

# The Watermark Evaluation Testbed (WET)

Hyung Cook Kim, Hakeem Ogunleye, Oriol Guitart and Edward J. Delp

Video and Image Processing Laboratory (VIPER)

School of Electrical and Computer Engineering

Purdue University

West Lafayette, Indiana USA

## ABSTRACT

While digital watermarking has received much attention within the academic community and private sector in recent years, it is still a relatively young technology. As such there are few widely accepted benchmarks that can be used to validate the performance claims asserted by members of the research community. This lack of a universally adopted benchmark has hindered research and created confusion within the general public. To facilitate the development of a universally adopted benchmark, we are developing at Purdue University a web-based system that will allow users to evaluate the performance of watermarking techniques. This system consists of reference software that includes both watermark embedders and watermark detectors, attack scenarios, evaluation modules and a large image database. The ultimate goal of the current work is to develop a platform that one can use to test the performance of watermarking methods and obtain fair, reproducible comparisons of the results. We feel that this work will greatly stimulate new research in watermarking and data hiding by allowing one to demonstrate how new techniques are moving forward the state of the art. We will refer to this system as the *Watermark Evaluation Testbed* or *WET*.

**Keywords:** digital watermarking, watermark evaluation, watermark benchmarking, image database, Taguchi loss function

## 1. INTRODUCTION

Digital Watermarking is the practice of hiding a message in an image, audio, video or other digital media element. Since the late 1990s, there has been an explosion in the number of digital watermarking algorithms published [1–5]. The sudden increase is mostly due to the increase in concern over copyright protection of content [6]. Because digital devices store content in digital form, there is no degradation in quality of data after copy is made [7]. The popularization of Internet and digital recording devices caused piracy to increase and content providers are seeking technologies to protect their rights [8]. Currently, cryptographic techniques are the most popular method used to prevent piracy [9]. The problem is that after the content is decrypted to be consumed, the content is no longer protected against piracy. Because digital watermarking can embed information related to the content in the digital media element and can be designed to survive many changes such as format conversion, D/A and A/D conversion and compression, it is seen as a technique to complement cryptography in preventing piracy. Applications of watermarking [5, 10–12] include broadcast monitoring, owner identification, proof of ownership, transaction tracking, authentication, copy control, and device control.

An important and often neglected issue in the design of digital watermarking methods is proper evaluation and benchmarking [13–15]. This lack of a proper evaluation in designing watermarking methods causes confusion among researchers and hinders the adoption of digital watermarking in various applications. 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 proposed scheme performances for end users. To address this issue, a few still image digital watermarking benchmarks have been proposed. These include StirMark [15–17], Certimark [18], Checkmark [19], Optimark [20, 21] and the new *Watermark Evaluation Testbed (WET)* system being developed at Purdue University which is described in this paper.

---

This work was supported by the Air Force Research Laboratory, Information Directorate, Rome Research Site, under research grant number F30602-02-2-0199. Address all correspondence to E. J. Delp, ace@ecn.purdue.edu.

This paper is organized as follows: In section 2, we describe the architecture and implementation issues of *WET*. In section 3, we describe an watermark evaluation example we will be able to do in the next version of *WET*. The conclusion and future work is given in section 4.

## 2. WATERMARK EVALUATION TESTBED ARCHITECTURE

In the following, we discuss the architecture and implementation of *WET*. The testbed consists of three major components: the Front End, the Algorithm Modules and the Image Database. Each component will be described below. Currently, *WET* runs on a 2.4 GHz Pentium 4 computer using the Red Hat Linux 8.0 operating system\*.

### 2.1. Front End

#### 2.1.1. Overview

The Front End is the end users' main interface into *WET*. The user interface abstracts much of the underlying architecture and allows the user to focus on the tasks to be performed. A top level implementation overview of the Front End is shown in Figure 1. The Front End consists of three major components: the web server, the database server, and the GIMP-Perl server. It provides a web interface whereby a user can select various tasks to be performed. These tasks include selecting images to be watermarked, selecting the algorithm with which to embed a watermark into the image, attacking a watermarked image, or detecting the presence of a watermark in a particular image. Two versions of *WET*, the *initial version* and *advanced version*, are available.

The *initial version* provides a static image database of about fifty images and a intuitive user interface. It is intended for either first time users to familiarize themselves with the system or for users who want to get a feel for how watermarking works. Several watermarking algorithms and attack algorithms are available to the user. The basic operations in the *initial version* are illustrated in Figure 2. The user selects a single image from the available images, watermarks the image, optionally attacks the watermarked image, and can detect the presence of a watermark in the watermarked (and possibly attacked) image. Several statistics including the CPU time used for embedding/detecting the watermark, the Mean Square Error (MSE) between the original and watermarked image, the difference image between the watermarked and original image, and some algorithm specific statistics such as the correlation value, number of bit errors, threshold value are provided as part of the result of these steps.

The *advanced version* provides an extensive image database. The user can select images with particular attributes, i.e. chrominance, resolution, height, width, category, etc., to work with. All metrics reported in the *initial version* are also reported in the *advanced version*. The *advanced version* is available in two modes: interactive and batch modes.

