**“DIGITAL WATERMARKING WITH STEGANOGRAPHY”**

**APPLICATION DEVELOPMENT – III & IV**

**IMPLEMENTATION PHASE**

**AND**

**TESTING AND DOCUMENTATION**

**Submitted in partial fulfillment for the award of the degree**

**Of**

**Master of Science**

**In**

**Information Technology**

**By**

**Vrushali Shivaji Walke**

**(14MIN3242)**



**School of Information Technology and Engineering**

**March, 2018**

INDEX

Contents

[1 INTRODUCTION 5](#_Toc509521410)

[1.1 LITERATURE SURVEY 5](#_Toc509521411)

[1.2 DIGITAL WATERMARKING PRINCIPLE 5](#_Toc509521412)

[1.3 DIGITAL WATERMARKING TECHNIQUES 6](#_Toc509521413)

[1.4 Spatial Domain Watermarking 9](#_Toc509521414)

[1.5 Transform Domain Watermarking 9](#_Toc509521415)

[2 Benefits of Watermarking: 10](#_Toc509521416)

[2.1 Pure Steganography 10](#_Toc509521417)

[2.2 Information Hiding in Noisy Data 10](#_Toc509521418)

[2.3 Active and Malicious Attackers During the design of a stenographic 10](#_Toc509521419)

[2.4 Detecting Secret Messages A passive attacker : 11](#_Toc509521420)

[3 Watermarking: 11](#_Toc509521421)

[3.1 Purpose 22](#_Toc509521422)

[3.2 Scope 23](#_Toc509521423)

[3.3 Definitions, Acronyms, And Abbreviations 23](#_Toc509521424)

[3.4 OVERVIEW 23](#_Toc509521425)

[3.5 OVERALL DESCRIPTION 23](#_Toc509521426)

[3.5.1 User classes and characteristics 23](#_Toc509521427)

[3.5.2 Design and implementation constraints 23](#_Toc509521428)

[3.5.3 User Documentation 24](#_Toc509521429)

[User manual for administrator 24](#_Toc509521430)

[4 SPECIFIC REQUIREMENTS 24](#_Toc509521431)

[4.1 EXTERNAL INTERFACE REQUIREMENTS 24](#_Toc509521432)

[4.1.1 User Interfaces 24](#_Toc509521433)

[4.1.2 Hardware Interfaces 24](#_Toc509521434)

[4.1.3 Software Interfaces 24](#_Toc509521435)

[4.2 FUNCTIONAL REQUIREMENTS 25](#_Toc509521436)

[5 System Design 25](#_Toc509521437)

[5.1 DIGITAL WATERMARKING 25](#_Toc509521438)

[5.2 Retrieve 25](#_Toc509521439)

[5.3 Control 25](#_Toc509521440)

[6 PROJECT PLAN 25](#_Toc509521441)

[6.1 System Information engineering and modeling 26](#_Toc509521442)

[6.2 Software Requirement Analysis 26](#_Toc509521443)

[6.3 Design 27](#_Toc509521444)

[6.3.1 Code Generation 27](#_Toc509521445)

[6.3.2 Testing 27](#_Toc509521446)

[6.4 PHASES 27](#_Toc509521447)

[6.4.1 Project Plan of our project 27](#_Toc509521448)

[6.4.2 Survey on Phase Diagram 28](#_Toc509521449)

[6.5 PROJECT SCHEDULE 29](#_Toc509521450)

[7 IMPLEMENTATION 29](#_Toc509521451)

[7.1 USE CASE DIAGRAM 29](#_Toc509521452)

[7.2 CLASS DIAGRAM 31](#_Toc509521453)

[7.3 SEQUENCE DIAGRAM 32](#_Toc509521454)

[7.4 ACTIVITY DIAGRAM 34](#_Toc509521455)

[7.5 DEPLOYMENT DIAGRAM 36](#_Toc509521456)

[8 Benefit Analysis 36](#_Toc509521457)

[9 Working 37](#_Toc509521458)

[9.1 Embedding 37](#_Toc509521459)

[9.2 Attacks 37](#_Toc509521460)

[9.3 Detection 38](#_Toc509521461)

[10 Code: 38](#_Toc509521462)

[11 SNAP SHOTS 46](#_Toc509521463)

[12 TEST PLAN 49](#_Toc509521464)

[12.1 TEST STATERGY 49](#_Toc509521465)

[12.1.1 BLACK BOX TESTING 49](#_Toc509521466)

[12.1.2 WHITE BOX TESTING 49](#_Toc509521467)

[12.1.3 UNIT TESTING 50](#_Toc509521468)

[12.1.4 INTEGRATION TESTING 50](#_Toc509521469)

[12.2 TEST CASES 50](#_Toc509521470)

[12.2.1 INTRODUCTION 50](#_Toc509521471)

[12.2.2 ITEMS TO BE TESTED 50](#_Toc509521472)

[12.3 FEATURES TO BE TESTED 51](#_Toc509521473)

[12.3.1 Performance 51](#_Toc509521474)

[12.3.2 Functionality 51](#_Toc509521475)

[12.4 Item pass/fail criteria-Test Cases 51](#_Toc509521476)

[13 FUTURE WORK 53](#_Toc509521477)

[14 CONCLUSION 53](#_Toc509521478)

# INTRODUCTION

A digital watermark is pattern of bits inserted into a digital image, audio and video file that identify the files copyright information (author, rights, etc.)

The aim of this project was to investigate techniques that are robust in the transform domain and develop a watermarking program that implemented a robust technique. A watermarking system was also proposed for use with digital images.

## LITERATURE SURVEY

The expansion of the Internet has frequently increased the availability

of digital data such as audio, images and videos to the public. Digital watermarking is a technology being developed to ensure and facilitate data authentication, security and copyright protection of digital media. This paper incorporate the detail study watermarking definition, concept and the main contributions in this field such as categories of watermarking process that tell which watermarking method should be used.

It starts with overview, classification, features, framework, techniques, application, challenges, limitations and performance metric of watermarking and a comparative analysis of some major watermarking techniques. In the survey our prime concern is image only.

## DIGITAL WATERMARKING PRINCIPLE

1. Privacy by design –

Address privacy considerations in the early design and planning phases of digital watermarking applications, not late in the processes an afterthought;

2. Avoid embedding independently useful identifying information directly in watermark.

So that even if unauthorized third parties learn how to read the watermarks, no meaningful information will be exposed;

3. Provide notice to end users –

Disclose the existence and other key information about individualized watermarks, with a prominence appropriate to the extent and likelihood of any possible privacy impact;

4. Control access to reading capability –

so that members of the public who happen to obtain a watermarked file will not have easy access to the devices or software needed to read the watermarks;

5. Respond appropriately when algorithms are compromised –

reconsider how much reliance to place on watermarking systems whose workings have been exposed, particularly if there is a risk that watermarks could be altered or forged;

6. Provide security and access controls for back‑end databases –

Adopt rules and security safeguards to protect databases containing information about individuals from unauthorized access.

## DIGITAL WATERMARKING TECHNIQUES

In the field of digital watermarking, digital image watermarking has attracted a lot of Awareness in the research community for two reasons: one is its easy availability and the other is it convey enough redundant information that could be used to embed watermarks. Digital watermarking contains various techniques for protecting the digital content. The Entire digital image watermarking techniques always works in two domains either spatial Domain or transform domain. The spatial domain techniques works directly on pixels. It embeds the watermark by modifying the pixels value. Most commonly used spatial domain Techniques are LSB. Transform domain techniques embed the watermark by modifying the Transform domain coefficients. Most commonly used transform domain techniques is DCT, DWT and DFT. For achieving the robustness and imperceptibility, the transform domain techniques are more effective as compare to the spatial domain. We further elaborated these two domains and its techniques.

Some Applications of Information Hiding Unobtrusive communications are required by military and intelligence agencies**:** even if the content is encrypted, the detection of a signal on a modern battlefield may lead rapidly to an attack on the signaler. For this reason, military communications use techniques such as spread spectrum modulation or meteor scatter transmission to make signals hard for the enemy to detect or jam. Criminals also place great value on unobtrusive communications and their preferred technologies include prepaid mobile phones and hacked corporate switchboards through which calls can be re-routed . As a side effect, law enforcement and counterintelligence agencies are interested in understanding these technologies and their weaknesses, so as to detect and trace hidden messages. Information hiding techniques also underlie many attacks on "multilevel secure" systems used by military organizations. A virus or other malicious code propagates itself from "low security" to ''high security" levels and then signals data downwards using a covert channel in the operating system or by hiding information directly in data that may be declassified . Information hiding techniques can also be used in situations where plausible deniability6 is required. "The obvious motivation for plausible deniability is when the two communicating parties are engaged in an activity which is somehow illicit, and they wish to avoid being caught" but more legitimate motives include fair voting, personal privacy, or limitation of liability. One possible mechanism providing such property is the stenographic file system, presented by Anderson, Needham, and Shamir: if a user knows a file's name, he can retrieve it; but if he does not, he cannot even obtain evidence that the file exists.

