



**National University of Computer and Emerging Sciences**



## **Self-Embedding Watermarking System**

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

Supervisor: Dr. Asif Mahmood Gilani

B.S. Computer Science  
Final Year Project  
April 2022

## Anti-Plagiarism Declaration

This is to declare that the above publication produced under the:


**Title:** Self-Embedding Watermarking System

is the sole contribution of the author(s) and no part hereof has been reproduced on **as it is** basis (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: 06-04-2022

Student 1

Name: Arbab Hamd Rizwan

Signature: 

Student 2

Name: Usama Aslam

Signature: 

Student 3

Name: Aashar Naseem

Signature: 

Student 4

Name: M. Hunzlah Malik

Signature: 



## Table of Contents

Table of Contents .....	i
List of Tables .....	iv
List of Figures .....	v
Abstract .....	1
Chapter 1: Introduction .....	2
1.1 Goals and Objectives .....	2
1.2 Scope of the Project .....	2
Chapter 2: Literature Survey / Related Work .....	3
2.1 Image Watermarking .....	3
2.1.1 Triple Recovery Information Embedding Approach .....	3
2.1.2 Pixel-wise Authentication Based Self-Embedding Fragile Watermarking Method 3 .....	3
2.1.3 Hash Function Based Fragile Watermarking .....	3
2.1.4 Fragile Watermarking based on Adaptive Selection of Embedding Mode .....	4
2.2 Video Watermarking .....	4
2.3 Spatiotemporal Trident Networks: Detection and Localization of Object Removal Tampering .....	4
2.4 Keyframe Extraction .....	4
2.4.1 Key-Frame-Extraction Based on Perceived Motion Energy Model .....	5
2.4.2 Cascaded Map Reduce Method .....	5
2.4.3 Keyframe Extraction Based on Dynamic Color Histogram .....	5
2.5 Conclusion .....	5
Chapter 3: Requirements and Design .....	6
3.1 Functional Requirements .....	6
3.2 Non-Functional Requirements .....	6
3.2.1 Performance Requirements .....	6
3.2.2 Security Requirements .....	6
3.2.3 Usability Requirements .....	6
3.2.4 Sustainability .....	6
3.2.5 Reliability .....	7
3.2.6 Extensibility .....	7
3.3 Hardware and Software Requirements .....	7
3.3.1 Hardware Requirements .....	7
3.3.2 Software Requirements .....	7
3.4 System Architecture .....	8
3.4.1 Image Authentication Module .....	8
3.4.2 Video Authentication Module .....	8
3.4.3 Image Recovery Module .....	9
3.4.4 Video Recovery Module .....	9
3.5 Architectural Strategies .....	9
3.5.1 External Dependencies .....	9
3.5.2 React, Node and Python .....	10
3.5.3 Concurrency .....	10
3.5.4 Future Goals .....	10
3.5.5 User Interface Paradigms .....	10
3.5.6 Database .....	10
3.6 Use Cases .....	11
3.6.1 Watermark Image .....	11
3.6.2 Watermark Video .....	12

3.6.3	Image Tamper Detection.....	13
3.6.4	Video Tamper Detection.....	14
3.6.5	Video Reconstruction.....	15
3.6.6	Image Reconstruction .....	16
3.7	GUI .....	17
3.7.1	Add Watermark.....	17
3.7.2	Upload Image.....	18
3.7.3	Upload Video .....	18
3.7.4	Video Authentication .....	19
3.7.5	Video Restoration .....	20
3.8	System Requirements.....	20
3.9	Design Considerations .....	20
3.9.1	Assumptions.....	20
3.9.2	Dependencies .....	20
3.10	Development Methods .....	21
3.10.1	Algorithms .....	21
3.10.2	Development Methodology Used .....	22
3.11	Class diagram.....	22
3.12	Sequence diagram .....	23
3.12.1	Image Authentication.....	23
3.12.2	Video Authentication .....	23
3.12.3	Image Restoration .....	24
3.12.4	Video Restoration .....	24
3.12.5	Image Watermarking .....	25
3.12.6	Video Watermarking.....	25
3.13	Policies and Tactics.....	26
3.13.1	Tools to be Used .....	26
3.13.2	Policy for User Interface .....	26
3.13.3	Coding Guidelines .....	26
3.13.4	Plans for using Latest Technologies .....	26
3.13.5	Policy for Software Testing .....	26
3.13.6	Accessing the System .....	26
3.13.7	Policy for System Maintenance .....	26
Chapter 4:	Implementation .....	27
4.1	Watermarking .....	27
4.1.1	Hash Value Generation for Authentication Bits .....	27
4.1.2	Multiple Variants of Lookup Table .....	27
4.1.3	Implementation of Image Division .....	27
4.2	Tamper Detection and Recovery .....	28
4.2.1	Triple Recovery Method .....	28
4.2.2	Pixel-wise Authentication and Recovery.....	28
4.3	Video Authentication and Restoration.....	28
4.3.1	Video Authentication .....	29
4.3.2	Video Restoration .....	30
Chapter 5:	Conclusion.....	31
References	.....	32
Appendix	.....	33
Appendix A:	Time Complexity .....	33



**List of Tables**

Table 1: Time Complexity for Watermark Embedding .....	33
Table 2: Time Complexity for Restoration .....	34

## List of Figures

Figure 1: High Level System Architecture .....	8
Figure 2: Interface Before Image is Watermarked .....	17
Figure 3: Interface After Image is Watermarked .....	17
Figure 4: Upload Image .....	18
Figure 5: Interface Before Video is Uploaded .....	18
Figure 6: Interface After Video is Uploaded .....	19
Figure 7: Video Authentication .....	19
Figure 8: Video Restoration.....	20
Figure 9: Class Diagram .....	22
Figure 10: Image Authentication Sequence Diagram.....	23
Figure 11: Video Authentication Sequence Diagram .....	23
Figure 12: Image Restoration Sequence Diagram .....	24
Figure 13: Video Restoration Sequence Diagram .....	24
Figure 14: Image Watermarking Sequence Diagram.....	25
Figure 15: Video Watermarking Sequence Diagram.....	25
Figure 16: Keyframe Selection .....	29
Figure 17: Video Restoration Process.....	30
Figure 18: (a) Lake, (b) Baboon, (c) Lena, (d) Goodhill, (e) Sailboat, (f) Airplane.....	34





## Abstract

These days, media editing software is very easily accessible on the internet, which leads to the issue of manipulating and tampering with either an image or a video. To address this issue, we created a system that is a web application that digitally watermarks media and then allows users to detect tampering and restore the media to its original state. In the case of an image, our system watermarks it by inserting both the authentication and restoration bits into the LSB of the image's sub blocks, which are used to detect image tampering and restoration. Similarly, in the case of video, our system first selects key frames from the stream of frames and watermarks it by inserting both the authentication and restoration bits into the LSB of the video's extracted frame/s sub blocks. To determine whether or not the media has been tampered with, compute the hash value of the watermarked media and compare it to the authentication bits, which is actually the hash value computed from the original image. If the hash values are the same, the image has not been tampered with; otherwise, it has been tampered with. Similarly, the original media content is restored using data extracted from restoration bits. The experimental results shows that the image can be restored up to 75%, implying that even if 75% of the image has been altered, it can be restored to its original state. However, in the case of video, the restoration percentage is 67 percent, implying that it can withstand attacks of up to 67 percent. The system's main functionalities include the ability to watermark media, detect tampering, and finally restore media. Furthermore, the system enables users to upload and download content from the web application. The entire processing is done on the server side in our system, so the user only needs to upload the content and download either watermarked or restored content.

## Chapter 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. 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 [1 - 3]. By first detecting the keyframes and then processing those frames as if they were images, this approach can also be utilized on video [4].

### 1.1 Goals and Objectives

The goals of this project are to detect the tempering of image and video and their restoration so that it can help in law enforcement and content ownership.

- Watermarking an image.
- Detecting image tampering using watermarking.
- By using self-embedded watermarking, reconstructing the image.
- Identifying the key frames in a video
- Watermarking the identified key frames in a video.
- Video tampering detection.

By using self-embedded watermarking of the key frames, restoring the video.

### 1.2 Scope of the Project

The project mainly focuses on image and video tampering detection and reconstruction. So, the area of work here is to use Digital Image Processing and Computer Vision for watermarking an image and video. A watermarked superimposed into an image directly however in a video the key frames are first detected using Artificial Intelligence and then these key frames are watermarked. In this project the data encryption algorithms used are SHA-256 hash function, pixel-wise authentication-based self-embedding fragile watermarking method and finally self-embedded fragile watermarking method. So, our goal is using these above methods to find an optimal solution to our problem.

## Chapter 2: Literature Survey / Related Work

Many researches have been conducted in the past relating to this field. Each research either has its own unique method or it improves a previous method. Here, we will discuss the various methods that are related to our project.

### 2.1 Image Watermarking

The main method for image watermarking, includes embedding data in bits into its least significant bits (to avoid reducing image quality). During our study, we came across the following image watermarking methods.

#### 2.1.1 Triple Recovery Information Embedding Approach

The embedded bits in fragile watermarking of the image contain both the authentication bits and the recovery bits, which are used to detect tamper detection and recover the image. This method [1] divides the original image into 16x16 non-overlapping main blocks, after which a lookup table is generated from which four random main blocks, known as partner blocks, are chosen. Then, each partner block is divided into 4x4 blocks, the average value of the partner block is calculated and converted to binary, and similarly, other blocks are converted to binary, and each of these binary bits from the partner blocks is merged into other partner blocks. The recovery bits are then permuted using a key. The partner blocks are then subdivided into 16x16 blocks, and these 16x16 blocks are subdivided even further into 8x8 blocks. The recovery bits are then embedded in the first and second LSBs of the first three 8x8 subblocks. The algorithm then checks to see if all partner blocks have been grouped before combining the divided blocks to create the original watermarked image.

