Diploma Thesis

Athanasios Koureas

Development of an Application on Smart Devices for the Creation of Watermarked Multimedia Content

Supervisor: Stavros D. Nikolopoulos

Ioannina, July 2024

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Περίληψη

Η παρούσα διπλωματική εργασία διερευνά τη δημιουργία και υλοποίηση ενός συστήματος υδατογράφησης εικόνας σε πραγματικό χρόνο με τη χρήση μιας εφαρμογής κάμερας Android. Ο κύριος στόχος είναι η ενσωμάτωση δεδομένων που αφορούν τον χρήστη, όπως ένα προσωπικό PIN και το μοναδικό για κάθε συσκευή Android ID, σε ψηφιακές εικόνες με τη χρήση ενός ισχυρού αλγορίθμου υδατογράφησης. Χρησιμοποιώντας τη βιβλιοθήκη CameraX και ενσωματώνοντας αρχεία Python μέσω της βιβλιοθήκης Chaquopy, αναπτύξαμε ένα σύστημα που εξασφαλίζει την ασφάλεια και την αυθεντικότητα των ψηφιακών εικόνων που λαμβάνονται σε πραγματικό χρόνο. Εκτεταμένες δοκιμές έδειξαν ότι αυτό το σύστημα προστατεύει αποτελεσματικά τις ψηφιακές εικόνες από μη εξουσιοδοτημένη χρήση και αλλοιώσεις. Η μέθοδος υδατογράφησης είναι ιδιαίτερα ανθεκτική σε κοινές επιθέσεις όπως η περικοπή και η συμπίεση, αν και η απόδοση του συστήματος μπορεί να διαφέρει ανάλογα με την επεξεργαστική ισχύ της φορητής συσκευής. Η μελλοντική έρευνα θα μπορούσε να επικεντρωθεί στη βελτιστοποίηση του αλγορίθμου, επεκτείνοντας τη χρήση του σε άλλους τύπους μέσων όπως βίντεο και ήχος.

Λέξεις-κλειδιά: υδατοσήμανση εικόνας, εφαρμογή κάμερας Android, αλγόριθμος υδατοσήμανσης, προσωπικό PIN, android ID, επεξεργασία εικόνας, ενσωμάτωση δεδομένων, ζωντανή υδατοσήμανση, ψηφιακή αυθεντικότητα, κατάτμηση εικόνας, επίπεδο, μπλοκ, πλέγμα, κελί, διαδικασία ενσωμάτωσης.

Abstract

This thesis explores the creation and implementation of a real-time image watermarking system using an Android camera application. The main goal is to embed user-specific data, such as a personal PIN and the device-specific Android ID, into digital images using a robust watermarking algorithm. By utilizing the CameraX library and integrating Python scripts through the Chaquopy plugin, we have developed a system that ensures the security and authenticity of digital images captured in real-time. Extensive testing has shown that this system effectively protects digital images from unauthorized use and alterations. The watermarking method is highly resistant to common attacks like cropping and compression, although the system's performance may vary based on the device's processing power. Future research could focus on optimizing the algorithm, expanding its use to other media types like video and audio.

Keywords: Image Watermarking, Android Camera Application, Watermarking Algorithm, personal PIN, android ID, Image Processing, Data Embedding, Live Watermarking, Digital Authenticity, image partition, plane, block, grid, cell, embedding process.

Contents

Chapter 1. Introduction

In the first chapter of this thesis, we introduce the essential techniques for embedding hidden information in digital images, with a focus on digital watermarking using an Android camera. We review recent research, discuss the importance of digital image watermarking in general so as also live watermarking, and outline our objectives to create a robust real-time watermarking system that ensures the security and authenticity of digital images.

1.1 Image Watermarking

Image watermarking is a method used to protect ownership of digital images through the insertion of a watermark into it. This watermark functions as a brand or logo to verify that a person or company owns a particular image and to prevent it from being altered. Good watermarking scheme means the feature of the image which can be watermarked as well as the ability to extract the watermark from the image and also the ability to resist changes made to it. Image watermarking has become highly important as the daily exchange of image data continues to expand. As the digital images are important resources in business environment, their protection is crucial. This protection spreads not only the commercial data but also the images in the social networks, so that the rights of the intellectual property and the rights of the personal data are protected. For instance, image watermarking plays a critical role in ensuring that images used in advertising, production of content for the internet and other social media platforms are protected. However, image watermarking plays a significant role in industries, especially in the entertainment industry, as it is vital to protect digital media products' copyrights. These has necessitated the development of the watermarking techniques given that data is often considered to be valuable and sensitive. Advancements in computing technologies has seen advanced watermarking techniques being developed, and with this there is high assurance that the watermark that is to be embedded remains invariant and can be easily retrieved after going through several transformations and attacks. The advancement in this area of watermarking shows the significance of the method in preserving the integrity of images in today's world of growing technological advancement.

1.2 Live Watermarking

Live watermarking is an extension of the conventional watermarking that embeds watermarks into the images or frames of the video stream as it is being captured. This technique is very relevant for use in real-time applications like live streaming, surveillance, and real-time data transfer, for which content protection is imperative at first instance. The first problem in live watermarking is to obtain real-time processing while maintaining the strength and invisibility of the watermark. This calls for fast algorithms that are capable of analyzing each frame in order to match the rates at which the camera is capturing. In live watermarking and especially for digital images, the watermarking process is integrated into the image capture pipeline, involving the following steps:

Image Capture:

The first step involves acquiring the digital image from the camera in real-time. This process must be efficient to ensure that the digital image is captured at a consistent rate without any delays or loss of quality. Then, the captured image may be buffered temporarily to manage the flow of data and prepare it for the next stage of processing.

Watermark Embedding:

Then, the captured image is processed through the watermarking algorithm. The algorithm embeds a unique identifier (watermark) into the digital image, ensuring that it is both robust and imperceptible to the viewer. This step must be performed quickly to maintain real-time processing. For this reason, in this step it is critical to choose a watermarking algorithm that is both efficient in terms of image quality and robustness to different attack scenarios, and as fast as possible in its execution.

Image Display/Storage:

The watermarked image is then displayed in real-time. This step involves rendering the processed image onto a display device, ensuring that there is no noticeable delay between capture and display. Alternatively, or in addition, the watermarked digital image may be stored for later use. This involves saving the processed image to a storage medium, ensuring that the watermark remains intact and retrievable even after compression or other transformations.

When incorporating live watermarking to a camera application, one is required to insert the watermarking process in the imaging and processing chain. This needs to be done cautiously so as not to make the extra processing time apparent or to slow down the application. This project introduces a live watermarking system in Android with the help of the CameraX library. CameraX is convenient and flexible in using the camera features and taking pictures which is suitable for this application. The watermarking algorithm is also coded in Python and the Chaquopy plugin to include Python scripts into the Android application. In the next parts of the article, further description of the implementation of the application will be presented, which will include detailed explanation of the used watermarking techniques, as well as the process of integration of the CameraX library.

1.3 Related Work

In [1], Hashmi et al., developed a real-time copyright protection algorithm that uses both visible and invisible watermarking techniques. The authors used this approach for real-time image and video processing on Android and embedded platforms to enhance privacy protection. Chroni et al., proposed in [2] a comparative study of two watermarking techniques, both based on frequency domain transformations but differing on watermark construction. Many state-of-the-art approaches utilize binary images as watermarks. An alternative way of watermark construction, which incorporates error correction properties, proposes the transformation of an integer number w into a self-inverting permutation (SiP). SiP can be represented in a two-dimensional (2D) object, since images are 2D structures, and that constitutes a watermark. Chroni et al., in [3], introduced a novel watermarking scheme that repetitively applies watermarks throughout a digital image. This method strengthens the image's defense against crop and compression attacks by embedding the entire watermark into multiple sections of the image. H.Al-Otum et al., in [4] developed a color image watermarking scheme specifically for Android-based smartphones. They adapted traditional image watermarking techniques to be more suitable for power-limited smartphones, ensuring effective copyright protection even on mobile devices. As mentioned in [5] by C.P. Hernandez et al., the security challenges posed by the rapid spread of digital images via the Internet and wireless communication. They emphasized the importance of fragile watermarking for image authentication and tampering localization, demonstrating that mobile devices now have the capability to support such applications. Y. Zhang et al. (see [6]) examined the issue of app repackaging attacks on smartphones. To mitigate this attack, they embed watermarks into Android apps. Specifically, to make the watermarks robust, they embed a new kind of watermark called picture-based watermarks into Android apps. In [7], Chroni et al. proposed efficient codec algorithms for watermarking images intended for uploading on the web under intellectual property protection. H.B. Al-Gahtani et al., as cited in [8], highlighted the increasing challenge of protecting digital multimedia, especially images, on smartphones. They proposed a flexible watermarking system that embeds watermarks into RAW images on Android devices, offering protection from the very beginning of the image processing stage. As described by N. Miao et al. in [9], the fast growth in the ubiquitous use of digital-camera-equipped smartphones has created a need to protect multimedia data on battery-powered mobile devices. They presented the design and implementation of an efficient digital watermarking application, hymnMark, to perform watermark embedding and detection for digital images on the Android platform. Finally, in [10], H. Al-Otum et al. proposed a color image watermarking scheme for copyright protection tailored for Android-based smartphones. They modified characteristic-based image watermarking techniques to be suitable for power-limited smartphones. The article in [11] focuses on the importance of watermarking for securing and managing medical information, particularly in maintaining its integrity and authenticity. The authors highlight real-world applications of watermarking in healthcare systems, noting its valuable contribution. In [12], Kahlessenane et al. propose a robust, blind approach to protect medical images during telemedicine exchanges, enhancing security by embedding patient information and image data within the image with minimal distortion, preserving clinical interpretation. Alshanbari's approach in [13] aims to improve medical image watermarking by using multiple watermarks of the host image. This method employs principal components-based insertion to bolster security against ownership attacks. The work in [14] by Khare and Srivastava introduces a new medical image watermarking technique for telemedicine, utilizing homomorphic transform (HT), redundant discrete wavelet transforms (RDWT), and singular value decomposition (SVD) to create an effective watermarking method. Narima Zermi et al. in [15] suggest a