The interactive mode allows the user to manually step through the process illustrated in Figure 3. The process of the interactive mode is similar to that of the *initial version* with the following exceptions:

1. The user can select up to 25 images. Any selected task would be performed on all selected images.
2. Multiple attack algorithms can be performed on the same watermarked images. The *initial version* allows the user to repeatedly attack a watermarked image with the same attack algorithm but it does not allow the user to perform multiple different attacks on the same watermarked image.
3. For non-blind algorithms, the user can detect the presence of a watermark in all watermarked images against a single image or against as many as 25 images.

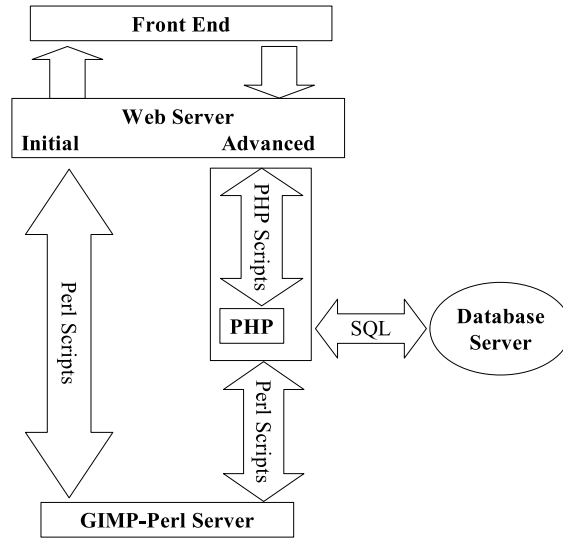
The interaction between all components that implements the interactive mode is illustrated in Figure 4.

The batch mode allows for user submitted jobs. The user selects the images, the watermark algorithm and corresponding parameters, the attack algorithms (in order of which they should be performed), and the detection(s) to be performed. The user created job is executed and the results are sent back to the user upon completion of the tasks in the form of email.

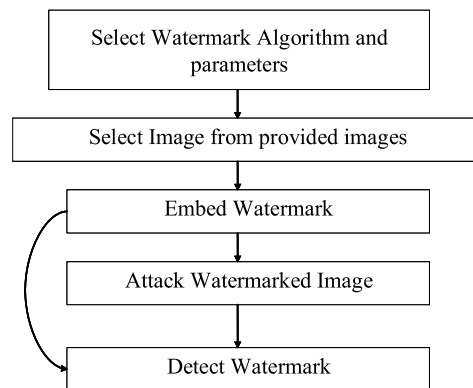
The Front End also has an administrative interface. The administrative interface is provided to allow an administrator to perform various tasks as image database management and security functions.

---

\*The system is located at <http://www.datahiding.org>. To obtain access to use *WET* contact [wetbug@ecn.purdue.edu](mailto:wetbug@ecn.purdue.edu).



**Figure 1.** Front End Architecture



**Figure 2.** Watermarking steps for the initial version

### 2.1.2. Software Implementation

The programming languages used to implement the Front End are:

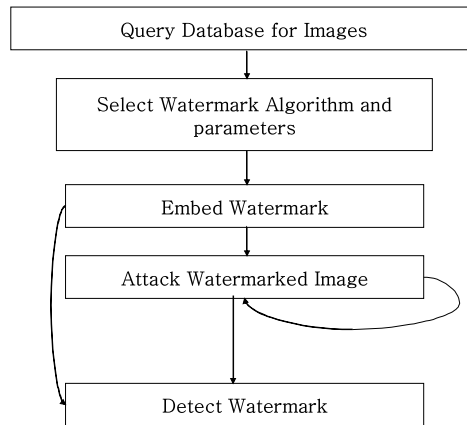
**JavaScript** Used for validating user input before sending the request from the web browser. It is also used to improve the user interface to the available options.

**Perl** Used for scripts that communicate with GIMP to execute the available algorithms available in *WET* and report the result to the user. It is also used for scripts that perform administrative tasks such as restarting *WET* and generating image thumbnails.

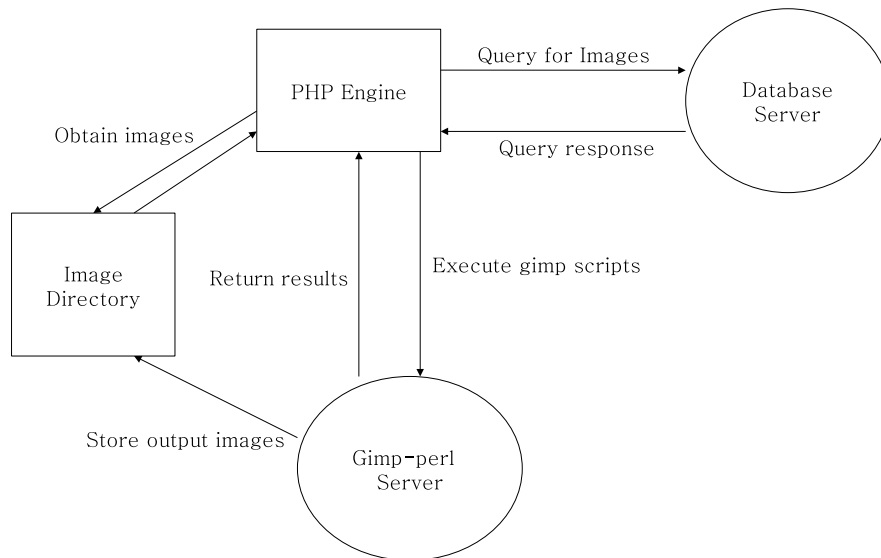
**HTML** Used for providing the static components of the user interface.

