Two Key Estimation Techniques for the Broken Arrows Watermarking Scheme

Patrick Bas
CNRS - Lagis
Ecole Centrale de Lille, Avenue Paul Langevin
BP 48, 59651 Villeneuve d’Ascq, France
CNRS - Gipsa-Lab Grenoble
961 rue de la Houille Blanche
B.P. 46 38402 Saint Martin d’Heres, France
Patrick.Bas@inpg.fr

Andreas Westfeld
HTW Dresden
University of Applied Sciences
01008 DRESDEN, PF 120701, Germany
andreas.westfeld@htw-dresden.de

ABSTRACT
This paper presents two different key estimation attacks targeted for the image watermarking system proposed for the BOWS-2 contest. Ten thousand images are used in order to estimate the secret key and remove the watermark while minimizing the distortion. Two different techniques with distinct strategies are proposed. The first one combines a regression-based denoising process to filter out the component of the original images and a clustering algorithm to compute the different components of the key. The second attack is based on an inline subspace estimation algorithm, which estimates the subspace associated with the secret key without computing eigen decomposition. The key components are then estimated using Independent Component Analysis and a strategy designed to leave efficiently the detection region is presented. On six test images, the two attacks are able to remove the mark with very small distortions (between 41.8 dB and 49 dB).

1. INTRODUCTION
If watermarking robustness deals with the performance of a watermarking scheme against common processing operations (re-compression, transcoding, editing operations), the security of a watermarking scheme is addressed whenever an adversary is part of the game and tries to remove the watermark. There are two important families of security attacks:

- Sensitivity attacks [1] aim at removing the watermark by using the watermark detector as an oracle,
- Information leakage attacks [2] aim at estimating the secret key analysing contents watermarked with the same key.

In order to assess the security of a robust watermarking scheme against information leakage attacks, the third episode of the BOWS-2 contest [3] was run during 3 months (the first two episodes were focused on robustness and sensitivity attacks). During the third episode, the adversary had access to the description of the embedding and detection watermarking schemes; this is compliant with the Kerckhoffs’ principle [4] used in cryptanalysis. Moreover, 10,000 images watermarked with the same secret key were also available to the adversary and his ultimate goal was consequently to analyse these images in order remove the watermark while minimizing the PSNR between the three marked original and attacked images. In such a framework performing a security attack enables to estimate the secret carriers and to weaken the watermark up to the point where it becomes no longer detectable.

Classical information leakage attacks encompass key estimation using blind source separation schemes such as Principal Component Analysis [5], Independent Component Analysis [2] and clustering schemes such as set-membership approaches [6] or K-Means [7].

This paper presents and compares two attacks that have been used against the watermarking scheme called Broken Arrows [8] (BA) used during BOWS-2. The first one, designed by Andreas Westfeld, was the most efficient one during the contest, and relies on a denoising step inspired from [9], a clustering step and an estimation step. The second attack has been designed by Patrick Bas later on and relies on the global estimation of the secret subspace using an inline PCA algorithm.
The paper is organised as follows: the next section presents a description of the main features of the Broken Arrows technique. The third section describes a first attack mixing denoising and clustering. The fourth section presents an alternative method to perform the attack by estimating the secret subspace using inline subspace tracking and estimate the secret key using independent component analysis. Finally, the results of the two attacks are presented and compared in Section 5.

2. BROKEN ARROWS IN A NUTSHELL

The whole diagram of the embedding scheme is depicted in Figure 1 and for an extended description of the watermarking system, the reader is invited to read [8]. The BA watermarking scheme first performs a wavelet decomposition of the image $I_X$ and it watermarks all the components but the low frequency ones. For a $512 	imes 512$ grey-level image, $N_s = 258,048$ wavelet coefficients of 9 subbands are arranged in a vector $s_X$ to be watermarked.

