Hindawi Publishing Corporation
EURASIP Journal on Information Security
Volume 2007, Article ID 64521, 10 pages
doi:10.1155/2007/64521

Research Article

A Workbench for the BOWS Contest

Andreas Westfeld

Instititute for System Architecture, Department of Computer Science, Technische Universität Dresden, 01062 Dresden, Germany

Correspondence should be addressed to Andreas Westfeld, westfeld@inf.tu-dresden.de

Received 8 May 2007; Revised 24 August 2007; Accepted 22 October 2007

Recommended by A. Piva

The first break our watermarking system (BOWS) contest challenged researchers to remove the watermark from three given images. Participants could submit altered versions of the images to an online detector. For a successful attack, the watermark had to be unreadable to this detector with a quality above 30 dB peak signal-to-noise ratio. We implemented our experiments in R, a language for statistical computing. This paper presents the BOWS package, an extension for R, along with examples for using this experimental environment. The BOWS package provides an offline detector for several platforms. Furthermore, the particular watermarking algorithm used in the contest is analysed. We show how to find single coefficient attacks and derive high-quality images (62.6 dB PSNR) with full knowledge of the key.

1. INTRODUCTION

The first break our watermarking system (BOWS) contest challenged researchers to remove the invisible watermark from three given greyscale images (of a strawberry, a landscape, and a church). The Watermarking Virtual Lab of the European Network of Excellence ECRYPT announced this contest in the middle of December 2005 [1]. Participants could submit altered versions of the images to an online detector through a web interface [2]. For a successful attack, the watermark had to be unreadable to the BOWS detector with a quality above 30 dB peak signal-to-noise ratio (PSNR).

PSNR is a simple and widely used quality metric based on the mean squared error (MSE), which is computed by averaging the squared intensity differences between the distorted image a with $w \times h$ pixels and its reference b:

$$\mathrm{PSNR}(a,b) = 10 \cdot \lg\left(\frac{255^2 \cdot w \cdot h}{\sum_{x=1}^{w}\sum_{y=1}^{h}\left(a_{x,y} - b_{x,y}\right)^2}\right). \quad (1)$$

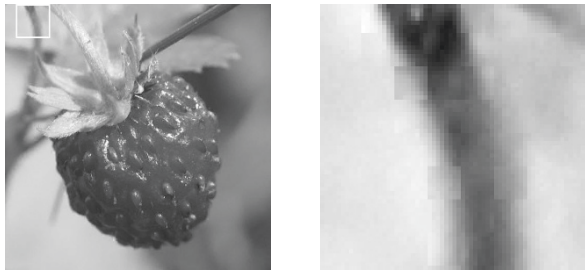
The PSNR measure suggests that geometric attacks are ruled out or are at least at a disadvantage. We were somewhat surprised when we noticed that in particular the geometric attacks yielded the higher PSNR values compared to pixel-oriented attacks after just a few detector requests. Section 2 confirms this with a couple of simple examples. Section 3 presents the sensitivity attack, which was used during the second phase of the contest. In Section 4, we introduce a matrix notation for the decoding procedure, which is used for the description of the single coefficient attack (cf. Section 5) and single step attack (cf. Section 6). Section 7 summarises the results and concludes the paper.

2. GETTING STARTED

2.1. Domain of the watermark

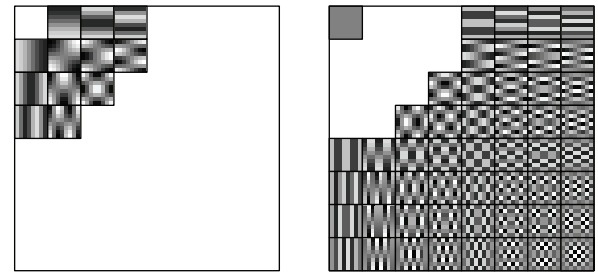
A careful examination of the images revealed that there exist several squares of 8×8 pixels with decreased or increased contrast (cf. Figure 1). Apart from these visible artefacts, the 8×8 block discrete cosine transform (DCT) is widely used in watermarking. Experiments have shown that the watermark resides in 12 low-frequency subbands of the DCT domain. These subbands have been determined by global replacement with zeros: there is no individual subband that can remove the watermark; however, we discovered 12 DCT subbands (cf. Figure 2(a)) that affect the watermark when nullified in pairs, triples, and so forth. (There are single coefficients that remove the watermark (cf. Section 5). This is no contradiction. A single coefficient attack saturates one particular coefficient there, while all coefficients of whole subbands are set to zero here.)

Knowing the responsible subbands, we can create a bandpass filter that is virtually invariant to the watermark.



(a) (b)

Figure 1: Visible 8×8 block distortions. Original image (a) and detail of it (b).



(a) (b)

Figure 2: DCT subbands used for the watermark (a) and subbands the watermark is invariant to (b).

The filtered image combines the 12 low-frequency subbands from the attacked image (cf. Figure 2(a)) with the 52 other subbands from the marked original (cf. Figure 2(b)). In earlier workshop contributions [3, 4], we reported that, for many simple attacks, the PSNR will be increased to just above the required 30 dB by such a filter.

