14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

BENCHMARKING IMAGE WATERMARKING ALGORITHMS WITH OPENWATERMARK

Benjamin Michiels and Benoît Macq

Laboratoire de Télécommunications et Télédétection, Université catholique de Louvain
2 Place du Levant, B-1348, Louvain-La-Neuve, Belgium
phone: +32 10 47 80 74, fax: +32 10 47 20 89, email: michiels, macq@tele.ucl.ac.be
site: www.openwatermark.org

ABSTRACT

In this paper, we present the Openwatermark framework, which is an open-source web-based system dedicated to the benchmarking of watermarking algorithms. We show how to use this system with practical examples and demonstrate how its may significantly improve collaboration between members of the watermarking community, as it does not impose the usage of a specific operating system or programming language to the user.

1. INTRODUCTION

Recent years have seen the emergence of a substantial number of watermarking schemes aiming at fulfilling certain requirements (such as robustness, security, imperceptibility/transparancy, complexity and possibility of verification [11]) related to a specific usage scenario. As the optimization of these parameters is mutually competitive, a reasonable trade-off is always necessary [18]. The role of benchmarking is to provide a fair and automated evaluation of these parameters [16].

Since then, a number of benchmarking systems have been proposed (for a more complete overview, refer to [18]):

- The StirMark benchmark system [6, 21] provided several attacks modeling common image processing operations, and introduced a couple of malicious attacks based on random bilinear geometric distortions [20]. Later, a similar system was elaborated to assess the performance of audio watermarking schemes [5, 12].
- The OptiMark benchmark [4, 23] provided a graphical user interface allowing the evaluation of several statistical characteristics of an image watermarking scheme (such as the receiver operating characteristics (ROC) curves [25]).
- The CheckMark benchmark [2, 19] provided a broad range of attacks, and introduced new watermark estimation-based attacks [24]. This project, as the former one, is closely related to the CertiMark [1] project whose resources and results are not publicly available.

Even though these systems brought significant improvements in a number of watermark benchmarking issues, it also appeared that their practical use was only relevant to certain watermarking schemes and imposed heavy constraints (such as the operating system and/or the programming language) to the user. This lack of flexibility impede the collaboration process between scientists wanting to share their resources. In order to fulfill this urgent need, two projects, the OpenWatermark framework [17] and the Watermark Evaluation Testbed (WET) [7, 15], are being developed.

This paper is structured as follows. In section 2, we briefly present the architecture and characteristics of the OpenWatermark framework. In section 3, we show how to browse the system and review the resources already available on the framework. In section 4, we introduce two simple use cases illustrating two common watermarking scheme evaluation scenarios (that is, a watermark detection test and a watermark extraction test) and show how to implement them with the OpenWatermark framework. In section 5, we draw a conclusion presenting the advantages and drawbacks of the proposed framework.

2. OPENWATERMARK ARCHITECTURE

2.1 OpenWatermark components

OpenWatermark is a distributed application system whose initial purpose is to allow the execution and the comparison (i.e. benchmarking) of programs uploaded by the user. Its architecture is composed of three parts, as illustrated below (*cf* figure 1).

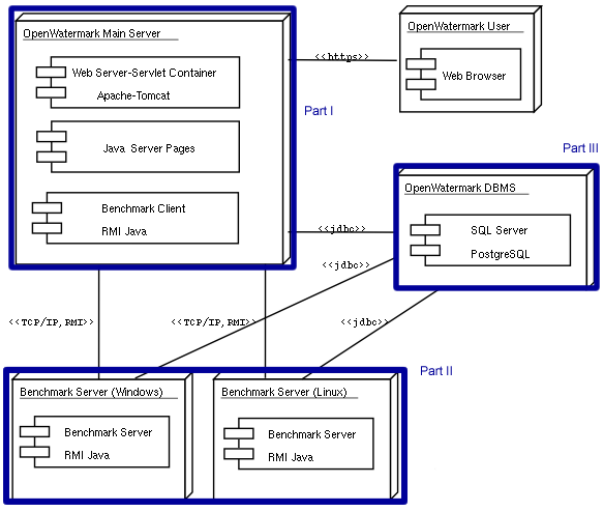


Figure 1: OpenWatermark deployment diagram

- **Part I.** The web server and a *RMI* client compose the first part (noted *Part I* in figure 1) of the OpenWatermark platform. First, the user logs into a web site using her/his preferred web browser, fills a form where she/he is asked to provide some specifications of her/his program (such as the programming language used, the syn-