#### 2.1.2 Pixel-wise Authentication Based Self-Embedding Fragile Watermarking Method

This method [2] involves watermarking the original image by dividing it into four equal parts known as main blocks. The algorithm then divides the main blocks into 4x2 or 2x4 subblocks, depending on the type of block. The recovery bits of each of the four main blocks are formed by computing the average values of the divided blocks and combining them. The recovery bits for each main block are then created by combining the recovery bits from the two main blocks in the other half of the image. Following the creation of the recovery bits, the four MSBs of the pixel, the recovery bit, and the two-pixel position bits are used to generate two authentication bits for each pixel. The authentication bits are then embedded in the pixel's first and second LSBs, while the recovery bit is embedded in the pixel's third LSB.

#### 2.1.3 Hash Function Based Fragile Watermarking

This method is an older version of the triple recovery method, the main difference between this method and triple recovery method is the accuracy and process of blocks division. The image is watermarked in this method [3] by first dividing it into 32x32 non-overlapping blocks, which are then divided into 16x16 non-overlapping sub blocks. The 256-bit hash value of the first three subblocks is then calculated using a hashing algorithm. The hash value 16x16 bit binary watermarked is then generated. The watermark computed by the first three subblocks is then modified in the fourth subblock. The same procedure is followed for all 32x32 blocks. The blocks are then combined to form the original, watermarked image.

### **2.1.4 Fragile Watermarking based on Adaptive Selection of Embedding Mode**

This method is from older research as compared to the ones mentioned before [1-3]. Just like other watermark embedding schemes, this method also divides the image into  $N/b^2$  (where  $N$  is assumed to be multiple of  $b$ ) non-overlapping image blocks with sizes  $b \times b$ , which is later allocated an authentication bit generated through a hash function. These bits are then stored in the LSBs and are used as references when an image needs to be checked for any kind of tampering [8]. However, this method was not considered for implementation of this project due to the lower tamper recover rates and due to the fact that other method can be considered as a latest work in this field.

## **2.2 Video Watermarking**

The concept of image watermarking was extended to videos as well. This method was proposed in "Video-Tampering Detection and Content Reconstruction via Self-Embedding," [4]. This paper makes use of the already present image watermarking methods and applies those to videos. This is carried out by first selecting a keyframe in a window of frames and then watermarking that keyframe by any of the image watermarking methods available. However, a video can have many keyframes and each keyframe will need to be processed and watermarked to completely watermark the whole video. This process can be time consuming and thus, it will require a lot of processing power and good programming logic for faster watermarking process.

## **2.3 Spatiotemporal Trident Networks: Detection and Localization of Object Removal Tampering**

Unlike other methods, which embed data into the image or video. This method makes use of artificial intelligence techniques to detect object removal tampering in videos. This is carried out by training a CNN model [6] by giving it various datasets of videos. Once the model has finished training, it can process a video and detect object removal tampering without placing any kind of data into the video itself. The inputted video is processed in a series of odd numbered frames, which are divided and inserted through three different branches of inputs. Once they have been processed, they are labelled as pristine or forged frames. If even one forged frame is found in a frame window, the whole video is marked as tampered. This method may seem useful and reliable, but it comes with its own limitations. The main limitation of this method was that it can only detect object removal tampering. The other limitation is that unlike other methods, it is not possible to restore a tampered video using this method. This is due to the reason that no data is placed in the video which can be used as a way to restore the original video.

## **2.4 Keyframe Extraction**

In animation and cinematography, a key frame (or keyframe) is a graphic or shot that marks the beginning and finish locations of any seamless transition. Because their position in time is measured in frames on a strip of film or on a digital video editing timeline, these are referred to as frames. The position of the key frames on the film, video, or animation determines which movement the viewer will see, whereas the sequence of key frames determines the timing of the movement. The remaining frames are filled with "inbetweens" because only two or three crucial frames over the course of a second do not generate the appearance of movement. There are many methods for keyframe extraction, but only a few methods will be discussed here which include histogram method, motion perceived energy model and cascaded map reduce method.

### **2.4.1 Key-Frame-Extraction Based on Perceived Motion Energy Model**

This method uses motion patterns of a shot to determine whether that frame is a keyframe or not. A motion pattern of a shot is usually composed of a motion acceleration process, followed by deceleration process. Such a motion pattern usually reflects an action in events [10]. This method works by building a motion model, which is used in the paper as a triangle model of perceived motion energy (PME). Motion triangle is generated for each frame and its PME value is compared. The turning point of motion acceleration and deceleration of each motion pattern is selected as a key frame.

### **2.4.2 Cascaded Map Reduce Method**

This method makes use of Apache Hadoop, MapReduce Framework and HDFS. These technologies are used in light of the fact that depending on the size and frames per second of a video, it can have a lot of frames even if the video size is 5MB. For a video, 20 – 30 frames per second is quite common. If a video that is even 10 seconds long, it would result in a lot of frames and for our system, a 10 second video is quite small. The cascaded algorithm contains three MapReduce algorithms for the whole algorithm. This method works on a similar method as histogram method that will be discussed next. That is, a color histogram is calculated for each frame and frame difference is calculated using Euclidean distance, mean and standard deviation of absolute difference. These are used to compute a threshold which will be used as base to identify keyframes by checking if a specific frame has a value above the threshold. [11]

### **2.4.3 Keyframe Extraction Based on Dynamic Color Histogram**

This is the method that we will be using for our keyframe extraction algorithm. One frame is used to generate two different types of histograms. One is a color histogram and other is a fast wavelength histogram, they both give a sequence of keyframes. The keyframes are selected through an optimized k-means algorithm for both attribute features (color and fast wavelength histogram). The two keyframe sequences are compared through mutual information and redundant keyframes are removed. This method gives a better performance of keyframe extraction due to the fact that its color information and texture detail features are extracted by descriptors, that are selected automatically by the optimal k-means algorithm [12].

## **2.5 Conclusion**

In conclusion, it is necessary to properly consider the optimal method for tamper detection and restoration of multimedia. Therefore, we will follow two different methods [1, 2] that give best results in terms of percentage of tamper detection and restoration while also keeping in account the quality of multimedia. Both these methods shall be implemented and their results will be compared. Moreover, based on the obtained results, the method with best results will be extended to support video tamper detection and restoration. For keyframe extraction algorithm, a number of algorithms were taken into consideration. While the final algorithm has not been decided yet but it will make use of color histograms to compare frames and determine them using a threshold value or optimal k-means algorithm.

## Chapter 3: Requirements and Design

The functional requirements can be divided between the user and the watermarking system.

### 3.1 Functional Requirements

1. The system shall allow the user to place a watermark in an image or video.
2. The system shall allow the user to download watermarked images or videos.
3. The system shall allow the user to detect any tampering in watermarked images or videos.
4. The system shall allow the user to view detection results on tampered images or videos.
5. The system shall allow the user to reconstruct tampered images or videos that have been watermarked by the system.
6. The system shall allow the user to download reconstructed images or videos.
7. The system shall generate reports on tampered regions of watermarked images or videos.

### 3.2 Non-Functional Requirements

The following are non-functional requirements identified for the system:

#### 3.2.1 Performance Requirements

The System shall meet the following performance requirements:

- The system shall limit the size of the uploaded images or videos to 50MB.
- The system webpage shall load in 2 seconds.
- The system shall take an approximate time of 180 seconds (3 minutes) to embed watermark in an image [1].
- The system shall take an approximate time of 180-199 seconds to recover tampered image depending on the percentage of tampering that ranges from 25% - 75% (see Appendix A) [1].

#### 3.2.2 Security Requirements

The system shall meet the following security requirements:

- The system shall run image or video processing algorithms on server side to maintain secrecy of watermarking and reconstruction algorithms.
- The system shall use a permutation key to place watermarks on images.

#### 3.2.3 Usability Requirements

The following usability requirements shall be met by the system:

- The system shall provide an easy to learn and navigate user interface.
- A consistent theme shall be used throughout the system.

#### 3.2.4 Sustainability

The system shall be accessible from any computer browser assuming the hardware and software specifications mentioned in hardware and software requirements are met.

### **3.2.5 Reliability**

The system will have 100% uptime guarantee.

### **3.2.6 Extensibility**

The system shall be generic i.e., it shall work with most image or video formats.

## **3.3 Hardware and Software Requirements**

The project shall have the following requirements:

### **3.3.1 Hardware Requirements**

The hardware requirements for the usage of this system are the following:

- Cloud server for the deployment of the system. The server will be responsible for handling the complex computations of images or videos.
- A GPU, preferably GTX 1050 Ti.
- CPU Intel Core i5-7500 (or higher) 3.40 GHz.
- 4GB RAM or higher.