Anonymous communications, including anonymous remailers and Web proxies, are required by legitimate users to vote privately in online elections, make political claims, consume sexual material, preserve online free speech, or to use digital cash. But the same techniques can be abused for defamation, blackmail, or unsolicited commercial mailing. The ethical positions of the players in the information hiding game are not very clear so the design of techniques providing such facilities requires careful thought about the possible abuses, which might be nonobvious. The healthcare industry and especially medical imaging systems may benefit from information hiding techniques. They use standards such as DICOM (digital imaging and communications in medicine) which separates image data from the caption, such as the name of the patient, the date, and the physician. Sometimes the link between image and patient is lost, thus, embedding the name of the patient in the image could be a useful safety measure. It is still an open question whether such marking would have any effect on the accuracy of the diagnosis but recent studies by Cosman et al.Revealing that lossy compression has little effect, let us believe that this might be feasible. Another emerging technique related to the healthcare industry is hiding messages in DNA sequences [48]. This could be used to protect intellectual property in medicine, molecular biology or genetics. A number of other applications of information hiding have been proposed in the context of multimedia applications. In many cases they can use techniques already developed for copyright marking directly; in others, they can use adapted schemes or shed interesting light on technical issues. They include the following:

• Automatic monitoring of copyrighted material on the Web: A robot searches the Web for marked material and hence identifies potential illegal usage.

• Automatic audit of radio transmissions: A computer can listen to a radio station and look for marks, which indicate that a particular piece of music, or advertisement, has been broadcast .

• Data augmentation: Information is added for the benefit of the public. This can be details about the work, annotations, other channels , or purchasing information (nearest shop, price, producer, etc.) so that someone listening to the radio in a car could simply press a button to order the compact disc. This can also be hidden information used to index pictures or music tracks in order to provide more efficient retrieval from databases .

• Tamper proofing: The information hidden in a digital object may be a signed "summary" of it, which can be used to prevent or to detect unauthorized modifications . Some of these applications and techniques will be detailed in the next chapters. We tried to keep chapters simple enough such that any computer science graduate student can understand them without much problem. Note however that steganography and digital watermarking require some background in various disciplines including cryptography, image processing, information theory, and statistics. It is outside the scope of this book to detail all the basic techniques on which information hiding techniques are built. If more background is required, we refer the reader to Menezes for cryptography, Jain for image processing, and Cover for information theory.

## Spatial Domain Watermarking

The spatial domain represents the image in the form of pixels. The spatial domain Watermarking embeds the watermark by modifying the intensity and the color value of some Selected pixels [12]. The strength of the spatial domain watermarking is

1. Simplicity.

2. Very low computational complexity.

3. Less time consuming.

The spatial domain watermarking is easier and its computing speed is high than transform Domain but it is less robust against attacks. The spatial domain techniques can be easily applied to any image. The most important method of spatial domain is LSB. Least Significant Bit (LSB):

The LSB is the simplest spatial domain watermarking technique to embed a watermark in the least significant bits of some randomly selected pixels of the cover image. Example of least significant bit watermarking [12]:

Image:

10010101 00111011 11001101 01010101….

Watermark:

1 0 1 0…..

Watermarked Image:

10010101 00111010 11001101 01010100…..

The steps used to embed the watermark in the original image by using the LSB [13]:

1) Convert RGB image to grey scale image.

2) Make double precision for image.

3) Shift most significant bits to low significant bits of watermark image.

4) Make least significant bits of host image zero.

5) Add shifted version (step 3) of watermarked image to modified (step 4) host image.

## Transform Domain Watermarking

The transform domai2n watermarking is achieving very much success as compared to the spatial domain watermarking. In the transform domain watermarking, the image is represented in the form of frequency. In the transform domain watermarking techniques, firstly the original image is converted by a predefined transformation. Then the watermark is

Embedded in the transform image or in the transformation coefficients. Finally, the inverse transform is performed to obtain the watermarked image. Most commonly used transform domain methods is Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT) and Discrete Fourier Transform (DFT).

# Benefits of Watermarking:

## Pure Steganography

We call a stenographic system which does not require the prior exchange of some secret information (like a stego-key) pure steganography.

## Information Hiding in Noisy Data

As we have noted in Section, steganography utilizes the existence of redundant information in a communication process. Images or digital sound naturally contain such redundancies in the form of a noise component. In this section, we will assume without loss of generality that the cover c can be represented by a sequence of binary digits. In the case of a digital sound this sequence is just the sequence of samples over time; in the case of a digital image, a sequence can be obtained by vectorizing the image (i.e., by lining up the grayscale or color values in a left-to-right and top-to-bottom order). Let l(c) be the number of elements in the sequence, m the secret message, and l(m) its length in bits.

## Active and Malicious Attackers During the design of a stenographic

system special attention has to be paid to the presence of active and malicious attackers. Active attackers are able to change a cover during the communication process; we could capture one stego-object sent modify it and forward the result. It is a general assumption that an active attacker is not able to change the cover and its semantics entirely, but only make minor changes so that the original and the modified cover-object stay perceptually or semantically similar. An attacker is malicious if he forges messages or starts steganography protocols under the name of one communication partner.

## Detecting Secret Messages A passive attacker :

Disabling Hidden Information Detecting the existence of hidden information defeats steganography. Methods that follow the transform domain produce results which are far more difficult to detect without the original image for comparison. Sometimes it may be desirable to let the stego-object pass along the communication channel, but disable the embedded message. With each method of hiding information, there is a trade-off between the size of the payload (amount of hidden information) that can be embedded, and the survivability or robustness of that information to be manipulated. Information hidden in text in the forms of appended spaces and ''invisible" characters can be easily revealed by opening the file with a word processor. Extra spaces and characters can be quickly stripped from text documents. Hidden information may also be overwritten. If information is added to some media such that the added information cannot be detected, then there exists some amount of additional information that may be added or removed within the same threshold which will overwrite or remove the embedded covert information. Caution must be used in hiding information in unused space in files or file systems. File headers and reserved spaces are common places to look for "out of place" information. In file systems, unless the stenographic areas are in some way protected (as in a partition), the operating system may freely overwrite the hidden data, since the clusters are thought to be free. This is a particular annoyance of operating systems that do a lot of caching and creating temporary files. Utilities are also available which "clean" or wipe unused storage areas. In wiping, clusters are overwritten several times to ensure data removal. Even in this extreme case, utilities exist that can recover portions of the overwritten information.

# Watermarking:

This Terminology Today, we are of course concerned with digital, rather than analog, communication and media. As in analog media, there is interest in stenographic and watermarking methods that allow the transmission of information hidden or embedded in other data. Several names have been coined for such techniques. However, the terms are often confused, and therefore it is necessary to clarify the differences. Visible watermarks, as the name says, are visual patterns like logos which are inserted into or overlaid on images (or video), very similar to visible paper watermarks. Visible watermarks are mainly applied to images, for example, to visibly mark preview images available in image databases or on the Web in order to prevent commercial use of such images. The applications of visible watermarks to video is of course also possible and under some circumstances one might even think of embedding an audible watermark into audio. An example of visible watermarking has been developed in the context of the IBM Digital Libraries projects. The technique combines the watermark image with the original image by modifying the brightness of the original image as a function of the watermark and a secret key. The secret key determines pseudorandom scaling values used for brightness modification in order to make it difficult for attackers to remove the visible mark. For the rest of this book we will focus on imperceptible watermarks. Watermarking, as opposed to steganography, has the additional notion of robustness against attacks. Even if the existence of the hidden information is known it should be hard for an attacker to destroy the embedded watermark without knowledge of a key. A practical implication of the robustness requirement is that watermarking methods can typically embed much less information into cover-data than stenographic methods. Steganography and watermarking are thus more complementary than competitive approaches. In the remainder of this chapter, we will mainly focus on watermarking methods, and not on stenographic methods in general, since they have been covered in the previous chapters. Fingerprinting and labeling are terms that denote special applications of watermarking. They relate to watermarking applications where information such as the creator or recipient of digital data is embedded as watermarks. Fingerprinting Generic digital watermarking scheme. Means watermarking where the embedded information is either a unique code specifying the author or originator of the cover-data, or a unique code out of a series of codes specifying the recipient of the data. More details on fingerprinting are presented in Chapter 8. Labeling means watermarking where the embedded data may contain any information of interest, such as a unique data identifier. Bitstream watermarking is sometimes used for watermarking of compressed data, for example compressed video. The term embedded signatures has been used instead of ''watermarking" in early publications, but is usually not used anymore, since it potentially leads to confusion with cryptographic signatures. Cryptographic signatures serve for authentication purposes. They are used to detect any alteration of the signed data and to authenticate the sender. Watermarks, however, are only used for authentication in special applications, and are usually designed to resist alterations and modifications. Fragile watermarks are watermarks that have only very limited robustness. They are applied to detect modifications of the watermarked data, rather than conveying unerasable information.

**Basic Watermarking** Principles All watermarking methods share the same generic building blocks: a watermark embedding system and a watermark recovery system (also called watermark extraction or watermark decoder). The input to the scheme is the watermark, the cover-data and an optional public or secret key. The watermark can be of any nature such as a number, text, or an image. The key may be used to enforce security, that is the prevention of unauthorized parties from recovering and manipulating the watermark. All practical systems employ at least one key, or even a combination of several keys.

