DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-075

Separating Steganographic Images by Embedding Key

Tu-Thach Quach; University of New Mexico; Albuquerque, NM, USA

Abstract

Payload location and key search techniques often rely on having a collection of stego images that use the same embedding key. This is a plausible scenario if the steganographer reuses the embedding key. In practice, however, the collection may contain several embedding keys. As a consequence, we must be able to separate these stego images by embedding key prior to applying these attacks. This paper addresses this fundamental problem. We first investigate the situation where the cover images are also available. Our analysis shows that it is possible to determine whether two stego images share the same embedding key provided that the payload sizes are not too large. We then present a practical algorithm to separate stego images by embedding key in the case when the cover images are not available. Our algorithm uses a cover estimator to reconstruct the cover images. Using the residuals computed from the stego-estimate image pairs, our algorithm separates the stego images via spectral clustering. Experimental results show that our algorithm performs well against popular matrix embedding schemes. Once separated, we perform exhaustive key search, using the computed residuals, to recover the embedding key.

Introduction

Digital image steganography often embeds the payload by making small changes to the cover image using least significant bit (LSB) matching so that the resulting stego image appears innocuous to an unintended observer. Popular embedding algorithms include simple LSB matching and matrix embedding [1]. These algorithms further utilize an embedding key to distribute the payload over the entire image, making it more difficult to detect by steganalysis detectors. If detected, the use of an embedding key also complicates the task of extracting the hidden message as information about the location of the payload is unknown.

Despite this difficulty, if the steganographer reuses the embedding key, it is possible to locate and extract the payload [2, 3, 4, 5, 6, 7, 8]. The success of these techniques relies primarily on having a collection of stego images that use the same embedding key. This is a plausible scenario if the steganographer reuses the embedding key and the images are the same size. As an example, the steganographer takes several pictures using his digital camera and then embeds messages into these pictures using the same password, e.g., embedding key. In practice, the stego image collection may consist of not one, but several embedding keys, reflecting the fact that the steganographer may change the embedding key over time. This is analogous to a user changing their computer password.

Taking this into consideration, our first task is to separate these stego images by embedding key so that the subsequent analysis can be performed. Our current work addresses this fundamental problem. To the best of our knowledge, no existing research has explored the key separation problem. We first assume that the cover images are available and approach the problem from a hypothesis testing perspective: given two stego images, determine whether they use the same embedding key. We show that it is possible to make this distinction provided that the payload sizes are not too large. We then propose a practical algorithm to separate stego images by embedding key when the cover images are not available. The algorithm uses existing cover estimators to estimate the cover images. It then computes residuals using the stego-estimate pairs. The residuals are used to partition the stego images by embedding key via spectral clustering. Our algorithm performs well against popular matrix embedding schemes in experiments using real images. Once separated, it is straightforward to recover the embedding key via a brute-force key search procedure that makes use of the computed residuals.

In the next section, we investigate the key-separation problem under the setting that the cover images are known. We then present the practical algorithm addressing the situation when the cover images are unavailable. Our key search algorithm is also presented in that same section. We follow with experimental results demonstrating the effectiveness of our algorithm. Concluding thoughts are provided in the last section.

Key Separation

We consider the class of steganographic algorithms that embeds b bits using k pixels. To embed a payload of m bits into cover image $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ to generate stego image $\mathbf{s} = (s_1, s_2, \ldots, s_n)$, we need to use $\lceil \frac{km}{b} \rceil \leq n$ pixels. For simplicity, we assume that b divides m so that the number of payload pixels is $\frac{km}{b}$. In order to embed the payload, some payload pixels may need to be modified. The probability that any payload pixel is changed is p_c. This class of embedding algorithms encompasses a wide range of practical algorithms including simple LSB matching[1], group-parity steganography, and the entire family of matrix embedding algorithms based on Hamming codes [1]. For simple LSB matching, we have $k = 1$, $b = 1$, and $p_c = \frac{1}{2}$. For the popular (3, 2) matrix embedding, we have $k = 3$, $b = 2$, and $p_c = \frac{1}{4}$.

