

ABSTRACT

This work introduces an advanced approach to fortify security and communication capabilities of an Iris-Based Authentication System (IBAS) using Daugman's Algorithm. By seamlessly integrating Daugman's Algorithm, crypto-steganography, and Advanced Encryption Standard (AES) encryption, this system enhances the safeguarding of sensitive data and ensures secure information exchange. System's core features encompass user registration, secure message posting, and reliable message retrieval. During user registration, a unique iris watermark is generated and embedded within an iris image, serving as both an authentication marker and a concealed channel for encrypted message embedding. AES encryption further secures message content, adding an extra layer of protection against unauthorized access. The process of message posting involves encrypting messages with AES and embedding them within iris watermark. On recipient's side, crypto-steganography extracts concealed message, which is then decrypted using AES key. Experimental results underscore system's efficacy, validating its potential to enhance authentication and communication security. The fusion of Graphical iris-based authentication, crypto-steganography, and AES encryption creates a robust and confidential communication framework, effectively mitigating risks related to data interception and unauthorized access. This work contributes to advancement of secure information exchange within contemporary information systems.

Keywords: Graphical Iris-Based Authentication System, Crypto-steganography, Advanced Encryption Standard (AES), Security, Communication, Authentication, Data Protection, Information Exchange, User Registration, Message Posting, Message Retrieval, Watermark, Encryption, Decryption, Confidentiality, Information Security, Modern Information Systems.

CONTENTS

ABSTRACT	iv
LIST OF FIGURES	ix
LIST OF TABLES	ix
1. INTRODUCTION	1
1.1. Security System	1
1.2. What are Biometrics?	2
1.3. What are Iris Biometrics?	2
1.4. Comparison of Iris to other biometrics	3
1.5. Approaches to Iris Segmentation	3
1.6. Objectives	4
1.7. Scope	4
2. LITERATURE REVIEW	6
2.1 Literature Survey Analysis	6
3. DAUGMAN'S ALGORITHM DESCRIPTION	9
3.1 Daugman's operator	9
3.2 Gaussian filter	10
3.3 Convolution	12
4. IRIS WATERMARKING METHODOLOGY	15
4.1 Crypto-steganography	15
4.2 AES Algorithm	19
5. PROPOSED SYSTEM	24
5.1 User Registration Module	24
6. IRIS RECOGNITION	29
6.1 find_iris Function	30
6.2 daugman Function	33
7. IRIS WATRMARKING	40
8. USER VALIDATION	42
8.1 Exchange of Messages	43
9. RESULTS	44
9.1 User Sign-up Page	45
9.2 User Login page	47

9.3 Transfer of messages	48
9.4 Invalid Test cases	49
10. CONCLUSION AND FUTURE SCOPE	53
11. REFERENCES	54
12. APPENDIX	56

LIST OF FIGURES

Figure 1.1: Image of Eye	2
Figure 3.1: Graph of 1D Gaussian filter	12
Figure 4.1: Block Diagram of AES	21
Figure 5.1: Django's components	26
Figure 6.1: Flowchart of Iris Function	32
Figure 6.2: Intensity Formula	35
Figure 6.3: Flowchart of Daugman Function	36
Figure 7.1: Flowchart of Iris recognition and watermarking	41
Figure 8.1: Website Functionality	42
Figure 9.1: Command prompt showing server link	44
Figure 9.2: Home page of website	45
Figure 9.3: Signup screen	45
Figure 9.4: Uploading Eye image	46
Figure 9.5: Successful Signup process	46
Figure 9.6: User Login Screen	47
Figure 9.7: Login success	47
Figure 9.8: Post Message Screen	48
Figure 9.9: View-Message Column	48
Figure 9.10: Invalid Watermark message	49
Figure 9.11: Invalid Email Address	49
Figure 9.12: Invalid phone number	50
Figure 9.13: Invalid image	50
Figure 9.14: Duplicate Registration	51
Figure 9.15: Invalid username or password	51
Figure 9.12: Invalid image or watermark message	52

LIST OF TABLES

Table 1: Comparison of Various Biometric Technologies	7
---	---

CHAPTER-1

INTRODUCTION

1.1 Security System

A security framework alludes to a set of measures and controls planned to secure security and astuteness of data, physical resources, or people from different dangers such as unauthorized get to, robbery, harm, or impedances. Security frameworks can be classified into distinctive categories based on their reason and strategies they utilize, counting:

1. Physical Security Frameworks:

These are outlined to anticipate unauthorized physical get to to offices, gear, and assets. Cases incorporate locks, reconnaissance cameras, security watches, fencing, and get to control frameworks that oversee passage through recognizable proof and confirmation instruments.

2. Data Security Frameworks:

These ensure information and communications from capture attempts, adjustment, or pulverization. This envelops utilize of cryptographic frameworks (encryption), firewalls, anti-virus computer program, interruption location frameworks (IDS), and other shapes of cybersecurity measures.

3. Staff Security Frameworks:

These include strategies and approaches to guarantee that representatives or individuals of an organization are dependable and don't posture a security hazard. Foundation checks, security clearances, and continuous checking are common illustrations.

4. Operational Security Systems:

This includes forms and choices for taking care of and securing information resources. It incorporates arrangements for information taking care of, client get to controls, and review logs that offer assistance track how data is utilized and by whom.

Each sort of security framework employments a combination of mechanical devices, procedural methods, and physical measures, custom fitted to particular dangers and vulnerabilities of environment it is outlined to ensure. Extreme objective is to hinder potential assailants, identify

any unauthorized get to endeavors, delay and minimize affect of a breach, and react viably to security occurrences.

1.2 What are Biometrics?

Biometrics are characterized as specialized term for body calculations and estimations. It alludes to estimations related to human characteristics. Biometrics confirmation is utilized in computer science as a frame of recognizable proof and get to control. It is additionally utilized to distinguish people in bunches that are beneath reconnaissance. Biometric identifiers are particular, quantifiable characteristics utilized to name and portray people.

Most security frameworks utilize biometrics for verification. Biometrics are utilized for recognizable proof and confirmation of a individual. Based on plan, framework can be utilized for either confirmation or distinguishing proof or both. Biometrics recognize and confirm a individual based on person's physical and behavioral characteristics. Biometric framework may be a pattern-recognition framework recognizes a individual based on include vector inferred from a particular natural characteristic such as physiological biometric identifiers and behavioral identifiers. These biometrics are more invaluable than knowledge-based authentications such as passwords, Stick. There are a few biometrics out which incorporate vein design, hand geometry, unique mark, voice design, iris design, signature elements, confront acknowledgment, stride acknowledgment, DNA. These biometrics are isolated into two sorts such as physiological and behavioral. Unique mark design, confront acknowledgment, iris design, hand geometry falls beneath physiological biometrics and walk acknowledgment, voice design falls beneath behavioral biometrics.

Particularly, this report centers on application of iris acknowledgment as chosen biometric methodology.

1.3 What are Iris Biometrics?

Iris biometrics includes distinguishing proof of people based on their special iris designs, which are determined from pictures of their eyes. Outlined in Figure 1-1, human eye comprises three fundamental components:

Pupil (deepest dark parcel), iris (colored section), and sclera (white locale). Strikingly, iris and student are not concentric but or maybe show non-concentric characteristics [6].

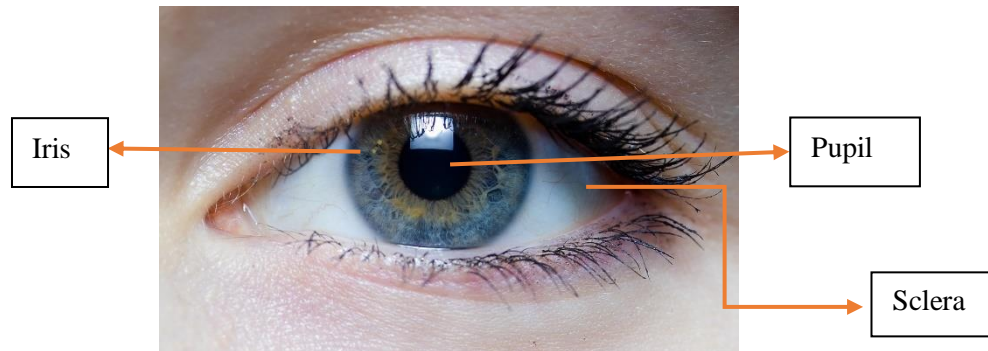


Fig. 1.1 - Picture of Eye

Pupil (deepest dark parcel), iris (colored section), and sclera (white locale). Strikingly, iris and pupil are not concentric but may show non-concentric characteristics [6]. Border between iris and pupil, also known as inward iris border, contains a variable sweep due to changes in pupil size incited by shifting light levels. Each individual has a unmistakable iris design [6]. This design can be extricated from eye pictures and interpreted into a code. This code is at that point comparable with codes determined from other eye pictures, counting those from same person. Result of this comparison evaluates difference between compared codes, subsequently empowering assurance of whether compared iris designs begin from same or distinctive eyes.

It refers to measurements related to human characteristics. Biometrics authentication is used in computer science as a form of identification and access control. It is also used to identify individuals in groups that are under surveillance. Biometric identifiers are distinctive, measurable characteristics used to label and describe individuals.

1.4 Comparison of Iris to other biometrics

In setting of biometrics, iris acknowledgment holds a particular advantage over other biometric strategies due to its broad design differing qualities [7]. Eminently, concurring to John Daugman, the conditional wrong dismissal likelihood for iris designs stands at an amazing 109.6, likening to one in 4 billion [14]. The complexity of iris checks permits investigation of over 200 special iris properties such as wrinkles, crown, rings, and spots [8]. Not at all like biometrics like voice or facial highlights that can experience changes over time, solidness of iris design remains steady all through an individual's lifetime [9]. Indeed outside variables such as contact focal points, glasses, or eye surgeries have no affect on iris characteristics [10].

1.5 Approaches to Iris Segmentation:

To attain successful iris acknowledgment, exact iris division is an fundamental prerequisite [11, 12]. Two broadly recognized strategies for iris division are Wilde's and Daugman's methods. Wilde's approach includes a two-step handle: to begin with, a double edge outline is produced utilizing intensity gradients from iris picture, and after that Hough change is connected to recognize iris's inward and external borders [13]. Daugman's calculation, named after Teacher John Daugman, utilizes an integrodifferential administrator to find circular understudy and limbic borders of iris inside an eye picture. Working as a circular edge finder, this calculation decides parameters of circular border by maximizing alter in circumferential pixel escalated values along circular edge, compared to a marginally more extensive circular locale centered on same pixel. This proposal utilizes Daugman's calculation for iris division.

1.6 Objectives:

Essential goals of this extend are as takes after:

Iris Authentication: Executing an Iris Confirmation framework to improve client confirmation and get to control. Iris designs, being one of a kind to each person, give a solid biometric identifier for client verification.

Steganography Integration: Joining cryptographic steganography procedures to insert scrambled messages inside pictures. This makes a difference in guaranteeing incognito communication whereas keeping up appearance of conventional pictures.

AES Encryption: Utilizing Progressed Encryption Standard (AES) encryption for secure message transmission. AES could be a broadly recognized encryption calculation that ensures secrecy and information judgment.

User-friendly Interface: Creating a user-friendly web interface for simple interaction with framework. This interface encourages client enrollment, login, and message trade.

Message Logging: Making a framework that logs scrambled messages, their senders, collectors, and timestamps. This include helps in keeping up a record of communications for review purposes.

1.7 Scope:

Scope of this project encompasses various areas of secure communication:

Client Enrollment and Confirmation: Clients can safely enlist utilizing their qualifications and an iris picture. Iris verification guarantees as it were authorized clients can get to framework.

Message Encryption and Unscrambling: The framework empowers clients to send scrambled messages. These messages are scrambled utilizing AES and can as it were be unscrambled by aiming beneficiary.

Steganographic Message Implanting: Messages are inserted into pictures utilizing cryptographic steganography, improving incognito communication and security.

Message Logging and Recovery: The framework records sent and gotten messages, giving clients with a history of their communications. Messages can be decoded and seen by aiming beneficiaries.

User-friendly Interface: The extend offers an intuitive web interface that encourages client enlistment, login, message composition, and seeing

Security Measures: The extend joins solid encryption methods, iris verification, and steganography to guarantee information protection, client verification, and message privacy.

CHAPTER- 2

LITERATURE REVIEW

2.1 Literature Survey Analysis

In this segment, I have displayed an broad study of important papers and distributions fundamental for our inquire about.

My writing audit investigates integration of cryptographic strategies, especially Progressed Encryption Standard (AES) and crypto-steganography, with iris-based biometric confirmation frameworks. The center is on upgrading security and unwavering quality of communication frameworks by inserting scrambled messages inside iris watermarks. This novel approach not as it were secures message substance against unauthorized get to but moreover employments biometric information as a vigorous layer of security.

In the Within the paper titled “Iris Biometric Cryptography for Identity Document” [1], utilization of iris acknowledgment innovation to improve security in verification forms. By creating special cryptographic keys from iris templates, the framework points to supply a secure encryption and decoding instrument. Utilize of Progressed Encryption Standard (AES) calculation guarantees strong security for character information. Separate measurements like hamming remove and Euclidean remove are utilized for format coordinating distinguishing proof, coming about in moo wrong dismissal or wrong acknowledgment rates. Investigate highlights significance of biometric innovations in progressing security, with iris acknowledgment being a special and exact biometric identifier. Integration of biometric cryptography utilizing iris highlights and AES calculation exhibits a promising approach for secure confirmation frameworks.

Within the paper titled “A Framework for Iris Biometrics Protection: A Marriage Between Watermarking and Visual Cryptography” [3], a system for ensuring iris biometrics through a combination of watermarking and visual cryptography. Proposed plot comprises of two layers: a strong watermarking calculation to ensure judgment of biometric picture and visual cryptography to secure iris layout. The watermarking calculation inserts individual data in iris picture by arbitrarily intercity DCT coefficients, whereas visual cryptography separates layout into shares for capacity

Within the paper titled “Iris Biometric Security using Watermarking and Visual Cryptography” [2], The paper presents a novel security engineering for iris assurance utilizing watermarking and visual cryptography procedures, implemented in MATLAB, with three stages: watermarking calculation, layout era, and visual cryptography, pointing to upgrade biometric security.

Within the paper titled “Combining crypto with biometrics effectively” [4], The paper proposes a secure and viable strategy to coordinated iris biometrics into cryptographic applications by producing a biometric key from iris pictures with error-correction information put away in a tamper-resistant token, permitting for creation of a 140-bit biometric key suitable for encryption. The plot points to combine biometrics, cryptography, and tamper-resistance to supply tall security whereas addressing privacy concerns and guaranteeing key disavowal.

Within the paper titled “How Iris Recognition Works [5], The technique includes creating calculations for iris acknowledgment based on measurable freedom on iris stage structure encoded by wavelets and testing them in field and research facility trials with millions of comparison tests.

TABLE I

Comparison of Various Biometric Technologies Based on Perception of Authors. High, Medium, and Low are Denoted by H, M, and L, respectively

Biometric identifier	Universality	Distinctiveness	Permanence	Collectability	Performance	Acceptability	Circumvention
Face	H	L	M	H	L	H	H
Fingerprint	M	H	H	M	H	M	M
Hand geometry	M	M	M	H	M	M	M
Iris	H	H	H	M	H	L	L
Keystroke	L	L	L	M	L	M	M

Signature	L	L	L	H	L	H	H
Voice	M	L	L	M	L	H	H

Subariah Ibrahim talked about around distinctive biometrics in his paper “Iris Biometric Cryptography for Personality Document” [1], as Comparison of a few of biometric identifiers based on seven components is given in Table 1. Properties of biometric identifiers are; All-inclusiveness, uniqueness, changelessness and collectability. Execution (speed and exactness), worthiness (readiness of individuals to utilize), and circumvention (idiot proof) are properties of biometric frameworks [3]. Based on comparison of Table 1, we taken note that utilizing iris as biometric identifier would be more viable, solid and precise for confirmation prepare accessible nowadays when compared to others. Each eye is interesting and remains steady from birth to conclusion of life.

In outline, overview investigates potential of combining biometric information, especially iris acknowledgment, with advanced cryptographic strategies to make profoundly secure and productive verification frameworks. The surveyed papers collectively highlight headways in biometric encryption, vigor of iris acknowledgment methods, and inventive approaches to secure biometric information utilizing crypto-steganography. These discoveries fortify viability and security enhancements described in center framework beneath survey, which combines AES encryption, Daugman's Calculation, and crypto-steganography for an moved forward Iris-Based Confirmation Framework (IBAS). This blend of advances gives a promising course for progression of secure data trade in modern data frameworks, viably tending to dangers related to information interferences and unauthorized get to.