**Generic watermark recovery scheme.** referred to as secret and public watermarking techniques, respectively. The output of the watermarking scheme is the watermarked data. For real-world robust watermarking systems, a few very general properties, shared by all proposed systems, can be identified. They are:

• Imperceptibility. The modifications caused by watermark embedding should be below the perceptible threshold, which means that some sort of perceptibility criterion should be used not only to design the watermark, but also quantify the distortion. As a consequence of the required imperceptibility, the individual samples (or pixels, voxels, features, etc.) that are used for watermark embedding are only modified by a small amount.

• Redundancy. To ensure robustness despite the small allowed changes, the watermark information is usually redundantly distributed over many samples (or pixels, voxels, features, etc.) of the cover-data, thus providing a global robustness which means that the watermark can usually be recovered from a small fraction of the watermarked data. Obviously watermark recovery is more robust if more of the watermarked data is available in the recovery process.

• Keys. In general, watermarking systems use one or more cryptographically secure keys to ensure security against manipulation and erasure of the watermark. As soon as a watermark can be read by someone, the same person may easily destroy it because not only the embedding strategy, but also the locations of the watermark are known in this case. These principles apply to watermarking schemes for all kinds of data that can be watermarked, like audio, images, video, formatted text, 3D models, model animation parameters, and others. The generic watermark recovery process is depicted in Figure

**Watermarking Applications** The requirements that watermarking systems have to comply with are always based on the application. Thus, before we review the requirements and the resulting design considerations, we will present some applications of watermarking. For obvious reasons there is no "universal" watermarking method. Although watermarking methods have to be robust in general, different levels of required robustness can be identified depending on the specific application-driven requirements.

Watermarking for Copyright Protection Copyright protection is probably the most prominent application of watermarking today. The objective is to embed information about the source, and thus typically the copyright owner, of the data in order to prevent other parties from claiming the copyright on the data. Thus, the watermarks are used to resolve rightful ownership, and this application requires a very high level of robustness. The driving force for this application is the Web which contains millions of freely available images that the rightful owners want to protect. Additional issues besides robustness have to be considered. For example, the watermark must be unambiguous and still resolve rightful ownership if other parties embed additional watermarks. Hence, additional design requirements besides mere robustness apply.

**Fingerprinting for Traitor Tracking** There are other applications where the objective is to convey information about the legal recipient rather than the source of digital data, mainly in order to identify single distributed copies of the data. This is useful to monitor or trace back illegally produced copies of the data that may circulate, and is very similar to serial numbers of software products. This type of application is usually called "fingerprinting" and involves the embedding of a different watermark into each distributed copy. Because the distribution of individually watermarked copies allow collusion attacks , the embedded watermarks have to be designed as collusion-secure. Also, for some fingerprinting applications it is required to extract the watermark easily and with a low complexity, for example, for World Wide Web applications where special Web crawlers search for pirated watermarked images. Watermarks for fingerprinting applications also require a high robustness against standard data processing as well as malicious attacks.

**Watermarking for Copy Protection** A desirable feature in multimedia distribution systems is the existence of a copy protection mechanism that disallows unauthorized copying of the media. Copy protection is very difficult to achieve in open systems; in closed or proprietary systems, however, it is feasible. An example is the DVD system where the data contains copy information embedded as a watermark. A compliant DVD player is not allowed to playback or copy data that carry a "copy never" watermark. Data that carry a "copy once" watermark may be copied, but no further consecutive copies are allowed to be made .

**Watermarking for Image Authentication** In authentication applications, the objective is to detect modifications of the data. This can be achieved with socalled "fragile watermarks" that have a low robustness to certain modifications like compression, but are impaired by other modifications. Furthermore, the robustness requirements may change depending on the data type and application. Nevertheless, among all possible watermarking applications, authentication watermarks require the lowest level of robustness by definition. It should be noted that new approaches have emerged in which data attributes, such as block average or edge characteristics, are embedded and check if the received image still has the same attributes. It is clear that such schemes may require a higher robustness if identification of the modified areas is of interest. **Requirements and Algorithmic Design** **Issues** Depending on the watermarking application and purpose, different requirements arise resulting in various design issues. Watermark imperceptibility is a common requirement and independent of the application purpose. Additional requirements have to be taken into consideration when designing watermarking techniques:

• Recovery with or without the original data. Depending on the application, the original data is or is not available to the watermark recovery system. If the original is available, it is usually advantageous to use it, since systems that use the original for recovery are typically more robust. However, in applications such as data monitoring, the original data is of no use because the goal is to identify the monitored data.

• Extraction or verification of a given watermark. There are two inherently equivalent approaches for watermark embedding and recovery. In the first approach, one watermark out of a predefined set of admissible watermarks is embedded, and the watermark recovery tests the watermarked data against the admissible set. The output of the watermark recovery is the index of the embedded watermark or a symbol "no watermark found." In the second approach, the embedded watermark is the modulation of a sequence symbols given to the watermark embedding system. In the detection process the embedded symbols are extracted through demodulation.

• Robustness. Robustness of the watermarked data against modifications and/or malicious attacks is one of the key requirements in watermarking. However, as said before, there are applications where it is less important than for others.

• Security issues and use of keys. The conditions for key management differ greatly depending on the application. Obvious examples are public key watermarking systems like DVD versus secret key systems used for copyright protection.

Imperceptibility One the most important requirements is the perceptual transparency of the watermark, independent of the application and purpose of the watermarking system. Artifacts introduced through a watermarking process are not only annoying and undesirable, but may also reduce or destroy the commercial value of the watermarked data. It is therefore important to design marking methods which exploit effects of the human visual or auditory system in order to maximize the energy of the watermark under the constraint of not exceeding the perceptible threshold. Two problems are related to this issue. The first one is the reliable assessment of the introduced distortion. The second problem occurs when processing is applied to the watermarked data. For example, in image watermarking the visibility of the watermark may increase if the image is scaled.

**Robustness** The ultimate watermarking method should resist any kind of distortion introduced by standard or malicious data processing. No such perfect method has been proposed so far, and it is not clear yet whether an absolutely secure watermarking method exists at all. Thus, practical systems must implement a compromise between robustness and the competing requirements like invisibility and information rate. Depending on the application purpose of the watermarking methods, the desired robustness therefore influences the design process. For example in image watermarking, if we need a method that is resilient to JPEG compression with high compression factors, it is probably more efficient to employ a method working in a transform domain than to use a method that works in the spatial domain. Similarly, if the method should accommodate generalized geometrical transformations, that is rotation, nonuniform scaling, and shearing, an approach in the spatial domain is probably more suitable. Looking at the distortion that the watermarked data is likely to undergo by either intentional or unintentional modifications, two groups of distortion can be distinguished. The first one contains distortions which can be considered as additive noise to the data whereas the distortion in the second group are due to modifications of the spatial or temporal data geometry with the intent to introduce a mismatch between the watermark and the key used for embedding. These two distortions or attacks are often referred to as destruction attacks and synchronization attacks, respectively. Finer categorization of attacks were also proposed . Depending on the application and watermarking requirements, the list of distortions and attacks to be considered includes, but is not limited to:

• Signal enhancement (sharpening, contrast enhancement, color correction, gamma correction);

• Additive and multiplicative noise (Gaussian, uniform, speckle, mosquito);

• Linear filtering (lowpass-, highpass-, bandpass filtering);

• Nonlinear filtering (median filtering, morphological filtering);

• Lossy compression (images: JPEG, video: H.261, H.263, MPEG-2, MPEG-4, audio: MPEG-2 audio, MP3, MPEG-4 audio, G.723);

• Local and global affine transforms (translation, rotation, scaling, shearing);

• Data reduction (cropping, clipping, histogram modification);

• Data composition (logo insertion, scene composition);

• Transcoding (H.263 → MPEG-2, GIF → JPEG);

• D/A and A/D conversion (print-scan, analog TV transmission);

• Multiple watermarking;

• Collusion attacks;

• Statistical averaging;

• Mosaic attacks. Some of these attacks, and others. The basic principle is to design watermarking methods, which are robust enough such that successful attacks would also impair the commercial value of the cover-data.

**Watermark Recovery** with or without the Original Data Watermarking methods using the original data set in the recovery process usually feature increased robustness not only towards noise-like distortions, but also distortions in the data geometry since it allows the detection and inversion of geometrical distortions. In many applications, such as data monitoring or tracking, access to the original data is not possible. In other applications, such as video watermarking applications, it may be impracticable to use the original data because of the large amount of data that would have to be processed. While most early watermarking techniques require the original data for recovery, there is a clear tendency to devise techniques that do not require the original data set. This is probably due to the larger fields of applications for such techniques.