**SQL** Used for interacting with the database server to select images matching user specified criteria.

**PHP** Used extensively in the *advanced version*. It communicates with the Database Server, the Perl Scripts, and also generates pages to present results from other components to the user. It is used to dynamically generate Perl scripts to perform the tasks selected in the batch mode.



**Figure 3.** Watermarking steps for the advanced version



**Figure 4.** Interaction of major components in interactive mode

The MySQL database engine is used as the database server [22]. It maintains the attributes of all images available in *WET*. We use PHP to interface with the database server. Using PHP, we send queries and receive responses from the MySQL server.

The GIMP-Perl server is an add-on to the GNU Image Manipulation Program (GIMP) [23] that performs batch mode image manipulations. It takes programs written in Perl and executes them using GIMP and returns the result to the Perl interpreter. We use the GIMP-Perl server to run Perl scripts that execute GIMP plug-ins.

All the scripts used for the *initial version* are implemented in Perl. Each algorithm requires an image and specific parameters from the user. The images and parameters are selected on the user interface and passed to web server. The web server in turn calls the appropriate Perl script with the parameters. All Perl scripts used for embedding a watermark, detecting a watermark, or attacking a watermarked image have similar structures because algorithms are implemented as GIMP plug-ins [23].

A similar approach is used for the interactive mode of the *advanced version* except that the PHP engine acts as an intermediary between the web server and the Perl scripts. Once the user selects the images to watermark along with the corresponding parameters, the request is sent to the PHP engine via the web server and the PHP engine executes the appropriate Perl scripts. The results from the Perl scripts are passed back to the PHP engine. Upon completion of the Perl script, the PHP engine generates a page with the results and sends it to the web server for onward display to the user.

In the batch mode, the user selects the algorithms, images, other tasks to be performed, and the correct order of execution and then submits the request to the web server. The web server in turn forwards the request to the PHP Engine. The PHP Engine creates a Perl script by using several functions that generate the required code to perform the specific algorithms. Upon completion of the script generation, the PHP Engine executes the scripts. Currently, the results obtained upon job completion are sent to the user in the form of an electronic mail.

## 2.2. Algorithm Modules

It is desirable to develop tools that users can use standalone in their own test environments, allowing them to validate their tests locally before submitting them to a watermark benchmark site. To achieve this, the GNU Image Processing Program (GIMP) [23] was selected for use in *WET*. GIMP is designed to be augmented with plug-ins and extensions. Additionally, it provides full scripting support in various languages (Scheme, Perl, Java, Python, etc.). The advanced scripting interface of GIMP allows everything from the simplest task to the most complex image manipulation procedures to be easily scripted. The combination of extensibility and scripting support make GIMP a powerful environment for *WET*. The GIMP community also provides a large selection of plug-ins, which by their nature (geometric distortion, difference image) could be used for attack or measurement components of *WET*.

As previously mentioned, by allowing users to duplicate the test environment locally in a form they are familiar with, there is no distinction between their local development components and the ones needed for *WET*. Stated another way, an Experimenter can natively develop in GIMP and after they are ready for testing, they can submit the same binary to a watermark benchmark site they used in their local development. This eliminates the step of packaging the component for an external benchmark and never knowing where a bug was introduced.

We have been implementing several plug-ins for GIMP. Our plug-ins can be used in two different modes: Interactive Mode and Non-Interactive Mode. The former has a user interface where the user can choose the input parameters and the output parameters are displayed after using the plug-in. The Non-Interactive Mode performs the same function as the Interactive Mode but allows the plug-in to be called from another GIMP plug-in or scripts. The Non-Interactive Mode is used in *WET*.

### 2.2.1. Watermarking Algorithms

To extend watermarking algorithms to color images, we used the reversible color transform (RCT) [24] developed for JPEG2000. The Red, Green and Blue components of an image are transformed by RCT and we embed the watermark into the luminance component. The RCT is shown in Figure 5. We chose RCT among different color transforms because it preserves the image when a watermark is not embedded. We watermarked the luminance component because a watermark should be placed in the perceptually most significant components of an image [2].