2.2. Additive white Gaussian noise attack

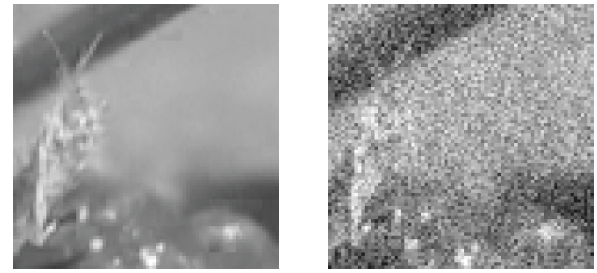
The watermarking system used in the BOWS contest is very robust against additive noise. Figure 3(b) shows annoying distortions in the strawberry. The watermark is still detected with only 19.69 dB PSNR.

To mount this attack, we need one normal distributed pseudorandom value for each pixel. For reproducible results, we set the seed for the random generator using set.seed. We use the R function rnorm to instantiate 262 144 standard normal random numbers and scale it by 26.73 to just remove the watermark.[1]

```
> require(bows)
> set.seed(123)
> noisevector <- rnorm(length(strawberry))∗26.73
> noisyimage <- as.gscale(noisevector + strawberry)
> bows(noisyimage, 1)
      The watermark has been removed.
      The PSNR of the image after your attack is 19.69 dB.
```

The function as.gscale clips values below 0 and above 255 and rounds all pixels to integer values as required by the detector function BOWS. The detector function has one more parameter (besides the image to test), which determines the reference image for the PSNR calculation (1 for strawberry, 2 for woodpath, and 3 for the church).

To determine the scale σ at the detection boundary, we implement a detector function addwgn, based on bows(), which takes σ as a parameter and returns a positive value if the watermark has been removed, otherwise a negative value. This is easily achieved by subtracting 0.5 from the logical value, since TRUE $\equiv$ 1 and FALSE $\equiv$ 0. The root of this function is the detection boundary. We use the standard R function uniroot to find the corresponding σ (sigma).



(a) (b)

Figure 3: The bug on the strawberry before and after adding noise. The distorted image still contains the watermark.

```
addwgn<-function(sigma, noisevector, image,
                                        imagenumber){
    noisyimage <- as.gscale(sigma∗noisevector +
                                              image)
    res <- bows(noisyimage, imagenumber)
    res$removed − 0.5
}
set.seed(123)
init.cache()
uniroot(addwgn, interval=c(0,100),
    noisevector=rnorm(length(strawberry)),
    image=strawberry, imagenumber=1)$root
```

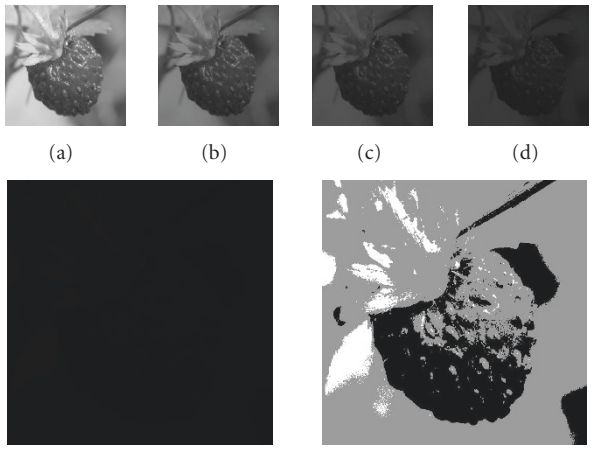
The bandpass filter mentioned earlier can be applied to the best image in the cache to improve the PSNR:

```
> bows(bandpass(strawberry, get.cimg(strawberry)), 1)
      The watermark has been removed.
      The PSNR of the image after your attack is 27.01 dB.
```

2.3. Valumetric attack

The watermarking system used in the BOWS contest is also very robust against valumetric attacks. Figure 4 shows a black image that still contains the watermark with only 6.09 dB PSNR.

[1] We repeated the experiment about 20 000 times with other instances of the noise yielding an average scale of 28.56.



(a) (b) (c) (d)

(e) (f)

Figure 4: The watermark is very robust against valumetric scaling (a)–(d). The contrast can be reduced to 0.8% (e) while the watermark is still readable. There are only 3 levels of grey in the image (85 times amplified, (f)).

To determine the contrast level at the detection boundary, we write again a detector function that takes the parameter of interest and has its root at the detection boundary.

```
contrast <- function(level, image, imagenumber)
      bows(as.gscale(level∗image), imagenumber
                                    )$removed − 0.5
init.cache()
uniroot(contrast, interval=c(0,1),
      image=strawberry, imagenumber=1)$root
```

Even with the bandpass, the required quality level is missed:

```
> bows(bandpass(strawberry, get.cimg(strawberry)), 1)
      The watermark has been removed.
      The PSNR of the image after your attack is 26.69 dB.
```

2.4. Cropping attack

In an earlier report [3], we filled margins of the images with grey to demonstrate that the watermark has different power in different regions. If a small stripe on the right in the church image (65 pixels wide) is replaced by neutral grey, the watermark is not detected anymore. Together with the bandpass filter, this is a successful attack that achieves a PSNR of 30.2 dB.

2.5. Rotation attack

We found that geometric attacks are more successful in preserving high PSNR. This sounds odd since operations like rotation and translation decrease the PSNR very fast. But the watermarking method used here seems to be even more fragile against such operations. Rotation is successful for the strawberry image after bandpass filtering:

(a) (b)

Figure 5: Fill a margin (width = 65 pixels) with neutral grey to remove the watermark. This is a successful attack after bandpass filtering (30.2 dB).