technique to enhance the security of medical image sharing and transfer by watermarking images with hospital signature information and patient data. A notable approach by Vaidya in [16] improves authentication and privacy of medical images using the patient's fingerprint as a watermark. This method ensures the image's authenticity by embedding the fingerprint, supported by lifting wavelet transform (LWT) and discrete wavelet transform (DWT) techniques. Singh et al. in [17] propose dividing medical images into the Region of Interest (RoI) and the Region of Non-Interest (RoNI), inserting tamper detection and recovery bits into the RoI to maintain its integrity. Balasamy et al. in [18] introduce an algorithm that preprocesses clinical images using the Switched Mode Fuzzy Median Filter (SMFMF) to replace noisy pixels with median pixel values. Moad et al. in [19] present a watermarking technique focused on patient identification and watermark integrity verification by splitting the watermark into two parts: the patient's information fingerprint and the encrypted image of the patient. Finally, in [20], a watermarking technique for medical images in e-healthcare systems is described, where the medical image is treated as a vector picture and the watermark is a replica of the patient's health card. Both the image and the watermark are modified using the Discrete Wavelet Transform (DWT) for embedding.

1.4 Motivation and Contribution

The rationale for this work has arisen due to the growing demand for methods of real-time protection of digital images. Due to the current use of smartphones and social media platforms, users are always taking pictures and sharing them at any one time thus making them vulnerable to violation and unauthorized use. Existing techniques of watermarking employs image processing methods where the watermark has to be placed after the image has been captured, and hence are not suitable for real time applications. This work aims to address the need for real-time image protection by developing a live watermarking system that integrates seamlessly with an Android camera application. Key contributions of this project include ensuring that watermarks are robust against alterations like cropping. Traditional methods often struggle to verify authenticity in altered or truncated segments due to insufficient hidden data. Our approach overcomes this by distributing the entire watermark across larger areas of the image using the 2DM representation of Self-inverting Permutations (SIPs). By dynamically positioning the grid within image blocks, we improve both the embedding process and watermark extraction, minimizing quality loss. This makes our method highly resistant to attacks and effective for real-time use. We have embedded this watermarking algorithm into an open-source Android camera application using the CameraX library. This allows users to capture watermarked images directly through their camera, making the process straightforward and user-friendly. Additionally, our system creates unique watermarks by combining user-specific inputs, like a personal code (e.g., a PIN), with device-specific information, such as the Android ID of the mobile device, which enhances the security and traceability of the watermarked images. To ensure the system's effectiveness, we conducted thorough tests. The results confirm that our watermarking system reliably protects digital images in real-time settings, demonstrating its robustness and practicality for everyday use. By addressing the challenges of real-time watermarking on Android-based devices, this thesis not only enhances the protection of digital images but also paves the way for further advancements in real-time image sharing and authentication.

Chapter 2. Theoretical Background

This chapter lays the groundwork for our real-time image watermarking system. It explores the concepts of hiding information in digital images, including how to encode integers as Self-inverting Permutations (SiPs) and embed them into digital images using their Two-Dimensional Matrix (2DM) representation. This chapter also covers techniques for segmenting images to facilitate this process. Additionally, we discuss the software tools and frameworks that make real-time image watermarking possible, focusing on the use of open-source camera applications and ensuring compatibility with Android devices through integration with Android Studio.

2.1 Information Hiding in Digital Images

2.1.1 Encoding Integers as Self-Inverting Permutations

There are different approaches to convert integer numbers into permutations and self-inverting permutations. The embedding algorithm proposed by the authors in research [7] utilizes a unique watermark to indicate the cells where the SiP algorithm will be reapplied. In recent research, there were developed algorithms that efficiently encode an integer w and decode it to get a Self-inverting Permutation π. These algorithms take O(n) time in which n is the number of digits in the binary representation of w. The principles on which these algorithms are based are called bitonic permutations, which are an important component of the encoding and decoding process. The encoding algorithm starts with the given integer w to transform it to its binary form namely B. These binary sequences are then extended and complemented to form another sequence known as the sequence of 'star' or B* which defines the indices required to form two other sequences: X and Y. These subsequences are then merged and transformed into a bitonic permutation which is required to construct the Self-inverting Permutation π. Finally, in order to embed this Self-inverting Permutation (SiP) to the digital image, the permutation is transformed into its 2DM representation using the method described in the next sub-section.

2.1.2 Embedding Integer to Image (2DM)

The Two-Dimensional Matrix (2DM) technique is a sophisticated method used to embed encoded integers into digital images. This approach involves breaking the image down into specific regions and subtly embedding the encoded data within these areas. The primary goal is to ensure that the hidden information is invisible to the human eye but can be easily read by a computer program. The process starts by dividing the image into smaller, manageable blocks. Each of these blocks is then further divided into grids or cells. These smaller areas are where the encoded data will be hidden. By partitioning the image in this way, the data embedding process becomes more efficient and less detectable. The next step involves modifying specific pixels within the cells to encode the data. These modifications are tiny and strategically placed to ensure they do not alter the overall appearance of the image. However, they are structured in such a way that the encoded information can still be extracted by specialized decoding algorithms. The 2DM technique is a highly structured and effective method for embedding data in digital images. By carefully dividing the image into smaller sections and optimizing the embedding process, this technique ensures that the watermark remains both resilient to tampering and invisible to viewers. This makes it particularly valuable in applications where the integrity and authenticity of digital images are of utmost importance, such as in copyright protection and digital forensics.

2.1.3 Image Segmentation

To embed the authenticity information into an image, the image is segmented into several structural elements. Here's how it's done:

- **Plane:** A plane is utilized to define specific areas within the image, with dimensions (n x m). This plane divides the entire digital image into an even number of equally sized, non-overlapping blocks to accommodate the watermark. The common divisors of (n, m) are used to determine a value, denoted as d, which represents the size of each block. The blocks are square shaped with dimensions (d x d), resulting in a total of d^2blocks.

- **Blocks:** The plane is subdivided into even smaller sections, the so-called blocks. A block has dimensions ($\frac{M}{d} \times \frac{N}{d}$). Every block will contain a part of the watermark, that will guarantee the authenticity of that part of the digital image. In that way, there is an integer of a sequence residing in each block.

- **Grid:** In each block, it is possible to distinguish the smaller rectangular area that we call a grid. This grid is located precisely in the middle of the block and the dimensions of this grid are (m′ × m′), where m' equals the minimum of the block's width and height. At this point it is worth noting that in our approach the watermarking algorithm uses a more optimal way of placing the grid inside the block.

- **Cells:** The cells are the smallest structures in this partitioning method but at the same time the most crucial components. This information of the hidden watermark is stored in selected cells of the array. The typical dimensions of a cell are ($\frac{\mathrm{m}'}{c} \times \frac{m'}{c}$) pixels. The technique utilized for storing hidden information within these cells is based on the approach described in [3]. This hidden

information plays a vital role in evaluating the integrity and genuineness of digital forensic images.

This hierarchical partitioning makes the watermark spread uniformly over the area of the digital image, which makes it a reliable technique for the image authentication.

