Hindawi Publishing Corporation
EURASIP Journal on Information Security
Volume 2007, Article ID 43034, 7 pages
doi:10.1155/2007/43034

Research Article

Reverse-Engineering a Watermark Detector Using an Oracle

Scott Craver, Idris Atakli, and Jun Yu

Department of Electrical and Computer Engineering, Binghamton University, Binghamton, NY 13902, USA

Correspondence should be addressed to Jun Yu, jyu7@binghamon.edu

Received 7 May 2007; Accepted 22 October 2007

Recommended by A. Piva

The Break Our Watermarking System (BOWS) contest gave researchers three months to defeat an unknown watermark, given three marked images and online access to a watermark detector. The authors participated in the first phase of the contest, defeating the mark while retaining the highest average quality among attacked images. The techniques developed in this contest led to general methods for reverse-engineering a watermark algorithm via experimental images fed to its detector. The techniques exploit the tendency of watermark algorithms to admit characteristic false positives, which can be used to identify an algorithm or estimate certain parameters.

1. INTRODUCTION

The Break Our Watermarking System (BOWS) contest gave researchers a unique opportunity to test existing techniques for breaking a watermarking algorithm [1]. The contest also posed the researcher with a separate problem: given an unknown watermark detector, can one deduce the underlying algorithm from its output? This can also be attacked using adaptive inputs to a detector, except in this case the inputs are not used to find a better image, but to leak information about a detector structure's components. We used this approach to reverse-engineer the BOWS watermark, by posing carefully designed images to the watermark detector. Afterwards we extended our techniques to a general algorithm for plumbing a detection region with the goal of determining unknown algorithm parameters.

This paper is organized as follows: in Section 2, we summarize our participation in the BOWS contest, and the tactics we used to reverse-engineer the underlying watermarking algorithm. In Section 3, we extend our strategy to a general mathematical approach to deducing an unknown algorithm using oracle attacks. In Section 4, we show results for reverse-engineering a normalized correlation detector. We conclude that reverse-engineering a watermark detector is possible, although an intelligent human being can presently deduce far more with far fewer experimental inputs.

2. THE BOWS CONTEST

The Break Our Watermarking System (BOWS) contest challenged researchers to render an undetectable image watermark of unknown design. The goal was to attack watermarked images while maintaining a minimum quality level of 30 dB PSNR; the winner was the participant who maintained the highest average PSNR over three test images. In the first phase of the contest, the algorithm was secret; for the second phase, the algorithm was published.

Our research team applied the strategy of reverse-engineering the watermark before attacking the images. We determined the frequency transform, subband, and then an exploitable quirk in the detector that made it sensitive to noise spikes. This allowed us to achieve the highest average PSNR, 39.22 dB PSNR, by the end of the first phase of the contest [2]. The second phase was won by Andreas Westfeld, who achieved an amazing quality level of 58.07 dB PSNR. This quality level exceeded even the quantization error incurred by digitizing the image in the first place [3].

2.1. *Reverse-engineering the detector*

Our first step to defeating the watermark was determining the watermark "feature space," the image features which were modified in order to embed the watermark. Watermarks are often embedded in well-known transform domains as well as

(a) Image 1 (b) Image 2 (c) Image 3

Figure 1: The three images from the BOWS contest.

Figure 2: A severely degraded image (3.956 dB PSNR) with the watermark still detectable, suggesting a feature space of 8-by-8 AC DCT coefficients.

in the spatial domain; we suspected a common domain and constructed experimental images to test our suspicions.

A critical aspect of our attack was to construct severe false positives, images which should fail to trigger the detector unless a specific watermark feature space was being used.[1] We call this property *super-robustness:* the property of a watermarking algorithm to survive select types of quality degradation far beyond what any reasonable person would expect. If we can distort an image in a way to make it unrecognizable and if the watermark is still detectable, then we have found an attack to which it is super-robust. This is in fact a security weakness, because an unusual immunity to one attack (which we call a *mode of super-robustness*) can leak information about the underlying algorithm.