CHAPTER- 3

DAUGMAN'S ALGORITHM DESCRIPTION

Daugman's calculation may be a broadly recognized strategy for iris acknowledgment, created by John Daugman in 1990s. It plays a significant part within the field of biometric confirmation, which employments special physiological characteristics for recognizing people. This calculation has ended up a standard for iris acknowledgment frameworks and is celebrated for its strength and exactness.

3.1 Daugman's operator:

The center of Daugman's calculation rotates around utilize of a numerical concept known as 2D Gabor wavelet change. This strategy is viable in capturing both spatial (location-specific) and recurrence (detail and design) data from complex visual surface of an individual's iris. The transform produces a phase-based format speaking to iris's one of a kind designs, which incorporate rings, wrinkles, and spots.

The key to this strategy lies in utilize of integrodifferential administrator to distinguish circular boundaries based on varieties in pixel concentrated, which ordinarily manifest as changes in softness along these boundaries. Central mathematical tool in Daugman's approach to identifying these circular edges is integrodifferential equation, defined as follows:

$$\max_{(r, x_0, y_0)} \left| G_{\sigma}(r) * \frac{\partial}{\partial r} \oint_{r, x_0, y_0} \frac{I(x, y)}{2\pi r} ds \right|$$

where: $I(x, y)$ is intensity of pixel at coordinates (x, y) in image of an iris.

r denotes radius of various circular regions with center coordinates at (x_0, y_0) .

σ is standard deviation of Gaussian distribution.

$G_{\sigma}(r)$ denotes a Gaussian filter of scale sigma (σ).

(x_0, y_0) is assumed center coordinates of iris.

s is contour of circle given by parameters (r, x0, y0).

This condition looks for to maximize form fundamentally of the slope of the picture concentrated around a circular way, viably finding the circle (either the iris or the understudy) where there's the most prominent alter in escalated. The administrator alters the parameters of the circle – particularly, the center facilitates and the sweep – to find the most excellent fit that typifies the boundary of the iris or understudy.

By applying this strategy, Daugman's calculation proficiently depicts the circular districts, pivotal for consequent steps in iris acknowledgment frameworks, counting normalization and design encoding. This exactness in recognizing the circular borders of the iris and student is what empowers the tall precision of iris-based biometric distinguishing proof systems.

3.2 Gaussian filter:

A Gaussian channel may be a sort of picture channel utilized to decrease commotion and detail in digital images. Named after the mathematician Carl Friedrich Gauss, it leverages the properties of the Gaussian function to realize smoothing. The Gaussian channel is especially favored in picture preparing due to its properties within the recurrence space and its normal reaction within the spatial space.

3.2.1 Basic Concept

The Gaussian channel applies a bit where the values are calculated utilizing the Gaussian work. The work in one measurement is given by:

In 1D:

$$g_{\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

where σ is the standard deviation, controlling the width of the bell-shaped curve of the Gaussian distribution. A higher σ value results in more blurring as the kernel becomes wider.

For a 2D image, the Gaussian function becomes:

In 2D:

$$G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

A bit is produced based on this work, and it is convolved with the picture. Each pixel esteem is supplanted by a weighted whole of its neighbors, where the weights are decided by the Gaussian work. This comes about in a smoothing impact that damps higher frequencies, in this way lessening commotion and little points of interest.

3.2.2 Illustration of Gaussian Filtering

Assume we have a basic 3x3 Gaussian part to apply to a picture. Expect ($\sigma = 1.0$). The part might see something like this after normalization:

$$\begin{bmatrix} 0.075 & 0.123 & 0.075 \\ 0.123 & 0.204 & 0.123 \\ 0.075 & 0.123 & 0.075 \end{bmatrix}$$

Here's how this filter might be applied to a simple 3x3 region of an image with pixel values:

$$\begin{bmatrix} 200 & 202 & 205 \\ 198 & 200 & 203 \\ 204 & 206 & 208 \end{bmatrix}$$

The new value of center pixel after applying Gaussian filter will be:

$$(0.075 * 200) + (0.123 * 202) + (0.075 * 205) + (0.123 * 198) + (0.204 * 200) + (0.123 * 203) + (0.075 * 204) + (0.123 * 206) + (0.075 * 208) \approx 201.93$$

3.2.3 Sorts of Gaussian Channels

1. Standard Gaussian Channel: This can be the fundamental type where the part measure and σ are chosen based on the specified sum of smoothing.

2. Adaptive Gaussian Channel: The esteem of σ can be shifted over the picture, depending on the neighborhood fluctuation of pixel values. This permits for versatile smoothing where more commotion is show.

3. Multi-scale Gaussian Channel: This includes applying Gaussian channels with distinctive values of σ to an picture and combining the comes about in different ways. This method is often used in scale-space representation.

The Gaussian channel in our case is planned in Python and could be a 1 by 5 (lines by columns) vector with concentrated values given by vector $A = [0.0003 \ 0.1065 \ 0.7866 \ 0.1065 \ 0.0003]$. The outlined channel is appeared in figure 3.1.

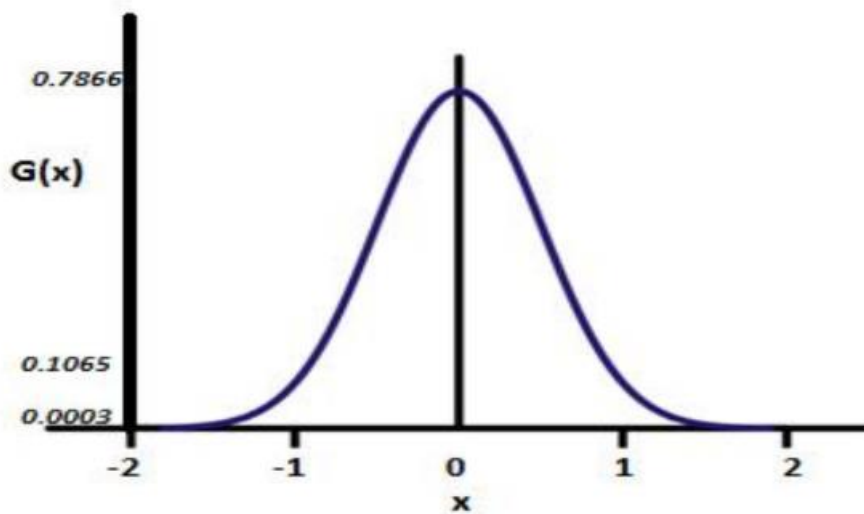


Fig. 3.1 – Graph of 1D Gaussian filter

In rundown, the Gaussian channel may be a foundational apparatus in picture processing, offering the capacity to successfully smooth pictures and diminish clamor without noteworthy artifacts, making it vital for both fundamental and progressed picture examination assignments.

3.3 Convolution:

Convolution could be a principal numerical operation broadly utilized in different areas such as flag preparing, picture preparing, and information investigation. It is an operation on two capacities that produces a third work, communicating how the shape of one is adjusted by the other. In less difficult terms, convolution makes a difference in understanding how each portion of one work influences each portion of another when the two connected.

3.3.1 Essential Guideline::

In its easiest shape, convolution includes two capacities, say f and g , which are frequently

alluded to as the input and the part, separately. The result of convolving these two capacities could be a modern work, indicated as $f * g$.

Mathematically, the convolution is defined a

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

This integral compute the range beneath the item of the two capacities as one work is turned around and moved over the other. The variable τ speaks to an integration variable, and t speaks to the move.

3.3.2 Discrete Convolution

In digital applications such as picture handling, we regularly bargain with discrete signals. In this manner, the convolution is characterized for discrete cases as:

$$(f * g)[n] = \sum_{k=-\infty}^{\infty} f[k]g[n - k]$$

In this condition, $f[k]$ and $g[k]$ are discrete capacities, and n is the list of the coming about work after applying convolution. For viable usage, particularly in computing situations, the entireties and arrangements are limited.

3.3.3 Application in Picture Handling

In picture handling, convolution plays a pivotal part in operations such as obscuring, honing, edge location, and more. An picture can be thought of as a lattice of pixel values, and the bit (or channel) is another, littler lattice. Convolution includes sliding the bit over the picture, frequently pixel by pixel, and computing the entirety of the items of the covering passages at each position.

For occasion, consider a straightforward 3x3 part connected over an picture. At each position, the kernel's center overlays a particular pixel, and the convolution operation includes increasing the kernel's values by the comparing basic picture pixel values, at that point summing these to create a single pixel within the yield picture.

3.4 Steps Involved in Daugman's algorithm

The method of utilizing Daugman's calculation for iris acknowledgment by and large takes after a few key steps:

1. **Picture Capture:** An introductory photo of the eye is taken, ordinarily utilizing near-infrared light to decrease reflection from the cornea and to clearly capture the perplexing structures of the iris.

2. **Iris Localization:** The calculation distinguishes the circular locale of the eye where the iris is found, recognizing it from the understudy and the sclera (the white portion of the eye). This includes recognizing the boundaries of the iris (both the inward boundary close the understudy and the external boundary close the sclera) and regularly rectifying for the circular distortion due to the pupil's development and withdrawal.

3. **Normalization:** To account for variations within the imaging viewpoint, estimate, and student widening, the iris locale is normalized into a fixed-size rectangular square of information. This handle, regularly alluded to as rubber-sheet show normalization, remaps each point inside the iris locale to a match of polar facilitates.

4. **Include Encoding:** Using the 2D Gabor wavelet transform, Daugman's calculation encodes the iris design in a compact bit stream shape, which captures the special textural designs of the iris in a parallel IrisCode. This IrisCode is what will be utilized for coordinating and confirmation.

5. **Coordinating:** To decide in case two iris pictures (IrisCodes) have a place to same individual, Hamming remove — degree of similitude between two double strings — is calculated. A lower Hamming remove demonstrates the next degree of closeness. Ordinarily, a limit is set to choose whether a coordinate is fruitful, adjusting the rates of untrue acknowledgments and wrong dismissals.

Daugman's calculation has illustrated tall levels of precision in recognizing people, with moo untrue coordinate and non-match rates. It has been broadly utilized in different applications, extending from airplane terminal security frameworks to keeping money and healthcare get to controls. Its adequacy and unwavering quality proceed to create it a foundation innovation in iris acknowledgment and broader biometric security areas.

CHAPTER – 4

IRIS WATERMARKING METHODOLOGY

4.1 Crypto-steganography:

Crypto-steganography, an interesting combination of cryptography and steganography, includes concealing mystery messages inside apparently harmless carrier records utilizing cryptographic methods.

4.1.1 Cryptography:

Cryptography is the science and craftsmanship of securing communication and information by changing over it into a non-readable arrange, commonly known as ciphertext, utilizing scientific calculations and keys. It guarantees privacy, astuteness, and genuineness of data.

Here are a few key components and strategies of cryptography:

1. Encryption: Encryption involves method of changing over plaintext (unique information) into ciphertext utilizing an encryption calculation and a mystery key. There are two fundamental sorts of Encryption:

- Symmetric Encryption: In symmetric encryption, the same key is utilized for both encryption and unscrambling. Calculations like AES (Progressed Encryption Standard) and DES (Information Encryption Standard) are broadly utilized symmetric encryption calculations.

- Asymmetric Encryption: Asymmetric encryption moreover known as public-key encryption, utilizes a combine of keys: a open key for encryption and a private key for decoding. RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Bend Cryptography) are illustrations of topsy-turvy encryption calculations.

2. Hashing: Hashing may be a one-way cryptographic work that changes over input information (plaintext) into a fixed-size string of characters, known as a hash esteem or process. The key characteristics of hashing are:

- It's deterministic: The same input continuously produces the same hash esteem.
- It's irreversible: It's for all intents and purposes impossible to invert the method to get the

initial input.

- It's collision-resistant: It's computationally infeasible to discover two distinctive inputs that deliver same hash esteem.

3. Advanced Marks: Computerized marks give a implies of confirming the genuineness and judgment of computerized messages or records. They include utilize of deviated encryption to sign and confirm messages. The method ordinarily includes:

- Producing a advanced signature utilizing sender's private key.
- Confirming the signature utilizing sender's open key.

4. Key Management: Key administration may be a basic viewpoint of cryptography that includes safely generating, storing, dispersing, and repudiating cryptographic keys.

4.1.2 Steganography:

Steganography is the honeof concealing mystery data inside other non-secret information, known as the carrier, in such a way that the presence of the covered up message isn't clear to spectators. Not at all like cryptography, which centers on keeping the substance of a message mystery, steganography points to stow away the truth that communication is taking put. Here are a few key focuses around steganography:

1. Carrier Mediums: Steganography can cover up information inside different sorts of carrier mediums, counting:

- Pictures: Concealing information inside the pixels of computerized pictures by unpretentiously modifying their color values.
- Sound: Implanting information inside the recurrence or sufficiency of sound signals.
- Video: Stowing away data inside the outlines of computerized recordings by altering pixel values or outline groupings.
- Content: Concealing messages inside apparently harmless content by utilizing strategies like whitespace control or adjusting particular characters.

2. Procedures: Steganographic procedures shift depending on the chosen carrier medium. A few common procedures incorporate:

- LSB (Least Significant Bit) Replacement: Adjusting the slightest noteworthy bits of advanced information (such as picture pixels) to insert the mystery message without altogether changing the carrier.

- Spread Spectrum: Conveying the mystery information over numerous carrier components to play down the affect on any single component.

- Advanced Watermarking: Inserting intangible designs or data inside advanced media to demonstrate possession or realness.

3. Discovery: Identifying steganographically covered up data can be challenging, particularly in the event that the changes made to the carrier are unpretentious and troublesome to recognize from irregular commotion. Specialized devices and calculations are frequently utilized to identify the nearness of covered up information in carrier records.

In outline, cryptography and steganography are two complementary methods utilized for securing and concealing data, individually. Whereas cryptography centers on scrambling information to keep it mystery, steganography points to cover up the fact that communication is happening within the to begin with put. When utilized together, these methods give upgraded security and protection for advanced communication and information capacity.

4.1.3 Implementation in Python:

I will demonstrate a simple example of crypto-steganography using Python with the following steps:

1. Choose Carrier File: Select a carrier file (such as an image) to embed the secret message.
2. Encrypt the Message: Encrypt the secret message using a cryptographic algorithm like AES (Advanced Encryption Standard).
3. Embed Encrypted Message: Embed the encrypted message into the carrier file using steganographic techniques. In this example, we'll use LSB (Least Significant Bit) technique to modify the least significant bits of pixels in the image.
4. Extraction: Extract the hidden message from the carrier file.

Example Code:

```
from PIL import Image

from Crypto.Cipher import AES

from Crypto.Random import get_random_bytes

import numpy as np
```

Step 1: Load carrier image and secret message

```
carrier_image = Image.open("carrier_image.png")

secret_message = "This is a secret message."
```

Step 2: Encrypt the message

```
key = get_random_bytes(16) # 128-bit key for AES encryption

cipher = AES.new(key, AES.MODE_ECB)

encrypted_message = cipher.encrypt(secret_message)
```

Step 3: Embed encrypted message into image

```
def embed_message(image, message):

    img_array = np.array(image)

    message_len = len(message)

    index = 0

    for i in range(len(img_array)):

        for j in range(len(img_array[0])):

            if index < message_len:

                pixel = img_array[i][j]

                img_array[i][j] = tuple(pixel[:-1]) + (message[index],)
```

```

        index += 1

    else:

        break

    return Image.fromarray(img_array)

stego_image = embed_message(carrier_image, encrypted_message)

stego_image.save("stego_image.png")

```

Step 4: Extract the hidden message

```

def extract_message(image, key):

    img_array = np.array(image)

    extracted_message = ""

    cipher = AES.new(key, AES.MODE_ECB)

    for i in range(len(img_array)):

        for j in range(len(img_array[0])):

            pixel = img_array[i][j]

            extracted_message += chr(pixel[-1])

    decrypted_message = cipher.decrypt(extracted_message)

    return decrypted_message.decode()

extracted_message = extract_message(stego_image, key)

print("Extracted Message:", extracted_message)

```

4.2 AES Algorithm:

AES (Advanced Encryption Standard) may be a broadly utilized encryption calculation that makes a difference secure delicate data by changing it into an garbled arrange. It's like putting your message into a secure box some time recently sending it over. Here's how it works:

1. Key Generation:

- To begin with, a secret key is produced. This key is just like the key to the secure box. As it were those who have this key can bolt and open the box.