2.2 Real-time Image Watermarking

2.2.1 Software Basis

The foundation of the software solution for real-time image watermarking includes the application of sophisticated algorithms and frameworks for the purpose of watermarking digital images in real time. The first and foremost is that the watermarking should not be very time consuming. Among the critical pieces of this software is the use of complex algorithms in the actual watermarking of the material. These algorithms are the ones that create an imprint of a digital signature that shows that the image belongs to a particular owner. The problem is to achieve this in the shortest time possible, and at the same time make the process resistant to different types of fraud or malicious attempts. The watermarking algorithm employed in this system work in the manner that a unique watermark is generated by the combination of user related data like a PIN and the device related data like the Android ID of the mobile device where the digital image was captured and stored. This makes the watermark unique to the user and the device thus increasing the security and the ability to trace the watermarked images. To reach the real-time processing, the following steps must be solved by the software effectively. First, an image is captured with the help of the device, which is usually a built-in camera. The captured image is then pre-processed to fit the right format for watermarking to be done on it. This pre-processing step may encompass resizing of the image, conversion of the image to an appropriate format and format for the embedding process. The watermarking process itself is done with the help of a Python script that is embedded into the Android application. This script achieves the embedding operation through the placement of the unique watermark into the image data. Python enables one to implement complicated algorithms which, due to the Python libraries and frameworks, can be run smoothly. During each of these steps, it is critical to involve the software to ensure that the watermark is placed in the image without compromising its quality. The objective is to achieve the invisibility of the watermark to the human observer while at the same time ensuring its resistance to possible manipulations. This is a delicate process of achieving the right levels of embedding strength while at the same time maintaining the image integrity. Also, the software has components for the management of the initiation and the settings of the watermarking procedure. For instance, it determines if the system is in its initial state and configures suitable variables if that is the case. This setup may include copying necessary files to local storage, setting up environment variables, and starting up any necessary services or libraries. In conclusion, the software foundation of real time image watermarking can be defined as a complex of the modern image processing methods and the effective methods of the watermarking. Using Python and the incorporation of the language into an Android application, the system can offer a quick, secure, and efficient method for adding watermarks to digital images in real-time. This foundation gives an assurance that digital images can be protected and authenticated as they are being captured and shared.

2.2.2 Open-Source Camera Application

To implement real-time image watermarking, we use a freely available camera application platform built on the CameraX library. CameraX is a robust and flexible Android framework that simplifies camera operations while providing advanced features for handling photos and videos. The CameraXHelper class in the code illustrates how to manage different camera tasks, such as taking pictures, recording videos, and processing the captured media, using CameraX.

- **Setting Up the Camera:** The CameraXHelper class starts by setting up the camera. It links the camera's lifecycle to the app's lifecycle, ensuring the camera is managed effectively and its resources are released when not needed. This setup includes configuring the camera for previewing, capturing images, and recording videos, making sure everything works smoothly for the user.

- **Taking Photos and Recording Videos:** The class has separate components for taking photos and recording videos. When taking photos, it sets up the camera's rotation and aspect ratio to ensure high-quality pictures. For video recording, it selects the best video quality settings to deliver clear and high-resolution videos.

- **Real-Time Image Analysis and Barcode Scanning:** Besides taking photos and videos, the CameraXHelper class can analyze images in real-time, including scanning barcodes. It processes each frame on-the-fly, enabling tasks like barcode detection and other image analysis activities.

- **Controlling the Flash and Focus:** The class includes features to control the camera's flash (torch) and focus settings. This allows for adjustments to the lighting and focus, improving the quality of photos and videos in different lighting conditions.

- **Managing Camera Lifecycle**: By observing the camera's flash state and linking the camera provider to the app's lifecycle, the CameraXHelper class ensures that the camera operations are integrated seamlessly with the app. This means the camera is ready when needed and properly shut down when not in use, preventing any resource leaks.

- **Switching Between Photo and Video Modes:** The CameraXHelper class allows users to switch between taking photos and recording videos without needing to reinitialize the camera. This makes the app more efficient and user-friendly.

- **Displaying the Camera Preview**: The preview configuration makes sure that what the camera captures is displayed correctly on the screen. It handles the preview's display settings and touch events for adjusting focus and metering, ensuring a smooth user experience.

In summary, the CameraXHelper class makes full use of the CameraX library to manage camera operations effectively in an Android app. By packaging these functionalities into one helper class, the code demonstrates how to seamlessly integrate advanced camera features, supporting real-time image watermarking while maintaining performance and a great user experience.

2.2.3 Android Studio – Device Integration

Integrating the watermarking application with Android Studio and ensuring it works well across various Android devices involves setting up the development environment, adding necessary dependencies, and configuring them for optimal performance. This process ensures the application runs smoothly on different devices.

- **Setting Up the Development Environment:** First, we need to configure the development environment in Android Studio correctly. This involves setting up the project structure, including specifying the SDK versions, application ID, and other configuration settings in the build.gradle file. These settings ensure that the application is compatible with a wide range of devices and Android versions.

- **Managing Dependencies:** Managing dependencies is a crucial part of the development process. The build.gradle file lists all the necessary libraries and frameworks for the project, such as CameraX for camera operations and Chaquopy for running Python scripts within the Android app. Ensuring all dependencies are correctly listed and configured allows the application to use these tools effectively.

- **Integrating Python Scripts:** To incorporate Python algorithms into the Android app, we use the Chaquopy plugin. Chaquopy allows us to run Python code seamlessly, enabling the execution of complex algorithms for image processing and watermarking. The configuration in the build.gradle file includes installing necessary Python packages, ensuring the Python environment is set up within the app.

- **Ensuring Compatibility and Performance:** The build.gradle file also includes configurations to ensure the app performs well on different devices. This involves setting the appropriate compile and target SDK versions, enabling support for vector drawables, and specifying ABI filters for the Native Development Kit (NDK). These settings help the app run efficiently across various Android devices, providing a consistent user experience.

By following these steps, the project establishes a robust and effective real-time image watermarking solution on Android. Using Java/Kotlin for the Android components and Python for the image processing algorithms ensures the application delivers optimal performance and usability. This integration supports the seamless embedding of watermarks into images, enhancing the security and authenticity of digital media.

Chapter 3. The Model

This chapter delves into the model behind our real-time image watermarking system. We explore how we embed multiple integers (code-sequence) into an image and set various grid displacements. Additionally, we discuss how user-defined data, like PINs and grid positions, are combined with device-specific information, such as the Android ID, to create a unique watermark. We also explain how the embedding algorithm is integrated with the camera application. This includes capturing digital images, embedding the code-sequence into the images, and exporting the final watermarked images. Each section provides a detailed explanation of these processes to ensure a comprehensive understanding of how the model is implemented.

3.1 Image Watermarking

3.1.1 Embedding Multiple Integers (code-sequence) to Image

The process of embedding a user-generated code into an image starts by splitting the image into 36 different sections, where each section represents one digit of the 36-digit code. At this point we should emphasize that these 36 digits of the code consist of a combination of the Android ID of the mobile device as well as the user's password (e.g., the PIN). This method, derived from research [3], begins with transforming the user code into a format made up entirely of class 11 watermarks(e.g., integers w in range [16-31]) using a specific encoding algorithm. Following this, we use an embedding algorithm that requires two main inputs: each digit of the code and the corresponding image block. Each digit of the code is converted into a Self-inverting Permutation (SiP), which is then embedded into its respective section of the digital image. This algorithm processes each digit one by one, embedding it into the image and returning the section of the image with the watermark applied. By repeating this for all 36 digits, we end up with 36 watermarked blocks of the digital image captured by the mobile device, each containing one digit of the user’s code. These blocks together hide the entire code within the image, ensuring that the encoded information is securely embedded within the digital content. Finally, it is necessary to emphasize that the embedding algorithm described above performs the process of watermark embedding in an optimized way, since during the execution of the algorithm an optimization algorithm is executed in parallel, which aims to optimize the embedding of each digit (watermark) of the code in the corresponding block in terms of the quality of each block and, by extension, the overall quality of the digital image and also the robustness of the processed image.

3.1.2 Setting Multiple Grid Displacements

An alternative way to position the grid within a block is to move the grid continuously. First, the dimensions of the grid are smaller as far as pixels are concerned. For example, if each grid cell was initially 24×24 pixels, we now reduce it to 12×12 pixels depending mainly on the image dimensions which usually reach 4K. This makes the grid occupy only half the space it used to. We start by placing the grid in the upper left corner of the block. Next, we shift the grid one position to the right until it reaches the upper right corner of the block. Moving one position to the right means that if a grid cell is 12×12 pixels, the entire grid moves 12 pixels to the right. After the grid reaches its final horizontal position, we move it back to the original position but shift it down by a few pixels, say 12 pixels in this case, along the vertical axis. Eventually, the grid will end up in the lower right corner of the block. This step-by-step approach ensures that the grid covers all possible positions within the block, allowing for comprehensive embedding of information across the entire block area. The offset for both rows and columns of the image is measured in pixels, and for a typical high-resolution image, the values are adjusted accordingly. In the context of the application developed in this thesis, because the computational cost of the grid placement algorithm mentioned above is quite high and therefore quite time consuming, we needed to develop a faster version of the algorithm that randomly selects the placement of the grid inside the block by selecting and recording the offsets inside the block on the x and y axes. This small change allows the watermarking of the digital image to be performed in 1 to 2 minutes on a standard Android mobile device.