In general, an embedding key is used to shuffle the pixels so that the payload is distributed over the entire image. The embedding key is shared with the receiver so that the message can later be extracted. If the steganographer reuses the embedding key, the same pixel order is produced by the shuffling process provided that the cover images are the same size. Conversely, if the steganographer changes the embedding key, a different pixel order is used to embed the subsequent payloads. This leads to our current problem: given stego images $\mathbf{s}$ carrying a payload of m_1 bits and $\mathbf{s}'$ carrying a payload of m_2 bits, both of length n, determine whether they use the same embedding key.

In the following analysis, we assume the cover images, $\mathbf{c}$ and

[1]We note that simple LSB matching is (1, 1) matrix embedding and will use both names interchangeably throughout the paper.

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-075

$\mathbf{c}'$, are known or can be estimated with vanishing error rate [9]. The analysis provides insight on how well we can expect to address this problem in practice. Using $\mathbf{c}$ and $\mathbf{s}$, it is straightforward to form the residual vector $\mathbf{r}$, where $r_i = |c_i - s_i|$, that identifies the modified pixels. The residual vector $\mathbf{r}'$ is constructed similarly using $\mathbf{c}'$ and $\mathbf{s}'$. The number of overlapping modified pixels is $x = \sum_i r_i r_i'$. Using x, we decide whether $\mathbf{s}$ and $\mathbf{s}'$ use the same embedding key. This is a hypothesis testing problem. Our hypotheses are

H_0: $\mathbf{s}$ and $\mathbf{s}'$ use different embedding keys,
H_1: $\mathbf{s}$ and $\mathbf{s}'$ use the same embedding key.

We accept H_0 when $p(x|H_0) > p(x|H_1)$ and accept H_1 otherwise.

We now construct $p(x|H_0)$ and $p(x|H_1)$. Let $n_1 = \frac{km_1}{b}$ and $n_2 = \frac{km_2}{b}$. Without loss of generality, let $n_1 \leq n_2$. Under H_1, the number of shared payload pixels in both stego images is n_1, e.g., the cardinality of the intersection of the payload pixels used in both images. Since p_c is the probability of modifying any payload pixel, the probability that a payload pixel is changed in both images is p_c^2. Since x is the number of payload pixels changed in both stego images, $p(x|H_1)$ is a binomial distribution:

$$p(x|H_1) = \binom{n_1}{x} \left(p_c^2\right)^x \left(1 - p_c^2\right)^{n_1 - x}. \quad (1)$$

Under H_0, let l be the number of shared payload pixels in both stego images. Since the two keys are different, its distribution is hypergeometric:

$$p(l|H_0) = \frac{\binom{n_2}{l}\binom{n-n_2}{n_1-l}}{\binom{n}{n_1}}. \quad (2)$$

Similar to H_1, given l,

$$p(x|l, H_0) = \binom{l}{x} \left(p_c^2\right)^x \left(1 - p_c^2\right)^{l-x}. \quad (3)$$

Combining (2) and (3), we have

$$\begin{aligned} p(x|H_0) &= \sum_l p(l|H_0) p(x|l, H_0) \quad (4) \\ &= \sum_l \frac{\binom{n_2}{l}\binom{n-n_2}{n_1-l}}{\binom{n}{n_1}} \binom{l}{x} \left(p_c^2\right)^x \left(1 - p_c^2\right)^{l-x}, \quad (5) \end{aligned}$$

where the summation over l is from $\max\{x, n_1 + n_2 - n\}$ to n_1.

To further support our analysis, we perform the following experiment using images from the BOSSbase 0.92 database [10], which consists of 9074 images of size 512-by-512 in the raw PGM format. We randomly select 1000 images. For each image, we randomly crop a 25-by-40 region to form a cover image, e.g., $n = 1000$ pixels. For each cover image, we embed a relative payload size of 0.4 bits per pixel (bpp), or 400 bits, using simple LSB matching with different keys. We compute x using all unique image pairs and plot the obtained distribution in Figure 1(a). The plot also shows the expected distribution from (5). We repeat the same experiment, this time using the same embedding key to generate the stego images. The resulting distribution along with the expected distribution from (1) are shown in Figure 1(b). The plots confirm that the experimental results match our analysis.

