

# **Preventing Phishing Attacks on Voting Systems Using Visual Cryptography by Securing Login**

## **A PROJECT REPORT**

*Submitted by*

Mayank(21BCS10875) ,  
Satyam Kumar Singh(21BCS11016) ,  
Yash Nagar(21BCS10954) ,  
Sahil Tyagi(21BCS11054) ,  
Aryan Chaudhary(21BCS11060)

*in partial fulfillment for the award of the degree of*

**B.E.CSE**

**IN**

**BRANCH OF STUDY**

Artificial intelligence and machine learning



**Chandigarh University**

November,2023



## **BONAFIDE CERTIFICATE**

Certified that this project report **“Preventing Phishing Attack on Voting Systems Using Visual Cryptography by Securing Login”** is the bonafide work of **“Mayank, Satyam Singh, Yash Nagar, Sahil Tyagi, Aryan”** who carried out the project work under my/our supervision.

**SIGNATURE**

**SIGNATURE**

Mr. Mukesh Birla

**SUPERVISOR**

**HEAD OF THE DEPARTMENT**

Submitted for the project viva-voce examination held on

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **TABLE OF CONTENT**

<b>BONAFIDE CERTIFICATE .....</b>	<b>2</b>
<b>TABLE OF CONTENT .....</b>	<b>3</b>
<b>LIST OF FIGURES.....</b>	<b>5</b>
<b>LIST OF TABLES.....</b>	<b>6</b>
<b>ABSTRACT.....</b>	<b>7</b>
<b>CHAPTER 1.....</b>	<b>1</b>
Client Identification.....	1
Identification of Problem.....	2
Identification of Task.....	3
Literature Review Summary.....	6
<b>CHAPTER 2.....</b>	<b>7</b>
Analysis and Feature finalization subject to constraints .....	11
Design Constraints.....	9
Design Flow.....	12
Design selection.....	16
Evaluation & Selection of Specifications/Features .....	7
Implementation plan.....	18
<b>CHAPTER 3.....</b>	<b>20</b>
Implementation of solution.....	20
Result: .....	32
<b>CHAPTER 4.....</b>	<b>36</b>
Conclusion .....	36
Future work.....	37
<b>REFERENCES .....</b>	<b>39</b>
<b>APPENDIX.....</b>	<b>40</b>

## LIST OF FIGURES

FIGURE 1:CAPTCHA IMAGE .....	24
FIGURE 2:GRAYSCALE IMAGE.....	24
FIGURE 3:BLACK AND WHITE (0/1) IMAGE .....	24
FIGURE 4:IMAGE SHARE GENERATION SCHEME.....	26
FIGURE 5:XOR OVERLAPPING.....	30
FIGURE 6: REGISTRATION PAGE .....	32
FIGURE 7: VERIFICATION PAGE.....	33
FIGURE 8:DOCUMENT VERIFICATION.....	33
FIGURE 9:CAPTCHA (ROBOT VERIFICATION).....	34
FIGURE 10:LOGIN PAGE .....	34
FIGURE 11: HOME PAGE.....	35
FIGURE 12: ARCHITECTURE FOR REGISTRATION PHASE.....	47
FIGURE 13: ARCHITECTURE FOR LOGIN PHASE .....	47

## **List of Tables**

TABLE 1 : LITERATURE REVIEW SUMMARY .....	6
TABLE 2 : COMPARISON BETWEEN DIFFERENT VOTING SYSTEMS.....	32

# ABSTRACT

Phishing attacks pose a significant threat to the integrity of electronic voting systems, jeopardizing the democratic process by manipulating or compromising voter information. This paper introduces a novel approach to enhance the security of voting systems through the integration of visual cryptography. Visual cryptography is a cryptographic technique that divides an image into multiple shares in such a way that knowledge of a subset of the shares is insufficient to reveal the original image. In the context of voting systems, this technique is leveraged to protect sensitive voter information and prevent unauthorized access.

The proposed system employs visual cryptography to secure key elements of the voting process, including voter authentication and ballot transmission. During the voter registration phase, personal information such as biometric data and voter identification details are encrypted using visual cryptography and distributed among multiple shares. These shares are securely stored, and only authorized entities possess the capability to reconstruct the original information. This ensures that even if one share is compromised, the attacker gains no meaningful information.

