OR-Benchmark: An Open and Reconfigurable Digital Watermarking Benchmarking Framework

Hui Wang[1][0000−0001−6620−3319], Anthony T.S. Ho[2,3,4][0000−0001−9208−1255], and Shujun Li[5][0000−0001−5628−7328]

[1] Hangzhou Dianzi University, China
h.wang@hdu.edu.cn
[2] Tianjin University of Science and Technology, China
[3] Wuhan University of Technology, China
[4] University of Surrey, UK
a.ho@surrey.ac.uk
[5] University of Kent, UK
S.J.Li@kent.ac.uk

Abstract. Benchmarking digital watermarking algorithms is not an easy task because different applications of digital watermarking often have very different sets of requirements and trade-offs between conflicting requirements. While there have been some general-purpose digital watermarking benchmarking systems available, they normally do not support complicated benchmarking tasks and cannot be easily reconfigured to work with different watermarking algorithms and testing conditions. In this paper, we propose OR-Benchmark, an open and highly reconfigurable general-purpose digital watermarking benchmarking framework, which has the following two key features: 1) all the interfaces are public and general enough to support all watermarking applications and benchmarking tasks we can think of; 2) end users can easily extend the functionalities and freely configure what watermarking algorithms are tested, what system components are used, how the benchmarking process runs, and what results should be produced. We implemented a prototype of this framework as a MATLAB software package and demonstrate how it can be used in three typical use cases. The first two use cases show how easily we can define benchmarking profiles for some robust image watermarking algorithms. The third use case shows how OR-Benchmark can be configured to benchmark some image watermarking algorithms for content authentication and self-restoration, which cannot be easily supported by other digital watermarking benchmarking systems.

Keywords: Digital Watermarking · Benchmarking · Performance Evaluation · Reconfiguration · Content Authentication · Self-restoration

1 Introduction

Digital watermarking, a branch of information hiding, involves research on the process of embedding digital information (watermark) within a cover signal to achieve different (often security-related) functionalities related to the cover signal and/or its consumption by end users [1]. Since the late 1980s a large number of digital watermarking algorithms have been proposed for many applications with different system requirements mostly for protecting different types of multimedia data such as still images, audio, video, 3-D models [5,11,24,27,31]. In copyright protection applications, robust watermarking schemes [11,13,32] are desired to embed copyright information as a watermark in the digital media that can be hard to remove. In some multimedia content authentication applications, fragile watermarking schemes [3,4] and semi-fragile watermarking schemes [16,18] are desired because of the need to capture the content changes. Besides, other applications of digital watermarking include transaction tracking, usage control, self-restoration, broadcast monitoring, *etc.*

There are a number of properties associated with a digital watermarking algorithm depending on different application requirements. It is well accepted that imperceptibility and robustness are the two most important but normally conflicting requirements. Besides, embedding capacity/efficiency, security (*i.e.*, the ability to resist malicious attacks) and computational complexity are also important properties for most digital watermarking systems. However, the importance of each property is different in different applications. Some properties also overlap with each other.

As in many other multimedia systems, a general-purpose, flexible and fair benchmarking environment with appropriate test criteria is of particular importance for performance evaluation and comparison of digital watermarking algorithms.With a properly-designed benchmarking system, end users and researchers can conduct performance evaluation of a given algorithm and compare performance of multiple algorithms more easily and fairly to know more about pros and cons of different algorithms and to draw more insights about how to further improve existing algorithms. Since the 1990s, a number of digital watermarking benchmarking systems have been proposed [15,19,22,23,25,28].

Generally speaking, benchmarking performance of digital watermarking algorithms is not an easy task because different digital watermarking applications often have very different sets of requirements and trade-offs among conflicting requirements. When multiple digital watermarking algorithms with changeable parameters have to be evaluated against each other, the benchmarking task becomes more complicated. Furthermore, for systems involving more than one type of watermarks, *e.g.*, content authentication watermarking with the capability of self-restoration, the complexity of the benchmarking task becomes even higher. While there have been some general-purpose digital watermarking benchmarking systems available, most of them can be applied to only certain digital watermarking systems for a limited range of applications. In addition, existing benchmarking systems normally do not support complicated benchmarking tasks and cannot be easily reconfigured to work with different algorithms and testing conditions. It is thus still a challenge to design an efficient and general-purpose benchmarking system that can be used to benchmark different digital watermarking algorithms.

In this paper, we propose OR-Benchmark, an open and highly reconfigurable general-purpose framework for benchmarking digital watermarking algorithms, which is designed to meet the needs of different digital watermarking algorithms and various benchmarking tasks. Its main features include:

– The framework has *open* interfaces for (re)configuring different parts of the benchmarking system and addition of new modules. The framework itself is implementation-independent, but we implemented a prototype in MATLAB (to be released once the paper is published).
– The framework defines a unified procedure of benchmarking different digital watermarking algorithms against different attacks and using different performance indicators to make the comparison more systematic.
– The framework is designed to be independent of the media type, so it can be applied to digital watermarking algorithms for different media types although in this paper we will only demonstrate it for image watermarking.

The rest of the paper is organized as follows. In Section 2, related work on digital watermarking benchmarking is introduced. Section 3 gives a detailed description of our proposed benchmarking framework, including our abstract modelling of digital watermarking systems, important evaluation criteria, the proposed OR-Benchmark framework, and comparison with other existing digital watermarking benchmarking systems. Next, in Section 4, we describe how we implemented a first prototype of OR-Benchmark in MATLAB with three use cases for different application scenarios. The paper is concluded by Section 5 with future work.

2 Related Work

While there have been a substantial number of digital watermarking algorithms proposed for different applications and usage scenarios, there are relatively less research on digital watermarking benchmarking especially general-purpose frameworks capable of handling multiple applications with different sets of requirements. Most existing digital watermarking benchmarking systems focus on some well-defined sub-areas among which image watermarking received the most attention. In this section, we briefly overview some representative work.