3.2 Combining User-defined and Standard Device Information

3.2.1 User Data

In watermarking techniques, user data like a PIN and specific grid positions within the image are essential for making each watermark unique and secure. This personalized data ensures that the watermark is closely tied to the user's input, providing both security and traceability.

- **PIN:** The Personal Identification Number (PIN) is a unique identifier defined by the user. This code acts like a key and is used during the watermarking process to create a unique watermark. The user is asked to enter a 4-digit number through a simple user interface dialog. This PIN is then incorporated into the encoding process, ensuring that the resulting watermark is distinct and secure for that user. The PIN of the user is integrated with the code-sequence that in the end we embed to the captured image.

- **Grid Positions:** These are specific spots within the image where parts of the watermark are embedded. The exact positions are calculated during the watermark embedding process. This ensures the watermark is spread across the image in a way that makes it hard to tamper with. The placement of these grid positions depends on the image's dimensions and properties as well as the user's PIN, to ensure the watermark is robust and effective.

By using user-defined data like the PIN and grid positions, the watermarking process becomes more personalized and secure. Each watermark is unique to the user's input, making it difficult for unauthorized users to duplicate or alter the watermark. This method not only enhances security but also gives users control over the watermarking of their digital images. At this point we have to say that specifically the placements of the grid within each block are calculated only during the first execution of the application by the user, i.e. practically when the first image is captured. The placements are written to an encrypted file which, during the capturing of all subsequent images by the user, is decrypted in order to read the grid placements of each block and thus watermark the images.

3.2.2 Device Data

Device information, particularly the Android ID, is a key component in the watermarking process as it provides a unique identifier for each device. This identifier ensures that the watermark is uniquely tied to the specific device, enhancing both security and individuality.

- **Android ID**: This is a unique identifier assigned to every Android device. It acts like a digital fingerprint, ensuring no two devices share the same ID. By using this ID in the watermarking process, the system can ensure that each watermark is uniquely associated with the device it was created on.

To retrieve the Android ID, the system accesses the device's secure settings. This ID is a string of characters that remains consistent unless the device undergoes a factory reset. By incorporating this ID into the watermark, the system adds an extra layer of security, making each watermark unique to both the user and the device. In practice, the watermarking application fetches the Android ID and formats it into a UUID-like structure for consistency and ease of use during the encoding process. This formatted ID becomes part of the embedded data within the image, enhancing the security and traceability of the watermark. Using the Android ID in the watermarking process ensures that the watermark is not just a generic mark but is specifically tied to the unique identifier of the device. This approach significantly reduces the risk of watermark duplication, as the watermark would be invalid if transferred to another device. Thus, incorporating device data like the Android ID is crucial for making the watermarking process secure, and personalized.

3.2.3 Combining User and Device Data

The integration of user and device data involves merging user-specific details, such as the PIN and grid positions, with the device-specific Android ID to create a unique code-sequence. This sequence is then embedded into the image, ensuring that the watermark is both unique and closely tied to the specific user and device.

- **Retrieving and Combining Data:** To merge the user data with the device data, the necessary information is retrieved from shared preferences and then passed to a Python script that handles the embedding process. In the CameraXHelper class, a function extracts the user's PIN and the device's Android ID, which is formatted to ensure consistency and uniqueness. These two pieces of

information are combined to create a code-sequence that will be embedded into the image.

- **Combining Data in the Python Script:** The Python script takes the combined data (the Android ID and the PIN) and processes it to generate a sequence that can be embedded into the image. This sequence ensures the watermark is unique to both the user and the device, enhancing the security and individuality of the watermark.

- **Enhancing Security and Uniqueness:** By combining the Android ID and the PIN to generate a unique code-sequence, the watermarking process becomes highly specific to both the user and the device. This significantly increases the security of the watermark, making it difficult for unauthorized users to replicate or alter it.

The unique combination of user-defined and device-specific data ensures that each watermark is distinct and securely embedded within the image, providing strong protection against tampering and unauthorized use. The Python script is crucial in this process as it handles the actual embedding of the combined data into the image. This ensures that the watermarking is done efficiently and securely, maintaining the integrity and authenticity of the digital content.

3.3 Aligning the Embedding Algorithm with the Camera Application

3.3.1 Intermediate Digital Image Capturing

Intermediate digital image capturing involves several key steps to ensure each image is ready for watermarking. This process is managed within the CameraXHelper class, where specific functions handle the immediate steps after the image is captured. When an image is taken, a callback function is triggered to save the image with a unique timestamp. This ensures each image file has a distinct name, preventing any accidental overwriting. The image is then stored in a specific directory within the app's storage space. Next, the image is resized to meet the necessary dimensions for watermarking. This step is crucial as it ensures the watermark can be properly embedded without compromising the image's quality. The resized image is then saved again. The final step in this intermediate capturing process is preparing the image for watermarking. This is done by a function in the CameraXHelper class that processes the image and calls a Python script. This script embeds the watermark, which is created using both user and device data, into the image. This sequence ensures that each captured image is promptly processed and ready for watermarking, maintaining the real-time aspect of the application. It guarantees that images are not only captured efficiently but also securely prepared with watermarks, enhancing their protection and authenticity.

3.3.2 Embedding Code-Sequence to Image

The process of embedding a user-generated code into an image begins by splitting the image into 36 separate sections, each representing one digit of the 36-digit code. This step is crucial for preparing the image for the embedding process. Next, as we mentioned in subsection 3.1.1, the user code is converted into a sequence entirely made up of specific watermark patterns, ensuring it's in the right format for embedding. The embedding technique then takes over, using two main inputs: the digit of the code and its corresponding image section. Each digit is transformed into a unique pattern through an encoding process and is embedded into its respective section of the image. By repeating this process for all 36 digits, we create a complete watermarked image, with each section containing part of the user's code. Additionally, a Python function manages the embedding of this code sequence into the captured image. It takes the image and the data as inputs and processes the image to embed the code. Initially, the function sets up the necessary parameters and opens the image. The image is then converted into a format suitable for embedding. The user's code is combined with a PIN to create a unique sequence, which is then mapped and converted into special patterns. The image is divided into smaller sections, with each section corresponding to a part of the sequence. For each section, the sequence data is encoded and embedded. Once all sections are processed and embedded, they are merged back into a single image, resulting in a watermarked image that securely holds the encoded information. This method ensures that the watermark is well-hidden and can withstand different types of image manipulations.

3.3.3 Exporting Watermarked Image

After the watermarking process is completed, the next important step is to properly export and save the watermarked image. This involves defining the specific directory path where the watermarked images will be stored. In this application, the watermarked images are saved in the directory:

- /storage/emulated/0/Android/data/com.vungn.camerax/files/Pictures/CameraXApp/watermarked

If this directory does not exist, it is created to ensure there is a designated place for storing the watermarked images. Handling the file name and path for the watermarked image is crucial. It's essential to make sure that any existing files with the same name are removed to avoid conflicts and ensure the uniqueness of each saved watermarked image. This step helps maintain the integrity of the saved images and prevents overwriting issues. The process also includes rotating the image to ensure it is saved with the correct orientation. This adjustment is important to ensure that the final output appears as intended, with the right alignment and presentation. Finally, the image is saved with high-quality settings to ensure that the watermark remains intact, and the overall image quality is preserved. By following these steps, we ensure that the watermarked image is not only created but also securely saved in the correct directory. This careful handling ensures that the watermarked image is securely saved, making it easily accessible and protected within the application's file structure. This approach maintains the integrity and usability of the watermarked content, making the digital watermarking process both effective and reliable.

3.4 Extracting Information from Watermarked Images

3.4.1 Retrieving the Embedded Code Sequence

To retrieve the user code from the watermarked image, we rely on the established mapping between the user code and the generated watermark sequence. This mapping links specific watermarks to corresponding digits. During the extraction process, when we pull a watermark from a block, we compare its value against the mapping to determine the associated digit. For example, if the mapping indicates that watermark W = 21 represents the digit 4, then extracting watermark W = 21 from any block will place the digit 4 in the corresponding position of the hexadecimal code. This is done for each of the 36 blocks in the image. As the algorithm extracts watermarks from each block, the digits are filled in their respective positions in the hexadecimal code. By consistently applying this procedure across all blocks, we successfully reconstruct the entire user code, ensuring that the correct digits are assigned based on the extracted watermark sequence. This way, the mapping between watermarks and code digits enables the accurate retrieval of the user code from the watermarked image.