(a) (b)

Figure 6: Principle of counter-rotating zones with zone radius 100 pixels (a); (b): rotation angle 0.3° for inner zone, and −0.2° for marginal zone, bandpass filter applied (successful with 30.97 dB PSNR).

```
init.cache()
frotate <- function(angle, image, imagenumber) {
      rotimage <- as.gscale(rotate(image, angle))
      bows(rotimage, imagenumber)$removed − 0.5
}
alpha <- uniroot(frotate, interval=c(0,2),
      image=strawberry, imagenumber=1)$root
rotstraw <- get.cimg(strawberry)
bows(bandpass(strawberry, rotstraw), 1)
      The watermark has been removed.
      The PSNR of the image after your attack is 31.74 dB.
```

Figure 6 shows a successful attack for the church image with two counter-rotating zones.

2.6. Limited translation attack

It was rather easy to find a suitable attack for the strawberry and the church image. The woodpath image was more resistant against geometric attacks because its high frequencies are stronger. The quality penalty based on PSNR rises with the square of the introduced error. We decided to revert pixels to their original value if the error exceeds a given limit. Figure 7 shows two versions of the woodpath image that are translated by 50 pixels to the right. In the version on the right, all pixels that would be changed by more than 80 levels of

(a) (b)

Figure 7: Translation of woodpath ((a), shifted by 50 pixels), limited translation of woodpath ((b), limiting difference: 80 levels of grey).

grey have been replaced by their original value to take care of the PSNR.

For successful attacks, the image is shifted by 2 or 3 pixels. The limit is between 33 and 42.

```
> transstraw <- translate.with.limit(strawberry, 42, 3)
> bows(bandpass(strawberry, transstraw), 1)
      The watermark has been removed.
      The PSNR of the image after your attack is 32.46 dB.
> transwood <- translate.with.limit(woodpath, 33, 2)
> bows(bandpass(woodpath, transwood), 2)
      The watermark has been removed.
      The PSNR of the image after your attack is 31.99 dB.
> transchurch <- translate.with.limit(church, 38, 3)
> bows(bandpass(church, transchurch), 3)
      The watermark has been removed.
      The PSNR of the image after your attack is 31.45 dB.
```

3. SENSITIVITY ATTACK

The sensitivity attack can estimate the watermark of correlation-based systems with linear complexity. It was introduced by Cox and Linnartz in 1997 [5]. The work by Comesaña et al. [6] generalised it to watermarking schemes that are not correlation-based.

The sensitivity attack is usually applied in the spatial domain and does not require any prior knowledge. However, since we already knew the domain of the watermark and the subbands used for it, we decided to apply it directly to the DCT coefficients. The sensitivity attack is not restricted to a particular domain. When applied to the DCT coefficients, the attack would also work if the watermark resided in the wavelet or spatial domain. One reason for this choice was to eliminate unnecessary oracle calls for coefficients that we know to be insensitive.

The starting point for the sensitivity attack is a boundary image: a random image close to the detection boundary in which the watermark is still detected. Some minor modifications to this image cause the detector to respond "removed" while the watermark is still detected after other modifications. We construct the boundary image as follows. First, we order all coefficients from the 12 subbands that affect the watermark (v contains these 12 subband indices) by decreasing magnitude, because large coefficients can be altered by a large amount without saturation.

Table 1: Parameters for boundary images.

Image	n	Consolidated set	PSNR	p
Strawberry	3	{3}	39.339 dB	1990
Woodpath	18	{1, 10, 14}	36.113 dB	1338
Church	23	{5, 7, 23}	34.948 dB	1756

```
> dimage <- dct(strawberry)
> oimage <- order(abs(dimage[v,]), decreasing=T)
```

Second, we collect some large coefficients (3 are enough for the strawberry image) and invert their sign until the watermark is removed.

```
> selection <- 1 : 3
> k <- dimage[v,][oimage[selection]]
> dimage[v,][oimage[selection]] <- k − 2000∗sign(k)
> bows(as.gscale(idct(dimage)), 1)
      The watermark has been removed.
      The PSNR of the image after your attack is 34.5 dB.
```

Third, we consolidate, that is, we remove all coefficients from the selection that are not necessary to remove the watermark.

```
> dimage <- dct(strawberry)
> selection <- 3
> k <- dimage[v,][oimage[selection]]
> dimage[v,][oimage[selection]] <- k − 2000∗sign(k)
> bows(as.gscale(idct(dimage)), 1)
      The watermark has been removed.
      The PSNR of the image after your attack is 39.31 dB.
```

Finally, we decrease the change p of the coefficients that still remain in the selection to a level at which the watermark is just detected. In our example, p has to be reduced from initially 2000 to 1990:

```
> dimage <- dct(strawberry)
> selection <- 3
> k <- dimage[v,][oimage[selection]]
> dimage[v,][oimage[selection]] <- k − 1990∗sign(k)
> bows(as.gscale(idct(dimage)), 1)
      The watermark is still there.
      The PSNR of the image after your attack is 39.32 dB.
```

The reader may follow the implementation in the supplementary file sensitivity.R to create a boundary image:

(1) get.boundary.n(⟨image number⟩) is used to determine the initial number n of large coefficients;

(2) consolidate.changes(⟨image number⟩, $1 : n$) to remove coefficients that do not contribute (return a consolidated selection);

(3) find.boundary.psnr(⟨selection⟩, ⟨image number⟩)

to determine the PSNR of the boundary image as a hurdle for additional contributing coefficients.