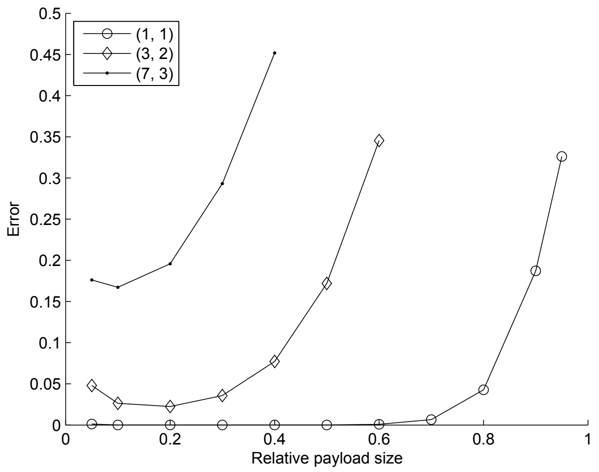


Figure 2. *The error probability for three matrix embedding algorithms at different payload sizes. We can expect to be most successful against simple LSB matching, or (1, 1) matrix embedding. In all cases, the error increases as payload size gets sufficiently large.*

An important metric in hypothesis testing is the error probability. Let the acceptance region for H_0 be $\mathscr{R}_0 = \{x : p(x|H_0) > p(x|H_1)\}$. Similarly, let $\mathscr{R}_1 = \{x : p(x|H_0) \leq p(x|H_1)\}$ be the acceptance region for H_1. The error probability is

$$p_E = \frac{1}{2} \sum_{x \in \mathscr{R}_0} p(x|H_1) + \frac{1}{2} \sum_{x \in \mathscr{R}_1} p(x|H_0). \quad (6)$$

With $n = 1000$, we compute p_E at different relative payload sizes for 3 different embedding algorithms: (1, 1), (3, 2), and (7, 3) matrix embedding. The results are shown in Figure 2. It is clear that our success depends on the embedding algorithm and the payload size. Among the three algorithms, we can expect to be most successful against (1, 1) matrix embedding, e.g., simple LSB matching. In all cases, however, our success is limited as payload size becomes sufficiently large. This is expected as our decision is based primarily on the number of overlapping modified pixels, x. For large payload sizes, the majority of the pixels are payload pixels. This remains true regardless of whether or not the steganographer reuses the embedding key. For the (7, 3) matrix embedding scheme with a relative payload size of 0.4 bpp, the proportion of payload pixels is 0.93. In this case, the error probability is only slightly better than random guessing.

Practical Algorithm

While informative, the above analysis assumes the cover images are known. In practice, the forensic analyst may have a collection of stego images obtained from the steganographer, but the cover images are not available. In this case, the cover images must be estimated using existing estimators [4, 5, 7]. These estimators, however, are noisy. As a consequence, the resulting estimate and the cover image can differ significantly. Applying the hypothesis testing framework would have limited success.

The problem can still be addressed by recognizing the fact that while each individual estimate is noisy, their aggregate is not. This is the exact reason why steganographic payload loca-