The security of the system relies on a secret projection on $N_c = 256$ pseudo-orthogonal vectors generated using a pseudo-random number generator seeded using the key. The embedding is performed in this secret subspace by using both informed coding and informed embedding [10]. Informed coding is used by selection of the one vector that is closer to the host vector from a set of $N_c = 30$ pseudo-orthogonal vectors out of 256. This way the embedding distortion is minimised. Informed embedding is performed by pushing the host content as far as possible from the border of the detection region and looks at maximizing the robustness. Once the watermark vector $s_W$ is generated, two embeddings are possible: a constant embedding which does not consider psychovisual requirements corresponding to

$$ s_Y(i) = s_X(i) + s_W(i), $$

and a proportional embedding that acts as a psychovisual mask:

$$ s_Y(i) = s_X(i) + |s_X(i)|s_W(i), $$

where $s_X(i)$ and $s_Y(i)$ denote respectively the original and the watermarked wavelet coefficients. BA uses proportional embedding.

2.1 Security analysis of the system

In the end, most of the information about the secrecy of BA relies in the set of $N_c$ vectors $c_i$ of size $N_c$. The $N_c - N_s$ other vectors are used during the embedding, but their contributions are very small and as we will see in Section 5, powerful removal attacks can already be devised by estimating only the subspace of size $N_c$.

The two algorithms that are presented below aim at estimating the $N_c$ vectors $c_i$ from the set of watermarked images, but use different strategies.

The first approach can be considered as a bottom-up approach: (1) the watermarked images are processed in order to remove as much as possible the signal coming from the host images, (2) the processed signals are clustered into $N_c$ clusters and (3) estimates of $c_i$ are finally computed.

The second approach is a top-down approach: (1) the secret subspace is estimated from the pool of observations and (2) the secret vectors are estimated from the projections of the observation in the subspace.

3. A CLUSTERING APPROACH BASED ON DENOISING

3.1 The denoising process

The key idea behind this section is to extract the watermark from $s_Y$. We cannot do so without the key. Our approach will only approximate the watermark, which is easily done in two steps if we already have a sufficiently well performing attack to the watermark:

1. deploy an already existing attack to estimate the unmarked original $s_X$,
2. estimate the watermark $s_W$:

$$ \hat{s}_W = s_Y - \hat{s}_X. $$

If this “extraction” of the watermark is followed by a clustering process and a key estimation (cf. Sect. 3.2 and 3.3), we can remove the watermark with a performance that is increased by about 20 dB PSNR compared to the already existing attack [9], which was developed during the first episode of BOWS-2.

There are several kinds of noise that we should distinguish:

1. the watermark $s_W$, which is a random vector that is independent of the image content,
2. the image noise that is not independent of the local surrounding in the image, and
3. the estimation error that is added by the denoising process described in this section.

These three kinds of noise may have similar spectral properties. In contrast to the usual meaning of the word “denoising”—the reduction of random visual image artefacts—this denoising process will not reduce any visible noise and might even increase such artefacts. So this procedure is rather a de-watermarking process than a denoising. In the figurative sense it is comparable to the self similarities attack [11]. Parts from the image are restored from the surrounding. Because locally close values in images strongly depend on each other, but the elements of the watermark do not, the image can be preserved by estimation from the surrounding while the watermark is completely removed (cf. Figure 2). We use simple linear regression to predict the absolute value of $N_c$ wavelet coefficients $s_Y(j)$ from $k$ “neighbouring” coefficients $s_Y(i_1),\ldots,s_Y(i_k)$:

$$ |s_Y(j)| = \beta_1|s_Y(i_1)| + \cdots + \beta_k|s_Y(i_k)| + \epsilon_j, $$

where $\beta_1\ldots\beta_k$ are unknown parameters and $\epsilon_j$ is an error term. The number of terms $k$ depends on the decomposition level that the coefficient belongs to (cf. Table 1). The regression model collects the local dependencies between the

<table>
<thead>
<tr>
<th>Table 1: Number of terms $k$ for prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
</tr>
<tr>
<td>in</td>
</tr>
<tr>
<td>level 0</td>
</tr>
<tr>
<td>level 1</td>
</tr>
<tr>
<td>level 2</td>
</tr>
</tbody>
</table>
its superior counterparts from the first and second level decomposition (4 and 16 per subband, respectively).