3.4.2 Computing Extraction Rate

The extraction rate, denoted as *Re*, is a measure of the efficiency of the watermark extraction process. It is calculated using the formula below:

$$Re = \frac{|\text{code digits extracted}|}{|\text{total code digits}|}$$

In this formula:

- |code digits extracted|: Represents the number of digits successfully retrieved from the watermarked image.
- |total code digits|: Represents the total number of digits originally embedded in the image. In our case the total number of code digits is 36.

Currently, this calculation has not been implemented, but it provides a theoretical basis for evaluating the performance of the extraction process.

Chapter 4. System Architecture and Deployment

This chapter focuses on explaining the core parts of our proposed real-time image watermarking system. We aim to give you a clear understanding of the key elements, such as the watermarking algorithm we use, the user credentials needed, the Android camera application, and the permissions required. We will guide you through the steps involved in capturing images, embedding watermarks, and storing the images securely. Moreover, we will discuss the system's modular design and the deployment procedures that help make the capturing, embedding, storing, and validating of watermarked images smooth and efficient.

4.1 System Components

4.1.1 Algorithm

The core of our system is a sophisticated watermarking algorithm designed to embed user-specific data, such as a PIN, into digital images. The process starts by dividing the image into multiple blocks, with each block holding a part of the user data. Specifically, in our case we embed in the image that we capture through the camera application, a 36-digit code where each block of the digital image receives the corresponding digit of the code. This ensures that the watermarks are spread evenly across the image, making it more resilient against typical image alterations like cropping, resizing, and compression. The algorithm uses advanced mathematical techniques to securely convert the user data into a watermark. The main tool used by the watermarking algorithm that we incorporate in our android application is the Discrete Fourier Transform (DFT) which gives the ability to process the digital image in the frequency domain resulting in the hidden placement of the watermarks within the digital image. In that way, these watermarks are then embedded into the image in a way that is both difficult to detect and remove. The strength of this approach lies in its ability to withstand various attempts to tamper with the image while still maintaining the image's integrity and quality. This means that the watermarks can be reliably extracted when needed, verifying the image's authenticity and confirming its ownership. For that reason, during the processing and embedding of each watermark, which through an encoding process is derived from the corresponding digit of the user code, in the corresponding block of the digital image, at the same time, as we have mentioned earlier, an optimization algorithm is running which undertakes to optimize the process of embedding, maximizing the quality of the image but at the same time minimizing the probability of unsuccessful extraction of the watermark even under the scenario where the image has

suffered a malicious attack. In research [2], the authors mention the parameter C which is a parameter that affects both the fidelity of the digital image and also its robustness. The optimization algorithm is running with a goal to fine-tune this parameter.

4.1.2 Credentials

User-specific credentials play a crucial role in our watermarking system. This includes data such as a PIN that the user can provide to create their unique watermarks. This information is combined with device-specific details, like the Android ID, resulting in a watermark that is uniquely tied to both the user and his device. This dual-layered security approach greatly enhances the robustness of the watermark, making it much harder for unauthorized individuals to replicate or modify it. By linking the watermark to specific user credentials and device identifiers, the system ensures that each watermarked image can be traced back to its original source, thereby increasing security and accountability. It is necessary to emphasize at this point that the final code that is embedded in the digital image results from the union of the android device's identifier (e.g., Android ID) with the user's personal PIN. The Android Id consists of 16 digits of the hexadecimal system and the user's PIN consists of 4 digits of the decimal system. The way in which the 36-digit code is derived is shown in the formula below:

36-Digit Code = Android ID + Android ID + User’s PIN

4.1.3 Android App (Camera)

The Android application serves as the primary interface for users to interact with our watermarking system. It is built using the CameraX library, which simplifies the integration of camera functionalities, making the app both user-friendly and efficient. CameraX provides a consistent and intuitive API that supports various camera operations, such as capturing high-quality images and recording videos. This library is crucial for ensuring smooth and real-time image processing, which is vital for our live watermarking system. Additionally, the application incorporates Python scripts through the Chaquopy plugin. This integration allows the complex watermarking algorithm to run seamlessly within the Android environment. By leveraging the computational power of Python for data processing, the app ensures robust and efficient watermark embedding while maintaining an easy-to-use interface for users. This combination of CameraX and Chaquopy makes our application both powerful and accessible, providing a reliable tool for real-time image watermarking.

4.1.4 Android Permissions

To function effectively, the app requires specific permissions to function correctly:

- **Camera Permission:** This permission is crucial for capturing images and recording videos directly within the app. Without access to the camera hardware, the app cannot perform its primary functions of image and video capture.

- **Storage Permission:** The app needs access to the device's storage to save the captured media. This permission ensures that images and videos can be securely stored on the device, allowing for later retrieval and processing.

- **Record Audio Permission:** This permission is necessary for capturing audio during video recording. It allows the app to access the device's microphone, ensuring that videos recorded with the app include sound.

These permissions are critical to the app's full functionality. Properly requesting and obtaining these permissions ensures the seamless operation of the watermarking system, enabling key features such as capturing, processing, and storing images and videos effectively.

4.2 Procedure

4.2.1 Image Capturing

Using the CameraX library, the app effortlessly captures high-quality photos and videos. CameraX provides a consistent and user-friendly interface for various camera functions, including taking pictures, previewing images, and recording videos. The library makes the most of the device's camera features to ensure top-notch media quality. The capturing process is designed to be swift, introducing minimal delay so that the images can be quickly processed for watermark embedding. By using CameraX, the app can adapt to different camera settings, making the capturing process smooth and efficient for real-time use, ensuring that users can take pictures easily and quickly.

4.2.2 Image Watermarking

After capturing an image, the watermarking algorithm begins its work by embedding user-specific data, such as a PIN, and device-specific information, like the Android ID, into various parts of the image. This ensures that the watermark remains secure and invisible to the naked eye, preserving the image's visual quality. The algorithm is carefully optimized for speed and efficiency, utilizing powerful Python libraries such as OpenCV, NumPy, and SciPy. These libraries, integrated through the Chaquopy plugin, enable complex image processing tasks to run smoothly within the Android app. This integration ensures that the watermarking process is not only fast but also robust, making the watermarks resilient to common manipulations such as cropping, resizing, and compression, thereby protecting the authenticity and integrity of the images.

4.2.3 Storage

Once the watermarking process is complete, the images are systematically stored on the device. This involves creating and organizing directories to ensure that the watermarked images are saved methodically. The app assigns meaningful names to the saved images, facilitating easy retrieval and management. Proper storage practices are essential for maintaining the integrity and accessibility of the watermarked images. Additionally, the storage process includes verifying that the images are saved with the correct orientation and quality settings. This ensures that the watermarked content

remains intact and is easily accessible for future use or verification. By managing storage carefully, users can reliably access their watermarked images whenever needed, ensuring both convenience and security.

4.3 System Architecture

4.3.1 Modular Architecture

The system is designed with a modular architecture, allowing different components to perform specific tasks independently. This modularity ensures that each part of the system can be developed, tested, and maintained separately, promoting flexibility and scalability. For instance, the camera operations, watermark embedding, and storage functions are handled by separate modules that work together seamlessly.