**Watermark Extraction** or Verification of Presence for a Given Watermark As was said before, two types of watermarking schemes exist: systems that embed a specific information or pattern and check the existence of the (known) information later on in the watermark recovery process, and systems that embed arbitrary information into the data. For example, copyright protection can be achieved with systems verification of the presence of a known watermark. Watermarking schemes embedding arbitrary information are, for example, used for image tracking on the Internet with intelligent agents where it might not only be of interest to discover images, but also to classify them. The embedded watermark can be used as an image identification number or as a pointer to a database entry. It should be noted that both schemes can be interchanged. A scheme which allows watermark verification can be considered as a one-bit watermark recovery scheme, and can easily be extended to any number of bits by modulation with the arbitrary information to be embedded. The inverse is also true: a watermark recovery scheme can be considered as a watermark verification scheme assuming the embedded information is known.

Watermark Security and Keys In most applications, such as copyright protection, the secrecy of embedded information needs to be assured. This and related issues are often referred to as watermark security. Applications in which security is not an issue include image database indexing. If secrecy is a requirement, a secret key has to be used for the embedding and extraction process. Two levels of secrecy can be identified. In the highest level of secrecy an unauthorized user can neither read or decode an embedded watermark nor can he detect if a given set of data contains a watermark. The second level permits any user to detect if data is watermarked, but the embedded information cannot be read without having the secret key. Such schemes may, for example, be useful in copyright protection applications for images. As soon as a copyrighted image is opened in a photo editing software program the user is informed by a note indicating that the image is protected. Such schemes can, for example, contain multiple watermarks, with public and secret keys. It is also possible to combine one or several public keys with a private key and embed a joint public/private watermark. When designing a working overall copyright protection system, issues like secret key generation, distribution, and management (possibly by trusted third parties), as well as other system integration aspects have to be considered as well. Resolving Rightful Ownership In order to successfully resolve rightful ownership, it must be possible to determine who first watermarked a data set in case it contains multiple watermarks. This can be achieved by imposing design constraints, such as the noninvertibility of the watermark [18] or using additional functionalities, such as time-stamping. Evaluation and Benchmarking of Watermarking Systems Besides designing digital watermarking methods, an important issue addresses proper evaluation and benchmarking. This not only requires evaluation of the robustness, but also includes subjective or quantitative evaluation of the distortion introduced through the watermarking process. In general, there is a trade-off between watermark robustness and watermark perceptibility. Hence, for fair benchmarking and performance evaluation one has to ensure that the methods under investigation are tested under comparable conditions — Performance Evaluation and Representation Independent of the application purpose type of data, the robustness of watermarks depends on the following aspects:

• Amount of embedded information. This is an important parameter since it directly influences the watermark robustness. The more information one wants to embed, the lower the watermark robustness.

• Watermark embedding strength. There is a trade-off between the watermark embedding strength (hence the watermark robustness) and watermark perceptibility. Increased robustness requires a stronger embedding, which in turn increases perceptibility of the watermark.

• Size and nature of data. The size of the data has usually a direct impact on the robustness of the embedded watermark. For example, in image watermarking very small pictures do not have much commercial value; nevertheless, a marking software program needs to be able to recover a watermark from them. This avoids a "mosaic" attack on them and allows tiling, used often in Web applications. For printing applications high-resolution images are required but one also wants to protect these images after they are resampled and used on the Web. In addition to the size of the data, the nature of the data also has an important impact on the watermark robustness. Again taking image watermarking as an example, methods featuring a high robustness for scanned natural images have an surprisingly reduced robustness for synthetic, such as computer generated, images.

• Secret information (e.g., key). Although the amount of secret information has no direct impact on the perceptibility of the watermark and the robustness of the watermark, it plays an important role in the security of the system. The key space, that is, the range of all possible values of the secret information, must be large enough to make exhaustive search attacks impossible. The reader should also keep in mind that many security systems fail to resist very simple attacks because the system designers did not obey basic cryptographic principles in the design [20, 21]. Taking these parameters into account, we realize that for fair benchmarking and performance evaluation, watermarking methods need to be tested on different data sets. Furthermore, in order to compute statistically valid results the methods have to be evaluated using many different keys and varying watermarks. The amount of embedded information is usually fixed and depends on the application. However, if watermarking methods are to be compared, it has to be assured that the amount of embedded information is the same for all methods under inspection. As we have seen above, there is a trade-off between the watermark perceptibility and the watermark robustness. For fair evaluation and comparison it is therefore necessary to consider the perceptibility of the watermark in the evaluation process. Evaluating the perceptibility of the watermarks can be done either through subjective tests or a quality metric. When using a subjective test, a testing protocol has to be followed, describing the testing and evaluation procedure. Such tests usually involve a two-step process. In a first round, the distorted data sets are rank ordered from best to worst. In the second round, the subject is asked to rate each data set, describing the perceptibility of the artifacts. This rating, can be based, for example, on the ITU-R Rec. 500 quality rating. The ratings and the corresponding perception and quality. Work done within the European projects OCTALIS (Offer of Content Through Trusted Access Links) has shown that individuals with different experience, for example professional photographers and researchers, generate quite different results in subjective tests on watermarked images. Subjective tests are practical for final quality evaluation and testing, but are not very useful in a research and development environment. ITU-R Rec. 500 quality ratings on a scale from 1 to 5. Rating Impairment Quality 5 Imperceptible excellent 4 Perceptible, not annoying good 3 Slightly annoying fair 2 Annoying poor 1 Very annoying bad In such an environment, quantitative distortion metrics are much more efficient and allow fair comparison between different methods as the results do not depend on subjective evaluations. Table 5.3 shows pixel-based differencedistortion metrics commonly used in image and video processing. Most of the shown metrics are also applicable, after adaption of the dimension, to other types of data than images, for example audio. Nowadays, the most popular distortion measures in the field of image and video coding and compression are the signal-to-noise ratio (SNR), and the peak signal-to-noise ratio (PSNR). They are usually measured in decibels (dB): SNR(dB) = 10 log10(SNR). It is well known that these difference distortion metrics are not very well correlated with the human visual or auditory system. This might be a problem for their application in digital watermarking since sophisticated watermarking methods exploit in one way or the other effects of these systems. Using the above metric to quantify the distortion caused by a watermarking process might therefore result in misleading quantitative distortion measurements. It might be useful, therefore, to use distortion metrics adapted to the human visual and auditory systems. In recent years more and more research concentrates on such adapted distortion metrics and it is very probable that future benchmarking of digital watermarking systems employs such quality metrics. After fixing the parameters and deciding on a distortion metric, the next issue addresses efficient robustness evaluation and visual representation. For all evaluation strategies, it is very important to perform the tests using different keys and a variety of data sets (e.g., many different images). The results should then be averaged and plotted. If performance evaluation on individual data sets is desirable for direct performance comparison of two methods for one data set, it is still important that all tests are repeated several times using different keys. In the next paragraphs we will briefly explain the four different graphs. The term attack refers to any attack as outlined in and robustness describes the watermark resistance to these attacks. The robustness is usually measured by the bit-error rate, defined as the ratio of wrong extracted bits to the total number.

## Purpose

Digital watermarking can also be used to convey side-channel information with the purpose of enhancing functionality of the system or adding value to the content it is embedded in this type of applications, where a device is designed to react to watermark for the benefit of the user, is also referred to as device control applications

## Scope

The project ‘Steganography’ will basically deal with data security. It will provide security of data by mechanism which is popularly known as ‘Steganography’ in the world of internet security. In this project, the focus is on image watermarking visible as well as invisible. Visible Watermarking involves only insertion of the watermark. Whereas Invisible Watermarking involves insertion of the watermark and extraction of a Watermark.

## Definitions, Acronyms, And Abbreviations



## OVERVIEW

This project will basically implement the Steganography technique of hiding the data or information behind the image/audio/video file. It will also help in the transfer of the information from one machine to another machine.

## OVERALL DESCRIPTION

The main aim of this project will be the data security aspect.

### User classes and characteristics

DIGITAL WATERMARKING is defined as the imperceptibly altering a work in order to embed information about that work. In the recent years copyright protection of digital content became a serious problem due to rapid development in technology.

### Design and implementation constraints

Digital system designs are the product of valuable effort and know-how. Their embodiments, from software and hardware description language program down to device-level net list and mask data, represent carefully guarded intellectual property (IP).

### User Documentation

### User manual for administrator

# SPECIFIC REQUIREMENTS

## EXTERNAL INTERFACE REQUIREMENTS

### User Interfaces

The UI should present the user with a clear model of the effects of watermarking and protect him from misuse. It needs to iron out users misconceptions about the watermarking technique. Since watermark embedding programs come with image processing applications, the user might accidentally remove the watermark by working on the image. In this case, it will be the watermarking tool.

### Hardware Interfaces

* Display drive that should support 32-bit color scheme.
* Display resolution that should be 1024 x 768 pixels.
* Graphics Drive that can support 800 x 600 display resolution
* Minimum of 128 MB RAM is required
* The processor preferably should be Pentium III or above /its equivalent Justification.
* The screen resolution of 1024 x 768 pixels must be used for the better vision and clarity of the system.
* Minimum 128 MB RAM must be used for faster access of the system
* Processor of Pentium III or equivalent must be used for faster access of the system.