### **3.3.2 Software Requirements**

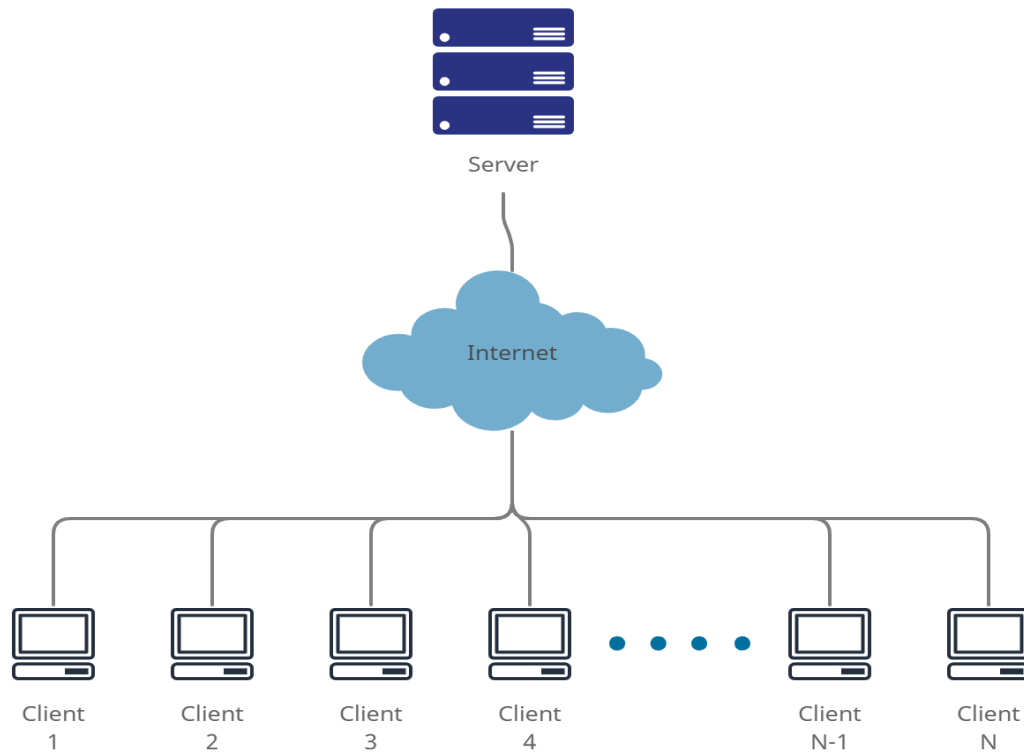
The software requirements for the usage of this system are as follows:

- NodeJS
- ReactJS
- Bootstrap 5
- Python
- Django
- JavaScript, specifically TypeScript
- Ant Design UI



### 3.4 System Architecture

This section describes system architecture of our system



**Figure 1: High Level System Architecture**

*Figure shows high level system architecture (client-server architecture)*

#### 3.4.1 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.

##### 3.4.1.1 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.

##### 3.4.1.2 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.

#### 3.4.2 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 is why we also use image authentication module in this part because it treated keyframes same as image and detect the tampering. Frames other than keyframes are embedded with frame index that is used to check if any frame was inserted or deleted.

### **3.4.2.1 Frame Index Embedding**

All frames will have its frame index embedded in them. This will help us make sure if any frame was deleted or inserted. In case, there is any alteration in frame index order, we can easily add the frame using keyframe near that frame index.

### **3.4.2.2 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. [4]

### **3.4.2.3 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.

### **3.4.3 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.

### **3.4.4 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.

## **3.5 Architectural Strategies**

Following architectural strategies will be followed to implement this project.

### **3.5.1 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.

### 3.5.2 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.

### 3.5.3 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.

### 3.5.4 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.

### 3.5.5 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.

### 3.5.6 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.

### 3.6 Use Cases

#### 3.6.1 Watermark Image

Name		Watermark Image	
Actors		User	
Summary		The user shall upload an image which will then be watermarked by the system.	
Pre-Conditions		None	
Post-Conditions		The page reloads and the user can download the watermarked image from the same page.	
Special Requirements		Max image size is 50MB, dimensions of image should be n×n	
Basic Flow			
Actor Action		System Response	
1	User uploads an image.	2	Place a watermark on the image.
Alternative Flow			
3	The user uploads an image which exceeds the limit.	4-A	The system responds with an error message: <i>Image size too large.</i>

### 3.6.2 Watermark Video

Name		Watermark video	
Actors		User	
Summary		The user shall upload a video which will then be watermarked by the system.	
Pre-Conditions		None	
Post-Conditions		The page reloads and the user can download the watermarked video from the same page.	
Special Requirements		Max image size is 50MB, dimensions of image should be n×n	
Basic Flow			
Actor Action		System Response	
1	User uploads a video.	2	The system places a watermark on the video.
Alternative Flow			
3	The user uploads a video which exceeds the limit.	4-A	The system responds with an error message: <i>Video size too large.</i>

### 3.6.3 Image Tamper Detection

<b>Name</b>		Image Tamper Detection	
<b>Actors</b>		User	
<b>Summary</b>		The user shall upload a watermarked image which will then be processed by the system to detect any kind of tamper to the image	
<b>Pre-Conditions</b>		The image must be watermarked by the system.	
<b>Post-Conditions</b>		Users shall be able to view reports of tampering.	
<b>Special Requirements</b>		Image size must not be larger than 50MB.	
<b>Basic Flow</b>			
<b>Actor Action</b>		<b>System Response</b>	
1	User uploads an image.	2	System processes the image and notifies the user if it is tampered.
<b>Alternative Flow</b>			
3	Users upload an image which exceeds the limit.	4-A	The system responds with an error message: <i>Image size too large</i>

### 3.6.4 Video Tamper Detection

<b>Name</b>		Video Tamper Detection	
<b>Actors</b>		User	
<b>Summary</b>		The user shall upload a watermarked video which will then be processed by the system to detect any kind of tamper to the video	
<b>Pre-Conditions</b>		The video must be watermarked by the system.	
<b>Post-Conditions</b>		Users shall be able to view reports of tampering.	
<b>Special Requirements</b>		Video size must not be larger than 50MB.	
<b>Basic Flow</b>			
<b>Actor Action</b>		<b>System Response</b>	
1	User uploads a video.	2	System processes the video and notifies the user if it is tampered.
<b>Alternative Flow</b>			
3	Users upload a video which exceeds the limit.	4-A	The system responds with an error message: <i>Video size too large</i>

### 3.6.5 Video Reconstruction

<b>Name</b>		Video Reconstruction	
<b>Actors</b>		User	
<b>Summary</b>		The user shall upload a watermarked video which will then be processed by the system to reconstruct.	
<b>Pre-Conditions</b>		The video must be watermarked by the system.	
<b>Post-Conditions</b>		Users shall be able to download recovered video.	
<b>Special Requirements</b>		Video size must not be larger than 50MB.	
<b>Basic Flow</b>			
<b>Actor Action</b>		<b>System Response</b>	
1	User uploads a video.	2	System processes the tampered video and reconstructs the original video for the user to download.
<b>Alternative Flow</b>			
3	Users upload a video which exceeds the limit.	4-A	The system responds with an error message: <i>Video size too large</i>



### 3.6.6 Image Reconstruction

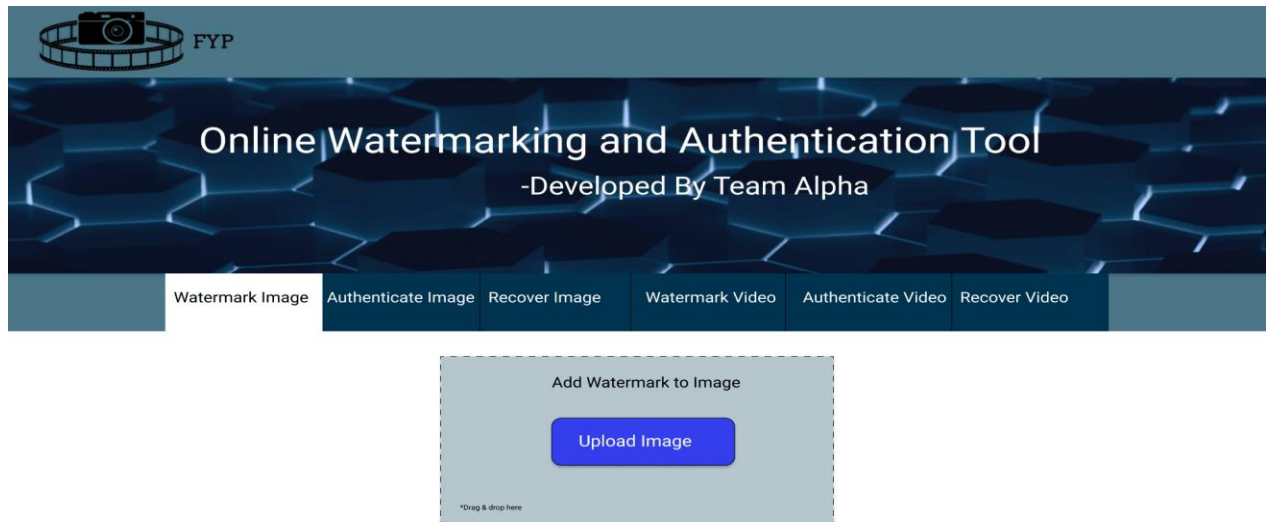
<b>Name</b>		Image Reconstruction	
<b>Actors</b>		User	
<b>Summary</b>		The user shall upload a watermarked image which will then be processed by the system to reconstruct.	
<b>Pre-Conditions</b>		The image must be watermarked by the system.	
<b>Post-Conditions</b>		Users shall be able to download recovered images.	
<b>Special Requirements</b>		Image size must not be larger than 50MB.	
<b>Basic Flow</b>			
<b>Actor Action</b>		<b>System Response</b>	
1	User uploads an image.	2	System processes the tampered image and reconstructs the original image for the user to download.
<b>Alternative Flow</b>			
3	User uploads an image which exceeds the limit.	4-A	The system responds with an error message: <i>Image size too large</i>

### 3.7 GUI

This section provides a prototype GUI of the project.