One of the key properties of this denoising process is its non-interactivity. The attacked images are produced without submitting trials to the detector. All computations can be done locally on the attacker’s side.

3.2 The clustering process

In this section we will cluster the images into $N_c = 30$ bins, depending on the version $v = 1 \ldots N_v$ of the watermark $s_W(v)$ that has been selected during the informed coding stage. BA defines a numbering of the bins, because the versions of the watermark are consecutive chunks of $N_c$ bits from the pseudorandom number generator that was seeded with the secret key. However, since the clustering works without this key, the numbering of the bins is determined by the clustering and might be different. The version $v$ that is used depends on the feature vector $s_X$ to be marked:

$$v = \arg\max_{i=1 \ldots N_c} |\text{cor}(s_{W(i)}, s_X)|.$$ 

Since the image content in the marked feature vector $s_Y$ is much stronger than the embedded watermark $s_W$ (PSNR of the watermark is about 43 dB), it is impossible to correctly decide whether two images $I_1$ and $I_2$ that are marked using the same key belong to the same or different bins, based on the (Pearson) correlation of their feature vectors $s_{Y1}$ and $s_{Y2}$ alone. However, the chances are higher, if we can take an estimate of the embedded watermark(s) instead and decide based on their correlation. The difference between the marked original $s_Y$ and the dewatermarked image from the denoising process $\hat{s}_X$ forms such an approximation $\hat{s}_W$ (cf. Eq. 3).

We can pick one image of the BOWS-2 database $\mathcal{D}$ with the approximated watermark $\hat{s}_W$ and determine the absolute correlation $c$ with all $\hat{s}_W(j)$ ($j = \{1 \ldots |\mathcal{D}|\}$):

$$c = |\text{cor}(s_Y(j), \hat{s}_W(j))|.$$ 

We picked the “Sheep” image, which is one of the three to be attacked during Episode 3. Let’s call this image the leader of bin 1. The clustering started with $i = 3661$, which is the index of Sheep in the BOWS-2 database. We expect a “strong” absolute correlation ($c \approx 0.01$) if two images belong to the same bin and a weaker ($c \approx 0.002$) if they don’t.

We tried to define the membership of a bin by $c$, which exceeds a certain threshold. This first approach did not work very well, because $c$ of the bin members and its leader ranges from 0.03 almost to 0 (cf. Figure 3).
A better approach, which we finally used for clustering, works “by exclusion.” The idea is to select the image with the smallest $c$ to lead the next bin (cf. Algorithm 1). The more leaders are selected (with growing $k$ in the algorithm), the clearer the bins are clustered. Step 5 updates the current leader in each bin by new leader, that might have a stronger discriminating power. We consolidated the BOWS-2 database by removing all clones. (We replaced the files 6533.pgm, 7263.pgm, 7265.pgm, 7602.pgm, and 7856.pgm by 9998.pgm, 9999.pgm, 10000.pgm, 0.pgm, and .pgm [sic]; sheep.pgm was inserted as the missing 3661.pgm, so $\mathcal{D}$ contains 1.pgm ... 9997.pgm.) This algorithm makes some assumptions. Step 3 assumes that $\ell_{k+1}$ belongs to a new bin. This is sometimes not the case. At the end of this algorithm one bin was split, i.e., we had two bins with about 170 members and about 340 in all others. (We did not suppose a biased database and expected $|\mathcal{D}|/N_c \approx 333$ images in each bin.) So we continued the algorithm for $k = 30$ and $k = 31$, removed one leader of the split bin that is revealed by its unexpectedly low number of members, and finally rerun the loop in Steps 4...6 of Algorithm 1 for all 30 bins, yielding all 30 bin leaders $\ell_1 \ldots \ell_{30}$. Based on these bin leaders we define an operator $\text{bin}(i)$ that maps an image with index $i$ in the database to the index of its bin:

$$\text{bin}(i) := \arg \max_{j=1 \ldots N_c} \left| \text{cor}(\hat{s}_i W(\ell_j), \hat{s}_i W(i)) \right|.$$

A posteriori we tested that the clustering defined by $\text{bin}(i)$ is correct: no image was assigned to the wrong bin.