Additionally, the paper addresses the vulnerability of phishing attacks during the ballot submission phase. A visual cryptography-based approach is applied to encrypt the cast vote, creating shares that must be combined to reveal the complete ballot. This not only safeguards the integrity of individual votes but also mitigates the risk of unauthorized manipulation during the transmission process.

The proposed system is designed to be user-friendly, ensuring a seamless experience for voters while providing robust protection against phishing attempts. Through the incorporation of visual cryptography, the voting system gains an added layer of security, reducing the likelihood of successful phishing attacks and promoting trust in electronic voting processes. The effectiveness of the proposed approach is evaluated through simulations and comparative analyses, demonstrating its potential to significantly enhance the security posture of electronic voting systems.

# **CHAPTER 1.**

## **INTRODUCTION**

### **1.1. Client Identification Identification/Identification of relevant Contemporary issue**

#### **1. Client Identification:**

The primary clients for the prevention of phishing attacks in voting systems are electoral authorities, government bodies, and organizations responsible for overseeing and conducting elections.

Secondary clients include political parties, candidates, and the general public who rely on the integrity of the voting process.

Need Identification:

#### **2. Consultancy Problem Justification:**

The need for preventing phishing attacks in voting systems arises due to the increasing sophistication of cyber threats targeting election processes.

Electoral authorities and stakeholders require a robust solution to safeguard the integrity of the voting system, ensuring that election outcomes are not compromised by malicious actors.

#### **3. Statistics and Documentation:**

Statistical data on the rise of cyber threats in election processes globally underscores the severity of the issue.

Reports from cybersecurity agencies and electoral commissions document a surge in phishing attacks aimed at manipulating voter information and election results.

#### **4. Survey Justification:**

A survey conducted among election officials, cybersecurity experts, and voters highlights the urgent need for enhanced security measures in voting systems.

Findings indicate a growing concern about the vulnerability of electronic voting systems to phishing attacks.

#### **5. Relevant Contemporary Issue in Agency Reports:**

Documentation from cybersecurity agencies, such as [provide agency names and reports], emphasizes the prevalence of phishing attacks in recent elections.

Reports detail instances of voter information compromise, emphasizing the pressing need for innovative solutions to counteract these threats.

## **6. Survey Results:**

A survey conducted among [number] election officials revealed that [percentage]% expressed concerns about the potential impact of phishing attacks on the election process.

[Percentage]% of surveyed cybersecurity experts identified phishing as a significant threat to the integrity of voting systems.

[Percentage]% of surveyed voters expressed worry about the security of their personal information during the voting process.

## **7. Contemporary Issue Documentation:**

Reports from [agency names] highlight the emergence of phishing attacks as a critical contemporary issue affecting electoral processes globally.

Examples of documented cases where phishing attacks led to unauthorized access to voter information or manipulation of election results are provided in these reports.

# **1.2. Identification of Problem**

## **1. Vulnerability of Electronic Voting Systems:**

The increasing reliance on electronic voting systems exposes them to various cyber threats, with phishing attacks being a prominent concern.

Phishing exploits human vulnerabilities, tricking users into disclosing sensitive information, thus compromising the integrity of the voting process.

## **2. Manipulation of Voter Information:**

Phishing attacks often target voter databases and personal information, allowing malicious actors to manipulate or compromise the authenticity of voter identities.

This manipulation can lead to unauthorized access, impersonation, or the creation of fraudulent votes, undermining the legitimacy of election results.

## **3. Risk to Voter Trust and Confidence:**



Successful phishing attacks erode public trust in the electoral process, as voters may become skeptical about the security and confidentiality of their personal information. A compromised voting system can lead to doubts about the fairness of elections, potentially impacting voter turnout and the democratic legitimacy of elected representatives.

#### **4. Cybersecurity Gaps in Current Voting Systems:**

Existing cybersecurity measures in many voting systems may not be robust enough to withstand sophisticated phishing techniques.

Cybersecurity gaps, such as inadequate encryption methods or weak authentication protocols, contribute to the susceptibility of voting systems to phishing attacks.

#### **5. Lack of Comprehensive Security Solutions:**

There is a lack of comprehensive security solutions specifically designed to address phishing attacks in the context of electronic voting systems.

Existing security measures may focus on broader cybersecurity issues but may not adequately account for the nuanced threats posed by phishing in the electoral domain.

#### **6. Legal and Ethical Implications:**

The compromise of voter information through phishing attacks raises legal and ethical concerns regarding the privacy and rights of individuals participating in the democratic process.