#### 3.7.1 Add Watermark

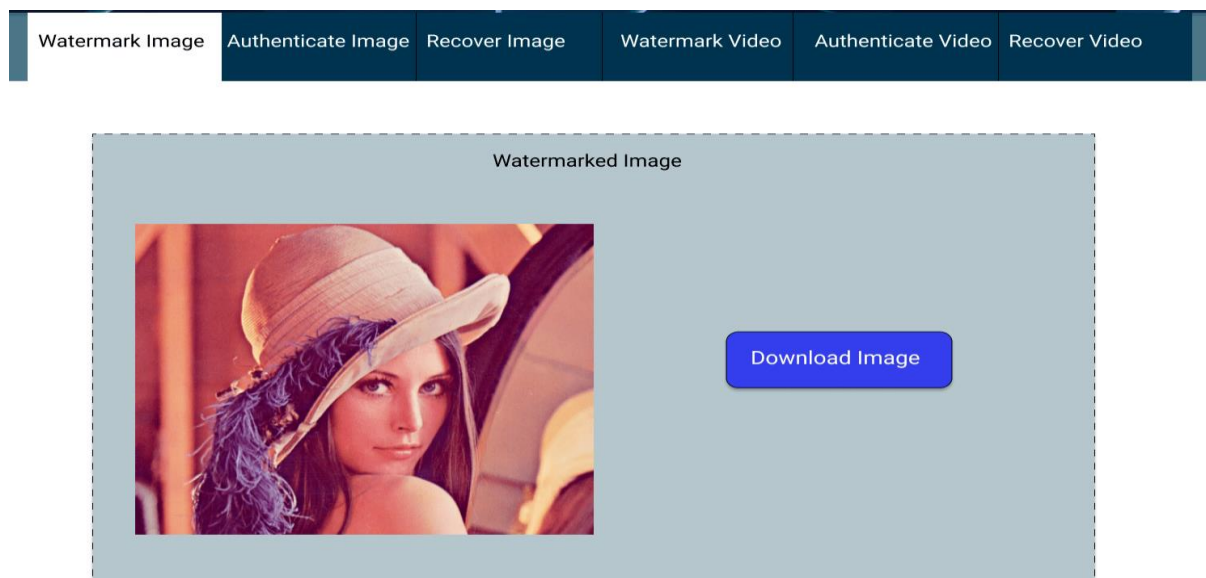
##### 3.7.1.1 Interface Before Image is Watermarked



**Figure 2: Interface Before Image is Watermarked**

*Figure shows the user interface for before watermarking an image*

##### 3.7.1.2 Interface After Image is Watermarked

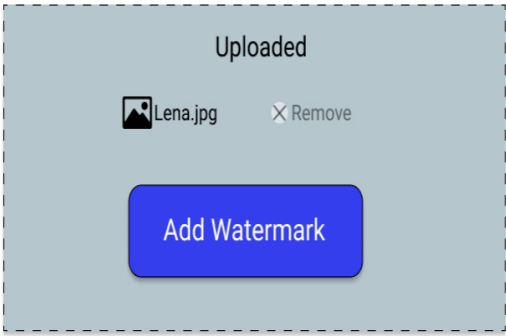


**Figure 3: Interface After Image is Watermarked**

*Figure shows user interface after an image has been watermarked*

3.7.2 Upload Image

Watermark Image	Authenticate Image	Recover Image	Watermark Video	Authenticate Video	Recover Video
-----------------	--------------------	---------------	-----------------	--------------------	---------------

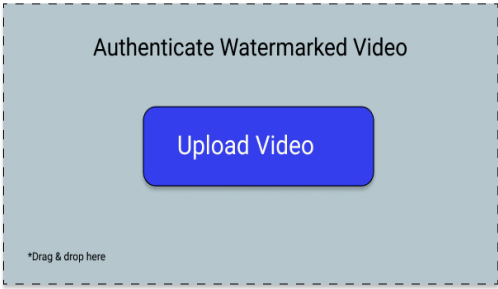


**Figure 4: Upload Image**  
*Figure shows the user interface for uploading an image*

3.7.3 Upload Video

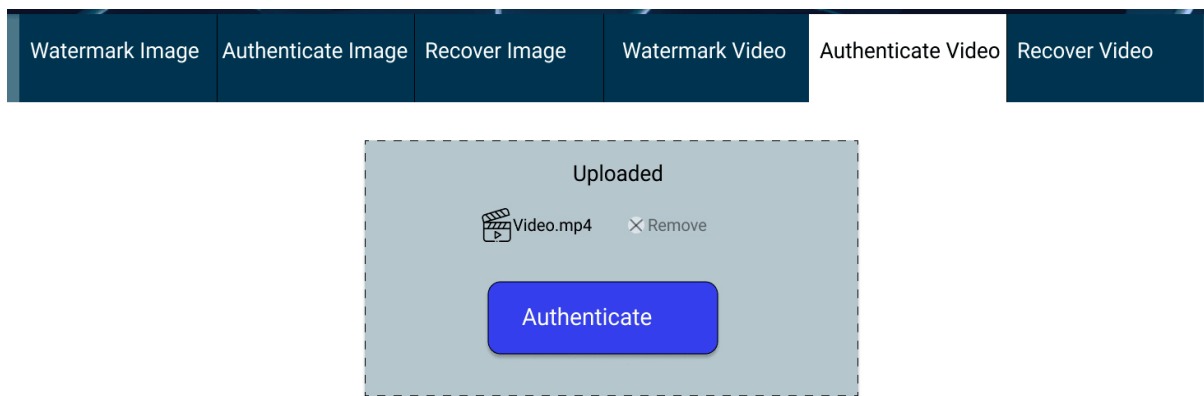
3.7.3.1 Interface Before Video is Uploaded

Watermark Image	Authenticate Image	Recover Image	Watermark Video	Authenticate Video	Recover Video
-----------------	--------------------	---------------	-----------------	--------------------	---------------



**Figure 5: Interface Before Video is Uploaded**  
*Figure shows user interface before uploading video*

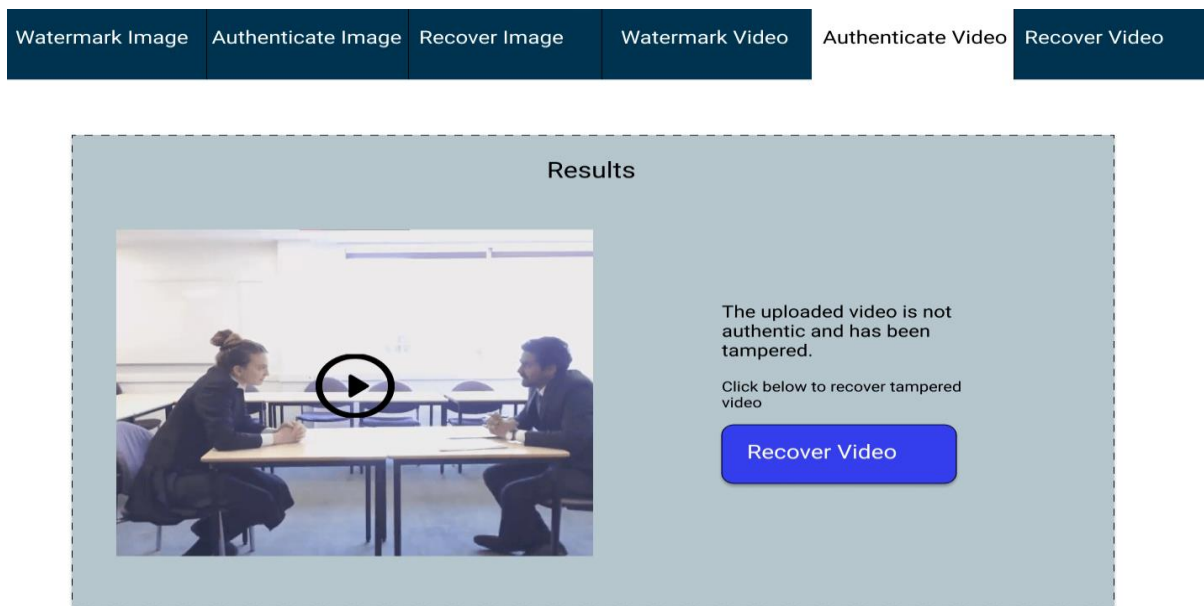
### 3.7.3.2 Interface After Video is Uploaded



