Attribute Permutation Steganography Detection using Attribute Position Changes Count

Iman Sedeeq, Frans Coenen and Alexei Lisitsa
Department of Computer Science, University of Liverpool, Liverpool, U.K.
{*iman.sedeeq, coenen, lisitsa*}@*liverpool.ac.uk*

Keywords: Steganography, Attribute Permutation, Classification.

Abstract: An approach to detecting the presence of HTML Attribute Permutation Steganography (APS) is proposed and founded on the idea of using a classification (prediction) model. To this end a position changes count metric, the Attribute Position Changes Count (APCC), is presented with which to capture attribute ordering information. The main advantage offered by the APCC metric, unlike other APS detection metrics, which tend to use average values, is that it captures the full range of attribute position changes. A second advantage is that it can be readily used to define a feature space from which feature vectors can be generated which in turn can be used to generate a steganography classification model. With a combination of three most known attribute permutation steganography algorithms and three well known classifiers APCC showed high performance in each case compared with alternative attribute detection approaches. In terms of AUC metric APCC achieved best eight out of nine cases and in terms of ACC metric APCC produced best seven out of nine cases. The reported evaluation demonstrates that the APCC APS detection can be successfully employed to detect hidden messages embedded in WWW pages using APS, outperforming a number of alternative approaches.

1 INTRODUCTION

Because of the increased prevalence of Internet usage in daily life, and the large volumes of WWW material that is accessed on a daily bases, there is an increased concern regarding the security risk posed by steganography; the practice of hiding data, within some *carrier*, typically for malicious purpose. For steganography to be effective the carrier needs to be a frequently used medium that does not raise suspicion. HTML files provide a perfect carrier because of their popularity (hence they do not raise suspicion) and because the look and feel of the WWW pages represented by HTML files tends to remain unchanged in the presence of steganography.

The most common forms of HTML steganography are: (i) usage of invisible characters, such as blank and tab characters, in some prescribed ordering (Sui and Luo, 2004), (ii) changing the letter case in tags using some predetermined system (Zhao and Lu, 2007) and (iii) attribute permutation whereby the attributes associated with HTML tags are reordered in some prescribed manner for the purpose of messages hiding (Forrest, 2006; Huang et al., 2008; Shen and Zhao, 2010). This last type of HTML steganography, Attribute Permutation, has some attractive features because: (i) its usage does not increase the file size (unlike in the case where embedded invisible characters are used) and (ii) it is not easily noticeable by inspection of the HTML source file (unlike in the case where tag letter case switching is used). Three of the most established algorithms whereby Attribute Permutation Steganography (APS) can be conducted, and those considered in this paper, are: (i) Deogol (Forrest, 2006), (ii) Huang et al. (Huang et al., 2008) and (iii) Shen et al. (Shen and Zhao, 2010).

Most APS detection algorithms to date use a statistical approach to identifying APS, see for example (Polak and Kotulski, 2010) and (Sedeeq et al., 2016) . In (Polak and Kotulski, 2010) a predominant attribute pair ordering statistic is used, while in (Sedeeq et al., 2016) a standard deviation measure is adopted. The central idea presented in this paper is to train a classification model which can then be used to classify WWW pages as being either “stego pages” or “normal pages”. Our approach is inspired by the work presented in (Jian-feng et al., 2014). Classifier generation is well understood (Ayodele, 2010; Smola and Vishwanathan, 2008), the challenge is how to transform the application focused data into a form appropriate for classifier generation. Most generators take as input a feature vector representation of some sort.

Sedeeq, I., Coenen, F. and Lisitsa, A.
Attribute Permutation Steganography Detection using Attribute Position Changes Count.
DOI: 10.5220/0006166400950100
In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pages 95-100
ISBN: 978-989-758-209-7

In the context of APS the requirement is thus to represent attribute data in a feature vector format. In (Jian-feng et al., 2014) an attribute position mean distance and variance statistics were used to populate feature vectors. However, it is argued in this paper that using average positions ignores the spread of position changes, which may in turn be important in the context of APS detection; in other words important information is lost when using average values. Instead, this paper proposes the Attribute Position Changes Count (APCC) mechanism. The idea is motivated by the observation that APS entails frequent attribute position changes which can thus be used to distinguish APS WWW pages from non-APS WWW pages. The APCC metric is fully described, as is its incorporation into the APCC ASP detection approach. The proposed approach was evaluated using three WWW data sets, each seeded with one of the commonly referenced APS approaches listed above (Forrest, 2006; Huang et al., 2008; Shen and Zhao, 2010), and three different classifier generation model paradigms (Neural Networks, SVM and Naive Bayes). In each case the operation of the proposed APCC approach was compared with three alternative algorithms (Polak and Kotulski, 2010; Sedeeq et al., 2016; Jian-feng et al., 2014). In eight of the nine cases the proposed approach produced the best performance.

