Self Embedding Watermarking System

High-Level and Low-Level Design

Project Advisor:

Dr. Asif Mahmood Gilani

Group Members:

Arbab Hamd Rizwan	18L-0756
Usama Aslam	18L-0972
Aashar Naseem	18L-1131
M. Hunzlah Malik	18L-1139

National University of Computer and Emerging Sciences

Department of Computer Science

Lahore, Pakistan

Table of Contents

Table of Contents

List of Tables

List of Figures

1. Introduction

Editing software has come a long way in the last decade. As a result, the general public now has access to complex editing tools, making it easier to manipulate digital assets like photographs and movies. This casts considerable question on the credibility of any media offered. Image tampering is primarily concerned with changing the image as a whole. Video, on the other hand, is made up of multiple frames that can be regarded as individual images.

Therefore, video tampering has been categorized into two types: [1]

1. Temporal tampering refers to interframe editing manipulation. This type of tampering includes adding, removing, or changing the sequence of frames in a video.

2. Spatial tampering includes manipulating objects within a frame and is referred to as intraframe manipulation.

Watermarking is a technique that can be used to detect tampering. Our project will be focused on detecting and restoring these altered images and videos. Watermarking images entails breaking them down into smaller chunks and inserting data into the LSB, which can then be used to detect alteration and restore the original image. Numerous approaches are investigated in research publications, varying depending on the number of blocks the image is divided into and the watermarking method [2 - 4]. By first detecting the keyframes and then processing those frames as if they were images, this approach can also be utilized on video [1].

1.1 Purpose of this Document

This project aims to successfully detect tampering in any image or video and reconstruct them to their original state. We have made extensive use of Digital Image Processing (DIP) concepts in this project. This will be helpful in many areas, such as copyright infringement of content on social media. Moreover, this project will also be an essential asset in law enforcement agencies where CCTV footage is presented as evidence. Our project can help place watermarks on such footage and will successfully detect any tampering done by some third party to erase any concrete evidence from the footage. We have studied various watermarking algorithms from research papers mentioned in the reference section. However, the main watermarking algorithms used will include the use MD5 hashing function [5] to self-embed the watermark into an image.

Additionally, we have made use of this same methodology to watermark videos. The main idea is to identify the keyframes from a window of frames and watermark that frame according to the same algorithm used to watermark the image [1]. We will perform watermarking methods on pictures and videos to later be detected for any tampering through our system. In case of any tampering, the watermarked image or video will be recovered depending on the percentage of tampering.

1.2 Scope

The goal of this project is to create a complete web application that can perform watermarking, image and video tamper detection, and finally image and video tamper restoration using digital image processing and artificial intelligence. Furthermore, the function's scripts will be executed on the server, ensuring that the application is accessible 24 hours a day, seven days a week. The application will be scalable and efficient, requiring less maintenance in the future, and we will be able to improve it.

1.3 Intended Audience

This document is intended for our team of developers, the FYP supervisor, the evaluation team, and the FYP coordinator. All of the people mentioned above are the document's intended audience because they are all involved in this project.

1.4 Version

The older version of this document was referred to as functional specification document.

Table 1: Document Version

Following table shows previous document version along with its authors

Date	Version	Description	Authors
December 15, 2021	1.0	High-Level and Low-Level Design 1.0	• Arbab Hamd Rizwan • Usama Aslam • Aashar Naseer • M. Hunzlah Malik

1.5 Related Documents

This project's related documents include a function specification and methodology document, as well as a software requirement specification document. The first document describes the system's functionalities, requirements, system architecture, and graphical user interface. It is also mentioned how the functionalities will be implemented. The second document, on the other hand, is primarily concerned with the system's requirements, which include both software and hardware requirements.

1.6 Prerequisite Documents

The project's prerequisite documents include a function specification, a methodology document, and a software requirement specification document. The system's functionalities, requirements, system architecture, graphical user interface, and risk analysis are all described in the first document. It is also stated how the features will be implemented. The second document, on the other hand, is primarily concerned with system requirements, which include both software and hardware requirements, as well as the system description, which includes both domain overview and user characteristics.

1.7 Background of this Document

The background of the document contains the information on which this document is built, so the main features in this project are media watermarking, media tamper detection, and finally media restoration. The user provides media via the upload and download options and retrieves watermarked or restored media via the download option. As a result, the requirements remain unchanged because no change request has been made. If a change request is made, it will be mentioned in the next version of the document, along with the background.

1.8 Definitions, Acronyms, and Abbreviations

CZ refers to the central cropping attack on an image (center of image) . VCZ means vertical center attack on an image. Square image refers to an n×n dimension image where n is any positive number. For example, 256×256 or 512×512 image dimensions have been used for experimental setup. Image tampering includes cropping attacks and editing attacks on an image. Video tampering includes object removal attacks on video frames. Key frames are selected from a window of frames which have the most amount of information of that specific window of frames. Authentication refers to the detection of multimedia tampering, which may include various attacks such as cropping attacks, changing pixels etc. and it does not deal with ownership authentication. Fragile watermarking is one of the integrity protection mechanisms of multimedia data like image, video, and audio.

1.9 Summary

Multimedia tampering can be categorized into two parts i.e., image tampering and video tampering. Image tampering is primarily concerned with changing the image as a whole. Video, on the other hand, is made up of multiple frames that can be regarded as individual images. Our project makes use of various algorithms and works done in the field of multimedia authentication and restoration and implements the one with best results. Our project will have three main functionalities for each module i.e., image module and video module. This would include authentication and recovery bits embedding process, multimedia tamper detection process and restoration of tampered multimedia. Our project will be a web application which will handle all processing load on server side, this is due to the reason that not all users may have a high-end CPU to process faster. Therefore, it will follow a client server architecture where a single server will deal with multiple clients. To accomplish this task, the server will be developed so that it can concurrently handle multiple users at the same time and reduce user queueing delay. Moreover, all of the data will be embedded in the multimedia uploaded by the users, therefore there will be no use of a database to store any kind of information. This web-based project will be quite useful in various fields ranging from content copyrights to validation of multimedia content provided as proofs against criminal activities.

2. System Overview

The project's system overview, which includes the number of functionalities in the project as well as the design, which explains how the system will be built, and finally the system's working, which explains how the user will use the system as well as what choices the system will give, what functions the system will perform, and in what order the functions are performed, are provided below.