**Figure 6: Interface After Video is Uploaded**

*Figure shows interface after video is uploaded*

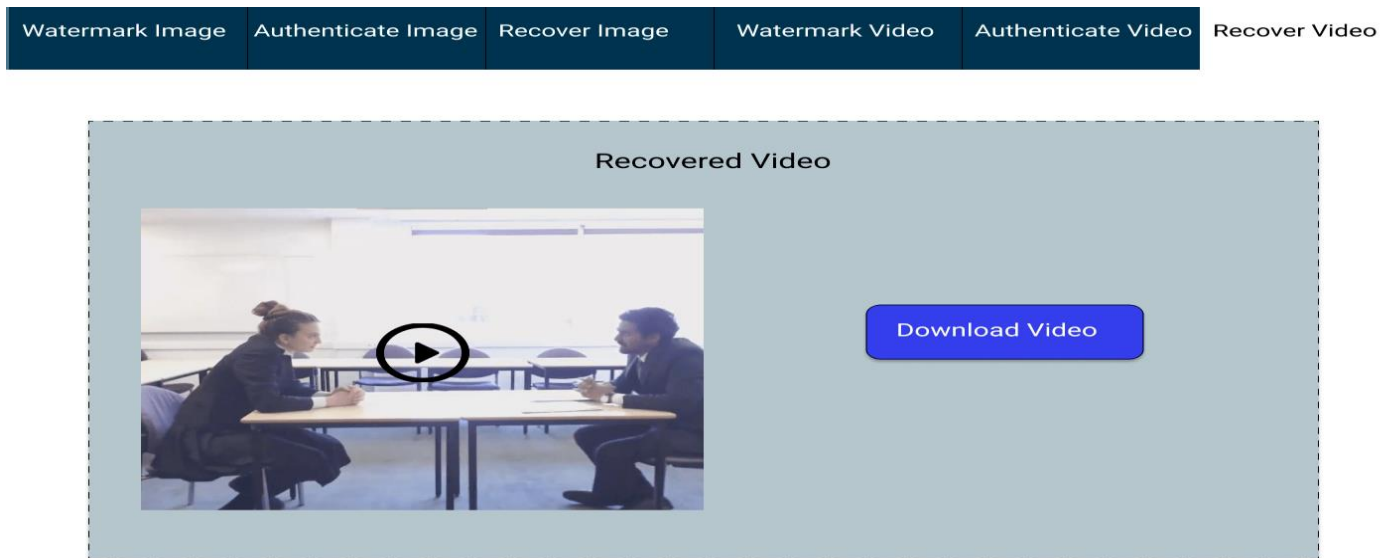
### 3.7.4 Video Authentication



**Figure 7: Video Authentication**

*Figure shows interface after video is uploaded*

### 3.7.5 Video Restoration



**Figure 8: Video Restoration**  
*Figure shows interface after video is restored*

## 3.8 System Requirements

The system will require the following features to process as required:

- Stable internet connection so that the user can successfully upload files to server and download the required files from the website.
- UpToDate web browser which supports the required UI libraries.
- NodeJS runtime environment.
- Python
- Ant Design UI.
- A good webserver to handle all the server-side processing load.

## 3.9 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.

### 3.9.1 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.

### 3.9.2 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  $512 \times 512$ .
- Multimedia processing may require a lot of resources, therefore, a good server with high end GPU would be needed to run scripts on backend.

### 3.10 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.

#### 3.10.1 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 therefore, we will compare the results of the two methods and then decide which one shall be extended to video authentication and restoration .

##### 3.10.1.1 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 permutes 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.

##### 3.10.1.2 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.

### 3.10.1.3 Pixel-wise Authentication and Recovery

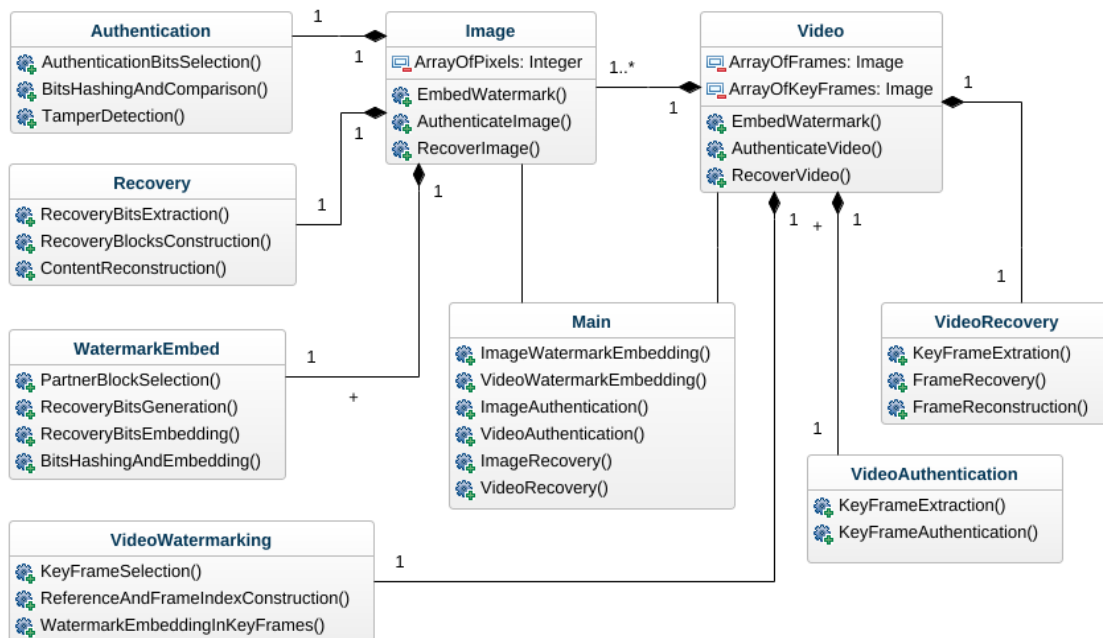
This method is the latest research in this field, which made use of pixel wise processing unlike other methods that relied on block-wise image authentication and restoration. This method is more efficient in terms of tamper detection and processes on a smaller area (pixels) as compared to block wise 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 [1, 2]. Which is why we will thoroughly compare the results of both methods which would include their PSNR, authentication percentage and recovery percentage.

### 3.10.2 Development Methodology Used

Upon looking into various software development models [7], 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.

### 3.11 Class diagram

This section shows the class diagram of the system.



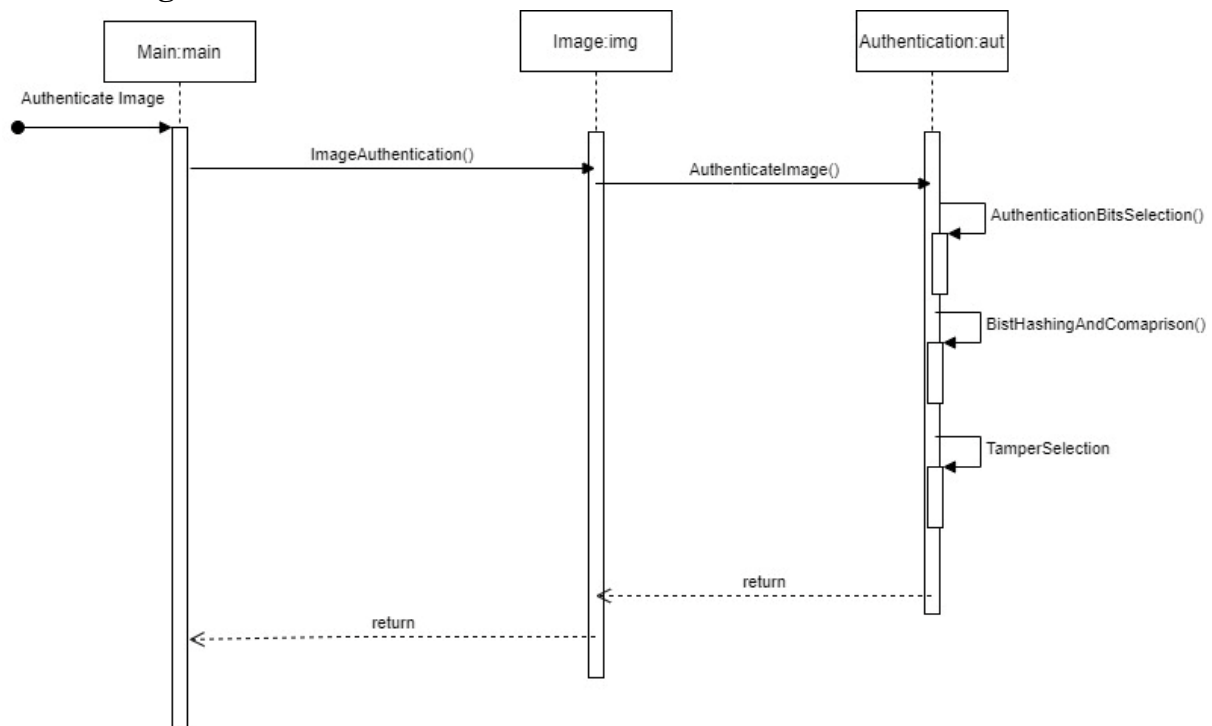
**Figure 9: Class Diagram**

Figure shows class diagram of the system

### 3.12 Sequence diagram

Following are the sequence diagrams for the system.