One of the performance measures of a watermarking algorithm is computational complexity. We currently measure the computational complexity in terms of CPU execution time. Our GIMP plug-ins are written such that they return the CPU execution time as an output parameter. The watermarking algorithms we implemented as a GIMP plug-in include:

**Secure spread spectrum watermarking [2]:** This is a multiplicative spread spectrum watermarking algorithm and is based on the assumption that the watermark should be placed in the perceptually most significant components of an image.

$$\begin{array}{ll}
Y &= \lfloor \frac{R+2G+B}{4} \rfloor & G &= Y - \lfloor \frac{Dr+Db}{4} \rfloor \\
Dr &= R - G & R &= Dr + G \\
Db &= B - G & B &= Db + G
\end{array}$$

(a) Forward RCT

(b) Inverse RCT

**Figure 5.** Reversible Color Transform

**Blind wavelet watermarking [25]:** This is a blind wavelet watermarking algorithm. A typical problem with spread spectrum watermarking is that the order and the number of significant coefficients can change due to various image manipulations. This technique addresses this problem by using two different thresholds to embed and detect the watermark.

**Semi-Fragile watermarking [26]:** This is a semi-fragile watermarking method that can detect information altering transformations even after the watermarked content is subjected to information preserving alterations. It is also capable of tolerating some degree of change to the watermarked image.

**Quantization Index Modulation (QIM) watermarking [27, 28]:** QIM embeds a message by using an ensemble of quantizers. Our algorithm is based on low complexity method that is known as dither modulation.

**LOT Based Adaptive Watermark [29]** The LOT watermark algorithm embeds an invisible, robust watermark in an image using the spread spectrum scheme. The Lapped Orthogonal Transform (LOT) is chosen for the block-based orthogonal transform, instead of DCT, to avoid blocking effects. Moreover, the Human Visual System (HVS) properties are exploited to adjust the intensity of our embedded watermark according to the local features of the image.

**Locktography [30]** Locktography is a steganography technique which embeds data into a color still image by modifying the least-significant bits (LSB) of the image.

### 2.2.2. StirMark 3.1

StirMark 3.1 is from [16, 17, 31]. We implemented the default test mode in StirMark 3.1 as a GIMP plug-in. StirMark 3.1 implements the following possible attacks on a watermarked image: linear filtering, median filtering, Frequency Mode Laplacian Removal (FMLR), JPEG compression, color quantization, scaling, shear, aspect ratio, general linear transform, rotate and crop, rotate and scale, cropping, flip, remove row and column and original StirMark attack (random bilinear geometrical distortion).

### 2.2.3. Mean Square Error Module

One important performance measure is fidelity. Fidelity is the perceptual similarity between the original image and the watermarked image [5]. We currently measure the fidelity in terms of Mean Square Error (MSE). Mean Square Error is obtained as follows

$$MSE = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (p(i, j) - \hat{p}(i, j))^2$$

where  $M$  is the number of color components,  $N$  is the number of pixels,  $p$  is the original image and  $\hat{p}$  is the modified image. We implemented a plug-in that takes two same size images as input, and outputs the mean square error and the difference image.

### 2.3. Image Database

As mentioned before we use MySQL as the database engine. It maintains the attributes of all images available in the test-bed. We currently have approximately 1301 images which are copyright free. The images are from a variety of cameras and sensors including scanned photographs, x-ray images, ultrasound images, astrometrical images, line drawings, digital cameras, maps, and computer generated images. Each image in the database is stored with its attributes including chrominance, resolution, height, width, and category.

## 3. WATERMARK EVALUATION: A STATISTICAL FRAMEWORK

Our goal is to develop a “theory” for watermark evaluation and use *WET* to deploy the techniques. The following is a new statistical approach we will integrate into *WET*. We will use PSNR as our fidelity measure and Bit Error Rate(BER) and Receiver Operating Characteristic (ROC) analysis with or without attacks as our performance measure [20, 32].

### 3.1. Watermark Evaluation Parameters

We use similar parameters used in Optimark [20, 21]. These are:

- a set of images  $\mathbf{I} = \{I_i | i = 1 \dots N_I\}$
- a set of attacks  $\mathbf{A} = \{A_j | j = 1 \dots N_A\}$
- $N_K$ : Number of keys we use for each image
- a set of fidelity specifications  $\mathbf{Q} = \{Q_l | l = 1 \dots N_Q\}$

From this, a set of  $N_I \times N_K$  watermarked images with attack  $A_j$  and fidelity specification  $Q_l$  are generated, where the fidelity specification is described below. For the message, we use the bits generated by a pseudo binary random number generator with current time as the seed. To test as many keys as possible, we arbitrary use keys  $k = 1, \dots, N_I \times N_K$  to test the watermarking algorithm and use keys  $(i - 1)N_K + 1, \dots, iN_K$  for each test image  $I_i$ , where  $i = 1, \dots, N_I$ . We define  $PSNR_k, k = 1, \dots, N_I N_K$  to be the PSNR of the  $k$ th watermarked image with no attack.

### 3.2. Fidelity and Robustness Specification

Fidelity specification  $Q_l$  for an application is usually defined in terms of the Lower Specification Limit (LSL) [33]. For example, we could define average PSNR as

$$PSNR_{average} = \frac{1}{N_K N_I} \sum_k PSNR_k$$

and require the average PSNR be greater than LSL=45dB. Or we could define percentile as

$$\text{Percentile}(PSNR, \text{LSL}) = \frac{1}{N_K N_I} \sum_k \mathbf{I}(PSNR_k \geq \text{LSL})$$

where  $\mathbf{I}$  is the indicator function and then require  $\text{Percentile}(PSNR, 40\text{dB}) > 99\%$  where 99% is the yield requirement.