2.1 StirMark

StirMark, one of the earliest and the most well-known digital watermarking benchmarking systems, was firstly proposed by Petitcolas *et al.* in 1998 [23] as a generic tool for benchmarking digital image watermarking algorithms against various attacks, which was later contributed by more researchers in 2001 [20] to become a more general framework for benchmarking digital watermarking algorithms. Subsequently, several enhanced versions of StirMark were developed to include more attacks and cover audio watermarking [9,26]. The main aim of StirMark is to develop a fully automated evaluation service, which could encapsulate different performance evaluation indicators and allow continuous development of new attacks to be integrated into the whole system. Since StirMark is among the most widely-used benchmarking systems by the digital watermarking community, we discuss it in greater detail below.

Interfaces To use StirMark for benchmarking a digital watermarking algorithm, the user is required to supply three functions, *Embed* and *Extract* functions, and one *GetSchemeInfo* function which provides meta-information about the algorithm such as the name, version, author(s), the maximum byte-length of the embedded message, the maximum bite-length of the stego-key, *etc.*

Evaluation Criteria The main performance indicators of a digital watermarking algorithm StirMark can evaluate include imperceptibility, capacity, robustness to attacks, false alarm rate and execution speed.

Benchmarking Framework StirMark as a framework contains six main components including the marking scheme library, test library, evaluation profile library, quality metrics library, multimedia database and results database. According to different application requirements, there are different evaluation profiles, each of which is composed by a list of tests or attacks to be applied and a list of multimedia signals required for the test. The end user is required to add the watermarking algorithm under testing (in the form of three C++ functions including *GetSchemeInfo*, *Embed* and *Extract*) to the marking scheme library. The end users also selects evaluation profiles written as INI files with limited static structure. And it is not available to extend the structure of the evaluation profiles without StirMark source code changing. According to the information provided by the end user, StirMark runs the defined benchmarking process automatically by using its multimedia database, the tests (attacks) library and the quality metrics library. The results are stored in a database (an SQL server as stated in [20] and simple files as in actual implementations).

Implementation StirMark was originally developed by Kuhn in 1997 [7] as a generic software tool for simple robustness testing of image watermarking algorithms. It simulates many common attacks to image watermarking algorithms including random bilinear geometric distortions to de-synchronize watermarking algorithms. Subsequently, StirMark was extended by Petitcolas and other researchers to support more tests and attacks [8,23]. Later on some more development work took place, including a set of tests for audio watermarking developed by Steinebach *et al.* [26] and by Lang and Dittmann [9]. There were also efforts of making StirMark a public automated web-based evaluation service made by Petitcolas *et al.* [20] which led to the 4.0 version of Stirmark [21].

Limitations Although StirMark has been widely used as a tool for robustness and security evaluation of digital watermarking algorithms, we feel it has the following limitations.

The modelling and interface do not cover all digital watermarking algorithms. For instance, there are only two types of output for watermark detection (*i.e.*, the *Extract* function): the extracted watermark and a certainty to show the probability whether the watermark is detected correctly. This is insufficient for watermarking algorithms for tamper localization and/or image restoration.

StirMark is reconfigurable but the level is limited. Reconfiguring StirMark for a digital watermarking algorithm can be done by defining the input and output arguments according to one of the six pre-defined types of algorithms, but adding new parameters and extending existing parameter settings will require changing the source code of the StirMark implementation (in C++). For example, the *strength* parameter in StirMark is set to be a single floating-point number with many hard constraints (*e.g.*, minimum and maximum values are linked to specific PSNR values), but for digital watermarking algorithms the *strength* could be a more complicated parameter.

Although StirMark allows adding new tests, attacks and PQA metrics, the unclear boundaries among components make it hard to do so without making changes to the source code of the StirMark implementation. Adding some new test, attack and quality metric may require a re-design of the framework, *e.g.*, if a non-PSNR PQA metric is introduced the *strength* parameter will need re-defining and many existing components need adapting to the new PQA metric.

The StirMark framework defined in [20] does not follow a clear data flow, *e.g.*, the test library does not really flow into the evaluation profile but reads data from the latter and the multimedia database.

In [20] StirMark is described to work with an SQL server to store all the evaluation results which can then be converted into web pages for reporting. However, the SQL-based web service has not been actually implemented. Instead, the latest C++ implementation of StirMark [21] produces a plain data sheet to store the evaluation results which cannot be easily converted into other formats or used to do further analysis.

2.2 Other Benchmarking Systems

Checkmark was developed by Pereira *et al.* [19] and downloadable from `http://cvml.unige.ch/ResearchProjects/Watermarking/Checkmark/` (now discontinued). Checkmark was based on StirMark with the following main changes. First of all, a number of new attacks, which take statistical properties of images and watermarks into account, are incorporated into Checkmark.Secondly, weighted PSNR and Watson's metric are used as new metrics for evaluating image quality instead of just PSNR. Thirdly, evaluation results are represented in a flexible XML format and can be automatically converted into HTML web pages.Despite the changes to StirMark, the reconfigurability of Checkmark remains relatively low so normally users have to make changes to Checkmark's source code.

Optimark [25] is a benchmarking software package for image watermarking algorithms downloadable at `http://poseidon.csd.auth.gr/optimark/`, providing a graphical user interface (GUI) developed using C/C++. To use Optimark for benchmarking a digital watermarking algorithm, the user can choose a set of test images, define different watermark embedding keys and watermark messages for multiple trials of the watermarking detector and decoder, and select a set of attacks among 14 types of attacks and attack combinations. It allows evaluation of several statistical characteristics of an image watermarking algorithm, including Receiver Operating Characteristic (ROC) curves as watermark detection performance metrics.