### 3.3 The key estimation and removal process

The key estimation process for a particular image $I_k \in \mathcal{D}$ combines all estimated watermarks belonging to bin($k$) in order to find an improved estimate $\hat{s}_W^*(k)$. The pairwise correlation of two members in the same bin can have a positive or negative sign. The element-wise sum of all estimated watermarks in the bin will be neutral if we do not watch the sign of their correlation:

$$I_k := \{i | I_k \in \mathcal{D}, \text{bin}(i) = \text{bin}(k)\},$$

$$\hat{s}_W(k) = \sum_{i \in I_k} \text{sign}(\text{cor}(\hat{s}_i W(\ell), \hat{s}_i W(k))) \cdot \hat{s}_i W(i).$$

Finally we remove the watermark from the feature space by subtracting a scaled version of the PN sequence sign($\hat{s}_W^*(k)$):

$$s_X(k) = s_Y(k) - \gamma \cdot \text{sign}(\hat{s}_W^*(k)).$$

“sign” returns the element-wise sign of the vector. The scalar value $\gamma$ is optimised to produce the unmarked image that is closest to the detection boundary.

### 4. A SUBSPACE ESTIMATION APPROACH

In this section, we propose an approach based on a partial estimation of the secret projection used by the embedding algorithm (see Figure 1). Our rationale relies on the fact that the embedding increases considerably the variance of the contents within the secret subspace, in particular along the axes of the $N_c$ vectors $c_i$ that are used during the embedding. To illustrate this phenomenon, Figure 4 depicts a comparison between the histograms of the maximum absolute correlations between the $c_i$ and original or watermarked contents on 10,000 images of the BOWS-2 challenge (embedding distortion of about 43 dB). This shows clearly an important increases of the variance within the secret subspace; consequently the strategy that is developed in this section is to estimate the subspace spanned by the vectors $\{c_i\}$ by estimating the components of important variances from the observations.

![Figure 3: Distribution of the absolute correlation between the approximated watermarks of the BOWS-2 database and the image “Sheep”](image)

![Figure 4: Distribution of the maximum of the 30 correlations (in absolute value) for 10,000 images (PSNR=43 dB), proportional embedding](image)
Contrary to the systems addressed in [5, 2, 12], the proposed method used a secret subspace of large dimension (30) in order to avoid basis estimation techniques such as averaging,

- The dimension of the host signal itself is very important (258,048),
- The system is used in real-life conditions on 10,000 images on a watermarking scheme that fulfills the different constraints regarding robustness but also visual distortion.

In order to perform subspace estimation, one usually uses Principal Component Analysis (PCA) which can be performed by an Eigen Decomposition (ED) of the covariance matrix obtained using the different observations. In our practical context however, the ED is difficult to perform because of the following computational considerations:

- The covariance matrix if of size \(N_s \times N_s\), which means that 248 gigabytes are required if each element of the matrix is stored as a float,
- The computation of the covariance matrix requires around \(O(N_s^2) \approx 10^{12}\) flops,
- The computational cost of the ED is \(O(N_s^3) \approx 10^{15}\) flops.

Consequently, we have looked for another way to compute the principal components of the space of watermarked contents. One interesting option is to use an inline algorithm which compute the principal vectors without computing any \(N_s \times N_s\) matrices.

4.1 The OPAST algorithm

The OPAST algorithm [13] (Orthogonal Projection Approximation Subspace Tracking) is a fast and efficient iterative algorithm that uses observations as inputs to extract the \(N_p\) principal components (e.g. the component associated with the \(N_p\) more important eigenvalues). The goal of the algorithm is to find the projection matrix \(W\) in order to minimize the projection error \(J(W) = E(||r - WW^t r||^2)\) on the estimated subspace for the set of observations \(\{r_i\}\). This algorithm can be decomposed into eight steps summed-up in Algorithm 2.