For fair benchmarking, one has to ensure that the methods under investigation are tested under comparable conditions [32]. In this paper, we will use the Taguchi loss function [33, 34] to specify PSNR and BER requirements for fair evaluation. The advantage of the Taguchi loss function over average and percentile is that it considers both the mean and variance of the distribution. The Taguchi method was used in [34] to build better watermarking algorithms and in [35] to select optimal parameters for video compression. Since PSNR and BER is defined for each watermarked image, we can define the fidelity and robustness requirements using the Taguchi loss function.

Since PSNR is a larger-is-better measure and BER is a smaller-is-better measure, we define a PSNR and BER Taguchi loss function as follows:

$$\text{PSNR}' = \left( \frac{1}{N_I N_K} \sum_k \left( \frac{1}{\text{PSNR}_k} \right)^2 \right)^{-\frac{1}{2}}.$$

$$\text{BER}' = \left( \frac{1}{N_I N_K} \sum_k (\text{BER}_k)^2 \right)^{\frac{1}{2}}.$$

We chose PSNR' as a measure of fidelity and use BER' and ROC analysis as a robustness measure for a watermarking technique for a set of image, key and attack pairs  $(I_i, K_i, A_i)$ . If we do not have enough data to obtain the ROC at a particular detection threshold, we assume that the watermark detection statistics for true positive and true negative are normally distributed and independent.

### 3.3. Experiments

We set the lower specification limit for PSNR' as 45dB. We chose the data payload to be 16 bits. This specification can be used as a specification for “fingerprinting” applications [36]. We tested  $N_K=1$  key for each image. For the image test set  $\mathbf{I}$ , we used our image database. It has  $N_I = 1301$  images. We test  $N_K \times N_I=1301$  keys.

#### 3.3.1. Watermarking Algorithms

Let  $m_n=0, 1, (n = 1, \dots, N_m)$  be the bits we want to embed.  $\mathbf{X}_n$  is the watermarked vector,  $\mathbf{S}_n$  is the original host vector or image,  $\mathbf{W}_n$  is a normalized watermark vector ( $\|\mathbf{W}_n\|=1$ ) and  $\alpha$  is the scaling factor. In this experiment, we generate a normalized watermark vector by normalizing an i.i.d. Gaussian Random Vector with distribution  $N(0,1)$ . For  $\alpha$ , we use a fixed value that satisfies the fidelity requirement for each test. We find  $\alpha$  by iteration. We define  $(\mathbf{X}_n)_i$  to be the  $i$ th element of  $\mathbf{X}_n$ .

We embed multiple bits using Spatial Division Multiplexing. We divide the image into 8x8 blocks and apply the Discrete Cosine Transform (DCT) to each block [37]. We evenly and randomly distribute each blocks to form  $N_m$  groups. We construct an original host vector  $\mathbf{S}_n$  by row concatenating all the blocks in the  $n$ th group excluding the DC components.

As test algorithms, we choose three blind watermarking algorithms: Additive Spread Spectrum Watermarking (ASSW), Multiplicative Spread Spectrum Watermarking (MSSW) [38] and Improved Spread Spectrum Watermarking [39]. Each algorithm can use a Gaussian Watermark Pattern. Each algorithm is a blind watermarking algorithm, that is, it does not require the original host image to detect the watermark. Summary of each algorithm in terms of embedding, decoding and detection is shown in Table 1, Table 2 and Table 3. As shown in Table 2, the three algorithms use the same decoding scheme.

#### 3.3.2. Attacks

We use StirMark [15–17] 3.1 software for our attacks. We evaluate each watermarking algorithms for the following attacks:

- Gaussian filtering (blur):  $\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} / 16$
- Simple sharpening:  $\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$
- JPEG compression with quality factor 70

Name	Embedding
ASSW	$\mathbf{X}_n = \mathbf{S}_n + \alpha(2m_n - 1)\mathbf{W}_n$
MSSW	$(\mathbf{X}_n)_i = (\mathbf{S}_n)_i + \alpha(2m_n - 1) (\mathbf{S}_n)_i (\mathbf{W}_n)_i$
ISSW	$\mathbf{X}_n = \mathbf{S}_n + (\alpha(2m_n - 1) - \mathbf{S}_n \cdot \mathbf{W}_n)\mathbf{W}_n$

**Table 1.** Watermark Embedding

Name	Decoding
ASSW	$\hat{m}_n = \begin{cases} 1 & \text{if } \mathbf{Y}_n \cdot \mathbf{W}_n > 0 \\ 0 & \text{otherwise} \end{cases}$
MSSW	$\hat{m}_n = \begin{cases} 1 & \text{if } \mathbf{Y}_n \cdot \mathbf{W}_n > 0 \\ 0 & \text{otherwise} \end{cases}$
ISSW	$\hat{m}_n = \begin{cases} 1 & \text{if } \mathbf{Y}_n \cdot \mathbf{W}_n > 0 \\ 0 & \text{otherwise} \end{cases}$