The remainder of this paper is organized as follows. Section 2 presents some related work relevant to the work presented in this paper. Section 3 presents the Attribute Position Changes Count (APCC) mechanism. The proposed APCC feature vector generation mechanism is then presented in Section 4. Section 5 presents the evaluation of the proposed approach. The paper is completed with some conclusions presented in Section 6.

2 RELATED WORK

The APCC APS detection algorithm proposed in this paper is designed, unlike other proposed stegongraphy detection algorithm (Jian-feng et al., 2014), to detect hidden messages embedded using any of the three most well known APS methods: (i) Deogol as in (Forrest, 2006), (ii) Huang et al. (Huang et al., 2008) and (iii) Shen et al. (Shen and Zhao, 2010). The first two methods operate in a similar manner as follows. The message is transformed to a large number M, then the remainder of $\frac{M}{n!}$, where n is the number of a tag attributes to be utilized, is calculated. This number is then transformed to a permutation, and the current tag replaced with this permutation. The difference between the two methods is in how the desired permutations are generated. The third method adopted a different APS approach. Here a binary relation was constructed, between tag attributes, using a binary string. This binary string was transformed to a permutation and the current tag replaced with this permutation.

Previous work on detection of the presence of messages hiding using HTML APS can be categorized according to whether the systems described are used in a dynamic (monitoring) context or a static (non-monitoring) context. The work presented in this paper falls into the static APS detection category.

In the dynamic context a website is continuously monitored for APS. Here webpage ”snapshots” are recorded using some predefined sampling interval. Two significant examples of this approach can be found in Polak and Kotulski (Polak and Kotulski, 2010) and Sedeeq et al. (Sedeeq et al., 2016). In Polak and Kotulski (Polak and Kotulski, 2010) an APS detection method is presented that assumes the APS is conducted using Deogol (Forrest, 2006). The method is based on the idea of identifying a predominant attribute pair ordering. For each pair of attributes the detection algorithm counts how many times the first attribute appears before the second and vice-versa. The algorithm then calculated a value W, the number of attributes pairs that occur in different orderings divided by the total number of pairings:

$$W = 1 - \frac{\sum_n \sum_i R_{n,i}(a_x, a_y)}{C} \tag{1}$$

Where: (i) n is a tag identifier and i is a sequential occurrences number for the tag; (ii) $R_{n,i}$ then represents the ith occurrence of the tag n with respect to the attribute pair (a_x, a_y) in the predominant order; and (iii) C represents the total number of all occurrences of all attribute pairs in a webpage. W is thus the fraction of attribute pairs that occur in a different order to the predominant order (a_y, a_x). If W is high, this is an indicator of APS.

In Sedeeq et al. (Sedeeq et al., 2016) a statistical approach to dynamic APS detection was proposed whereby the Standard Deviation (SD) of the attribute positions in webpage tags was used. In this case it was assumed that the APS had been conducted using Shen et al. (Shen and Zhao, 2010). For each attribute, all the positions in all tags are collected and then the standard deviation for the webpage is calculated. In (Sedeeq et al., 2016) it was found that a “stego webpage” always features a higher SD than a non-stego WWW page.

The APS detection methods described in Polak and Kotulski and in Sedeeq et al. are both dynamic methods, for the purpose of the evaluation presented later in this paper, these methods were thus adapted so that they can be used in a static context. The evaluation is presented in Section 5 below.