The notations are the following: the projection matrix \(W_0\) is \(N_s \times N_p\) and is initiated randomly, the parameter \(\alpha \in [0, 1]\) is a forgetting factor, \(y, q\) are \(N_p\) long vectors, \(r, p\) and \(p'\) are \(N_s\) long vectors, \(W\) is a \(N_s \times N_p\) matrix, \(Z\) is a \(N_p \times N_p\) matrix.

Step 6 is an iterative approximation of \(\alpha\) the covariance matrix for the \(N_p\) principal dimensions. Steps 7 and 8 are the translations of the orthogonalisation process.

Since the complexity of OPAST is only \(4N_sN_p + O(N_p^2) \approx 10^7\) flops per iteration, the use of the OPAST algorithm is possible in our context. Furthermore, it is easy to use and only relies on the parameter \(\alpha\) for the approximation of the pseudo covariance matrix and does not suffer from instability.

4.2 Estimation assessment

In order to run experiments and to assess the behaviour of the subspace estimation algorithm we used the Square Chordal Distance (SCD) to compute a distance between two subspaces (the one coming from the secret key and the estimated subspace). The use of chordal distance for watermarking security analysis was first proposed by Pérez-Freire et al. [14] and is convenient because the SCD = 0 if the estimated subspaces are equal and SCD = \(N_p\) if they are orthogonal.

Given \(C\), a matrix with each column equal to one \(c_i\), the computation of the SCD is defined by the principal angles \([\theta_1 ... \theta_{N_p}]\) (the minimal angles between two orthogonal bases [15]) that are singular values of \(C^tW\) (note that this matrix is only \(N_c \times N_p\):

\[
SCD = \sum_{i=1}^{N_c} \sin^2(\theta_i) \tag{12}
\]

A geometric illustration of the principal angles is depicted on Figure 5.

![Figure 5: Principal angles between 2 planes \(\pi_1\) and \(\pi_2\)](image)

4.3 OPAST applied on Broken Arrows

We present here the different issues that we have encountered and are specific to the studied embedding algorithm: the impact of the weighting method, the influence of the host signal and the possibility to use several times the same observations to refine the estimation of the secret subspace.

4.3.1 Constant vs Proportional embedding

We have first compared the impact of the embeddings given by the constant embedding (Eq. 1) and proportional embedding (Eq. 2). The behaviour of the OPAST algorithm is radically different for these two strategies since the estimated subspace is very close to the secret subspace for constant embedding and nearly orthogonal to it for proportional
embedding. The evolution of the SCD in both cases is depicted on Figure 6.

Such a problematic behaviour can be explained by the fact that the variance of the contents in the secret subspace is more important using constant embedding than using proportional embedding (compare Figure 6 of [8] with Figure 4). The second explanation is the fact that the proportional embedding acts as a weighting mask which is different for each observation. This makes the principal directions less obvious to find since the added watermark is no more collinear to one secret projection.

One solution to address this issue is to try to decrease the effect of the proportional weighting and to reduce the variance of the host signal. This can be done by feeding the OPAST algorithm with a calibrated observation $\hat{s}_Y$ where each sample is normalised by a prediction of the weighting factor $|s_X(i)|$ according to the neighbourhood $N$ of $N$ samples:

$$\hat{s}_Y(i) = s_Y(i)/\hat{s}_X(i),$$

where

$$|\hat{s}_X(i)| = \frac{1}{N} \sum_{j \in N} |s_Y(i)|.$$  \hfill (13)

The result of the calibration process on the estimation of the secret subspace is depicted on Figure 6 using a $5 \times 5$ neighbourhood for each subband. With calibration, the SCD decreases with the number of observations.

![Figure 6: SCD for different embeddings and calibrations (PSNR=43 dB, $N_p = 36$)](image)

### 4.3.2 Principal components induced by the subbands

Whenever watermarking is performed on non-iid signals like natural images, the key estimation process can face issues regarding interferences from the host-signals [16]. Figure 7 depicts the cosine of the principal angles for $N_s = 36$ and $N_p = 30$ out of 30 basis vectors of the subspace were accurately estimated.

Consequently, depending on the embedding distortion, one might choose $N_p > N_s$.