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-075

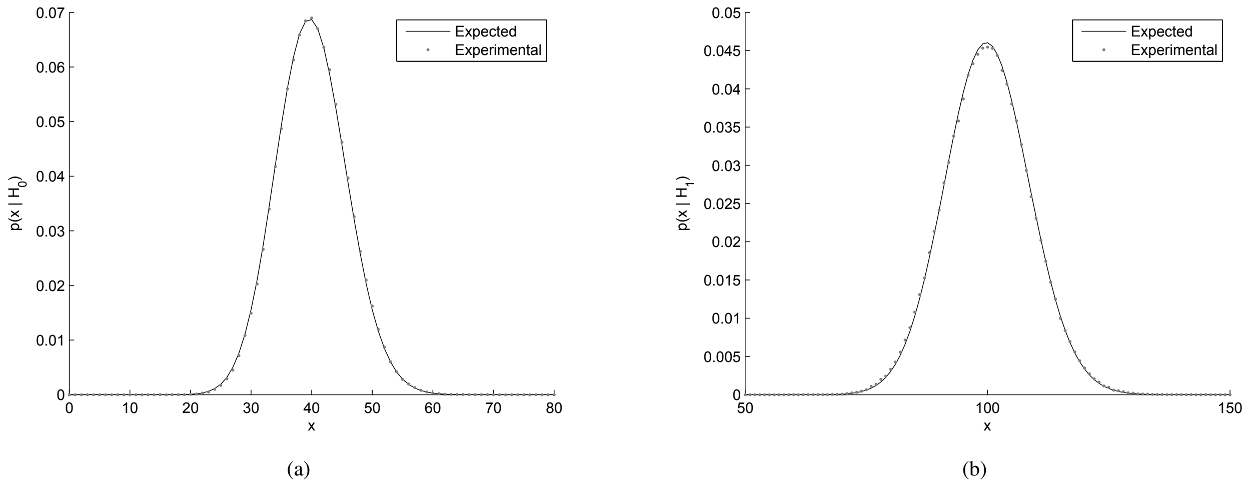


Figure 1. *The empirical distribution of x, the number of overlapping modified payload pixels in both stego images, under (a) H_0 and (b) H_1 for simple LSB matching at 0.4 bpp. The results match our expectation.*

tion works. Using this insight, we present a practical approach that makes use of a collection of stego images. We assume the forensic analyst has a collection of N stego images, each of size n, that may contain several embedding keys. It is important to note that we do not assume any knowledge about the number of embedding keys. Under this setting, we propose to partition these stego images by embedding key using the following spectral clustering algorithm [11]:

1. Estimate the cover images.
2. Compute the residual vector for each stego-estimate pair.
3. Form the N-by-n residual matrix $\mathbf{R}$ by stacking the residual vectors vertically.
4. Compute the similarity matrix $\mathbf{X} = \mathbf{R}\mathbf{R}^T$.
5. Let $\mathbf{D}$ be the diagonal matrix where $D_{ii} = \sum_j X_{ij}$.
6. Compute $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{X}\mathbf{D}^{-1/2}$.
7. Compute d eigenvectors of $\mathbf{L}$ that correspond to the largest d unique eigenvalues.
8. Form the N-by-d feature matrix $\mathbf{F}$ using these eigenvectors.
9. Normalize $\mathbf{F}$ so that each row has unit-length norm.
10. Cluster the rows of $\mathbf{F}$ using agglomerative hierarchical clustering.

Steps 4 - 9 are standard in spectral clustering. The original algorithm assumes that the number of clusters is known and sets d to be this quantity. In our current setting, this is not true. This is why we use agglomerative hierarchical clustering in the last step. This clustering technique builds a tree of clusters, starting with each object as its own cluster, and repeatedly combines two nearest clusters until a single cluster is formed. This method requires a function to compute the distance between two clusters. We use the average Euclidean distance. Once the tree is formed, we can cut it at some depth to partition it into several clusters. This flexibility gives the analyst full control over the maximum number of clusters to examine, which determines the value of d. The analyst may first partition the image collection into two clusters and examine them. If no useful information is obtained, the analyst may further cut the tree to obtain four clusters to examine, etc. If the majority of the stego images in any cluster use the same embedding key, the analyst may be able to locate the payload or perform key search on that cluster. Note that success on a single cluster may allow the analyst to obtain valuable information that allows for subsequent attacks on the remaining images.

In order to quantify the effectiveness of our approach, we define the following accuracy metric: $\max_{i,j} g_{ij}$, where g_{ij} is the proportion of images in the i-th cluster that use the j-th embedding key. This quantity captures how well we group stego images that use the same embedding key. It is only meaningful when the cluster size is relatively large. A cluster of size 1 is meaningless. As a consequence, in our experiments, we provide both the accuracy and the size of the corresponding cluster.

Once separated, each image set can be further processed to extract the hidden messages. Given a collection of N stego images, each carry a payload of m bits, that share the same embedding key, the residual of pixel i in image j is r_{ij}. The mean residual of pixel i is

$$\bar{r}_i = \frac{1}{N}\sum_{j=1}^{N} r_{ij}. \qquad (7)$$

Given an exhaustive collection of embedding keys, it is straightforward to use the mean residuals to search for the correct embedding key. Specifically, let $E(m)$ be the set of pixels returned by embedding key E to embed m bits. The total mean residual for embedding key E is $\sum_{i\in E(m)} \bar{r}_i$. We can perform a brute-force search over the key collection to find the key that the steganographer used. The right key is the outlier with the largest total mean residual. Our key-search technique is different from a previous technique [12]; we do not assume prior knowledge about the embedded message, e.g., random binary sequence. Both methods, however, have limited success against sophisticated embedding algorithms that make use of the entire image to embed the payload [13]. This is expected as the total mean residuals are identical for such embedding schemes. Our key-search method is complementary to a recent technique that exploits embedded metadata, namely message length [14]. As pointed out in that work, instead

DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-075

of embedding the message length, a sophisticated steganographer may choose to use a special character to denote the end of the message. Doing so denies key search techniques that rely on such metadata, but comes at a cost of having to use a fixed payload size. This situation, however, is ideal for our proposed key search technique.