In the static context WWW pages are considered in isolation (no need for a webpage history presented in the terms of a sequence of "snapshots"). An example can be found in Jian-feng et al. (Jian-feng et al., 2014) where an approach is presented, that uses the idea of learning a classification model that can be used to predict the presence or absence of APS. In Jian-feng et al. SVM classification was adopted (Scholkopf and Smola, 2002), but any other form of classification model generator could equally well have been used. Feature vectors were generated using two statistics: (i) the distance between the attribute mean position benchmark and the sample attribute mean position of the suspicious webpage and (ii) the variance of attribute positions; APS was again realised using Deogol (Forrest, 2006). The work presented in this paper, although also a static approach, differs from that presented in (Jian-feng et al., 2014) in that position change counts are used to populate the feature vectors representation to support classifier generation. The approach to ASP detection proposed by Jian-feng et al. (Jian-feng et al., 2014) was used to compare the operation of the APCC approach, this is reported on later in this paper in Section 5. Note that with respect to the Jian-feng et al. approach to APS the three parameters used were: (i) the number of attributes to be considered (BA), (ii) the number of attribute appearances in terms of mean distance (TD) and (iii) the number of attribute appearance in terms of variance (TV). The following parameter settings were recommended: (i) BA = 3, (ii) TD = 3 and (iii) TV = 3. These were therefore adopted with respect to the evaluation presented later in this paper.

3 ATTRIBUTE POSITION CHANGE COUNTING

In normal circumstances (no steganography present) attributes within HTML tags tend to feature a consistent ordering, but once attribute permutation steganography has taken place this consistent order is no longer the case. Each attribute will have a position associated with it within the tag where the first position is given the number 0, the second position 1 and so on. Thus, given an attribute a_i belonging to the set of attributes A featured in a HTML page this will have a set positions $P_{a_i} = \{p_1, p_2, \dots\}$ associated with it. In the absence of APS we would expect the values in P to be more or less constant. Thus by counting the number of position changes we have an indicator of the presence (or not) of APS. Given two consecutive positions, p_j and p_{j+1} we increment the count so far by one if $p_j \neq p_{j+1}$.

Table 1: Attribute position changes before and after APS.

Attribute	Att. pos. before APS	Pos. change count	Att. pos. after APS	Pos. change count
`content`	111111111	0	101001010	7
`rel`	000000000001111	1	101000110104242	11

This is illustrated in Table 1 with respect to a fictitious web site, and considers two attributes: (i) `content`[1] and (ii) `rel`[2]. The second column in Table 1 shows the positions of the attributes in the HTML document that might exist if no APS has taken place. Note that the first attribute features 9 times and the second 15 times (hence 9 and 15 positions respectively). The third column then lists the associated position change counts. The forth column shows the positions of the attributes in the HTML document that might exist if APS has taken place, whilst the fifth column shows the associated position change counts. Note that a clear distinction can be observed before and after APS.

4 THE APCC ATTRIBUTE PERMUTATION STEGANOGRAPHY DETECTION ALGORITHM

Given the above, idea is to use attribute position change information to represent WWW pages and, given an appropriately defined training set featuring both stego and non-stego WWW pages, to train a classification model for steganography detection. There are many algorithms available that can be used to build classification models (also known as prediction models). Common examples include: (i) Neural Networks, (ii) Support Vector Machines and (iii) Naive Bayes. What these algorithms have in common is that they take as input a set of feature vectors of length n generated from an n-dimensional feature space. In our case the feature space represents the set of attributes A that we wish to consider. Each dimension represents an attribute whose values range from 0 to some

[1] The `content` attribute is used with the `meta` HTML tag to provide additional information that can be used by, for example, www browsers.

[2] The `rel` is used with the `a` HTML tag to indicate the relationship between the current document and the linked resource.

maximum number of position change counts.

The required training data comprises an $n \times m$ matrix where n is the number of webpages in the training set and m is the number of attributes to be considered. The value associated with each attribute, or webpage, is the position change count generated as described above. Before presenting the proposed algorithm used for counting the attribute position changes, the following definitions should be noted:

- H is an input webpage.
- S: is a set of tags in H that have two attributes or more, $S = \{T_1, T_2, \dots\}$.
- FV is an output feature vector of attributes position change counts in S
- $attName = \{a_1, a_2, ...a_n\}$ is the complete set of attributes in S.
- $attPos(a_i)$: A set of positions in S for attribute a_i, $attPos(a_i) = \{p_1, p_2, \dots p_m\}$. Each attribute has m occurrences such that $|attPos(a_i)| = m$.
- $apcc(a_i)$: is the number of an attribute (a_i) position changes which is incremented in case of $p_{j+1} \neq p_j$.