The resulting images near the detection boundary with their initial selection of one to three coefficients are listed in Table 1. Now we can try individual coefficients in combination with this initial consolidated set to test their sensitivity. The sensitivity of the coefficients depends on the initial set. The initial selection determines the state of the trellis that is attacked. We can distinguish the following three cases.

(1) Most changes to coefficients do not change the detector result neither in positive nor in negative direction, mostly because they do not collaborate with the initial selection of coefficients. For these coefficients, we remember a "sensitivity" of 0.[2]
(2) Some coefficients contribute to the removal of the watermark in both directions, positive and negative. The reason for this is that there is more than one possibility to flip the bit that is encoded in a particular step. This is a property of dirty paper codes; several codewords encode the same message. The encoder will choose the most appropriate one for the host image [7]. It is not possible to combine the coefficients that fall in this second case to achieve an attacked image with better quality. Consequently, these coefficients have also a "sensitivity" of 0.
(3) Rarely, we encounter coefficients that remove the watermark only in positive direction ("sensitivity" 1), or only in negative direction ("sensitivity" −1).

If we finally checked all coefficients that affect the watermark, we determined a vector $\vec{s}$ of sensitivity sings for the coefficients. With this vector, we can create an attacked image $\vec{a}$ from the marked original $\vec{m}$ according to the following equation:

$$\vec{a} = \vec{m} + p\vec{s}. \tag{2}$$

The parameter p is chosen to be as small as possible using interval bisection so that the watermark is just removed. Because of a daily limit of 3000 oracle calls per IP address, we can only test 1500 coefficients in 24 hours. However, these 3000 calls could be finished in about two hours. So, we include some "duty cycles" to reduce the server load. Instead of already twiddling thumbs after these two hours, we rather optimise p for each individual test. Let T be our initial set of coefficients yielding a boundary image with 44.4 dB PSNR. When the same p is applied to T and different coefficients under test, the PSNR will fluctuate (cf. Figure 8). To guarantee a gain in PSNR when the watermark is removed in the test, we would have to decrease p by a security margin. However, this security margin could mask the sensitivity of coefficients that needed a larger parameter $p' > p$ to remove the watermark but still increase the PSNR in that case. For instance, if coefficient 41 is tested, the attacked image with fixed p will score the maximum PSNR 45.0 dB. If the watermark is still there, how can we know that it is not removed for a larger parameter p' that causes 44.8 dB? This would be a contribution of 0.4 dB compared to the boundary image. But when this parameter p' is used for testing coefficient 2, the PSNR is certainly lower than the 44.4 dB for the boundary image. With p', we would find the watermark removed for many coefficients that do not contribute to the removal. For this reason we adjust p individually to have an attacked image with a quality that is just not poorer than the quality of the boundary image.

[2] Actually the sensitivity might be nonzero. We remember only the signs of the sensitivity or, like in this case, 0 if the coefficient is considered not usable.

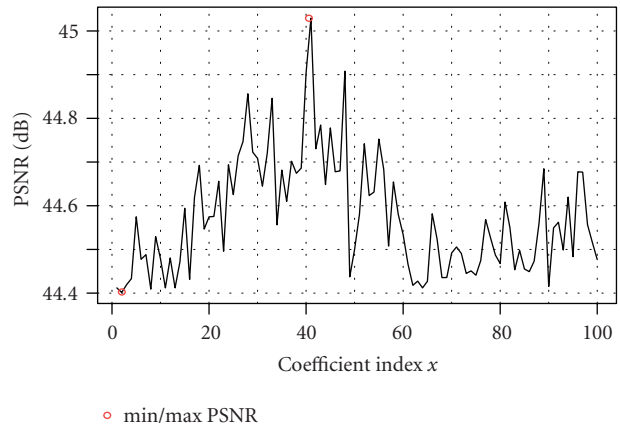


Figure 8: Different PSNR for fixed parameter p when testing the sensitivity of the first 100 coefficients in woodpath.

Table 2: Time needed for particular PSNR levels.

PSNR	# calls	BOWS server	Local detector	# coefficients
40 dB	343	0.6 hours	0.2 hours	5
45 dB	1426	3 hours	0.8 hours	14
50 dB	5124	1.2 days	2.8 hours	40
55 dB	21165	7 days	12 hours	111
57 dB	33619	11 days	19 hours	178
59 dB	62743	21 days	35 hours	293
60 dB	81051	27 days	45 hours	365

Table 2 lists the amount of time that is necessary to gain a particular PSNR level for the strawberry image. While the 50 dB level can be reached in less than two days, it takes another 4 weeks to increase the PSNR by 10 dB again. However, because of the daily limit of 3000 oracle calls, we can already knock off work after 6 hours.

4. DECODING

In the rest of the paper, we assume that the attacker completely knows the decoder. This section describes the decoding procedure of the dirty paper trellis watermark decoder. It covers four main steps as follows.

