Turnitin Originality Report

Self-Embedding Watermarking System by Arbab Hamd Rizwan

From General 123 (General 123)

- Processed on 2022년 06월 02일 10:04 PKT
- ID: 1848909085
- Word Count: 11453

Similarity Index
9%
Similarity by Source

Internet Sources:
3%
Publications:
3%
Student Papers:
6%

sources:

1 3% match (student papers from 03-Jun-2020)
Submitted to Higher Education Commission Pakistan on 2020-06-03

2 1% match (student papers from 10-Nov-2012)
Submitted to Higher Education Commission Pakistan on 2012-11-10

3 1% match (Internet from 27-Jan-2022)
https://link.springer.com/article/10.1007/s00530-021-00751-3?code=114bcb60-e767-4c14-869a-032590125824&error=cookies_not_supported

4 < 1% match (student papers from 03-Jun-2020)
Submitted to Higher Education Commission Pakistan on 2020-06-03

5 < 1% match (student papers from 13-Nov-2014)
Submitted to Higher Education Commission Pakistan on 2014-11-13

6 < 1% match (student papers from 09-Aug-2012)
Submitted to Higher Education Commission Pakistan on 2012-08-09

7 < 1% match (Internet from 05-Dec-2021)
https://link.springer.com/article/10.1007/s11042-020-09548-4?code=4457cf48-e1d1-403c-ba68-df6d2e085bbc&error=cookies_not_supported

8 < 1% match (Internet from 08-Apr-2020)
http://cgit.nutn.edu.tw:8080/cgit/PaperDL/CMS_080926061412.pdf

9 < 1% match (student papers from 23-Feb-2022)
Submitted to Bury College on 2022-02-23

10 < 1% match (Internet from 25-Jan-2022)
https://123dok.com/article/use-case-diagram-case-narrative-perancangan-sistem-informasi.ye18vxez

11 < 1% match (publications)
Amani Alharbi, Sara Aloufi, Rahaf Assar, Aisha Alturkistani, Reham Abdullah, Bahjat Fakieh. "Arabic-Chinese Language Mobile App for Children", 2021 International Conference on Advanced Enterprise Information System (AEIS), 2021

12 < 1% match (Internet from 29-Nov-2017)
https://documents.mx/documents/sds-template.html

13 < 1% match (publications)
Hsien-Chu Wu, Wen-Li Fan, Chwei-Shyong Tsai, Josh Jia-Ching Ying. "An image authentication and recovery system based on discrete wavelet transform and convolutional neural networks", Multimedia Tools and Applications, 2021

14 < 1% match (publications)
Ertugrul Gul, Serkan Ozturk. "A novel triple recovery information embedding approach for self-embedded digital image watermarking", Multimedia Tools and Applications, 2020

15 < 1% match (publications)
Ferda Ernawan, Afrig Aminuddin, Danakorn Nincarean, Mohd Faizal Ab Razak, Ahmad Firdaus. "Three Layer Authentications with a Spiral Block Mapping to Prove Authenticity in Medical Images", International Journal of Advanced Computer Science and Applications, 2022

16 < 1% match (student papers from 05-May-2006)
Submitted to University of Technology, Sydney on 2006-05-05

17 < 1% match (student papers from 05-May-2006)
Submitted to University of Technology, Sydney on 2006-05-05

18 < 1% match (publications)
Ertugrul Gul, Serkan Ozturk. "A novel pixel-wise authentication-based self-embedding fragile watermarking method", Multimedia Systems, 2021

19 < 1% match (Internet from 06-Jan-2022)
https://www.h2kinfosys.com/blog/software-testing-classification/

20 < 1% match (Internet from 20-Dec-2020)
https://en.wikipedia.org/wiki/Key_frame

21 < 1% match (student papers from 26-Mar-2007)
Submitted to University of Lancaster on 2007-03-26

22 < 1% match (publications)
Manasi Jana, Biswapati Jana, Subhankar Joardar. "Local feature based self-embedding fragile watermarking scheme for tampered detection and recovery utilizing AMBTC with fuzzy logic", Journal of King Saud University - Computer and Information Sciences, 2022

