A public automated web-based evaluation service for watermarking schemes: StirMark Benchmark

Fabien A. P. Petitcolas[a], Martin Steinebach[b], Frédéric Raynal[c],
Jana Dittmann[b], Caroline Fontaine[d], Nazim Fatès[a]

[a] Microsoft Research, fabienpe@microsoft.com, n_fates@scientist.com
[b] GMD-IPSI, {dittmann, martin.steinebach}@darmstadt.gmd.de
[c] INRIA Rocquencourt, frederic.raynal@inria.fr
[d] USTL-LIFL, caroline.fontaine@lifl.fr

1. ABSTRACT

One of the main problems, which darkens the future of digital watermarking technologies, is the lack of detailed evaluation of existing marking schemes. This lack of benchmarking of current algorithms is blatant and confuses rights holders as well as software and hardware manufacturers and prevents them from using the solution appropriate to their needs. Indeed basing long-lived protection schemes on badly tested watermarking technology does not make sense.

In this paper we will present the architecture of a public automated evaluation service we have developed for still images, sound and video. We will detail and justify our choice of evaluation profiles, that is the series of tests applied to different types of watermarking schemes. These evaluation profiles allow us to measure the reliability of a marking scheme to different levels from low to very high.

Beside the known StirMark transformations, we will also detail new tests that will be included in this platform. One of them is intended to measure the real size of the key space. Indeed, if one is not careful, two different watermarking keys may produce interfering watermarks and as a consequence the actual space of keys is much smaller than it appears. Another set of tests is related to audio data and addresses the usual equalisation and normalisation but also time stretching, pitch shifting. Finally we propose a set of tests for fingerprinting applications. This includes: averaging of copies with different fingerprint, random exchange of part between different copies and comparison between copies with selection of most/less frequently used position differences.

2. NEED FOR EVALUATION

The growing number of attacks against watermarking systems (e.g., [1, 2, 3]) has shown that far more research is required to improve the quality of existing watermarking methods so that, for instance, the coming JPEG 2000 (and new multimedia standards) can be more widely used within electronic commerce applications.

We already pointed out that most papers have used their own limited series of tests, their own pictures and their own methodology and that consequently comparison was impossible without re-implementing the method and trying to test them separately[4]. But then, the implementation might be very different and probably weaker than the one of the original authors. This led to suggest that methodologies for evaluating existing watermarking algorithms were urgently required and we proposed a simple benchmark for still image marking algorithms.

With a well-defined benchmark, researchers and watermarking software manufacturers would just need to provide a table of results, which would give a good and reliable summary of the performances of the proposed scheme. So end users can check whether their basic requirements are satisfied. Researchers can compare different algorithms and see how a method can be improved or whether a newly added feature actually improves the reliability of the whole method. As far as the industry is concerned, risks can be properly associated with the use of a particular solution by knowing which level of reliability each contender can achieve.

3. EVALUATION TOOL

As a first step towards a widely accepted way to evaluate watermarking schemes we started to implement an automated benchmark server. The idea is to allow users to send a binary library of their scheme to the server which in turns runs a series of tests on this library and keeps the results in a database accessible to the scheme owner or to all 'watermarkers' through the Web.

3.1. METHODOLOGY – NEED FOR THIRD PARTY

To gain trust in the reliability of a watermarking scheme, its qualities must be rated. This can be done by:

- trusting the provider of the scheme and his quality assurance (or claims);
- testing the scheme sufficiently oneself;
- having the scheme evaluated by a trusted third party.

Only the third option provides an objective solution to the problem but the general acceptance of the evaluation methodology implies that the evaluation itself is as transparent as possible. This was the aim of StirMark and this remains the aim of the project to build a next generation of StirMark Benchmark. This is why the source code and methodology must be public so one can reproduce the results easily.

A question one may ask is: does the watermarking system manufacturer need to submit any program at all or can everything be done remotely using some interactive proof? Indeed, watermarking system developers are not always willing to give out software or code for evaluation, or company policy for intellectual property prevents them from doing this quickly. Unfortunately there is no protocol by which an outsider can evaluate reliably such systems without having access to the watermarking li-abrary[5].