By testing various severe alterations of the image, we determined that the watermark followed an 8-by-8 block transform, surviving attacks like the one in Figure 2. We suspected a block DCT transform, which we further confirmed by experiment. We then submitted images with bands removed from each block. We determined the largest bands we could remove without hurting the watermark, to determine the suspected DCT subband used by the detector.

We used the knowledge that watermarks commonly reside in low-frequency and middle-frequency bands, a tactic described by Miller et al. [4]. We also guessed that a watermark algorithm would employ subbands following geometric patterns, like upper triangular or square subsets of the DCT matrix. Thus we erased lower-triangular sections of the matrix and the gnomonic sections. The union of these two attack regions gave us the largest "pattern" we could remove without damaging the watermark. Figure 3 shows the smallest region of geometric significance, which matched that used by the BOWS watermark [5].

2.2. Breaking the watermark

To break the watermark, we first damaged a large interval of feature space coefficients until the watermark was removed. Then, we iteratively fixed the damaged coefficients while the watermark remained undetectable. Our algorithm was as follows.

(1) Let $C_1, \ldots, C_n$ be all in-band DCT coefficients, sorted by decreasing magnitude.
(2) Find the smallest k such that the watermark fails when coefficients $C_1, \ldots, C_k$ are multiplied by a distortion value D.
(3) For $m = k - 1 \cdots 1$,
    (a) restore coefficient C_m to its original value;
    (b) if the watermark becomes detectable, redestroy coefficient C_m.

Our initial distortion value was $D = 0$, meaning that we eliminated DCT coefficients. Later we found that we could achieve a higher quality by amplifying target coefficients instead of zeroing them out. Table 1 shows the result for image 1, where scaling four coefficients destroyed the watermark.

[1] While a "false positive" typically denotes an unwatermarked image mistaken for a watermarked one, we also consider an image a "false positive" if it is so thoroughly altered that we expect it *should* have no detectable watermark.

Table 1: Successive attacks for image 1. By amplifying a small set of AC coefficients, the detector fails.

(Coefficient numbers)*amplification	PSNR
(12,69,107,127,132,140,141)*3.4	37.53 dB
(12,69,127,132,140,141)*3.4	38.19 dB
(12,69,127,140,141)*3.4	38.97 dB
(12,69,127,141)*3.55	39.67 dB

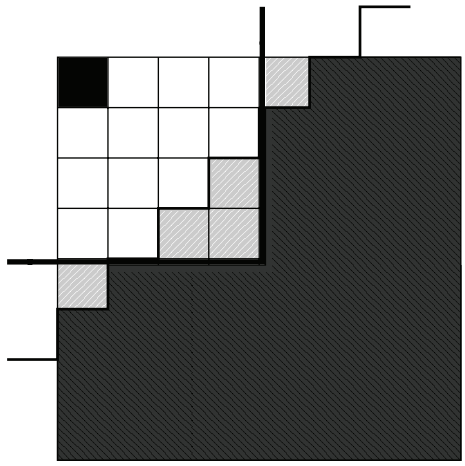


Figure 3: Experimental removal of DCT subbands. Shaded regions are the largest lower-triangular and gnomonic subbands removable without detector failure.

It is curious that so few coefficients need be modified: our previous analysis suggested a subband of 49152 coefficients per image (512-by-512 grayscale images, 4096 8-by-8 blocks, 12 AC coefficients taken per block,) so we suspected that our attack was exploiting some detector weakness.

3. REVERSE-ENGINEERING USING AN ORACLE

In the first BOWS contest, the sensitivity attack was widely used and proved to be very successful [3, 6, 7]. In this paper, we use oracle attacks for a different purpose: rather than removing the watermark, we seek to learn as much as possible about an unknown watermarking algorithm. We model a watermark detector as a three-stage algorithm. Images are first subjected to a *transform,* for example, a DCT or wavelet transform, to produce features used by the watermark embedder. Then, a particular *subband* is chosen for embedding and detection. Finally, the selected features are fed into a specific detection algorithm, which we model as computing a detection statistic which is compared to a threshold. Watermark detectors need not follow this structure, but many do. If common transforms and detector statistics are used, this structure implies a geometrically simple detection region that facilitates our attacks.