The pseudo code for the APCC algorithm is presented in Algorithm 1. The input to the algorithm (line 1) is a WWW page H. The output is FV a feature vector recording the number of attributes position changes with respect to all the tags within S, the set of tags with two attributes or more. The algorithm commences by finding S (line 3) then initializing the sets *attName* and *attPos* to be empty sets (lines 4 and 5). Next (lines 7 to 12) the algorithm loops through each tag in S and each attribute a_i in each tag and records the attribute name and position to record, in *attName*, all attributes names and their associated positions (recorded in the set *attPos*). Next (lines 13 to 22) the algorithm loops through each attribute in *attName* to calculate the attribute position change count *apcc* according to the *attPos* value associated with each. Each calculated *apcc* value is stored in FV to generate the desired feature vector for the webpage.

Note also that for the training data each feature vector has a class label associated with it: class 1 represents a stego webpage while class 2 represents a normal webpage. In real life of course it is this class label we wish to predict.

5 EVALUATION

To evaluate the proposed APCC approach a collection of 74 landing webpages from popular websites were downloaded that covered different domains (education, news, shopping and business). The conjectured advantage of the proposed approach was that it would operate well regardless of how the APS was conducted. To demonstrate this the three most commonly used APS algorithms, identified in the introduction to this paper, were used to generate three evaluation data sets: (i) Deogol, (ii) Huang et al. and (iii) Shen et al. In each case half of the selected WWW pages were seeded using APS. The hidden message was a natural text. For the purpose of incorporating APS five attributes were selected. Consequently our feature vectors have five attributes each holding a position change count. Thus the entire data matrix in each case measures 74×5.

For the evaluation Ten-fold Cross Validation (TCV) was used throughout whereby the input data was divided into 10 folds and the classifier generation process conducted and tested 10 times, each time using a different fold for the testing. The evaluation metrics used were those recommended in (Demsar, 2006), namely:

- The standard accuracy (Acc) measure that refers to the percentage of correctly classified samples.
- The Area Under the receiver operator characteristic Curve (AUC); the area under the Receiver Operator Characteristic (ROC) curve, a graphical

Algorithm 1: Calculate attribute position changes in a webpage.

```
1: Input: $H$
2: Output: FV a feature vector holding position
   changes count for attributes in $H$
3: $S$ = The set of tags with two attributes or more
4: $attName$= {}
5: $attPos$= {}
6: $k$=0
7: for each $tag\ T \in S$ do
8:    for each attribute $a_i \in T$ do
9:       $attName$= $attName$+$a_i$.name
10:      $attPos(a_i)$= $attPos(a_i)$ + $a_i$.position
11:   end for
12: end for
13: for each attribute $a_i \in attName$ do
14:   $apcc = 0$
15:   for $i = 1$ to $i = m$ do
16:      if $attPos(p_{j+1}) \neq attPos(p_j)$ then
17:         $apcc = apcc + 1$
18:      end if
19:      $FV[k] = apcc$
20:      $k = k + 1$
21:   end for
22: end for
```

plot of the true positives rate versus the false positives rate.

The objectives of the evaluation were firstly to analyze the effectiveness of the proposed APCC mechanism in terms of a number of classification model generators and secondly to compare the operation of the proposed approach with existing approaches. Both are considered in further derail in the following two sub-sections.

5.1 Effectiveness of the APCC Algorithm

For the first of the above objectives three classifier generation models were considered, as implemented in Weka machine learning environment (Hall et al., 2009) : (i) Multi-Layer Perceptron (MLP) Neural Network, (ii) SVM and (iii) Naive Bayes. Note that SVM was used because this was the classification model used with respect to the APS detection approach proposed by Jian-feng et al. (Jian-feng et al., 2014). Recall that the significance of the later is that, to the best knowledge of the authors, this is the only other previous work that adopts a classification model approach to APS detection, however using a very different feature vector representation.