23 < 1% match (student papers from 30-Apr-2013)
Submitted to Anglia Ruskin University on 2013-04-30

24 < 1% match (student papers from 26-Nov-2015)
Submitted to Park Lane College on 2015-11-26

25 < 1% match (Internet from 03-Apr-2021)
https://ieeexplore.ieee.org/document/9301329/

26 < 1% match (publications)
Chia-Chen Lin, Si-Liang He, Chin-Chen Chang. "Pixel-based fragile image watermarking based on absolute moment block truncation coding", Multimedia Tools and Applications, 2021

27 < 1% match (publications)
Gokhan Azizoglu, Ahmet Nusret Toprak. "A novel reversible fragile watermarking in DWT domain for tamper localization and digital image authentication", 2021 9th International Symposium on Digital Forensics and Security (ISDFS), 2021

28 < 1% match (Internet from 21-Nov-2020)
https://jivp-eurasipjournals.springeropen.com/articles/10.1186/s13640-019-0462-3

29 < 1% match ()
Kannappan, Sivapriyaa. "Key-frame Analysis and Extraction for Automatic Summarization of Real-time Videos", 2019

30 < 1% match (publications)
"Proceedings of the 11th International Conference on Robotics, Vision, Signal Processing and Power Applications", Springer Science and Business Media LLC, 2022

31 < 1% match (publications)
Afrig Aminuddin, Ferda Ernawan. "AuSR1: Authentication and self-recovery using a new image inpainting technique with LSB shifting in fragile image watermarking", Journal of King Saud University - Computer and Information Sciences, 2022

paper text:

Self-Embedding Watermarking System 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

27**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]. 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. ? Watermarking the frames in a video in less time. ? Video tampering detection. By using self-embedded watermarking of the key frames, restoring the video. 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 all frames are watermarked. In this project the data encryption algorithms used are SHA- 256 hash function,

3**pixel-wise authentication-based self-embedding fragile watermarking** Introduction **method and**

finally self-embedded fragile watermarking method. So, our goal is using these above methods to find an optimal solution to our problem. 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. Image Watermarking There 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

15**Triple Recovery Information Embedding Approach** The **embedded**

bits in fragile watermarking

22**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]

> 26**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,

> 7**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.

> 14**The partner blocks are then** subdivided **into** 16x16 **blocks**

, and these 16x16 blocks are subdivided even further into 8x8 blocks. The recovery bits are

> 7**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

> 3**Pixel-wise Authentication Based Self-Embedding Fragile Watermarking Method This method**

[2] involves watermarking the original image by dividing it

> 3**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.

> 18**The recovery bits of each** of the four **main** blocks **are**

formed by computing the

> 18**average values of the** divided **blocks and**

combining them. The

> 18**recovery bits for each main block are**

then created by combining the

> 3**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

> 31**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. Literature Survey / Related Work 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 $N/b2$ (where N is assumed to be multiple of b) non-overlapping image blocks with sizes b x 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. 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. However, he above mentioned, algorithm for video watermarking is not the best solution for video recovery. The main reason is due to the watermarking process in which we only watermark keyframes and then use those frames to replace the tampered frames. In case some frames are missing, the same watermarked keyframe extracted from a frame window is placed into the position of the frames that were removed (temporal tampering). This will work provide excellent results when a few frames are

removed. Assuming the attacker tampers and removes two frames, in this case, the above-mentioned method will work perfectly. On the contrary, when a large portion of video is trimmed (assuming 5 seconds of video), we would be able to deal with temporal tampering by adding the keyframes in place of removed frames. However, this will create a sort of glitch in the video because same keyframe would be repeated to cover all the missing frames. The above-mentioned method is unable to deal with such scenario, which is why we will not deal with temporal tampering and will shift our main focus to detecting and recovering spatial tampering. One of the main drawbacks of this approach is the computation cost because we would be applying image watermarking on all of the frames of video which can take a lot of time. To deal with this we will implement multithreading on server side by using Lithops, a multi-cloud serverless programming framework, this will be discussed in the later on section of our literature review.