2.1 Functionalities

The Self Embedding Watermarking System performs three primary functions: media watermarking, media tamper detection, and media restoration. The media in this case includes both images and videos. The fourth function which is upload and download is secondary but still important because it assists in obtaining media from the user and delivering watermarked and recovered media to the user. The detailed functionalities of the system are listed below.

2.1.1 Watermarking

Watermarking is the main feature of our system in which the media is watermarked the system by embedding the authentication bits, which are created by inputting the image's bits into the MD5 hash function, and its hash value is the bits embedded into the media's LSB. The embedded bits are used by the system's tamper detection. [1]

2.1.2 Tamper Detection

In this feature, the system first determines whether the media uploaded is watermarked or not; if it is not, the system displays an error message to the user; if it is, the system first extracts the authentication bits inserted into the media and creates a hash of the image's bits before comparing the hash value to the extracted authentication bits. If they are the same, it indicates that the image has not been tampered with; however, if the values are not the same, it indicates that the image has been corrupted.

2.1.3 Media Restoration

The system first confirms whether the media uploaded is watermarked or not; if it is not, an error message is displayed to the user. In case the uploaded media has been watermarked before by our system, it will first extract the recovery bits inserted into the media. The system will then use these recovery bits to reconstruct the media.

2.1.4 Upload and Download

This feature allows the user to upload a media file (image or video) to the web application which will be sent to the server through backend APIs. Once the server has finished handling users request, the server will send a file in same format to the user. This file could either be watermarked multimedia or recovered multimedia depending upon the user's request. The user will then be allowed to download either the watermarked, or recovered media.

2.2 System Working

The system works as follows:

- It provides the user with three options for both image and video which are watermarking, authentication, and reconstruction.
- The system allows the user to upload an image or video, after which the user can select whether he only wants to use tamper detection, watermark an image or video, or restore a watermarked image or video.
- If a user chooses to watermark an image or video, the media is watermarked and the user is given the option to download the watermarked content. Second, if the user only wants to detect media tampering, a new tab is opened and the tampering results are displayed.
- Finally, if the user wants to restore a watermarked media, he or she should upload the watermarked media, and the system will restore the media and open another tab with the option to download the restored media.
- In a nutshell, our system allows users to watermark a video or image, authenticate a video or image, and finally restore a video or image to its original state.

3. Design Considerations

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution. These issues include assumptions and dependencies along with general constraints of the system.

3.1 Assumptions and Dependencies

3.1.1 Assumptions

We assume the following scenarios when a user is interacting with our system:

- On multimedia tamper detection or restoration requests, it is assumed that the user has already watermarked that multimedia through our system.
- Uploaded media is within the upload limit defined by the server.
- Multimedia uploaded for watermarking is original i.e., it has not been tampered before the watermarking process.

3.1.2 Dependencies

Our system has the following dependencies:

- Since all the data is embedded in the digital multimedia, there is no need for a database to store any kind of information regarding the file user uploads to the server.
- Multimedia tamper detection is dependent on the fact that it should be watermarked through our system beforehand.
- Similarly, multimedia recovery also depends whether it was watermarked by our system or not.
- Multimedia recovery has an upper limit depending on the type of multimedia.
    - For image, it is approximately 75%.
    - For video, approximately 67.5% of tampered video can be recovered.
- Minimum dimension of image is 512×512.
- Multimedia processing may require a lot of resources, therefore, a good server with high end GPU would be needed to run scripts on backend.

3.2 General Constraints

Following are the dependencies of the system:

- Since the system requires high computing power, we would need a server with a good GPU.
- Most of the servers which have good GPUs are paid servers, the free ones have various limitations.

4. Goals and Guidelines

Our project has an important role in-terms of security and content copyright protection of multimedia content. There it is important to have well defined goals and guidelines, which are as follows:

- Providing the ability to recover as much of the multimedia content as possible. Currently, for image recovery, the upper limit is 75% of tampered images in the best-case scenario. Whereas, for video it is approximately 67% recovery of tampered video (see Appendix A) [1]. These can vary according to the sections in which the tamper attack was applied. Therefore, our main goal is to increase the recovery percentage as much as possible.

- The high recovery rate of multimedia content may sound very useful and fascinating on paper, but it comes with a major issue and that is the processing time. Currently, it is approximately 180 seconds [1] (see Appendix A) for the whole process of embedding recovery and authentication bits along with recovering an image. This could increase even more in case of video; hence our second goal would be to reduce processing time. One of the solutions for this can be the use of multi-threading to run tasks in parallel.

5. Development Methods

This section describes the methods that were studied and the methods that will be used to implement this project. Moreover, this section also includes the methods that were considered but were not used due to reasons mentioned in the later subsections.

5.1 Algorithms

Various algorithms were researched and taken into consideration [1-3] before finalizing on one algorithm to implement. The mentioned algorithm uses triple recovery for effective authentication and recovery. Moreover, there was one algorithm which also generated good results while being efficient but was not taken into account due to its lower rate of restoration as compared to triple recovery method

5.1.1 Image Authentication and Restoration

The algorithm used to authenticate the image and reconstruct it is from [1]. Let's first talk about the authentication of the image. The algorithm does it in such a way that it first divides the image into 16 equal blocks then makes a lookup table to select the partner block in each region. Then it divides the 16 main blocks into 4x4 blocks and takes the average of the bits of each image block. Then these average bits of the partner blocks are combined and hashed to generate keys. Then each partner block is divided into 16x16 blocks and then these divided blocks are further divided into 8x8 blocks then the key generated is inserted into the first and second LSB. Then the whole image is combined together and used wherever the user wants. Now to authenticate the algorithm first extract the authentication bits from the image by dividing it the same as above as computing the has again if the computed hash and extracted hash are same then image has not been tampered else it has been tampered. Similarly, the recovery bits are also added into the image to watermark it. Then, on recovery the system first checks whether the image has been tampered or not, if it has been tampered then it divides the image into the same block as above and then it combines the recovery bits extracted from the image and permutates recovery bits with the help of key. Then it first recovers 4x4 blocks and then it recovers 16x16 blocks then it replaces the recovery block with the tamped block and combines the whole image.

5.1.2 Video Authentication and Restoration

This module makes extensive use of image modules for authentication and restoration. Video has a large number of frames that can be treated the same way as images. Moreover, these frames have a few specific frames with most information in a window of frames. These frames are referred to as keyframes and these frames go through the process of embedding. Later on, these extracted keyframes are used for authentication and restoration of a frame window.