2. Encryption:

- To scramble a message, AES isolates it into squares and applies a arrangement of complex numerical operations. Think of it like rearranging the letters of your message agreeing to a particular design.

- Each piece of the message is combined with the mystery key, creating a special mixed block. This prepare is rehashed for each square until the whole message is changed.

3. Decryption:

- To unscramble the scrambled message and recover the initial data, the beneficiary employments the same mystery key. It's like having the key to open the secure box.

- AES applies a switch process to the scrambled squares utilizing the mystery key. This prepare unscrambles the pieces back to their unique shape.

4. Security:

- AES is considered secure since it's safe to different assaults. Its quality lies in its capacity to deliver scrambled information that shows up arbitrary and garbled without the right key.

- The security of AES depends intensely on the mystery of the key. As long as the key remains private, the scrambled information remains secure from unauthorized get to.

5. Applications:

- AES is broadly utilized in different applications where information security is fundamental, such as securing communications over the web, securing put away information on gadgets, and shielding touchy data in databases.

- It's too utilized in conventions like HTTPS, VPNs, and secure informing apps to guarantee protection and privacy.

In substance, AES acts as a secure lockbox, changing plain text into garbled ciphertext utilizing a mystery key. As it were those with the key can open and get to the first data, guaranteeing the

privacy and judgment of delicate information.

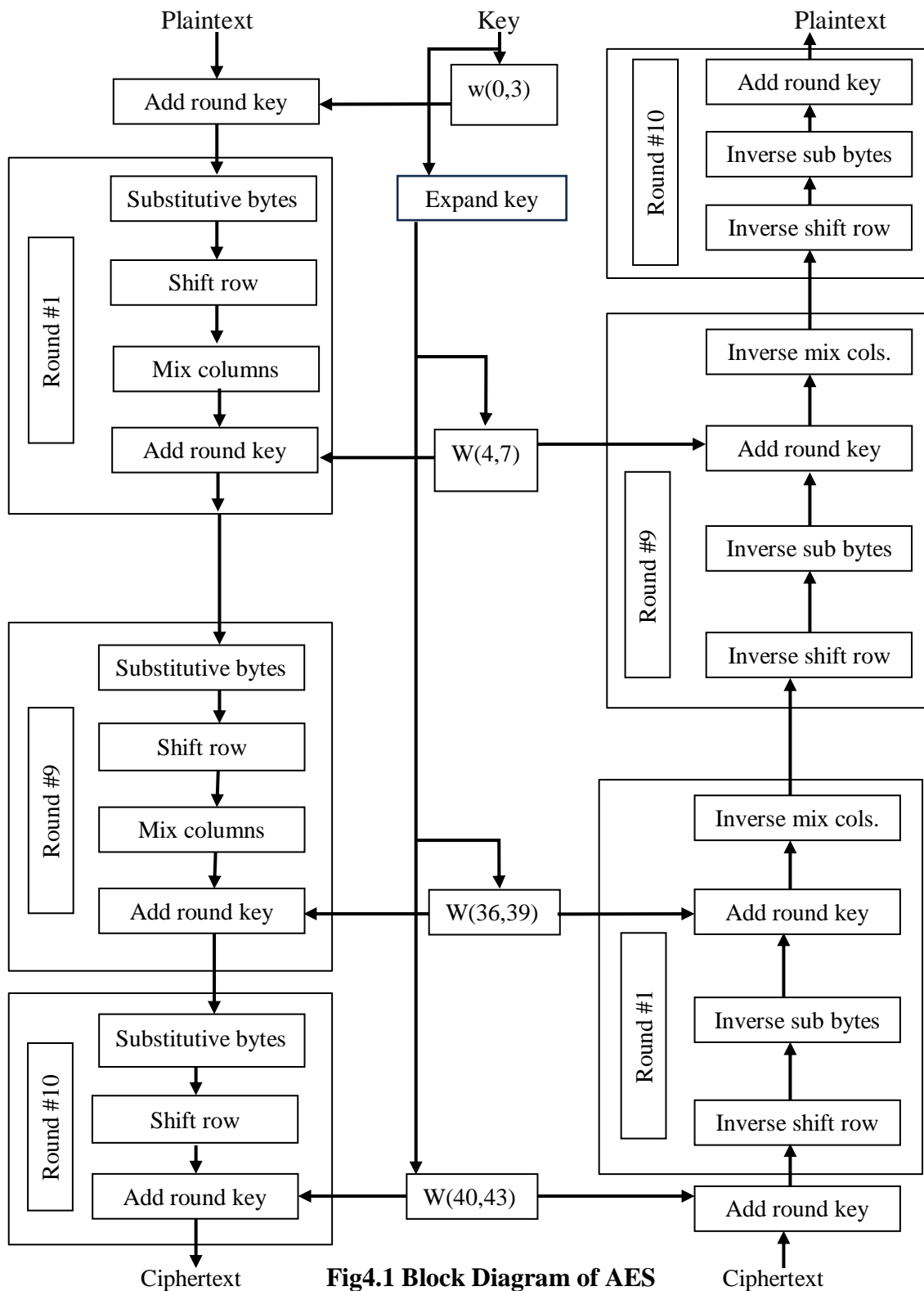


Fig4.1 Block Diagram of AES

4.2.1 Implementation in Python:

```
import pyaes, pbkdf2, binascii, os, secrets
```

```

import base64

def getKey(): #generating key with PBKDF2 for AES

    password = "s3cr3t*c0d3"

    passwordSalt = '76895'

    key = pbkdf2.PBKDF2(password, passwordSalt).read(32)

    return key

def encrypt(plaintext): #AES data encryption

    aes = pyaes.AESModeOfOperationCTR(getKey(),
pyaes.Counter(3112954703500004730295243396765419539812423984456632288417216363
7846056248223))

    ciphertext = aes.encrypt(plaintext)

    return ciphertext

def decrypt(enc): #AES data decryption

    aes = pyaes.AESModeOfOperationCTR(getKey(),
pyaes.Counter(3112954703500004730295243396765419539812423984456632288417216363
7846056248223))

    decrypted = aes.decrypt(enc)

    return decrypted

```

Explanation:

1. getKey Function:

- This function generates a key for AES encryption using the PBKDF2 (Password-Based Key Derivation Function 2) algorithm.
- It takes a password and a password salt as input parameters.
- PBKDF2 iteratively applies a cryptographic hash function (such as SHA-256) to the password along with the salt, producing a derived key.

- The function then returns the derived key, truncated to 32 bytes (256 bits), which is the appropriate key size for AES-256 encryption.

2. encrypt Function:

- This function performs AES encryption on plaintext data.
- It uses the AES CTR (Counter) mode of operation for encryption.
- The `getKey` function is called to obtain the encryption key.
- AES CTR mode encrypts plaintext by XORing it with a keystream generated from the encryption key and a counter.
- The AESModeOfOperationCTR object is initialized with the encryption key obtained from `getKey` and a counter value.
- The plaintext is then encrypted using the AESModeOfOperationCTR object, producing the ciphertext.
- The ciphertext is returned by the function.

3. decrypt Function:

- This function performs AES decryption on ciphertext data.
- Similar to encryption, it uses the AES CTR mode of operation.
- The `getKey` function is called to obtain the decryption key.
- An AESModeOfOperationCTR object is initialized with the decryption key obtained from `getKey` and the same counter value used for encryption.
- The ciphertext is decrypted using the AESModeOfOperationCTR object, producing the original plaintext.
- The decrypted plaintext is returned by the function.

CHAPTER – 5

PROPOSED SYSTEM

5.1 Client Enlistment MODULE:

We have made a web application utilizing Django system and python. The client to begin with have to be Sign-up where they give their individual information such as Username, Watchword, contact number, E-mail ID, address, at that point Transfer Eye picture additionally compose the watermark message. All these information will be stored within the Database.

5.1.1 Creating Web application:

Django may be a high-level web system composed in Python that permits designers to rapidly construct web applications by giving a set of apparatuses, libraries, and traditions. It takes after the Model-View-Template (MVT) architecture, which is comparable to the Model-View-Controller (MVC) design.

Here's an clarification of Django's components and how they work together, in conjunction with a basic case:

5.1.2 Model:

- The Model component speaks to the information structure of the application. It characterizes the database construction and typifies the data access rationale.

- Django gives an Object-Relational Mapping (ORM) framework that permits engineers to characterize models as Python classes. Each demonstrate lesson speaks to a database table, and its qualities speak to areas within the table.

- Django's ORM abstracts absent the complexities of database interaction, permitting designers to connected with the database utilizing high-level Python objects and questions.

- Models are regularly characterized in Python classes that subclass 'django.db.models.Model'.

- **Case:**

```
from django.db import models
```

```
class Post(models.Model):
```

```
title = models.CharField(max_length=100)
```

```
content = models.TextField()
```

```
date_posted = models.DateTimeField(auto_now_add=True)
```

5.1.3. View:

- The View component handles the business rationale and client interaction rationale of the application. It gets input from the client, forms it, and returns an fitting reaction.

- In Django, views are executed as Python capacities or classes. They get HTTP demands from the client, perform vital operations (such as questioning the database or processing form data), and produce HTTP reactions to send back to the client.

- Views in Django are capable for rendering HTML layouts, creating JSON reactions for AJAX demands, or diverting clients to other URLs.

- **Case:**

```
from django.shortcuts import render
```

```
from .models import Post
```

```
def post_list(request):
```

```
    posts = Post.objects.all()
```

```
    return render(request, 'blog/post_list.html', {'posts': posts})
```

5.1.4. Template:

- Templates represent the presentation layer of the application. They define the structure and layout of the user interface.

- Templates are HTML files with embedded Django template language syntax.

- Templates are rendered with context data passed from views.

- **Example (post_list.html):**

```
<!DOCTYPE html>
```

```

<html>

<head>

    <title>Post List</title>

</head>

<body>

    <h1>Posts</h1>

    {% for post in posts %}

        <h2>{{ post.title }}</h2>

        <p>{{ post.content }}</p>

        <p>Posted on: {{ post.date_posted }}</p>

    {% endfor %}

</body>

</html>

```

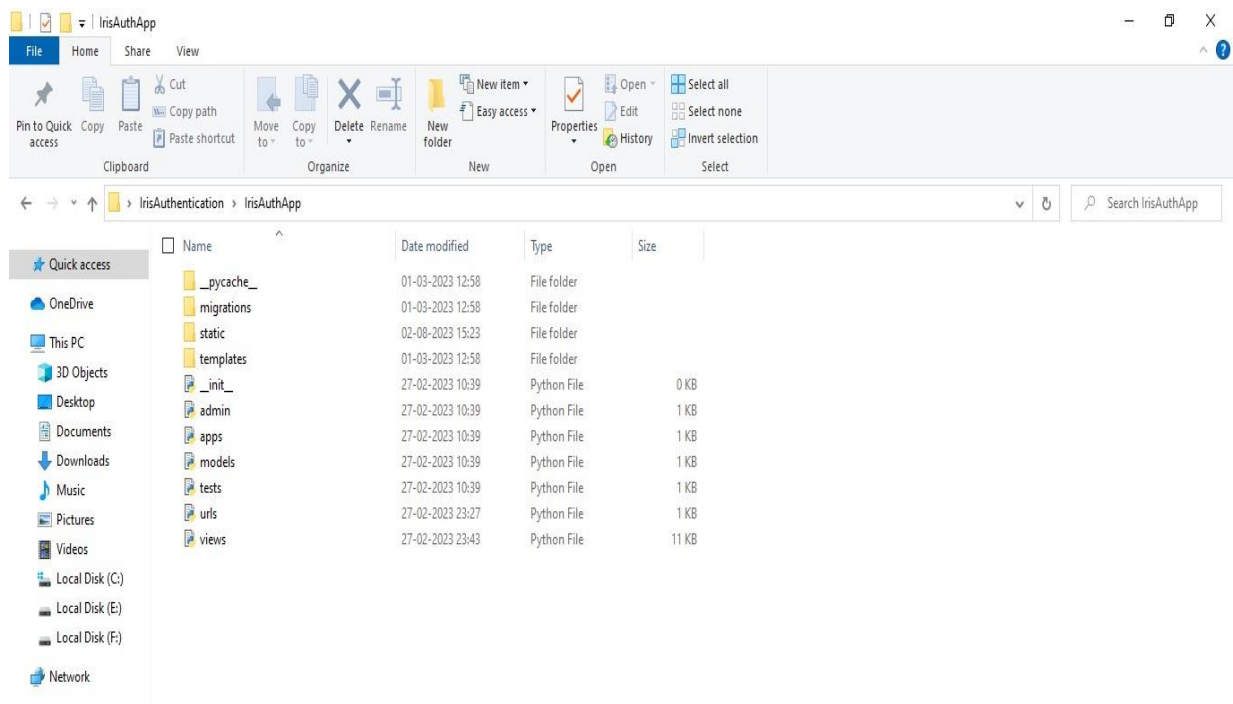


Fig 5.1 Django's components

5.1.5. URL Dispatcher:

- URL dispatcher maps URL patterns to corresponding views.
- It directs incoming HTTP requests to the appropriate view function or class based on the requested URL.
- URL patterns are defined in the project's URL configuration file (`urls.py`).
- Example (urls.py):

```
from django.urls import path

from .views import post_list

urlpatterns = [

    path("", post_list, name='post_list'),

]
```

5.1.6. Settings:

- Settings define the configuration of the Django project. They include database settings, static files, middleware, and more.
- Settings are typically stored in a Python module named `settings.py`.
- Example (settings.py):

```
DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE_DIR / 'db.sqlite3',

    }

}
```

5.1.7. Admin Interface:

- Django provides a built-in admin interface for managing site content.
- It allows administrators to create, read, update, and delete data using a web-based interface.
- Admin interface can be customized by registering models and defining admin classes.
- Example (admin.py):

```
from django.contrib import admin  
  
from .models import Post  
  
admin.site.register(Post)
```

This example demonstrates how Django's components work together to create a simple blog application. Models define the structure of blog posts, views retrieve posts from the database, templates render HTML pages to display posts, URL dispatcher maps URLs to views, settings configure the project, and the admin interface provides a convenient way to manage posts for web development.

Overall, Django's comprehensive set of features, including its powerful ORM, flexible URL routing, templating engine, and built-in admin interface, make it a popular choice for building web applications quickly and efficiently. Its emphasis on DRY (Don't Repeat Yourself) principles, security features, and scalability options make it suitable for projects of all sizes.

CHAPTER – 6

IRIS RECOGNITION

From the user Registration module, the eye image which is uploaded by the user need to be segmented and iris should be extracted. The widely used and well-known method of iris segmentation is Daugman's method.

```
img = cv2.imread('IrisAuthApp/static/watermark/'+user+"_original.png")
```

- By using the above line of code, we can read the eye image which is given by user while registration using OpenCV in Python .
- The image size initially we are taking to be (width- 320; height-240), so the total number of pixels in the image will be 76,800.

```
img = cv2.resize(img,(128,128), interpolation=cv2.INTER_CUBIC)
```

- This line of code is resizing the image `img` to a new size of 128x128 pixels using OpenCV's `resize` function.so, the total number of pixels will be 16,384.
- `interpolation=cv2.INTER_CUBIC`: This parameter specifies the interpolation method to be used during the resizing process. In this case, `cv2.INTER_CUBIC` is chosen, which represents bicubic interpolation. Bicubic interpolation is a method commonly used for resizing images as it produces smoother results compared to other interpolation methods, especially when the resizing factor is not too large.

```
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

- `cv2.cvtColor`: This function is used to convert the color space of an image from one color space to another.
- `img`: This is the image data that we have previously resized.
- `cv2.COLOR_BGR2GRAY`: This parameter specifies the conversion from the BGR color space (commonly used in OpenCV for reading images) to the grayscale color space. Grayscale images have only one channel, representing the intensity of light, ranging from 0 (black) to 255 (white), with shades of gray in between.
- After executing this line of code, the variable `gray_img` will contain the grayscale

representation of the original image ``img``. This conversion is often performed before applying certain image processing algorithms or when color information is not required for further processing.

6.1 find_iris Function

```
answer = find_iris(gray_img, daugman_start=10, daugman_end=30, daugman_step=1,  
                  points_step=3)
```

- This line of code is calling a function named ``find_iris`` with some parameters. Let's break down the parameters and their significance:
- ``gray_img``: This parameter represents the grayscale image obtained after converting the original image to grayscale. The iris detection algorithm typically operates on grayscale images as they contain only intensity information, which simplifies the processing.
- ``daugman_start=10``: This parameter specifies the starting radius for the Daugman's algorithm, which is used to detect the iris boundaries. It's common to start searching for the iris with a relatively small radius.
- ``daugman_end=30``: This parameter specifies the ending radius for the Daugman's algorithm. It determines the maximum radius within which the algorithm searches for the iris boundary.
- ``daugman_step=1``: This parameter defines the step size or increment used by the Daugman's algorithm while searching for the iris boundary. A smaller step size can lead to finer detection.
- ``points_step=3``: This parameter specifies the step size for sampling points within the iris boundary. It determines the spacing between sampled points, affecting the granularity of iris representation.
- After executing this line of code, the variable ``answer`` likely contains the result of the iris detection process, which could include information such as the coordinates and radius of the detected iris. This information can be further used for iris recognition or authentication purposes.