Certimark is the outcome of an EU-funded research project (`http://www.certimark.org/`, lasting from 2000 to 2002). The objectives of Certimark are to design a benchmarking suite which permits users to assess the appropriateness and to set application scenarios for their needs, and to set up a standard certification process, for watermarking technologies [28].Although the reconfigurability level of Certimark is higher than earlier systems, Certimark seems to have been discontinued and there is no source code publicly available.

Watermark Evaluation Testbed (WET) [2, 6] is a web-based system developed by researchers from the Purdue University to evaluate the performance of image watermarking algorithms. WET consists of three major components: front end, algorithm modules, and image database. To achieve the goal of extensibility, the GNU Image Processing Program (GIMP) is used because it support plug-ins and extensions. Some watermarking algorithms, StirMark 4.0 and some evaluation metrics were implemented as GIMP plug-ins to be part of WET's algorithm modules. The end users can select some images, one or more watermarking algorithms, attacks, and specify needed parameters via a web interface of the front end. The evaluation results can be shown as ROC curves. Similar to other systems, WET has a limited reconfigurability. In addition, its source code is not publicly available.

OpenWatermark [14, 17] is a web-based system for benchmarking digital watermarking algorithms. It is composed of three parts: 1) a web server and a remote method invocation (RMI) client for users to submit their benchmarking requests with specifications of the benchmarked algorithms, 2) a cluster of RMI benchmark servers automating the benchmarking process, and 3) a SQL database sorting all data used in the benchmarking process and results produced by the benchmark servers.OpenWatermark allows benchmarked algorithms to be submitted as Windows/Linux executables or MATLAB/Python scripts and all its components were developed in Java, so it has some reconfigurability. However, to support more features such as benchmarking profiles and other media types its source code has to be modified.OpenWatermark implementation was available to registered members at its website `http://www.openwatermark.org/` which is currently unaccessible.

Mesh Benchmark [30] was proposed for 3D mesh watermarking. It contains three different components: a data set, a software tool and two evaluation protocols.As a benchmarking system focusing on 3D mesh watermarking only, it considers only the payload, distortion and robustness for performance evaluation. Besides, the evaluation protocols are defined with fixed steps and thresholds so the reconfigurability of the mesh benchmark is low.

3 Proposed OR-Benchmark Framework

In this section, our proposed framework OR-Benchmark will be introduced in details. Firstly, we discuss general modelling of digital watermarking systems used in Sec. 3.1. Then the evaluation criteria considered are discussed in Sec. 3.2.After that, the architecture of the OR-benchmark framework and the open interfaces for end users are explained in details in Secs. 3.3 and 3.4, respectively.

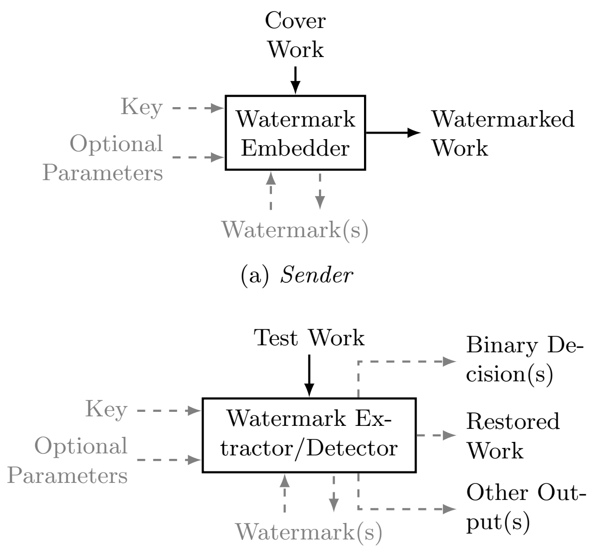
3.1 Modelling of Watermarking Systems

Following the community's common understanding, OR-benchmark models a digital watermarking system as two separate processes: the *Sender* which embeds one or more watermarks into a given cover work to generate a watermarked work; the *Receiver* which extracts and/or detects one or more watermarks that may have been embedded in a received test work. Compared with Stirmark, the modelling of watermarking systems in OR-benchmark aims to be suitable for any kinds of watermarking schemes in different application scenarios including some unsupported by Stirmark such as self-restoration watermarking. We define the sender and the receiver to take at least one input (the cover and test work, respectively) and to produce one or more outputs. There are more optional inputs and outputs (some are system parameters and some are user-defined ones). Therefore, the users can easily reconfigure the sender/receiver's input/output setting according to the watermarking application scenario benchmarked.

The general models of the watermark embedding and extraction/detection processes are shown in Fig. 1. As shown in Fig. 1(a), the *Sender* will always have the cover work as the input and the watermarked work as the output. There are three groups of optional inputs including the watermark(s) to be embedded, the embedding key, and other optional parameters controlling the embedding process. Note that the watermark(s) in the embedding process can be either an input (if supplied by the user) or an output (if generated by the *Sender* automatically), which can be further used for performance evaluation purposes. As shown in Fig. 1(b), the *Receiver* takes at least one input (a test work) and possibly some other inputs and parameters to produce one or more outputs including one or more extracted watermarks, one or more binary decisions (if some given watermark(s) is/are detected), a restored work (if the watermarking algorithm supports self-restoration), and other outputs, *e.g.*, the confidence level and error rates. We model the inputs and outputs of the *Sender* and the *Receiver* this way to cover the full range of digital watermarking algorithms and applications.

3.2 Performance Evaluation Criteria

In OR-benchmark performance evaluation criteria (*i.e.*, indicators) are organized into two categories: 1) built-in indicators that can be selected by users directly; 2) user-defined indicators that are supported indirectly by generating a comprehensive set of raw results for users to further processing. In this section, the commonly required performance indicators are further discussed.



(a) *Sender*

(b) *Receiver*

Fig. 1: Modelling of the *Sender* and the *Receiver* in OR-Benchmark. Dashed lines denote optional input/output.