Our methods for reverse-engineering mirror the strategy used in the BOWS contest: create severe false positives to identify an algorithm by its modes of super-robustness. However, we also attempt to use the false positives to estimate parameters of the detection region.

3.1. Constructing a noise snake

Our generic method to create a useful false positive to a detector is to grow a "noise snake" using incremental uniform noise vectors. The technique of noise snakes entails the growth of multiple false positives along the surface of a detection region, leaking information about the watermark [8]. For certain detectors, such as normalized correlation detectors, vectors along the detection region boundary have a significant component along the watermark vector. This is because the detection region for normalized correlation is conical [9]. In this case, an expanding noise vector will move outward in a direction with a significant component along the watermark vector, and so a severe false alarm will leak information about the watermark.

Our noise snakes are constructed via the following algorithm.

(1) Start with test image I, treated here as a vector.
(2) Initialize our snake vector to $J \leftarrow I$.
(3) Do for $k = 1, 2, \ldots, K$ the following.
   (a) Choose a vector uniformly over the n-dimensional unit hypersphere $\mathbb{S}_n$.
   This can be accomplished by constructing an n-dimensional Gaussian vector $X \mathcal{N}(0, \sigma^2 I)$ and scaling the vector to unit length.
   (b) Choose a scaling factor α, which for normalized correlation is proportional to the length of J.
   (c) If $J + \alpha X$ still triggers the watermark detector, $J \leftarrow J + \alpha X$.
   (d) Else, discard X and leave J unchanged.

In high dimensions, a noise snake seems to converge quickly to the detection region boundary, and grow outward. Instead of snakes within the detection cone, we have snakes *on* a cone, which provide useful information about the detection region.

3.2. Estimation of a detection threshold

To use noise snakes to estimate detector parameters, we first need the following lemma.

Lemma 1. *If W is chosen uniformly over the unit n-sphere $\mathbb{S}_n$, and v is an arbitrary vector, the probability* $\Pr[W \cdot v > \cos\theta]$ *is*

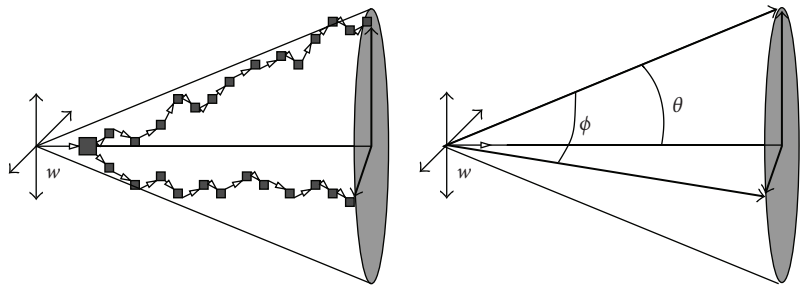
$$\frac{\mathcal{S}_{n-1}}{\mathcal{S}_n} \int_0^\theta \sin^{n-2} x \, dx, \tag{1}$$

where $\mathcal{S}$ is the surface volume of $\mathbb{S}$.

Proof. Since W is uniform, the probability of any set of vectors is proportional to its measure. Let us integrate over the v axis: consider point $t \in [-1, 1]$ representing the v component of the hypersphere. For each t, we have a shell of radius $r = \sqrt{1 - t^2}$, contributing a total hypersurface measure of $\mathcal{S}_{n-1} r^{n-2} \sqrt{dt^2 + dr^2}$. For example, a sphere in three

(a) Image 1 (b) Image 2 (c) Image 3

Figure 4: The three images after the attack. Note the few, but obvious, block artifacts.