```
def find_iris(gray: np.ndarray, *, daugman_start: int, daugman_end: int,
daugman_step: int = 1, points_step: int = 1,) -> Tuple[Tuple[int, int], int]:
```

- The function will apply: func: `daugman` on every pixel in the calculated image slice. Basically, we are calculating where lies set of valid circle centers.
- It is assumed that iris center lies within central 1/3 of the image.
- param points_step: it will run daugman for each ``points_step``th point. It has linear correlation with overall iris search speed
- param daugman_start: bottom value for iris radius in pixels for: func: ``daugman``
- param daugman_end: top value for iris radius in pixels for: func: ``daugman``
- param daugman_step: step value for iris radii range in pixels for: func: ``daugman``. It has linear correlation with overall iris search speed
- return: radius with biggest intensiveness delta on image as `` ((xc, yc), radius) ``

```
h, w = gray.shape
```

```
if h != w:
```

```
    print ('Your image is not a square!')
```

- We first check whether the gray image is square or not, if not it prints as 'Your image is not a square!'.
- It is assumed that iris center lies within central 1/3 of the image.
- So, we reduce step for better accuracy. we will look only on dots within central 1/3 of image

```
single_axis_range = range(int(h / 3), h - int(h / 3), points_step)
```

- As we know h= 128, points_step = 3; therefore single_axis_range= range(42,86,3).

```
all_points = itertools.product(single_axis_range, single_axis_range)
```

```
print ("the points are:", all_points);
```

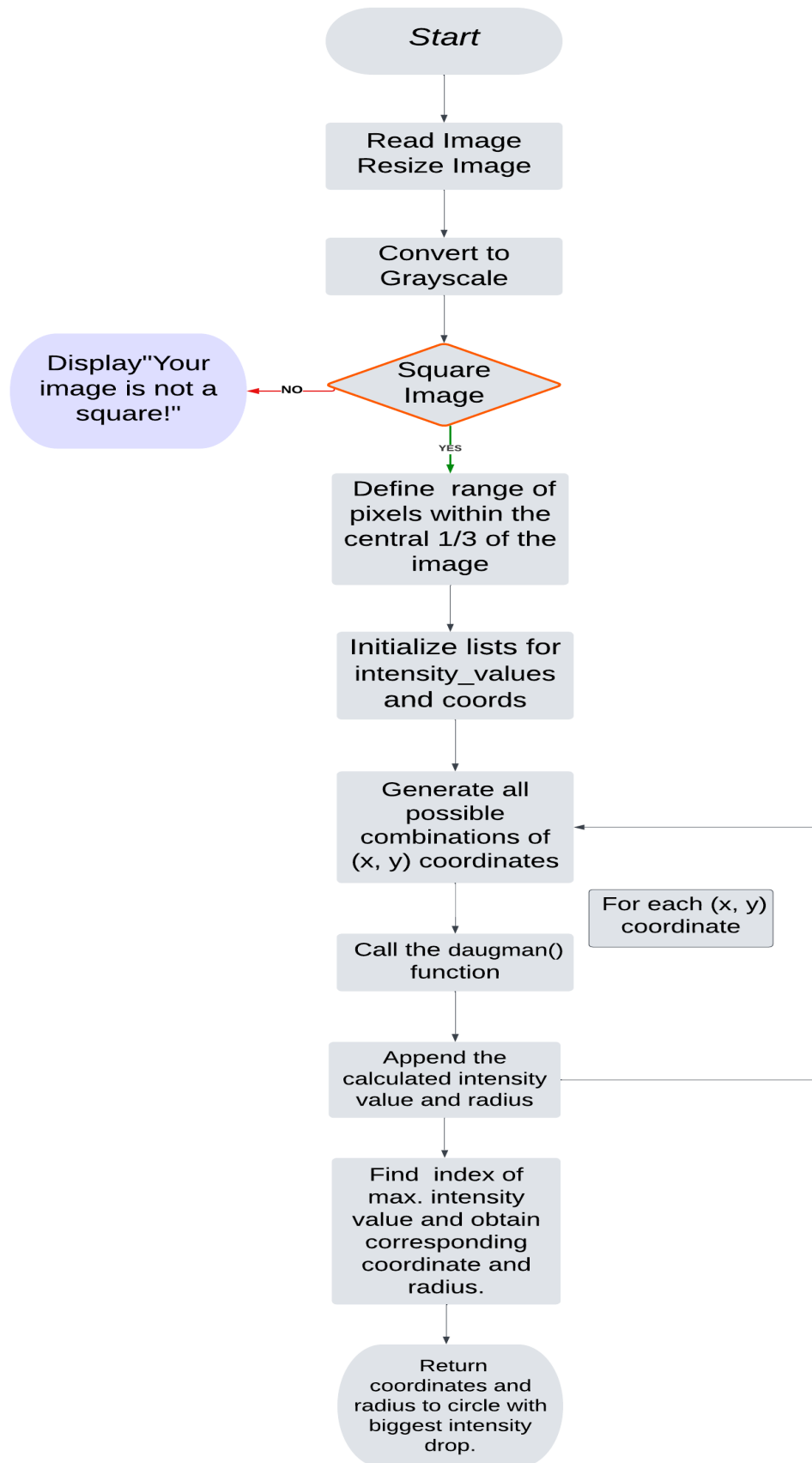


Fig 6.1 Flowchart of Iris Function

- The `itertools.product()` function generates the Cartesian product of the input iterables. In this case, `single_axis_range` is used as the input iterable twice.
- Since `single_axis_range` contains values ranging from 42 to 86, with a step size of 3, the output of `itertools.product(single_axis_range, single_axis_range)` will be all possible pairs of values from `single_axis_range`.

- Output:

(42, 42)

(42, 45)

(42, 48)

...

(85, 82)

(85, 85)

- Each tuple represents a pair of values `(x, y)` where `x` and `y` are both within the range of `single_axis_range` (from 42 to 86 with a step size of 3), covering all possible combinations of points within that range.

for point in all_points:

```
    val, r = daugman(gray, point, daugman_start, daugman_end, daugman_step)
```

```
    intensity_values.append(val)
```

```
    coords.append((point, r))
```

- Then we apply daugman function to every point we got from the above and we get maximum intensity value for every point with respect to the coordinates which are (center and radius) and we store them in `intensity_values`, `coords` respectively.

6.2 daugman Function:

```
def daugman(gray_img: np.ndarray, center: Tuple[int, int],
```

```
            start_r: int, end_r: int, step: int = 1)
```

- The function will calculate pixel intensities for the circles in the ``range(start_r, end_r, step)` for a given center, and find a circle that precedes the biggest intensity drop
- param start_r: bottom value for iris radius in pixels = 10
- param end_r: top value for iris radius in pixels = 30
- param step: step value for iris radii range in pixels = 1
- return: intensity_value, radius
- Let us define 2 parameters x, y = center ; intensities = [] which are used to store the point values and the output intensity values.

```
mask = np.zeros_like(gray_img)
```

- After executing this line of code, mask will be an NumPy array of zeros with the same dimensions as gray_img.
- We define radii in range (10, 30,1) which will be (10,11,12,13,14,15,16,17,18,19, ...,28,29).

1. Loop over Radii and Create Circular Mask:

- For every radius we draw a circle on the mask with the center as point we got from find-iris function.

```
radii = list (range (start_r, end_r, step))
for r in radii:
    cv2.circle(mask, center, r, 255, 1)
    diff = gray_img & mask
    intensities.append(np.add.reduce(diff[diff > 0]) / (2 *
                                                    math.pi * r))
    mask.fill(0)
```

- Here the mask will be in black color(as we initialized it with all zeros), then we are drawing circles with value 255(white) and thickness of 1 for every radii from 10 to 29.

3. Compute Differences:

- Then we taking new binary image which is Bitwise AND operation between the gray-image and the mask. This operation retrieves only common “ON” pixels(1’s) from both images and save in “diff” variable.

4. Compute Intensities:

- Then we compute intensity of the region of interest within the grayscale image and appends the result to the `intensities` list.
- “(diff[diff > 0])” :This part is responsible for extracting only the non-zero (1's) pixels from the diff binary image. It selects the pixels that are within the circular region defined by the current radius r.
- “np.add.reduce(...)” : This performs a reduction operation on the selected pixels. In this case, it calculates sum of pixel values within circular region.
- “(2 * math.pi * r)” : This part calculates the circumference (perimeter) of the circular region with the current radius r.

$\text{Average intensity of pixels with range (r)} = \frac{\text{Sum of pixel values within circular region}}{\text{Circumference of the circle}}$
--

Fig 6.2 Intensity formula

5. Reset Mask:

- `mask.fill(0)`: This line resets the `mask` image by filling it with zeros, preparing it for the next iteration with a different radius.
- Overall, this code will be iterating over a range of radii, creating circular masks at each radius, computing intensity values within the corresponding regions of interest in the grayscale image, and storing the intensity values in the `intensities` list for further analysis.

```
intensities_np = np.array(intensities, dtype=np.float32)
```

```
del intensities
```

- This line converts the intensities list to a NumPy array `intensities_np`. The `dtype=np.float32` specifies that the elements of the array should be of type float32 (single-precision floating-point numbers).

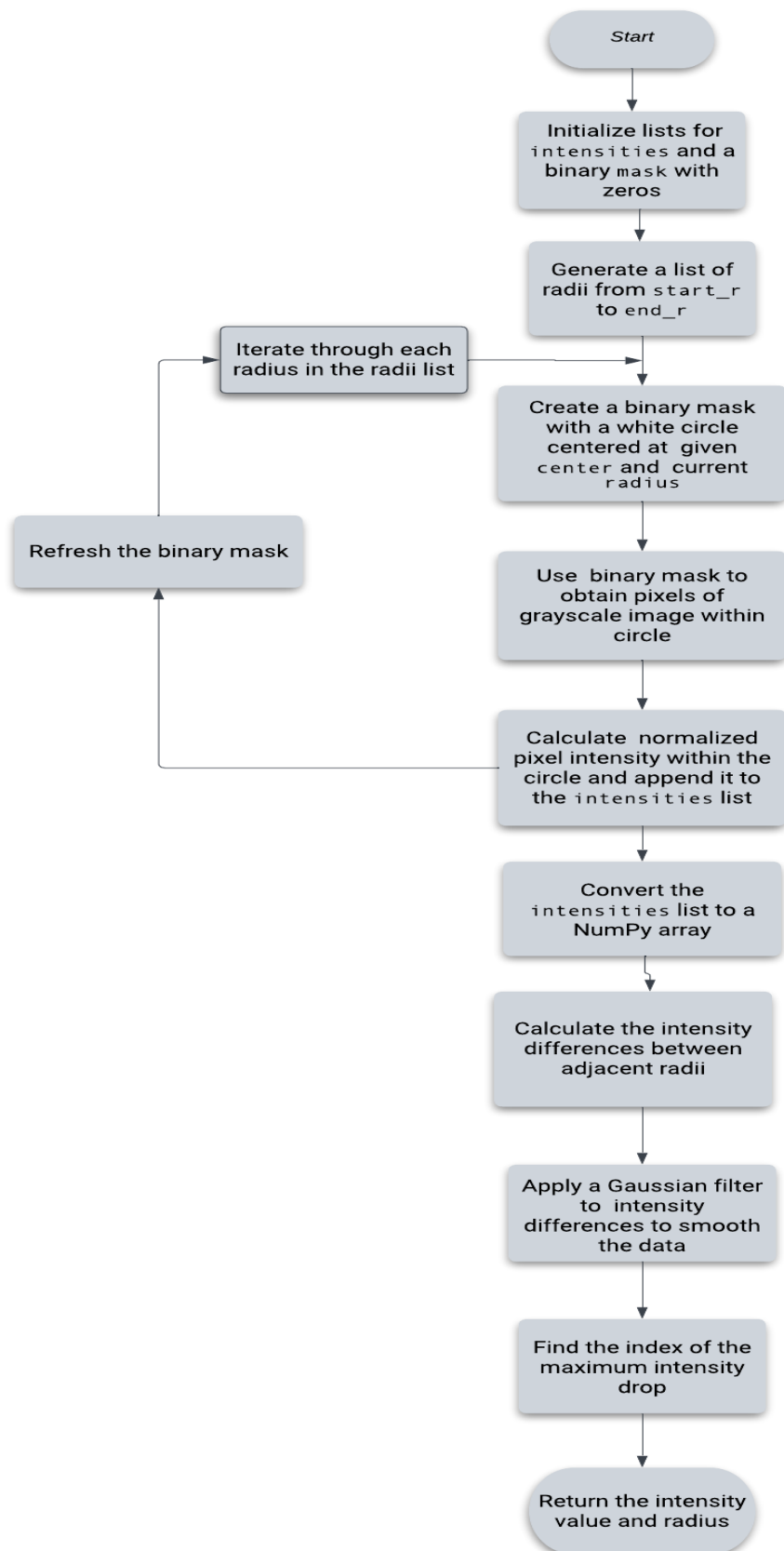


Fig 6.3 Flowchart of Daugman Function

- Converting the list to a NumPy array allows for more efficient calculations and operations on the data.
- `del intensities`: After creating the NumPy array, the original intensities list is deleted using the `del` statement. This step is performed to free up memory occupied by the list since it is no longer needed once the data is converted to a NumPy array.

$$\text{intensities_np} = \text{intensities_np}[:-1] - \text{intensities_np}[1:]$$

- The above code calculates the intensity differences between adjacent elements of the `intensities_np` NumPy array. This calculation is performed to find delta of radius intensiveness for each radius value in the array.
- `intensities_np[:-1]`: This slicing operation selects all elements of the `intensities_np` array except the last element. The slicing notation `[:-1]` indicates that we are excluding the last element from the array.
- `intensities_np[1:]`: This slicing operation selects all elements of the `intensities_np` array starting from the second element. The slicing notation `[1:]` indicates that we are excluding the first element from the array.
- This operator performs element-wise subtraction between the two sliced arrays.
- By subtracting the second array (excluding the first element) from the first array (excluding the last element), we effectively calculate the differences between adjacent elements of the original `intensities_np` array.
- For example, if `intensities_np = [1.5, 2.0, 1.8, 3.2]`,

$$\text{then } \text{intensities_np}[:-1] = [1.5, 2.0, 1.8],$$

$$\text{intensities_np}[1:] = [2.0, 1.8, 3.2]$$

$$\text{the result of the operation} = [1.5 - 2.0, 2.0 - 1.8, 1.8 - 3.2] = [-0.5, 0.2, -1.4]$$

- These differences represent the changes in intensity values from one radius to the next. The negative values indicate a decrease in intensity, while positive values indicate an increase in intensity as the radius changes. This information is important in determining the radius with the most significant intensiveness delta, which corresponds to the likely iris radius during the iris detection process.

```
intensities_np = abs(cv2.GaussianBlur(intensities_np, (1, 5), 0))
```

- Then we apply gaussian filter to our intensity values NumPy array. The Gaussian filter in our case is designed in MATLAB and is a 1 by 5 (rows by columns) vector with intensity values given by vector A = [0.0003 0.1065 0.7866 0.1065 0.0003]
- (1, 5): This tuple represents the size of the Gaussian kernel. The first value (1 in this case) indicates the width of the kernel, and the second value (5) indicates the height of the kernel. The kernel is essentially a matrix with specified dimensions and intensity values that are used to convolve with the image data.
- 0: This value represents the standard deviation of the Gaussian kernel. A value of 0 means that OpenCV will automatically calculate the standard deviation based on the kernel size.
- abs(...): The abs() function is applied to the result of the Gaussian blur. This step is used to ensure that the values in the intensities_np array remain positive. Gaussian blur can introduce negative values due to its filtering process, but in this case, the absolute values are taken to retain only the magnitude of the differences.
- The Gaussian blur applied to intensities_np helps to smooth and highlight the significant changes in intensity values between radii. By convolving the intensities_np array with the Gaussian kernel, the subtle changes in intensiveness are emphasized, making it easier to identify the radius with the most significant intensiveness delta, which corresponds to the probable iris radius.