Similar to StirMark, properties that designers and users of digital watermarking algorithms may wish to evaluate include imperceptibility or perceptual quality of the watermarked work (which is one aspect of security), embedding capacity, robustness to benign processing and attacks, resistance to malicious processing and attacks (which is another aspect of security), false positive/negative rates, and the speed of execution of both the sender and the receiver. Since these common criteria have been well studied in related work, here we focus on two other important properties for content authentication and self-restoration watermarking algorithms.

Authentication Accuracy For content authentication watermarking, there are two basic metrics to measure the authentication accuracy of the detection process: the false positive (FP) rate indicating the level of errors for areas reported as “tampered”, and the false negative (FN) rate indicating the level of errors for areas reported as “untampered”. Many other performance metrics can be derived from the FP and FN rates, *e.g.*, the average authentication rate and the area under the ROC curve. OR-Benchmark supports the two main metrics and also provides needed raw data in the benchmarking results to allow users to define more metrics that cannot be derived directly from the FP and FN rates.

Perceptual Quality of Recovered Work For self-restoration watermarking algorithms (which require the use of content authentication watermarks as a pre-

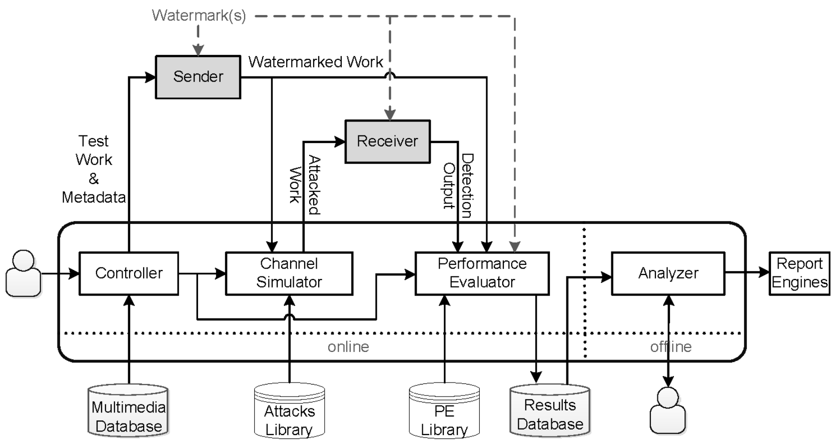


Fig. 2: The architecture of the OR-Benchmark framework.

requisite), a key performance indicator is the perceptual quality of the recovered work. In OR-Benchmark, some commonly used image quality assessment (IQA) metrics such as PSNR and SSIM are incorporated but users can add their own metrics (for any media formats not limited to digital images) easily via the open interface discussed in Sec. 3.4.

3.3 Our Benchmarking Framework

In this subsection, we introduce the overall architecture of OR-Benchmark in details. Figure 2 gives a schematic overview of the framework, which can be split into two parts: an *Online Benchmarker* takes input from the user and automates the benchmarking process to generate results for further analysis, and an *Offline Analyzer* allowing the user to conduct user-specific tasks (*e.g.*, statistics and visualization) based on the (raw) results produced by the *Online Benchmarker*. The *Offline Analyzer* can be equipped by one or more *Report Engines* to produce more user-friendly reports of benchmarking tasks. The *Report Engines* may also access the results from the *Online Benchmarker* without passing the *Offline Analyzer* (in that case the *Offline Analyzer* can be seen as a simple data forwarder).

The *Online Benchmarker* contains three groups of components: 1) the user-provided components – the *Sender* and the *Receiver* provided by the user as the subject of benchmarking, 2) a *Multimedia Database* holding the test media, an *Attacks Library* and a *PE (Performance Evaluation) library* providing attacks and performance evaluation algorithms, respectively, and 3) the core benchmarker composed of a central *Controller*, a *Channel Simulator* enabling incorporation of different types of attacks and processing on a watermark work, and a *Performance Evaluator* which produces results to sore in a *Results Database* as the output of the whole benchmarking process. The central *Controller* interacts with the user to define the benchmarking profile, and with other components of the online benchmarker to automatically execute the profile. A benchmarking profile allows automatic testing of parameter(s) of the same digital watermarking algorithm, multiple attacks, multiple PE algorithms and multiple performance indicators. The *Controller* can also automatically determine default settings based on information given by the user to reduce the burden of defining the benchmarking profile.

3.4 Open Interfaces

OR-Benchmark is designed to have open interfaces so that users can easily (re)configure and extend the framework and define different benchmarking tasks easily. There are mainly the following interfaces as shown in Fig. 2.

The interfaces between the Sender/Receiver and the core benchmarker allow users to define digital watermarking algorithms for benchmarking. Following the general models of the *Sender* and the *Receiver* discussed in Sec. 3.1, the interfaces are materialized as the input and output interfaces of two functional units:
`Sender: (Original Cover Work, [Watermark(s)], [Key], [...])` → `(Watermarked Work, [Watermark(s)]);`
`Receiver: (Test Work, [Watermark(s)], [Key], [...])` → `([Watermark(s)], [Decision], [Restored Work], [...]),`
where arguments in the square brackets are optional and "..." denotes more optional (user-defined) arguments. A proper mechanism is required to inform the *Controller* about valid values each input argument can take and other meta information (*e.g.*, the display name of each argument), in order to create benchmarking profiles for enumerating all values of interest for any input argument. Such mechanisms can include a graphical user interface (GUI) and a machine-readable textual specification (*e.g.*, an XML schema) for defining a set of sample values for any given argument.

The interface between the Multimedia Database and the core benchmarker allows users to reconfigure and extend the *Multimedia Database*. This can be achieved by an agreed structure of the *Multimedia Database* such as a hierarchy structure of folders and files or using a human-readable configuration file (such as XML) to allow the system and end users to find test multimedia works. Note that OR-Benchmark can support any media types so the *Multimedia Database* can be a mixture of different types of media files.