4.3.2 Architecture

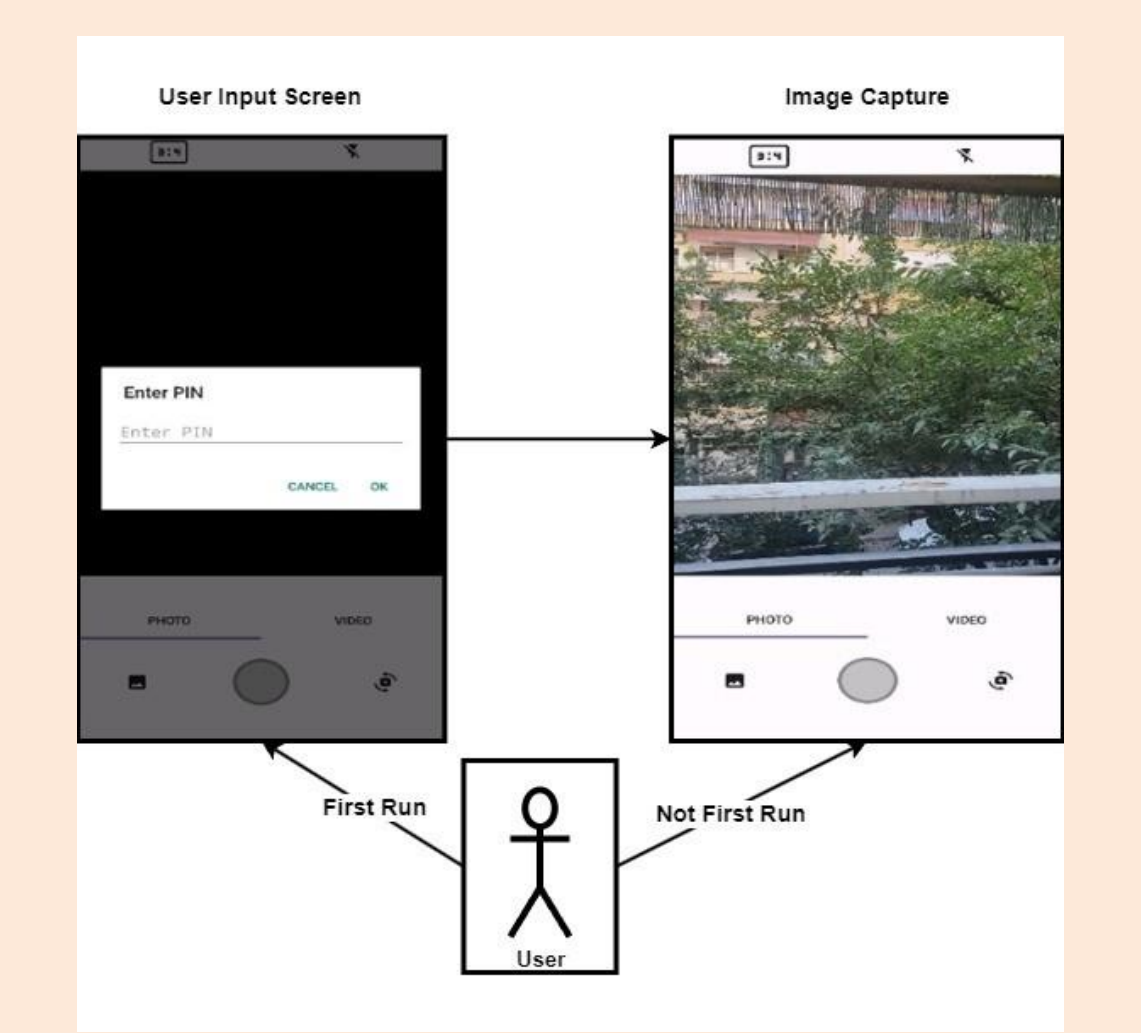

Fig. 1: Architecture of our proposed Live-Watermarking System - User Interface and Interaction

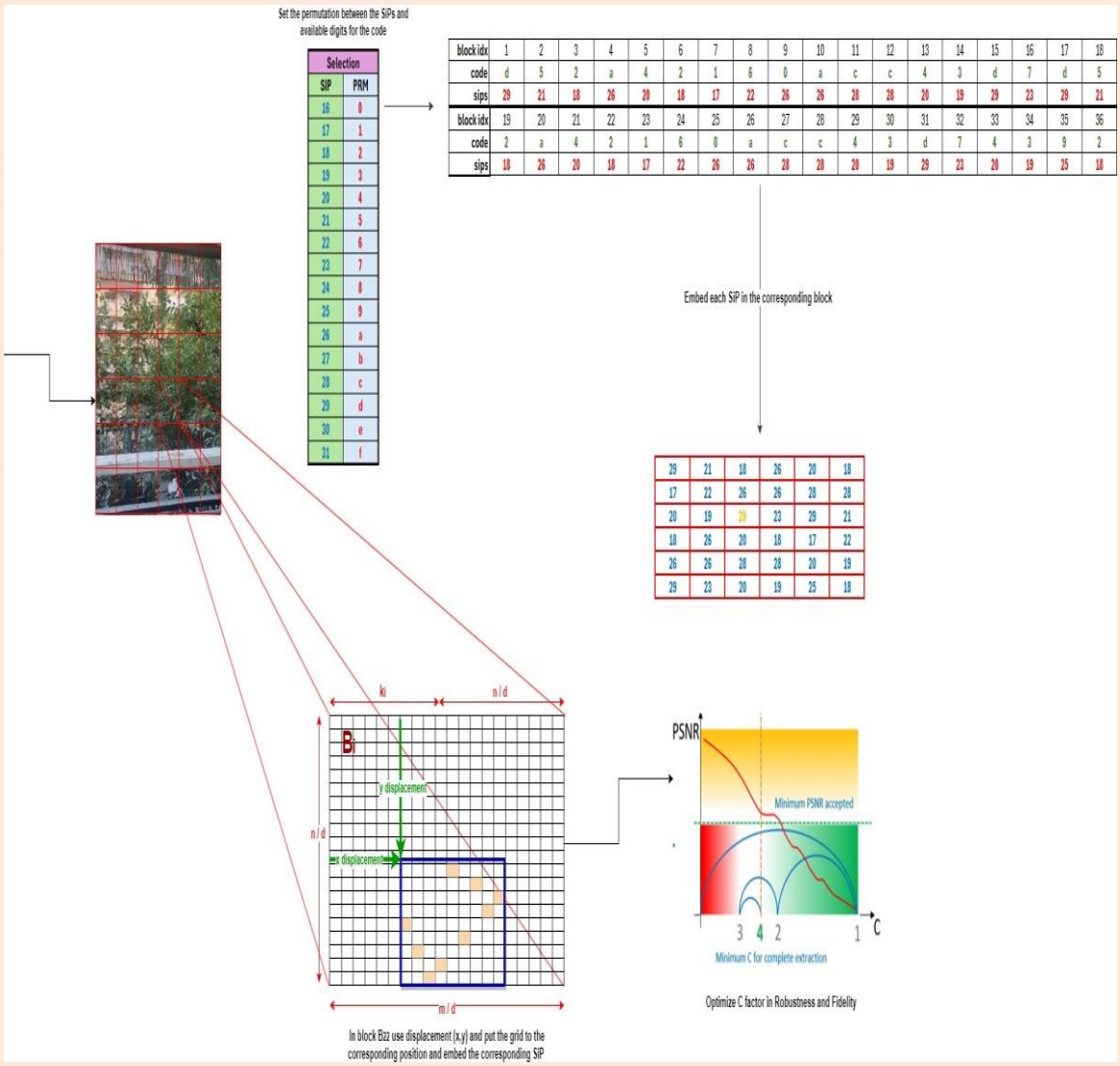

Fig. 2: Architecture of our proposed Live-Watermarking System – Image Processing

4.4 Procedure Deployment

4.4.1 Capturing Procedures

The capturing procedures are built around the CameraX library, which provides a user-friendly interface for camera interaction. CameraX is designed to work seamlessly with a wide range of Android devices, ensuring that the app performs well across various hardware configurations. This library simplifies the process of taking photos and recording videos by managing the complex aspects of camera operations behind the scenes. As a result, developers can easily integrate camera features into the app without needing to delve into intricate technical details, allowing them to focus on the app's main functionalities. CameraX also offers advanced features like image analysis and live previews, making it an excellent choice for a real-time watermarking system. The capturing process is optimized to minimize delays, enabling quick image processing and immediate watermark embedding. This ensures that users can capture and watermark their images efficiently and effortlessly, providing a straightforward interface for camera interaction and facilitating easy integration into the app.

4.4.2 Embedding Procedures

The embedding procedures are essential for integrating user and device data into the captured images. The watermarking algorithm begins by dividing the image into smaller blocks, creating specific sections where the data can be securely and imperceptibly embedded. This method ensures that the watermark is distributed throughout the image, making it difficult to remove or alter without affecting the overall image quality. The algorithm uses advanced techniques to maintain the visual integrity of the image while embedding the data. Python libraries such as OpenCV and NumPy play a crucial role in this process, providing powerful tools for handling complex image processing tasks like resizing, rotating, and transforming images. These libraries ensure that the watermarking process is efficient and robust against various forms of manipulation, preserving the integrity of the embedded data.

4.4.3 Storing Procedures

The storing procedures ensure that watermarked images are saved in the appropriate directory on the device. This process begins by verifying the existence of the designated directory; if it doesn't exist, the app creates it to ensure a specific location for saving the images. Each image is saved with a proper file name, making retrieval and management straightforward. This systematic approach to storage is crucial for maintaining the organization and accessibility of the images. Additionally, the process includes verifying that images are saved with the correct orientation and quality settings, preserving the integrity of the watermarked content. By carefully managing the storage process, users can reliably access their watermarked images, ensuring that the digital content remains secure and usable.

4.4.4 Validation Procedures

The validation procedures ensure that the watermarking process has been successful by checking that the watermarks are correctly embedded and can be extracted as needed. This involves confirming that the embedded data is intact and accurately represents the user and device-specific information. The validation process is critical for maintaining the integrity and authenticity of the watermarked images, even after they undergo various manipulations such as cropping, resizing, or compression. By performing detailed checks, validation helps ensure that the watermarking remains effective, and the images remain secure and unaltered. Regularly conducting these validation checks guarantees that the watermarking process is robust and reliable, protecting digital images from unauthorized use or modification and providing confidence in the system's ability to safeguard digital content.