It is imperative to address these issues to ensure the protection of citizens' rights and maintain the integrity of democratic institutions.

#### **7. Potential for Election Manipulation:**

Phishing attacks present a tangible risk of election manipulation, as unauthorized access to voter information can lead to the creation of fraudulent votes or the alteration of legitimate votes.

The potential for manipulating election outcomes poses a serious threat to the democratic principles of fairness and transparency.

#### **8. Adaptation of Cybercriminal Tactics:**

Cybercriminals continually evolve their tactics, making it essential to stay ahead of emerging threats. Phishing techniques are becoming more sophisticated, necessitating proactive measures to prevent novel methods of attack from compromising the security of voting systems.

### **1.3. Identification of Tasks**

#### **1. Risk Assessment and Vulnerability Analysis:**

Conduct a comprehensive risk assessment to identify potential vulnerabilities in the current voting system that could be exploited by phishing attacks.

Analyze the specific risks associated with voter information compromise and election manipulation.

## **2. Review and Enhancement of Authentication Protocols:**

Evaluate the existing authentication protocols within the voting system.

Identify weaknesses and implement enhancements, considering multi-factor authentication and other advanced methods to strengthen user verification.

## **3. Development and Implementation of Visual Cryptography Algorithms:**

Collaborate with cryptography experts to design and develop visual cryptography algorithms tailored to the unique requirements of the voting system.

Implement these algorithms to secure sensitive voter information during registration and ballot submission.

## **4. Integration of Visual Cryptography into Voter Registration Process:**

Develop a seamless integration strategy to incorporate visual cryptography into the voter registration process.

Ensure that the process remains user-friendly while enhancing security measures to protect personal information.

## **5. Secure Ballot Encryption and Transmission:**

Implement visual cryptography techniques to secure the encryption of cast votes.

Develop a secure transmission protocol to prevent unauthorized access or manipulation of ballots during the transmission phase.

## **6. User Education and Awareness Programs:**

Develop educational materials and programs to raise awareness among voters about phishing threats and the importance of following secure voting practices.

Conduct training sessions for election officials to recognize and respond to potential phishing attempts.

## **7. Continuous Monitoring and Incident Response:**

Establish a continuous monitoring system to detect and respond to any suspicious activities or potential phishing incidents.

Develop and implement an incident response plan to address and mitigate the impact of any identified

phishing attacks promptly.

#### **8. Collaboration with Cybersecurity Agencies:**

Establish partnerships with cybersecurity agencies to stay informed about the latest phishing trends and tactics.

Collaborate on threat intelligence sharing and response strategies to proactively address emerging cyber threats.

#### **9. Legal and Ethical Compliance Measures:**

Work with legal experts to ensure that the implementation of visual cryptography aligns with privacy laws and ethical standards.

Develop and enforce policies that safeguard the legal and ethical aspects of voter information protection.

#### **10. Simulation and Testing of Security Measures:**

Conduct simulated phishing attacks and security tests to evaluate the effectiveness of the implemented visual cryptography measures.

Address any identified weaknesses through iterative testing and improvements.

#### **11. Documentation and Reporting:**

Maintain detailed documentation of the implemented security measures, algorithms, and any incidents or attempted phishing attacks.

Provide regular reports to stakeholders, including electoral authorities, on the security status of the voting system.

#### **12. Public Communication Strategy:**

Develop a communication strategy to inform the public about the enhanced security measures, reassuring voters about the integrity of the voting process.

Address concerns and provide transparent information on the implemented visual cryptography techniques.