The interface between the Attacks Library and the core benchmarker allows users to reconfigure and extend the *Attacks Library* used by the *Channel Simualtor*. As discussed in Sec. 3.1, an attack in the *Attacks Library* is a simple functional unit as follows: `Attack: (Input Work, [...])` → `(Output Work)`. Again, a mechanism is needed to convey meta information about any optional input arguments. Compared with Stirmark, where one attack test can only contain a single one with relevant values of parameter setting, the configuration of the combined attacks for once test with the values setting for more than one arguments of the combined attacks' functions is allowed by the interface of our benchmarking system.

The interface between the PE Library and the core benchmarker allows users to reconfigure and extend the *PE Library* used by the *Performance Evaluator*. There are different types of PE algorithms depending on the performance indicators used, so there are different input and output interfaces. An important class of PE algorithms are perceptual quality assessment (PQA) algorithms which can be defined as follows: `PQA: (Work1, Work2, [...])` → `(Metric)`, where the output is a numeric rating of the perceptual quality. Again, optional input arguments are used to define parameters of some PQA algorithms. While PQA algorithms are generally objective ones based on automated computer programs, OR-Benchmark's interface allows a visual quality assessment (VQA) algorithm to interact with human raters (*e.g.*, those recruited from crowdsourcing websites) to return subjective quality ratings since the user interface can be wrapped inside the PQA function thus transparent to end users of OR-Benchmark.

System search paths can be set up for all the above interfaces so that the *Controller* and other components of the core benchmarker can automatically discover candidate algorithms and test multimedia works. Each path can be a local file path or a URL representing a web address.

The interface between the core benchmarker and the Results Database allows users to reconfigure and extend the format of the results used by the *Offline Analyzer* and *Report Engines*. This is achieved by a machine-readable configuration file indicating the format of the results.

The interface between the user and the Controller allows creation of benchmarking profiles. Core elements of a benchmarking profile include digital watermarking algorithm(s) tested and candidate values of input parameters, test multimedia works, selected attacks, selected PE algorithms, and format of the results. This can be implemented as a graphical user interface (GUI) and/or a human-readable configurable file.

The user interfaces of the Offline Analyzer and Report Engines allow users to investigate the raw results recorded in the *Results Database* in an interactive way and to produce more user-friendly reports. The interface for the *Offline Analyzer* can be implemented as a GUI, but the *Report Engines* could be standalone tools which can be invoked from the *Offline Analyzer*'s GUI. The format of the produced reports can be defined using a human-readable configurable file and be represented in a more user-friendly way, *e.g.*, as a web page.

4 Case Studies

In this section, we demonstrate how our implemented OR-Benchmark prototype (in MATLAB) was used for three case studies. The first two cases are about two main applications of robust watermarking. For the two cases we explain how OR-Benchmark was used to benchmark for a given robust watermarking scheme without giving experimental results since the configurations and expected results are straightforward. The third case is about digital watermarking algorithms for content authentication and self-restoration. Such algorithms are among the most complicated ones with two types of watermarks per block of the cover work and are not supported by other benchmarking systems. For this case we will give details on how we used OR-Benchmark to conduct a full benchmarking task involving three different watermarking algorithms.

4.1 Case 1: Copyright Protection

In this case study, we report a benchmarking task on a blind robust digital watermarking scheme used for copyright protection purposes, which needs a given copyright declaration as the watermark for the *Sender* but not for the *Receiver*. For this case the benchmarking task was set up in the OR-Benchmark prototype as follows:

- Set system paths for the target digital watermarking algorithms, the *Multimedia Database*, the *Attacks Library*, the *PE Library* and the *Result Database*.
- For the test images, we collected 100 8-bit gray-scale images of size 256 × 256, 384×256 and 512×512, which were added to a sub-folder of the folder holding the *Multimedia Database*.
- The *Sender* and *Receiver* functions were implemented as MATLAB functions with the following interface:
  `Sender: (Cover Work, Watermark, Key) → (Watermarked Work)`
  `Receiver: (Test Work, Key) → (Watermark)`
  Both functions were added to the folder holding target digital watermarking algorithms. The benchmarking profile was set to select a number of predefined copyright claims and random keys as input parameters of *Sender* (and the keys for *Receiver* as well).
- A list of attacks was defined for the *Channel Simulator* to create watermarked images, including both malicious attacks for watermark removal and some benign image processing operations. All the attacks were implemented as separate MATLAB functions and were added to the folder holding the *Attacks Library*.
- A list of PE algorithms was created, which includes the imperceptibility property (*i.e.*, visual quality of watermarked images) in terms of PSNR and SSIM, the watermark detection accuracy in terms of correlation coefficient (CC) and bit error rate (BER), and the run-time performance in term of the processing times of the *Sender* and the *Receiver* functions. Each performance indicator was implemented as one MATLAB function which was added to the folder holding the *PE Library*.

After setting up the benchmarking profile and preparing all files needed, the online benchmarker was run to execute the profile automatically. All the benchmarking results were recorded in a MAT file and saved into the *Result Database*. A simple *Offline Analyzer* was produced to visualise results.

4.2 Case 2: Content Integrity Verification

In this case study, we report a benchmarking task on an informed watermarking scheme used to detect content integrity of digital images, which needs a given watermark at both the *Sender* and *Receiver* sides. The benchmarking profile of this case was configured and executed in a similar way as Case 1 but with the following changes:

– The *Receiver* function was implemented as a MATLAB function with the following interface:
  `Receiver: (Test Work, Watermark, Key)` $\rightarrow$ `(Decision)`
– A hypothesis test is added as a new attack, which assign the `Test Work` to the `Watermarked Work` or `Original Cover Work` according to the binary hypothesis parameter.
– In the list of PE algorithms, the metric for watermark detection accuracy was changed to false negative and false positive rates.