5.1.3 Pixel-wise Authentication and Recovery

This method is the latest research in this field, which made use of pixel wise processing unlike other methods that relied on block-wise image authentication and restoration. This method is more efficient in terms of tamper detection and processes on a smaller area (pixels) as compared to block wise image authentication which processes the whole image block even if a single pixel was tampered. However, this method could not yield as great a restoration percentage as compared to the triple recovery method which is why this method was not used as a development method for this project. [2]

5.2 Development Methodology Used

Upon looking into various software development models [4], we decided to follow the scrum methodology, which is based on iterative and incremental process, for the implementation of this project. According to the mentioned timeline of modules, we have divided the project in various phases which will be implemented in a series of sprints and will later undergo through a process of various test cases [5]. Using this method, we will be able to develop and test the system in small pieces and then at the end of the whole process, it will be integrated at server end and the finished web application will be deployed on a selected server.

6. Architectural Strategies

Following architectural strategies will be followed to implement this project.

6.1 External Dependencies

Our system will be a web application so it will have a backend server along with the use of APIs. For the use of APIs, we will make use of REST APIs, which are implemented using NodeJS, to connect backend with our frontend. For now, we have not finalized which server we will be using; however, the python scripts of our system will be running on the server side. Since our system requires more resources to embed authentication and recovery bits, along with the process of authentication and restoration, therefore, it is important to select a suitable server. In our case, it could either be a local server or a web server with adequate resources.

6.2 React, Node and Python

When it comes to image analysis and processing MATLAB is considered the best language, but the drawback of using MATLAB is that it is difficult with MATLAB when it comes to web-based projects. The second-best option in this regard is Python, it is easy to code and due to its famous libraries like OpenCV, scikit-image, etc. It is very easy and efficient to work with Python in the field of image processing. Another benefit of Python includes the ease of running scripts on web-based applications. Therefore, an image processing module for our system will be developed using Python. Moving on the front-end development of our project we will be using React JS, because React is a very popular and globally recognized JavaScript library. React is fast to use and easy to code as well due to the reusability of its components. Due to this reason, our backend development will be implemented using NodeJS.

6.3 Concurrency

We will implement server concurrency so that it can handle multiple users at a time. However, this will also split the server resources and might turn into a drawback for our system which requires more computing power to process the multimedia uploaded by the users. Consequently, we will have to put a limit to the number of users that can access our services at once. This limit is for the initial stage of our system and can be later extended to handle more users at a time by extending server resources. Moving on to the implementation of our system, it will also require a certain degree of concurrency so that embedding, authentication and restoration processes can execute faster. This can be accomplished by using multi-threading and fine graining our problem into smaller independent parts that can be executed in parallel.

6.4 Future Goals

Image and video tamper detection and content reconstruction systems are considered as one of the needs of the digital world. We came across various cases of image and video tampering which can cause defamation or false acquisitions, although we are developing a software to overcome these types of tampering cases but there is a chance of improvement as well. Our software can only detect tampering in those images and videos which are watermarked through our software because we are using active watermarking which means we place data in the media and authenticate using that data, so our future goal will be to implement passive watermarking [6] that is implemented using neural networks which will be able to detect tampering in image and video without placing any data inside the multimedia content. This ensures that the quality of multimedia content is not compromised by embedding more data into its LSB.

6.5 User Interface Paradigms

The UI for our website has been designed in a way that is easy to navigate and simple to use. The eight golden rules for interface designing, by Ben Schneiderman, would be utilized which are part of the standards of human computer interaction.

6.6 Database

We have decided not to implement a database for our system due to the working principle of our system, which embeds the required data inside the multimedia. This leaves no use of placing any data for the uploaded multimedia in a database. The embedded data is retrieved from the multimedia itself and its authentication bits are extracted and are compared to check for any tampering. In case of tampering, the recovery bits are then extracted from the multimedia and are used for restoration. In this whole process, there is no requirement for a database which would store user information.

7. System Architecture

This section describes the system architecture of our project.