### Software Interfaces

* Any windows based 32 /64 bit operating system
* JDK 1.7.0 or higher
* Net beans 6.9 or higher.

## FUNCTIONAL REQUIREMENTS

It defines the fundamental actions that must take place in the software for accepting and processing the inputs and generating the outputs.

# System Design

## DIGITAL WATERMARKING

The increasing amount of research on watermarking over the past decade has been largely driven by its important applications in digital copyrights management and protection.

## Retrieve

Digital watermarking, the art of robust information hiding in images, has gained a large interest in both research and industry in the past few years. Every watermarking system has two main parts: watermark embedding and watermark extraction/detection. Previous developments in the theoretical analysis of digital watermarking schemes have shown that there is a direct link between digital watermarking and communication theory. These parallels are extremely helpful when it comes to designing efficient and optimal watermark detectors. We focus on this issue and investigate how to design good watermark detectors taking into account the basics of communication theory. Furthermore, we discuss different ways to achieve robustness to geometrical attacks and elaborate on watermark attacks.

## Control

The Control module handles the terminal operations such as shutdown, restart and logoff, by specifying the IP address of the remote system. For this purpose we are using RMI (Remote Method Invocation) and Java Runtime class.

# PROJECT PLAN

Before developing any software system we need to plan its steps systematically. There are various methods of process of software development like waterfall model, rapid application development model, incremental model, spiral model, fourth generation techniques etc.

* Requirement analysis
* Planning
* Design & analysis
* Coding
* Testing

System/Information Engineering

Coding

Testing

**Figure :Software Development Model**

This model is called as Incremental model suggest a systematic incremental approach to software development that begun at the system level and progress through analysis, design, coding and testing.

## System Information engineering and modeling

Because software is always a part of larger system, work begins by establishing requirements to software. This system view is essential when software must interact with other elements such as hardware people and database. System engineering and analysis encompass requirements gathering at the system level with a small amount of top level design and analysis. Information engineering encompass information gathering at the strategic business level and at the business area level.

## Software Requirement Analysis

The requirement gathering process is intensified and focused specially on software. To understand the nature of the program to be built, the software engineer must understand the information domain of the software, as well as required function, behavior, performance and interface.

## Design

Software design is actually a multi-step process that focuses on 4 district attributes of program data structures, software architecture, interface representation and procedural details. The design process translate requirement into representation of software configuration.

### Code Generation

The design must be translated into machine readable form. The code generation step performs this task. If design is performed in detailed manner, code generation can be accomplished mechanically.

### Testing

Once code has been generated, program testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested on the functional externals that is conducting test to uncover errors and ensure that the defined input will produced actual results that agree with required results

## PHASES

### Project Plan of our project

**Table 6.1: Project plan**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Task** | **Description** |
| **Phase 1** | **Analysis** | **Analyze the current scenario of client server architecture.** |
| **Phase 2** | **Literature survey** | **Collect raw data and elaborate on literature surveys. Study the current case.** |
| **Phase 3** | **Design** | **Assign the module and design the process flow control.** |
| **Phase 4** | **Implementation** | **Implement the code for all the modules and integrate all the modules.** |
| **Phase 5** | **Testing** | **Test the code and overall process whether the process works properly.** |
| **Phase 6** | **Documentation** | **Prepare the document for this project with conclusion and future enhancement.** |

### Survey on Phase Diagram

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date**  **Phase** | **Mar 17** | **April 17** | **Sep 17** | **Dec 18** | **Feb 18** | **Mar 18** | **Mar 18** |
| **Phase 1** |  |  |  |  |  |  |  |
| **Phase 2** |  |  |  |  |  |  |  |
| **Phase 3** |  |  |  |  |  |  |  |
| **Phase 4** |  |  |  |  |  |  |  |
| **Phase 5** |  |  |  |  |  |  |  |
| **Phase 6** |  |  |  |  |  |  |  |

**Figure : Phase Diagram**

## PROJECT SCHEDULE

**Table: Project Schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task to be Accomplished** |  | **Start Date** | **End Date** |
|  | | | |
| Understanding the concept of Digital watermarking |  | 05th Mar, 2017 | 25th April, 2017 |
| Studying different existing algorithms for image watermarking |  | 2nd  May | 17th May |
| Drafting Problem Statement and Software Requirement Specifications |  | 22st Sept | 30th Sept |
| Downloading research papers and studying them. |  | 5th Dec | 25th Dec |
| collected material as per the project requirement |  | 01stJan | 15th Jan |
| Project Report Stage – I |  | 15th January | 20th January |
| Implementation of 1st module |  | 21th Jan | 15th Feb |
| Test Implemented modules |  | 16th  March | 21st March |
| Testing of the system |  | 1st  Feb | 21th Feb |
| Documentation |  | 1th March | 21th March |

# IMPLEMENTATION

## USE CASE DIAGRAM

There is one administrator who will start up and shut down the system. The user selects two images, one of them is the image which he wants to morph and the other image is relevant to the first image. The output of the program will be the watermark image.

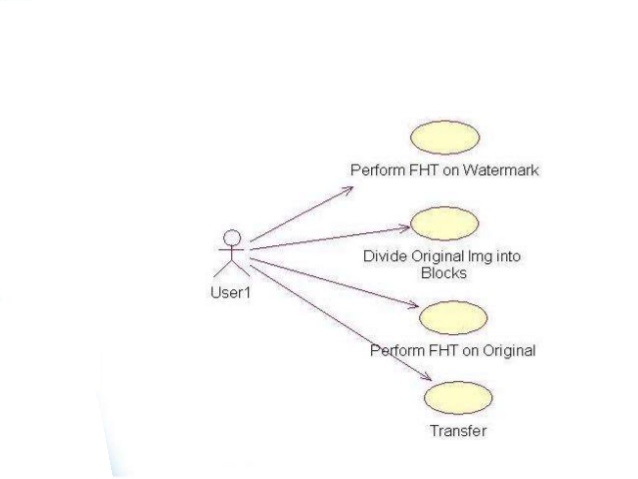


Figure 6.1 Use case diagrams

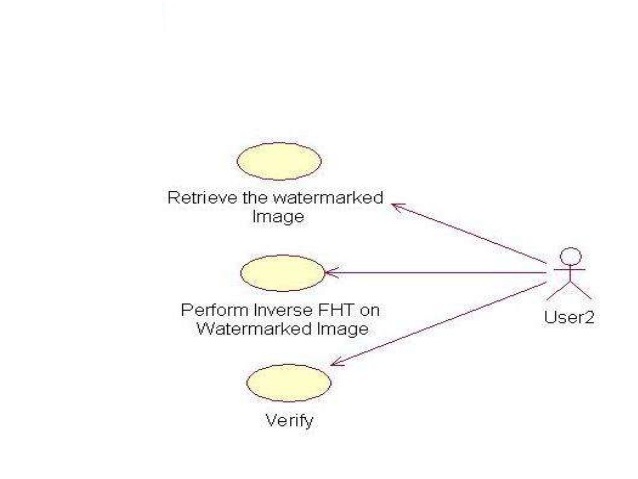
****

Figure :Use case diagrams

User can send the morphed image using the internet media to the destination. Actors are system; user and admin refer Figure 6.1.

## CLASS DIAGRAM

In the conceptual design of system, a number of classes are identified and group together in class diagram which helps to determines the statical relation between those objects. With detail modeling, the classes of the conceptual designer often split in a number of sub classes

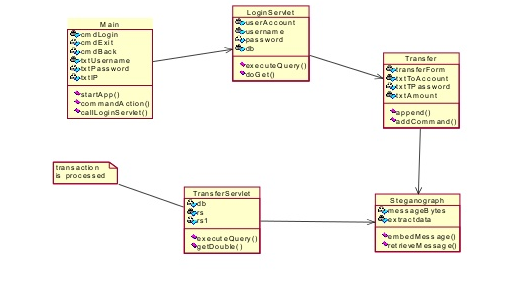


Figure : Class diagram

## SEQUENCE DIAGRAM

System, user, admin, morphing algo and internet are the objects or classes present in the figure As horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. In order to display interaction, messages are used. These are horizontal [arrows](http://en.wikipedia.org/wiki/Arrow_%28symbol%29) with the message name written above them. Solid arrows with full heads are synchronous calls, solid arrows with stick heads are asynchronous calls and dashed arrows with stick heads are return messages refer