(a) (b)

Figure 5: Two independently generated snakes have approximately perpendicular off-axis components.

dimensions is composed of circular shells, and each circular shell has contribution $2\pi r\sqrt{dt^2+dr^2} = \mathcal{S}_2 r^1\sqrt{dt^2+dr^2}$. The sphere portion with angle beneath θ is

$$\begin{aligned}\text{Area} &= \int_{r=0}^{r=\sin\theta} \mathcal{S}_{n-1} r^{n-2}\sqrt{dt^2+dr^2} \\ &= \int_{r=0}^{r=\sin\theta} \mathcal{S}_{n-1} r^{n-2}\sqrt{1+\frac{dt^2}{dr^2}}\,dr \\ &= \int_{r=0}^{r=\sin\theta} \mathcal{S}_{n-1} r^{n-2}\sqrt{1+\frac{r^2}{t^2}}\,dr \\ &= \int_{r=0}^{r=\sin\theta} \mathcal{S}_{n-1} r^{n-2}\frac{dr}{t}\end{aligned} \tag{2}$$

since $2t\,dt+2r\,dr = 0$. Substituting $r = \sin x$, we get $dr/t = dx$ and

$$\text{Area} = \mathcal{S}_{n-1}\int_0^{\theta} \sin^{n-2}x\,dx, \tag{3}$$

and we divide by the total surface area $\mathcal{S}_n$ to get the probability of hitting that region. □

The area of a unit hypersphere $\mathbb{S}_n$ is

$$\mathcal{S}_n = \begin{cases} \dfrac{2^{(n+1)/2}\pi^{(n-1)/2}}{(n-2)!!} & \text{for } n \text{ odd}, \\ \dfrac{2\pi^{n/2}}{((1/2)n-1)!} & \text{for } n \text{ even}. \end{cases} \tag{4}$$

The opening fraction $C_n = \mathcal{S}_{n-1}/\mathcal{S}_{n-2}$ therefore has a closed form [10]

$$C_n = \begin{cases} \dfrac{1}{2}\dfrac{(n-2)!!}{(n-3)!!} & \text{for } n \text{ odd}, \\ \dfrac{1}{\pi}\dfrac{(n-2)!!}{(n-3)!!} & \text{for } n \text{ even}. \end{cases} \tag{5}$$

Corollary 1. *For an arbitrary vector v, a uniformly chosen W, and a positive ϵ, $\Pr[W\cdot v < \cos(\pi/2+\epsilon)] = \Pr[W\cdot v > \cos(\pi/2-\epsilon)]$.*

Proof. We have $\Pr[W\cdot v < \cos(\pi/2+\epsilon)] = \Pr[W\cdot(-v) > \cos(\pi/2-\epsilon)]$. Because W is uniform, $\Pr[W\cdot(-v) > \cos(\pi/2-\epsilon)] = \Pr[W\cdot(u) > \cos(\pi/2-\epsilon)]$, where u is any vector. □

This means that if θ falls outside an interval of $\pi/2$, then the above probability drops exponentially with dimension n. This is the "equatorial bulge" phenomenon in high dimensions: as the dimension n increases, the angle between two

uniformly chosen direction vectors will be within ϵ of $\pi/2$ with high probability.

Corollary 2. *Two independently generated noise snakes have off-axis components which are close to perpendicular: as n increases, the angle between the off-axis components converges to $\pi/2$ in probability.*

Proof. The density function of the set of noise snakes is rotationally symmetric about the watermark axis due to the fact that each component, a uniform vector, has a rotationally symmetric density, and if s is a valid noise snake, so is Ts, where T is any rotation holding the cone axis constant.

Because of this, the probability $\Pr[S \in F] = \Pr[TS \in TF]$. If we subtract w and then normalize each snake, the symmetric distribution implies that $W = (S-w)/\|S-w\|$ uniformly distributed over the unit $n-1$ sphere.

To show that the angle converges to $\pi/2$ in probability, we observe that for noise snakes S and T, with off-axis components $W_1 = (S-w)/\|S-w\|$ and $W_2 = (T-w)/\|T-w\|$, and for an $\epsilon > 0$,

$$\begin{aligned}\Pr[|W_1 \cdot W_2| > \cos(\pi/2-\epsilon)] &= 2\frac{\mathcal{S}_{n-1}}{\mathcal{S}_n}\int_0^{\pi/2-\epsilon}\sin^{n-2}x\,dx \\ &\leq \frac{(n-2)!!}{(n-3)!!}\frac{\pi}{2}\sin^{n-2}(\pi/2-\epsilon).\end{aligned} \tag{6}$$