4.1. Extraction and preprocessing of coefficients

First, the input image is transformed using the 8×8 block DCT into the frequency domain, yielding 64 coefficients for each of the 4096 tiles. As mentioned earlier, only 12 low-frequency coefficients from every tile are regarded by the

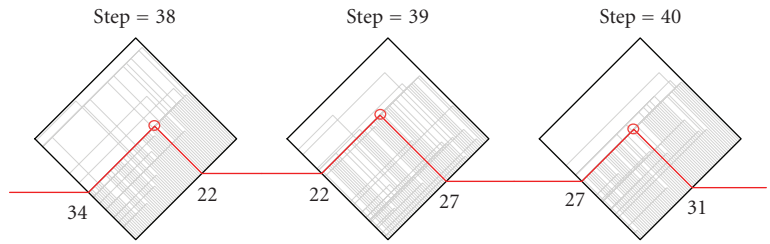


Figure 9: Best path through the trellis of woodpath (last three steps).

Table 3: Number of inputs assigned to the outputs in step 39 of woodpath's Viterbi decoding.

Output	22	52	20	47	50	62	7	10	19	24	40	42	53	56	57	60
Number of inputs	38	8	2	2	2	2	1	1	1	1	1	1	1	1	1	1

watermarking detector (cf. Figure 2). The decoding algorithm starts with $12 \cdot 4096 = 49152$ coefficients that are used for the watermark. These are shuffled by a key-driven permutation.

Comesaña and Pérez-González describe an approach to find the key by brute force in their SPIE'07 paper [8]. All three images contain the same payload, "BOWS*" (40 bits), which is embedded with the same key. We can check this by submitting the strawberry as woodpath (image 2), and the woodpath as church (image 3):

```
> init.cache()
> bows(strawberry, 2)
      The watermark is still there.
      The PSNR of the image after your attack is 10.8 dB.
> bows(woodpath, 3)
      The watermark is still there.
      The PSNR of the image after your attack is 9.84 dB.
```

The correct key extracts the same payload from all three images. It can be identified using this property.

4.2. Correlation

The shuffled coefficients are reshaped as a 40×1228 input matrix I.[3] Also driven by the same key, a 1228×4096 pattern matrix P is filled with 5 029 888 standard normal distributed pseudorandom values. Both are combined to the normalised correlation matrix

$$C = \frac{1}{1228} \cdot I \times P. \tag{3}$$

4.3. Processing of the correlation result

The resulting 40×4096 matrix C is partitioned into 40 matrices $A_1 \cdots A_{40}$ with 64×64 elements each. We yield A_i by filling a 64×64 matrix row by row with the 4096 elements of row i from C.

The modified trellis that is used in BOWS [7] is a graph with nodes (called states) and edges (called arcs). Every 64 nodes belong to a set, while the arcs belong to the 40 steps between these sets. Each arc in step k from state i to state j is assigned element $A_k(i, j)$ of the step matrix, called the arc metric. A path is an ordered set of 40 arcs which takes the 40 steps through the trellis. The path metric is defined as the sum of the arc metrics for the arcs that make up the path.

The Viterbi algorithm gives us an iterative procedure for finding the path with the largest path metric. We are interested in the closest path that decodes differently. Therefore, we also store intermediate results in the matrices $S_1, \ldots, S_{40}$. Let $S_1 = A_1$ (first state metrics are 0). We get S_k for $k > 1$ by adding the *column* maxima of S_{k-1} (64 state metrics before step k) to the *rows* of A_k (e.g., we add the maximum of column 3 from S_{k-1} to all elements of row 3 in A_k).

4.4. Decoding of the payload

Figure 9 shows the last three matrices $S_{38}, \ldots, S_{40}$ that are produced in the 40 steps of the iteration. Imagine the square matrices S_k as "connection centre" linking 64 inputs (columns of the matrix, states after step k) with outputs (rows, states before step k). The inputs are deflected at the maximum of their respective column (state metric). In the figure, grey lines connect the inputs with their actual output. The matrices are rotated counter-clockwise by 45° so that horizontal connection lines can be used between them. The decoding is again an iterative process like the encoding. It starts at step 40 with the global maximum of S_{40}, which is in row 27 and column 31. The one-bit-arc labels determine the message bits. The trellis is fully connected. The 4096 arc labels in each step are systematically chosen to be the sum of row and column index modulo 2. For example, bit 40 of the payload is decoded as the LSB of the sum of row and column number (27 + 31 is even, so the bit is 0).

The output 27 determines the input for the next step. In S_{39}, not all elements but only column 27 is decisive. The maximum is at row 22. We yield 1, since 22 + 27 is odd. We continue in step 38 at input 22. When we look at the grey lines in

[3] 1228 is yielded by dividing the number of coefficients in the 12 subbands by the number of payload bits ($12 \cdot 4096 : 40 = 1228$). The remaining 32 coefficients are ignored and not used for decoding.

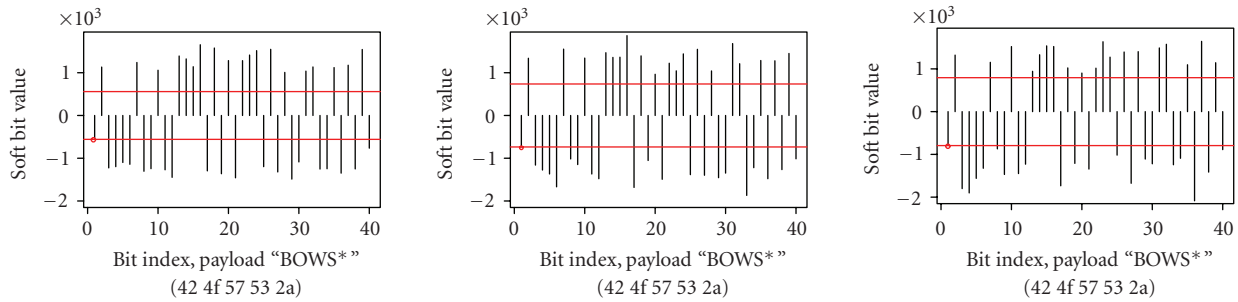


Figure 10: Soft bit values for strawberry (a), woodpath (b), and church (c). The horizontal lines show the overall absolute minimum bit.