tax of its command-line arguments), and to upload it. After that, she/he can launch the execution of her/his program and fetch the results when the execution terminated successfully[1]. This process is transparent to the user: technically speaking, the client binds the web site with the benchmark servers, which will execute programs.

- **Part II.** The *RMI* benchmark servers compose the second part of the platform (noted *Part II* in figure 1). These are constituted of a cluster of machines (2 for the moment) dedicated to dowload the executable from the database (as well as the associated data sets, the parameters that should be used and the expected output type), run it using the previously specified command-line options, and upload back to the database the results of this execution. The scheduling rule (here, the machine selected for the execution) is simply related to the type of the executable.
- **Part III.** The *SQL* database compose the third part of the platform (noted *Part III* in figure 1). It centralizes all information related to data sets and executables, as well as the information related to processes and users.

2.2 OpenWatermark characteristics

All the components of the OpenWatermark platform use the *Java* technology [2]. This architecture is therefore independent of the operating system used, and hence easily portable on any platform supporting *Java*, *RMI* and *JDBC*.

Furthermore, the OpenWatermark platform supports *Windows* and *Linux* executables (written in *C* or $C++$), as well as *Matlab* and *Python* scripts. As OpenWatermark is dedicated to evolve and develop according to the users' remarks and wishes, other possibly supported operating systems and/or programming languages could of course be added through time.

3. OPENWATERMARK USAGE

3.1 OpenWatermark browsing

The OpenWatermark website is divided in two parts. The first one, available to everyone, contains various articles and links in the area of watermarking. The second one contains the OpenWatermark application and is reserved to registered members[3].

Once you are logged on, you can access your private administration area. There, you can upload your own test data and/or executables[4], build the application of your choice (including or not the already available resources), launch its execution and fetch the results[5].

In addition, the user can also share her/his resources with other members (by adhering to the same group) or make them publicly available (by asking to the webmaster).

3.2 Available resources

At this moment, the following image-related resources are publicly available: 37 attacks, 2 distortion metrics, 1 watermarking algorithm and 35 test images.

3.2.1 Attacks

Image attacks are divided into 6 subcategories:

- ▷ **Geometrical** (all geometric transformations): flipping, cropping, rotation (with possible additional cropping), uniform and non uniform scaling (i.e. with or without change of aspect ratio), rolling (i.e. shifting of rows or columns), affine transformations, up and down-sampling, defo-process (combination of several geometric transformations; taken from [9]).
- ▷ **Noising** (all attacks consisting in the addition of noise): gaussian and multiplicative gaussian noising, poisson noising, laplacian noising, salt-and-pepper noising.
- ▷ **Denoising** (all attacks consisting in the removal of noise): gaussian filtering, mean filtering, median filtering, midpoint filtering.
- ▷ **Format-compression** (all manipulations concerning the image format): JPEG compression, format changing.
- ▷ **Image Processing** (all common image processing transformations): color quantization, black-and-white and grey-level transformation, gamma correction, histogram modification, sharpening.
- ▷ **Malicious** (all remaining attacks, intended to remove the watermark; all attacks are taken from [2] or [6]): collusion attack, copy attack, rows or columns removal, StirMark attacks, rows or columns removal, synchronization removal, template removal.

For a detailed of these algorithms, please refer to our tutorial [3]; for an overview, see [8], [20] or [24].

3.2.2 Distortion metrics

In addition to the *PSNR* (for *Peak Signal to Noise Ratio*) distortion measure, the OpenWatermark platform also include a distortion measure taking the geometric distortions into account [22].

3.2.3 Watermarking algorithms

There is one watermarking algorithm available, based on the previous works of [9].