4.3 Case 3: Tamper Localization and Self-Restoration

In this case study we report a benchmarking task on three content authentication and self-restoration watermarking algorithms used for detecting (localizing) and restoring tampered regions in an image: Lin and Chang's scheme [12] (M1), Li *et al.*'s scheme [10] (M2) and Wang *et al.*'s scheme [29] (M3). All watermarking algorithms use two different types of watermarks for each 8×8 block of the cover image, one for tamper localization and the other for self-restoration. Such algorithms are among the most complicated watermarking algorithms and are not (well) supported by other benchmarking systems.

Benchmarking Profile We used the *Controller*'s GUI to set up the benchmarking task as follows (setup of system paths is omitted):

– The *Multimedia Database* was set up to include 100 test images representing a broad range of image types, *e.g.*, outdoor or indoor scenes images, portraits, photos of natural or man-made objects, and texture images.
– The *Sender* and *Receiver* functions were implemented as MATLAB functions with the following interface:
  `Sender: (Cover Work, Key)` $\rightarrow$ `(Watermarked Work)`
  `Receiver: (Test Work, Key)` $\rightarrow$ `(Detected Tampered Regions, Recovered Work)`
  Here, the Detected Tampered Regions is a matrix storing the binary decision of tamper detection for each block.
– To ensure a fair comparison of the three watermarking schemes, we tuned their parameters so that the average visual quality of the 100 watermarked images is roughly aligned. This was achieved by conducting three separate smaller benchmarking tasks where each watermarking scheme was benchmarked with a number of parameters to produce a set of PSNR and SSIM values, and then the parameters were determined so that all three schemes have similar average PSNR and SSIM values.

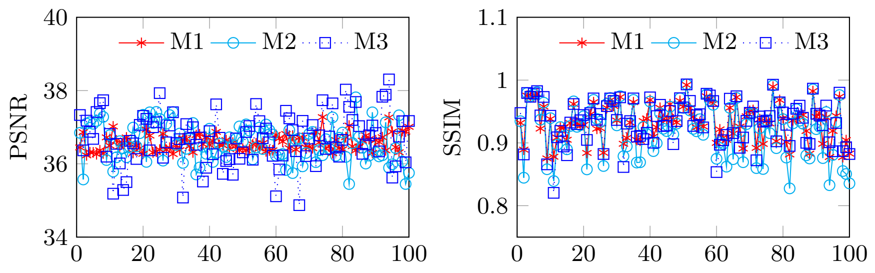


Fig. 3: The quality comparison of watermarked images produced by the three different watermarking schemes. The x-axis is the image index and the y-axis is the PSNR/SSIM value.

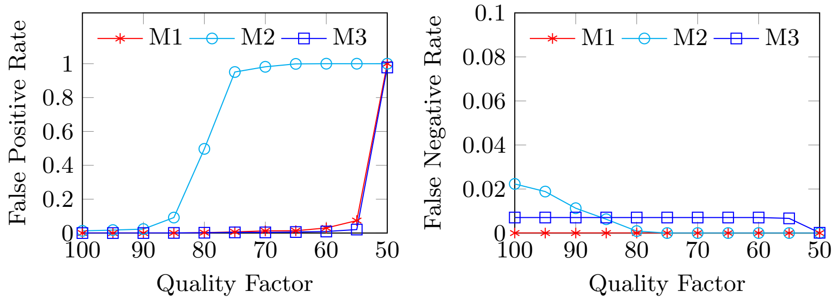
– For attacks, we chose simple "copy and paste attack", JPEG compression, additive and multiplicative Gaussian white noises as four separate attacking algorithms each of which is injected into the *Channel Simulator* to create attacked watermarked images sent to the *Receiver*. In addition to the simple "copy and paste attack" alone, we also considered combinations of the "copy and paste attack" with one of other attacks.
– For performance indicators, we used the following: PSNR and SSIM for visual quality of watermarked and recovered images, FP and FN rates for tamper detection accuracy, and processing times of the *Sender* and the *Receiver* functions for run-time performance.

The above benchmarking task was stored as a benchmarking profile which was then executed by the *Controller* to generate the results. The machine running the benchmarking task is a PC with an Intel Core 2 Duo CPU (3.16GHz) and 2GB RAM. The concurrency support of the dual-core CPU was disabled to get a more accurate estimate of the processing times.

After the results were produced by the core benchmarker, the *Offline Analyzer* was used to generate some 2-D plots for a better understanding of the performance of the three benchmarked image watermarking schemes. From the benchmarking results produced by OR-Benchmark, we were able to conclude that M3 has the best performance, followed by M1 and then M2. In the following, we show some selected benchmarking results we obtained.

Visual Quality of Watermarked Image Figure 3 shows the PSNR and SSIM values of all the 100 test images after going through each of the three digital watermark embedding processes. As mentioned above, we selected parameters of the three schemes properly so that they produce roughly equal PSNR and SSIM values for all 100 images.

Tamper Detection Accuracy To evaluate tamper detection accuracy of an image authentication watermarking scheme, attacks manipulating contents of



(a) FP against JPEG compression (b) FN against JPEG compression

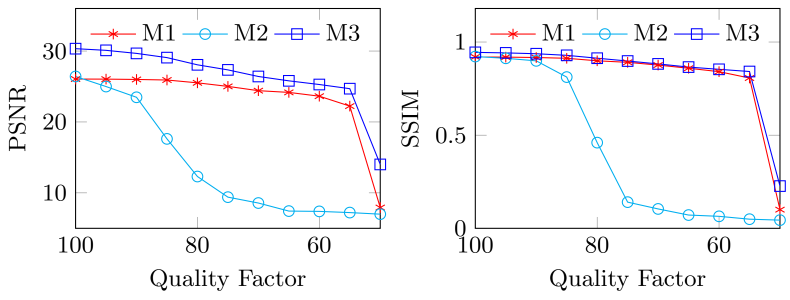
Fig. 4: Average FP and FN rates of M1, M2 and M3 w.r.t. different parameter values of JPEG compression.

watermarked images should be used at the *Receiver*. As mentioned above, we used a simple "copy and paste attack" as an example attack to manipulate 10% random-selected part of each test image after it is watermarked. The FP and FN rates are then calculated per image based on how many non-manipulated 8×8 blocks are reported as "manipulated" (false positives) and how many manipulated blocks are not detected (false negatives). The FP rates of M1 and M3 are nearly 0, and that of M2 is 1.36%. The FN rate of M1 remains close to 0, but those of M2 and M3 are 3.02% and 1.59%, respectively. It is thus clear that M1 is the best and M2 is the worst.