```
idx = np.argmax(intensities_np)
return intensities_np[idx], radii[idx]
```

- This line finds the index of the maximum value in the intensities_np array. The np.argmax() function returns the index of the first occurrence of the maximum value in the array. In the context of the code, it corresponds to the index of the radius with the highest intensity value.
- Finally, code returns the maximum intensity and its associated radius.
- So, the above code finds the maximum intensity and radius of one point which is taken from all_points in find_iris function. And appended in intensity_values array and coords array respectively.
- In the similar way , we calculate the maximum intensities and its respective radius for all the

points from(42, 42) to (85,85) with step value as 3 and appended in intensity_values array and coords array respectively.

```
best_idx = intensity_values.index(max(intensity_values))  
print("the center and radius of best intensiveness is:",coords[best_idx]);  
return coords[best_idx]
```

- This calculates the index of the maximum intensity value in the intensity_values list. This index corresponds to the radius with the most significant intensiveness delta during the iris detection process.
- After executing this line, best_idx will hold the index of the maximum intensity value, which corresponds to the radius and center with the most significant intensiveness delta during the iris detection process.
- This index will be used to retrieve the corresponding radius value from the radii list, and tuple (xc, yc), radius will be returned as the output of the find_iris function. This tuple represents the likely iris center and radius during the iris detection process.

```
result = img[start-radius:start+radius,end-radius:end+radius]
```

- Here start – It is X-coordinate of detected Iris center
end – It is Y-coordinate of detected Iris center
radius – It is iris radius
- img[start-radius:start+radius]: This part of the slicing selects the rows of the img image from start - radius to start + radius - 1 (exclusive). This represents the vertical region of the image centered at the detected iris center.
- , end-radius:end+radius]: This part of the slicing selects the columns of the img image from end - radius to end + radius - 1 (exclusive). This represents the horizontal region of the image centered at the detected iris center.
- By combining both parts of the slicing, the code selects the rectangular region around the detected iris center with a width and height of 2 * radius. This region effectively contains the iris region of the image, cropped based on the detected iris center and radius.

```
result = cv2.resize(result, (64,64), interpolation=cv2.INTER_CUBIC)
```

- The cropped image(result) is then resized to a new size of(64*64) using bicubic interpolation.
- The result image the detected iris image which will be saved as “iris.png”.

CHAPTER – 7

IRIS WATERMARKING

- After the iris image is extracted from the Eye, it is used for embedding the Watermark message.
- During the sign-up process the user will be entering a message which will be watermarked into the iris image.
- Here we have used Crypto-Steganography from Python Library.
- Here the watermark message which is to be hidden is first encrypted using cryptographic algorithm to make it unreadable without a key.
- Then, the encrypted data is embedded within Iris segment using steganography techniques.
- The iris segment appears unchanged to the casual observer, but the secret data can be extracted by the intended recipient using the correct decryption key.
- This Watermarked Iris image is then stored in the database.

```
crypto_steganography.hide('IrisAuthApp/static/watermark/iris.png',  
                           'IrisAuthApp/static/watermark/'+user+'.png', message)
```

- Here, `crypto_steganography.hide()` function is a part of a cryptography library that enables hiding a message within an image using a technique called steganography. Let's break down the process step by step:
- Input Parameters:
 - `'IrisAuthApp/static/watermark/iris.png'`: This parameter specifies the path to the source image file where the message will be hidden. It's assumed that this image contains the segmented iris region.
 - `'IrisAuthApp/static/watermark/'+user+'.png'`: This parameter specifies the path to the output image file where the message-hidden image will be saved. It's likely named based on the user for identification.
 - `message`: This parameter contains the message that will be hidden within the image.
- Image Encoding:
 - The function reads the source image file specified by the first parameter. This image is the segmented iris region extracted from the user's original image.
 - It encodes the message into the pixel values of the image. This process involves modifying the least significant bits of the pixel values to represent the message data. This modification is imperceptible to the human eye, ensuring that the image looks unchanged to casual

viewers.

- **Save Processed Image:**
 - After embedding the message, the function saves the modified image to the output file specified by the second parameter. This file will contain the message-hidden image.
 - The file name often includes the user's identifier to associate the processed image with the corresponding user.
- **Output:**
 - The output of this process is an image file containing the embedded message. This file looks like a regular image to the naked eye but contains hidden data encoded within it. This hidden message can later be extracted using a corresponding function from the same cryptography library.

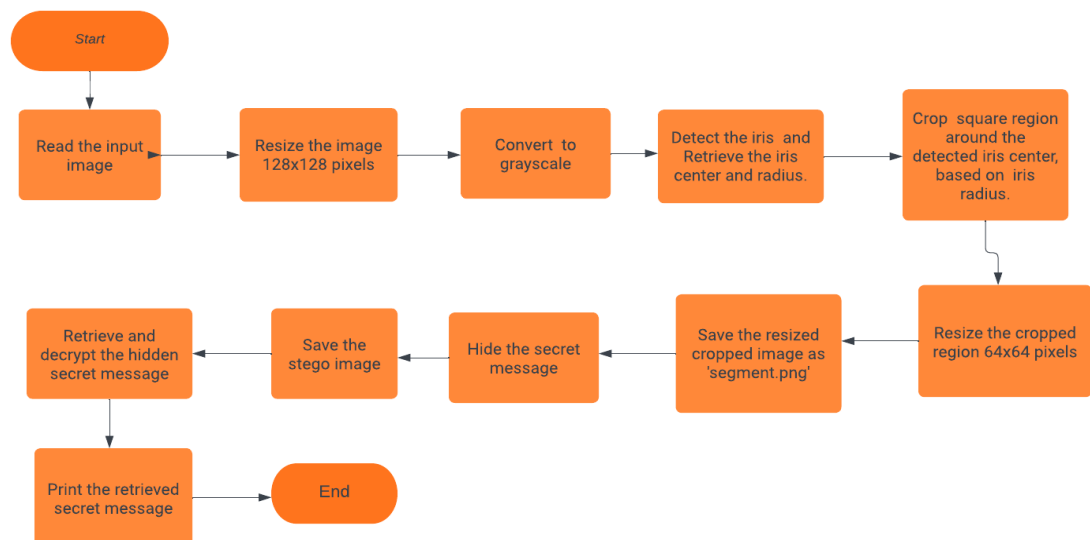


Fig 7.1 Flowchart of Iris recognition and watermarking

- We can also retrieve the secret message from the iris image using `crypto_steganography.retrieve()` function.
- So, in the given process whatever the eye image and message is given during the sign-up process of any user, the iris is detected and watermarked with the secret message and stored for further usage.

CHAPTER – 8

USER VALIDATION

- After the user registration is completed, they can login into their account by providing their username, password, watermarking message and also uploading their iris image.

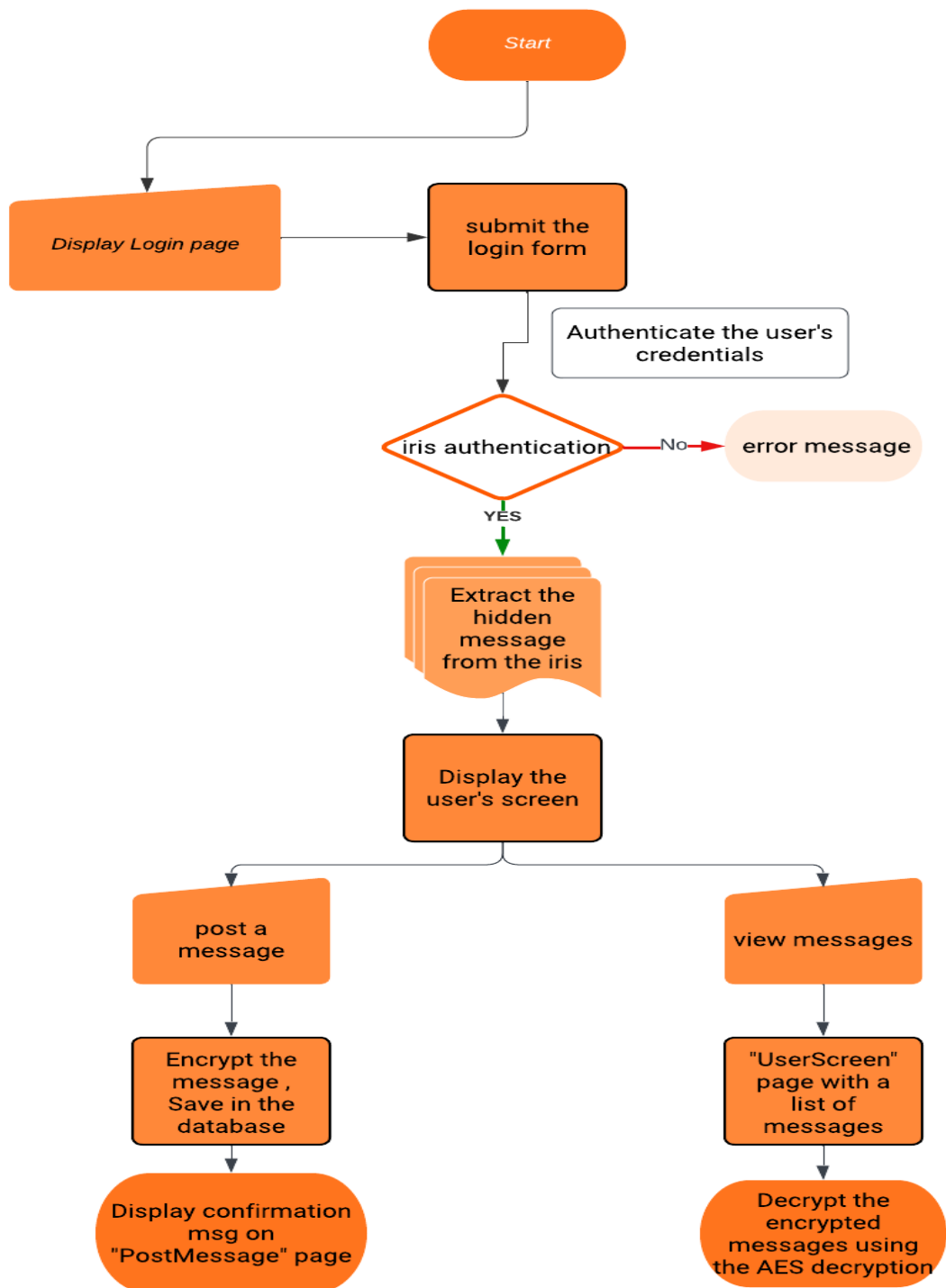


Fig 8.1 Website Functionality

- Once, the user entered all the information, the system will check for the details in the database and finds the watermarked iris image and then compares with the new iris image and its message.
- If every information is correctly validated then the user will enter into their account and securely send and receive the messages from one user to another.

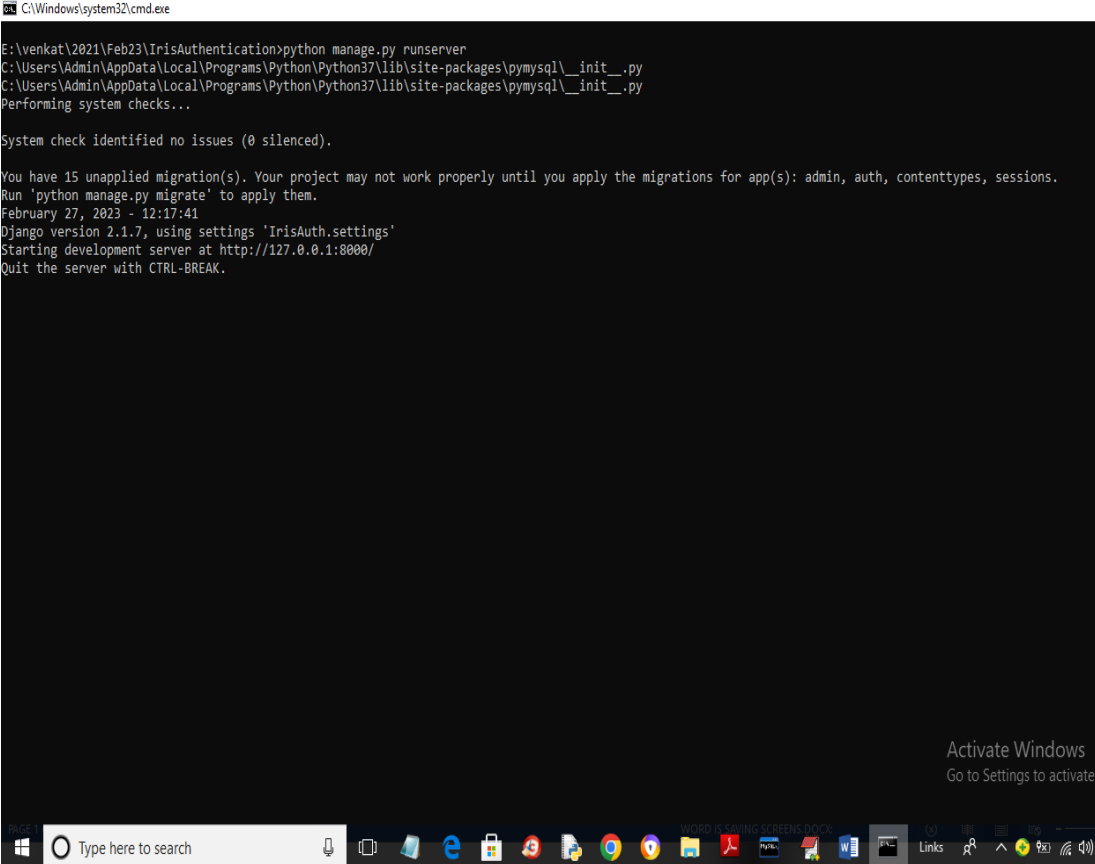
8.1 EXCHANGE OF MESSAGES:

- After both Logging and watermark authentication is Successful, the user can post messages, view message in the website.
- Here we are using AES Algorithm for Encryption and Decryption of messages for secure transfer of messages.
- The original message is changed throughout this procedure into a cipher text that cannot be decoded without the right decryption key.
- As a result, even if the message is intercepted, it will still be unclear.

CHAPTER – 9

RESULTS

- The proposed approach has been implemented using Python and Django Framework.
- The experimental result is based on 100 set of different irises.
- To run project first we should create database in MYSQL by copying content from DB.txt and paste in MYSQL and then double click on 'run.bat' file to start python DJANGO server and get below output



```
C:\Windows\system32\cmd.exe

E:\venkat\2021\Feb23\IrisAuthentication>python manage.py runserver
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\pymysql\__init__.py
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\pymysql\__init__.py
Performing system checks...

System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
February 27, 2023 - 12:17:41
Django version 2.1.7, using settings 'IrisAuth.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Fig 9.1 Command prompt showing server link

- In above screen server started and now open browser and enter URL as <http://127.0.0.1:8000/index.html> and press enter key to get below page.
- This website is created using HTML.