3.2.4 Test images

The test images currently available are taken from the StirMark project [6].

[1] The state of the execution (*succeed*, *failed*, *queued* or *running*) can be seen from the user interface at any time; these results are directly accessible either for download or (in the case of images) view from the user's web browser (*cf* section 3.1)

[2] The user interface is constituted of web pages written in *JSP* (*Java Server Page*), and communicates with the Java application responsible for the execution of the tested program running on each of benchmarking hosts using *RMI* (*Remote Method Interface*) and with the *SQL* server using *JDBC* (*Java Database Connectivity*).

[3] To get an account, just fill the registration form; the webmaster will then activate your account and provide you the required informations by e-mail.

[4] Except for the type of the executable (*cf* section 2.2), the only constraints it must obey concerns *(i)* its format, which has to be in command-line style with non-null value arguments (e.g. in $executableName\ [-optionName\ optionValue]$, if $optionName$ is mandatory, then the user must assign a non-null value $optionValue$ to this parameter; if $optionName$ is optional, then it must have a default value) and *(ii)* the uniqueness of its output result.

[5] For a detailed description of this procedure, please refer to the OpenWatermark HOWTO, available on-line [3] and frequently updated.

4. OPENWATERMARK USE CASES

4.1 Preleminary remarks and general assumptions

4.1.1 *General model of a watermarking system*

To illustrate use cases of the system, consider the following workflow (*cf* figure 2), representing the general model of a blind watermarking scheme[6] (inspired from [14]).

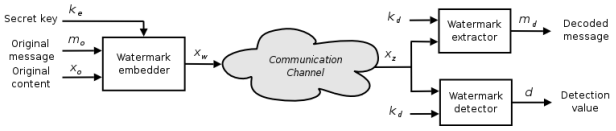


Figure 2: General model of a blind watermarking scheme

Here, an original content x_o is transformed into a watermarked version x_w, embedding an imperceptible message m_o with an encoding key k_e. Then, the watermarked content may suffer several kinds of transformations (unintentional or malicious) while passing through the communication channel; we will note as x_z the resulting content. After that, x_z can be processed to perform several tests; in this work, we will perform the following:

- **The watermark extraction test**. Using a decoding key k_d, the received content x_z is processed in order to obtain an estimate m_d of the original message. In the section 4.2, we show how to perform such a test on the OpenWatermark platform.
- **The watermark detection test**. Using a decoding key k_d, the received content x_z is processed in order to decide whether it has been watermarked. In the section 4.3 we show how to perform such a test on the OpenWatermark platform.

These tests will be carried out with the image watermarking scheme of D. Delannay; for a detailed description of this algorithm, see [9, 10].

4.1.2 *Result table format*

As mentioned earlier, if the execution is successfully terminated, the results are available for downloading. There are two different ways to fetch them: either *(i)* you manualy select each data instance (atomic fetch method), either *(ii)* you download a single archive file containing all the result tables associated with numeric results (aggregate fetch method). In OpenWatermark terms, what we mean by result table is illustrated below (*cf* figure 3): it represents the relation existing between the couple of input parameter values (each one being identified by a header, composed of the executable name and the option name) and the numeric output value generated by this couple[7].

As we will show throughout this section, this raw format possesses the advantages of **completeness** (because it contains all the information needed for data post-processing) and **flexibility** (because it allows a broad range of tests).

In the rest of the section, we will use several build-in spreadsheet functions to illustrate data post-processing. Ideally, the reader should have basic knowledge of a spreadsheet

[6] It means that the decoder has no access to the original content x_o.

[7] By convention, there is only one output per result table, value that can be found in the last column of the table.