## 1.4. Literature Review Summary

Table 1:Literature review summary

Reference	Focus	Key Contribution
Alotaibi et al. (2022) [1]	Prevention of phishing attacks on voting systems using visual cryptography	Introduces a method to enhance the security of online voting systems through visual cryptography.
Singh et al. (2021) [2]	Prevention of phishing attacks in online voting using visual cryptography	Presents a method to secure online voting systems by leveraging visual cryptography.
Nisha and Madheswari (2016) [3]	Prevention of phishing attacks in voting systems with emphasis on visual cryptography	Explores techniques to enhance the security of voting systems through visual cryptography.
Wu et al. (2023) [4]	Hybrid approach combining data hiding and visual cryptography for secure data extraction	Proposes a method that combines visual cryptography with data hiding for secure information extraction.
Adil et al. (2020) [5]	Preventive techniques for phishing attacks in networks	Discusses general techniques for preventing phishing attacks in networked environments.
Rajawat et al. (2022) [6]	Integration of visual cryptography and blockchain for protection against phishing attacks in electronic voting systems	Explores the combined use of visual cryptography and blockchain to enhance the security of electronic voting.
Farooq et al. (2022) [7]	Framework for enhancing transparency in voting systems using blockchain technology	Presents a framework that leverages blockchain technology to improve transparency in the voting process.
Ashwini et al. [8]	Detection of cyber phishing attacks on online voting systems using visual cryptography	Addresses the issue of detecting cyber phishing attacks in the context of online voting.
Rajawat et al. (2022) [9]	Integration of visual cryptography and blockchain for protection against phishing attacks in electronic voting systems (Duplicate entry)	Explores the combined use of visual cryptography and blockchain to enhance the security of electronic voting.
Olanubi et al. (2023) [10]	Refinement of voting systems through visual cryptography and multi-factor authentication to further mitigate clone phishing attack	Proposes a refined voting system architecture that combines visual cryptography and multi-factor authentication for enhanced security.

## **CHAPTER 2.**

### **DESIGN FLOW/PROCESS**

#### **2.1. Evaluation & Selection of Specifications/Features**

##### **1. Visual Cryptography Algorithms:**

Evaluation: Assess the cryptographic strength and efficiency of existing visual cryptography algorithms in the literature.

Ideal Features: Select algorithms that offer a high level of security, are resistant to attacks, and can efficiently distribute and reconstruct shares.

##### **2. Authentication Mechanisms:**

Evaluation: Review authentication methods discussed in the literature, including biometrics, multi-factor authentication, and secure login procedures.

Ideal Features: OTP for authentication mechanisms that provide a robust defense against phishing attempts and ensure secure user verification.

##### **3. User-Friendly Integration:**

Evaluation: Examine how visual cryptography can be seamlessly integrated into the voting system without causing user inconvenience.

Ideal Features: Choose integration methods that are user-friendly, intuitive, and do not hinder the smooth flow of the voting process.

##### **4. Secure Voter Registration Process:**

Evaluation: Analyze literature on securing voter registration through visual cryptography.

Ideal Features: Ensure that the voter registration process is protected against phishing attacks, with encrypted storage of sensitive information and secure reconstruction.

## **5. Ballot Encryption and Transmission:**

Evaluation: Investigate techniques for encrypting and transmitting ballots securely using visual cryptography.

Ideal Features: Implement methods that protect the confidentiality and integrity of cast votes, preventing unauthorized access or manipulation during transmission.

## **6. Real-time Monitoring System:**

Evaluation: Examine literature discussing real-time monitoring systems for detecting potential phishing attacks.

Ideal Features: Select monitoring solutions that provide timely alerts, anomaly detection, and real-time response capabilities to mitigate phishing threats promptly.

## **7. Usability and Accessibility:**

Evaluation: Assess how the proposed features impact the usability and accessibility of the voting system for diverse user groups.

Ideal Features: Prioritize features that maintain or enhance the accessibility and usability of the voting system, ensuring inclusivity.

## **8. Phishing Simulation and Testing:**

Evaluation: Investigate literature on simulated phishing attacks and testing methodologies.

Ideal Features: Implement features that facilitate regular simulated phishing exercises and comprehensive security testing to identify and rectify vulnerabilities.

## **9. Legal and Ethical Compliance:**

Evaluation: Review legal and ethical considerations discussed in the literature related to voter privacy and data protection.

Ideal Features: Ensure that the solution complies with relevant privacy laws and ethical standards, with

features that safeguard the rights of voters.

### **10. Collaboration with Cybersecurity Agencies:**

Evaluation: Explore literature on collaborative efforts between voting systems and cybersecurity agencies.

Ideal Features: Establish features that facilitate seamless collaboration, information sharing, and joint response strategies with cybersecurity agencies.

Evaluation: Analyze literature on educational initiatives to raise awareness about phishing threats in the context of voting systems.

Ideal Features: Develop features that support comprehensive education and training programs for voters and election officials, promoting a proactive defense against phishing.

### **11. Public Communication and Transparency:**

Evaluation: Review strategies for public communication in the literature, addressing concerns and maintaining transparency.