### 4.3.3 Multiple runs

In order to improve the estimation of the subspace, another option is to use the contents several times and consequently improve the estimation of the pseudo-covariance matrix in the OPAST algorithm. Figure 8 shows the evolution of the SCD after three multiple runs. We can notice that if the SCD decreases significantly between $10^4$ and $2 \times 10^4$ observations, the gain for using a third run is poor though.

![Figure 7: $\cos \theta$ for $N_p = 36$ and $N_p = 30$, Embedding PSNR = 43 dB](image)

![Figure 8: Evolution of SCD after 3 runs (30,000 observations, PSNR = 43 dB, $N_p = 36$)](image)

### 4.4 Cone estimation using ICA

The last step of the key estimation process is to estimate each $c_i$ by $\hat{c}_i$. Since all the variances along the different cone axes are equal, one solution to estimate the direction of each axis is to look for independent directions using Independent Component Analysis (ICA). This strategy has already been used in watermarking security by former key estimation techniques and more information on the usage of ICA in this context can be found in [12].

### 4.5 Leaving the detection region

The last step is to modify the watermarked content in order to push it outside the detection region of the hypercone of normalised axis $\hat{c}_k$ which is selected such that:

$$|s_Y \hat{c}_k| \geq \gamma s_Y |\hat{c}_i|.$$  \hfill (15)

Theoretically this is possible by cancelling the projection between $\hat{c}_i$ and $s_Y$ to create the attacked vector $s_Z$:

$$s_Z = s_Y - \gamma s_Y \hat{c}_k \hat{c}_k.$$  \hfill (15)

However, practically $\hat{c}_k$ may not be accurate enough to be sure that $s_Z \hat{c}_k = 0$, especially if the coordinates of the watermarked content are close to the cone axis. On Figure
It is possible to filter most of the image components respectively equal to 41.83 dB and 48.96 dB. Watermarked and attacked images for Sheep and Casimir are orthogonal to the cone axis. This second strategy (called Strat. 2) is depicted on Figure 9 and the PSNR between the orthogonal to the cone axis. This second strategy (called Strat. 2) is depicted on Figure 9 and the PSNR between the orthogonal to the cone axis. This second strategy (called Strat. 2) is depicted on Figure 9 and the PSNR between the watermarked and attacked images for Sheep and Casimir are respectively equal to 41.83 dB and 48.96 dB.

\[ s_Z = s_Y - \gamma s_Y \hat{c}_i \hat{c}_k + \sum_{j \neq k} (\beta s_Y \hat{c}_i - 1) \hat{c}_j. \] (16)

\[ \gamma \text{ and } \beta \text{ are constant factors specifying the amount of energy put in the directions which are respectively collinear and orthogonal to the cone axis.} \]

**5. RESULTS**

**5.1 Attacks after the first approach**

Building upon the watermark estimates from a regression-based approach, the clustering perfectly separates all images of the BOWS-2 database in 30 bins defined by the version of the watermark that has been selected by the informed coding step. Within these bins the watermark is simply determined by element-wise averaging of the watermark estimates, but two cases will be considered: positive and negative correlation with the watermark to be removed. The element-wise sign of the averaged watermark forms a PN sequence that is used to eliminate the watermark in the image under attack. Here the detector is needed only a small number of times to find the optimal scale \( \gamma \) of the PN sequence to just remove the watermark with the highest PSNR (cf. Table 2).

It takes about a minute to find a watermark estimate using the regression-based approach. So for 10,000 images it may easily take a week on a single computer. We assigned this task to a PC farm that returned the result in minutes. The clustering took about 24 hours on a single computer, the key estimation took about one minute per key (only three for the three given images are needed, but all 30 could be estimated).

**5.2 Attacks after the second approach**

Using the attack based on subspace estimation, the subspace is estimated on the 10,000 images provided by BOWS-2 contest. Each image is watermarked with a PSNR between 42.5 dB and 43 dB. As for Episode 3, proportional embedding is used.

OPAST is run using calibration on a 5 × 5 neighbourhood for each subband, \( N_x = 36 \) (see 4.3.2), and 2.10^4 observations (e.g. two runs, see 4.3.3), and the forgetting factor \( \alpha \) is set to 1. The run of OPAST (two runs) takes approximately 6 hours on a 3 GHz Intel Xeon.