3.2. REQUIREMENTS

Simplicity – In order to be widely accepted this service has a simple interface with existing watermarking libraries (only three functions must be provided by the user). It also takes into account the application of the watermarking scheme by proposing different evaluation profiles (sets of tests and images) and strengths[5]. These goals are reflected in Figure 1. Simplicity also means that the service should be easy to use:

- The client sends a library (which follows our general interface) to be evaluated and specifies the evaluation profile and level of assurance to be used;
- The StirMark Benchmark service automatically starts hundreds of tests on the library using its library of media;
- As soon as the test are finished the results are sent to the client and may later be published on the project website. Publication is optional or may be anonymised.

Customisation – For each type of watermarking scheme, we want to use a different evaluation profile without having to recompile the application tool. Definition of the profiles is not an easy task and requires agreement among the watermarking community. As we will see however, the choice of these profiles does not affect the design of the evaluation service and can be done later and tuned after experimenting the service.

Modularity and choice of tests – Watermarking algorithms are often used in larger system designed to achieve certain goals (e.g., prevention of illegal copying, trading of images). But here we are only concerned with the evaluation of watermarking (so the signal processing aspects) within the larger system not the effectiveness of the full system to achieve its goals. So the main functionalities we wish to evaluate include the perceptibility of the scheme, its capacity, its reliability (robustness to attacks and false alarm rate) and its performances (mainly the speed of execution). For each set of tests we are implementing ad-hoc libraries that are built easily on top of the core libraries.

- Perceptibility characterises the amount of distortion introduced during by the watermarking scheme itself. The problem here is very similar to the evaluation of compression algorithms. We allow the addition and use of different quality metrics, the simplest and most widely used one being the P.S.N.R.

- The capacity of a scheme is the amount of information one can hide. In most applications it will be a fixed constraint of the system so robustness test will be done with a random payload of given size. However knowing the relation between capacity and robustness is very important and our benchmark provide a test that help to analyse this trade-off by drawing different graphs[4].

- The robustness can be assessed by measuring the detection probability of the mark and the bit error rate for a set of criteria that are relevant to the application, which is considered. Part of these evaluation profiles can be defined using a finite and precise set of robustness criteria (e.g., S.D.M.I., IFPI or E.B.U. requirements) and one just needs to check them. Many of the possible tests can be deduced from previous works presenting attacks on watermarking schemes. In section 5.1 we present some that are dedicated to audio, as past research has mainly focused on still images.

- False alarms are difficult to measure and we are working on a method to estimate them automatically without having to do an exhaustive search of the key space. This will be detailed in section 5.3.

- The case of fingerprinting scheme will be considered in section 5.3.

- Finally, speed is very dependent on the type of implementation: software or hardware. Here we are only concerned with software implementation and our test just computes an average of the time required on a particular given platform to watermark and image depending on its size.