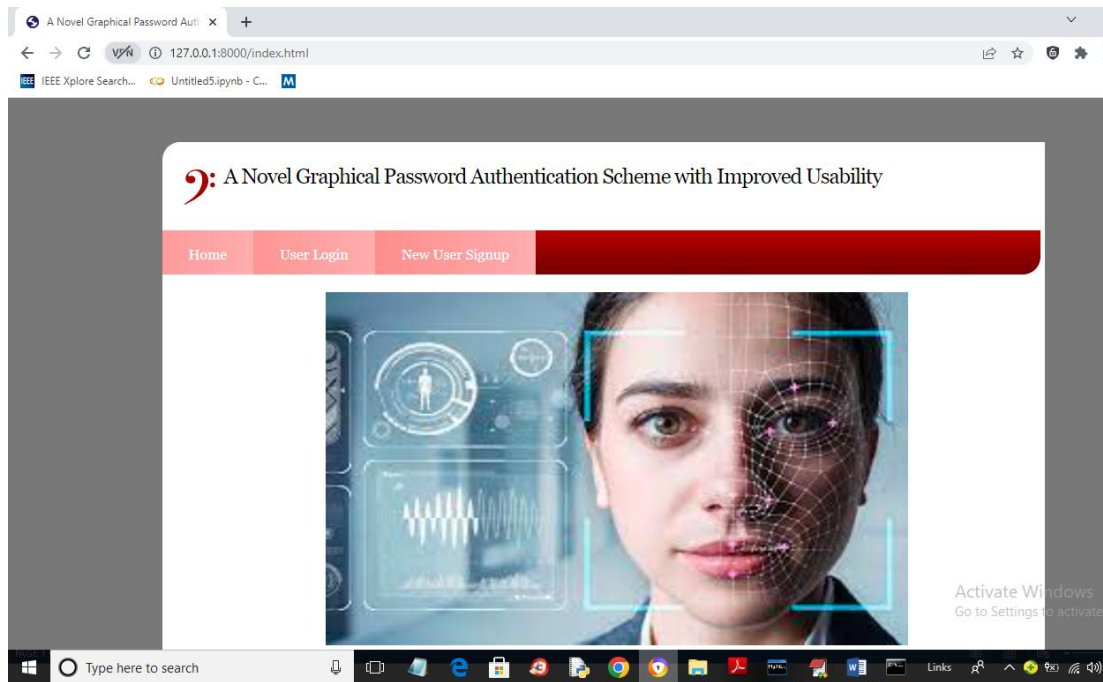


Fig 9.2 Home page of website

9.1 USER SIGN-UP PAGE:

- In above screen click on 'New User Signup' link to get below screen.

Fig 9.3 Signup screen

- Here, User need to provide the above information which is Username, Password, Contact No, Email Id, Address, Watermark Message. We need to upload our iris image also.

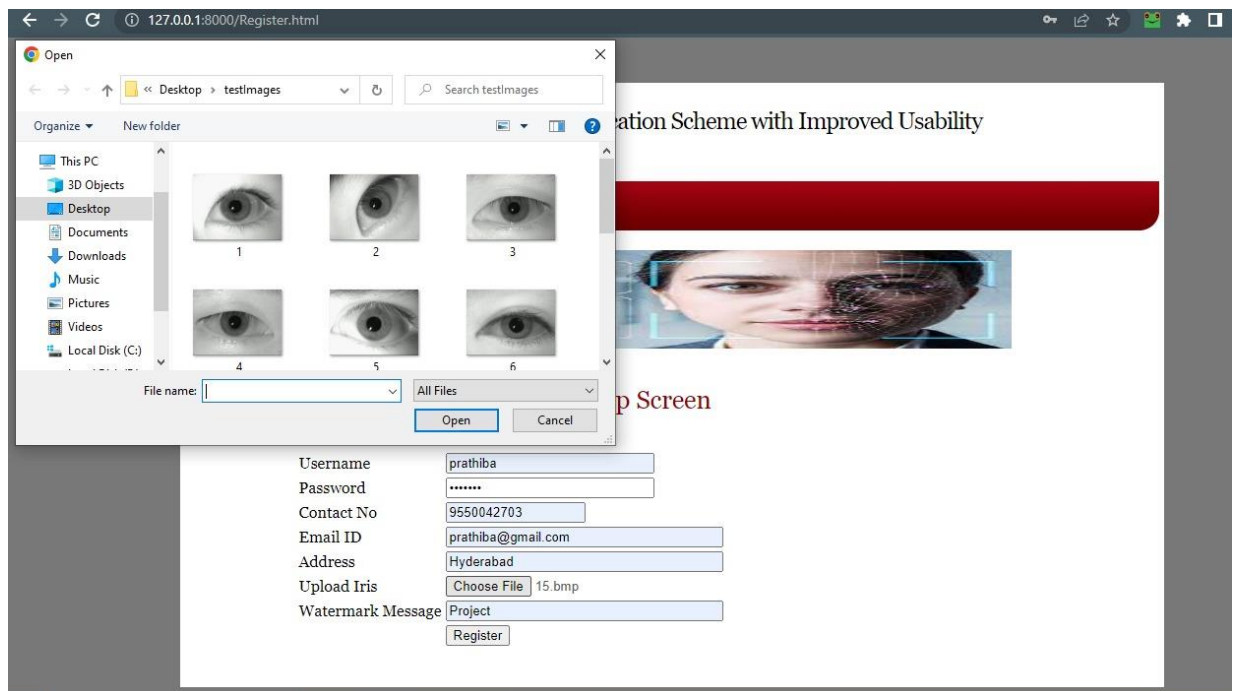


Fig 9.4 Uploading Eye image

- After entering all the information click on “Register” button. We get the below screen.

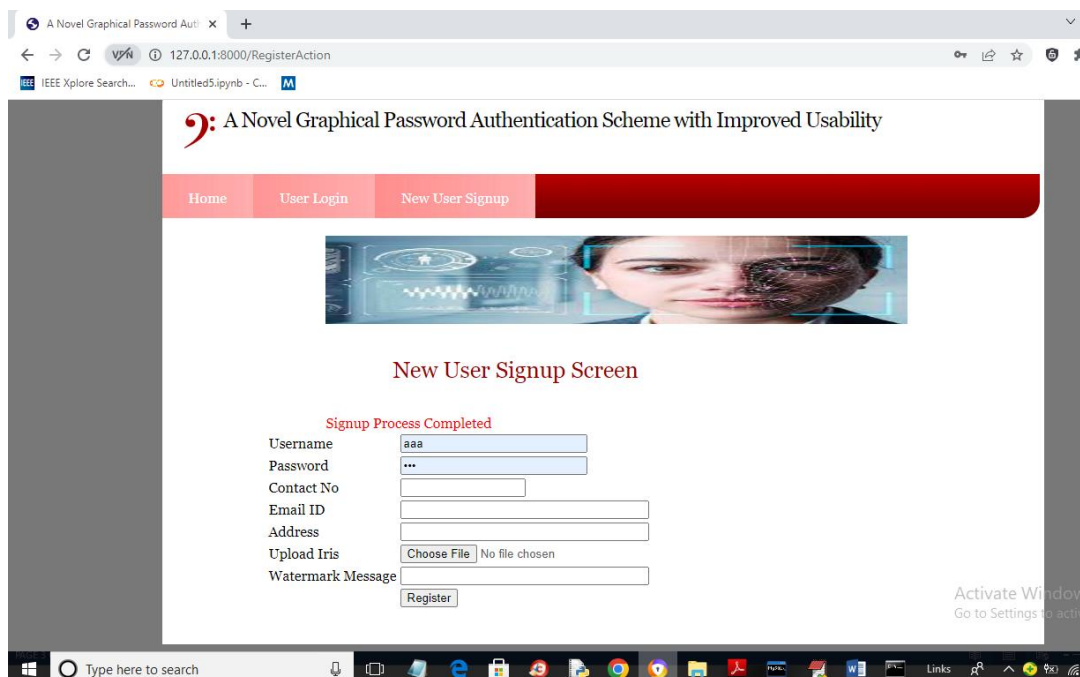
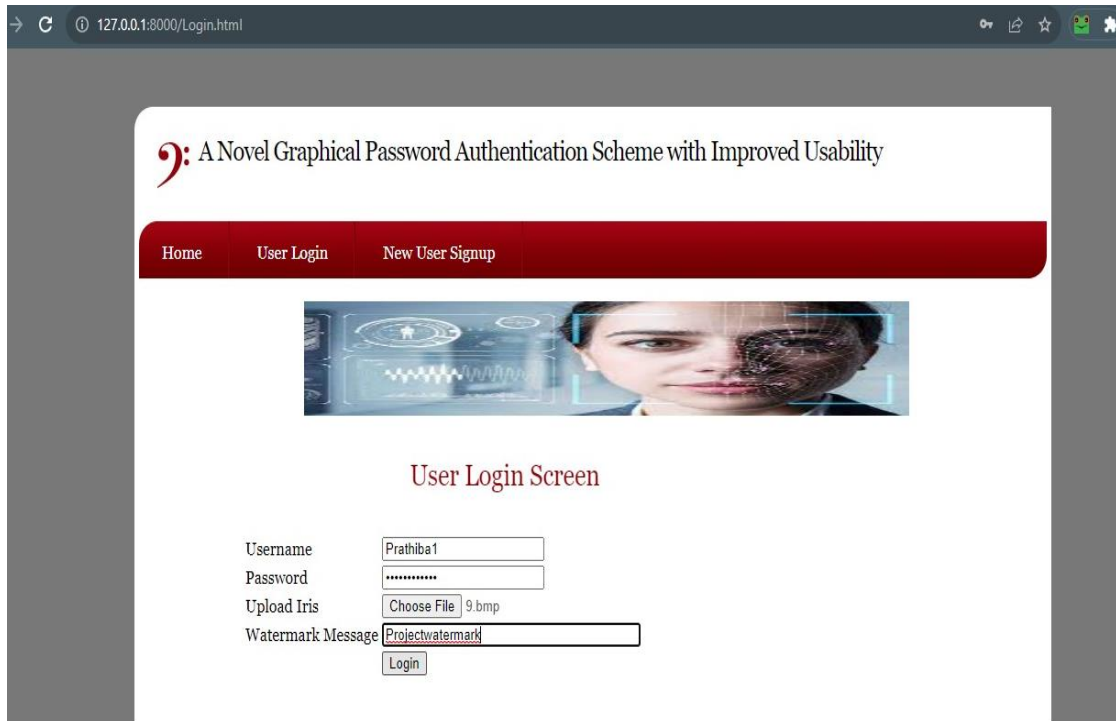


Fig 9.5 Successful Signup process

9.2 User Login page:

- In above screen signup details completed and now click on ‘User Login’ link to get below output



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/Login.html". The page title is "A Novel Graphical Password Authentication Scheme with Improved Usability". The navigation bar includes links for "Home", "User Login", and "New User Signup". The main content area features a large image of a woman's face with a graphical password overlay. Below the image, the text "User Login Screen" is displayed. The login form includes fields for "Username" (containing "Prathiba1"), "Password" (masked with asterisks), "Upload Iris" (with a "Choose File" button and "9.bmp" text), and "Watermark Message" (containing "Projectwatermark"). A "Login" button is positioned at the bottom of the form.

Fig 9.6 User Login Screen

- In above screen while entering all login details correctly and uploading correct image also and then click on Login button to get below output

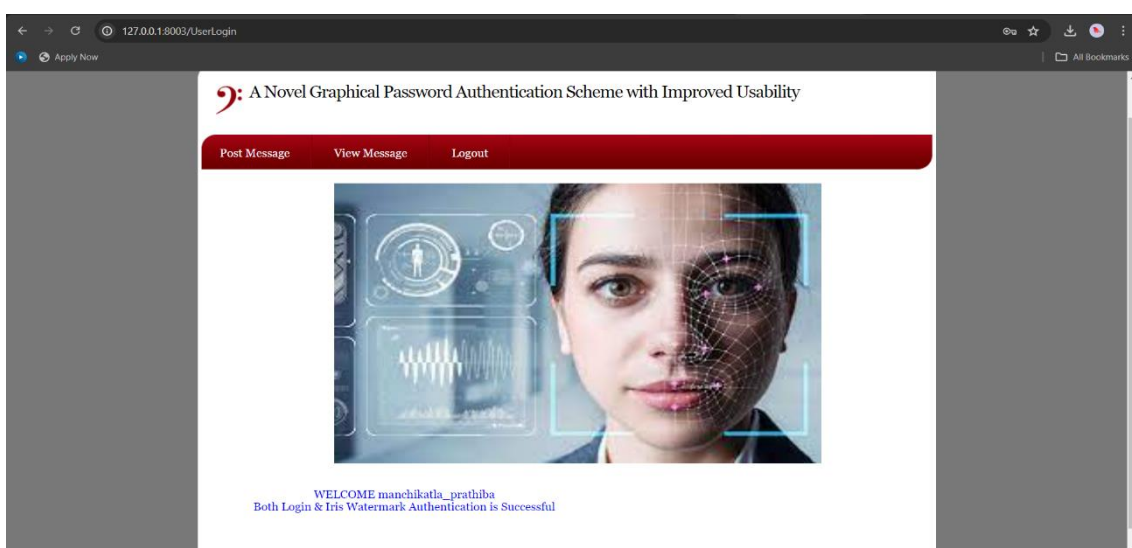


Fig 9.7 Login success

9.3 Transfer of messages :

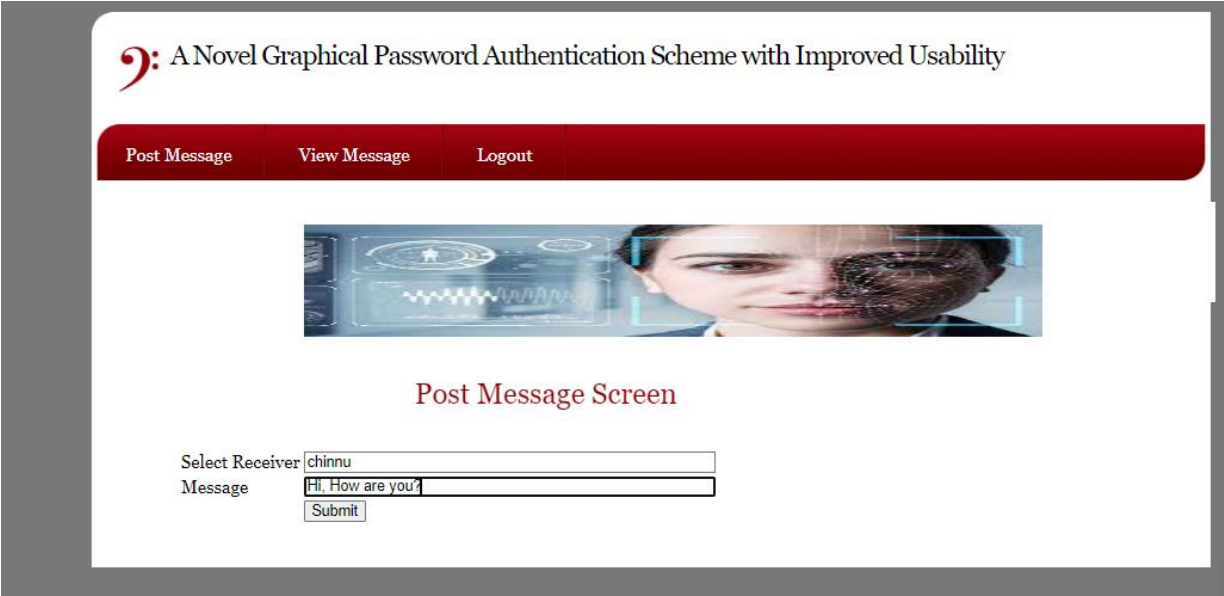


Fig 9.8 Post Message Screen

- After the user logged into the portal, they can Post message to any other users by just selecting the Receiver from the drop-down and submitting the message. We can also view the messages from View-Message column.