Visual Quality of Recovered Image Similar to the case of tamper detection accuracy, for visual quality of recovered images we also focused on the condition where the 10% "copy and paste attack" is applied without other attacks. The mean PSNR values of 100 images recovered by M1, M2 and M3 are 27.8, 28.0 and 32.3 dB, respectively, and the mean SSIM values are 0.925, 0.927 and 0.951, respectively. The results show that M3 is the best scheme with a significant margin (more than 4.4dB in PSNR and 0.023 in SSIM).

Robustness For benchmarking robustness, we combined the 10% "copy and paste attack" with one additional attack (JPEG compression, additive or multiplicative Gaussian white noises) to gauge the robustness of each digital watermarking scheme against each additional attack. The results of combining with JPEG compression are shown in Figs. 4 and 5.

Here we average the performance indicators cross all 100 images to get the average values which are then shown against the QF as parameter value of each compression to see how the strength of the attack influences the performance of each digital watermarking scheme. We can observe that M3 outperforms M2 significantly with similar or lower FP and FN rates, and higher PSNR and SSIM



(a) PSNR against JPEG compression (b) SSIM against JPEG compression

Fig. 5: Average perceptual quality of images recovered by M1, M2 and M3 w.r.t. different parameter values of JPEG compression.

values of recovered images. Between M1 and M3, we can also observe that M3 performs significantly better in terms of PSNR although just slightly for SSIM.

Processing Time Except the embedding process of M1 which took around 2.6s in average, all other processes of the three digital watermarking schemes consumed less than 1s. Considering MATLAB is much less effective than other compiled programming languages, the results suggest that all the three schemes are practical for real-world applications.

5 Conclusion and Future Work

In this paper, we present OR-Benchmark, an open and highly reconfigurable general-purpose benchmarking framework, to meet the needs of benchmarking different digital watermarking schemes. To the best of our knowledge, this is the first and the only benchmarking framework supporting all known types of digital watermarking schemes including complicated ones involving multiple types of watermarks. We implemented the framework in MATLAB, and discussed three use cases including one on authentication and self-recovery watermarking to showcase the usefulness of OR-Benchmark as a convenient and flexible tool.

Although OR-Benchmark as a general framework can easily support any media type, attacks, test multimedia datasets, and PE algorithms, our current implementation has mainly built-in functional units for digital images. The *Offline Analyzer* is also tailored towards our own needs for benchmarking some special types of digital watermarking schemes. In future we plan to add more functional units to the prototype so that users can use it without adding too many user-defined algorithms but focus on the digital watermarking schemes themselves. We also plan to release our MATLAB prototype under an open source license and call for contributions from the whole digital watermarking community to further extend the current implementation and to create implementations based on other languages. A dedicated website will be set up to host related documents and source code of our MATLAB implementation.

References

1. Cox, I., Miller, M., Bloom, J., Fridrich, J., Kalker, T.: Digital Watermarking and Steganography. Morgan Kaufmann Publishers (2007)
2. Guitart, O., Kim, H.C., III, E.J.D.: Watermark evaluation testbed. Journal of Electronic Imaging p. 041106 (2006). https://doi.org/10.1117/1.2400067
3. He, H., Chen, F., Tai, H.M., Kalker, T., Zhang, J.: Performance analysis of a block-neighborhoodbased self-recovery fragile watermarking scheme. IEEE Transactions on Information Forensics and Security **7**(1), 185–196 (2012). https://doi.org/10.1109/TIFS.2011.2162950
4. Ho, A.T.S., Zhu, X., Shen, J., Marziliano, P.: Fragile watermarking based on encoding of the zeroes of the z-transform. IEEE Transactions on Information Forensics and Security **3**(3), 567–569 (2008). https://doi.org/10.1109/TIFS.2008.926994
5. Houmansadr, A., Kiyavash, N., Borisov, N.: Non-blind watermarking of network flows. IEEE/ACM Transactions on Networking **22**(4), 1232–1244 (2014). https://doi.org/10.1109/TNET.2013.2272740
6. Kim, H.C., Lin, E.T., Guitart, O., III, E.J.D.: Further progress in watermark evaluation testbed (WET). In: Security, Steganography, and Watermarking of Multimedia Contents VII. Proc. SPIE, vol. 5681, pp. 241–251 (2005). https://doi.org/10.1117/12.593803
7. Kuhn, M.: StirMark – image-watermarking robustness test, `http://www.cl.cam.ac.uk/~mgk25/stirmark.html`
8. Kutter, M., Petitcolas, F.A.P.: A fair benchmark for image watermarking systems. In: Security and Watermarking of Multimedia Contents. Proc. SPIE, vol. 3657, pp. 226–239 (1999). https://doi.org/10.1117/12.344672
9. Lang, A.: StirMark Benchmark for Audio - SMBA, `http://omen.cs.uni-magdeburg.de/alang/smba.php`
10. Li, G., Pei, S., Chen, G., Cao, W., Wu, B.: A self-embedded watermarking scheme based on relationship function of corresponding inter-blocks DCT coefficient. In: Proc. CSCWD 2009. pp. 107–112. https://doi.org/10.1109/CSCWD.2009.4968043
11. Li, W., Xue, X., Lu, P.: Localized audio watermarking technique robust against time-scale modification. IEEE Transactions on Multimedia **8**(1), 60–69 (2006). https://doi.org/10.1109/TMM.2005.861291
12. Lin, C.Y., Chang, S.F.: Semi-fragile watermarking for authenticating JPEG visual content. In: Security and Watermarking of Multimedia Contents II. Processings of SPIE, vol. 3971, pp. 140–151 (2000). https://doi.org/10.1117/12.384968
13. Lin, C.Y., Wu, M., Bloom, J.A., Cox, I.J., Miller, M.L., Lui, Y.M.: Rotation, scale, and translation resilient watermarking for images. IEEE Transactions on Image Processing **10**(5), 767–782 (2001). https://doi.org/10.1109/83.918569
14. Lugan, S., Macq, B.: Thread-based benchmarking deployment. In: Security, Steganography and Watermarking of Multimedia Contents VI. Proc. SPIE, vol. 5306, pp. 248–255 (2004). https://doi.org/10.1117/12.538692
15. Macq, B., Dittmann, J., Delp, E.J.: Benchmarking of image watermarking algorithms for digital rights management. In: Proceedings Of the IEEE. vol. 92, pp. 971–984 (2004). https://doi.org/10.1109/JPROC.2004.827361