> 25**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. Keyframe Extraction

> 9**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**

> 24**viewer will see, whereas the** sequence **of key frames** determines **the** timing of **the** movement. **The**

remaining frames are filled with "in-betweens"

> 20**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

> 8**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.

> 8**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

> 8**a triangle model of perceived motion energy (PME). Motion**

triangle is generated for each frame and its PME value is compared. The

> 8**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,

> 29**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]. Literature Survey / Related Work Lithops, a Multi-cloud Serverless Programming Framework Lithops is a robust multi-cloud framework that allows local, multi-process Python programs to scale seamlessly into huge amounts of cloud

resources [14]. It is an open-source project that is also available on GitHub. Lithops makes it simple to spawn hundreds or thousands of jobs in order to complete a huge work in a matter of seconds. Lithops may be thought of as a dynamic job orchestrator that is geared for running jobs in a serverless computing environment. This will work well with our intention of using multithreading on server side so that the user does not have to do all the computing on their side. Through this framework, we can make use of various serverless platforms such as AWS Lambda, Google Cloud Run or IBM Cloud. Lithops works by transferring the locally created classes and transfers them to cloud. In background, it normally makes use of an object storage service (e.g., AWS S3) to store this information and other intermediate data. 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. Our method for video watermarking will mainly be to watermark all frames of the video but this is computationally very expensive, which is why we will be using different approaches to reduce this time as much as possible. Requirements and Design The functional requirements can be divided between the user and the watermarking system. Functional Requirements

17**1. The system shall allow the user to** place **a**

watermark in an image or video.

30**2. The system shall allow the user to**

download watermarked images or videos.

11**3. The system shall allow the user to**

detect any tampering in watermarked images or videos.

11**4. The system shall allow the user to**

view detection results on tampered images or videos. 5.

11**The system shall allow the user to**

reconstruct tampered images or videos that have been watermarked by the system.

11**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. Non-

16**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. Requirements and Design 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. Hardware and Software Requirements The project shall have the following requirements: 3.

5**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

6**Software Requirements** The **software requirements for the** usage of this **system are as follows**

: ● NodeJS ● ReactJS ● Bootstrap 5 ● Python ● Django ● JavaScript, specifically TypeScript ● Material UI ● Styled components System Architecture This section describes system architecture of our system Figure 1:

23**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 all frames are extracted into an array of frames. Each frame is then embedded using the “Triple recovery information embedding” method. For fast authentication, we will divide the number of frames in a frame window by a random number and check that frame for any kind of tamper attack. This frame window will vary according to the type of video. For example, in a 30-fps video, a frame window will contain 30 frames and according to our algorithm, we will divide 30 by a random and then check the number of frames at specific point in a time of one second. Using this method, we will can have varying number of frames for 1 second of video and only these frames will be checked for any kind of tampering. This reduces the probability of missing any tampered frames because videos are usually tampered for more than 1 second and in our case, we are checking many frames in one second. This is better than checking every single frame for tampering but even still this will be very costly in terms of computations and to deal with the computation time required to perform this action, we will be using Lithops to create threads using multi-cloud serverless programming. 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 For video recovery, we are only dealing with spatial tampering which is tampering within the frame. Since we have already embedded each frame using “Triple recovery information embedding” method, we can easily recovery the tampered frame by using the image recovery module and applying it on the tampered frames. However, this process is not so simple because first we will have to identify the tampered frames in a frame window. These tampered frames will be delt separately and to recover original frames and these original frames will be replaced with the tampered frames which will give us the original video. 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. 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

1**Basic Flow Actor Action System Response 1 User** uploads **an image**

. 2 Place a watermark on the image.

1**Alternative Flow 3 The user** uploads **an image** which 4- **The system responds with an error**

exceeds the limit. A message: Image size too large. 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

10**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 4- The system responds with an error the limit. A message: Video size too large. 3.6.3 Image Tamper Detection

Name Image Tamper Detection Actors User Summary The user shall upload a watermarked image which will then be processed by the system to detect any kind of tamper to the image Pre-Conditions The image must be watermarked by the system. Post-Conditions Users shall be able to view reports of tampering. Special Requirements Image size must not be larger than 50MB.