	A	B	C	D	E	F	G
1	simu_embed -SFILENAME	JpgCompr -q	JpgCompr -i	simu_detect2 -SFILENAME	ber.py -j	ber.py -i	BER
2	baboon.pgm	80	Im_wat.pgm	J80_Im_wat.jpg	DamienDe	DamienDe	0
3	baboon.ppm	20	Im_wat.pgm	J20_Im_wat.jpg	DamienDe	DamienDe	0
4	alu.ppm	50	Im_wat.pgm	J50_Im_wat.jpg	DamienDe	DamienDe	0
5	clock.pgm	80	Im_wat.pgm	J80_Im_wat.jpg	DamienDe	DamienDe	0
6	bandon.ppm	20	Im_wat.pgm	J20_Im_wat.jpg	DamienDe	Da_ienDE	0.06
7	arctic_hare.ppm	50	Im_wat.pgm	J50_Im_wat.jpg	DamienDe	Eamk_jE	0.16
8	boat.pgm	80	Im_wat.pgm	J80_Im_wat.jpg	DamienDe	DamienDe	0
9	bear.ppm	20	Im_wat.pgm	J20_Im_wat.jpg	DamienDe	DamiEnDe	0.02
10	baboon.ppm	50	Im_wat.pgm	J50_Im_wat.jpg	DamienDe	DamienDe	0
11	alu.ppm	80	Im_wat.pgm	J80_Im_wat.jpg	DamienDe	DamienDe	0
12	lena512.pgm	10	Im_wat.pgm	J10_Im_wat.jpg	DamienDe	@a-iEjDe	0.06
13	bandon.ppm	50	Im_wat.pgm	J50_Im_wat.jpg	DamienDe	DamienDe	0
14	arctic_hare.ppm	80	Im_wat.pgm	J80_Im_wat.jpg	DamienDe	Eami_nEa	0.11
15	bear.ppm	50	Im_wat.pgm	J50_Im_wat.jpg	DamienDe	DamienDe	0
16	arctic_hare.ppm	90	Im_wat.pgm	J90_Im_wat.jpg	DamienDe	Dami_nEa	0.09
17	glasses.pgm	10	Im_wat.pgm	J10_Im_wat.jpg	DamienDe	DImk%n_e	0.11
18	boat.pgm	30	Im_wat.pgm	J30_Im_wat.jpg	DamienDe	DamiEnDe	0.02

Sheet1

Figure 3: OpenWatermark raw result table

program, even if no prerequisite is needed for understanding; for a detailed description of the following data post-processing operations, consult our tutorial [3].

4.2 Watermark extraction test

4.2.1 *Model description and assumptions*

In this example, we will evaluate the robustness of the watermarking scheme against the JPEG compression. The communication channel will be modeled as a JPEG compressor (*cf* figure 4). As quality assessment metric, we will use the bit-error rate (i.e. the fraction of bits differing between the original message m_o and the decoded message m_d).

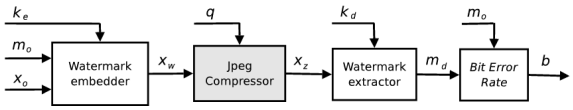


Figure 4: Watermarking extraction test

For the sake of simplicity we will consider that the following input parameters are constant (i.e. we don't assign values to these parameters, and let them take their respective default value):

- The encoding and decoding keys $k_e = k_d = k_e^*$, where k_e^* is the encoding key default value
- The original embedded message $m_o = m_o^*$, where m_o^* is the message default value.

In addition to that, we want to perform significant tests and thus apply watermark embedding and decoding processes on a large set of images. Here, for computational considerations, but without loss of generality, we will assume that:

- The original images $x_o \in X_0$, where $X_0 = \{x_{o,0}, \ldots, x_{o,9}\}$ is a subset of 10 images belonging to our image database.

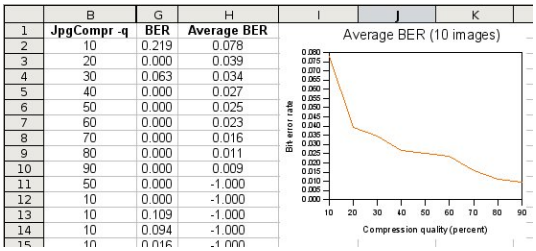
Finaly, we will evaluate the watermark extraction scheme performance by applying several compression quality parameters and will assume this assumption:

- The compression quality factor $q \in Q$, where $Q = \{10, 20, \ldots, 90\}$ (in percent).