**Table 2.** Watermark Decoding

Name	Detection
ASSW	Watermark Present, if $\sum_n (2\hat{m}_n - 1)\mathbf{Y}_n \cdot \mathbf{W}_n > T$ Watermark Not Present, otherwise
MSSW	Watermark Present, if $\frac{\sum_n (2\hat{m}_n - 1)\mathbf{Y}_n \cdot \mathbf{W}_n}{\sum_n \sum_i  (\mathbf{Y}_n)_i } > T$ Watermark Not Present, otherwise
ISSW	Watermark Present, if $\sum_n (2\hat{m}_n - 1)\mathbf{Y}_n \cdot \mathbf{W}_n > T$ Watermark Not Present, otherwise

**Table 3.** Watermark Detection

Name	No Attack	Gaussian Filtering	Sharpening	JPEG
ASSW	(45.0, 45.0)	(31.6, 33.4)	(21.9, 22.8)	(38.9, 39.7)
MSSW	(45.0, $\infty$ )	(31.5, 33.3)	(22.5, 23.7)	(38.0, 38.8)
ISSW	(45.0, 45.0)	(31.6, 33.4)	(21.9, 22.8)	(38.9, 39.7)

**Table 4.** (PSNR', PSNR<sub>average</sub>) pair for different attacks

### 3.3.3. Results

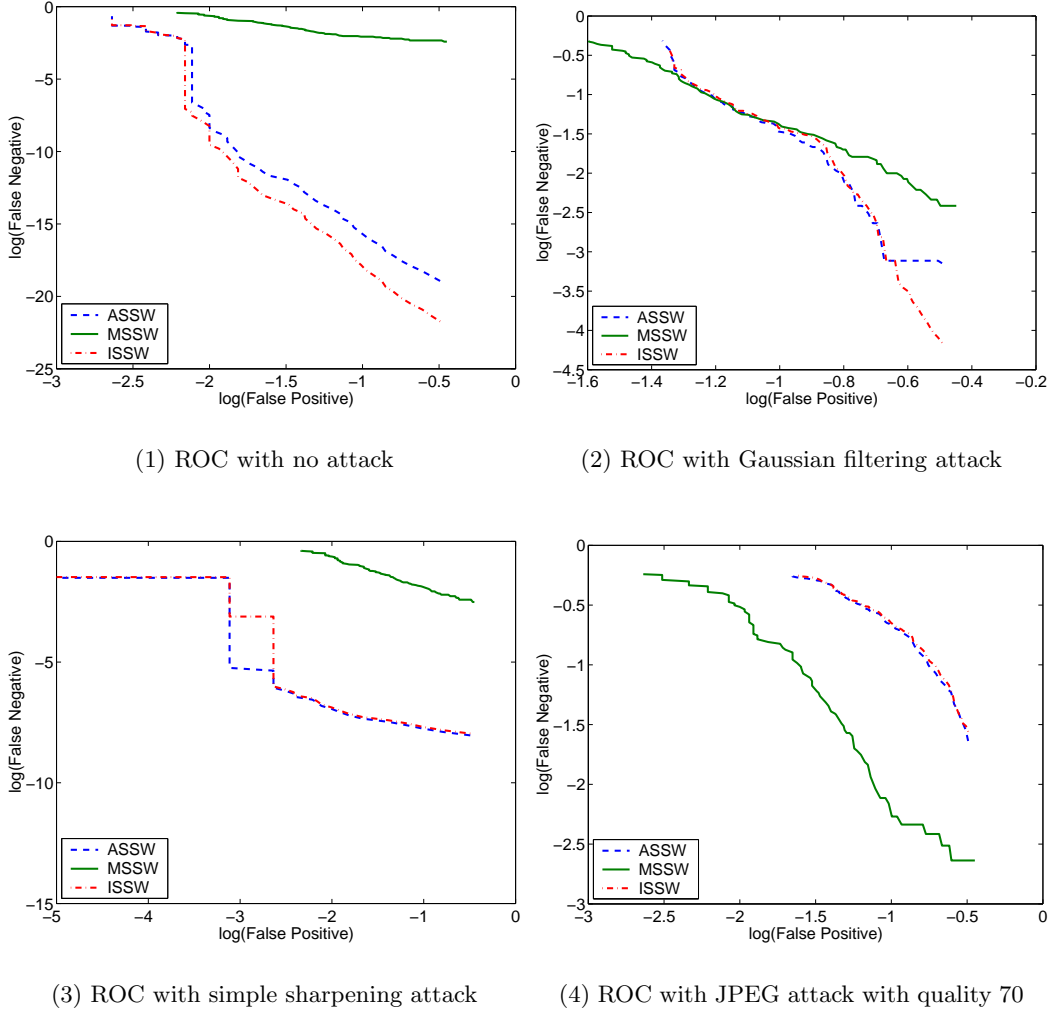
Table 4 shows that we can compare different watermarking algorithms using the Taguchi loss function, even in cases where a watermark algorithm fails to embed a watermark as shown in the MSSW average PSNR result. Table 5 shows (BER', BER<sub>average</sub>) pairs for different attacks. It shows that ISSW and ASSW produces similar results for the three attacks. When there is no attack ISSW is better than ASSW in terms of bit error rate. It also shows that the sharpening attack lowered the bit error rate for ASSW and MSSW compared to no attack. Figure 6 shows that ISSW has better detection error characteristics than ASSW for no attacks. This is due to the host signal cancellation properties of ISSW. As shown in Figure 6, MSSW seems to be better than other algorithms in terms of false positive probability and false negative probability for attacks that preserve the low frequency components. This is due to the fact that the test images we use has energy concentrated in the low frequency components and MSSW embeds the watermark in the significant components of the images. For sharpening attack, although it lowered the bit error rate for ASSW and MSSW compared to no attack, it increased the detection error probabilities for ASSW and ISSW as shown in Figure 6.