10**Basic Flow Actor Action System Response 1 User** uploads an image. **2**

System processes the image and notifies the user if it is tampered. Alternative Flow 3 Users upload an image which exceeds 4- The system responds with an error message: the limit. A Image size too large 3.6.4 Video Tamper Detection Name Video Tamper Detection Actors User Summary The user shall upload a watermarked video which will then be processed by the system to detect any kind of tamper to the video Pre-Conditions The video must be watermarked by the system. Post-Conditions Users shall be able to view reports of tampering. Special Requirements Video size must not be larger than 50MB.

10**Basic Flow Actor Action System Response 1 User** uploads a video. **2**

System processes the video and notifies the user if it is tampered. Alternative Flow 3 Users upload a video which exceeds the 4- The system responds with an error limit. A message: Video size too large 3.6.5 Video Reconstruction Name Video Reconstruction Actors User Summary The user shall upload a watermarked video which will then be processed by the system to reconstruct. Pre-Conditions The video must be watermarked by the system. Post-Conditions Users shall be able to download recovered video. Special Requirements Video size must not be larger than 50MB.

10**Basic Flow Actor Action System Response 1 User** uploads a video. **2**

System processes the tampered video and reconstructs the original video for the user to download. Alternative Flow 3 Users upload a video which exceeds the 4- The system responds with an error message: limit. A Video size too large 3.6.6 Image Reconstruction Name Image Reconstruction Actors User Summary The user shall upload a watermarked image which will then be processed by the system to reconstruct. Pre-Conditions The image must be watermarked by the system. Post- Conditions Users shall be able to download recovered images. Special Requirements Image size must not be larger than 50MB.

10**Basic Flow Actor Action System Response 1 User** uploads an image. **2**

System processes the tampered image and reconstructs the original image for the user to download. Alternative Flow 3 User uploads an image which exceeds the limit. 4- The system responds with an error A message: Image size too large 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 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 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 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 ? Material UI. ? A good webserver to handle all the server-side processing load.

12**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. o For image, it is approximately 75%. o 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. 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

7**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

15**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

13**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. 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. During the authentication, number of frames in a frame window will be divided by a single random number to generate a number which will then be used as a reference to extract the number of frames in one second. For example, if we get 5 as a result of dividing fps with a random number, we will extract 5 frames from random positions in a frame window. For video restoration, we will first identify the tampered frames from the frame window and will then execute image recovery modules on the tampered frames. 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. Class diagram This section shows the class diagram of the system.

21**Figure 9: Class Diagram Figure shows class diagram of the system**

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 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

19**white box testing and black box testing**. For **white box testing**, we will test **the**

system using control flow and data flow testing. For

19**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. Implementation and Test Cases Implementation and Test Cases 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. 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

28**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] 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

7**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

15**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

13**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. 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). 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. Processing on a large number of frames for a video can be computationally very expensive and it will also depend on the type of video, for example, computation time for a 30fps video will be different from a 60fps video. This is why we have optimized our way of checking for tampering in a video, by processing the video second by second and then checking any kind of tampering attack on specific frames in that frame window. This will reduce our computation time a lot as compared to processing every single frame of a video. Moreover, due to using randomization of frame selection in a frame window, we are even reducing the probability of missing any tampered frame. The backing reason for this assumption is that the tamper attacks on videos are not usually one frame only, but on many frames. 4.3.1 Video Authentication As mentioned before, instead of checking all frames in the whole video, we will introduce randomization in the process of frame monitoring. The video will be processed in a series of frame windows. Each frame window can have different frames according to the type of video. A 30fps video can have 30 frames in 1 frame window, a 60fps video can have 60 frames in 1 frame window. The process of random frame selection includes dividing the total number of frames in a frame window by a random positive number between one and ten. The number thus obtained from this division will the be used as a reference to check the required number of frames randomly in a frame window. Figure 16: Random Frame Selection Figure shows the random frame selection process from window of frames The image authentication module will be used to check the selected random frames for any kind of tampering. In case, tampering is detected from any one of the selected frames from a frame window. We will process all frames in that specific frame window. 4.3.2 Video Restoration The tampered frames will be extracted and stored separately along with window number and index of frame in that window. Once the whole video has been processed, we will recover the tampered frames by using the image recovery module on each frame. The recovered frames will then be replaced with the tampered frames using window number and frame index. Test Case Design and Description 4.4.1 Image Watermark Image Watermark Embedding Embedding