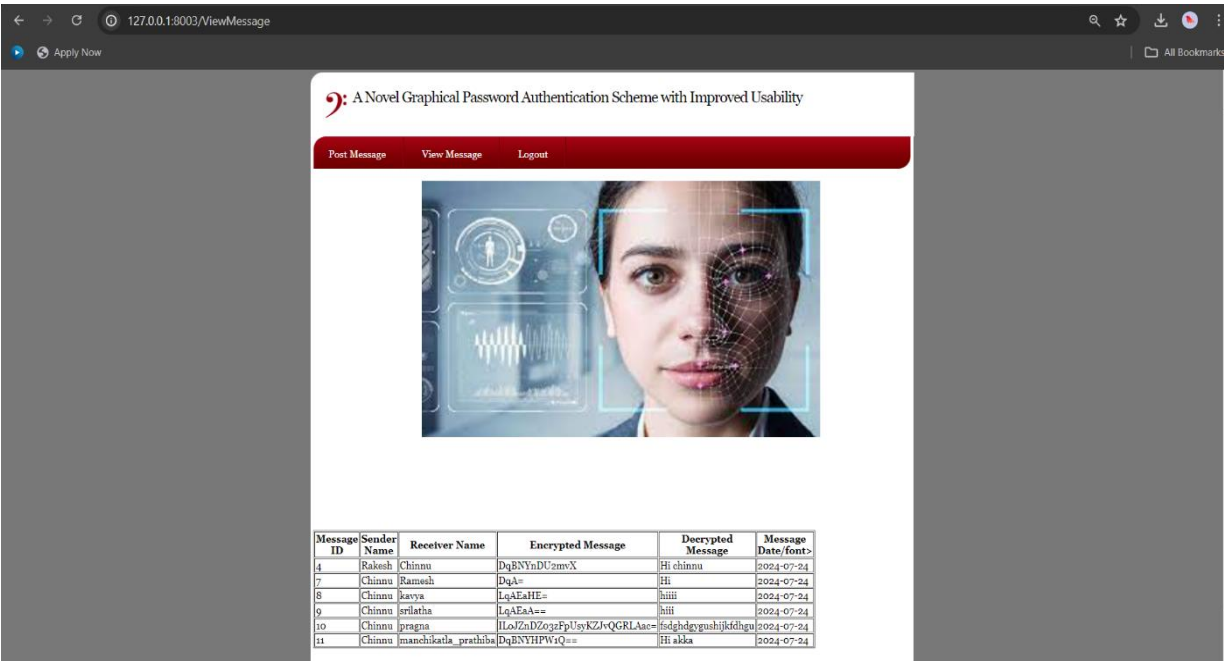


Fig 9.9 View-Message Column

9.4 Invalid Test cases:

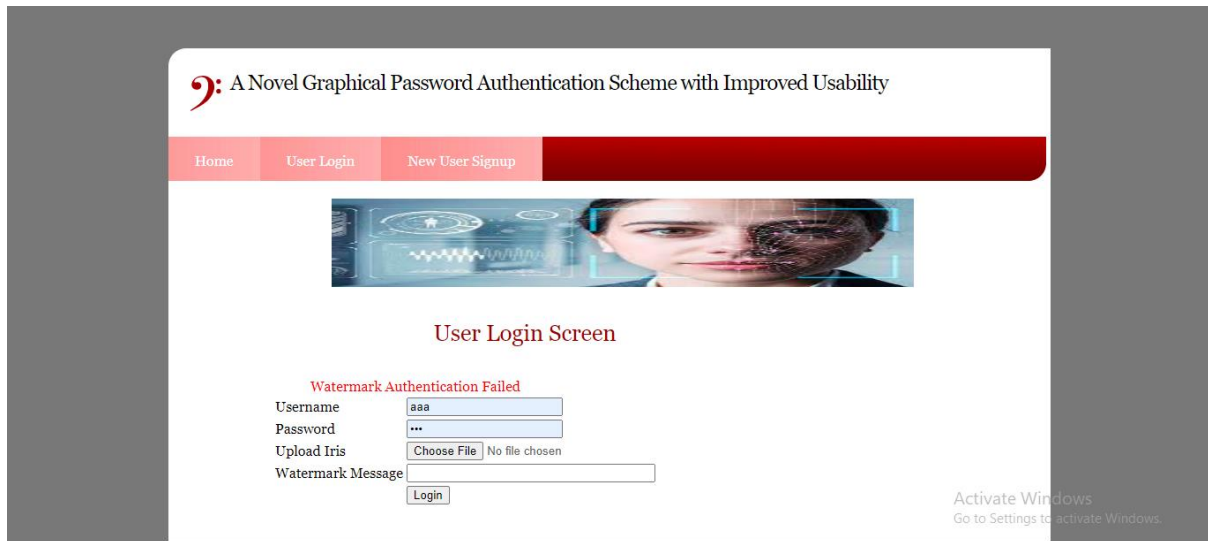


Fig 9.10 Invalid Watermark message

- In above screen I entered wrong watermark message and after pressing login will see watermark authentication failed.

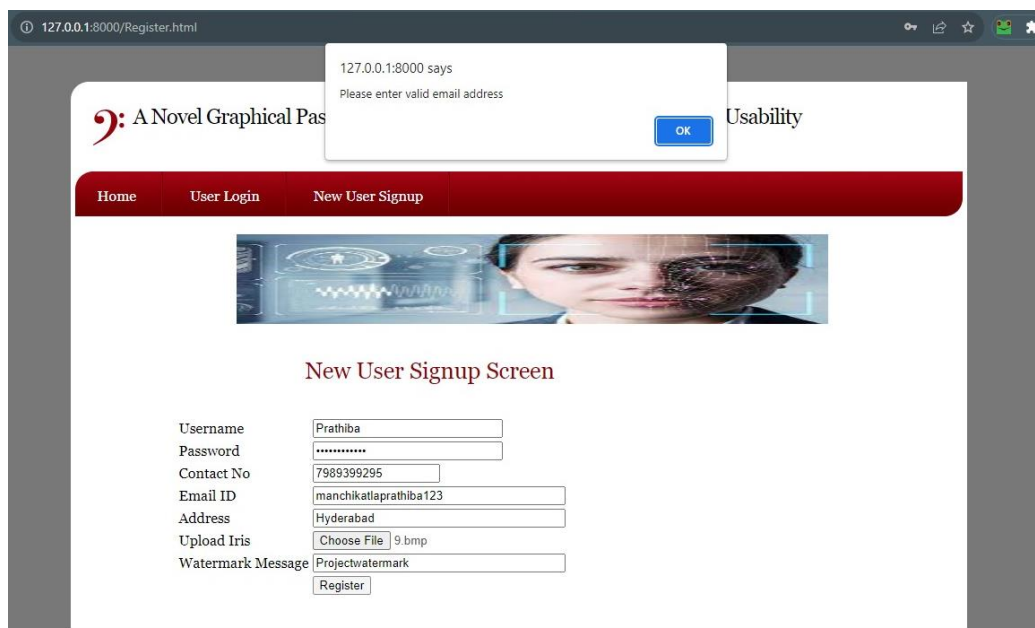


Fig 9.11 Invalid Email Address

- In above screen I entered invalid email-id in the signup process and it will be showing as “please enter valid email address”.
- In the same way If we enter Invalid phone number then it will be showing as “valid phone number must be entered”.

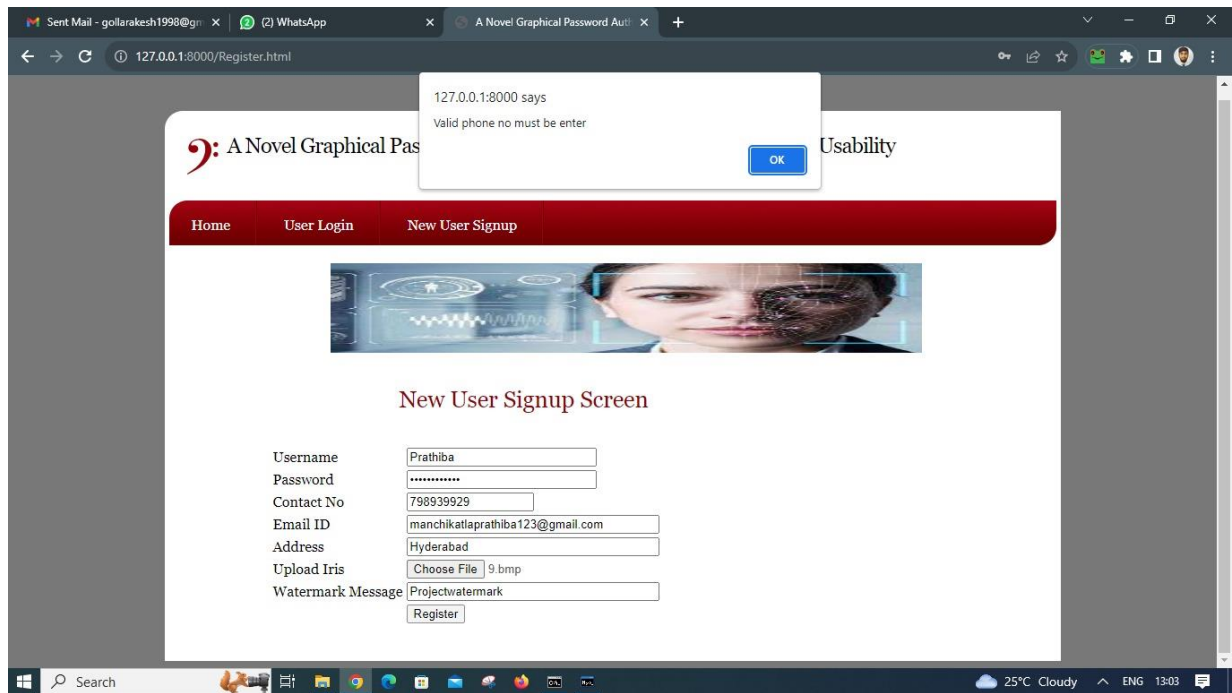


Fig 9.12 Invalid phone number

- In the similar way, if we enter some other image than an eye image, we will not be able to sign in.

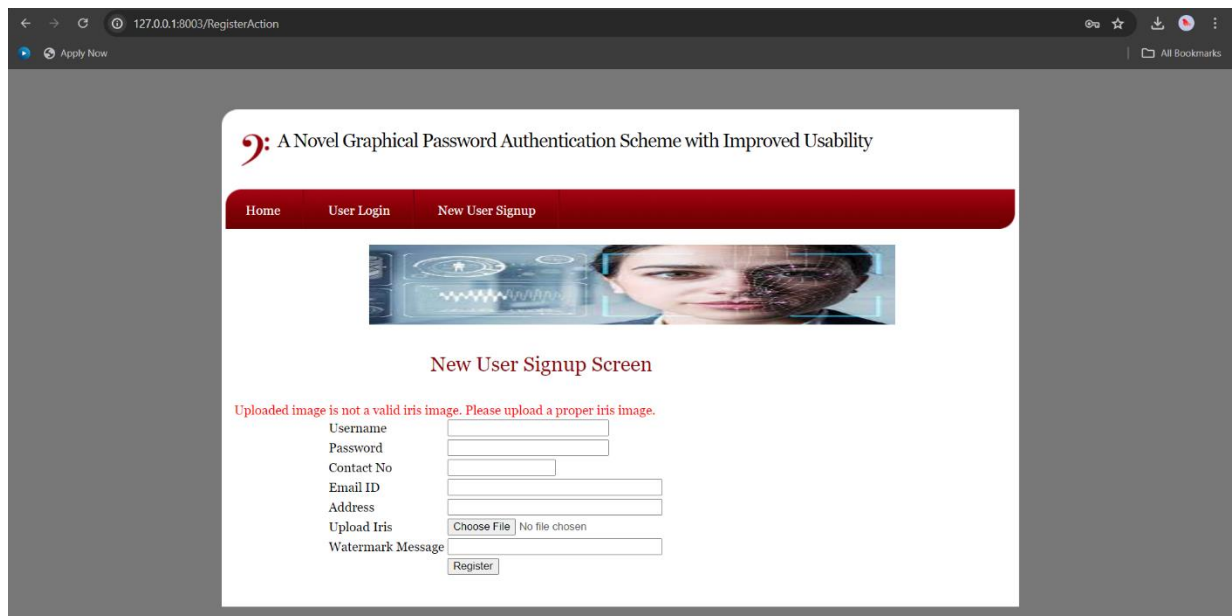


Fig 9.13 Invalid Image

- If a user already exists, we will not be able to register with the same username again.

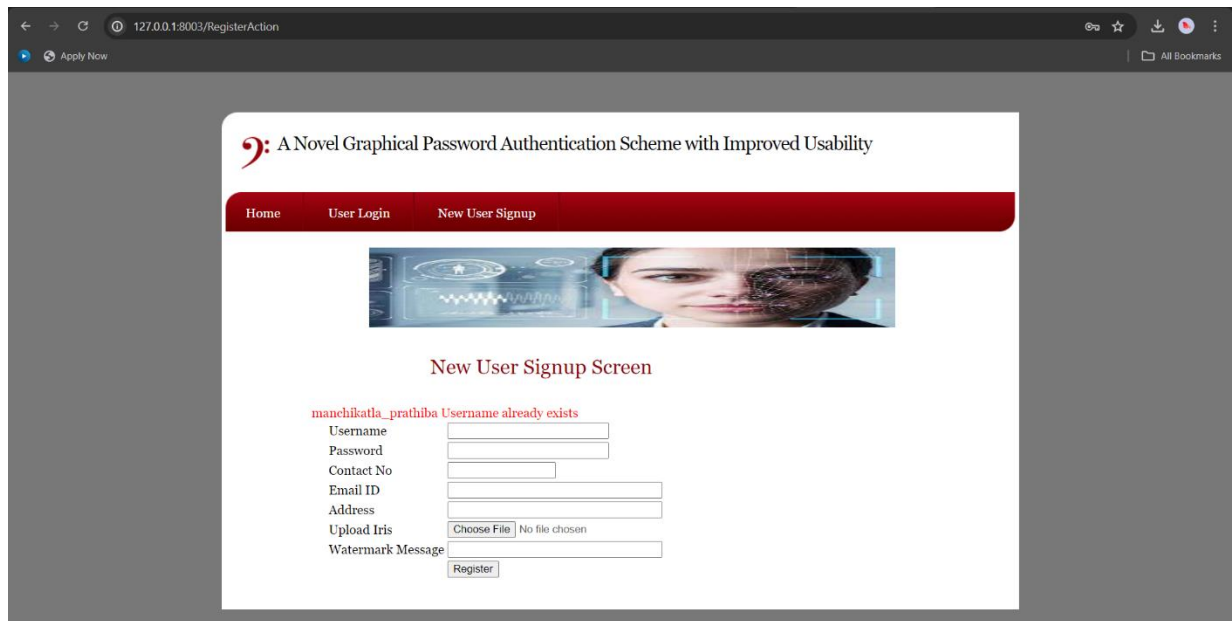


Fig 9.14 Duplicate Registration

- During Login, if any user enters wrong password or username, then they will not be able to login.

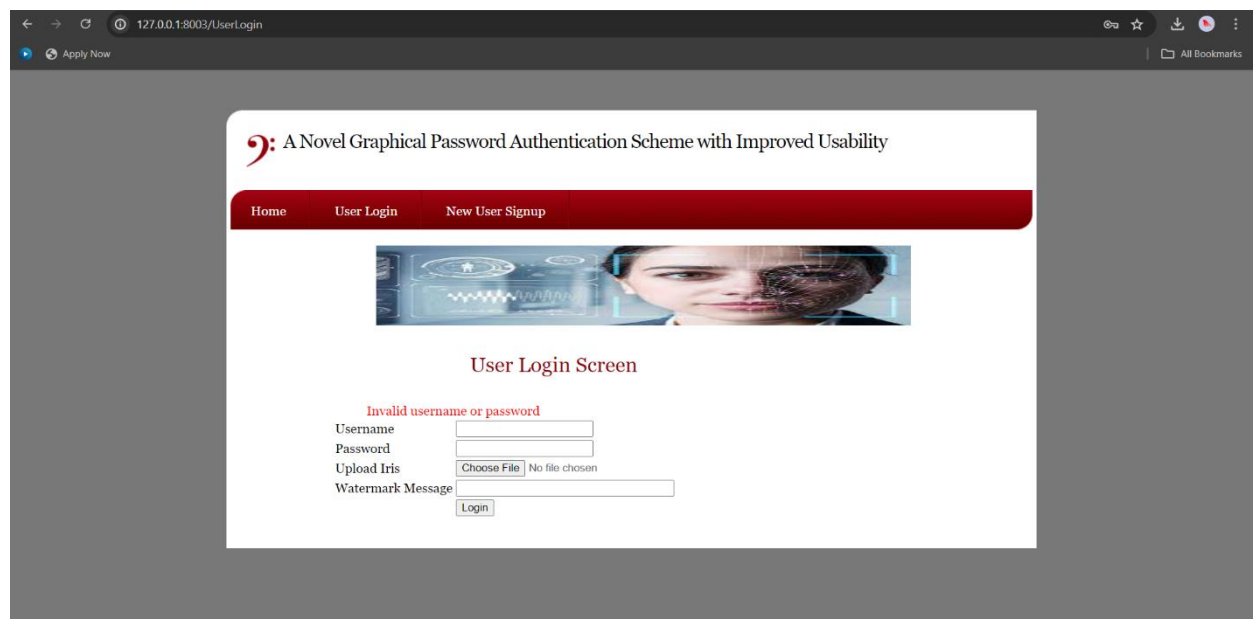


Fig 9.15 Invalid username or password

- During Login, if any user enters different image or wrong watermark image, they will not be able to login.

