D. Ker. The challenges of rich features in universal steganalysis. In Adnan. M. Alattar, Nasir D. Memon, and Chad D. Heitzenrater, editors, *Media Watermarking, Security, and Forensics 2013*, volume 8665 of *Proc. SPIE*, pages 0M01–0M15. SPIE, 2013.

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

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

[34] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[35] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[36] P. Schöttle, S. Korff, and R. Böhme. Weighted stego-image steganalysis for naive content-adaptive embedding. In *Proceedings of the 4th IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 193–198, 2012.

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

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

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

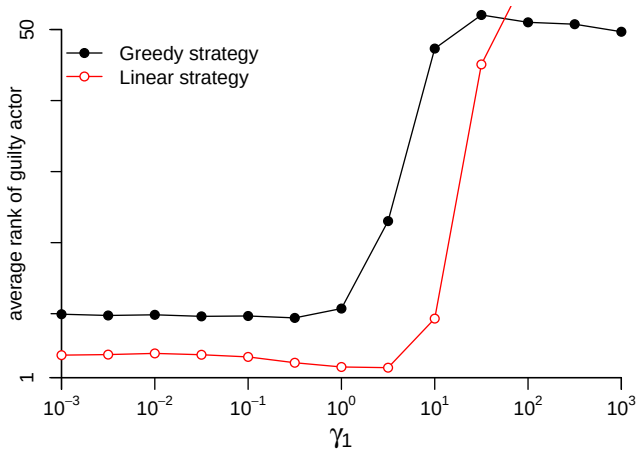

Fig. 6. Effect of the Gaussian kernel width. The embedder uses nsF5 embedding with total payload 0.2 bpnc, the features have been whitened, and the LOF parameter is $k = 10$.

APPENDIX

A. Gaussian Kernel Width

The quality of detection with the Gaussian MMD kernel $\kappa(x, y) = \exp(-\gamma\|x - y\|^2)$ depends on the choice of γ, and a wrong choice can decrease the performance. With the recommendation of [35], we investigated γ in a range around γ_0, the latter defined as the inverse of the median squared distances between (whitened) image features. We set $\gamma = \gamma_0 \cdot \gamma_1$ where $\gamma_1 \in \{10^n | n \in \{-3, -2, \ldots, 3\}\}$. Fixing on a payload of 0.2 bpnc, the nsF5 embedding algorithm, and the greedy and linear strategies, we tested all such γ and display the results in Fig. 6.

Because of the connections between linear and Gaussian kernel with small γ (see Appendix B), it is not surprising that small kernel γ does not penalise the performance very much; large γ causes performance to decrease to random guessing because the MMD calculations are dominated by outliers in

the feature point clouds of each actor. The optimum is indeed around the default $\gamma = \gamma_0$ in each case.

B. Connections Between Kernels

We have used primarily the centroid 'kernel'; here we demonstrate its connection with the true linear kernel, and also the Gaussian kernel for large kernel width.

First, use symmetry of the kernel to write

$$MMD_{linear}(X,Y) = \frac{1}{n(n-1)} \sum_{1\le i\ne j\le n} x_i^{\mathrm{T}} x_j - 2x_i^{\mathrm{T}} y_j + y_i^{\mathrm{T}} y_j.$$

Then we expand

$$\begin{aligned} &MMD_{centroid}(X,Y) \\ &\quad = \left(\tfrac{1}{n}\textstyle\sum_i x_i - \tfrac{1}{n}\sum_i y_i\right)^{\mathrm{T}} \left(\tfrac{1}{n}\textstyle\sum_i x_i - \tfrac{1}{n}\sum_i y_i\right) \\ &\quad = \tfrac{1}{n^2}\textstyle\sum_{i,j} x_i^{\mathrm{T}} x_j - 2x_i^{\mathrm{T}} y_j + y_i^{\mathrm{T}} y_j \\ &\quad = \tfrac{n}{n-1} MMD_{linear}(X,Y) + \tfrac{1}{n^2}\textstyle\sum_i (x_i - y_i)^{\mathrm{T}}(x_i - y_i). \end{aligned}$$

This demonstrates that the centroid 'kernel' approximates the true linear MMD for large n.

Now let k be the Gaussian kernel with inverse width γ, then

$$k(x,y) = 1 - \gamma\|x-y\|^2 + O(\gamma^2),$$

so that for small γ we have

$$MMD_{Gaussian}(X,Y) \approx \frac{\gamma}{n(n-1)} \sum_{1\le i\ne j\le n} 2\|x_i - y_j\|^2 - \|x_i - x_j\|^2 - \|y_i - y_j\|^2.$$

The first term measures average distances *between* the distributions X and Y, and the other terms measure average distances *within* them. Indeed, if the x_i (respectively y_i) are drawn from any multivariate distribution with mean μ_x (μ_y) and finite covariance matrix Σ_x (Σ_y) then elementary calculations give

$$\begin{aligned} E\big[\|x_i - y_j\|^2\big] &= \mathrm{Tr}(\Sigma_x + \Sigma_y) + \|\mu_x - \mu_y\|^2, \\ E\big[\|x_i - x_j\|^2\big] &= 2\,\mathrm{Tr}(\Sigma_x), \\ E\big[\|y_i - y_j\|^2\big] &= 2\,\mathrm{Tr}(\Sigma_y), \end{aligned}$$

and hence by the law of large numbers $MMD_{Gaussian}(X,Y) \to \gamma \cdot MMD_{centroid}(X,Y) + O(\gamma^2)$ as $n \to \infty$. The LOF method is scale insensitive, so the factor γ has no effect on it. This explains the behaviour seen in Fig. 6.