Identifying a steganographer in realistic and heterogeneous data sets

Andrew D. Ker[a] and Tomáš Pevný[b]

[a]Oxford University Department of Computer Science, Parks Road, Oxford OX1 3QD, England;
[b]Agent Technology Center, Czech Technical University in Prague,
Karlovo namesti 13, 121 35 Prague 2, Czech Republic.

ABSTRACT

We consider the problem of universal pooled steganalysis, in which we aim to identify a steganographer who sends many images (some of them innocent) in a network of many other innocent users. The detector must deal with multiple users and multiple images per user, and particularly the differences between cover sources used by different users. Despite being posed for five years, this problem has only previously been addressed by our 2011 paper.

We extend our prior work in two ways. First, we present experiments in a new, highly realistic, domain: up to 4000 actors each transmitting up to 200 images, real-world data downloaded from a social networking site. Second, we replace hierarchical clustering by the method called *local outlier factor* (LOF), giving greater accuracy of detection, and allowing a guilty actor sending moderate payloads to be detected, even amongst thousands of other actors sending hundreds of thousands of images.

1. INTRODUCTION

The traditional steganalytic scenario usually involves a steganalyst trying to detect whether a *single object* carries payload or not, and such a scenario is common in most literature. But in practice, one can expect to consider a rather different scenario, where the detector has to consider multiple users (here called *actors*), only some of whom are guilty of using steganographic embedding: for example when a network is monitored for information leakage. Each actor will transmit multiple objects. Moreover, the guilty actor(s) will perhaps sometimes act innocently, thus transmitting a mixture of cover and stego objects. The problem of detecting data hidden in a mixture of cover and stego objects, known as *pooled steganalysis*, was introduced in 2006 in Ref. 1, but no practical solution was proposed until 2011 in Ref. 2.

The approach proposed in 2011 presented a novel paradigm to steganalysis. It used traditional steganalysis features but replaced *classification* by *clustering*. Furthermore, it is the actors which are clustered, not their images. In a nutshell, the method works as follows. First, it calculates distances between every pair of actors, by means of maximum mean discrepancy between the sets of feature vectors of the actors' images. Then it uses a clustering algorithm to find the actor which deviates the most from the rest. The actor who behaves unlike the others is judged guilty.

In this way, the paradigm has removed the need for training, as it simultaneously judges the behavior of actors by assuming that most of them are innocent. Another benefit is its universality, as it does not specialize to any particular steganographic algorithm. Most importantly, it is not tuned only for a particular source of cover objects, and takes into account the way that different actors' images exhibit different characteristics. The requirement for its success is that the steganalytic features are sensitive to changes caused by the steganographic algorithm, and they are comparatively robust against differences between cover objects of different users.

This work extends the 2011 publication[2] in two ways. First, it presents experiments in a new, highly realistic, domain. While the original work had considered seven or thirteen actors represented by images from different

Further author information:
A. D. Ker: E-mail: adk@cs.ox.ac.uk, Telephone: +44 1865 283530
T. Pevný: E-mail: pevnak@gmail.com, Telephone: +420 22435 7608

cameras, this work considers up to 4000 actors each transmitting up to 200 images, for a total of 800 000 images. The images are real-world data downloaded from a social networking site, which resizes and recompresses the images thereby introducing difficult image processing artifacts, most notably double-compression. As such, we consider the image database to capture the challenge of real-world pooled steganalysis.

The experiments revealed that hierarchical clustering, proposed in our previous work,[2] has shortcomings with detrimental effects on the accuracy of the method. It was found to amplify the sensitivity of steganalytic features to the differences in actors' cover sources, and it does not provide any degree of "being an outlier" needed to identify the guilty user. These shortcoming have been addressed in this paper, by replacing hierarchical clustering by the method called *local outlier factor* (LOF).[3] We will demonstrate that the LOF method gives greater accuracy of detection, and allows a guilty actor sending moderate payloads to be detected or ranked in the top few most-outlying actors, even amongst thousands of other actors sending hundreds of thousands of images.

2. OUTLIER DETECTION AS A UNIVERSAL POOLED STEGANALYZER

Suppose that multiple actors each transmit multiple objects, all of which have been intercepted; we assume that each actor has just one source of objects, but that these sources are different. We do not anticipate that the same sources are available to us. Furthermore, suppose that we know which actor sent which object. For every object, we compute a feature vector which aims to distinguish innocent covers from stego objects.

We can think of this data as many clouds of points in the feature space, one cloud for each actor. A *guilty* actor is one who using steganography in (some of) their transmitted objects, and we aim to identify guilty actors if their clouds of feature vectors stand out, in some way, from the innocent actors.

2.1 Previous clustering methodology

Our prior work[2] identifies guilty actors rather than individual stego-objects. The method works as follows. First, the distances between all pairs of actors are calculated, by maximum mean discrepancy (MMD). Second, these distances are used by hierarchical clustering to group users, incrementally, together. Finally, the smaller cluster from the last two clusters before the final merge is considered as to be the cluster with guilty users.

The advantage of this approach is that, by pooling objects from each actor together, the method improves the signal-to-noise ratio compared with working on individual objects. This leads to better accuracy in the identification of guilty actor. Identifying individual stego-objects might still be valuable, but it is likely that any guilty party would be lost in a crowd of false positives. Note that it is generally hard to avoid an excessively large false positive rate, because the training model for cover images will not match any of the actors' sources exactly.

It would, of course, be possible to perform clustering on the individual objects, but we expect that this will provide little useful information. It is quite likely that this will simply tell us what we already know: each actor's objects will end up in a separate cluster characterizing their source. Instead, *actors* are clustered hoping to separate an innocent majority from a guilty minority.