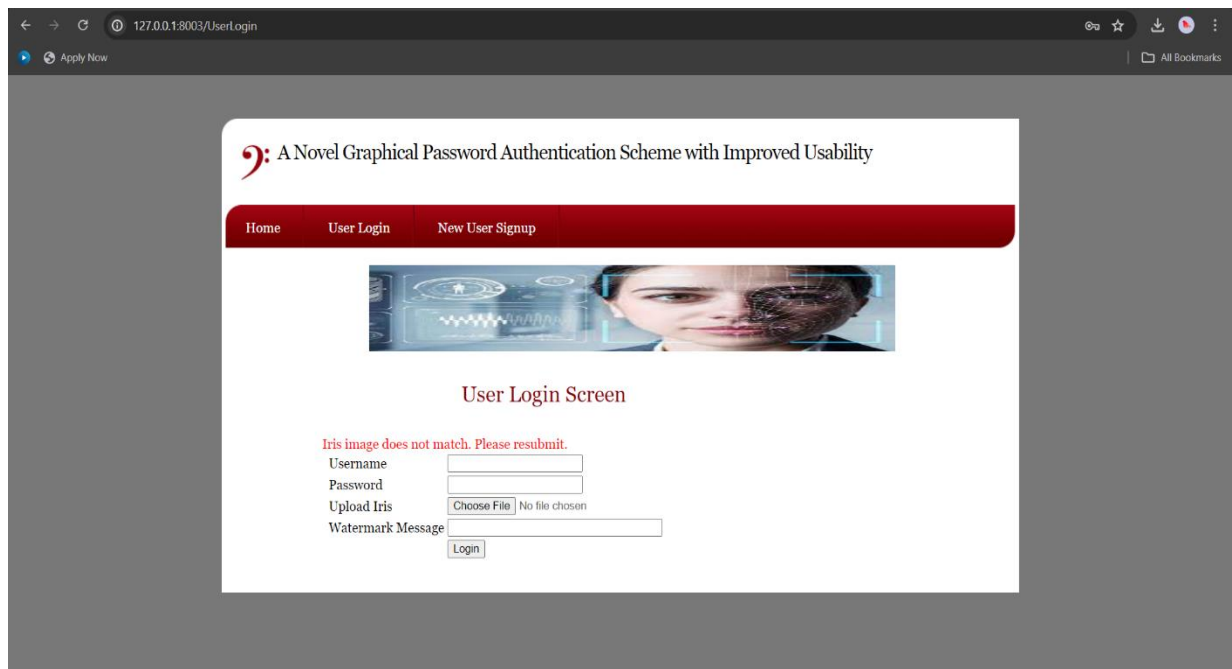


Fig 9.16 Invalid image or watermark message

CHAPTER -10

CONCLUSION AND FUTURE SCOPE

- In this project, a comprehensive system for secure message transmission and authentication using iris recognition and cryptographic techniques has been developed.
- The system successfully integrates iris authentication during login and utilizes AES encryption for message security.
- The watermarking technique has been employed to further enhance the security of iris authentication.
- The successful implementation of these technologies showcases the potential for robust and multi-layered security solutions in digital communication.
- The future scope of this project is multi-factor authentication where we can incorporate additional authentication factors such as biometric modalities or two-factor authentication for even stronger user verification.
- And also, Cross-Platform Compatibility where we can extend the system's compatibility to various platforms, including mobile devices, to enhance its usability and reach.

CHAPTER-11

REFERENCES

- [1] Sim Hiew Moi, Nazeema Binti Abdul Rahim, Puteh Saad, Pang Li Sim, Zalmiyah Zakaria, Subariah Ibrahim., “Iris Biometric Cryptography for Identity Document”, 2009 International Conference of Soft Computing and Pattern Recognition.
- [2] T Madhavi kumari and B Swathi “Iris Biometric Security using Watermarking and Visual Cryptography,” IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI-2017).
- [3] Mohammed A. M. Abdullah, Satnam S. Dlay, Wai L. Woo, Jonathon A. Chambers, “A Framework for Iris Biometrics protection: A Marriage Between Watermarking and Visual Cryptography”
- [4] Hao F., Anderson R., and Daugman J., “Combining crypto with biometrics effectively,” IEEE Transactions on Computers, vol. 55, no. 9, pp.1081–1088, 2006.
- [5] J. Daugman, “How iris recognition works,” IEEE Trans. Circuits Syst. Video Technol., vol. 14, no. 1, pp. 21–30, Jan. 2004.
- [6] Nathan D. Kalka, Jinyu Zuo, Natalia A. Schmid, Bojan Cukic Lane. Image quality assessment for iris. Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506, USA
- [7] Advances in biometric person authentication: 5th Chinese Conference on Biometric Recognition, SINOBIOMETRICS 2004, Guangzhou, China
- [8] R P Wildes. Iris recognition:an emerging biometric technology. Proc. IEEE, 85(9):1348–1363, 1997.
- [9] J G Daugman. High confidence visual recognition of persons by a test of statistical independence. IEEE Trans. Pattern Anal.Mach. Intell, 15(11):1148–1160, 1993.
- [10] L Dhir, N E Habib, D M Monro and S Rakshit. Effect of cataract surgery and pupil dilation on iris pattern recognition for personal authentication. 13 November 2009.

[11] <http://www.hrsid.com/iris-recognition>

[12] John Daugman. Recognizing persons by their iris patterns. Cambridge University Cambridge, UK. 2001.

[13] Peihua Li, Xiaomin Liu. An incremental method for accurate iris segmentation. Int. Conf. on Pattern Recognition, Florida, USA, 2008.

[14] J. Daugman. High confidence visual recognition of persons by a test of statistical independence. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, No. 11, 1993

CHAPTER-12

APPENDIX

12.1 Daugman.py Code:

```
import cv2
import numpy as np
import itertools
import math
from typing import Tuple, List

def daugman(gray_img: np.ndarray, center: Tuple[int, int], start_r: int, end_r: int, step: int = 1)
    x, y = center
    intensities = []
    mask = np.zeros_like(gray_img)
    radii = list(range(start_r, end_r, step)) # type: List[int]
    for r in radii:
        cv2.circle(mask, center, r, 255, 1)
        diff = gray_img & mask
        intensities.append(np.add.reduce(diff[diff > 0]) / (2 * math.pi * r))
        mask.fill(0)
    intensities_np = np.array(intensities, dtype=np.float32)
    del intensities
    intensities_np = intensities_np[:-1] - intensities_np[1:]
    intensities_np = abs(cv2.GaussianBlur(intensities_np, (1, 5), 0))
    idx = np.argmax(intensities_np) # type: int
    print("the intensity at index: ",intensities_np[idx]);
    print("the radius:",radii[idx]);
    return intensities_np[idx], radii[idx]
```

```

def find_iris(gray: np.ndarray, *,daugman_start: int, daugman_end: int,
             daugman_step: int = 1, points_step: int = 1):
    h, w = gray.shape
    if h != w:
        print('Your image is not a square!')
    single_axis_range = range(int(h / 3), h - int(h / 3), points_step)
    all_points = itertools.product(single_axis_range, single_axis_range)
    #print("the points are :",all_points)
    intensity_values = []
    coords = []
    for point in all_points:
        val, r = daugman(gray, point, daugman_start, daugman_end, daugman_step)
        intensity_values.append(val)
        coords.append((point, r))
    best_idx = intensity_values.index(max(intensity_values))
    print("the center and radius of best intensiveness is:",coords[best_idx]);
    return coords[best_idx]

```

12.2 Views.py code:

```

from django.shortcuts import render
from django.template import RequestContext
from django.contrib import messages
from django.http import HttpResponse
from django.conf import settings
import os
import pymysql
from django.core.files.storage import FileSystemStorage
from cryptosteganography import CryptoSteganography
from Daugman import find_iris
import cv2

```

```

from datetime import date

import pyaes, pbkdf2, binascii, os, secrets

import base64

from hashlib import sha256

global username

crypto_steganography = CryptoSteganography('securehiding')


def ExtractMessage(username, message):

    secret = "not exists"

    if os.path.exists('IrisAuthApp/static/watermark/'+username+'.png'):

        img1 = open("IrisAuthApp/static/test.png", "rb").read()

        img2 = open('IrisAuthApp/static/watermark/'+username+"_original.png", "rb").read()

        if img1 == img2:

            secret =

crypto_steganography.retrieve('IrisAuthApp/static/watermark/'+username+'.png')

            os.remove("IrisAuthApp/static/test.png")

        return secret


def getKey(): #generating key with PBKDF2 for AES

    password = "s3cr3t*c0d3"

    passwordSalt = '76895'

    key = pbkdf2.PBKDF2(password, passwordSalt).read(32)

    return key


def encrypt(plaintext): #AES data encryption

    aes = pyaes.AESModeOfOperationCTR(getKey(),
pyaes.Counter(3112954703500004730295243396765419539812423984456632288417216363
7846056248223))

    ciphertext = aes.encrypt(plaintext)

    return ciphertext

```

```

def decrypt(enc): #AES data decryption

    aes = pyaes.AESModeOfOperationCTR(getKey(),
pyaes.Counter(3112954703500004730295243396765419539812423984456632288417216363
7846056248223))

    decrypted = aes.decrypt(enc)

    return decrypted


def ViewMessage(request):

    if request.method == 'GET':

        global username

        font = '<font size="" color="black">'

        output = '<table border=1 align=center>'

        output+='<tr><th><font size=3 color=black>Message ID</font></th>'

        output+='<th><font size=3 color=black>Sender Name</font></th>'

        output+='<th><font size=3 color=black>Receiver Name</font></th>'

        output+='<th><font size=3 color=black>Encrypted Message</font></th>'

        output+='<th><font size=3 color=black>Decrypted Message</font></th>'

        output+='<th><font size=3 color=black>Message Date</font></th></tr>'

        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root',
database = 'irisauth',charset='utf8')

        with con:

            cur = con.cursor()

            cur.execute("select * FROM messages where receiver='"+username+"' or
sender='"+username+"'")

            rows = cur.fetchall()

            for row in rows:

                enc = row[3]

                enc = base64.b64decode(enc)

                decrypted = decrypt(enc)

                decrypted = decrypted.decode("utf-8")

                output += "<tr><td>"+font+str(row[0])+"</td>"

```

```

        output += "<td>" + font + row[1] + "</td>"
        output += "<td>" + font + row[2] + "</td>"
        output += "<td>" + font + row[3] + "</td>"
        output += "<td>" + font + decrypted + "</td>"
        output += "<td>" + font + row[4] + "</td></tr>"

output += "<br/><br/><br/><br/><br/><br/>"
context = {'data': output}
return render(request, 'UserScreen.html', context)

```

```
def PostMessage(request):
```

```
    if request.method == 'GET':
```

```
        global username
```

```
        output = '<select name="t1">'
```

```
        con = pymysql.connect(host='127.0.0.1', port=3306, user='root', password='root',
        database='irisauth', charset='utf8')
```

```
        with con:
```

```
            cur = con.cursor()
```

```
            cur.execute("select username FROM register")
```

```
            rows = cur.fetchall()
```

```
            for row in rows:
```

```
                if row[0] != username:
```

```
                    output += '<option value="' + row[0] + ">' + row[0] + '</option>'
```

```
        context = {'receivers': output}
```

```
        return render(request, 'PostMessage.html', context)
```

```
def PostMessageAction(request):
```

```
    global username
```

```
    if request.method == 'POST':
```



```

receiver = request.POST.get('t1', False)

msg = request.POST.get('t2', False)

today = str(date.today())

msg = encrypt(msg)

msg = str(base64.b64encode(msg),'utf-8')

msg_id = 0

con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root',
database = 'irisauth',charset='utf8')

with con:

    cur = con.cursor()

    cur.execute("select max(msg_id) FROM messages")

    rows = cur.fetchall()

    for row in rows:

        msg_id = row[0]

if msg_id is not None:

    msg_id = msg_id + 1

else:

    msg_id = 1

db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password =
'root', database = 'irisauth',charset='utf8')

db_cursor = db_connection.cursor()

student_sql_query = "INSERT INTO
messages(msg_id,sender,receiver,message,msg_date)
VALUES('"+str(msg_id)+"','"+username+"','"+receiver+"','"+msg+"','"+today+"')"

db_cursor.execute(student_sql_query)

db_connection.commit()

output = "Encrypted Msg = "+msg+"<br/>"

output += "Message ID = "+str(msg_id)+"<br/>"

output += "Message Sent to Receiver : "+receiver+"<br/>"

context= {'data':output}

return render(request, 'PostMessage.html', context)

```

```

def UserLogin(request):
    global username
    if request.method == 'POST':
        username = request.POST.get('t1', False)
        password = request.POST.get('t2', False)
        image = request.FILES['t3']
        message = request.POST.get('t4', False)
        if os.path.exists("IrisAuthApp/static/test.png"):
            os.remove("IrisAuthApp/static/test.png")
        fs = FileSystemStorage()
        fs.save("IrisAuthApp/static/test.png", image)
        status = "failed"

        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root',
        database = 'irisauth',charset='utf8')

        with con:
            cur = con.cursor()
            cur.execute("select username,password FROM register")
            rows = cur.fetchall()
            for row in rows:
                if row[0] == username and row[1] == password:
                    status = 'success'
                    break
            if status == 'success':
                output = 'Login Details Matched'
                extract_msg = ExtractMessage(username, message)
                if extract_msg == message:
                    status = 'Welcome username : '+username+'<br/>Both Logging & Iris Watermark
Authentication Successfull"

```

```

        context= {'data':status}

        return render(request, 'UserScreen.html', context)

    else:

        context= {'data':'Watermark Authentication Failed'}

        return render(request, 'Login.html', context)

    else:

        context= {'data':'Invalid username'}

        return render(request, 'Login.html', context)


def Register(request):

    if request.method == 'GET':

        return render(request, 'Register.html', {})


def index(request):

    if request.method == 'GET':

        return render(request, 'index.html', {})


def Login(request):

    if request.method == 'GET':

        return render(request, 'Login.html', {})


def watermarkImage(user, message):

    img = cv2.imread('IrisAuthApp/static/watermark/'+user+"_original.png")

    img = cv2.resize(img,(128,128), interpolation=cv2.INTER_CUBIC)

    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    answer = find_iris(gray_img, daugman_start=10, daugman_end=30, daugman_step=1,
points_step=3)

    iris_center, iris_rad = answer

    start = iris_center[0]

```

```

end = iris_center[1]
radius = iris_rad

result = img[start-radius:start+radius,end-radius:end+radius]
result = cv2.resize(result, (64,64), interpolation=cv2.INTER_CUBIC)
segmented = result

if os.path.exists('IrisAuthApp/static/watermark/'+username+'.png'):
    os.remove('IrisAuthApp/static/watermark/'+username+'.png')
cv2.imwrite('IrisAuthApp/static/watermark/iris.png', segmented)
crypto_steganography.hide('IrisAuthApp/static/watermark/iris.png',
'IrisAuthApp/static/watermark/'+user+'.png', message)

if os.path.exists("IrisAuthApp/static/watermark/iris.png"):
    os.remove("IrisAuthApp/static/watermark/iris.png")

def RegisterAction(request):
    if request.method == 'POST':
        global username
        username = request.POST.get('t1', False)
        password = request.POST.get('t2', False)
        contact = request.POST.get('t3', False)
        email = request.POST.get('t4', False)
        address = request.POST.get('t5', False)
        image = request.FILES['t6']
        message = request.POST.get('t7', False)
        if os.path.exists('IrisAuthApp/static/watermark/'+username+"_original.png"):
            os.remove('IrisAuthApp/static/watermark/'+username+"_original.png")
        fs = FileSystemStorage()
        fs.save('IrisAuthApp/static/watermark/'+username+"_original.png", image)
        output = "none"

        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root',
database = 'irisauth',charset='utf8')

```

```

with con:

    cur = con.cursor()

    cur.execute("select username FROM register")

    rows = cur.fetchall()

    for row in rows:

        if row[0] == username:

            output = username+" Username already exists"

    if output == "none":

        db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password
= 'root', database = 'irisauth',charset='utf8')

        db_cursor = db_connection.cursor()

        student_sql_query = "INSERT INTO
register(username,password,contact,email,address)
VALUES('"+username+"','"+password+"','"+contact+"','"+email+"','"+address+"')"

        db_cursor.execute(student_sql_query)

        db_connection.commit()

        print(db_cursor.rowcount, "Record Inserted")

        if db_cursor.rowcount == 1:

            watermarkImage(username, message)

            context= {'data':'Signup Process Completed'}

            return render(request, 'Register.html', context)

        else:

            context= {'data':'Error in signup process'}

            return render(request, 'Register.html', context)

    else:

        context= {'data':output}

        return render(request, 'Register.html', context)

```