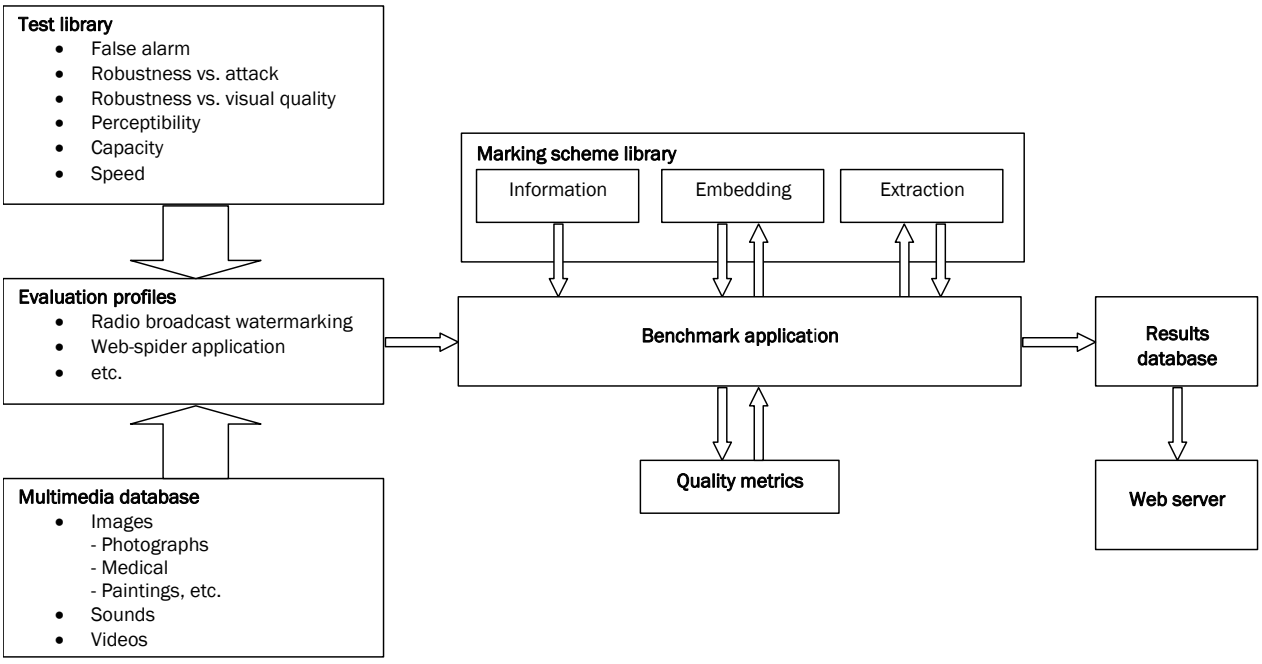

Figure 1 – Data flow for the watermarking evaluation service. The marking scheme is provided by the user as a library of functions (binary). This library exports in particular an information function which is used to select which evaluation profile has to be used. The evaluation profile is composed of a list of tests or attacks to be applied and a list of multimedia object required for the test and sorted by types and categories. All tests results are uploaded to an SQL server connected to a Web server.

One of the difficulties to set up a base of tests relates to the diversity of the algorithms and the images. Certain algorithms are created relative with a particular type of image when others want to be more general. Our base of images must be very wide. Thus, each algorithm will be tested on a randomly generated subset of images. In a second time, it will be possible for a user to specify one or more sets of images, either with the sight of the results obtained at the time of the first series of tests, or because its algorithm is dedicated to precise images in order to refine the results obtained.

4. ARCHITECTURE

4.1. SIMPLE INTERFACE

The evaluation service only requires three functions to be exported from the watermarking library supplied by the user. The first one, *GetSchemeInfo* provides information about the marking scheme such as its type and purpose, its operational environment, its author, version, release date, etc. The two other functions are the complementary *Embed* and *Extract* functions.

We tried to capture all possible cases and ended up with a solution where several parameters are provided but not all of them are mandatory. They include the original audio-visual signal, the watermarked signal, the embedding key, the 'strength' of the embedding, the payload, the maximum distortion tolerated and the certainty of extraction. This very simple technique allows interoperability with schemes of various types and only requires having a common unique source code header to maintain.

The *strength* parameter – which can be used to evaluate the compromise between imperceptibility, capacity and robustness – is assumed to have the following properties:

- *srength* is a floating point number between 0 and 100;
- the higher the value of *strength*, the lower the quality of the output image and the higher (hopefully) the 'robustness';
- *strength* = 0 corresponds to no watermarking (P.S.N.R. $\rightarrow \infty$);
- *strength* = 100 should correspond to a watermarked picture with P.S.N.R. around 20 dB;
- the distribution of *strength* should be 'harmonious'.

4.2. PROFILES

Support for various watermarking application is achieved by the use of an initialisation file per evaluation profile[5], in which each test has its own parameters stored. Table 1 gives an example of two different initialisation files that could correspond to the evaluation profile of a blind watermarking scheme that applies to radio broadcasting and to the evaluation profile of a non-blind watermarking scheme that applies to images for proof of ownership.

4.3. CLASS STRUCTURE

The project is being written using the C++ language to take full advantage of the inheritance and polymorphism features of an object-oriented language. Indeed, one of our ambitions is to provide a single tool that will be able to test different kinds of media such as images, sounds and videos. A UML simplified representation of the architecture is provided in Figure 2.