2.2 Local Outlier Factor

The rationale for using a clustering algorithm was that the group of guilty users should form a distinct cluster, that would be well separated from the innocent users. Although this worked well in the limited scenario (7 or 13 actors each using images from a single camera) in our prior work, its performance in this paper's realistic data set was mediocre. We believe that this was due to the wrong assumption that guilty users would form a tight cluster. In reality, differences in actors' cover sources means that there is already variation in feature vectors between different actors. The steganographic embedding would exhibit itself by a shift in the feature vectors in one direction[4*]. Consequently, the guilty actors would be represented as outliers rather then tight cluster, and we should switch our attention from clustering to outlier detection.

*So-called perturbed quantization embedding (PQ) exhibits this behavior only on low embedding rates.

From a vast number of outlier detection methods that can be found in the literature in Ref. 5, we have chosen the *local outlier factor* (LOF) method for the following reasons: (a) it requires only the pairwise distances between points, (b) it is not tied to any particular application domain, (c) it directly provides a measure of how much an outlier each point is, and (d) it relies on single hyper-parameter (number of neighbors, k), to which the method it is not particularly sensitive.

The local outlier factor identifies outliers in the set of points in some feature space $\mathcal{X}$. The only requirement on this space is that there is a distance function $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. The method robustly estimates the probability density at the point of interest, p, and it compares this estimate to the same estimates at its k^{th}-nearest neighbors. By means of this comparison, the method determines a threshold on the density, from which p is considered to be an outlier. This threshold is determined locally by using at most k^2 neighbors of the point p resulting in adaptation of the threshold to the vicinity of p. This nice feature allows the method to identify outliers in data with clusters of different densities, which is in sharp comparison to most methods determining the threshold globally. We now describe the local outlier factor method in brief; for more detailed description and analysis, we refer to the original publication.[3]

Assume that a set of points C (feature vectors) is available. For the sake of simplicity, it is assumed that the point, p, which degree of outlying one wants to estimate, belongs to C. The method uses a parameter $k \in \mathbb{N}$ with $1 < k < |C|$, specifying the number of nearest neighbors.

The **k-distance of the point** $p \in C$, (further denoted as $d_k(p)$), is defined as a distance between p and $o \in C \backslash \{p\}$, such that:

- for at least k objects $o' \in C \backslash \{p\}$ it holds $d(p, o') \leq d_k(p)$, and
- for at most $k-1$ objects $o' \in C \backslash \{p\}$ it holds $d(p, o') < d_k(p)$.

In most cases arising in steganography, the k-distance of point p is equal to the distance to the k^{th} nearest point in C, because the probability of two points having all components equal is almost zero.

The **k-neighborhood** of the point p is the set of points $N_k(p) = \{o \in C \,|\, d(p, o) \leq d_k(p)\}$. Note that the number of elements in the k-neighborhood, $|N_k(p)|$, can be greater than k.

The **reachability distance of point p w.r.t. point** o is defined as $r_k(p, o) = \max\{d_k(o), d(p, o)\}$. The reason for introducing reachability distance is to reduce statistical fluctuations of the normal distance d for close objects. This smoothing effect is controlled by the parameter k. Notice that the reachability distance is not symmetric, so it is not a true distance.

The **local reachability density of** p is defined as an inverse of the average reachability distance of point p to all points o in its k-neighborhood, i.e.

$$lrd_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} r_k(p, o)}.$$

The local outlier factor method uses reachability distance to estimate density of the probability at point p. The method also estimates the average density of the probability of all points in k-neighborhood, $o \in N_k(p)$, and compares these quantities. The level of outlying is therefore adaptive with respect to the closest neighborhood.

The **local outlier factor (LOF) of** p is defined as

$$lof_k(p) = \frac{1}{|N_k(p)|} \sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}.$$

The $lof_k(p)$ captures degree to which p can be called an outlier. Obviously a high value indicates that p is an outlier, since its reachability density, $lrd_k(p)$, is smaller in comparison to the local reachability density of its neighbors.

The local outlier factor method does not provide a threshold, above which the element should be considered to be an outlier, although for true "inliers" the value is around 1. Despite this weakness, the method has several

appealing properties. First, the value $lof_k(p)$ does not depend on the absolute value of distances $d(p,q)$: it is scale invariant. Second, the method is relatively insensitive to setting of the number of nearest neighbors k. According to the original publication,[3] k should be at least 10 with the following caveat: if one requires that small clusters with less then n points should be considered as a set of outliers rather then a small cluster within the data, then $k > n$.

Unless stated otherwise, we have used value $k = 10$ in our experiments, but by altering it we could encourage clusters with small number of actors to be identified as outliers. Notice that by switching to the outlier detection method, we have overcome another shortcoming of the clustering approach, which was the lack of any direct measure of "being an outlier". If we know that we want to identify a single guilty actor, we can take the one with the greatest LOF; if we want to identify a short list of the N most suspicious actors, we can those with the top N LOF values.

3. EXPERIMENTAL PARAMETERS

To implement the method, we first need an opponent, defined by their embedding algorithm. Then we need a steganalytic feature set, a method to pre-process the features (usually for normalization) and a distance metric between actors.

3.1 Embedding algorithm

Since our cover images are JPEGs, we have limited ourselves to the nsF5 algorithm.[6] This algorithm represents an advanced algorithm for JPEG images, but one which is detectable by modern steganographic features. As such, it suits our needs for the investigation of qualities and advantages of our paradigms for pooled steganalysis under realistic conditions. Payload is measured in bits per nonzero coefficient (bpnc).

3.2 Feature set

The steganalytic features should be well-chosen, so that the difference between actors' sources is less than the difference between guilty and innocent actors. Although the performance of the method is directly affected by the choice, we do not deal with it here, as it is not the main interest of this paper. In all experiments described below, we have simply used PF274 feature set[7] for JPEG images. We have experimented with the cartesian-calibrated version of these features,[8] but despite its superior accuracy in targeted steganalysis we found its performance to be inferior. We believe that this discrepancy is due to larger dimension of the cartesion-calibrated features (548 comparing to 274). In order to fully exploit them, we would to use more samples per actor. In our experiments, we have used maximum 200 samples per actor, limited by the number of images that can be obtained in practice.