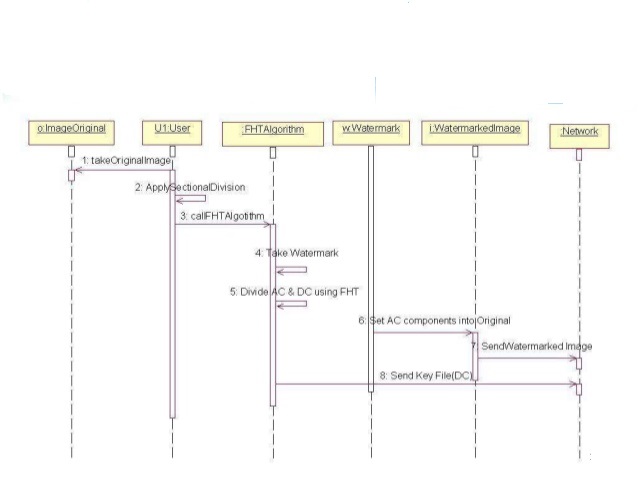


Figure 6.4: Sequence diagram

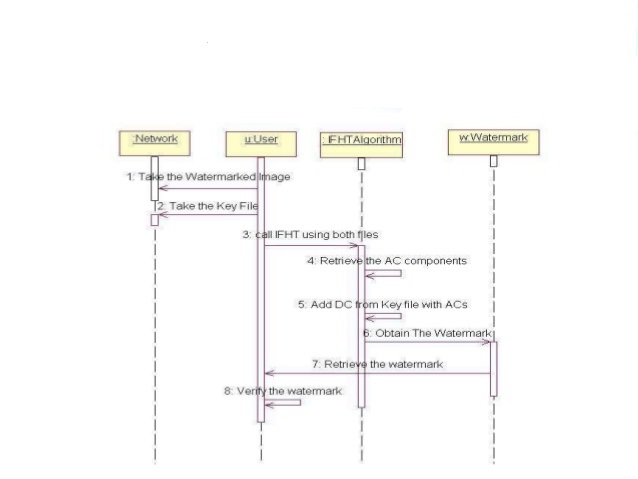


Figure : Sequence diagram

## ACTIVITY DIAGRAM

Activity diagram shows the various activities that are performed. Admin can turn on and off the system. The user will select two images and give them as input to the morphing algorithm. The output of the morphing algorithm will be the morphed image. The user can save the image or send it to the destination. If the user sends the morphed image to the destination the algorithm then generates stego keys which will be sent through layer and the image will be sent through application layer refer figure 6.4.



Figure : Activity diagram

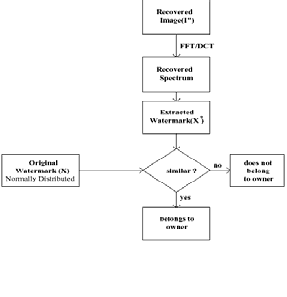


Figure:.Activity diagram

## DEPLOYMENT DIAGRAM

Multiple instances of the application server execution environment may be deployed inside a single device node to represent a application server clustering.

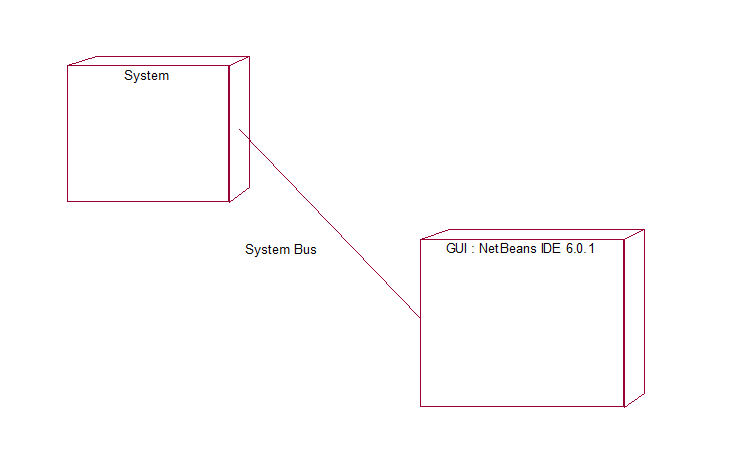


Figure : Deployment diagram

# Benefit Analysis

1. The system provides a very good user-friendly interface to enhance the communication between the user and the software.

2. The system is platform independent, thus allows the user to deploy on any operating system.

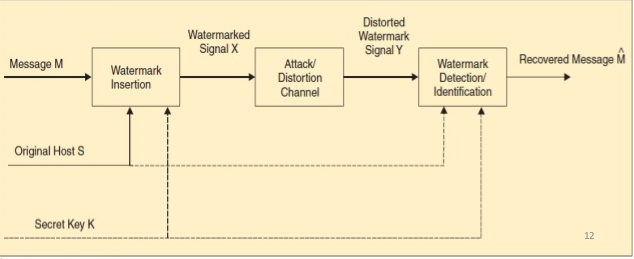
3. The information security provided by this application is very valuable as any organization cannot compromise on this aspect.

4. This application also provides the feature of transferring the files from one machine to another machine which is also the valuable aspect when combined with the security.

5. This application also provide a very good help support for the application.

# Working

**Digital watermarking Life Cycle Phase**



A watermarking system is divided into three steps:

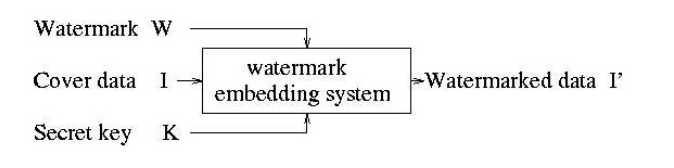
1. Embedding

2. Attack

3. Detection

## Embedding

In embedding, an algorithm accepts the host and the data to be embedded, and produce a watermarked signal

****

Inputs to the scheme are the watermark, the covert data and an optional public or secret key. The outputs are watermarked data. The key is used to enforce security.

## Attacks

The watermarked digital signal is transmitted or stored, usually transmitted to another person.

If this person makes a modification, this is called an attack.

Few possible attacks:

1. Robustness attack: which are intended to remove the watermark such as JPEG comparison, cropping, etc.

2. Presentation attack: under watermark detection failure they come into play.

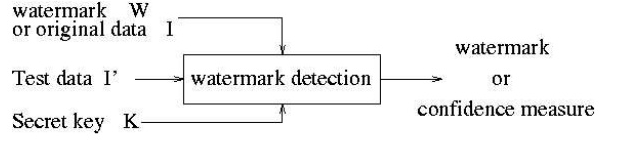
Geometric transformation, rotation, scaling, translation, change aspect ratio, etc.

3. Counterfeiting attacks: Rendering the original image, generate fake original.

## Detection

Detection is an algorithm which is applied to attacked signal to attempt to extract the watermark from it.

If the signal was unmodified during transmission, then the watermark still is present and it may be extracted.



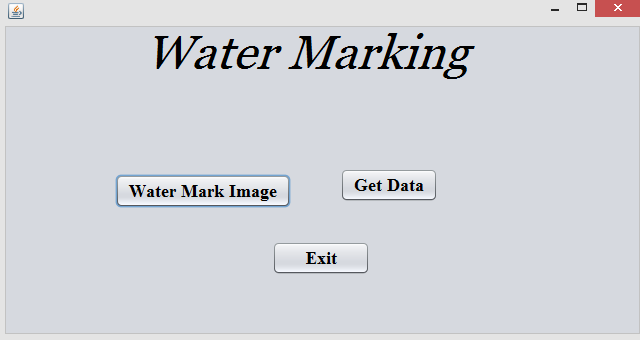
Input to scheme is the watermarked data, the secret or public key and, depending on the method, the original data and /or the original watermark. The output is the recovered watermarked W or some kind of confidence measure indicating how likely it is for given watermark at the input to present in the data under inspection.