CBench is the general wrapper class for all possible benchmark. It creates a list of tests and a list of audio-visual signals according to the evaluation profile being used. It is also responsible for the management of the watermarking libraries and creates a *CMarkingScheme* object. This latter class acts as an interface between the evaluation service object model and the libraries provided by the users. *CMedium* is a base class used to handle audio-visual data and in particular memory allocation.

CTest takes a list of media and a marking scheme as an input and performs a test on it. The test can be whatever we need to evaluate the basic functionalities. Typical tests are false-alarm tests, embedding time and robustness tests (embedding, transfor-

mation, extraction). At last *CMediumTransformation* performs a transformation on a medium (an image as shown on the figure). For images, these include filtering, geometric transformation and any other distortion required for testing.

Table 1 – Initialisation file samples (evaluation profile).

Blind audio watermarking	Non blind image watermarking
[Test list] Test 1=Mean embedding time Test 2=Mean extraction time Test 3=Sound Low pass filter [Mean embedding time] Number of tests=100 [Mean extraction time] Number of tests=100000 [Sound Low pass filter] Cut frequency=2000 [Samples] Set 1=Radio broadcasts Set 2=Voices Set 3=Songs	[Test list] Test 1=Mean embedding time Test 2=Noise addition Test 3=Image JPEG compression [Mean embedding time] Number of tests=100000 [Noise addition] Noise start level=0.25 Noise end level=0.75 Step=0.05 [Image JPEG compression] Quality start=100 Quality end=75 Step=5 [Database] Set 1=Medical pictures Set 2=Photographs

Although the marking methods vary from one medium to another, many tests are common to all the media. For example a robustness test can be expressed as follows:

1. For each medium in a determined set:

    a. Embed a random payload with the greatest strength, which does not introduce annoying effects. In other words, embed the mark such that the quality of the output – for a given quality metric – is greater than a given minima.

    b. Apply a set of given transformations to the marked medium.

2. For each distorted medium try to extract the watermark and measure the certainty of extraction. The measure for the robustness is the certainty of detection or the bit error rate after extraction.

This procedure, which is parameterised in the evaluation profile, must be repeated several times since the hidden information is random and a test may be successful by chance. It is clear that such a test does not have to be specific to a certain type of medium and can be written using a *CMedium* base class. However the transformations used in the test must be aware of the type of medium. For instance a geometric transformation of an image has to be written using the derived class *CImage* as it needs functions specific to image processing such as *GetPixel* for instance.

The advantage of using an object-oriented structure is clearly to simplify the use of the software. For instance if one wants to add a new attack, say adding noise to a still image, then one needs to create a *CAddNoise* object, derived from the abstract class *CImageTransform*, and this can be done by writing one or two lines of code describing how to change the value of the pixels to add the noise. The same mechanisms can be applied to the writing of new tests. The object-oriented structure is used here to handle all the 'administrative tasks' such as reading the media in the database (this is done by the *CBench* class), embedding the mark (this is done by the *CMarking* class), exporting results, etc. This means that the researchers can focus their energy in writ-

ing appropriate code for attacks and tests without having to care for the management. The application then can generate a broad range of results and plots like the one proposed by Kutter et al[4].