16. Maeno, K., Sun, Q., Chang, S.F., Suto, M.: New semi-fragile image authentication watermarking techniques using random bias and nonuniform quantization. IEEE Transactions on Multimedia **8**(1), 32–45 (2006). https://doi.org/10.1109/TMM.2005.861293
17. Michiels, B., Macq, B.: Benchmarking image watermarking algorithms with Openwatermark. In: Proc. EUSIPCO 2006
18. Ni, Z., Shi, Y.Q., Ansari, N., Su, W., Sun, Q., Lin, X.: Robust lossless image data hiding designed for semi-fragile image authentication. IEEE Transactions on Circuits and Systems for Video Technology **18**(4), 497–509 (2008). https://doi.org/10.1109/TCSVT.2008.918761
19. Pereira, S., Voloshynovskiy, S., Madueno, M., Marchand-Maillet, S., Pun, T.: Second generation benchmarking and application oriented evaluation. In: Proc. IH 2001. LNCS, vol. 2137, pp. 340–353 (2001)
20. Petitcolas, F.A.P., Steinebach, M., Raynal, F., Dittmann, J., Fontaine, C., Fates, N.: A public automated web-based evaluation service for watermarking schemes: StirMark benchmark. In: Security and Watermarking of Multimedia Contents III. Proc. SPIE, vol. 4314, pp. 575–584 (2001). https://doi.org/10.1117/12.435442
21. Petitcolas, F.: Stirmark benchmark 4.0, `http://www.cl.cam.ac.uk/~fapp2/watermarking/stirmark/`
22. Petitcolas, F.A.P.: Watermarking scheme evaluation. IEEE Signal Processing Magazine **17**(5), 58–64 (2000). https://doi.org/10.1109/79.879339
23. Petitcolas, F.A.P., Anderson, R.J., Kuhn, M.G.: Attacks on copyright marking systems. In: Proc. IH '98. LNCS, vol. 1525, pp. 218–238. https://doi.org/10.1007/3-540-49380-8_16
24. Podilchuk, C.I., Delp, E.J.: Digital watermarking: Algorithms and applications. IEEE Signal Processing Magazine **18**(4), 33–46 (2001). https://doi.org/10.1109/79.939835
25. Solachidis, V., Tefas, A., Nikolaidis, N., Tsekeridou, S., Nikolaidis, A., Pitas, I.: A benchmarking protocol for watermarking methods. In: Proc. ICIP 2001. pp. 1023–1026. https://doi.org/10.1109/ICIP.2001.958300
26. Steinebach, M., Petitcolas, F.A.P., Raynal, F., Dittmann, J., Fontaine, C., Seibel, S., Fates, N., Ferri, L.C.: StirMark benchmark: Audio watermarking attacks. In: Proc. ITCC 2001. pp. 49–54. https://doi.org/10.1109/ITCC.2001.918764
27. Stütz, T., Autrusseau, F., Uhl, A.: Non-blind structure-preserving substitution watermarking of H.264/CAVLC inter-frames. IEEE Transactions on Multimedia **16**(5), 1337–1349 (2014). https://doi.org/10.1109/TMM.2014.2310595
28. Vorbrüggen, J.C., Cayre, F.: The Certimark benchmark: Architecture and future perspectives. In: Proc. ICME 2002. pp. 485–488. https://doi.org/10.1109/ICME.2002.1035651
29. Wang, H., Ho, A.T.S., Zhao, X.: A novel fast self-restoration semi-fragile watermarking algorithm for image content authentication resistant to JPEG compression. In: Proc. IWDW 2011. LNCS, vol. 7128, pp. 72–85 (2012). https://doi.org/10.1007/978-3-642-32205-1_8
30. Wang, K., Lavoué, G., Denis, F., Baskurt, A., He, X.: A benchmark for 3D mesh watermarking. In: Proc. SMI 2010. pp. 231–235 (2010)
31. Wang, K., Lavoué, G., Denis, F., Baskurt, A.: A comprehensive survey on three-dimensional mesh watermarking. IEEE Transactions on Multimedia **10**(8), 1513–1527 (2008). https://doi.org/10.1109/TMM.2008.2007350
32. Zhu, X., Ding, J., Dong, H., Hu, K., Zhang, X.: Normalized correlation-based quantization modulation for robust watermarking. IEEE Transactions on Multimedia **16**(7), 1888–1904 (2014). https://doi.org/10.1109/TMM.2014.2340695