Figure 9 at step 39, it is apparent that most of the inputs end up in row 22. Even if a bit error in the previous step leads to another input, the decoding leads with high probability to the correct output (cf. Table 3). That is because the maximum state metric before step k was added to row 22.

To decode the whole payload, the iteration is continued until step 1 is finished.

5. SINGLE COEFFICIENT ATTACK

To render the watermark unreadable to the BOWS oracle, it is sufficient to flip one bit. In this section, we will determine the most suitable coefficient to remove the watermark. By chance, each of the 40 steps in the decoding chain could be the weakest because random numbers are incorporated in the correlation. The influence of a single coefficient on the watermark depends on 4096 standard normal distributed pseudorandom values from the pattern P. Apart from this probabilistic view, there are also structural reasons that the two matrices in steps 1 and 40 are less robust than the rest. Matrix $S_1 = A_1$ is the only one that directly results from the correlation without any row lifted (all initial state metrics are 0). This makes it easier to create a new maximum in its input column. Also matrix S_{40} offers a larger range to find appropriate coefficients because there is no predetermined input column. The global maximum of S_{40} determines both, output and "input." For all other steps k, the input is determined by the output of step $k+1$, that is, the maximum of only 64 values decides which output is yielded. We can choose from 2048 elements that are distributed across S_{40} like the squares of one colour on a chessboard. However, compared with S_1, we have about the same choice: in step 1, there is a predetermined input column but no lifted row, while there is a lifted row in step 40 but no predetermined input column. It is hard to tell which one will be easier to alter.

There are three alternatives to determine "softbits," that is, the resistance of individual bits of the payload to alteration: (1) the resistance to single coefficient attacks; (2) the resistance to single step attacks involving only coefficients of one particular row of I (cf. Section 4); and (3) involving all coefficients without any restriction. The first one is the easiest case, which is shown in Figure 10. The resistance is defined by the current bit and the absolute value $|p|$ by which the most suitable coefficient needed to be altered. The saturation is neglected, that is, an increased alteration could be necessary to equalise the impact of clipping the pixel values to the range $0, \ldots, 255$. The bows package provides a function to determine the resistance of all coefficients. If we add the determined resistance to the corresponding coefficient, the payload is changed. In the following example, we attack the weakest coefficient of step 1 in the strawberry image.

```
> resistance <- determine.resistance(strawberry)
> index <- resistance[[1]]$index[1]
> d <- dct(strawberry)
> d[v,][index] <- d[v,][index] + resistance[[1]]$value[1]
> singlestraw <- idct(d)
> dptdecode(singlestraw) # error in first payload bit
"ÂOWS*"
```

Let i be the input column index of S_k determined by S_{k+1}. Let o be the current output row index, and o' the envisaged row index that leads to an altered bit. Then $S_k(o,i)$ is the current maximum in column i, and $S_k(o',i)$ the current value for the envisaged one. Let $(j,k) = \pi(\ell)$ be the permutation used for shuffling the DCT coefficients in I, and P the pattern matrix (cf. Section 4). We determine the resistance $r_{\min}^{(k)}$ (the necessary alteration to flip the bit in step k) together with the corresponding coefficient index $\ell_{\min}$ as follows:

$$r_{\min}^{(k)} = \min_j \frac{S_k(o,i) - S_k(o',i)}{P(j, 64\cdot(o'-1)+i)},$$
$$\ell_{\min} = \pi^{-1}\left(\arg\min_j \frac{S_k(o,i) - S_k(o',i)}{P(j, 64\cdot(o'-1)+i)}, k\right). \tag{4}$$

The most suitable coefficient in the strawberry image preserves 3.9 dB more than the watermark itself (cf. Figure 11).

6. SINGLE STEP ATTACK

Let $r_j^{(k)}$ be resistance for coefficient $j = 1 \cdots 1228$ in step k:

$$r_j^{(k)} = \frac{S_k(o,i) - S_k(o',i)}{P(j, 64\cdot(o'-1)+i)}. \tag{5}$$

If their absolute values are arranged in ascending order of magnitude,

$$\left|r_{(1)}^{(k)}\right| \le \left|r_{(2)}^{(k)}\right| \le \left|r_{(3)}^{(k)}\right| \le \cdots \le \left|r_{(1228)}^{(k)}\right|, \tag{6}$$

Figure 11: Successful attack with one particular coefficient in the strawberry image (visible at the right edge of the fruit, 46.04 dB PSNR).

$r_{(n)}^{(k)}$ is called the resistance of coefficient with rank n (=resistance[[k]]$value[$n$]) and corresponding index $\ell_n = \pi^{-1}((n), k)$ (=resistance[[k]]$index[$n$]).