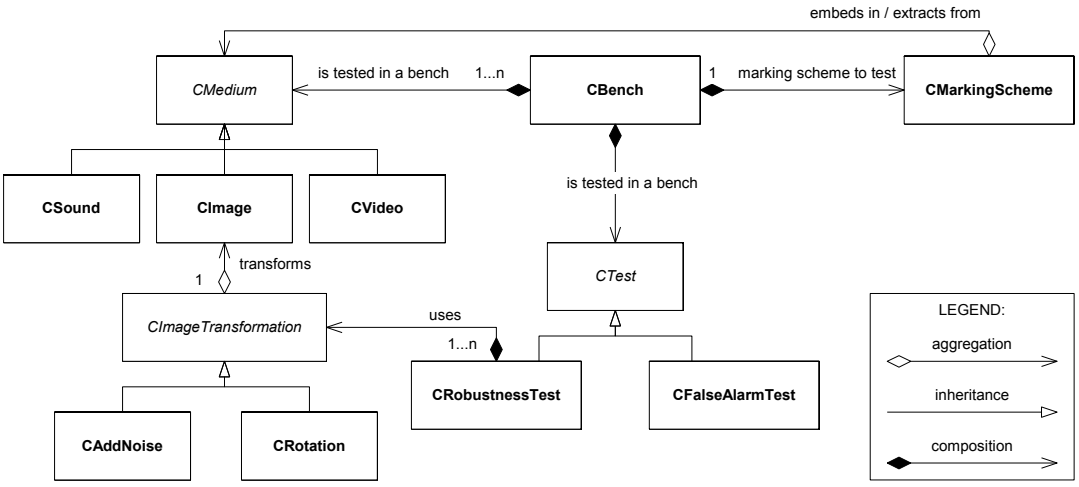

Figure 2 – Simplified class diagram of the core of the evaluation tool.

5. NEED FOR NEW TESTS

Many of the possible tests can be deduced from previous works presenting attacks on watermarking schemes. In this section we present some tests which are dedicated to audio – as past research has mainly focused on still images –as well as fingerprinting schemes. We also deal with the real size of key spaces.

5.1. ATTACKS AGAINST AUDIO MARKING SCHEMES

Any manipulation of an audio file can result in an attack on the embedded watermarks. Depending on the way the audio information will be used, some attacks are more likely than others. Based on this, we set up postproduction models for different environments. An example for such a model could be the preparation of audio material to be transmitted at a radio station: the material will be normalised and compressed to fit the loudness level of the transmission. Equalisation will be used to optimise the perceived quality. A denoiser or dehisser reduces unwanted parts of the audio information and filters will cut off any frequency which can not be transmitted.

If a watermark is used to detect radio transmission of commercials, it has to be robust against all the attacks described above, or the detection will not be possible as it is destroyed.

Another example is the Internet: if a company wants to embed watermarks as copyright protection, the watermark has to be robust against all operations usually worked on the material. In this case the main attack will be lossy compression like mp3, sometimes at high compression rates.

To make it possible to evaluate weaknesses of watermarking algorithms, we also build groups of attacks based on the way manipulation works. As the attacks in these groups work on the same principles, a weakness against one of the attacks makes it likely that the other attacks in the group will also destroy the watermark. Optimisation of the robustness of watermarking algorithms against one of the attacks in a group will also make it more robust against the rest. Recent tests have shown that if watermarking algorithms are robust against one attack in a group, they often also survive the other attacks of the same group. If, for example, an algorithm is robust against limiting – an attack of the dynamics group – it will also most likely survive compression or expansion as the changes in the material are similar.

Base on the attack models we have identified the following groups of attacks:

- *Dynamics* – These change the loudness profile of an audio file. Increasing or decreasing are the most basic attacks. Limiting, expansion and compression are more complicated, as they consist of non-linear changes depending on the material. There are even frequency dependent compression algorithms which only affect a part of the frequency range.

- *Filter* – Filters cut off or increase a selected part of the spectrum. The most basic filters are high-pass and low-pass filters but equalizers can also be seen as filters, they are usually used to increase or decrease certain parts of a spectrum.

- *Ambience* – This group consists of audio effects simulating the presence of a room. The most common effects are reverb and delay, which offer a lot of parameters depending on the quality of the effect.

- *Conversion* – Audio material is often subject to format changes. Mono data has to be doubled to be used in stereo environments. Sampling frequencies differ from 32 kHz to 48 kHz and now even 96 kHz. Sample size changes from 16 to 24 bit and back. Changes between digital and analogue representation is also still common. All these attacks induce quantisation noise and artefacts, as no conversion is perfect.

- *Lossy compression* – Audio compression algorithms based on psycho acoustic effects are used to reduce the amount of audio data by factor 10 or better. Examples are mp3, AAC, Windows Media, VQF and more. Compression algorithms differ on the way they work and which kind of data is reduced better. The basic approach is always to delete all information not perceived by the listener. As watermarking algorithms often work on similar principles, lossy compression can be a serious problem for watermarking.