Experiments

The following experiments are designed to evaluate the effectiveness of our key-separation algorithm against different embedding schemes, payload sizes, and the number of stego images, N, in the collection. We use images from the BOSSbase 0.92 database. The two embedding algorithms are simple LSB matching, or (1, 1) matrix embedding, and (3, 2) matrix embedding. We use the Markov random field (MRF) cover estimator [7]. This estimator needs a set of cover images to learn the joint pixel probabilities. We randomly select 8074 images for this purpose and set aside the remaining 1000 images to test our algorithm. The MRF cover estimator also has 2 model parameters: w and ρ. Parameter w controls the effect of neighboring pixels and ρ is the likelihood of a pixel being modified. In all of our experiments, we use the default parameter setting: $w = 1$ and $\rho = 0.25$. There is a good reason to set $\rho = 0.25$ even if the payload size is small. If we set ρ to a small quantity, the estimator might produce estimates that are identical to the stego images, e.g., unable to detect any changes. This would not provide any useful information.

Using the remaining 1000 images, we randomly select $\frac{N}{2}$ images to form image set $\mathscr{I}_1$ and another $\frac{N}{2}$ images to form image set $\mathscr{I}_2$. These two image sets are non-overlapping. We then embed a random payload into each image in both sets. All images in the same set share the same embedding key. The embedding key used in $\mathscr{I}_1$ is different than the one used in $\mathscr{I}_2$. The resulting image collection consists of N stego images embedded by two different keys. We apply our key-separation algorithm to this image collection. To make the results more precise, we average the accuracies and cluster sizes over 10 different formations of $\mathscr{I}_1$ and $\mathscr{I}_2$.

In the first experiment, we fix $N = 1000$ and vary the payload size from 0.1 bpp to 0.5 bpp in increments of 0.1 bpp. This allows us to evaluate the algorithm under different payload sizes. The results are shown in Figure 3. Similar to our analysis, we are more successful against simple LSB matching than (3, 2) matrix embedding. Further, as payload size gets sufficiently large, accuracy decreases. For (3, 2) matrix embedding, we do not get high accuracies until we use more clusters. With 8 clusters, we achieve more than 0.93 accuracy for payload sizes of up to 0.4 bpp. We emphasize that these clusters do provide useful information as evident by their sizes. With 8 clusters, the average cluster size is 151.48 for simple LSB matching and 113.22 for (3, 2) matrix embedding.

In our second experiment, we fix the payload size at 0.4 bpp and vary N from 200 to 1000 in increments of 200. This allows us to evaluate the effectiveness of our algorithm as a function of the number of stego images in the collection. The results are shown in Figure 4. Once again, we obtain high accuracies by using more clusters. With 8 clusters, the average accuracy is 0.999 for simple LSB matching and 0.959 for (3, 2) matrix embedding. Accuracies remain relatively constant except when $N = 200$. This is due to small cluster sizes. With $N = 200$, the average cluster size is 7.2 for simple LSB matching and 9.6 for (3, 2) matrix embedding. Cluster size increases as N increases, suggesting that these clusters capture the statistics needed to separate stego images by embedding key.

As mentioned earlier, once the stego images are separated by embedding key, we can perform a brute-force search on a collection of embedding keys to find the one that the steganographer used. Using 5 randomly chosen stego images (and their cover estimates), each carry a payload of 0.1 bpp, generated by (3, 2) matrix embedding with the same key, we compute the total mean residuals for 10^6 keys, one of which is the same key used to generate the stego images. The histogram of the total mean residuals is shown in Figure 5(a). The correct key corresponds to the outlier with the largest total mean residual. It is not necessary to require a collection of stego images to perform key search. With a good cover estimator, such as the MRF cover estimator, the key search procedure can still find the correct key using a single stego image. We repeat the experiment, this time using only a single stego image. The histogram of the total mean residuals is shown in Figure 5(b). Again, the correct key corresponds to the outlier with the largest total mean residual. The distance separating the outlier from the rest, however, is smaller compared to the previous case of multiple stego images (226.3 vs 603.8). This is expected because the residuals are noisier with a single estimate.