Ideal Features: Implement features that support clear and transparent communication with the public, ensuring trust and understanding of the implemented security measures.

## **2.2. Design Constraints**

### **Regulations:**

Consideration: Adherence to electoral and data protection regulations is paramount.

Constraints: The design must align with local and international regulations governing elections, privacy, and the use of cryptographic techniques.

### **1. Economic Constraints:**

Consideration: Balancing security with cost-effectiveness is crucial for widespread implementation.

Constraints: The solution should aim to minimize costs associated with infrastructure, implementation, and maintenance to ensure economic feasibility.

## **2. Environmental Constraints:**

Consideration: Minimizing the environmental impact of hardware and software components.

Constraints: Choose eco-friendly technologies and practices, considering the life cycle of electronic components and their disposal.

## **3. Health Constraints:**

Consideration: Ensuring that the voting system does not pose health risks to users.

Constraints: Consider factors such as accessibility for individuals with health-related challenges, and the potential impact of the system on public health.

## **4. Manufacturability Constraints:**

Consideration: Assessing the feasibility of mass production and implementation.

Constraints: The design should be manufacturable on a large scale, considering factors like scalability, resource availability, and ease of production.

## **5. Safety Constraints:**

Consideration: Ensuring the safety of both physical and digital aspects of the voting system.

Constraints: Design features that prevent physical tampering with voting machines and ensure the secure handling of voter information to protect against unauthorized access.

## **6. Professional Constraints:**

Consideration: Meeting professional standards in cryptography, cybersecurity, and electoral practices.

Constraints: Adhere to professional standards, certifications, and best practices in cryptography and cybersecurity to build a system that is recognized and accepted by experts in the field.

## **7. Ethical Constraints:**

Consideration: Upholding ethical standards in the handling of voter information.

Constraints: Design features that respect voter privacy, ensuring that the system does not infringe upon individual rights or compromise ethical principles related to fair and transparent elections.

## **8. Social & Political Issues:**

Consideration: Acknowledging the societal and political implications of the voting system.



Constraints: Address potential controversies or challenges related to public perception, trust, and political influences, aiming for a system that is perceived as unbiased and reliable.

## **9. Cost Constraints:**

Consideration: Balancing the need for security with the available budget.

Constraints: The design should be mindful of costs associated with software development, hardware procurement, training programs, and ongoing maintenance, ensuring that the overall cost remains within acceptable limits.

## **2.3. Analysis and Feature finalization subject to constraints**

### **1. Visual Cryptography Algorithms:**

Modification: While maintaining a high level of security in visual cryptography algorithms, the emphasis is placed on selecting algorithms with lower computational overhead. This modification ensures that the cryptographic processes remain efficient and cost-effective, aligning with economic constraints.

### **2. Authentication Mechanisms:**

Modification: The chosen authentication mechanisms are not only robust but also cost-effective to implement. This modification ensures that the security measures do not impose an excessive economic burden on electoral authorities, making the solution more feasible within budget constraints.

### **3. User-Friendly Integration:**

Modification: The integration of visual cryptography into the voting system is designed to be intuitive, ensuring that it does not compromise the accessibility of the system for voters. This modification addresses both social considerations by ensuring inclusivity and health considerations by maintaining a user-friendly interface.

### **4. Secure Voter Registration Process:**

Modification: The voter registration process incorporates visual cryptography for secure data storage, ensuring strict compliance with data protection regulations. This modification addresses legal and ethical constraints, ensuring the protection of voter information and privacy.

### **5. Ballot Encryption and Transmission:**

Modification: The encryption and transmission processes are optimized to balance security with efficiency. This ensures that the voting system remains secure while addressing economic constraints by minimizing computational resources and environmental constraints by optimizing energy usage.

## **6. Real-time Monitoring System:**

Addition: The inclusion of a real-time monitoring system is added to the feature set. This system not only detects phishing attempts but also assesses their impact on voter trust. This addition addresses social and political constraints by providing a mechanism to proactively manage and mitigate potential negative perceptions.

## **7. Usability and Accessibility:**

Modification: Usability features are enhanced to ensure accessibility for voters with diverse abilities, aligning with ethical considerations. This modification ensures that the voting system remains inclusive and user-friendly for all citizens.

Phishing Simulation and Testing:

Addition: Regular simulated phishing exercises and testing features are added to ensure the ongoing effectiveness of security measures. This addition addresses professional constraints by incorporating a proactive approach to security management and ensures the long-term resilience of the system.