- *Noise* – Noise can be the result of most of the attacks described above. Most hardware components in an audio chain also induce noise into the signal. A very common attack also is to try to add noise to destroy the watermark.

- *Modulation* – Modulation effects like vibrato, chorus, amplitude modulation or flanging[1] are usually not used in post-production. As most audio processing software includes such effects, they can be used as attacks to watermarks.

- *Time stretch* and *pitch shift* – These either change the length of an audio event without changing its pitch or change the pitch without changing the length. They are used for fine tuning or fitting audio parts into time windows.

- *Sample permutations* – This group consists of algorithms not used for audio manipulation in usual environments. Theses are specialised ways to attack watermarks embedded in audio files[3]. Examples are sample permutation, dropping samples and similar approaches.

All these attack types are being tested and their impact on audio material is being analysed. For the StirMark Benchmark environment most of the attacks will be implemented. Others can be simulated based on the information gained from the analysis.

5.2. WATERMARKING-KEY SPACE ISSUE

Test of the key space can be described as follow. If the document has been marked by key k, then the verification program must say *yes* only if it has been parameterised by k and it is important to verify that it answers *no* for other keys. The total number of keys is also really important since an attacker must not be able to try all keys and identify the right one. Hence it is necessary for the key space to be as large as possible, that is, it must contain at least 2^{64} elements. If all the bits of the key stream are free, then this implies that the key length must be at least 64, but if it is not the case, this means that the key must contain at least 64 free bits (bits of entropy).

Unfortunately, it is not really sufficient: if one is not careful, two different watermarking keys could produce interfering watermarks and as a consequence the actual space of keys is much smaller than it appears. So it is important to check with different

[1] Flanging is usually created by mixing a signal with a slightly delayed copy of itself, where the length of the delay is constantly changing.

detection keys if such interferences may appear. This can be done by trying different keys for detection: the right key k, and some others; some of these will be close to k, according to the Hamming distance, and some other will be picked at random quite far of it. Such a test will be included in the platform.

5.3. FALSE POSITIVES

There are two types of false positives:

- The detector/extractor finds a mark in a medium without mark;
- The detector/extractor finds a mark w' in a medium marked with w.

The first test is easily performed by taking randomly some mediums and trying to detect a mark inside. For the second one, we will take a marked medium and try to extract another mark, using the right stego key. The distance between those two marks could be an important factor to estimate the sensitivity to false positives.

5.4. MULTIPLE MARKS

It is important to know how an algorithm can handle multiple marks on a medium. Several issues are possible: either some or none will be detected[6]. If some are, what will allow selecting the mark from the right owner? If none is, this proves that the algorithm is not robust enough to collusions.

Since the marks have very different aspects, it is very difficult to estimate the real effects of multiple marking. According to the Hamming distance on the space of all possible marks and given a reference mark, one may try to mark the same medium with the reference mark and another mark whose distance is growing from the reference one. A related problem is fingerprinting.

5.5. POSSIBLE TESTS FOR FINGERPRINTING APPLICATIONS

Embedding unique customer identification as watermark into data is called fingerprinting[7]; this is typically used to track infringers. Because fingerprints are different for each customer, attackers can compare several fingerprinted copies to find and destroy the embedded identification string by altering the data in those places where a difference was detected. In our test suite we integrate four collusion attacks to compare different copies and to remove the fingerprinting information. Out of a great variety of attacks, we have chosen the most significant ones:

1. Comparison of copies from 2 customers and selection of the average of the difference values;
2. Comparison of copies from 2 customers and replacement of the differences with surrounding colours;
3. Comparison of n copies and selection of the most frequently used position differences for all n copies of the medium;
4. Comparison of n copies and selection of the less frequently used position differences for all n copies of the medium.

Table 2 demonstrates visually how the 2-customer attack works (in the case of images). The figure shows the images of two customers and their difference image. In our example we have used a block based watermarking technique. The watermarking information is embedded into 8×8 blocks by modifying DCT values. Our attack manipulates the found differences, stores the new image and performs the retrieval. The following steps are performed:

1. Fingerprint generation for customer 1 and embedding in image A;
2. Fingerprint generation for customer 2 and embedding in image B;
3. Generation of difference image of A and B.