# Code:

Main Function :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | \* encoding text into image; and decode(), for decoding text from | |  | \* an image. | |  | \*/ | |  |  | |  | import java.awt.image.BufferedImage; | |  | import java.io.File; | |  | import java.io.IOException; | |  | import java.util.ArrayDeque; | |  | import java.util.Deque; | |  |  | |  | import javax.imageio.ImageIO; | |  |  | |  | public class SteganoImgProcess { | |  |  | |  | String ext; | |  | int encodedMsgOffset; | |  |  | |  | //Method for encoding text into source image | |  | boolean encode(BufferedImage input, BufferedImage output, int width, int height, String msg, String outputName) { | |  |  | |  |  | |  | int msgLength = msg.length(); //original message length | |  | //add overhead "!encoded!" to identify start of encoded message, | |  | //along with original message length which will instruct when to stop | |  | //the decoding process, once the encoded message has been extracted | |  | String message = "!encoded!" + msgLength + "!" + msg; | |  | msgLength = message.length(); //message length included the overhead | |  |  | |  | //Creating an array of integers which will hold the entire message | |  | //in the form of bit format. A message is made up of characters, | |  | //each character is of the size of one byte. 1 byte = 8 bits. | |  | //Each byte here is divided into 4 parts of 2 bits, and stored in | |  | //the twoBitMessage array. Thus, the size of twoBitMessage is | |  | //4 times the length of the actual message. | |  | int[] twoBitMessage = new int[4 \* msgLength]; | |  |  | |  | char currentChar; | |  | for(int i =0; i < msgLength ; i++) { | |  | currentChar = message.charAt(i); // extracting character at position i from string | |  | twoBitMessage[4\*i + 0] = (currentChar >> 6) & 0x3; //storing 1st and 2nd bit from the left | |  | twoBitMessage[4\*i + 1] = (currentChar >> 4) & 0x3; //storing 3rd and 4th bit from the left | |  | twoBitMessage[4\*i + 2] = (currentChar >> 2) & 0x3; //storing 5th and 6th bit from the left | |  | twoBitMessage[4\*i + 3] = (currentChar) & 0x3; //storing 7th and 8th bit from the left | |  | } | |  |  | |  | int pixel, pixOut, count = 0;; | |  | loop: for(int i = 0; i < width; i++) { | |  | for(int j = 0; j < height; j++) { | |  | if(count < 4\*msgLength) { //ensuring that loop only iterates till the entire message has been encoded | |  | pixel = input.getRGB(i, j); //Grab the RGB value from the pixel of the source image at position (i,j) | |  |  | |  | //Bit operator AND is used to convert the two LSB to zero. | |  | //Once that is established, bit operator OR is used to copy | |  | //the two bit message in place of these two LSB. | |  | //Thus, we encode two bits of our message in one pixel. | |  | //Effectively, 4 pixels, carrying 8 bits of encoded bits | |  | //in total, will carry the information of one character. | |  | pixOut = (pixel & 0xFFFFFFFC) | twoBitMessage[count++]; //modified RGB value with encoded data | |  |  | |  | output.setRGB(i, j, pixOut); //Set the modified RGB value at given pixel. | |  |  | |  | } else { | |  | break loop; | |  | } | |  | } | |  |  | |  | } | |  |  | |  | try { | |  | ImageIO.write(output, "png", new File(outputName)); //create .PNG file with encoded image data | |  | return true; | |  | } catch (IOException e) { | |  | e.printStackTrace(); | |  | return false; | |  | } | |  |  | |  | }// end of encode() | |  |  | |  |  | |  | String decode(BufferedImage input, int width, int height) { | |  |  | |  | if(!isEncoded(input, width, height)) { | |  | return null; | |  | } | |  |  | |  | int msgLength = getEncodedLength(input, width, height); | |  |  | |  | StringBuffer decodedMsg = new StringBuffer(); | |  | Deque<Integer> listChar = new ArrayDeque<Integer>(); | |  |  | |  | int pixel, temp, charOut, ignore = 0, count = 0; | |  | loop: for(int i = 0; i < width; i++) { | |  | for(int j = 0; j < height; j++) { | |  | if(ignore < 36 + 4\*(String.valueOf(msgLength).length()+1)) { | |  | ignore++; | |  | continue; | |  | } | |  |  | |  | if(count++ == 4\*msgLength) { | |  | break loop; | |  | } | |  | pixel = input.getRGB(i, j); //grab RGB value at specified pixel | |  | temp = pixel & 0x03; //extract 2 LSB from encoded data | |  |  | |  | listChar.add(temp); //add the extracted data to a queue for later processing | |  |  | |  | if(listChar.size() >=4) { //once we have 8 bits of data extracted | |  | //combine them to create a byte, and store this byte as a character | |  | charOut = (listChar.pop() << 6) | (listChar.pop() << 4) | (listChar.pop() << 2) | listChar.pop() ; | |  | decodedMsg.append((char)charOut); | |  | } | |  | } | |  |  | |  | } | |  |  | |  | String outputMsg = new String(decodedMsg); //generate extracted message | |  |  | |  | return outputMsg; | |  | } //end of decode() | |  |  | |  | boolean isEncoded(BufferedImage input, int width, int height) { //Check for "!encoded!" at starting | |  |  | |  | StringBuffer decodedMsg = new StringBuffer(); | |  | Deque<Integer> listChar = new ArrayDeque<Integer>(); | |  |  | |  | int pixel, temp, charOut, count = 0; | |  | loop: for(int i = 0; i < width; i++) { | |  | for(int j = 0; j < height; j++, count++) { | |  |  | |  | if(count == 45) { //remain in loop till first 9 characters are extracted | |  | break loop; | |  | } | |  | pixel = input.getRGB(i, j); //grab RGB value at specified pixel | |  | temp = pixel & 0x03; //extract 2 LSB from encoded data | |  |  | |  | listChar.add(temp); //add the extracted data to a queue for later processing | |  |  | |  | if(listChar.size() >=4) { //once we have 8 bits of data extracted | |  | //combine them to create a byte, and store this byte as a character | |  | charOut = (listChar.pop() << 6) | (listChar.pop() << 4) | (listChar.pop() << 2) | listChar.pop() ; | |  | decodedMsg.append((char)charOut); //else add character to a StringBuffer | |  | count++; | |  | } | |  | } | |  | } | |  |  | |  | String check = new String(decodedMsg); | |  | System.out.println(check + " " + check.length()); | |  | if (check.compareTo("!encoded!") == 0) { | |  | System.out.println("true"); | |  | return true; | |  | } else { | |  | return false; | |  | } | |  |  | |  | } //end of isEncoded() method | |  |  | |  | int getEncodedLength(BufferedImage input, int width, int height) { //method to get actual length of message encoded | |  |  | |  | StringBuffer decodedMsg = new StringBuffer(); | |  | Deque<Integer> listChar = new ArrayDeque<Integer>(); | |  |  | |  | int pixel, temp, charOut, count = 0; | |  | loop: for(int i = 0; i < width; i++) { | |  | for(int j = 0; j < height; j++) { | |  | if(count < 36) { //ignore the 36 bits or 9 bytes, equal to "!encoded!" | |  | count++; | |  | continue; | |  | } | |  |  | |  | pixel = input.getRGB(i, j); //grab RGB value at specified pixel | |  | temp = pixel & 0x03; //extract 2 LSB from encoded data | |  |  | |  | listChar.add(temp); //add the extracted data to a queue for later processing | |  |  | |  | if(listChar.size() >=4) { //once we have 8 bits of data extracted | |  | //combine them to create a byte, and store this byte as a character | |  | charOut = (listChar.pop() << 6) | (listChar.pop() << 4) | (listChar.pop() << 2) | listChar.pop() ; | |  | if((char)charOut == '!') { //terminate process if character extracted is '!' | |  | break loop; | |  | } else { | |  | decodedMsg.append((char)charOut); //else add character to a StringBuffer | |  | } | |  | } | |  | } | |  |  | |  | } | |  |  | |  | String length = new String(decodedMsg); | |  | System.out.println("length is " + Integer.parseInt(length)); | |  |  | |  | return Integer.parseInt(length); | |  | } //end of getEncodedLength() method | |  |  | |  | String getExt() { | |  | return ext; | |  | } | |  |  | |  | int getOffset() { | |  | return encodedMsgOffset; | |  | } | |  | } | | |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |