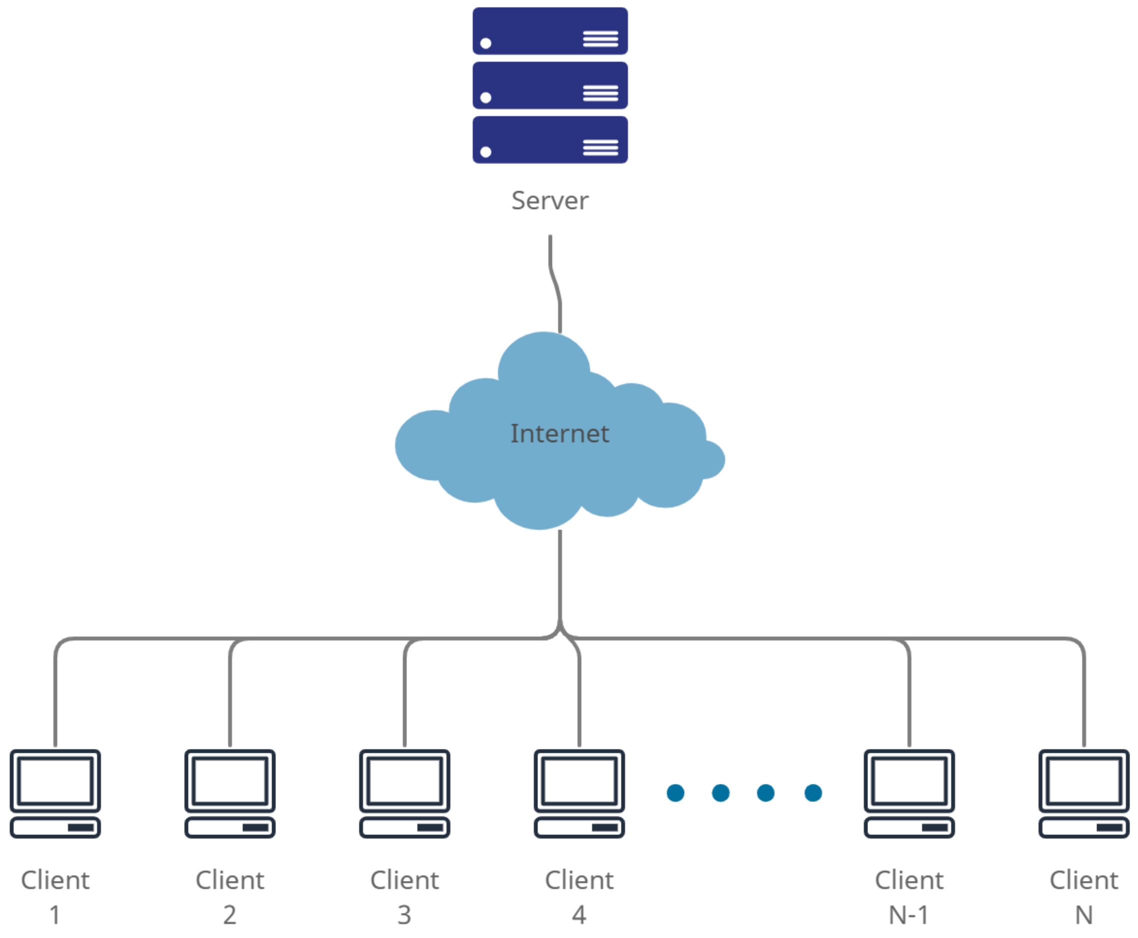


Figure 1: High Level System Architecture

Figure shows high level architecture of the system

7.1 Image Authentication Module

This is the main module which will help in authentication and restoration of watermarked images. This module will also be reused in the video authentication module. This is due to the fact that video keyframes will also be considered as images and will be processed through this module to watermark keyframes of a video. This module has two sub modules which deal with authentication bits embedding whereas, the other module deals with recovery bits embedding.

7.1.1 Authentication Bits Embedding

Once an image has been divided into the required blocks according to our algorithm, the images will then be embedded with authentication bits. This process will be carried out by taking LSB of the divide blocks, taking hash value of those blocks and then storing that hash value inside the LSBs of some other image block.

7.1.2 Recovery Bits Embedding

This sub module will deal with embedding recovery bits into an image. This will be carried out by taking the mean of LSBs of a specific block and storing it in other blocks. Upon restoration, these bits will be extracted and will be placed back into the original block.

7.2 Video Authentication Module

The video Authentication Model is mainly concerned with the authentication of video, in this module first the key frames are extracted and these frames are images as well so they are treated in the same way an image is authenticate That's why we also use image authentication module in this part because it treated keyframes same as image and detect the tampering.

7.2.1 Keyframe Extraction

In both the video authentication and Recovery key frames are of most importance because both the recovery and authentication bits are added in these frames. So, a first thing to do when either authenticating a video or reconstructing video first thing to do is to extract key frames from the video because the size of video is very large and processing it takes a lot of time so, in order to reduce the time but keeping the efficiency the key frames are used instead if the whole video. [7]

7.2.2 Keyframe Authentication Bit Embedding

In this module the first thing to do is to extract the key frames from the video and then traits each key frame as an image. The system then divides the frame into blocks according to the algorithm and then takes a hash of the bits in the block. Then the hash value is inserted into the key frames which will be used to authenticate the video in the future.

7.3 Image Recovery Module

In this module the first thing the system does is to divide the image into blocks and extract the recovery bits embedded into the image using the algorithm mentioned above then. then it uses the authentication bits embedded in the image to test its authenticity. If image has been altered then it uses the recovery bits to permute the block of image to the whole image.

7.4 Video Recovery Module

In this module the first thing the algorithm does is not extract the key frames from the video and then divide the video into frames. Then it localizes the tampered region in the key frame and checks whether frames have been inserted or deleted from the video. The frames are checked by making sure they are correctly indexed. If the frames are not correctly indexed it means the video has been corrupted and the recovery bits are extracted from the key frames to reconstruct the image.

8. Class Diagram

This section shows the class diagram of the system.

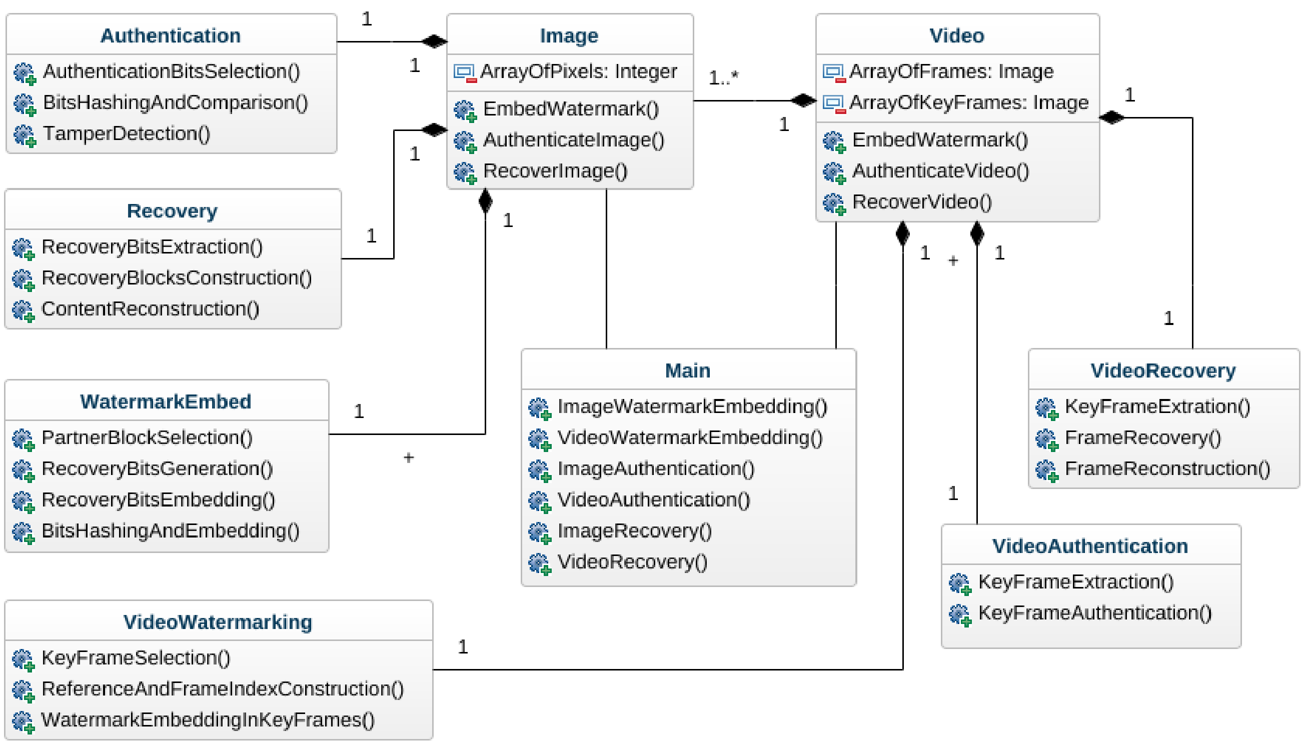


Figure 2: Class Diagram

Figure shows class diagram of the system

9. Sequence Diagrams

Following are the sequence diagrams for the system.