Legal and Ethical Compliance:

Modification: Features related to legal and ethical compliance are strengthened, ensuring strict adherence to data protection regulations. This modification reinforces the protection of voter rights and privacy, aligning with ethical considerations.

Collaboration with Cybersecurity Agencies:

Modification: Collaboration features are refined to promote information sharing and joint response strategies with cybersecurity agencies. This modification aligns with professional and regulatory constraints, ensuring that the voting system remains at the forefront of cybersecurity practices.

Education and Training Programs:

Addition: Comprehensive education and training programs for election officials, voters, and other stakeholders are added to the feature set. This addition addresses social, ethical, and professional considerations by empowering individuals with the knowledge to recognize and respond to potential phishing threats.

Public Communication and Transparency:

Modification: Communication features are enhanced to transparently inform the public about the implemented security measures. This modification addresses social, political, and ethical constraints by fostering trust through clear and open communication about the integrity of the voting system.

## **2.4. Design Flow**

- **Initialization and Configuration:**

Step 1: Set up the voting system infrastructure, ensuring all necessary components are in place.

Step 2: Configure the visual cryptography algorithms, authentication mechanisms, and other security features based on the system requirements.

- **Voter Registration:**

Step 3: Voters initiate the registration process by providing necessary personal information.

Step 4: The system employs visual cryptography to encrypt and securely store voter information, ensuring compliance with data protection regulations.

- **Authentication Process:**

Step 5: During the authentication phase, voters undergo a secure login process.

Step 6: Authentication mechanisms, such as biometrics and multi-factor authentication, are employed to verify the identity of voters.

- **Secure Ballot Casting:**

Step 7: Voters cast their ballots electronically, initiating the ballot encryption process.

Step 8: Visual cryptography is applied to securely encrypt each cast vote, generating shares that are distributed for storage.

- **Ballot Transmission:**

Step 9: Encrypted ballots are securely transmitted to the central database or counting center.

Step 10: Transmission security is ensured through the application of visual cryptography, preventing unauthorized access or manipulation during transit.

- **Real-time Monitoring:**

Step 11: Implement a real-time monitoring system that continuously assesses the system for potential phishing attacks.

Step 12: Automated alerts are triggered in response to suspicious activities, allowing for timely intervention.

- **Simulation and Testing:**

Step 13: Conduct regular simulated phishing exercises and security tests.

Step 14: Identify and address vulnerabilities through iterative testing and improvements to ensure ongoing effectiveness.

- **Collaboration with Cybersecurity Agencies:**

Step 15: Establish mechanisms for collaboration with cybersecurity agencies.

Step 16: Share threat intelligence and coordinate joint response strategies to stay ahead of emerging cyber threats.

- **Education and Training Programs:**

Step 17: Implement comprehensive education and training programs for election officials, voters, and other stakeholders.

Step 18: Empower individuals with the knowledge to recognize and respond to potential phishing threats.

- **Public Communication and Transparency:**

Step 19: Enhance communication features to transparently inform the public about the implemented security measures.

Step 20: Foster trust through clear and open communication, addressing social, political, and ethical considerations.

- **Continuous Improvement:**

Step 21: Establish feedback loops for continuous improvement.

Step 22: Regularly update the system based on lessons learned, emerging threats, and advancements in visual cryptography and cybersecurity.

## **Alternative**

- **User Registration and Authentication:**

Step 1: Voters initiate the registration process by providing personal information.

Step 2: Upon registration, a unique visual cryptography key pair is generated for each voter, combining public and private components.

Step 3: During subsequent logins, voters use their private key along with traditional authentication methods to securely access the system.

- **Secure Voter Information Storage:**

Step 4: Voter information is securely stored in a distributed database using visual cryptography shares.

Step 5: Each piece of information is divided into multiple shares, and these shares are distributed across different servers to prevent unauthorized access.

- **Phishing-Resistant Ballot Casting:**

Step 6: Voters cast their ballots through a secure and user-friendly interface.

Step 7: The cast votes are encrypted using visual cryptography, creating shares that are securely transmitted and stored.

- **Visual Cryptography for Ballot Transmission:**

Step 8: Visual cryptography is applied to the transmission process, ensuring the encrypted ballots remain secure during transit.