3.3 Comparison with prior art

We will wish to compare the universal pooled steganalyzer with "traditional" steganalysis such as one- or two-class support vector machines (SVMs). However, we must stress that a direct comparison is *not possible,* because traditional steganalysis provides a single positive/negative decision (or a single level of certainty) for a single image. Further work is necessary, therefore, to convert these individual steganalysis outcomes into an identification of guilty actor(s). This is the topic discussed in the original publication which posed the pooled steganalysis question.[1] Additionally, of course, most machine learning steganalysis methods must be trained, whereas our universal pooled steganalyzers are unsupervised, so the results must not be considered as a like-for-like comparison.

We will compare our methods by using one- and two-class support vector machines as feature pre-processing functions, reducing the feature vector for each image to one-dimensional quantity which can be then be fed into the LOF method for a decision about each actor. There may be other ways of using traditional steganalysis to solve pooled steganalysis problems, but to our knowledge none has been published.

3.4 Pre-processing of features

We found that pre-processing the extracted features significantly influences the accuracy of our methods. We used four types of pre-processing in this work, two of which are dimensionality reduction methods for comparison with prior art. We never used features without pre-processing, since the raw features have different scales and their influence should not be the same.

Let $\mathbf{X} \in \mathbb{R}^{nl,d}$ be the data matrix containing features extracted from all images of all actors. Every row of $\mathbf{X}$ corresponds to one feature vector. If we have n images from every actor, and there is l actors, than $\mathbf{X}$ will have nl rows and d columns, where d is number of steganalytic features (274 in case of PF274 features). Let $\tilde{\mathbf{X}} \in \mathbb{R}^{nl,d}$ denote the pre-processed version of $\mathbf{X}$.

3.4.1 Normalization

By *normalization*, we mean linear scaling of features such that every column of the data matrix $\tilde{\mathbf{X}}$ has zero mean and unit variance. This means that $(\forall r)\left(\frac{1}{nl}\sum_{i=1}^{nl}\tilde{\mathbf{X}}_{ir}=0\right)\left(\frac{1}{nl}\sum_{i=1}^{nl}\tilde{\mathbf{X}}_{ir}^2=1\right)$.

3.4.2 Principal Component Transformation (PCT)

By *principal component transformation*, we mean projecting rows of $\mathbf{X}$ into new space, where the elements of row vectors are not correlated. This means that

$$(\forall r)\left(\frac{1}{nl}\sum_{i=1}^{nl}\tilde{\mathbf{X}}_{ir}=0\right)\left(\frac{1}{nl}\sum_{i=1}^{nl}\tilde{\mathbf{X}}_{ir}^2=1\right) \quad \text{and} \quad (\forall r,\, s)\left(\frac{1}{nl}\sum_{i=1}^{nl}\tilde{\mathbf{X}}_{ir}\tilde{\mathbf{X}}_{is}=0\right).$$

In our application, all components with eigenvalues smaller than 0.01 are discarded, as they are not considered to hold sufficient information.

3.4.3 One-Class SVM Projection

To compare our methods with the one-class SVM,[9] we use a one-class SVM which has *not* been trained on known-clean cover data. First, the features are normalized as per subsect. 3.4.1. Then we train a one-class ν-SVM on 6000 randomly-selected rows of $\mathbf{X}$. Since these features are selected from the observed set, they will likely include some of stego images from the guilty actors, but the nature of a ν-SVM is to discard proportion ν of the training data as outliers. We set $\nu = 0.01$, which corresponds to discarding 1% of samples observed during training (it would be cheating to discard exactly the right proportion of stego objects, since a detector does not know how many there are). The ν-SVMs used a Gaussian kernel, with width set to the inverse of the median of distances between samples (the so-called median heuristic).

After the one-class SVM is trained, it is applied to each row of $\mathbf{X}$ to produce a scalar: this is the output *without thresholding*, corresponding to the distance of the observed feature from the separating hyperplane. This type of pre-processing can be viewed as a projection of features from $\mathbb{R}^d$ to $\mathbb{R}$, which can is a sort of measure of "being an outlier" similar to LOF, and results in a pre-processed vector $\tilde{\mathbf{X}} \in \mathbb{R}^{nl,1}$.

3.4.4 Two-Class SVM Projection

To apply two-class SVMs, we must use supervised learning. We trained two-class SVMs (with Gaussian kernel) to detect the nsF5 algorithm used in our experiments. Then the trained SVM was used in the same fashion as the one-class SVM described above, a projection from $\mathbb{R}^d$ to $\mathbb{R}$ measuring the distance to the separating hyperplane. In this case the application can be viewed as a type of pre-processing exploiting knowledge of the embedding algorithm. This gives us an insight as to how knowledge of the steganographic algorithm helps to improve the accuracy of detection, but it means that the results cannot be compared on a like-for-like basis with those of untrained universal steganalysis.

The features were normalized to the interval $[-1, +1]$ for two-class SVMs. This differs from the normalization of subsect. 3.4.1, but it is the norm for two-class SVMs. Training occurred before conducting any experiments: the images were divided into two parts, those from actors $\{1, \ldots, 3500\}$ to be used for all testing purposes, while images from actors $\{3501, \ldots, 4000\}$ used for this training. Soft-margin two-class SVMs were trained on 3000

cover images and 3000 stego images selected randomly from all images available for training. The payload in stego images was random as well, with a selection of payloads in the training set used to avoid over-fitting the classifiers to particular payload.[10] The two-class SVMs used the Gaussian kernel, which has been shown to produce the best outcomes.[11] The training of such SVMs is controlled by two parameters: the width of Gaussian kernel γ and penalization parameter C.[12] We have found suitable values of γ and C by utilizing the usual approach relying on estimating classification accuracy on a grid $(C,\gamma) \in \left\{(10^i, 2^j) \,|\, i \in \{1,\ldots,6\},\, j \in \{-10,\ldots,3\}\right\}$. The accuracy has been estimated by five-fold cross-validation and typical values of hyper-parameters were $C = 10^3$ and $\gamma = 2^{-9}$.