9.1 Image Authentication

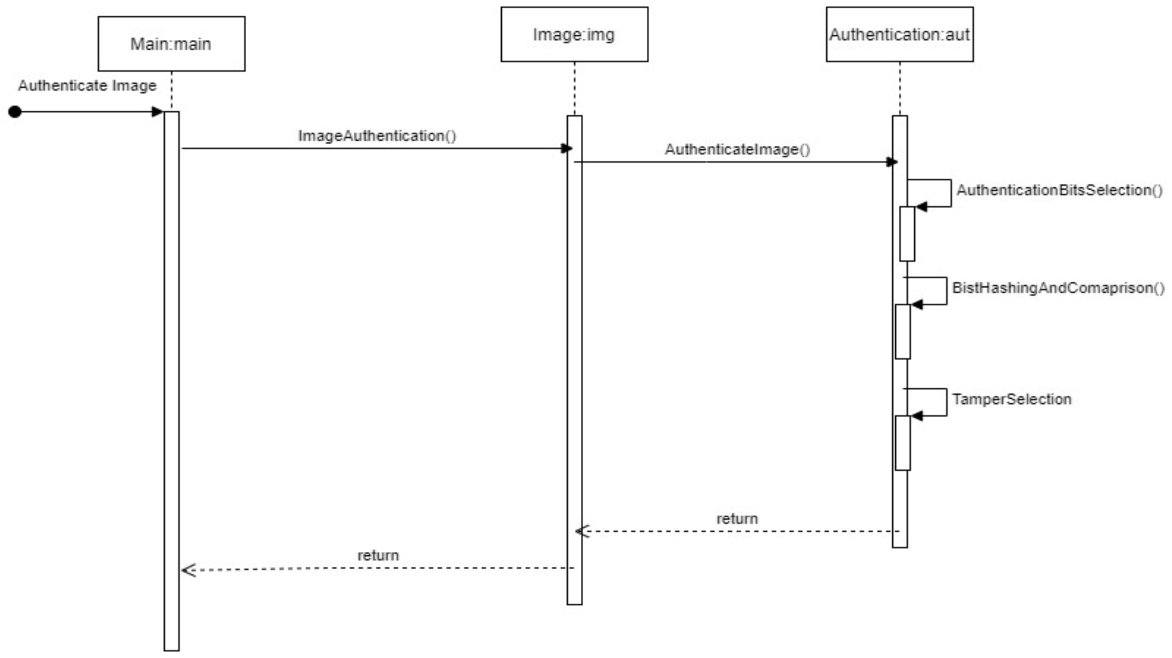


Figure 3: Image Authentication Sequence Diagram

Figure shows the sequence diagram for image authentication

Sequence Diagrams

9.2 Video Authentication

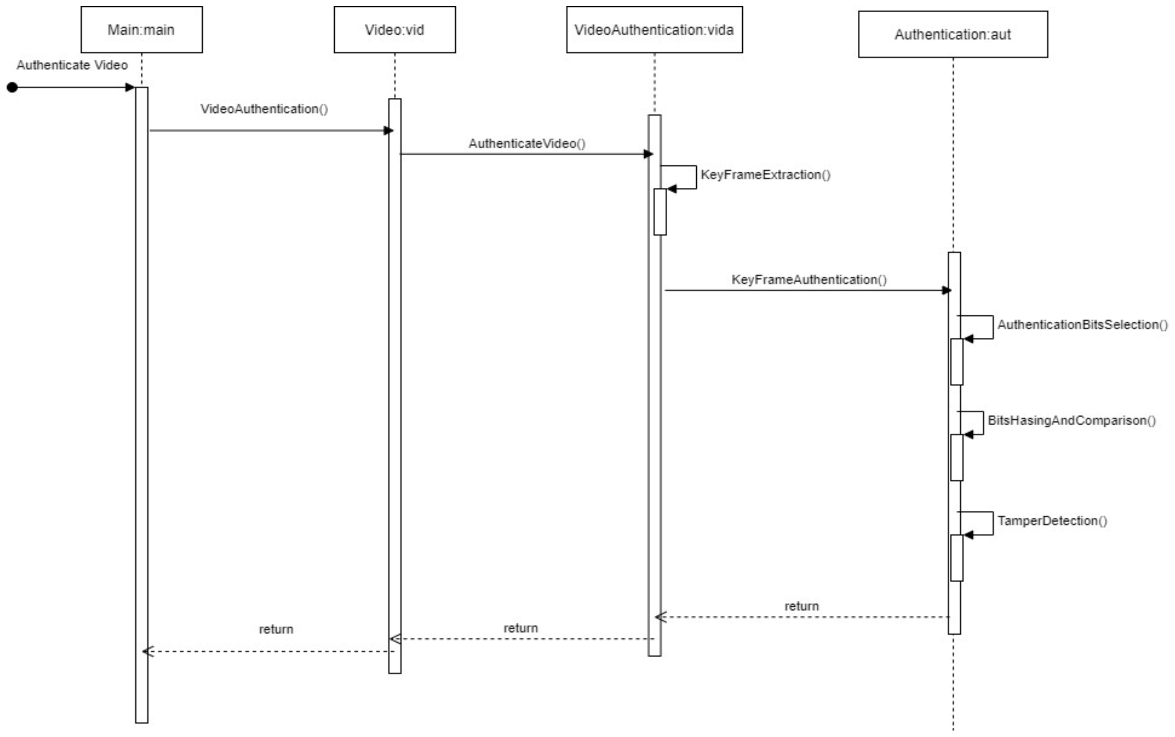


Figure 4: Video Authentication Sequence Diagram

Figure shows the sequence diagram of video authentication

9.3 Image Restoration

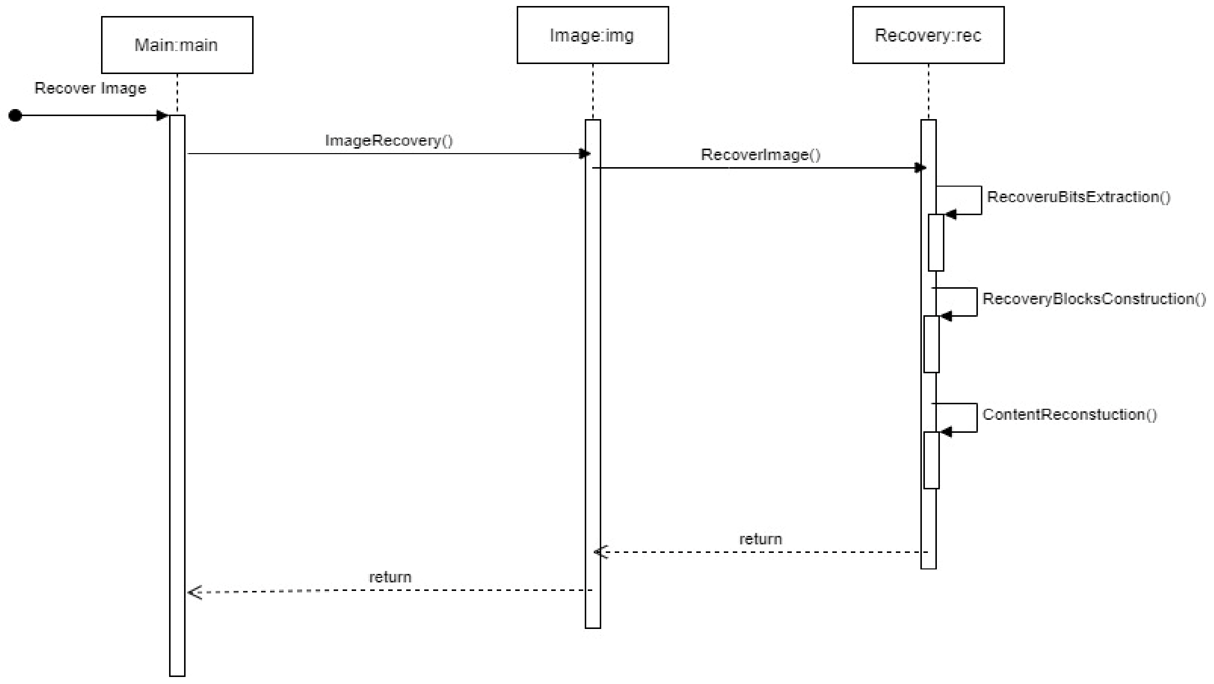


Figure 5: Image Restoration Sequence Diagram