The ICA step was performed using fastICA [17, 18], with a symmetric strategy and the tanh function to estimate negentropy. All the other parameters are set to defaults values.

Watermark removal (see 4.5) uses normalised estimated vectors \( \hat{c}_i \) orientated such that \( s_Y \hat{c}_i > 0 \). The second strategy is used and the parameters are set to \( \gamma = 1.1 + 0.1i \) (where \( i \) is a number of iterations) and \( \beta = 50 \).

The attack was performed on the five images used during the contest and available on the BOWS-2 website. Figure 9 shows the effects of the attacks in the MCB plan for “Casimir” and “Sheep”.

Table 3 presents the PSNRs after the attack and the number of necessary iterations. The coordinate of the original images in the MCB plane are also presented. As can be seen, the distortion is between 41.8 dB and 49 dB, which yields very small or imperceptible artefacts. Since the norm of the attacking depends of \( s_Y \hat{c}_i \), the farther the images are from the detection boundary, the more important the attacking distortion is.

**6. CONCLUSIONS AND PERSPECTIVES**

We point out the weaknesses of a very robust watermarking scheme in terms of security. Theses weaknesses comes from the facts that:

1. It is possible to filter most of the image components using regression-based denoising and consequently to increase the watermark to content ratio,

2. AMD Athlon 64 Processor 3200+ at 2.2 GHz

**Table 2: Final PSNR for the three images under attack in Episode 3 (\( \gamma \) represents the scale of the PN sequence, cf. Eq. 11)**

<table>
<thead>
<tr>
<th>Image</th>
<th>PSNR</th>
<th>( \gamma )</th>
<th>MCB coord. after attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sheep</td>
<td>45.58 dB</td>
<td>1.360</td>
<td>(70.0, 196.5)</td>
</tr>
<tr>
<td>Bear</td>
<td>46.64 dB</td>
<td>1.202</td>
<td>(17.8, 50.9)</td>
</tr>
<tr>
<td>Horses</td>
<td>46.48 dB</td>
<td>1.226</td>
<td>(35.2, 99.7)</td>
</tr>
</tbody>
</table>

**Figure 9: Effects of the different strategies on the MCB plan for Casimir and Sheep**

**Table 3: PSNR after successful attack using subspace estimation (i represents the number of iteration necessary to obtain a successful attack)**

<table>
<thead>
<tr>
<th>Image</th>
<th>PSNR</th>
<th>i</th>
<th>MCB coord. after attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sheep</td>
<td>41.83 dB</td>
<td>1</td>
<td>(925,48) (62,223)</td>
</tr>
<tr>
<td>Bear</td>
<td>44.21 dB</td>
<td>0</td>
<td>(532,47) (88,253)</td>
</tr>
<tr>
<td>Horses</td>
<td>41.80 dB</td>
<td>0</td>
<td>(915,20) (77,233)</td>
</tr>
<tr>
<td>Louvre</td>
<td>48.95 dB</td>
<td>0</td>
<td>(321,194) (96,317)</td>
</tr>
<tr>
<td>Fall</td>
<td>46.76 dB</td>
<td>0</td>
<td>(553,250) (116,370)</td>
</tr>
<tr>
<td>Casimir</td>
<td>48.96 dB</td>
<td>0</td>
<td>(352,31) (59,234)</td>
</tr>
</tbody>
</table>
2. The embedding increases significantly the variance of the data in the secret subspace and subspace estimation techniques can consequently be used,

3. The number of hypercones $N_c$ used to create the detection region is rather small, which makes the estimation easier.

The future directions will consequently try to address these different issues in order to increase the security of the analysed algorithm. However, one has also to consider the inevitable trade-off between robustness and security.

7. ACKNOWLEDGEMENTS

Patrick Bas is supported by the National French projects Nebbiano ANR-06-SETIN-009, ANR-05-RIAM-O1903 Estivale, and ANR-ARA TSAR.

8. REFERENCES