2**Test Case ID: 4**.4.1 **QA Test Engineer**: Usama Aslam **Test case Version: 1.0** **Reviewed By**: Arbab Hamad **Test Date: 05**-18-2022 **Use Case Reference(s)**: Image watermark **Revision History: None Objective** Watermark the image **Product**

/Ver/Module: Watermarking module Environment: Software: Any modern Browser Hardware: PC Assumptions: Image should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the image and entered secret key. Image uploaded. 3 User select watermark embedding option. Image is watermarked. User is given the option to download the image Comments: Application works properly Passed Failed Not Executed 4.4.2 Image Watermark (Alternate-4A) Image Watermark Embedding Embedding

> 2**Test Case ID: 4**.4.2 **QA Test Engineer**: Usama Aslam **Test case Version: 1.0** **Reviewed By**: Arbab Hamad **Test Date: 05**-18-2022 **Use Case Reference(s)**: Image watermark **Revision History: None Objective** Watermark the image **Product**

/Ver/Module: Watermarking module Environment: Software: Any modern Browser Hardware: PC Assumptions: Image should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the image with wrong dimensions and entered secret key. Image uploaded. 3 User select watermark embedding option. Image is not watermarked, and error message displayed. Comments: Application works properly Passed Failed Not Executed 4.4.3 Video Watermark Video Watermark Embedding Embedding

> 2**Test Case ID: 4**.4.3 **QA Test Engineer**: Usama Aslam **Test case Version: 1.0** **Reviewed By**: Arbab Hamad **Test Date: 05**-18-2022 **Use Case Reference(s)**: Video watermark **Revision History: None Objective** Watermark the video **Product**

/Ver/Module: Watermarking module Environment: Software: Any modern Browser Hardware: PC Assumptions: Video should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the video and entered secret key. Video uploaded. 3 User select watermark embedding option. Video is watermarked. The user is given the option to download the image. Comments: Application works properly Passed Failed Not Executed 4.4.4 Video Watermark (Alternate-4A) Video Watermark Embedding Embedding

> 2**Test Case ID: 4**.4.4 **QA Test Engineer**: Usama Aslam **Test case Version: 1.0** **Reviewed By**: Arbab Hamad **Test Date: 05**-18-2022 **Use Case Reference(s)**: Video watermark **Revision History: None Objective** Watermark the video **Product**

/Ver/Module: Watermarking module Environment: Software: Any modern Browser Hardware: PC Assumptions: Video should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the video with wrong dimensions and enters secret key. Video uploaded. 3 User selects watermark embedding option. Video is not watermarked, and error message displayed. Comments: Application works properly Passed Failed Not Executed 4.4.5 Image Tamper Detection Image Tamper Detection Authentication

> 1**Test Case ID: 4**.4.5 **QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Image tamper detection **Revision History**: None **Objective** Authenticate **the**

watermarked image Product/Ver/Module: Authentication module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked image should be uploaded Pre-Requisite: System

should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the watermarked image and entered secret key. Image uploaded. 3 User selects tamper detection option. Tampering report is shown. Comments: Application works properly Passed Failed Not Executed Image Tamper Detection Authentication

> 1**Test Case ID: 4**.4.**6 QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Image tamper detection **Revision History**: None **Objective** Authenticate **the**

watermarked image Product/Ver/Module: Authentication module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked image should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the watermarked image and entered incorrect secret key. Image uploaded. 3 User selects tamper detection option. Error message displayed with option to re- enter the key. Comments: Application works properly Passed Failed Not Executed Image Tamper Detection (Alterante-4B) Authentication