For any ϵ, there exists an N such that for $n > N$, $(n-2)/(n-3)\sin\pi/2 - \epsilon < 1$, and so this bound goes to 0. Hence the probability of falling outside an ϵ of $\pi/2$ drops to 0, and so $\cos^{-1}(W_1 \cdot W_2)$ converges to $\pi/2$ in probability. □

This observation gives us a simple method to estimate the cone angle from two constructed noise snakes X and Y of equal length. Using trigonometry, we have $(\sqrt{2}r\sin\theta)^2 = r^2 + r^2 - 2rr\cos\phi$, where ϕ is the angle between the snakes and r is the snake length (see Figure 5). Rearranging,

$$\sin^2\theta = 1 - X \cdot Y, \tag{7}$$

then we can calculate the cone angle and detector threshold by generating two snakes of sufficient length, and computing their dot product.

3.3. Estimation of feature space dimension

Once we have an appropriate estimate for the cone angle, we can apply another technique to estimate the feature space size, a more useful piece of information. To achieve this, we use the detection oracle again to deduce the error rate under two different noise power levels.

If we have a watermark vector w which falls within the detection cone, and add a uniform noise vector r, the probability of detection is

$$\Pr[\delta = 1] = \frac{\mathcal{S}_{n-1}}{\mathcal{S}_n}\int_0^{\psi}\sin^{n-2}x\,dx, \tag{8}$$

$$\psi = \theta + \sin^{-1}\left(\frac{\|w\|}{\|r\|}\sin\theta\right), \tag{9}$$

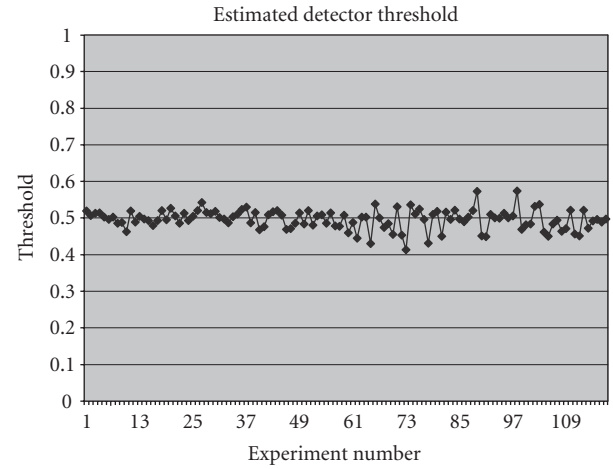


Figure 6: Estimated threshold using two noise snakes. The threshold value is $\tau = 0.5$.

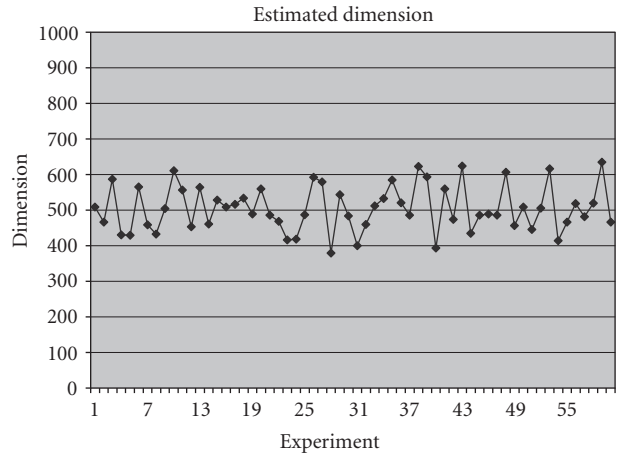


Figure 7: Estimated dimension using two noise snakes. The feature dimension is $n = 500$.

where θ is the cone angle, which we estimate using the technique described earlier. The second equation has one unknown, the watermark length $\|w\|$. The top equation has one unknown, n; the hit rate P_Y can be estimated by experiment.

If we then consider P_Y for uniform noise vectors of length A, and then for noise of length B, we can combine these equations into the following identity:

$$\tan\theta = \frac{A\sin\psi_A - B\sin\psi_B}{A\cos\psi_A - B\cos\psi_B}, \tag{10}$$

where ψ_A and ψ_B are the integration limits in (8). Here is our algorithm.