For the experiments described in next section, we have trained the total of 20 different realizations of two-class SVMs. All classifiers are trained following the methodology described above, but they differ in the images selected for the training set (the seed for random selection of the images was different). The intention of using more realizations of one classifier was to increase diversity in our experiments and to avoid using one particularly good/bad classifier. In every repetition of the experiments (as explained below), we have randomly selected one classifier out of the 20.

3.5 Distance calculation

Following the original method in Ref. 2 , the distance $d(i,j)$ between users i and j, is calculated as a maximum mean discrepancy (MMD) between feature vectors of i^{th} and j^{th} user. The general formula for MMD can be found in the literature,[13] but our previous work showed that the linear kernel performed better than MMD with Gaussian kernel, which was most probably due to better robustness against differences caused by different sources and image contents. Thus, we have used only a linear kernel, which makes the MMD calculation extremely simple: if I_i denotes indices of rows in the matrix $\mathbf{X}$ corresponding to the feature vectors extracted from all images of i^{th} user then simply

$$d(i,j) = \left\| \sum_{k \in I_i} \tilde{\mathbf{X}}_{k\cdot} - \sum_{k \in I_j} \tilde{\mathbf{X}}_{k\cdot} \right\|^2$$

where $\|-\|$ is the L_2 norm.

3.6 Image database

Our experiments were performed on a highly realistic data set, obtained from a leading social networking site. This site allowed users to make their photos open to the entire public, if they wished, and gave viewers the option to click "next photo" links to see the entire set of photos featuring each such user. Also, it offered links to photos of other users, tagged in the current photo, who chose to make theirs public too. Many users made their photographs public. By starting with a single public photo, and automating the process of clicking on these links, we downloaded over 4 million publicly-visible JPEGs from the site during December 2010. The crawl was restricted to users who publicly identified themselves with the Oxford University network and the data was then anonymized. We emphasize that only publicly-available photos were downloaded, and that no personally-identifiable information was retained.

Each photo is identified with the unique user who uploaded it. This enabled us to select a subset of 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. Our only caveat is the small possibility that some of the users were already using steganography in their images: if so, our set of "cover" images might not be entirely clean, and may contain a few stego images. Hopefully, use of steganography in social media is not (yet) widespread, at least in Oxford University.

These images are very difficult for conventional steganalysis: the social network offers to resize uploaded images to approximately 1 megapixel (but by no means to a uniform size), and also re-compresses almost every image to JPEG quality factor 85. This means that the cover images will have re-sampling and double-compression artefacts, the latter known particularly to affect steganalysis reliability.[11] The different users will be using different cameras, which makes the set heterogeneous (except for its ultimate compression factor, and

payload	HC	LOF	LOF - top 10
70	0	18	100
60	0	6	100
50	0	0	88
40	0	0	24
30	0	0	2

Table 1: Accuracy of identification of a guilty user using clustering (HC) and outlier detection (LOF), out of 100 experiments.

restricted range of images sizes). Some images are not even natural photographs: they may be montages, have captions, or be entirely synthetic. In a realistic scenario, we must deal with this type of difficult data.

The only way in which the data set has been screened was the removal of images smaller than 5KB (which deletes images with little or no content) or which do not follow the standard compression factor (this was less than 1% of the images downloaded).

4. EXPERIMENTAL RESULTS

4.1 Hierarchical clustering versus Local outlier Factor

Section 2 discussed why the approach relying on outlier detection (LOF) should be superior to the approach employing hierarchical clustering (HC). Here, results of a small-scale experiment are shown to quantify the advantages. The experiment used 400 randomly selected actors from actors $\{1, \ldots, 3500\}$, each emitting 100 images. One actor was always guilty and has embedded p% images each with $p/100$ bits per nonzero coefficient (bpnc). Both methods (local outlier factor and hierarchical clustering) used PCT pre-processing, since we found it to offer the best accuracy. The experiment was repeated 100 times for payloads $p \in \{30, 40, 50, 60, 70\}$ (more than 0.7bpnc cannot always be embedded using nsF5). The number of nearest neighbors in local outlier factor was set to $k = 10$.

The rate of correctly identifying the guilty user, out of 100 repeated experiments, is shown in Table 1. Neither method is able to identify the single guilty actor very often, but the hierarchical clustering method never manages a correct identification. We emphasize the issues related to the identification of a guilty actor in the case of hierarchical clustering: the original method in Ref. 2. randomly selects an actor from the smaller cluster in the last step of the hierarchical clustering, with the hope that the smaller cluster will contain only guilty actor(s). We have used this approach here as well. On the other hand, the local outlier factor method does not suffer from any ambiguity since it returns the degree of being outlier for every actor. The actor with the highest was judged as the guilty actor.

Clearly, it is difficult to make a precise identification of one guilty actor out of hundreds. The local outlier factor, however, gives us another option: to rank the actors by their degree of being an outlier, and make a shortlist of the most suspicious actors. Then we can measure how often the guilty actor appears in the top n on this list, or the top x%. The number of times, out of 100, that the true guilty actor was in the top 10 is displayed in the third column of Table 1 and we can see that the guilty actor is in this guilty short-list more often than not, for $p \in \{50, 60, 70\}$. Ranking the top n suspicious actors is not available using the HC method.

4.2 Large scale experiments

The large-scale experiment uses a single guilty actor. It investigates how the LOF method performs with different pre-processing (including supervised pre-processing) and how performance varies with the total number of actors and images in a realistic environment. The experiments were performed on a randomly selected actors from numbers $\{1, \ldots, 3500\}$($a \in \{100, 200, 400, 800, 1600, 3200\}$). One guilty actor emits p% stego-images with $p/100$ bpnc ($p \in \{30, 40, 50, 60, 70\}$). The steganalyst has m images (samples) from each actor ($m \in \{50, 100, 200\}$) .