> 1**Test Case ID: 4**.4.7 **QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Image tamper detection **Revision History**: None **Objective** Authenticate **the**

watermarked image Product/Ver/Module: Authentication module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked image should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the image which is not watermarked and entered secret key. Image uploaded. 3 Users select tamper detection option. Error message displayed. Comments: Application works properly Passed Failed Not Executed 4.4.8 Video Tamper Detection Video Tamper Detection Authentication

> 1**Test Case ID: 4**.4.8 **QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Video tamper detection **Revision History**: None **Objective** Authenticate **the**

watermarked Video Product/Ver/Module: Authentication module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked video should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the watermarked video and entered secret key. Video uploaded. 3 User selects tamper detection option. Tampering report is shown. Comments: Application works properly Passed Failed Not Executed Video Tamper Detection Authentication

> 1**Test Case ID: 4**.4.**9 QA Test Engineer**: Aashar Naseem **Test case Version: 1**.0 **Reviewed By**: Hunzlah Maiik **Test Date**: 05-18-2022 **Use Case Reference(s)**: Video

tamper detection **Revision History**: None **Objective** Authenticate **the**

watermarked Video Product/Ver/Module: Authentication module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked video should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the watermarked video and entered wrong secret key. Video uploaded. 3 User selects tamper detection option. Error message displayed with option to re- enter the key. Comments: Application works properly Passed Failed Not Executed Video Tamper Detection Authentication

1**Test Case ID: 4**.4.**10 QA Test Engineer**: Aashar Naseem **Test case Version: 1**.0 **Reviewed By**: Hunzlah Malik **Test Date**: 05-18-2022 **Use Case Reference(s)**: Video tamper detection **Revision History**: None **Objective** Authenticate **the**

watermarked Video Product/Ver/Module: Authentication module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked video should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the video which is not watermarked and entered secret key. Video uploaded. 3 User selects tamper detection option. Error message displayed. Comments: Application works properly Passed Failed Not Executed 4.4.11 Image Recovery Image Content Recovery Recovery

1**Test Case ID: 4**.4.11 **QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Image Content Recovery **Revision History**: None **Objective** Recover **the**

tampered image Product/Ver/Module: Recovery module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked image should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 Users open the web application. Main page opened. 2 User uploaded the image which is watermarked and tampered and entered secret key. Image uploaded. 3 Users select image recovery option. Image content is recovered and download option displayed. Comments: Application works properly Passed Failed Not Executed Image Content Recovery Recovery

1**Test Case ID: 4**.4.12 **QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Image Content Recovery **Revision History**: None **Objective** Recover **the**

tampered image Product/Ver/Module: Recovery module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked image should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the image which is not watermarked and entered secret key. Image uploaded. 3 User selects image recovery option. Error message displayed. Comments: Application works properly Passed Failed Not Executed Image Content Recovery Recovery

1**Test Case ID: 4**.4.13 **QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Image Content Recovery **Revision History**: None **Objective** Recover **the**

tampered image Product/Ver/Module: Recovery module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked image should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the image, which is watermarked and too much tampered, and entered secret key. Image uploaded. 3 User selects image recovery option. Image not recovered. Comments: Application works properly Passed Failed Not Executed Image Content Recovery Recovery

1**Test Case ID: 4**.4.14 **QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Image Content Recovery **Revision History**: None **Objective** Recover **the**

tampered image Product/Ver/Module: Recovery module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked image should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the image which is watermarked and tampered and entered wrong secret key. Image uploaded. 3 User selects image recovery option. Error message displayed with option to re-enter key. Comments: Application works properly

1**Passed Failed Not Executed 4**.4.**15 Video** Recovery **Video**

Content Recovery Recovery

1**Test Case ID: 4**.4.15 **QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Video Content Recovery **Revision History**: None **Objective** Recover **the**

tampered video Product/Ver/Module: Recovery module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked video should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the video which is watermarked and tampered and entered secret key. video uploaded. 3 User selects video recovery option. video content is recovered and download option displayed. Comments: Application works properly Passed Failed Not Executed Video Content Recovery Recovery