(1) Choose power levels A and B. They can be arbitrary, as long as the error rate under those noise levels is reasonably estimable.
(2) Use the watermark detector to estimate P_A and P_B, the detection rate under unform noise of lengths A and B, respectively.
(3) For all suspected values of n, do the following.

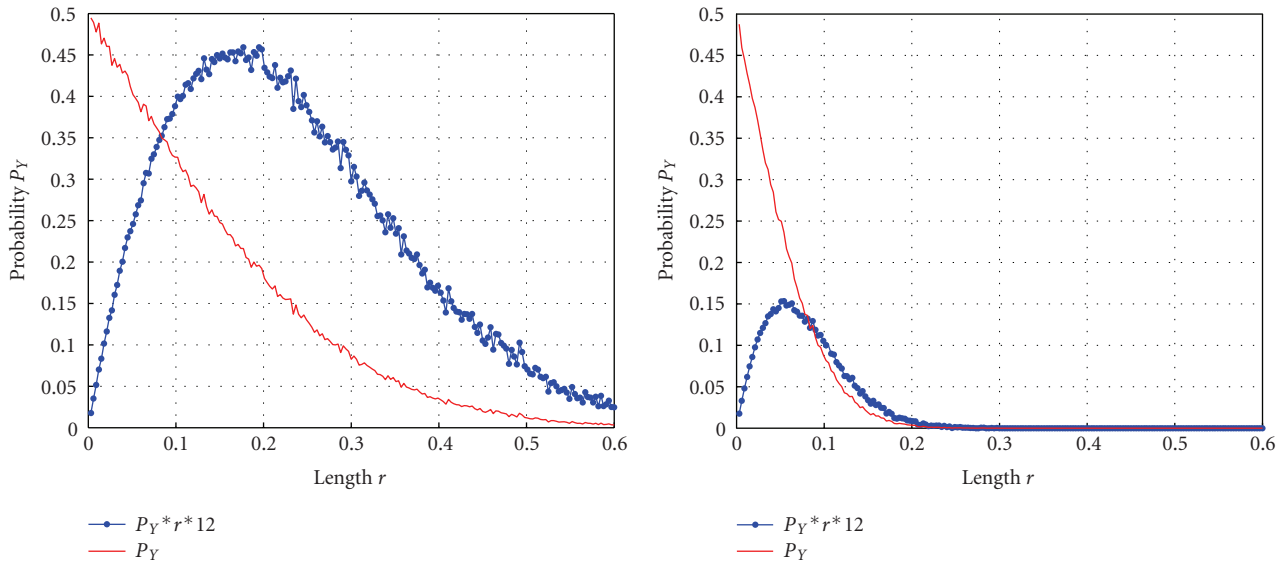


Figure 8: Detection rate P_Y and average growth rate αP_Y as a function of the step length α. Above, a cone angle of $\pi/3$ and $n = 1000$. Below $\pi/3$ and $n = 9000$. The optimal step length to maximize αP_Y is about 0.16 and 0.06, respectively.

(a) Use the hypothesized n and estimated detection rates in (8) to estimate ψ_A and ψ_B. These parameters are easily determined by Newton's method in one dimension, because P_A and P_B are integrals with respect to ψ_A and ψ_B, respectively.

(b) Compute $\epsilon_n \leftarrow |\tan\theta - (A\sin\psi_A - B\sin\psi_B)/(A\cos\psi_A - B\cos\psi_B)|$.

(4) Choose the value of n that minimizes the error ϵ.

4. RESULTS

4.1. Estimating detector parameters

We tested these techniques using a generic watermark detector with a feature space of variable size selected from 8×8 AC DCT coefficients. We used normalized correlation with a detection threshold of 0.5.[2] We first generated noise snakes to deduce the detector's threshold. Figure 6 shows our estimates for a detector with $\tau = 0.5$. This required an average of 1016 detector queries per experiment, to generate two snakes. Figure 7 shows the corresponding dimension estimates, once the threshold is deduced to be 0.5.

Note that in this detector the asymptotic false alarm probability is approximately 2.39×10^{-33}. This means that we can roughly estimate a very low false alarm probability in only thousands of trials.