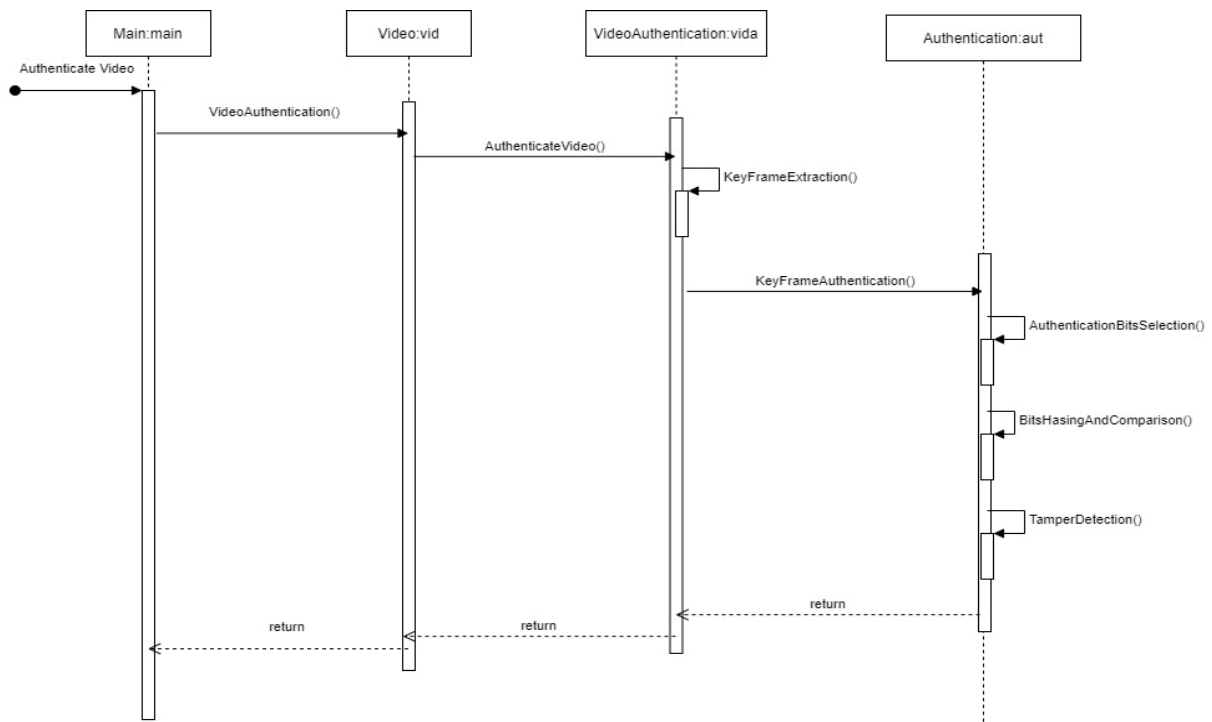
#### 3.12.1 Image Authentication



**Figure 10: Image Authentication Sequence Diagram**

Figure shows sequence diagram for image authentication

#### 3.12.2 Video Authentication

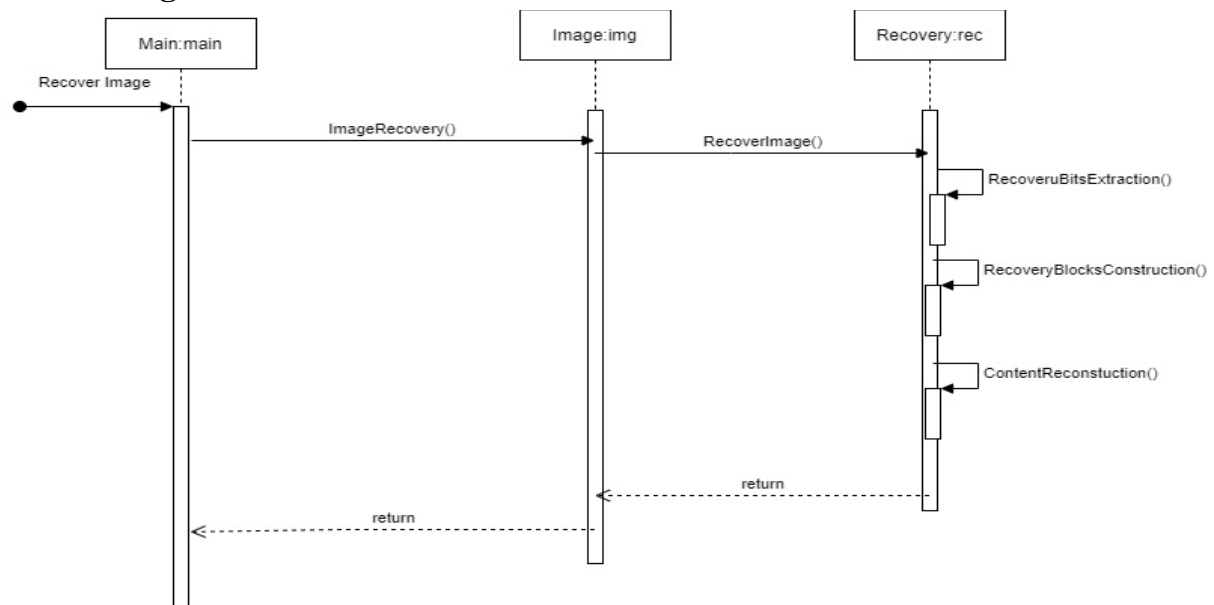


**Figure 11: Video Authentication Sequence Diagram**

Figure shows sequence diagram for video authentication

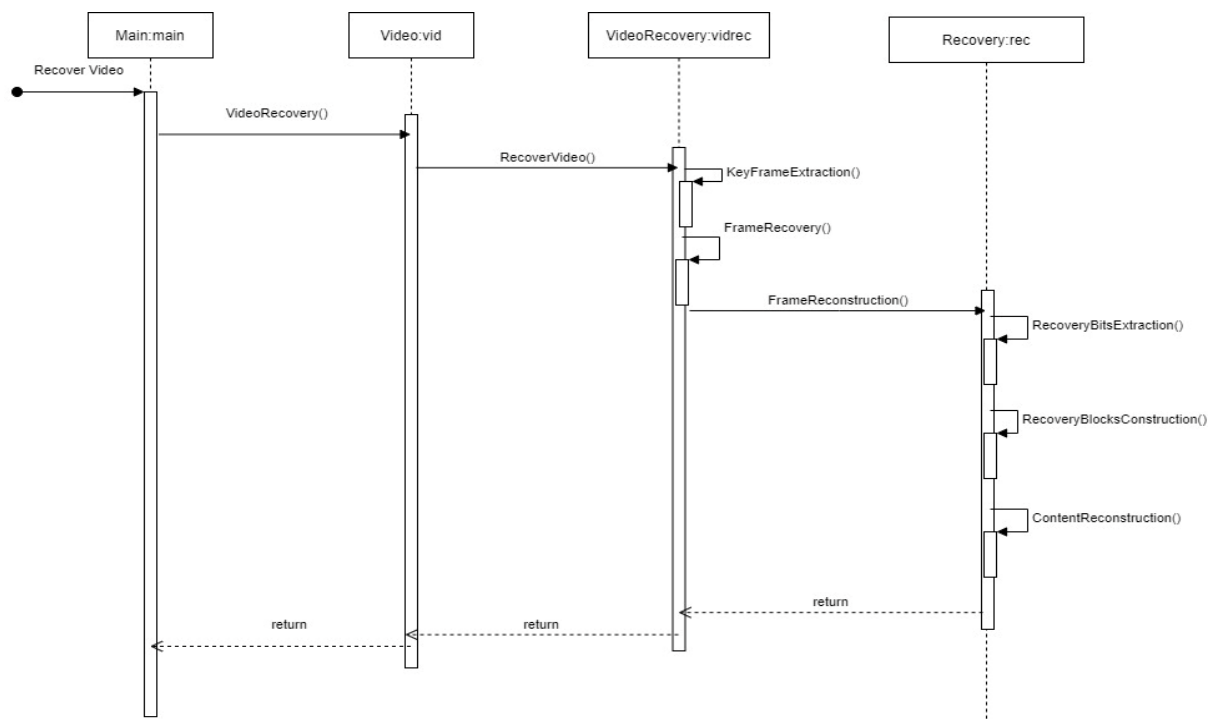


### 3.12.3 Image Restoration



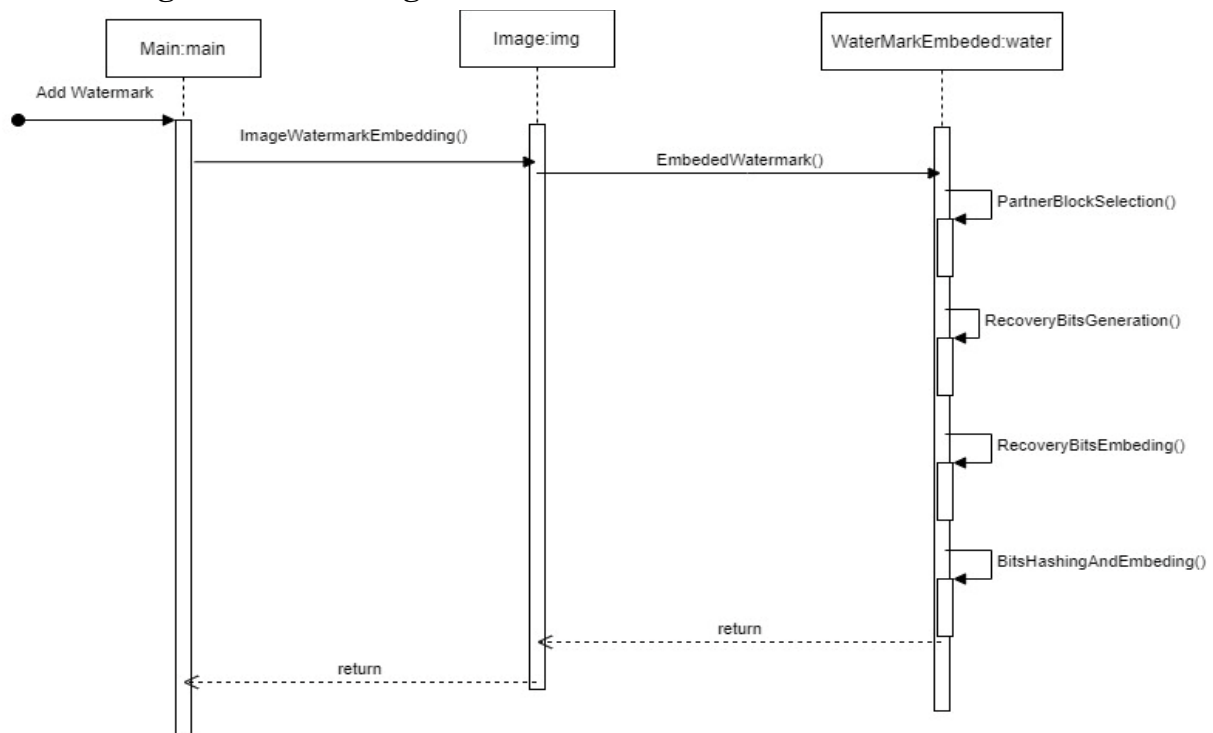
**Figure 12: Image Restoration Sequence Diagram**  
 Figure shows sequence diagram for image restoration