1**Test Case ID: 4**.4.16 **QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Video Content Recovery **Revision History**: None **Objective** Recover **the**

tampered video Product/Ver/Module: Recovery module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked video should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the video which is watermarked, too much tampered and entered secret key. video uploaded. 3 User selects video recovery option. Error message is displayed. Comments: Application works properly Passed Failed Not Executed Video Content Recovery Recovery

> 1**Test Case ID: 4**.4.17 **QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Video Content Recovery **Revision History**: None **Objective** Recover **the**

tampered video Product/Ver/Module: Recovery module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked video should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the video which is not watermarked, tampered, and entered secret key. video uploaded. 3 User selects video recovery option. Error message is displayed. Comments: Application works properly Passed Failed Not Executed Video Content Recovery Recovery

> 1**Test Case ID: 4**.4.18 **QA Test Engineer**: Usama Aslam **Test case Version: 1**.0 **Reviewed By**: Arbab Hamad **Test Date**: 05-18-2022 **Use Case Reference(s)**: Video Content Recovery **Revision History**: None **Objective** Recover **the**

tampered video Product/Ver/Module: Recovery module Environment: Software: Any modern Browser Hardware: PC Assumptions: Watermarked video should be uploaded Pre-Requisite: System should be working properly. Step No. Execution description Procedure result 1 User opens the web application. Main page opened. 2 User uploaded the video which is watermarked, tampered, and enters wrong secret key. video uploaded. 3 User selects video recovery option. Error message is displayed with the option of to re-enter key. Comments: Application works properly Passed Failed Not Executed Test

> 4**Metrics Metric: Purpose Number of Test Cases**: 18 **Number of Test Cases Passed**: 17 **Number of Test Cases Failed**: 1 **Test Case Defect Density: 0**.01 **Test Case Effectiveness**: 100 **Traceability Matrix: Traceability** matrix **is**

provided in a separate excel file Experimental Results and Analysis For our project, we have worked on two different research papers to compare their results so that we can extend the paper with better results to support video watermarking, authentication and recovery. Triple Recovery Information Embedding Approach Images were watermarked and authenticated using “Triple Recovery Information Embedding” approach. The embedding was tested on 512x512 gray scale image. The image was first watermarked and its PSNR value was calculated. The watermarked image was then tampered to some extent and image authentication module was tested on that tampered image. The results obtained from the implementation are as follows. Table 1: Image Watermarking (Triple Recovery Method) Following table shows details of watermark embedding process Original Image Tampered Image Execution Time (Seconds) PSNR (Watermarked Image) 2.52 50.09 From the result mentioned above, it took about 2.52 seconds to watermark a 512x512

grayscale image. An estimated time of 45 seconds was required to watermark a 4608x3456 grayscale image. Through code optimization and usage of NumPy library, we were able to reduce watermark embedding time by a large gap (see Appendix A). Table 2: Image Tampering (Triple Recovery Method) Following table shows the output image for the tampered image Tampered Image Output Image Experimental Results and Analysis The watermarked image was then tampered at two different areas and an output image was generated which marked the tampered areas. Note how the output image is a bit blurred, this is due to the image resolution. A clearer output image is obtained against a high-resolution watermarked image. Pixel-wise Authentication Method The same 512x512 gray scale image was watermarked and authenticated using Pixel-wise authentication method. Same as before, the PSNR values were calculated once again and the image was tampered to generate the output image. Table 3: Image Watermarking (Pixel-wise Method) Following table shows the details for watermark embedding process Original Image Watermarked Image Time (Seconds) PSNR Execution 2.33 38.38 This method also took almost same time however, the PSNR values were much worse as compared to the method used above. This image was then tampered and its output image was generated which is as follows. Table 4: Image Tampering (Pixel-wise Method) Following table shows the output image for the tampered image Tampered Image Output Image The output image generated is much more precise as compared to the above method but this method reduces image quality more, hence it is more of a tradeoff. Result Analysis The results obtained from both the methods were compared based on their PSNR values, execution time and output image generation. The output image defines the precision of a method. Triple recovery method works on block level, i.e., it works by dividing image into blocks. Pixel-wise method on the contrary works on pixel level and detects tampering on pixel level. This can be observed from the output images of both methods. In case of triple recovery, the output image detects the tampered area in forms of blocks which is why there are rectangular shapes on the output image. However, when the output image for pixel-wise method is generated, it can be observed that its precision is on a pixel level due to the fact that it can even show text in the output image. This is an important factor because during recovery phase, only the tampered pixels will be restored but in case of triple recovery method, the whole block will be recovery which will can be considered redundant. Looking the PSNR values of both methods, it can be observed that triple recovery method yields better PSNR values as compared to pixel-wise method. This is due the embedding process of pixel-wise method where 3 LSBs are used instead of 2 LSBs. Thus, from the results obtained from both methods, we decided to use triple recovery method for video authentication and recovery. Mainly because of the better PSNR values, we have compromised on precision as both methods can restore images. Moreover, triple recovery method can recover up to 75% of tampered image whereas, pixel-wise method is only limited to 50%. References 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. Due to a through comparison of both the methods, we were able to determine their accuracy, precision and computation time