Chapter 5. Evaluation

In this chapter, we provide a thorough evaluation of our real-time image watermarking system, covering various aspects including system specifications, user interface design, dataset details, quality measurements, and efficiency metrics.

5.1 System Specifications

For developing and testing our real-time image watermarking system, we used a Xiaomi Redmi Note 9S running Android 10. This device's robust performance ensured a realistic evaluation of our application. We developed the app using Android Studio version 2023.1.1, which provided all the tools needed for building and debugging. Key libraries and frameworks included CameraX for camera functionalities and Chaquopy for integrating Python scripts into the Android application. These tools were essential for enabling real-time image processing and watermarking, ensuring the application performed efficiently under various conditions. To ensure robust performance, we utilized several essential libraries:

- **CameraX:** CameraX is our primary library for handling all camera-related activities in the Android app. It simplifies tasks like taking photos, recording videos, and showing live previews, making it easier for us to integrate advanced camera features without diving into complex technical details. This library ensures that we capture high-quality images quickly, providing the raw material we need for watermarking.

- **Chaquopy:** Chaquopy helps us bring the power of Python into our Android app. By allowing Python scripts to run alongside Java/Kotlin, Chaquopy lets us use Python's extensive libraries for data processing and image manipulation. This integration is crucial for running our watermarking algorithm efficiently within the Android environment.

- **OpenCV-Python:** OpenCV-Python is a vital tool for our image processing needs. It offers a wide range of functionalities for manipulating images in real-time, such as feature detection and object recognition. We use it extensively to process captured images, perform necessary transformations, and embed watermarks, ensuring the process is both effective and preserves the image's visual quality.

- **Matplotlib:** Matplotlib is the library that manages the visual representation of our data. It helps us generate plots and graphs that illustrate metrics like image quality, PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index), and processing times. These visualizations are crucial for understanding and presenting the performance of our watermarking algorithm.

- **NumPy and SciPy:** NumPy and SciPy form the backbone of our numerical and scientific computations. NumPy handles large arrays and matrices, while SciPy offers modules for optimization and other complex calculations. We rely on these libraries for the heavy mathematical lifting required by our watermarking algorithm, ensuring precise and efficient data processing.

- **Pillow:** Pillow is our handy library for basic image manipulation. It allows us to open, resize, crop, rotate, and save images easily. We use Pillow to prepare images for more complex processing tasks, ensuring they are in the right format and orientation before embedding watermarks.

- **Scikit-Image:** Scikit-Image provides advanced image processing tools that build on SciPy's capabilities. We use it to enhance and optimize the watermark embedding process, making sure our watermarks are of high quality and resistant to tampering.

- **OpenPyXL:** OpenPyXL helps us manage data storage and reporting. It allows us to create and modify Excel documents, which is perfect for keeping track of our watermarking results and metrics. We use it to record and store information like PSNR and SSIM values, as well as processing times, in a structured and easily accessible format.

- **Cryptography:** The Cryptography library enhances the security of our watermarking system. It provides tools for encryption, digital signatures, and key management, ensuring that user data and embedded watermarks are protected from unauthorized access and tampering. This adds an extra layer of security, maintaining the integrity of our watermarked images.

5.2 User Interface

The user interface of the application is designed to be simple and user-friendly, making it easy for users to navigate and utilize the watermarking features. Key elements of the interface include:

- Capture Button: Easily accessible, this button allows users to take photos directly within the app.
- PIN Input: On the first installation, users are prompted to enter a PIN, adding a layer of security to the application.
- Processing Indicator: A loading animation is displayed during the watermarking process, informing users that the image is being processed.

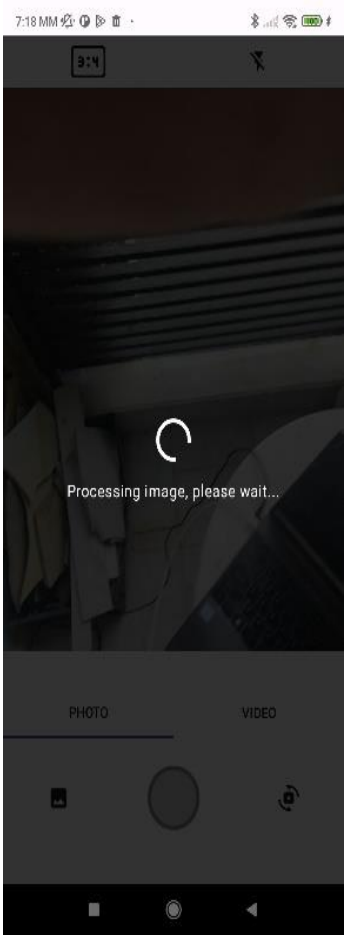

Fig. 3: Image Processing

Fig. 4: User Interface

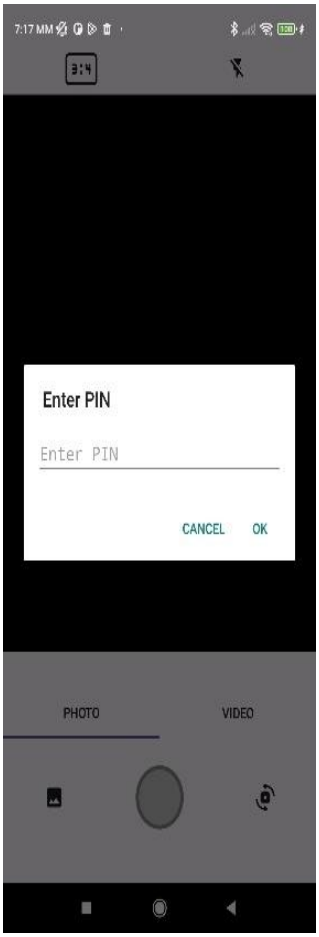

Fig. 5: PIN Input

In figure 3, the screen shows the interface while the image is being processed, providing real-time feedback to the user. In figure 4, this screen shows the main interface which is

clean and straightforward, with easy access to the capture functionality and settings. In figure 5 the PIN input screen adds an extra layer of security, ensuring that only authorized users can make changes to the watermark settings.

5.3 Dataset

To evaluate the watermarking system, we captured live images using the application at a 4K resolution. This high resolution tested the application's ability to handle large, detailed images, providing a rigorous assessment of its performance. The dataset included various lighting conditions and environments to ensure a comprehensive evaluation and demonstrate the robustness of the watermarking algorithm.

a. Image 1 **b. Image 2** **c. Image 3**

d. Image 4 **e. Image 5**

Fig. 6: General Image Dataset

5.4 Quality Measurements

We used two primary metrics to evaluate the quality of the watermarked images: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). These metrics are essential for assessing image quality, especially in image processing and compression contexts. PSNR measures the peak error between the original and watermarked images, with higher values indicating better quality. SSIM evaluates the structural similarity between the images, considering changes in luminance, contrast, and structure. The formulas through which the calculation of the above metrics is done are shown below:

$$PSNR = 10 \cdot log_{10}(\frac{255^2}{MSE}),$$

$$SSIM(x, y) = L(I_s, I_a)C(I_s, I_a)S(I_s, I_a)$$

The results of our measurements on the quality of the watermarked images obtained through our real-time watermarking application are presented in the table below:

Image	Image 1	Image 2	Image 3	Image 4	Image 5
PSNR	30.45	32.15	29.75	31.05	30.85
SSIM	0.85	0.88	0.83	0.86	0.87

Table 1: PSNR and SSIM results for General Image dataset

These metrics demonstrate that the watermarking process maintains high image quality while securely embedding the watermark, making it difficult to detect or remove without specialized tools. A large role in the values of the metrics observed in Table 1 is played both by the way in which the grid is placed within each block and by the optimization algorithm of parameter C, which tries to maximize these values, always bearing in mind that at the same time it minimizes the probability of unsuccessful watermark extraction from the image blocks. It is therefore very important the choice of the watermarking algorithm, which, as we have explained, gives quite good results in terms of the preservation of the quality of the digital image.