The steganalyst ranks the most suspicious actors using the local outlier factor with $k = 10$. The four types of pre-processing described in Section 3.4 were tested: normalization, principal component transformation (PCT),

		pre-processing			
	payload	Normalization	PCT	One-class	Two-class
400 actors	30	5	5	3	83
	40	3	3	5	95
	50	11	31	3	100
	60	45	63	18	100
	70	65	91	40	100
3200 actors	30	5	2	3	47
	40	6	2	2	90
	50	68	8	2	98
	60	100	98	2	100
	70	100	100	10	100

Table 2: Rate (in %) of identification guilty actor among 1% most suspicious actors.

one-class SVM trained on the observed images (One-class), and two-class support vector machines trained on a separate set of images (Two-class). Notice that while first three types of pre-processings are universal, in that they do not utilize any external information such as the image source or the type of steganographic embedding, the fourth pre-processing requires knowledge of the embedding algorithm and a set of training images. This detector is included to observe how the accuracy of the detection increases when utilizing the additional knowledge.

These experiments, for each combination of parameters were performed a total of 62 times[†]and the actors ranked by their LOF in each case.

Figure 1 and 2 shows the rate of identifying the guilty actor amongst the top 10 and 1%, respectively, most suspicious actors. We can see that out of three universal detectors, the one utilizing principal component transformation works the best on the smaller number of actors (800 actors and less). On higher number of actors (1600 and 3200), plain normalization is slightly better. This phenomenon, which we do not understand yet, is nicely seen in Table 2, where the absolute values for for 400 and 3200 are tabulated. One of possible explanations might be the sensitivity of principal component transformation to outliers. Switching to a robust version of PCT might have some merit.

Figures 1 and 2 also shows that the accuracy measured as a rate of identification of guilty actors among top $x\%$ actors increases with the number of actors, while the same quantity related to top n actors decreases. This shows that that the accuracy of the method is not a linear function of the number of actors.

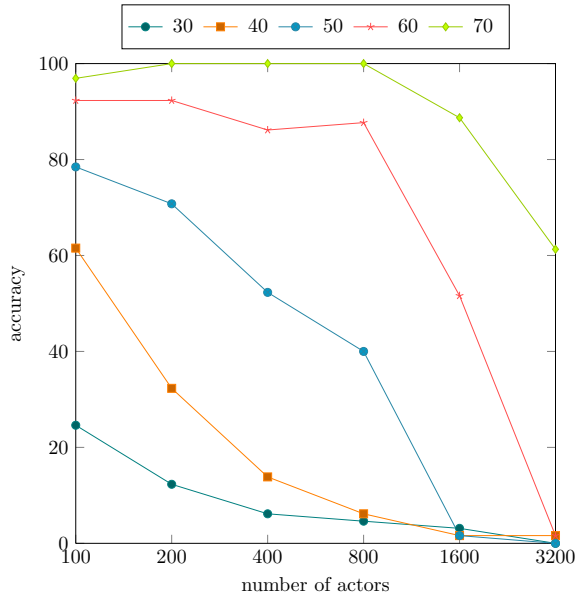
Finally, Figure 3 shows the accuracy of LOF method with PCT pre-processing with $m \in \{50, 100, 200\}$ images per actor. It is rather surprising to observe that the results are very similar, with little advantage to the steganalyst if the actors emit more than 50 images each. This is a curious result.

4.3 Detecting multiple guilty actors

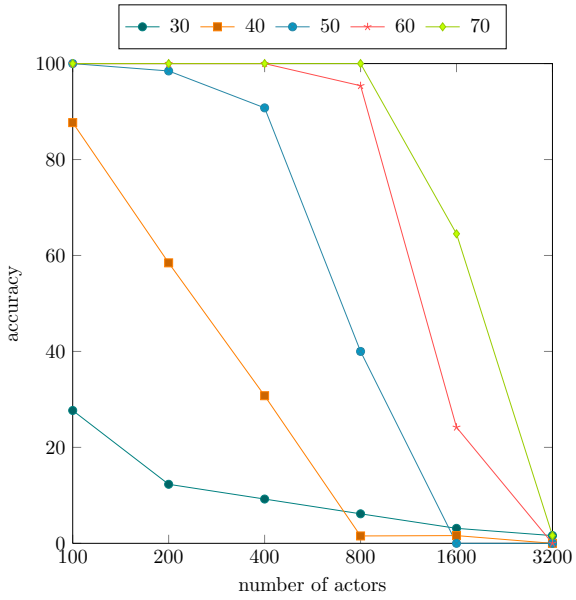
We briefly performed experiments to investigate whether the LOF method can be used with multiple guilty actors. We have fixed the number of actors to $a = 100$ and the number of images per actor to $m = 100$. As before, guilty actors have emitted $p\%$ stego images with $p/100$ bpnc. The steganalyst used the LOF to rank the actors with PCT pre-preocessing.

The rate of seeing guilty actors in the top 1% most suspicious actors is shown in Table 3 for 1, 2, 4, and 8 guilty actors. We can observe the the accuracy of the method quickly decreases as the number of guilty actors increases. This result shows a minor glitch of the LOF method. If there are more guilty users, their feature vectors start to occupy the same parts of space. Since the local outlier factor method measures the level of being outlier according to the neighbors, more outliers in the same space will be classified as "inliers".

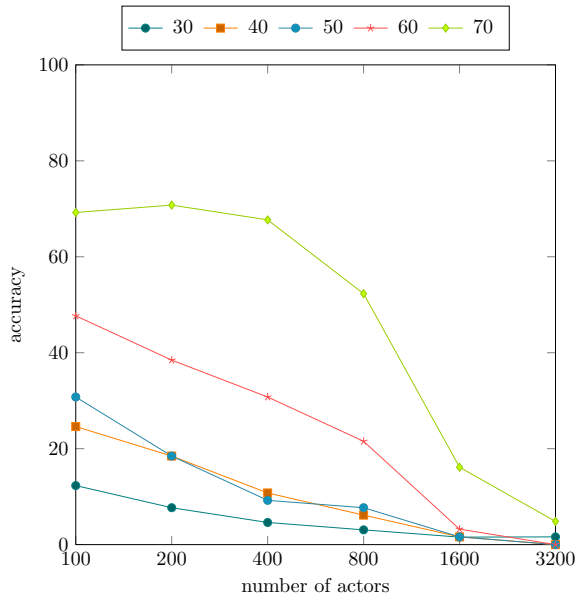
[†]Because of the need to identify k-nearest neighbours in large data sets, it takes a long time to perform each experiment. We had hoped for 100 repetitions, but software failure prevented us from reaching this target in time for publication.