which are important in our project. Our system can successfully watermark, authenticate and recovery images and videos without the need of any database or external storage. We are also making use serverless programming to implement multi-threading so that we can decrease processing time for video watermarking, authentication and recovery. In future, further changes can be made to our system to increase the PSNR values even more and we can also come up with methods to deal with temporal tampering. This project can also be extended as a browser extension where it will automatically watermark uploaded images. Appendix Appendix A: Time Complexity Table 5: Time Complexity for Watermark Embedding Following table shows time complexity for watermark embedding process Image Watermark embedding

14**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

7**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) (d) (e) (c) (f) Figure 17: (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 6: Time Complexity for Restoration Following table shows the time complexity for image recovering process Image

14**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 Self-Embedding Watermarking System 1 2 Self-Embedding Watermarking System 3 4 Self-Embedding Watermarking System 5 6 Self-Embedding Watermarking System 7 8 Self-Embedding Watermarking System 9 Requirements and Design 10 Self-Embedding Watermarking System 11 Requirements and Design 12 Self-Embedding Watermarking System 13 Requirements and Design 14 Self-Embedding Watermarking System 15 Requirements and Design 16 Self-Embedding Watermarking System 17 Requirements and Design 18 Self-Embedding Watermarking System 19 Requirements and Design 20 Self-Embedding Watermarking System 21 Requirements and Design 22 Self-Embedding Watermarking System 23 Requirements and Design 24 Self-Embedding Watermarking System 25 Requirements and Design 26 Self-Embedding Watermarking System

27 28 Self-Embedding Watermarking System 29 Implementation and Test Cases 30 Self-Embedding Watermarking System 31 Implementation and Test Cases 32 Self-Embedding Watermarking System 33 Implementation and Test Cases 34 Self-Embedding Watermarking System 35 Implementation and Test Cases 4.4.6 Image Tamper Detection (Alternate-4A) 36 Self-Embedding Watermarking System 4.4.7 Image Tamper Detection (Alternate-4B) 37 Implementation and Test Cases 38 Self-Embedding Watermarking System 4.4.9 Video Tamper Detection (Alternate-4A) 39 Implementation and Test Cases 4.4.10 Video Tamper Detection (Alternate-4B) 40 Self-Embedding Watermarking System 41 Implementation and Test Cases 4.4.12 Image Recovery (Alternate-4A) 42 Self-Embedding Watermarking System 4.4.13 Image Recovery (Alternate-4B) 43 Implementation and Test Cases 4.4.14 Image Recovery (Alternate-4C) 44 Self-Embedding Watermarking System 45 Implementation and Test Cases 4.4.16 Video Recovery (Alternate-4A) 46 Self-Embedding Watermarking System 4.4.17 Video Recovery (Alternate-4B) 47 Implementation and Test Cases 4.4.18 Video Recovery (Alternate-4C) 48 Self-Embedding Watermarking System 49 50 Self-Embedding Watermarking System 51 52 Self-Embedding Watermarking System 53 54 Self-Embedding Watermarking System 55