4.2.2 *Data post-processing: averaging on an input*

First, we build the application and assign the corresponding input values; then, we run the tests and fetch the numeric results. In this section, we will show how to average the output (the bit-error rate b) on a input (the original image x_o) in a first time, and to how plot the averaged output with respect to another input (the compression quality q) in a second time.

- ▷ **Step 1**: *Sorting the data with respect to the compression quality.* From the original result table (*cf* figure 3), we first select and sort all the data with respect to the compression quality in ascending order, thanks to the `sort` build-in spreadsheet function.
- ▷ **Step 2**: *Averaging on the compression quality.* The averaging of the data is a more complex operation, due to the fact that the resulting data set differ in size. Fortunately, spreadsheet programs provide a large number of build-in functions to manipulate data. For our purpose, we will use the `average`, `row`, `mod` and `if` functions [8].
- ▷ **Step 3**: *Sorting with respect to the averaged output and plot the result.* Finaly, we select all the data once again, sort it with respect to the averaged output data (in descending order) and plot the results thanks to the `plot` build-in function (*cf* figure 5).



	B	G	H
1	JpgCompr -q	BER	Average BER
2	10	0.219	0.078
3	20	0.000	0.039
4	30	0.063	0.034
5	40	0.000	0.027
6	50	0.000	0.025
7	60	0.000	0.023
8	70	0.000	0.016
9	80	0.000	0.011
10	90	0.000	0.009
11	50	0.000	-1.000
12	10	0.000	-1.000
13	10	0.109	-1.000
14	10	0.094	-1.000
15	10	0.016	-1.000

Figure 5: result table sorting with respect to the averaged output and plotting of the results

4.3 Watermark detection test

4.3.1 Model description and assumptions

In this example, we will evaluate the performance of the watermarking detection scheme (*cf* figure 6). As illustrated below, two different tests will be performed: in the first one, the detector will process watermarked images, non-watermarked images in the second.

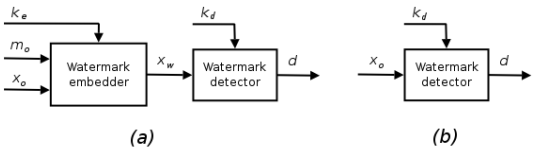


Figure 6: Watermarking detection test: *(a)* detection performed on watermarked images, *(b)* detection performed on non-watermarked images.

As in the preceding section, we will assume the following assumptions:

- The encoding and decoding keys $k_e = k_d = k_e^*$, where k_e^* is the encoding key default value.
- The original embedded message $m_o = m_o^*$, where m_o^* is the message default value.
- The original images $x_o \in X_0$, where $X_0 = \{x_{o,0}, \ldots, x_{o,9}\}$ is a subset of 10 images belonging to our image database.

Furthermore, it will be assumed in this section that the watermarking detection paradigm is the following (inspired from [13]). Given a content x, estimate which of the two following events is true: *(i)* H_0: there is no watermark in x, or *(ii)* H_1: there is a watermark in x.

As a detection rule, we will decide that : *(i)* H_0 is true if $d \leqslant \tau$, or *(ii)* H_1 is true if $d > \tau$ (where τ is a threshold we can decrease or increase in order to modify P_{FP} and P_{TP}).

To assess the quality of the detection scheme, we will use the detection value d produced by the detector (a high value meaning a high probability of watermark presence) to estimate the false positive P_{FP} and true positive P_{TP} probabilities[9] and establish an estimation of the receiver operating characteristics (ROC) curves.

Now, if we assume that the two probability density functions (noted $P_0(x)$ for non-watermarked and $P_1(x)$ for watermarked contents, respectively) follow a gaussian distribution[10] (with a mean and a standard deviation of μ_0 and σ_0, and μ_1 and σ_1 respectively), then it can be shown [13] that these functions will be given by:

$$P_{FP} = \begin{cases} \frac{1}{2}\, erfc\left(\frac{(\tau-\mu_0)}{\sqrt{2}\sigma_0}\right) & \text{if } \tau > \mu_O \\ \frac{1}{2}\left[1+erf\left(\frac{(\mu_0-\tau)}{\sqrt{2}\sigma_0}\right)\right] & \text{if } \tau \leqslant \mu_O \end{cases} \quad (1)$$

$$P_{TP} = \begin{cases} \frac{1}{2}\, erfc\left(\frac{(\tau-\mu_1)}{\sqrt{2}\sigma_1}\right) & \text{if } \tau > \mu_1 \\ \frac{1}{2}\left[1+erf\left(\frac{(\mu_1-\tau)}{\sqrt{2}\sigma_1}\right)\right] & \text{if } \tau \leqslant \mu_1 \end{cases} \quad (2)$$