## 4. CONCLUSION AND FUTURE WORK

We described our watermark evaluation architecture in this paper. We also evaluated three watermark algorithms in terms of fidelity, bit error rate, false positive probability and false negative probability. We used the Taguchi loss function to define the fidelity and robustness requirements.

Name	No Attack	Gaussian Filtering	Sharpening	JPEG
ASSW	(1.62e-2, 1.68e-3)	(6.01e-2, 1.52e-2)	(1.41e-2, 4.8e-4)	(7.58e-2, 2.28e-2)
MSSW	(5.11e-2, 1.41e-2)	(9.51e-2, 4.11e-2)	(3.66e-2, 5.24e-3)	(6.35e-2, 1.83e-2)
ISSW	(3.87e-3, 1.44e-4)	(6.00e-2, 1.52e-2)	(1.41e-2, 4.8e-4)	(7.58e-2, 2.28e-2)

**Table 5.**  $(BER', BER_{average})$  pair for different attacks



**Figure 6.** ROC for various attacks

For future work, we need to add features to *WET* to produce the results shown in the experiments. Furthermore, we need to improve the Front End of *WET* to support evaluation procedures similar to StirMark 4.0 [15]. In StirMark 4.0, the watermarking algorithm is evaluated using an evaluation profile that describes which images, fidelities and attacks are used in the evaluation procedure. An evaluation profile is necessary to evaluate watermarks in a fair manner. We only evaluated “robust” watermark in this paper. Authentication watermarks have different requirements. We need to define the requirements and find a way to measure the parameters included in the requirements. We obtained the ROC by assuming the detection statistics for true positive and true negative are independent and they are normally distributed. We also made a judgment on the ROC by just looking at it. A better method would be to develop a quantitative ROC analysis technique for watermarking based on the

techniques already developed for medical research [40].

## REFERENCES

1. R. B. Wolfgang, C. I. Podilchuk, and E. J. Delp, "Perceptual watermarks for digital images and video," *Proceedings of the IEEE*, Vol. 87, No. 7, pp. 1108–1126, Jul. 1999.
2. I. Cox, J. Kilian, T. Leighton, and T. Shamon, "Secure spread spectrum watermarking for multimedia," *IEEE Transactions on Image Processing*, Vol. 6, No. 12, pp. 1673–1687, 1997.
3. J. Fridrich and M. Goljan, "Comparing robustness of watermarking techniques," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, Vol. 3657, San Jose, CA, Jan. 1999, pp. 214–225.
4. C. I. Podilchuk and E. J. Delp, "Digital watermarking: Algorithms and applications," *IEEE Signal Processing Magazine*, Vol. 18, No. 4, pp. 33–46, Jul. 2001.
5. I. Cox, M. Miller, and J. Bloom, *Digital Watermarking*. Morgan Kaufmann, 2001.
6. E. J. Delp, "Is your document safe: An overview of document and print security," *NIP18: International Conference on Digital Printing Technologies*, San Diego, CA, Sep. 29–Oct. 4 2002, presented at.
7. A. M. Eskicioglu and E. J. Delp, "An overview of multimedia content protection in consumer electronics devices," *Signal Processing: Image Communication*, Vol. 16, pp. 681–699, 2001.
8. A. M. Eskicioglu, J. Town, and E. J. Delp, "Security of digital entertainment content from creation to consumption," *Signal Processing: Image Communication*, Vol. 18, pp. 237–262, 2003.
9. B. Schneier, *Applied Cryptography*, 2nd ed. New York: John Wiley and Sons, 1996.
10. I. J. Cox and M. L. Miller, "The first 50 years of electronic watermarking," *EURASIP Journal of Applied Signal Processing*, Vol. 2002, No. 2, pp. 126–132, 2002.
11. I. J. Cox, M. L. Miller, and J. A. Bloom, "Watermarking applications and their properties," *International Conference on Information Technology: Coding and Computing*, 2000, pp. 6–10.
12. F. Mintzer, G. Braudaway, and M. Yeung, "Effective and ineffective digital watermarks," *Proceedings of the IEEE International Conference on Image Processing*, Santa Barbara, CA, Oct. 1997, pp. 9–12.
13. B. Macq, J. Dittmann, and E. J. Delp, "Benchmarking of image watermarking algorithms for digital rights management," *Proceedings of the IEEE*, 2004, to appear.
14. M. Kutter and F. A. P. Petitcolas, "A fair benchmark for image watermarking systems," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, P. W. Wong and E. J. Delp, Eds., Vol. 3657, San Jose, CA, Jan. 1999, pp. 226–239.
15. F. A. P. Petitcolas, M. Steinebach, F. Raynal, J. Dittman, C. Fontaine, and N. Fates, "A public automated web-based evaluation service for watermarking schemes: StirMark benchmark," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, P. W. Wong and E. J. Delp, Eds., Vol. 4314, San Jose, CA, Jan. 2001.
16. F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Attacks on copyright marking systems," *Information Hiding, Second International Workshop*, D. Aucsmith, Ed. Portland, OR: Springer-Verlag, Apr. 1998, pp. 219–239.
17. F. A. P. Petitcolas, "Watermarking schemes evaluation," *IEEE Signal Processing Magazine*, Vol. 17, No. 5, pp. 58–64, Sep. 2000.
18. J. C. Vorbruggen and F. Cayre, "The Certimark benchmark: architecture and future perspectives," *IEEE International Conference on Multimedia and Expo*, Vol. 2, Lausanne, Switzerland, Aug. 2002, pp. 485–488.
19. S. Pereira, S. Voloshynovskiy, M. Madueño, S. Marchand-Maillet, and T. Pun, "Second generation benchmarking and application oriented evaluation," *Information Hiding Workshop*, Pittsburgh, PA, Apr. 2001.
20. V. Solachidis, A. Tefas, N. Nikolaidis, S. Tsekeridou, A. Nikolaidis, and P. Pitas, "A benchmarking protocol for watermarking methods," *Proceedings of the IEEE International Conference on Image Processing*, Vol. 3, Thessaloniki, Greece, Oct. 2001, pp. 1023–1026.
21. "<http://poseidon.csd.auth.gr/optimark/>."
22. "<http://www.mysql.com>."
23. "<http://www.gimp.org>."