The results are summarized in Tables 2 and 3 (best results highlighted in bold font). Table 2 shows the results obtained in terms of average Accuracy (Acc) while Table 3 shows the results obtained in terms of the AUC measure. the averaged SD in each case is in parentheses. From the Tables it can be seen that the proposed APCC feature vector representation can be successfully used to train classifiers to distinguish between normal webpages and stego webpages regardless of the adopted APS algorithm. Although best performance was obtained with respect to Shen et al. APS. From the tables it can also be seen that there is little difference in operation between the selected classifier generators.

Table 2: Average accuracy (Acc) results of APCC (best results highlighted in bold font).

APS Algorithm	MLP	SVM	NB
Deogol (Forrest, 2006)	93.39% (9.71)	90.71% (10.54)	90.89% (10.46)
Haung et al. (Huang et al., 2008)	90.36% (11.17)	93.21% (7.18)	88.75% (13.10)
Shen et al. (Shen and Zhao, 2010)	**93.57%** (10.75)	**96.07%** (6.34)	**93.57%** (10.75)

Table 3: Average of AUC results of APCC (best results highlighted in bold font).

APS Algorithm	MLP	SVM	NB
Deogol (Forrest, 2006)	0.97 (0.06)	0.90 (0.11)	0.98 (0.05)
Haung et al. (Huang et al., 2008)	0.94 (0.11)	0.91 (0.08)	0.96 (0.07)
Shen et al. (Shen and Zhao, 2010)	**0.99** (0.03)	**0.95** (0.06)	**0.99** (0.03)

5.2 Comparison with Other Detection Methods

With respect to the second objective, comparisons were made with the operation of: (i) Polak and Kotulski (Polak and Kotulski, 2010), (ii) Sedeeq et al. (Sedeeq et al., 2016) and (iii) Jian-feng et al. (Jian-feng et al., 2014) (all three were discussed in Section 2 above). In order to acquire a complete picture of APS detection approaches performance in literature, we include dynamic APS detection approaches such as Polak and Kotulski (Polak and Kotulski, 2010) and Sedeeq et al. (Sedeeq et al., 2016) to compare with APCC. These approaches required a webpage history monitoring whereby with static approaches like APCC and that of Jian-feng et al. (Jian-feng et al., 2014) there is no need for a history of a webpage. In the case of Polak and Kotulski (Polak and Kotulski, 2010), a dynamic APS detection approach, the algorithm was adapted so that feature vectors were generated comprised of W values. Each webpage was thus represented by a single statistical feature W calculated as described in Section 2. In the case of Sedeeq et al.(Sedeeq et al., 2016), also a dynamic APS detection approach, the algorithm was adapted so that feature vectors were generated comprised of the standard deviation values. Again, this resulted in each webpage being represented by one feature, the standard deviation of attribute positions as described in Section 2. For Jian-feng et al.(Jian-feng et al., 2014), an alternative static approach to APS detection, the recommended parameter settings were used: (i) BA = 3, (ii) TD = 3 and (iii) TV = 3. The feature vectors were generated in the same manner as described in (Jian-feng et al., 2014).

For the evaluation the three APS algorithms considered previously (Deogol, Huang et al. and Shen et al.) and the three classifier generation paradigms considered previously (Neural Network MLP, SVM and Naive Bayes NB) were used. Thus nine different combinations. The results are presented in Table 4. In the table the shaded rows highlight the results obtained using the APCC algorithm. From the table it can be seen that the APCC mechanism can effectively detect hidden messages regardless of the APS

Table 4: Comparison of APCC results with other detection approaches (best results highlighted in bold font).