Figure shows sequence diagram for image restoration

9.4 Video Restoration

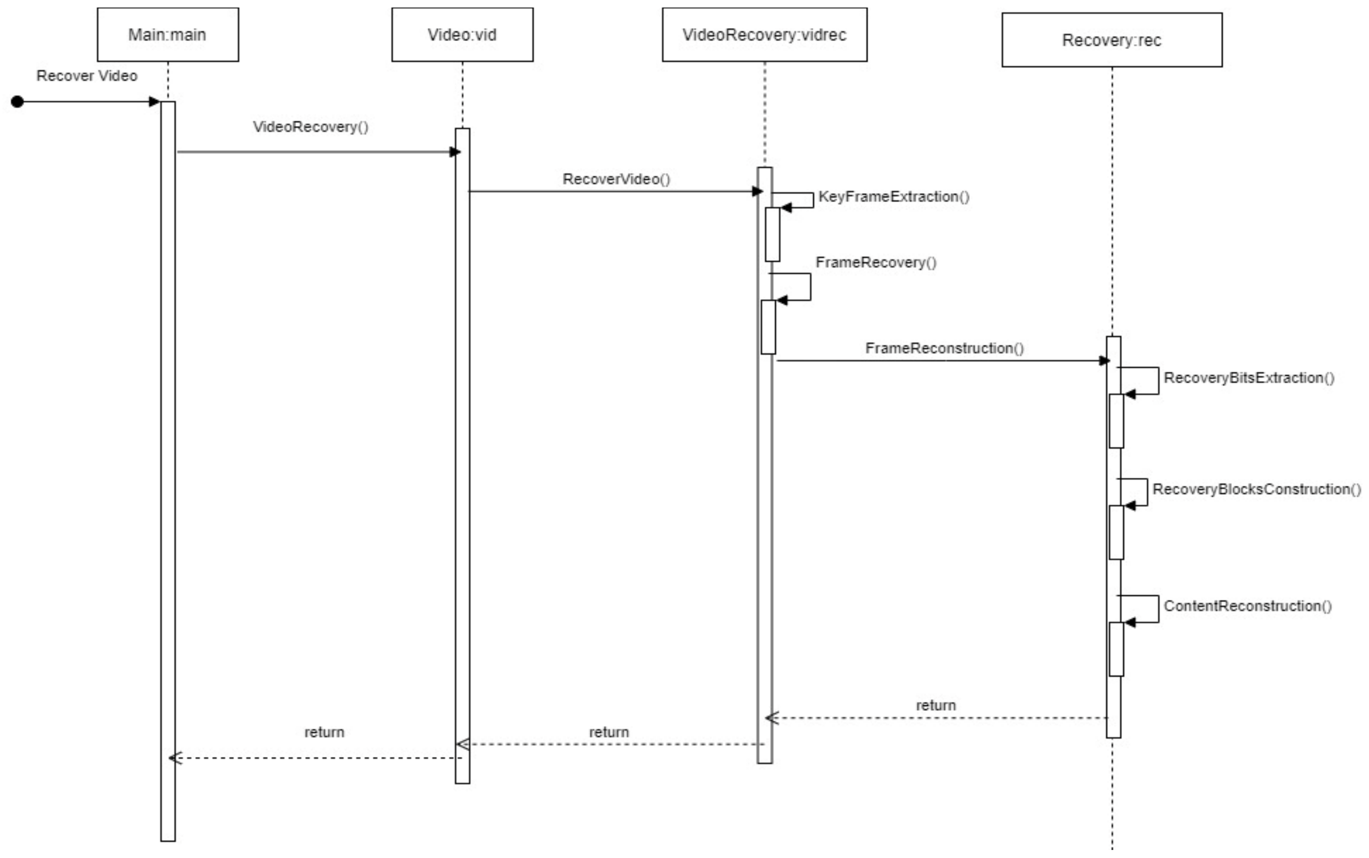


Figure 6: Video Restoration Sequence Diagram

Figure shows sequence diagram for video restoration

Sequence Diagrams

9.5 Image Watermarking

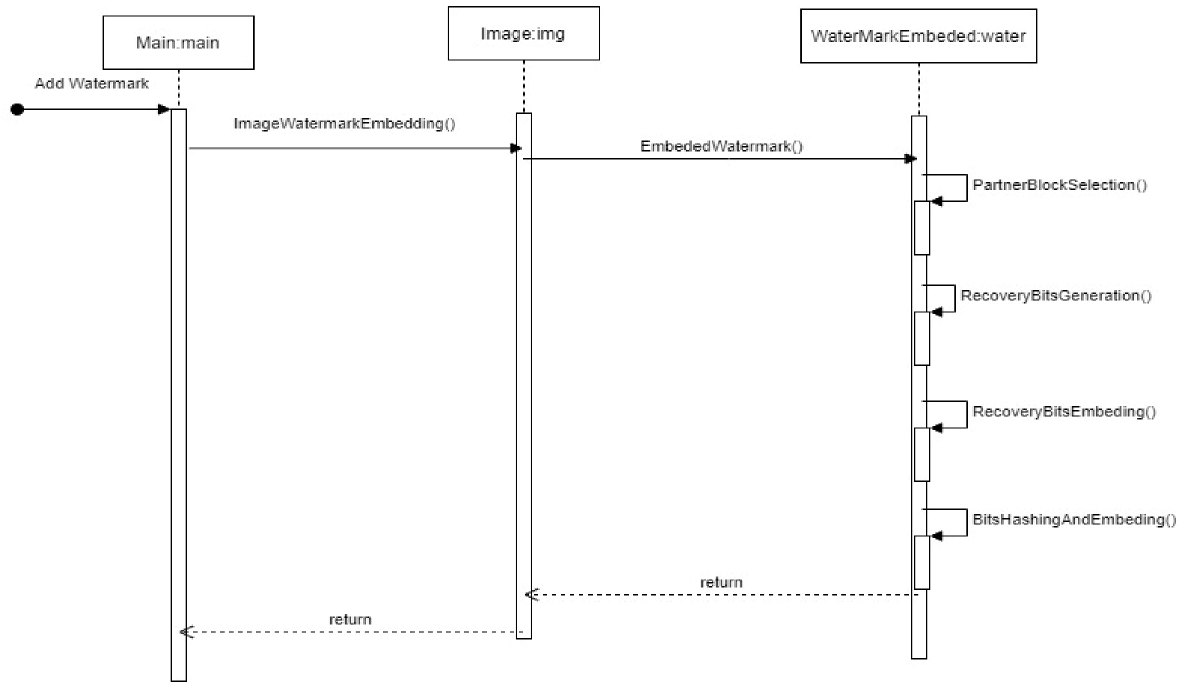


Figure 7: Image Watermarking Sequence Diagram

Figure shows sequence diagram for image watermarking

9.6 Video Watermarking

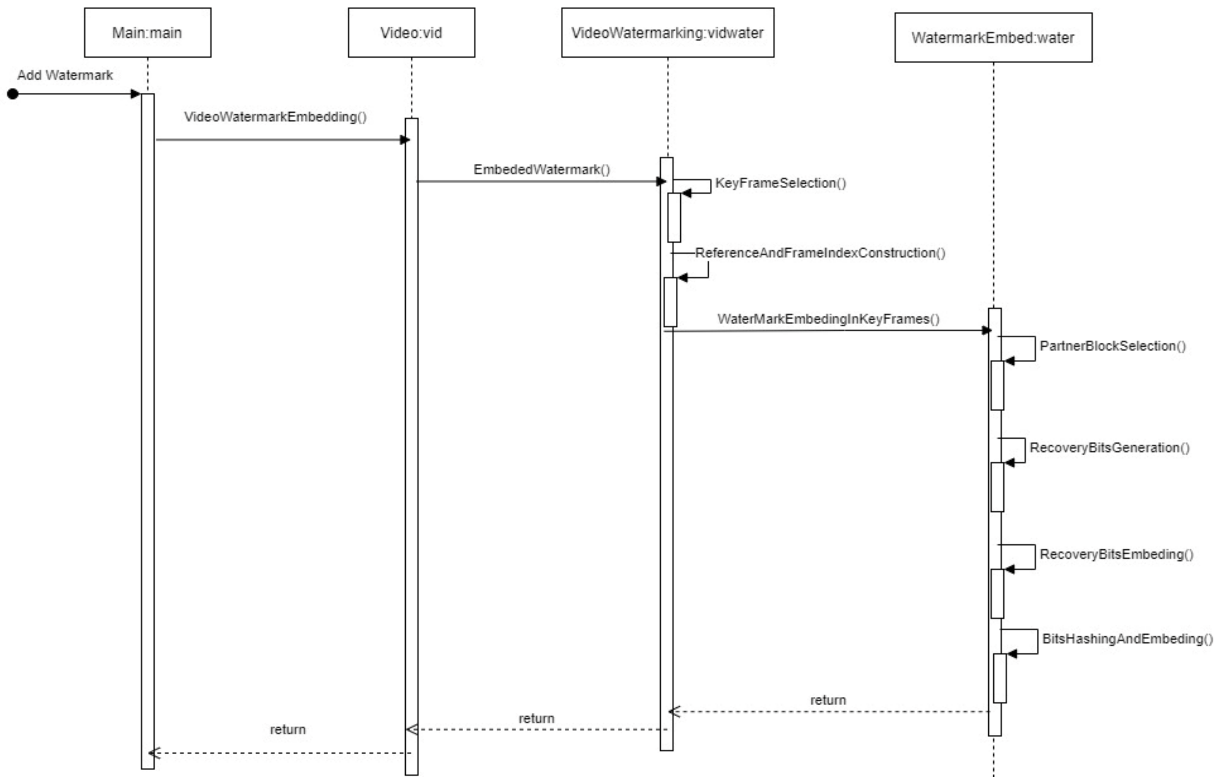


Figure 8: Video Watermarking Sequence Diagram

Figure shows sequence diagram for video watermarking

Sequence Diagrams

10. Policies and Tactics

10.1 Tools to be used

Visual studio code will be used for both frontend and backend development. Here the backend development means the multimedia processing scripts that will be running on the server side. We will use NodeJS for APIs to communicate with the server so that it can execute user requests. Python scripts will be tested on Google Colab because it has GPU sources integrated.