[2] A proper detector should have a much higher threshold, since for $\tau = 0.5$, the false alarm rate is unnecessarily low. However, in our experience watermark detectors are often designed in an *ad hoc* manner, and a threshold value exactly between 0.0 and 1.0 is not uncommon.

4.2. Optimal step length

When generating a noise snake by adding a uniform noise vector, we must confront two conflicting factors: large noise vectors are more likely to move us out of the detection region, but small noise vectors contribute little length per iteration. The ideal noise vector length is one which maximizes the expected increment $\|\alpha X\| \cdot \Pr[\delta = 1]$.

When choosing a noise increment for a noise snake, the appropriate amplification factor α is proportional to the length of the snake as it grows. This can be seen by a simple geometric argument: the cone is congruent to scaled versions of itself. Thus if there is an optimal length α to extend a snake of length 1, then $M\alpha$ is optimal to extend a snake of length M. We need only estimate the appropriate α for a snake of unit length. Unfortunately, the optimal growth rate depends on both cone angle and dimension, both of which are unknowns to the reverse-engineering.

The growth rate of a noise snake is thus exponential in the number of queries. However, the growth rate is slow: Figure 8 shows some estimates for α which range from 0.16 to 0.06, with n in the thousands. For larger feature sets, growth is small. We determine in our experiments that for realistic feature sizes, a snake of useful length requires a number of queries roughly proportional to the dimension n.

5. CONCLUSION

We have developed several techniques for the reverse-engineering of a watermark detector by construction of severe false alarms. This approach mirrors our strategy in the BOWS contest, in which we constructed experimental images by trial and error, rather than by generic algorithm.

Our experience in the BOWS contest shows us that human-guided reverse-engineering is much faster, requiring

dozens of queries rather than hundreds or thousands. Partially, this was due to human knowledge of common watermarking methods and transforms. Future research will explore how this knowledge can be codified to guide a generic algorithm.

On the other hand, there may be means to prevent this type of reverse-engineering attack. Suppose we are watermark designers, and we decide that enough is enough, that we have had it with these snakes on this cone, what are we going to do about it? The obvious remedy is preventing severe false alarms, but it is not clear how to guarantee this. Any watermark algorithm which embeds data in a set of features is prone to this attack, because the remainder of the image can be obliterated whilst preserving the watermark. This allows severe false positives which leak information about the detector.

ACKNOWLEDGMENTS

This research is made possible by support from the Air Force Office of Scientific Research, under Award FA9550-95-1-0440. Example code for algorithms in Sections 3.2 and 3.3, for estimating detector threshold and dimension, and for estimating optimal snake growth, can be found online at http://bingweb.binghamton.edu/~scraver/snakeCode.tar.

REFERENCES

[1] "The Break Our Watermarking System (BOWS) contest," http://lci.det.unifi.it/BOWS/.

[2] S. Craver, I. Atakli, and J. Yu, "How we broke the BOWS watermark," in *Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505 of *Proceedings of SPIE*, San Jose, Calif, USA, January 2007.

[3] Westfeld A., "Tackling bows with the sensitivity attack," in *Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505 of *Proceedings of SPIE*, San Jose, Calif, USA, January 2007.

[4] M. L. Miller, G. J. Doerr, 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.

[5] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1673–1687, 1997.

[6] A. Westfeld, "Lessons from the bows contest," in *Proceedings of the Multimedia and Security Workshop*, vol. 2006, pp. 208–213, Geneva, Switzerland, 2006.

[7] P. Comesaña and F. Pérez-González, "Two different approaches for attacking bows," in *Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505 of *Proceedings of SPIE*, San Jose, Calif, USA, January 2007.

[8] S. Craver and J. Yu, "Reverse-engineering a detector with false alarms," in *Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505 of *Proceedings of SPIE*, San Jose, Calif, USA, January 2007.

[9] M. L. Miller, I. J. Cox, and J. A. Bloom, *Digital Watermarking*, Morgan Kaufman, San Francisco, Calif, USA, 2002.

[10] "Hypersphere—from Wolfram MathWorld," http://www.mathworld.wolfram.com/Hypersphere.html.