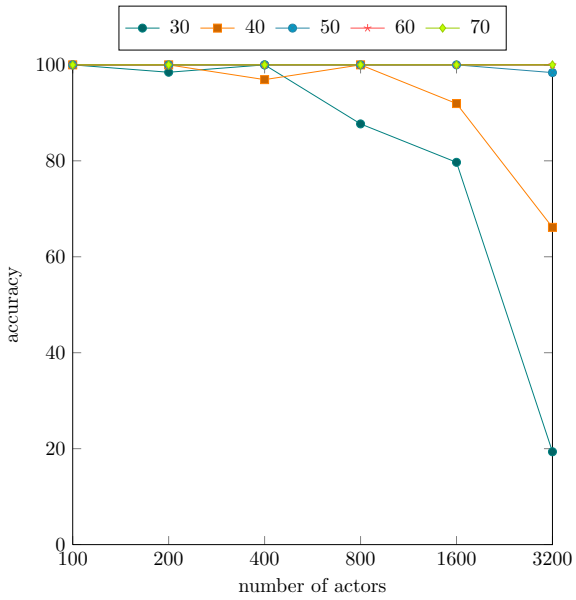
(a) Normalization + LOF



(b) PCT + LOF

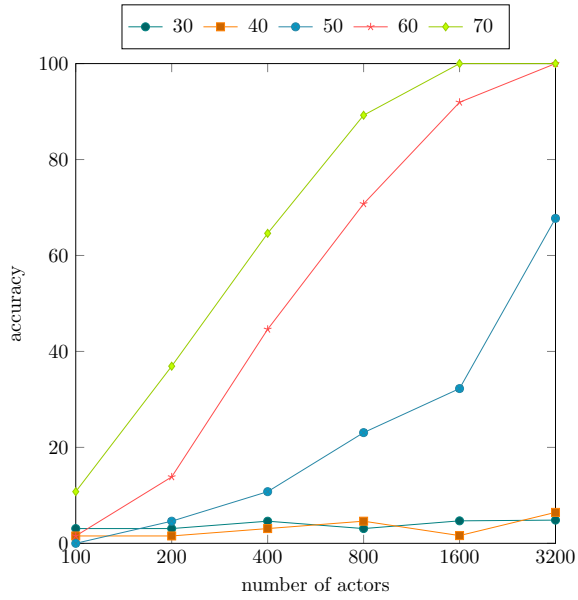


(c) One-Class SVM + LOF

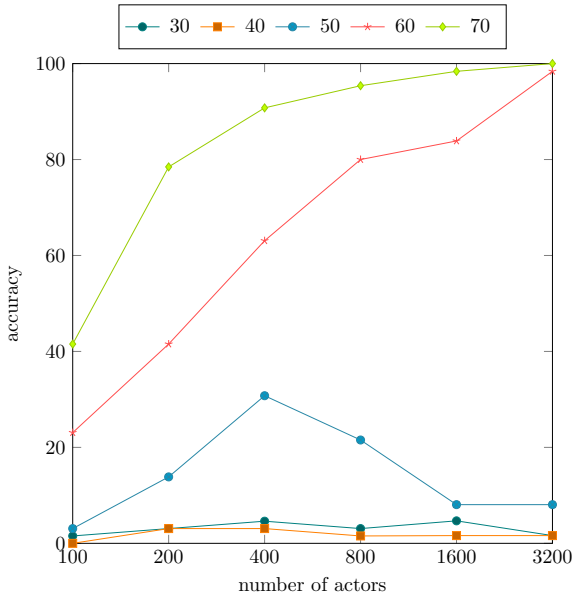


(d) Two-Class SVM + LOF

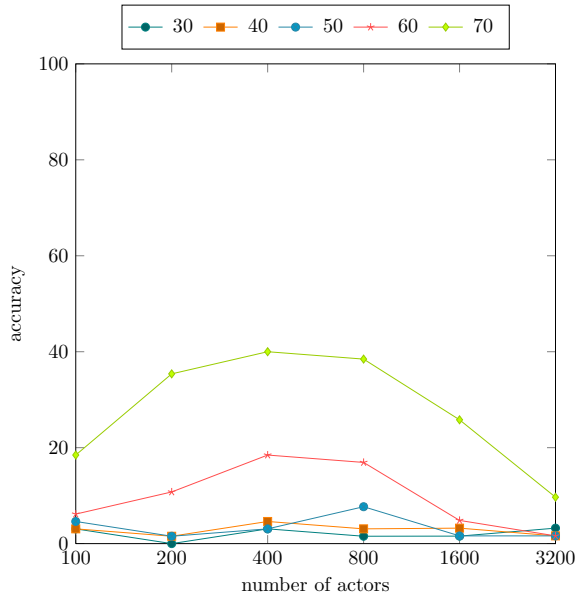
Figure 1: Rate (in %) of identifying the guilty actor in the 10 most suspicious actors, when 100 samples per actor are available.