### 3.12.4 Video Restoration



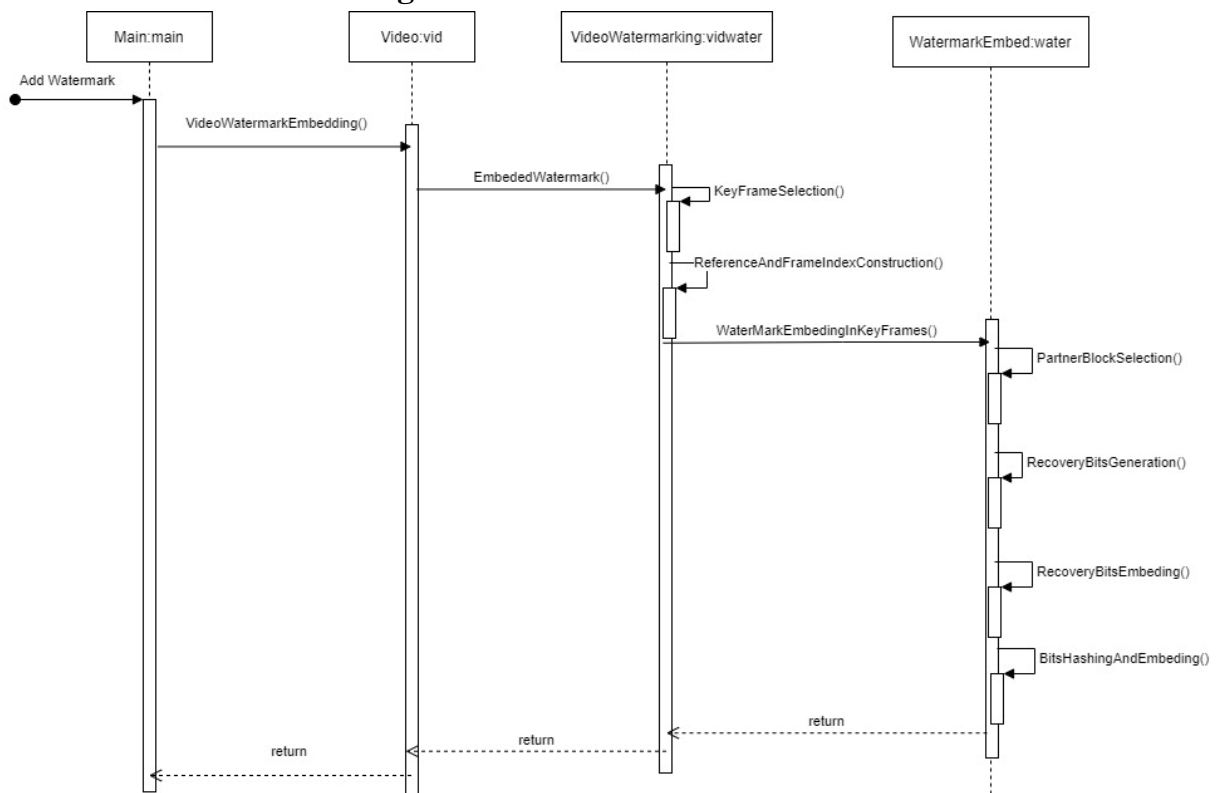
**Figure 13: Video Restoration Sequence Diagram**  
 Figure shows sequence diagram for video restoration

### 3.12.5 Image Watermarking



**Figure 14: Image Watermarking Sequence Diagram**  
 Figure shows sequence diagram for image watermarking

### 3.12.6 Video Watermarking



**Figure 15: Video Watermarking Sequence Diagram**  
 Figure shows sequence diagram for video watermarking

### **3.13 Policies and Tactics**

#### **3.13.1 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 Collab because it has GPU sources integrated.

#### **3.13.2 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.

#### **3.13.3 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.

#### **3.13.4 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.

#### **3.13.5 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.

#### **3.13.6 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.

#### **3.13.7 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.

## Chapter 4: Implementation

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.

### 4.1 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]

#### 4.1.1 Hash Value Generation for Authentication Bits

We will make a few changes to the normal authentication bits embedding process for triple recovery method. Firstly, we will use a python library called Blake2b which generates dynamic length hash which are unique at different lengths. For example, in our case, we do not need a full 128-bit hash, instead we need 104 bits hash. The remaining 24 bits will be used to store image size which is a multiple of 64 along with different variants of lookup table and block number. One way to get 104-bit hash was to first generate 128-bit hash value and truncate the extra bits, which would be an extremely bad way to get dynamic hash value. This is where the usage of Blake2b comes in which gives us dynamic 104-bit hash value without any truncation. This may raise concerns regarding hash collision which is why we studied a research paper for this method that shows us chances for hash collision and also tells us about how Blake2b generates secure and dynamic length hash values. [13]

#### 4.1.2 Multiple Variants of Lookup Table

Initially, a concern was raised regarding the security of lookup table according to which, a user may be able to exploit block placing in lookup table and compromise the efficiency for image restoration. This would result in our system failing to recover an image which was tampered less than the limit that is 75%. To solve this issue, we generated multiple lookup table which would not affect image restoration efficiency and would also introduce more secure watermarking. When a user will watermark an image through our system, they will need to provide a secure password which will be used to generate a value to select a lookup table variant. The user will need this password if they wish to authenticate or recover a tampered image. This will also prove as an ownership of that specific media because only the person with the right password will be able to recovery the original image from the tampered image. This also one of the reasons, we have reduced 128-bit hash size to 104-bit because we will be using few of the bits to store lookup table variant number which will be extracted and used during authentication and recovery process.

#### 4.1.3 Implementation of Image Division

Our initial work began with dividing the image into small blocks according to our implementation method. For image division into 4 x 2 or 2 x 4 blocks, NumPy library was used. However, it was not able to properly reshape the image blocks into the required form. This means that it successfully reshaped according to the required size of block but it was not able to properly add the correct data into those blocks. Upon further research on working of NumPy reshape, we came across an article that explained how image data is converted into another dimension before being divided into smaller parts. [9]

## 4.2 Tamper Detection and Recovery

The algorithm used to authenticate the image and reconstruct it is from [1, 2].

### 4.2.1 Triple Recovery Method

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 permutes 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.

### 4.2.2 Pixel-wise Authentication and Recovery

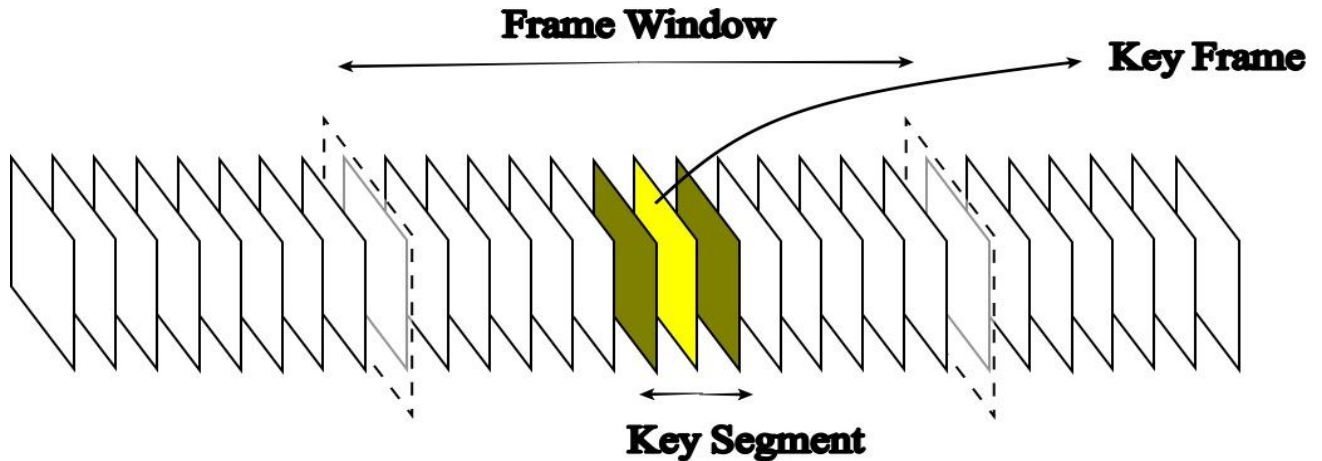
This method is the latest research in this field, which made use of pixel wise processing unlike other methods that relied on block-wise image authentication and restoration. This method is more efficient in terms of tamper detection and processes on a smaller area (pixels) as compared to block wise image authentication which processes the whole image block even if a single pixel was tampered. In this method, we first determine which orientation the image should be divided in (2 x 4 or 4 x 2 orientation). This is carried out by dividing the image into either of the two block types and comparing their PSNR values. The one with better PSNR value is selected for authentication and recovery bits embedding process. This process is similar to the one mentioned before but the plus point is that this method operates on pixel level instead of this block level. This can ensure image quality and due to this reason, we are considering this method even though it has lesser restoration rate (50% max).