Conclusion

The ability to separate stego images by embedding key is important in steganalytic forensic analysis. It allows the forensic analyst to further process each partition to locate and extract the hidden messages. The presented approach is the first to address this problem. Since the algorithm relies on estimating the cover images, its accuracy should improve with advances in cover estimation. Even though our experiments use spatial domain embedding, it should be clear that our approach applies to other embedding domains such as JPEG coefficients.

The results of this work present an interesting dilemma to the steganographer. It is well known that it is easier to detect stego images if they carry large payloads. So to minimize detection, the steganographer may hide smaller payloads. If detected, however, these stego images can be separated by embedding key, which may lead to a successful key recovery. As mentioned earlier, to evade our method, the steganographer can turn to adaptive embedding algorithms that make use of the entire image. Extending our method to address these embedding algorithms is the focus of our future work.

References

[1] Jessica Fridrich and David Soukal, Matrix embedding for large payloads, IEEE Trans. Inf. Forensics Security, 1, 3 (2006).

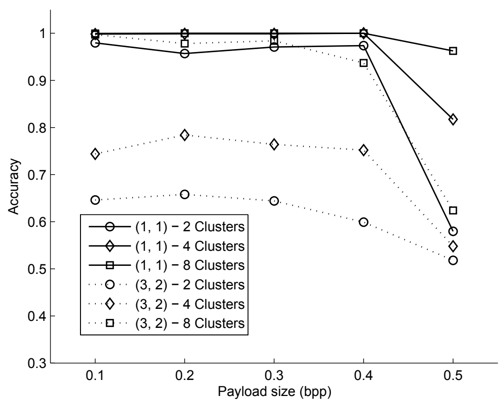
[2] Andrew D. Ker, Locating steganographic payload via WS residuals, Proc. 10th ACM Workshop on Multimedia and Security, pg. 27. (2008).

[3] Andrew D. Ker and Ivans Lubenko, Feature reduction and payload location with WAM steganalysis, Proc. IS&T/SPIE Media Forensics and Security, pg. 72540A. (2009).

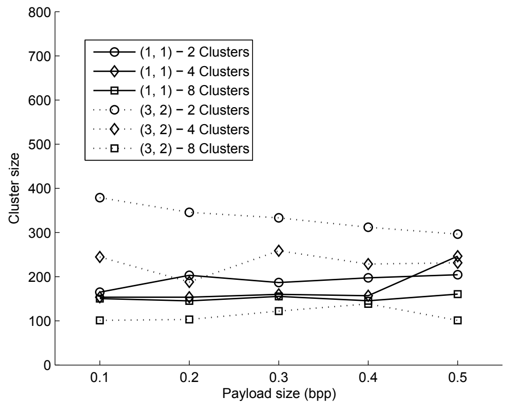
[4] Tu-Thach Quach, On locating steganographic payload using residuals, Proc. IS&T/SPIE Media Watermarking, Security, and Forensics III, pg. 78800J. (2011).

[5] Tu-Thach Quach, Optimal cover estimation methods and stegano-

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-075

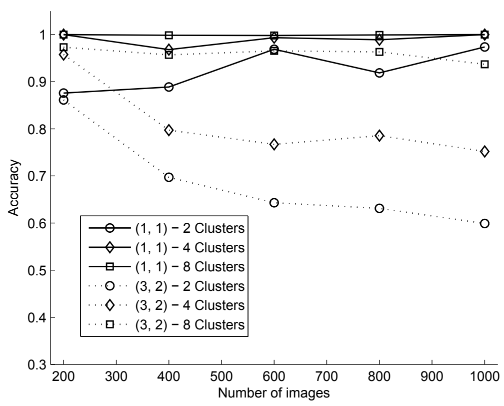


(a)