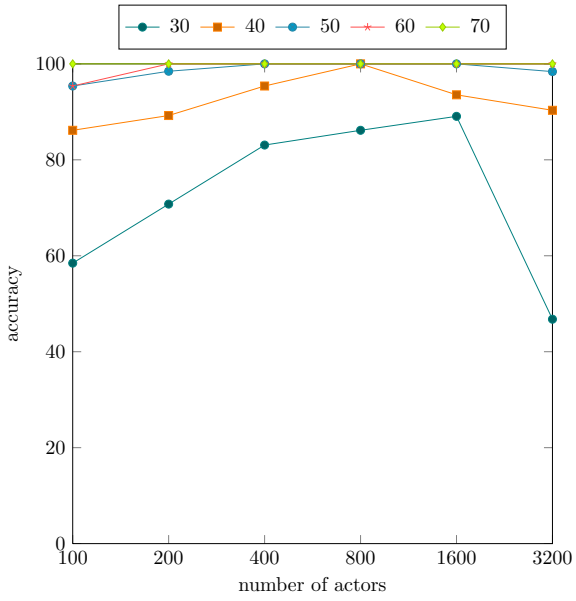
(a) Normalization + LOF



(b) PCT + LOF

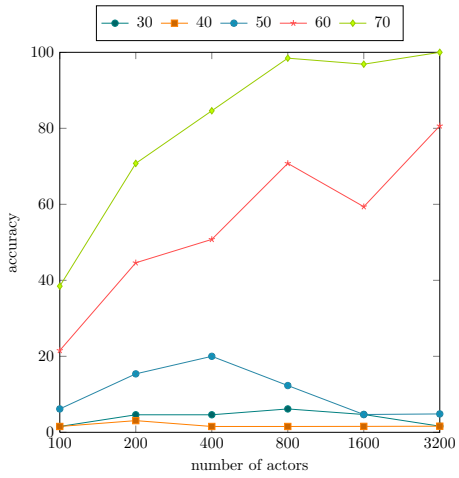


(c) One-Class SVM + LOF

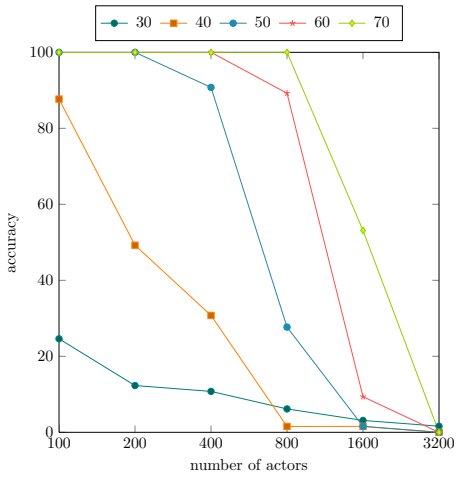


(d) Two-Class SVM + LOF

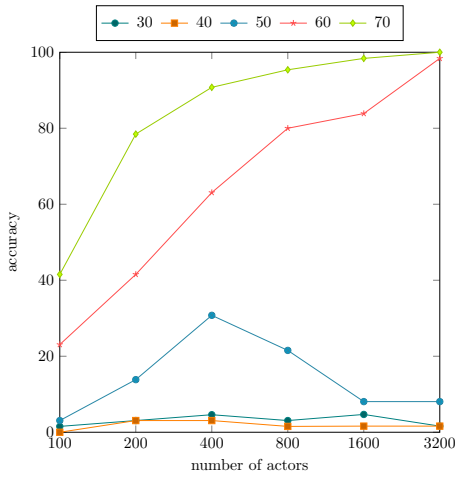
Figure 2: Rate (in %) of identifying the guilty actor in the1% most suspicious actors, when 100 samples per actor are available.



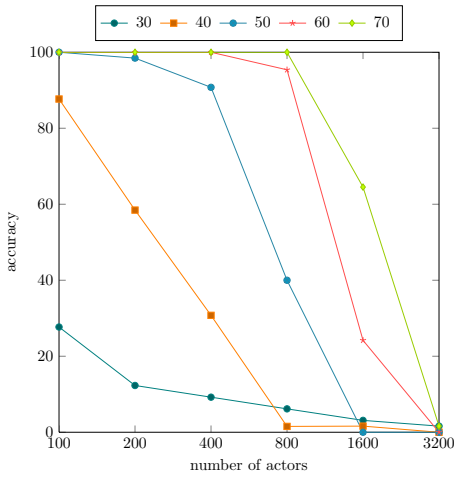
(a) 50 images per actor, top 1%



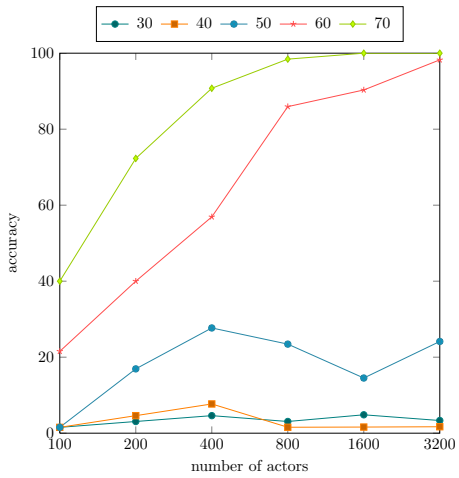
(b) 50 images per actor, top 10



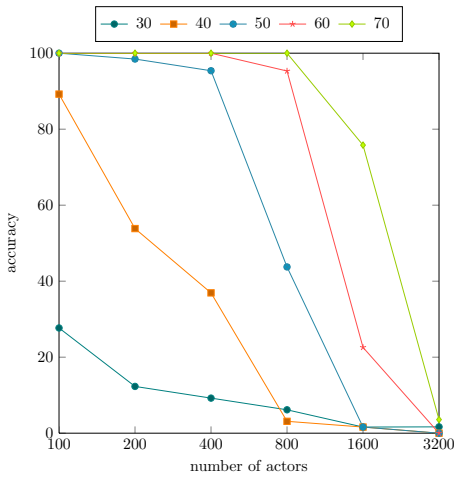
(c) 100 images per actor, top 1%



(d) 100 images per actor, top 10



(e) 200 images per actor, top 1%



(f) 200 images per actor, top 10

Figure 3: Rate (in %) of identifying the guilty actor in the 1% / 10 most suspicious actors.