10.2 Policy for user interface

A basic interface has been selected that shows the user all available functionalities of our system in one place. The user can choose any of the functionalities with ease by just clicking once.

10.3 Coding guidelines

The system will be implemented while following the coding standards, along with a well-documented code. All the coding standards will be followed to increase readability of the system code.

10.4 Plans for using latest technologies

We plan to use the latest technologies such as ReactJS for frontend and NodeJS for API calls. Whereas we will be using the latest version of python to create multimedia processing scripts.

10.5 Policy for software testing

Along with a full test of the system, we will do white box testing and black box testing. For white box testing, we will test the system using control flow and data flow testing. For black box testing, we will do unit testing and integration testing. In case there is an issue with testing we will also do a bit of regression testing.

10.6 Accessing the system

Since our system would be hosted on a server, it will be accessible through a URL. This will allow any computer (even the ones with lower specifications) to access the system and use its functionalities easily without any burden on their computer.

10.7 Policy for system maintenance

We will have to make regular maintenance checks depending on the amount of user requests at a given time. Since our system relies on computing power, if the server is busy handling one user, the other users would end up in a waiting queue. Therefore, the system would require regular maintenance checks to see if it requires more resources or not.

References

[1] Gul, E., Ozturk, S. A novel triple recovery information embedding approach for self-embedded digital image watermarking. Multimed Tools Appl 79, 31239–31264 (2020). https://doi.org/10.1007/s11042-020-09548-4

[2] Gul, E., Ozturk, S. A novel pixel-wise authentication-based self-embedding fragile watermarking method. Multimedia Systems 27, 531–545 (2021). https://doi.org/10.1007/s00530-021-00751-3

[3] Gul, E., Ozturk, S. A novel hash function based fragile watermarking method for image integrity. Multimed Tools Appl 78, 17701–17718 (2019).

https://doi.org/10.1007/s11042-018-7084-0

[4] Team, S., Parekh, J., Team, S., Tolley, S. and Cipot, B., 2021. Top 4 software development methodologies | Synopsys. [online] Software Integrity Blog. Available at: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies [Accessed 13 December 2021].

[5] Digite.com. 2021. What Is Scrum Methodology? & Scrum Project Management. [online] Available at: https://www.digite.com/agile/scrum-methodology [Accessed 13 December 2021].

[6] Q. Yang, D. Yu, Z. Zhang, Y. Yao and L. Chen, "Spatiotemporal Trident Networks: Detection and Localization of Object Removal Tampering in Video Passive Forensics," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 10, pp. 4131-4144, Oct. 2021, doi: 10.1109/TCSVT.2020.3046240.

[7] V. Amanipour and S. Ghaemmaghami, "Video-Tampering Detection and Content Reconstruction via Self-Embedding," in IEEE Transactions on Instrumentation and Measurement, vol. 67, no. 3, pp. 505-515, March 2018, doi: 10.1109/TIM.2017.2777620.

Appendix

Appendix A: Time Complexity

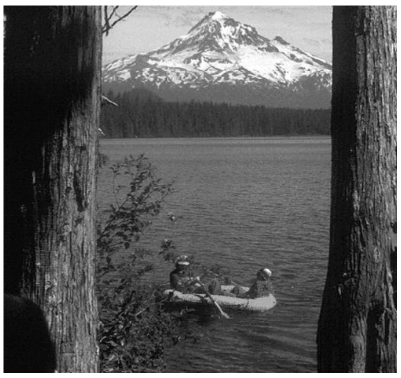
Table 2: Time Complexity for Watermarking

Following table shows time complexity for watermark embedding process

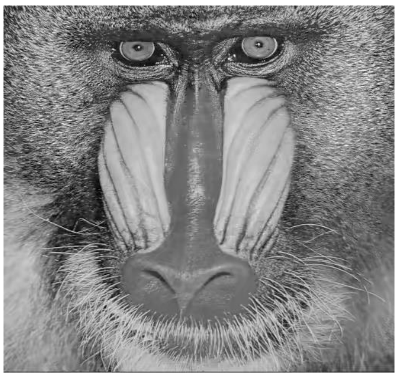
Image	Watermark embedding time (sec)		
	Recovery bits embedding	Authentication bits embedding	Total watermark embedding
Lena	133.128627	48.839553	179.968180
Sailboat	132.485935	46.644259	179.130194
Lake	132.879485	46.652739	179.532224
Airplane	132.357713	46.389058	178.746771
Baboon	133.314231	46.128864	179.443095
Goodhill	133.624994	46.151119	46.151119

Time complexity for watermark embedding is given in table 1. The images mentioned in the table are given below. All the mentioned images are square images, i.e., they have n×n dimensions. For the images mentioned in the table, the dimensions are 512×512. The computing time for watermark embedding depends on the size of the image. This is due to the whole image being divided into various blocks. These experiments have been conducted on a CPU Intel Core i5-7500 3.4 GHz with 4GB RAM and a GTX 1050 ti GPU [1]. Hence the processing time might be reduced depending on a higher specs CPU and GPU.

Appendix



(a)



(b)

(c)

(d)

(e)

(f)

Figure 9: (a) Lake, (b) Baboon, (c) Lena, (d) Goodhill, (e) Sailboat, (f) Airplane

Figure shows the square images used for experimental setup

For the process of recovering these watermarked images, the time complexity varies according to the amount of tampering done in the image. The details are mentioned in table 2. Maximum limit for image recovery is 75% which takes the longest time to recover.

Table 3: Time Complexity for Restoration

Following table shows the time complexity for image recovering process

Image	Tamper detection and recovery time (sec)		
	25% CZ	50% VCZ	75% CZ
Lena	180.296284	189.142496	198.514490
Sailboat	180.821834	190.707774	199.018766
Lake	179.811014	189.824777	199.324460
Airplane	180.477713	188.998814	198.762606
Baboon	180.199268	189.820586	198.406937
Goodhill	180.330294	189.730310	189.730310