5.5 Efficiency Measurements

The efficiency of the watermarking process was crucial to meet real-time requirements. The processing time for embedding the watermark into each image was recorded to evaluate the system's performance. Despite the high resolution of the images and the complexity of the watermarking algorithm, the processing times ranged from 2.4 to 2.8 minutes per image. For 4K images processed on a mobile device, this performance is commendable, as it demonstrates the system's capability to handle large data efficiently. By keeping the processing time within a few minutes, our system ensures that users can capture and watermark images promptly. This efficiency is particularly beneficial for applications in fields like journalism, surveillance, and social media, where immediate image protection is necessary. Overall, the evaluation confirms that our real-time image watermarking system is both effective and practical. It maintains high image quality, embeds secure watermarks, and operates efficiently on standard Android devices, making it a robust solution for protecting digital images against unauthorized use and alterations. The results of our measurements are shown in Table 2 below:

Image	Image 1	Image 2	Image 3	Image 4	Image 5
PT	2.5	2.7	2.8	2.6	2.4
Table 2: Processing Time results for General Image dataset					

At this point we need to highlight 2 important issues. Firstly, the values observed in the above table are measured in minutes. Furthermore, the way in which we obtain these values is indirectly related to the Python **time** library. Through this library we can obtain 2 points during the execution of the watermarking algorithm. The start point which is the point at which the watermarking of the image starts and the end point at which we get the watermarked image in the corresponding path. Taking these two points, we subtract the start point from the end point and, since the result is obtained in milliseconds, we perform the necessary conversion to obtain the values shown in Table 2.

Chapter 6. Conclusion

Finally, this chapter brings together the key findings from our research on developing a real-time image watermarking system using an Android camera application. We explored how embedding user-specific data, like a PIN and Android ID, into digital images can enhance security and authenticity. Utilizing the CameraX library and integrating Python scripts with the Chaquopy plugin, we created a system capable of capturing and protecting images in real-time. Our evaluation revealed that the system effectively resists common attacks, though its performance can vary depending on the processing power of the device. We also discussed future research possibilities, such as optimizing the watermarking process, extending the technology to video and audio. This chapter underscores the valuable contributions of our work and points to exciting directions for future advancements in digital watermarking.

6.1 Concluding Remarks

In this thesis, we explored how to implement a real-time image watermarking system using an Android camera application. Our focus was to embed user-specific data, like a PIN and the device-specific Android ID, into digital images using a unique and robust watermarking algorithm. Utilizing the CameraX library and integrating Python scripts through the Chaquopy plugin, we developed a system that effectively secures and authenticates digital images captured in real-time. By including the Android ID, we enhanced the traceability and uniqueness of the watermark, linking the digital content to a specific device. Our extensive testing showed that our method is practical and effective in protecting digital images from unauthorized use and alterations. This work underscores the importance of securing digital images in today's fast-paced digital world, where the exchange of content is both rapid and widespread. By offering a solution that integrates smoothly with existing mobile technology, we have made a significant step towards enhancing the security of digital content in real-time applications.

6.2 Potentials and Limitations

While our proposed watermarking system makes great strides in real-time image protection, it does have some limitations. One of its main strengths is the ability to embed strong watermarks that can withstand common attacks like cropping and compression. However, the system's performance can vary depending on the processing power of the Android device used, which might impact the speed and efficiency of the watermarking process. Another potential drawback is the reliance on specific libraries and frameworks, such as CameraX and Chaquopy. These tools are powerful and versatile, but their ongoing support and compatibility with future Android versions are essential for our system to remain effective. Additionally, the watermarking process may introduce a slight delay in image processing, which could be an issue for applications that require ultra-low latency. Despite these challenges, our approach provides a solid foundation for further development and improvement. Optimizing the algorithm to reduce computational demands and exploring more efficient data embedding techniques could enhance performance. Expanding the system to support a broader range of devices and operating systems would also increase its applicability and robustness.

6.3 Future Research

Future research can build on the modular architecture of our proposed model, as discussed in subsection 4.3.1. This design makes it easier to extend our watermarking techniques to other forms of digital media, such as video and audio. By adapting our algorithms, we can explore the potential for real-time watermarking in video streams and audio files, further enhancing the security and authenticity of various types of digital content. Additionally, future work could focus on optimizing the watermarking process to improve efficiency and reduce the computational load on mobile devices. This might involve developing more advanced algorithms and exploring the use of machine learning techniques to dynamically adjust the watermarking parameters based on the media's characteristics. In conclusion, our research has laid the groundwork for a robust and versatile real-time image watermarking system. With continued advancements and exploration, the potential applications of this technology can be significantly expanded, offering greater protection and authenticity for a wide range of digital media. This work not only advances the field of digital watermarking but also sets the stage for more secure and trustworthy digital interactions in the future.

References

[1] HASHMI, Mohammad Farukh; SHUKLA, Ronak Jayesh; KESKAR, Avinash G. Real time copyright protection and implementation of image and video processing on android and embedded platforms. *Procedia Computer Science*, 2015, 46: 1626-1634.

[2] CHRONI, Maria, et al. Digital image watermarking using self inverting permutations-a comparative study. In: 2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA). IEEE, 2022. p. 1-8.

[3] CHRONI, Maria, et al. A repetitive watermarking scheme for digital images based on self-inverting permutations. In: 2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA). IEEE, 2022. p. 1-8.

[4] CHEN, Yueh-Hong; HUANG, Hsiang-Cheh. A copyright information embedding system for android platform. In: 2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing. IEEE, 2011. p. 21-24.

[5] HERNANDEZ, Cynthia Palma; TORRES-HUITZI, Cesar. A fragile watermarking scheme for image authentication in mobile devices. In: 2011 8th International Conference on Electrical Engineering, Computing Science and Automatic Control. IEEE, 2011. p. 1-6.

[6] ZHANG, Yingjun; CHEN, Kai. AppMark: A picture-based watermark for android apps. In: 2014 Eighth International Conference on Software Security and Reliability (SERE). IEEE, 2014. p. 58-67.

[7] CHRONI, Maria; FYLAKIS, Angelos; NIKOLOPOULOS, Stavros D. Watermarking images in the frequency domain by exploiting self-inverting permutations. In: International Conference on Web Information Systems and Technologies. SCITEPRESS, 2013. p. 45-54.

[8] AL-GAHTANI, Hanan B.; AL-DARAISEH, Ahmed A. Developing an efficient digital image watermarking for smartphones. In: 2015 2nd World Symposium on Web Applications and Networking (WSWAN). IEEE, 2015. p. 1-6.

[9] MIAO, Nai; HE, Yutao; DONG, Jane. hymnMark: towards efficient digital watermarking on android smartphones. In: Proceedings of the International Conference on Wireless Networks (ICWN). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012. p. 1.

[10] AL-OTUM, Hazem; AL-SHALABI, Nour Emad. Copyright protection of color images for android-based smartphones using watermarking with quick-response code. Multimedia Tools and Applications, 2018, 77: 15625-15655.

[11] G. Coatrieux, L. Lecornu, B. Sankur and C. Roux, "A Review of Image Watermarking Applications in Healthcare," *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, New York, NY, USA, 2006, pp. 4691-4694, doi: 10.1109/IEMBS.2006.259305.

[12] Kahlessenane, F., Khaldi, A., Kafi, R. *et al.* A robust blind medical image watermarking approach for telemedicine applications. *Cluster Comput* **24**, 2069–2082 (2021). https://doi.org/10.1007/s10586-020-03215-x

[13] Alshanbari, H.S. Medical image watermarking for ownership & tamper detection. *Multimed Tools Appl* **80**, 16549–16564 (2021). https://doi.org/10.1007/s11042-020-08814-9

[14] Khare, P, Srivastava, VK. A Secured and Robust Medical Image Watermarking Approach for Protecting Integrity of Medical Images. *Trans Emerging Tel Tech*. 2021; 32:e3918. https://doi.org/10.1002/ett.3918

[15] Narima Zermi, Amine Khaldi, Med Redouane Kafi, Fares Kahlessenane, Salah Euschi, Robust SVD-based schemes for medical image watermarking, Microprocessors and Microsystems, Volume 84, 2021, 104134, ISSN 0141-9331, https://doi.org/10.1016/j.micpro.2021.104134.

[16] Vaidya, S.P. Fingerprint-based robust medical image watermarking in hybrid transform. *Vis Comput* (2022). https://doi.org/10.1007/s00371-022-02406-4

[17] P. Singh, K. J. Devi, H. K. Thakkar and K. Kotecha, "Region-Based Hybrid Medical Image Watermarking Scheme for Robust and Secured Transmission in IoMT," in *IEEE Access*, vol. 10, pp. 8974-8993, 2022, doi: 10.1109/ACCESS.2022.3143801.

[18] K. Balasamy & D. Shamia (2023) Feature Extraction-based Medical Image Watermarking Using Fuzzy-based Median Filter, IETE Journal of Research, 69:1, 83-91, DOI: 10.1080/03772063.2021.1893231

[19] Moad, M. S., Kafi, M. R., and Khaldi, A.: A wavelet based medical image watermarking scheme for secure transmission in telemedicine applications. Microprocessors and Microsystems, 90, 104490, (2022).

[20] Vaidya, S. P., and Kishore, V. R.: Adaptive Medical Image Watermarking System For E-Health Care Applications. SN Computer Science, 3(2), 1-10, (2022).