24. D. Taubman and M. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards, and Practice*. Kluwer Academic Publishers, 2002.
25. R. Dugad, K. Ratakonda, and N. Ahuja, "A new wavelet-based scheme for watermarking images," *Proceedings of the IEEE International Conference on Image Processing*, Chicago, IL, Oct. 1998.
26. E. T. Lin, C. I. Podilchuk, and E. J. Delp, "Detection of image alterations using semi-fragile watermarks," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, P. W. Wong and E. J. Delp, Eds., Vol. 3971, Jan. 2000, pp. 152–163.
27. B. Chen and G. W. Wornell, "Quantization index modulation methods for digital watermarking and information embedding of multimedia," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, Vol. 27, No. 1-2, pp. 7–33, Feb. 2001.
28. —, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Transactions on Information Theory*, Vol. 47, No. 4, pp. 1423–1443, May 2001.
29. Y. Liu, B. Ni, X. Feng, and E. J. Delp, "Lot-based adaptive image watermarking," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, P. W. Wong and E. J. Delp, Eds., Vol. 5306, San Jose, CA, Jan. 2004.
30. E. T. Lin and E. J. Delp, "Locktography: Technical description," Internal Purdue memorandum.
31. "<http://www.petitcolas.net/fabien/watermarking/stirmark31/>."
32. M. Kutter and F. A. P. Petitcolas, "Fair evaluation methods for image watermarking systems," *Journal of Electronic Imaging*, Vol. 9, No. 4, pp. 445–455, Oct. 2000.
33. E. E. Lewis, *Introduction to Reliability Engineering*. John Wiley and Sons, Inc., 1996.
34. T. F. Rodriguez and D. A. Cushman, "Optimized selection of benchmark test parameters for image watermark algorithms based on Taguchi methods and corresponding influence on design decisions for real-world applications," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, P. W. Wong and E. J. Delp, Eds., Vol. 5020, San Jose, CA, Jan. 2003.
35. S. Lee and V. K. Madiseti, "Parameter optimization of robust low-bit-rate video coders," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, No. 6, pp. 849–855, Sep. 1999.
36. *Deliverables D21: Watermarking applications and requirements for benchmarking*. The Certimark Consortium, Oct. 2000, Available from <http://www.certimark.org>.
37. M. Barni, C. I. Podilchuk, F. Bartolini, and E. J. Delp, "Watermark embedding: Hiding a signal within a cover image," *IEEE Communications Magazine*, Vol. 39, No. 8, pp. 102–108, Aug. 2001.
38. A. Piva, M. Barni, F. Bartolini, and V. Cappellini, "DCT-based watermark recovering without restoring to the uncorrupted original image," *Proceedings of the IEEE International Conference on Image Processing*, Vol. 3, 1997, pp. 520–523.
39. H. S. Malvar and D. A. F. Florencio, "Improved spread spectrum: A new modulation technique for robust watermarking," *IEEE Transactions on Signal Processing*, Vol. 51, No. 4, pp. 898–905, Apr. 2003.
40. J. Tilbury, W. V. Eetvelt, J. Garibaldi, J. Curnow, and E. Ifeachor, "Receiver operating characteristic analysis for intelligent medical systems—a new approach for finding confidence intervals," *IEEE Transactions on Biomedical Engineering*, Vol. 47, No. 7, pp. 952–963, Jul. 2000.