	#guilty users			
payload	1	2	4	8
30	0	2	5	6
40	0	6	0	6
50	25	16	3	3
60	67	78	26	6
70	92	139	60	5

Table 3: Rate of guilty actors being identified among 1% of most suspicious actors. The experiment used 400 actors, 100 images per actor, and guilty actor has emitted p% stego images with $p/100$ bpnc. Rates of greater than 100 are possible when more than one guilty actor appears in the top 1% most suspicious.

	one guilty			two guilty		
payload	$k = 10$	$k = 20$	mix	$k = 10$	$k = 20$	mix
30	0	0	0	2	2	2
40	0	0	0	6	4	5
50	25	10	10	16	14	13
60	67	58	60	78	76	76
70	92	91	90	139	139	136

	four guilty			eighty guilty		
	$k = 10$	$k = 20$	mix	$k = 10$	$k = 20$	mix
30	5	6	6	6	6	6
40	0	1	1	6	9	9
50	3	3	3	3	6	6
60	26	28	27	6	10	10
70	60	91	80	5	18	16

Table 4: Rate (in %) of identifying guilty actor(s) among 1% of most suspicious actors. The experiment used 400 actors, 100 images per actor, and guilty actor has emitted p% stego images with $p/100$ bpnc.

This weakness of LOF method has been already described in Ref. 3. The remedy proposed in the same publication is to execute the LOF method with different k and take maximum LOF values for every actor. We have tested this approach by running the LOF method twice with $k = 10$ and $k = 20$. The results are shown in Table 4. We observe that increasing k helps, but the problems remains if the number of guilty actors is high.

5. CONCLUSIONS

We propose the use of local outlier factor (LOF) as a measure of atypical behavior for multiple-actor, multiple-image, pooled steganalysis. This allows the actors to be compared with each other and ranked according to their level of being an outlier. When used with a steganalysis feature set, this gives a way to find the guilty actor(s) without knowledge of the embedding algorithm or any training whatsoever. We have shown that its performance, with the PF-274 feature set, is considerably better than the use of one-class SVMs but inferior to trained two-class SVMs. However, in the real world training is not always possible (in particular it requires that the embedding algorithm is known) or is likely subject to training model mismatch.

We have benchmarked this method in a highly realistic data set, obtained from a social networking site. These images are diverse and have recompression and image-processing artefacts which make it difficult to perform steganalysis in. The advantage of the method proposed here is that it takes into account the difference between actors' image sources, and judges the guilty actor(s) not by those which differ from the others, but by those which differ *more than expected* from the others.

In these experiments a single guilty actor was not always correctly determined, but was ranked in the top 10 or 1% of most suspicious actors in a high proportion of cases when payloads of at least 0.5 bpnc were embedded

in at least half of the guilty actor's covers. This is even true when there are thousands of actors. The LOF method can be used, therefore, for drawing up a short list of suspicious actors on whom further investigation could be performed. It seems that the number of images per actor is relatively unimportant to the performance, a strange finding which merits further investigation.

We briefly tested the same method with multiple guilty actors, where it seems to work less well, perhaps because the guilty actors start to look like a cluster of innocent actors. The investigation of this issue is the future work. Although this weakness might be inherent to the approach, we hope to mitigate it by pooling outputs of many diverse outlier detectors. This approach has been already used in network intrusion detection with good results.[14]

6. 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.

REFERENCES

[1] Ker, A. D., "Batch steganography and pooled steganalysis," in [*Proceedings of the 8th international conference on Information hiding*], *IH'06*, 265–281, Springer-Verlag, Berlin, Heidelberg (2007).

[2] Ker, A. D. and Pevný, T., "A new paradigm for steganalysis via clustering," *Media Watermarking, Security, and Forensics III* **7880**(1), 78800U, SPIE (2011).

[3] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J., "Lof: identifying density-based local outliers," in [*Proceedings of the 2000 ACM SIGMOD international conference on Management of data*], *SIGMOD '00*, 93–104, ACM, New York, NY, USA (2000).

[4] Pevný, T., Fridrich, J., and Ker, A. D., "From blind to quantitative steganalysis," *IEEE Transactions on Information Forensics and Security* (2012). To appear.

[5] Chandola, V., Banerjee, A., and Kumar, V., "Anomaly detection: A survey," *ACM Comput. Surv.* **41**, 15:1–15:58 (July 2009).

[6] Fridrich, J., Pevný, T., and Kodovský, J., "Statistically undetectable JPEG steganography: dead ends challenges, and opportunities," in [*Proceedings of the 9th workshop on Multimedia & security*], *MM&Sec '07*, 3–14, ACM, New York, NY, USA (2007).

[7] Pevný, T. and Fridrich, J., "Merging Markov and DCT features for multi-class JPEG steganalysis," in [*Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX, San Jose, CA, January 29–February 1*], Delp, E. and Wong, P., eds., **6505**, 03–14 (January 2007).

[8] Kodovský, J. and Fridrich, J., "Calibration revisited," in [*Proceedings of the 11th ACM workshop on Multimedia and security*], *MM&Sec '09*, 63–74, ACM, New York, NY, USA (2009).

[9] Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C., "Estimating the support of a high-dimensional distribution," *Neural Computation* **13**(7) (2001).

[10] Pevný, T., "Detecting messages of unknown length," *Media Watermarking, Security, and Forensics III* **7880**(1), 78800T, SPIE (2011).

[11] Pevný, T., *Kernel Methods in Steganalysis*, PhD thesis, Binghamton University, SUNY (May 2008).

[12] Schölkopf, B. and Smola, A., [*Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*], The MIT Press (2001).

[13] Gretton, A., Borgwardt, K. M., Rasch, M., Schólkopf, B., and Smola, A. J., "A kernel method for the two-sample-problem," in [*Advances in Neural Information Processing Systems 19*], Schölkopf, B., Platt, J., and Hoffman, T., eds., 513–520, MIT Press, Cambridge, MA (2007).

[14] Rehák, M., Pechoucek, M., Grill, M., Stiborek, J., Bartoš, K., and Celeda, P., "Adaptive multiagent system for network traffic monitoring," *IEEE Intelligent Systems* **24**(3), 16–25 (2009).