# SNAP SHOTS

** ** Figure. Login page Figure. Home page

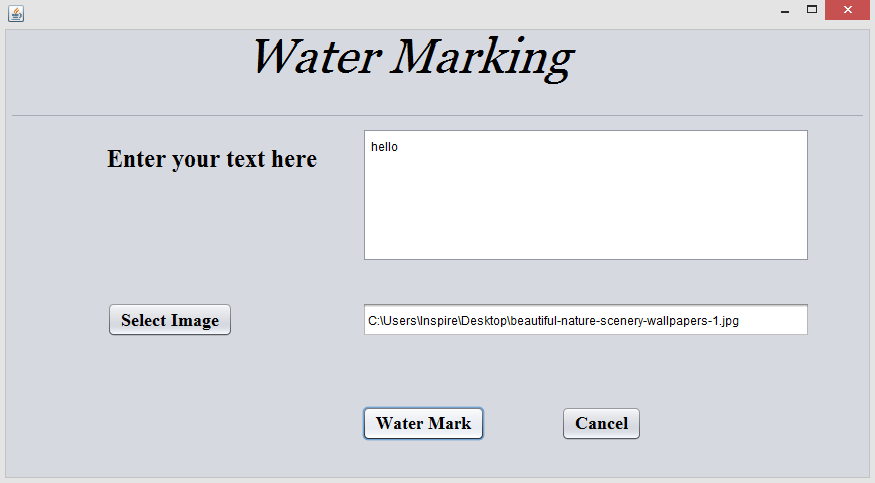
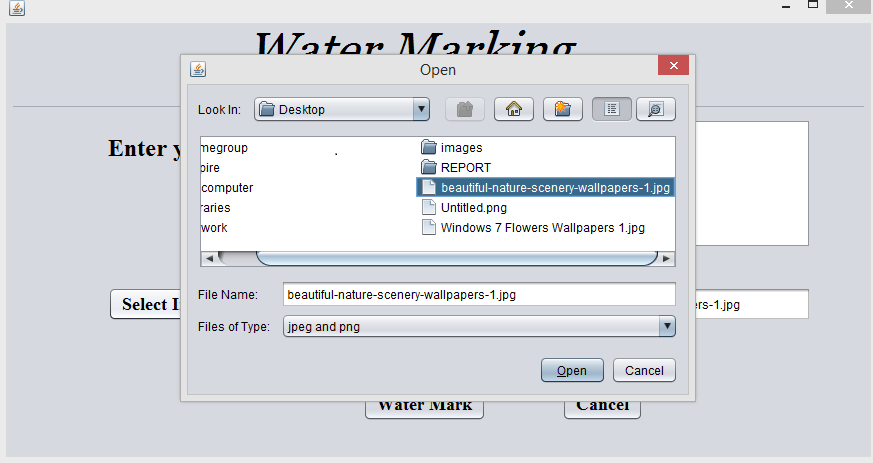
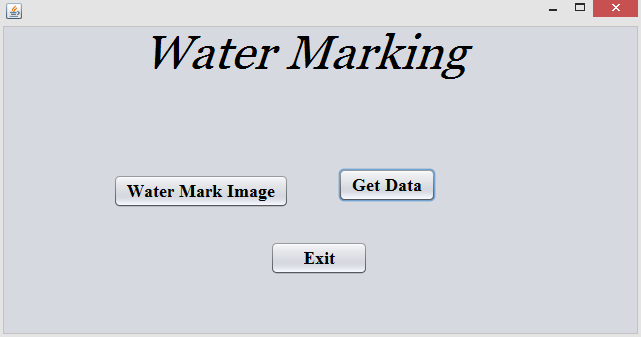
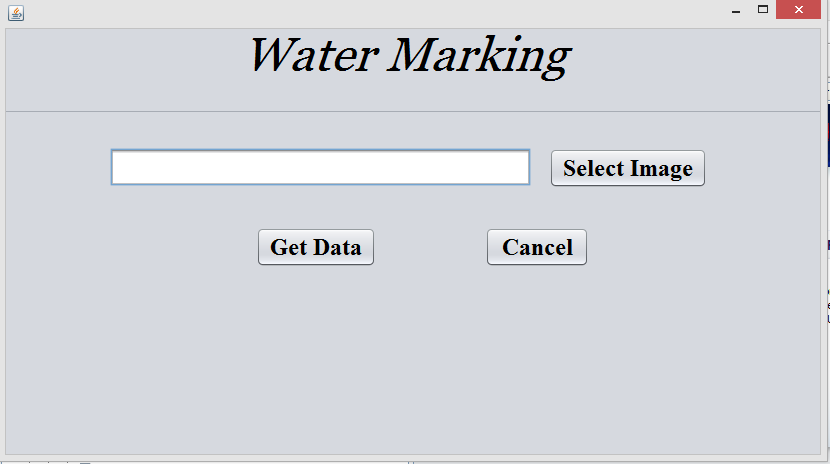
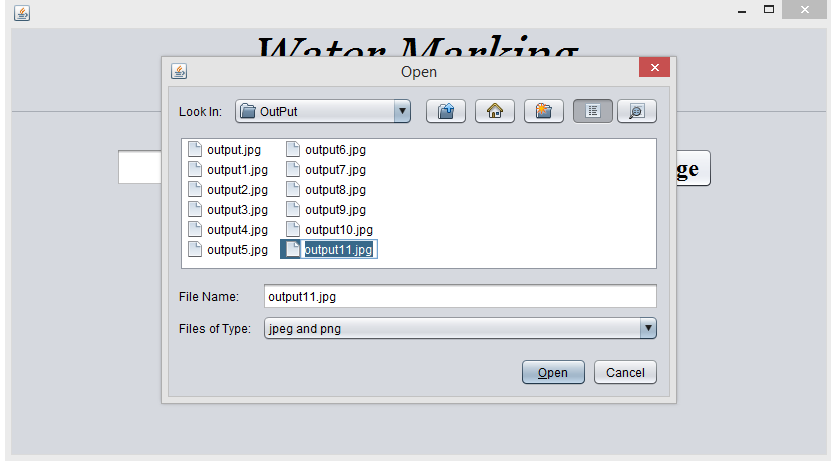
** **

Figure. Enter text Figure. Select image

** ** Figure. Watermarking completes Figure. Get data

  Figure. Select output image Figure. Open output image

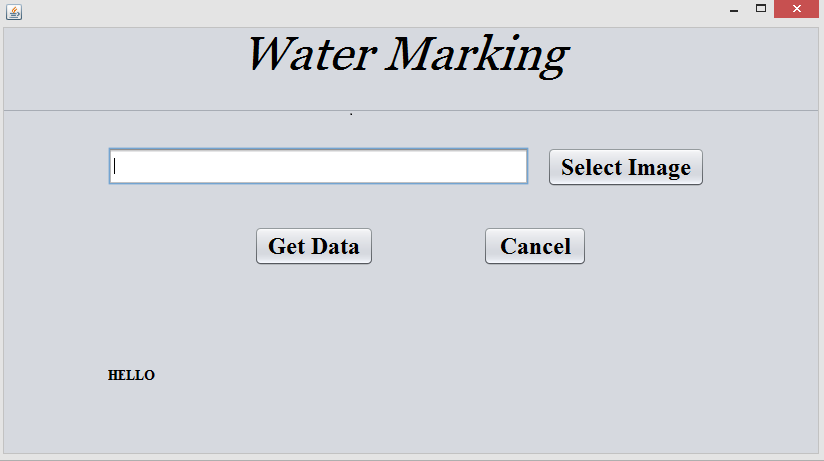
 ****

Figure. Decrypted process completes Figure. Get original message

# TEST PLAN

A test plan is a general document for the entire project that defines the scope, approach to be taken, and the schedule of testing as well as identifies the test items for the entire testing process and the personnel responsible for the different activities of testing. The test planning can be done well before the actual testing commences and can be done in parallel in the design and coding phase. The input for the test plan is:

1) Project Plan

2) Requirement Document

3) System Design Document.

The project plan is needed to make sure that the test plan can be consistent with the overall plan for the project and the testing schedule matches that of the project plan. The requirement documents and the design document are the basic documents used for selecting the test unit and the deciding the approaches to be used during testing. A test plan should contain following:

1) Test Unit Specification.

2) Features to be tested.

3) Approach for testing.

## TEST STATERGY

### BLACK BOX TESTING

In functional testing the internal logic of the system is not considered and the test cases are decided from the specifications or the requirements. It is often called “Black Box” testing.

### WHITE BOX TESTING

In structural testing, the test cases are decided entirely on the internal logic of the program or module being tested. The external specifications are not considered. In white box testing we check

1. Whether all the methods are implemented and working properly.

2. All the conditional statement is giving right output or not.

3. Whether the variable that is declared is reserved or not and also checked the scope of variable.

4. Whether all the control properties are used appropriately.

5. Proper validation of various fields is done or not.

### UNIT TESTING

The first level of testing is called unit testing. In this different modules are tested against the specifications produced during design for the modules. Unit testing is essentially for verification of code produced during the coding phase and hence the goal is to test internal logic of the modules. Each of the modules is tested independently.

### INTEGRATION TESTING

The next level of testing is often called integration testing. In this many unit tested modules are combined into sub systems which are then tested. The goal here is to see if the modules can be integrated properly.

## TEST CASES

### INTRODUCTION

The project is used to monitor the remote client machine. This allows the access remote network access to graphical desktop.

Following are the testing goals

1. Test the network is created.
2. Test the client machine is connected.
3. Test the performance of the system.
4. To ensure proper communication between the client and server.

### ITEMS TO BE TESTED

a) To test the GUI.

b) Test that client window sends image properly.

c) Test that server handles client efficiently

## FEATURES TO BE TESTED

### Performance

1. Must perform efficient connection.
2. Client and Server must perform proper communication.

### Functionality

a) Proper communication between the Client and Server.

b) Producing client window on the server screen.

c) The client sends the image properly to the server.

d) The server should receive the image properly.

e) After the client-server communication the connection is terminate.

## Item pass/fail criteria-Test Cases

**Table:** **Item pass/fail criteria**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Objective** | **Steps to Perform** | **Input Data** | **Expected Result** | **Actual result** | **Test Case**  **Pass/fail** | **Bug ID** |
| Login data authentication | To input the Username and password | 1)Enter username and password  2) Click on Login | Username and password | System must authenticate and login | Same as Expected Result. | Pass | - |
| Watermark image | To perform watermark on the source image. | 1) Select the grid.  2) Choose the watermark button. | Source image  Destination image  Grid. | Image watermark | Same as Expected Result. | Pass | - |
| Get data | To open watermark image | 1) Select watermark image. | Watermark image | Data must be get | Same as Expected Result. | Pass | - |
| Terminate | To exit the watermark | 1) Click on exit button. | N.A. | Watermark should be closed. | Watermark is closed. | Pass | - |

# FUTURE WORK

We propose the following functionality

1. In future, we are expecting the use of more intelligent statistical techniques in text watermarking.
2. Few possible extensions to our work that could be of immediate interest and may possibly have strong impact are given in the following:

* Hand written text, manual signatures, and fingerprints can also be taken as watermark and further experiments can be conducted on these.

Algorithms proposed in this dissertation can be further refined for specific text size and application e.g. text watermarking algorithm for a bank statement, passport text, emails, etc

# CONCLUSION

DRM systems and content management are important for protection of rights of digital multimedia creation that are distributed on the internet. Digital watermarking is an effective technique for embedding rights information in digital multimedia data.

Digital watermark technology can be used in consumer electronic devices like digital still camera, digital video camera, DVD players, MP3 players, etc., for various applications like providing controlled access, preventing illegal replication and watermark embedding