Classifier	Detection Approach in APS Algorithm	Deogol (Forrest, 2006)		Huang et al. (Huang et al., 2008)		Shen et al. (Shen and Zhao, 2010)	
		Acc	AUC	Acc	AUC	Acc	AUC
MLP	Polak and Kotulski (Polak and Kotulski, 2010)	74.64%	0.78	47.32%	0.48	44.46%	0.33
	Sedeeq et al.(Sedeeq et al., 2016)	75.36%	0.88	68.10%	0.84	61.37%	0.65
	Jian-feng et al. (Jian-feng et al., 2014)	86.61%	0.93	78.93%	0.91	77.32%	0.88
	APCC	**93.39%**	**0.97**	**90.36%**	**0.94**	**93.57%**	**0.99**
SVM	Polak and Kotulski (Polak and Kotulski, 2010)	61.96%	0.65	48.57%	0.50	47.32%	0.46
	Sedeeq et al. (Sedeeq et al., 2016)	76.79%	0.77	74.70%	0.74	58.10%	0.59
	Jian-feng et al.(Jian-feng et al., 2014)	88.04%	0.88	81.43%	0.82	75.54%	0.75
	APCC	**90.71%**	**0.90**	**93.21%**	**0.91**	**96.07%**	**0.95**
NB	Polak and Kotulski (Polak and Kotulski, 2010)	70.63%	0.75	51.43%	0.47	51.61%	0.53
	Sedeeq et al. (Sedeeq et al., 2016)	78.04%	0.89	74.70%	0.84	61.01%	0.65
	Jian-feng et al.(Jian-feng et al., 2014)	**91.96%**	**0.99**	**89.64%**	0.91	79.82%	0.89
	APCC	90.89%	0.98	88.75%	**0.96**	**93.57%**	**0.99**

algorithm used. Inspection of the table indicates that, with respect to accuracy Acc, the APCC approach produced best results in seven of the nine cases; and, with respect to AUC, the best result in eight of the nine cases. It is interesting to note that the dynamic techniques of Polak and Kotulski, and Sedeeq et al., did not perform well. This is probably because neither technique was well suited to usage in a static context. It is also interesting to note that the technique proposed by Jian-feng et al. worked well when using NB classification (Jian-feng et al, originally used SVM classification to evaluate their approach).

6 CONCLUSION

In this paper a novel approach to detecting HTML Attribute Permutation Steganography (APS) has been suggested. The approach is founded on the usage of a proposed Attribute Position Changes Count (APCC) metric, the main contribution of the paper. This metric offers the dual advantages that: (i) it serves to capture more detail concerning APS than methods that use average statistical values and (ii) it can be readily used to generate feature vectors with which to train an APS classification model. The evaluation was conducted by considering three alternative APS methods and three classifier generation paradigms (thus three-by-three combinations) and in each case comparing the proposed APCC APS detection approach with three alternative APS detection approaches from the literature. In eight out of the nine cases the proposed approach produced the best AUC value, and in seven out of the nine cases the best accuracy ACC value, thus indicating that the proposed approach can be successfully employed to detect attribute permutation steganography. A deeper analysis is required in order to understand cases when Jian-feng et al. (Jian-feng et al., 2014) performs better than APCC. Further work includes development and evaluation of novel metrics and algorithms for HTML steganography detection.

REFERENCES

Ayodele, T. (2010). Types of machine learning algorithms.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Machine Learning Research*, pages 1–30.

Forrest, S. (2006). Introduction to deogol.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and H., I. (2009). The weka data mining software: An update; sigkdd explorations, volume 11, issue 1.

Huang, H., Zhong, S., and Sun, X. (2008). An algorithm of webpage information hiding based on attributes permutation. In *4th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2008)*, pages 257–260.

Jian-feng, W., Liu-sheng, H., Miao-miao, T., Zhi-li, C., and Hai-bo, M. (2014). Detection of html steganography based on statistics and svm. *Journal of Chinese Computer Systems*, 35(6):1221–1225.

Polak, L. and Kotulski, Z. (2010). Sending hidden data through www pages detection and prevention. *Engn.Trans.*, 58:75–89.

Scholkopf, B. and Smola, A. (2002). *Learning with Kernals*. MIT Press.

Sedeeq, I., Coenen, F., and Lisitsa, A. (2016). A statistical approach to the detection of html attribute permutation steganography. *Second ICISSP Italy*.

Shen, D. and Zhao, H. (2010). A novel scheme of webpage information hiding based on attributes. In *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference on*, pages 1147–1150.

Smola, A. and Vishwanathan, S. (2008). *Introduction to Machine Learning*. Cambridge University Press, The Pitt Building, Trumpington Street, Cambridge, United Kingdom.

Sui, X.-G. and Luo, H. (2004). A new steganography method based on hypertext. In *Radio Science Conference, 2004. Proceedings. 2004 Asia-Pacific*, pages 181–184.

Zhao, Q. and Lu, H. (2007). Pca-based web page watermarking. *Pattern Recogn.*, 40(4):1334–1341.