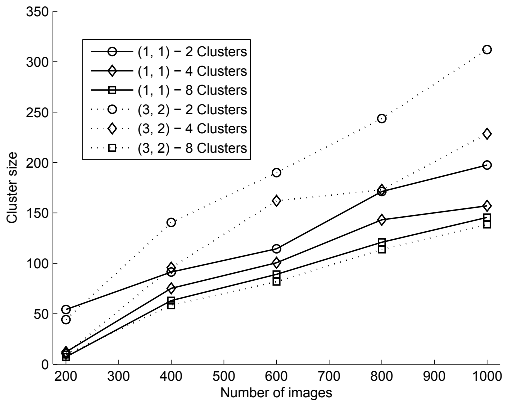


(b)

Figure 3. *Plots of (a) accuracy and (b) cluster size as a function of payload size for two embedding algorithms: (1, 1) and (3, 2) matrix embedding.*

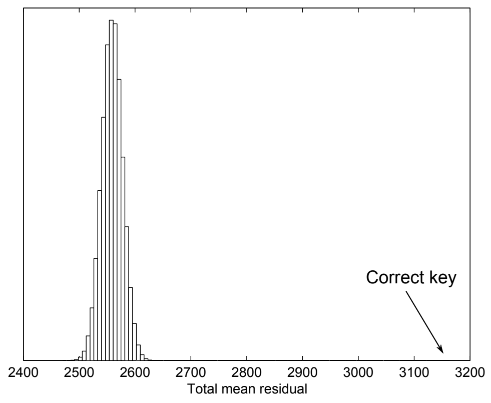


(a)

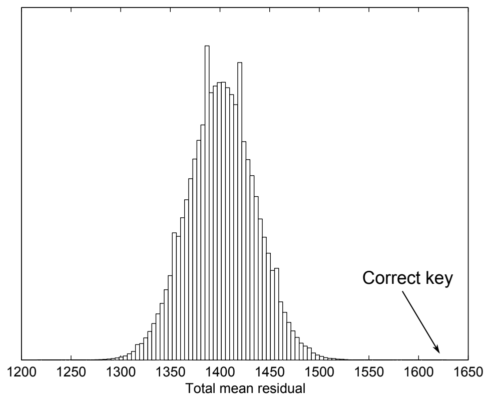


(b)

Figure 4. *Plots of (a) accuracy and (b) cluster size as a function of the number of stego images for two embedding algorithms: (1, 1) and (3, 2) matrix embedding.*



(a)



(b)

Figure 5. *Histograms of the total mean residuals computed from* 10^6 *keys on stego images embedded by (3, 2) matrix embedding with a payload of 0.1 bpp: (a) 5 stego images and (b) a single stego image. The correct key corresponds to the outlier with the largest total mean residuals in both cases.*

DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-075

graphic payload location, IEEE Trans. Inf. Forensics Security, 6, 4 (2011).

[6] Tu-Thach Quach, Locating payload embedded by group-parity steganography, Digital Investigation, 9, 2 (2012).

[7] Tu-Thach Quach, Cover estimation and payload location using Markov random fields, Proc. IS&T/SPIE Media Watermarking, Security, and Forensics 2014, pg. 90280H. (2014).

[8] Tu-Thach Quach, Extracting hidden messages in steganographic images, Proc. Digital Forensic Research Workshop, pg. S40. (2014).

[9] Tu-Thach Quach, Locatability of modified pixels in steganographic images, Proc. IS&T/SPIE Media Watermarking, Security, and Forensics 2012, pg. 83030Q. (2012).

[10] Tomas Filler, Tomas Pevný, and Patrick Bas, Break our steganography system, July 2010. www.agents.cz/boss/.

[11] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss, On spectral clustering: analysis and an algorithm, Proc. Neural Information Processing Systems, pg. 849. (2001).

[12] Jessica Fridrich, Miroslav Goljan, and David Soukal, Searching for the stego-key, Proc. IS&T/SPIE Security, Steganography, and Watermarking of Multimedia Contents VI, pg. 70. (2004).

[13] Jessica Fridrich and Tomas Filler, Practical methods for minimizing embedding impact in steganography, Proc. IS&T/SPIE Security, Steganography, and Watermarking of Multimedia Contents IX, pg. 650502. (2007).

[14] Tomas Pevný and Andrew D. Ker, Steganographic key leakage through payload metadata, Proc. Information Hiding and Multimedia Security, pg. 109. (2014).