## 4.3 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. This module will use either of the two above mentioned methods as a single frame processing algorithm. However, this will be finalized once we have successfully compared the results of the two methods. This comparison will help us decided between the compromise of quantity or quality i.e., one method gives higher restoration rate while the other method gives higher restoration quality.

### 4.3.1 Video Authentication

The video authentication module detects object removal and frame insertion tampering attacks. Key segment is first selected from a frame of windows. This key segment is then used to extract a key frame. These frames are used for reference construction and the frames which are not included in key segment are used to create frame index which also includes window number, frame number in the window and its distance from keyframe. [4]



**Figure 16: Keyframe Selection**

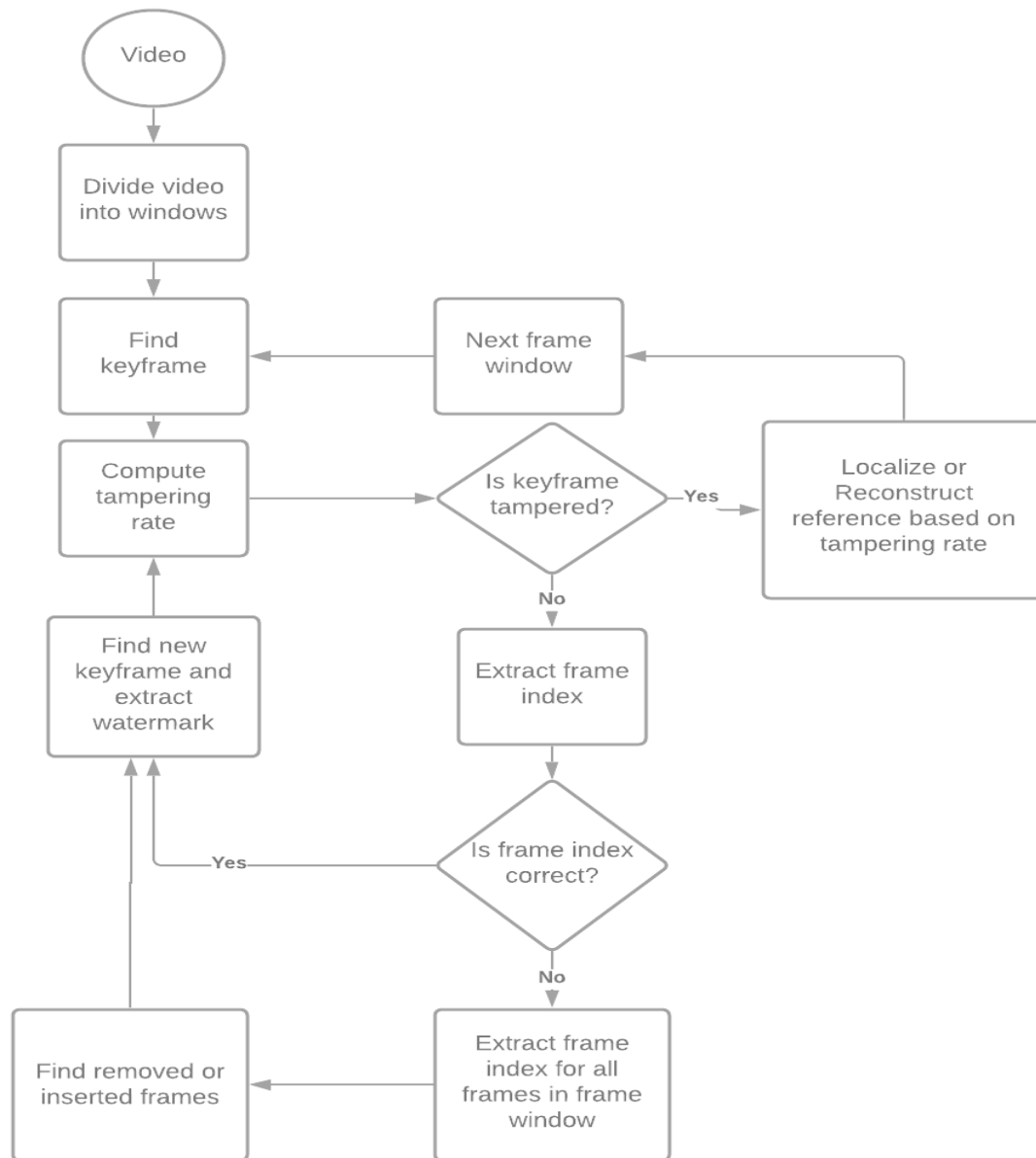
*Figure shows the keyframe selection process from window of frames*

The keyframe is then converted into 8x8 blocks, each block is divided into one of the four block types (Block type 1 to Block type 4). The bit allocation is determined according to the block type. Moreover, two additional bits are used to store block type. The watermark embedding process is similar to that of image watermarking however, it also embeds two more bits of data that is the frame index and second is reference.

## Implementation

### 4.3.2 Video Restoration

The video is decoded and undergoes through a repetitive process which checks for both interframe and intraframe tampering and also recovers the tampered frames in video. This process begins by partitioning video into windows and then extracting keyframe and then extracting watermark. Once extracted, the tampering rate is computed and it is then determined whether the frame is tampered or not. In case it is tampered, it is reconstructed using reference, else frame index is extracted and is compared whether its is correct or not. Incorrect frame index means that more frames were removed or inserted. Which would lead the system to extract all frame indexes of the windows and the removed or inserted frames would be corrected. This process is repeated till all frame windows have been processed and at the end of all this processing, the whole video is checked for tampering and is reconstructed in case of any tampering.



**Figure 17: Video Restoration Process**

*Figure shows video restoration process by decoding video*

## Chapter 5: Conclusion

There are various image watermarking methods, but we have studied the latest and the most useful ones. Among the ones that we studied, two methods which were latest and had most quality and quantity were selected. Through these methods, we will be able to successfully detect tampering attacks and restore the tampered multimedia to its original state. Up till now, we have completed the authentication bits embedding process. This also includes image blocks division and recombining the whole image. We have methods to calculate the PSNR of processed images which will later be used for results comparison and decision of which method to extend for video processing as well. For the testing process, we first use gray scale images to test the working of our scripts. Later on, they are tested with images of higher resolution to test processing time of our system for an image that will be divide into large number of blocks. Moreover, we have also completed the prototype GUI for our project.

Therefore, our project makes extensive use of web development technologies, image processing, as well as data science, due to the fact that we have to make use of python data processing libraries to process images. Our main goal is to reduce processing time as much as possible and due to this reason; we are using python libraries such as NumPy which can efficiently process large amounts of data which in our case is the large amount blocks. Currently, our scripts take around 40 seconds to process 2K resolution images.

Our next goal in till mid evaluation is to successfully show the results of both methods and extend the one with best results to support video processing. We have already started working on keyframe extraction algorithm, which will be demonstrated in the final evaluation along with the whole project with would include both image and video modules. We will also add test cases for image module and keyframe extraction module in the next report. Moreover, we will also link the finalized multimedia processing scripts to our GUI through REST APIs and add our scripts to server side.



## 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] 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.
- [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] 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].
- [8] Qin, C., Wang, H., Zhang, X. and Sun, X., 2016. Self-embedding fragile watermarking based on reference-data interleaving and adaptive selection of embedding mode. *Information Sciences*, 373, pp.233-250.
- [9] Doundoulakis, I., 2021. Efficiently splitting an image into tiles in Python using NumPy. [online] Medium. Available at: <https://towardsdatascience.com/efficiently-splitting-an-image-into-tiles-in-python-using-numpy-d1bf0dd7b6f7> [Accessed 28 December 2021].
- [10] Tianming Liu, Hong-Jiang Zhang and Feihu Qi, "A novel video key-frame-extraction algorithm based on perceived motion energy model," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 10, pp. 1006–1013, Oct. 2003, doi: 10.1109/TCSVT.2003.816521.
- [11] P. Deshmane, "MRKFE: Designing cascaded map reduce algorithm for key frame extraction," 2017 International Conference on Information Communication and Embedded Systems (ICICES), 2017, pp. 1–6, doi: 10.1109/ICICES.2017.8070719.
- [12] Z. Zong and Q. Gong, "Key frame extraction based on dynamic color histogram and fast wavelet histogram," 2017 IEEE International Conference on Information and Automation (ICIA), 2017, pp. 183–188, doi: 10.1109/ICInfA.2017.8078903.
- [13] Guo, J., Karpman, P., Nikolić, I., Wang, L., & Wu, S. (2014). Analysis of blake2. Retrieved January 24, 2022, from <https://eprint.iacr.org/2013/467.pdf>

## Appendix

### Appendix A: Time Complexity

**Table 1: Time Complexity for Watermark Embedding**

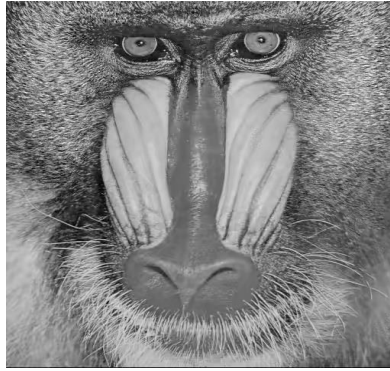
*Following table shows time complexity for watermark embedding process*

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  $n \times n$  dimensions. For the images mentioned in the table, the dimensions are  $512 \times 512$ . 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.



(a)



(b)



(c)



(d)



(e)



(f)

**Figure 18: (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 2: 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