4.3.2 Data post-processing: ROC curves generation

- ▷ **Step 1**: *Calculation of the gaussian estimations.* From the two original result tables, we first extract the output columns (i.e. *Detection value* 0 and *Detection value* 1). Then, we compute the mean and standard deviation of the two populations (thanks to the corresponding `average` and `stdev` functions), define an interval for the detection value abscissa, calculate the gaussian functions values for these points (thanks to the `normdist` function) and plot the results (*cf* left part of the figure 7).
- ▷ **Step 2**: *Plotting of the ROC curve estimation.* Then, we calculate the estimated P_{FP} and P_{TP} probabilities (given by the equations 1 and 2) (thanks to the `erf` and `erfc` functions) and plot the results on a XY chart.

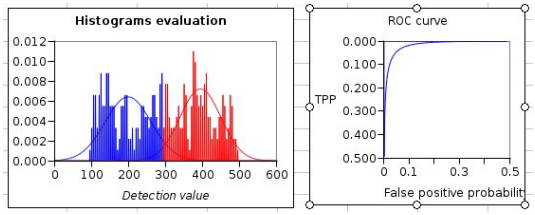


Figure 7: On the left side: histograms estimation; on the right side: estimated ROC curve

[8] See our tutorial [3] for technical details.

[9] That is, $P_{FP} = P\{H_1|H_0\}$ and $P_{TP} = P\{H_1|H_1\}$ (corresponding to the probabilities of detecting a watermark in a content x when there is effectively a watermark embedded and when there is none, respectively).

[10] Better approximations exist [8]; the purpose here is, rather, illustrative.

5. CONCLUSION

We have shown that the OpenWatermark features (independance with respect to the operating system and programming language, possibility of sharing resources, flexibility in application design and post-processing possibilities) make it a great tool for the watermark benchmarking community.

Future work will focus on improving the user interface (e.g. by including the possibility of defining benchmarking profiles), developing more algorithms, attacks and utilities for image (as well as for 3D meshes and video) watermarking schemes and improving the general administration and security policies of the web site. Of course, as by nature OpenWatermark is a collaborative project, all users' feedback, comments and suggestions are greatly appreciated.

REFERENCES

[1] Certimark [online]. Available at http://vision.unige.ch/certimark/.

[2] Checkmark [online]. Available at http://watermarking.unige.ch/checkmark/.

[3] Openwatermark [online]. available at http://www.openwatermark.org/.

[4] Optimark [online]. available at http://poseidon.csd.auth.gr/optimark/.

[5] Stirmark audio [online]. Available at https://amsl-smb.cs.uni-magdeburg.de/smfa//main.php.

[6] Stirmark [online]. Available at http://www.petitcolas.net/fabien/watermarking/stirmark/.

[7] Watermark Evaluation Testbed [online]. Available at http://www.datahiding.com/.

[8] Ingemar Cox, Jeffrey Bloom, and Matthew Miller. *Digital Watermarking, Principles & Practice*. Morgan Kaufmann, first edition, 2001.

[9] Damien Delannay. *Digital Watermarking Algorithms Robust Against Loss of Synchronization*. PhD thesis, Laboratoire de Télécommunications et Télédétection, Université Catholique de Louvain, April 2004.

[10] Damien Delannay and Benoît Macq. Generalized 2-D cyclic patterns for secret watermark generation. In *ICIP 2000 - IEEE Signal Processing Society International Conference on Image Processing, Vancouver, Canada, Sept. 10-13, 2000*, volume 2, pages =.