Table 2 – Calculation of difference blocks by attackers.

Customer 1	Customer 2	Difference Image Blocks
		

The first attack:

1. For each found difference position we use image A and B and calculate the average values of the image blocks in A and B;
2. Replacement of the blocks by the average values;
3. Store new images in A′ and B′;
4. Retrieval of the fingerprints from A′ and B′ and evaluation of correctness.

The second attack:

1. For each found difference position we use image A and B and replace the difference blocks with surrounding image parts;
2. Store new images in A′ and B′;
3. Retrieval of the fingerprints from A′ and B′ and evaluation of correctness;

Attack 3 and 4 work with *n* copies and in step 3 we generate the differences over all *n* images. For attack 3 we replace the most recognized difference positions in the difference image from all n images, attack 4 replaces the less recognized difference positions from all *n* images.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have briefly described the architecture of a fully automated evaluation tool for digital watermarking schemes and several new test that this tool should include. It is the logical continuation of the early benchmark introduced into StirMark[3]. This new benchmark is operated on a piece of code that is provided by the user through a library and uses an object-oriented language to make multimedia handling quite simple. It also relies on pre-defined evaluation profiles (configuration files), allowing testing of different types of watermarking schemes automatically to different levels of assurance.

Hopefully this new generation of watermarking testing tool will be very useful to the watermarking community as it will provide a standard way of testing and it will allow fair comparison between different watermarking schemes!

7. REFERENCES

[1] Jonathan K. Su and Bernd Girod. Fundamental performance limits of power-spectrum condition-compliant watermarks. In Ping Wah Wong and Edward J. Delp, editors, proceedings of *electronic imaging, security and watermarking of multimedia contents II*, vol. 3971, pp. 314–325, San Jose, California, U.S.A., 24–26 January 2000. The Society for imaging science and technology (I.S.&T.) and the international Society for optical engineering (SPIE). ISSN 0277-786X. ISBN 0-8194-3589-9.

[2] Martin Kutter. Watermark copy attack. In Ping Wah Wong and Edward J. Delp, editors, proceedings of *electronic imaging, security and watermarking of multimedia contents II*, vol. 3971, pp. 371–380, San Jose, California, U.S.A., 24–26 January 2000. The Society for imaging science and technology (I.S.&T.) and the international Society for optical engineering (SPIE). ISSN 0277-786X. ISBN 0-8194-3589-9.

[3] Fabien A. P. Petitcolas, Ross J. Anderson and Markus G. Kuhn. Attacks on copyright marking systems. In David Aucsmith, editor, *second workshop on information hiding*, in vol. 1525 of *lecture notes in computer science*, pp. 218–238, Portland, Oregon, U.S.A., 14–17 April, 1998. ISBN 3-540-65386-4.

[4] Martin Kutter and Fabien A. P. Petitcolas. A fair benchmark for image watermarking systems. In Ping Wah Wong and Edward J. Delp, editors, proceedings of *electronic imaging, security and watermarking of multimedia contents*, vol. 3657, pp. 226–239, San Jose, California, U.S.A., 25–27 January 1999. The Society for imaging science and technology (I.S.&T.) and the international Society for optical engineering (SPIE). ISSN 0277-786X. ISBN 0-8194-3128-1.

[5] Fabien A. P. Petitcolas. Watermarking schemes evaluation. *I.E.E.E. signal processing*, vol. 17, no. 5, pp. 58–64, September 2000. ISSN 1053-5888.

[6] Fred Mintzer and Gordon W. Braudaway. If one watermark is good, are more better? In proceedings of *international conference on acoustics, speech and signal processing* (ICASSP'99), pp. 2067–2070, Phoenix, Arizona, U.S.A., 15–19 May 1999. The Institute for electrical and electronic engineers (I.E.E.E.).

[7] Jong-Hyeon Lee. Fingerprinting. In Stefan C. Katzenbeisser et al., editors, *Information hiding techniques for steganography and digital watermarking*, pp. 175–190. Artech House Books, December 1999. ISBN 1-58053-035-4.