The optimal single step attack changes the first coefficients with index $\ell_1, \ldots, \ell_N$. The attacked image $\vec{a}$ is derived from the marked original $\vec{m}$ with minimal parameter p that just removes the watermark:

$$a_{\ell_n} = m_{\ell_n} + p \cdot \operatorname{sign}\left(r_{(n)}^{(k)}\right), \quad n = 1 \cdots N. \tag{7}$$

The ragged shape of the PSNR curve (cf. Figure 12, top) prevents fast optimisation of N. We did an exhaustive search to find the maximum quality.

The function fsstep takes the image, its number, the resistance structure, the parameters p and N, and optionally step k (default $k = 1$). Before we knew the PSNR curve, we used golden section search to find the maximum quality for each step. This yielded 57 dB ($k = 13$) to 62.52 dB ($k = 1$) for the strawberry, 55.51 dB ($k = 33$) to 62.79 dB ($k = 1$) for the woodpath, and 56.13 dB ($k = 23$) to 61.49 dB ($k = 40$) for the church. The second best result for the church image was achieved in step 20, the third best in step 1. Comparing steps 40 and 1 by exhaustive search turned out that step 1 is superior also for the third image.

```
fsstep <- function(image, nr, resistance, p, N, step=1) {
    dimage <- dct(image)
    index <- resistance[[step]]$index[1:N]
    sign <- sign(resistance[[step]]$value)[1:N]
    dimage[v,][index] <- dimage[v,][index] + p * sign
    bows(as.gscale(idct(dimage)), nr)
}
> resistance <- determine.resistance(strawberry)
> fsstep(strawberry, 1, resistance, 4.5, 302)
```

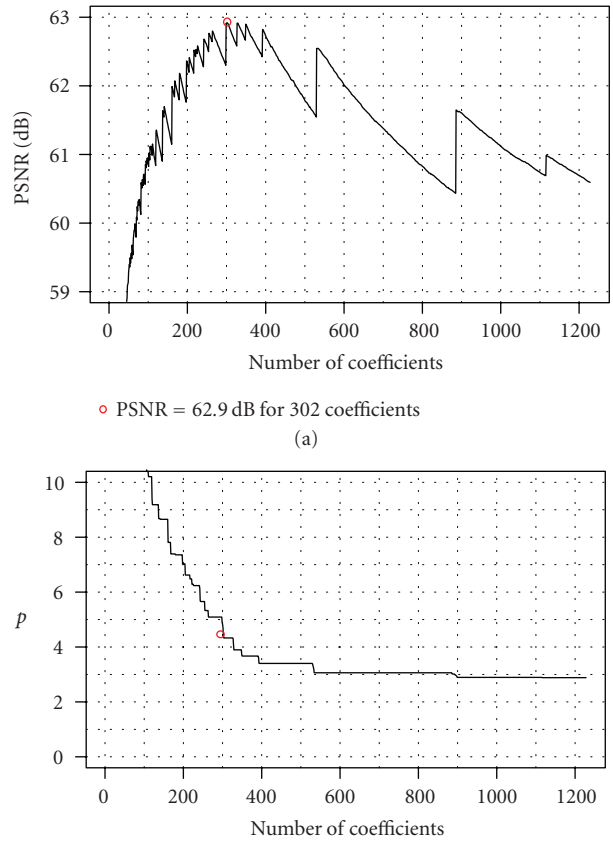


Figure 12: PSNR and p for attacking bit 1 (decoding step 1) with $1, \ldots, 1228$ most effective coefficients.

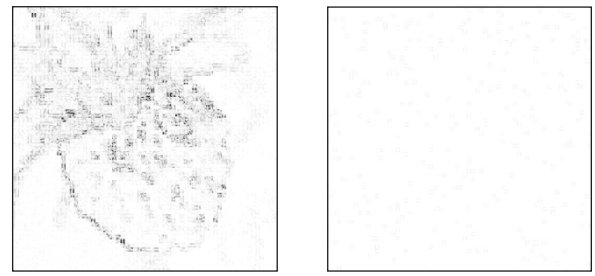


Figure 13: Absolute difference between pristine and marked original image (a, 42.14 dB), and between attacked and marked original image (b, 62.90 dB). Differences are 15 times amplified.

```
    The watermark has been removed.
    PSNR of the image after your attack is 62.9 dB.
> resistance <- determine.resistance(woodpath)
> fsstep(woodpath, 2, resistance, 4.02, 309)
    The watermark has been removed.
    The PSNR of the image after your attack is 63.1 dB.
> resistance <- determine.resistance(church)
> fsstep(church, 3, resistance, 4.94, 374)
    The watermark has been removed.
    The PSNR of the image after your attack is 61.91 dB.
```

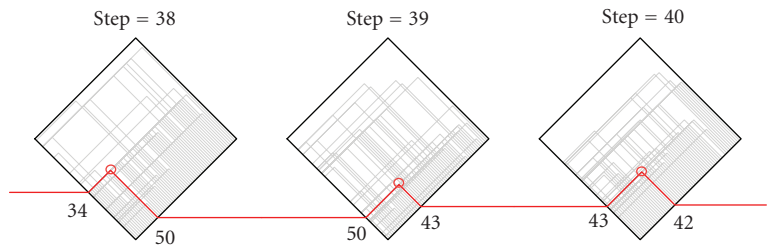


Figure 14: Changed path through the trellis of the attacked woodpath (last three steps).

Table 4: Summary of results.