[11] J. Dittmann, P. Wholmacher, and K. Nahrstedt. Using cryptographic and watermarking algorithms. *IEEE Multimedia*, 8(Oct-Dec):54–65, 2001.

[12] Jana Dittmann, Andreas Lang, and Martin Steinebach. Stirmark benchmark: Audio watermarking attacks based on lossy compression. In Edward J. III Delp and Ping Wah Wong, editors, *Proceedings of SPIE, Security and Watermarking of Multimedia Contents IV, San Jose, CA, USA*, volume 4675, pages 79–90, 2002.

[13] C. W. Helstrom. *Statistical Theory of Signal Detection*. Pergamon Press, New York, 1960.

[14] Juan R. Hernández and Fernando Pérez-González. Statistical analysis of watermarking schemes for copyright protection of images. *Proceedings of the IEEE, Special Issue on Identification and Protection of Multimedia Information*, 87(7):1142–1166, July 1999.

[15] Hyung C. Kim, Hakeem Ogunleye, Oriol Guitart, and Edward J. III Delp. The watermark evaluation testbed (WET). In E. J. III Delp and P. W. Wong, editors, *Proceedings of SPIE, Security and Watermarking of Multimedia Contents VI, San Jose, CA, USA*, volume 5306, pages 236–247, June 2002.

[16] Martin Kutter and Fabien A. P. Petitcolas. A fair benchmark for image watermarking systems. In Edward J. III Delp and Ping Wah Wong, editors, *Proceedings of SPIE, Security, Steganography, and Watermarking of Multimedia Contents, San Jose, California, USA*, volume 3657, pages 223–226, 1999.

[17] Sébastien Lugan and Benoît Macq. Thread-based benchmarking deployment. In Edward J. Delp and Ping W. Wong, editors, *Proceedings of SPIE, Security, Steganography, and Watermarking of Multimedia Contents VI, San Jose, California, USA*, volume 5306, pages 248–255, June 2004.

[18] Benoît Macq, Jana Dittmann, and Edward J. Delp. Benchmarking of image watermarking algorithms for digital rights management. *Proceedings of the IEEE*, 92(6):971–984, June 2004.

[19] Shelby Pereira, Sviatoslav Voloshynovskiy, Maribel Madueño, Stéphane Marchand-Maillet, and Thierry Pun. Second generation benchmarking and application oriented evaluation. In *Information Hiding Workshop III, Pittsburgh, PA, USA, April 2001*.

[20] Fabien A. P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn. Attacks on copyright marking systems. In *Lectures Notes in Computer Science, SecondWorkshop on information hiding, Portland, Oregon, USA, 14-17 April*.

[21] Fabien A. P. Petitcolas, Martin Steinebach, Jana Dittmann, Caroline Fontaine, Frédéric Raynal, and Nazim Fatès. A public automated web based evaluation service for watermarking schemes: Stirmark benchmark. In Edward J. III Delp and Ping Wah Wong, editors, *Proceedings of SPIE, Security, Steganography, and Watermarking of Multimedia Contents III, San Jose, California, USA*, volume 4314, pages 575–584, 2001.

[22] Iwan Setyawan, Damien Delannay, Benoît Macq, and R. L. Lagendijk. Perceptual quality evaluation of geometrically distorted images using relevant geometric transformation modeling. In *SPIE/IST 15th Electronic Imaging, Santa Clara, USA, January 20-24, 2003*, volume 5020, pages 85–94, 2003.

[23] V. Solachidis, A. Tefas, N. Nikolaidis, S. Tsekeridou, A. Nikolaidis, and I. Pitas. A benchmarking protocol for watermarking methods. In *IEEE Int. Conf. on Image Processing (ICIP'01), Thessaloniki, Greece*, pages 1023–1026, 2001.

[24] S. Voloshynovskiy, S. Peirera, V. Inquise, and T. Pun. Attack-modelling: towards a second generation watermarking benchmark. *Signal Processing*, 81:1177–1214, 2001.

[25] Mark H. Zweig and Gregory Campbell. Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. *Clinical Chemistry*, 39:561–577, April 1993.