	Strawberry	Woodpath	Church
Sensitivity attack	60.74 dB	57.05 dB	57.29 dB
Matrices affected	A_{19} A_{20}	A_{38} A_{39} A_{20}	A_{26} A_{27} A_{28}
Number of coefficients	217 207	98 97 84	58 64 63
Attacked payload	"BOgS*"	"BOWS+"	"BOWc*"
	424f67532a	424f57532b	424f57632a
Single-step attack (A_1)	62.90 dB	63.10 dB	61.91 dB
Number of coefficients	302	309	374
Parameter p	4.5	4.02	4.94
Single-coefficient attack	46.04 dB	44.88 dB	43.57 dB
Parameter p	757.7	809.5	1139.6
PSNR of watermark	42.14 dB	46.20 dB	44.38 dB
Worst PSNR	3.05 dB	2.72 dB	3.54 dB
Worst watermarked	3.14 dB	2.92 dB	3.76 dB

Figure 13 shows the energy (15 times amplified absolute differences) of the watermark and the single step attack for the strawberry image. The changes made by the attack are invisible, ranging from −2 to 2 levels of grey. The PSNR of the attack is 20.76 dB higher than for the watermark.

7. SUMMARY AND CONCLUSION

Table 4 presents the results for all three images, in the first rows for the sensitivity attack. We see the number of sensitive coefficients and the matrix to which they belong. Interestingly, the sensitivity attack has chosen coefficients of consecutive matrices although the key was not used. Thus the effect of shuffling in the watermarking process slows down the estimation, but it cannot prevent us from gaining the knowledge that coefficients belong to the same bit. Figure 14 illustrates the case of the attacked woodpath. Although the maxima of the last three steps are changed, only step 40 decodes to another bit. However, the changes in the other two matrices help to lift a different row in step 40. On one hand, the quality of the images is less reduced by the attack than by the watermark embedding itself. On the other hand, 95% of the attacked payloads are still intact.

The quality is slightly increased (by 4.83 dB on average) for the single step attacks that are possible with the knowledge of the key. Maybe another slight increase of quality is possible using multistep attacks that combine the most powerful coefficients of previous steps to affect the lifting as well. The single coefficient attacks yielded approximately the same PSNR as the watermark. For the strawberry, it is even exceeded.

It is hard to derive universal lessons from one particular contest. Craver et al. contributed an impressive list of advices to watermark designers [9]. Their main point is to avoid super-robustness attacks, that is, severe changes that will not affect the watermark. Such attacks are a powerful tool to identify the feature space of the watermark. However, knowledge of the watermarking algorithm is only of little help for watermark removal [10]. In addition, it is still unclear whether compliance with the advices regarding super-robustness will not enable further attacks that obtain comparable or even better levels of PSNR. Since it is possible to embed the same watermark in different images with the same key, it is difficult to develop a detector that inhibits severe false positives.

ACKNOWLEDGMENTS

The work on this paper was supported in part by the Air Force Office of Scientific Research under the research Grant no. FA8655-06-1-3046. The US government is authorised 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 the Air Force Office of Scientific Research, or the US government.

REFERENCES

[1] ECRYPT, "Network of excellence in cryptology," http://www.ecrypt.eu.org, 2006.

[2] ECRYPT, "BOWS, break our watermarking system," http://lci.det.unifi.it/BOWS, 2006.

[3] A. Westfeld, "Lessons from the BOWS Contest," in *Proceedings of the Multimedia and Security Workshop 2006 (MM and Sec'06)*, vol. 2006, pp. 208–213, Geneva, Switzerland, September 2006.

[4] A. Westfeld, "Tackling BOWS with the sensitivity attack," in *Proceedings of Security, Steganography and Watermarking of Multimedia Contents IX*, E. J. Delp III and P. W. Won, Eds., vol. 6505 of *Proceedings of SPIE*, San Jose, Calif, USA, January-February 2007.

[5] I. J. Cox and J.-P. M. G. Linnartz, "Public watermarks and resistance to tampering," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '97)*, vol. 3, pp. 3–6, Santa Barbara, Calif, USA, January 1997.

[6] P. Comesaña, L. Pérez-Freire, and F. Pérez-González, "The blind newton sensitivity attack," in *Proceedings of The International Society for Optical Engineering*, vol. 6072 of *Proceedings of SPIE*, pp. 149–160, 2006.

[7] M. L. Miller, G. J. Doërr, and I. J. Cox, "Applying informed coding and embedding to design a robust high-capacity watermark," *IEEE Transactions on Image Processing*, vol. 13, no. 6, pp. 792–807, 2004.

[8] P. Comesaña and F. Pérez-González, "Two different approaches for attacking BOWS," in *Proceedings of The International Society for Optical Engineering*, E. J. Delp III and P. W. Wong, Eds., vol. 6505 of *Proceedings of SPIE*, San Jose, Calif, USA, January 2007.

[9] S. Craver, I. Atakli, and J. Yu, "How we broke the BOWS watermark," in *Proceedings of The International Society for Optical Engineering*, E. J. Delp III and P. W. Wong, Eds., vol. 6505 of *Proceedings of SPIE*, San Jose, Calif, USA, January 2007.

[10] M. Barni and A. Piva, "Is knowledge of the watermarking algorithm useful for watermark removal?" in *Proceedings of the 2nd Wavila Challenge*, pp. 1–10, 2006, 2007.