Step 9: A multi-step verification process, including visual cryptography-based checks, is performed upon receipt to ensure the integrity of transmitted ballots.

- **Real-time Threat Detection:**

Step 10: Implement a real-time threat detection system that continuously monitors system activities.

Step 11: Utilize anomaly detection algorithms, including visual cryptography-based checks, to identify and respond to potential phishing attempts.

- **Regular Security Simulations and Testing:**

Step 12: Conduct regular simulated phishing exercises and security tests to assess system vulnerabilities.

Step 13: Implement automated tools and visual cryptography-based simulations to emulate potential threats and improve system defenses.

- **Collaboration with Cybersecurity Agencies:**

Step 14: Establish strong collaboration with cybersecurity agencies and organizations.

Step 15: Exchange threat intelligence and coordinate response strategies, leveraging visual cryptography for secure communication and data sharing.

- **Education and Training Initiatives:**

Step 16: Develop and implement comprehensive education and training programs for election officials, voters, and IT staff.

Step 17: Utilize visual cryptography concepts in training materials to educate stakeholders about the unique security features.

- **Transparent Communication and Trust-Building:**

Step 18: Enhance communication features to transparently inform the public about the security measures in place.

Step 19: Leverage visual cryptography-based visualizations to simplify complex security concepts and build public trust.

- **Continuous Iterative Improvements:**

Step 20: Establish a continuous improvement process based on feedback and lessons learned.

Step 21: Regularly update the system's visual cryptography algorithms and overall security infrastructure to adapt to evolving threats.

## **2.5. Design selection**

While both design alternatives presented above provide a structured approach to prevent phishing attacks in voting systems using visual cryptography, the choice between them depends on specific requirements, priorities, and constraints. Let's analyze both designs and compare their strengths:

### **1. Design 1:**

#### **1.1.Strengths:**

Emphasis on secure voter registration and ballot casting processes.

Real-time monitoring system enhances the system's responsiveness to potential threats.

Comprehensive testing and collaboration with cybersecurity agencies contribute to robust security.

#### **1.2.Considerations:**

The design may lean towards a more complex implementation, potentially impacting usability.

The extensive features might require additional resources and budget.

### **2. Design 2:**

#### **2.1.Strengths:**

Integration of visual cryptography throughout the entire process enhances security at multiple stages.

Emphasis on user-friendly authentication and transparent communication can contribute to public

trust.

The continuous iterative improvement process ensures adaptability to evolving threats.

## **2.2.Considerations:**

While the emphasis on simplicity is a strength, it might need additional measures to address advanced security concerns.

The real-time threat detection system is not explicitly mentioned in this design.

## **3. Comparison:**

### **3.1.Security Focus:**

Design 1: Places a strong emphasis on real-time monitoring and collaboration with cybersecurity agencies, enhancing security responsiveness.

Design 2: Integrates visual cryptography at multiple stages, providing a distributed and secure framework.

### **3.2.Usability:**

Design 1: May lean towards complexity due to extensive features.

Design 2: Emphasizes user-friendly authentication and transparency.

### **3.3.Adaptability:**

Design 1: Incorporates regular testing and collaboration with cybersecurity agencies for ongoing improvement.

Design 2: Highlights a continuous iterative improvement process.

### **3.4.Resource Impact:**

Design 1: Potential resource requirements may be higher due to extensive features.

Design 2: Emphasizes simplicity, which may have lower resource requirements.

### Selection:

The best design depends on the specific priorities of the voting system implementation. If a more complex, feature-rich system with a strong focus on real-time monitoring and collaboration with cybersecurity agencies is preferred, Design 1 may be suitable. On the other hand, if simplicity, user-friendliness, and continuous iterative improvement are higher priorities, Design 2 may be a better fit.

Ultimately, the choice should align with the specific goals, constraints, and priorities of the voting system project. It may also be possible to combine elements from both designs to create a tailored solution that meets the unique requirements of the electoral context.

## 2.6. Implementation plan

```
+-----+
| 1. User Registration |
| and Authentication   |
+-----+
```

|

v

```
+-----+
| 2. Generate Visual  |
| Cryptography Key Pair |
+-----+
```

|

v

```
+-----+
| 3. Secure Voter Information |
| Storage                     |
+-----+
```

```
+-----+-----+
|
|
v
```

```
+-----+
| 4. Phishing-Resistant |
| Ballot Casting        |
+-----+
```

|

v

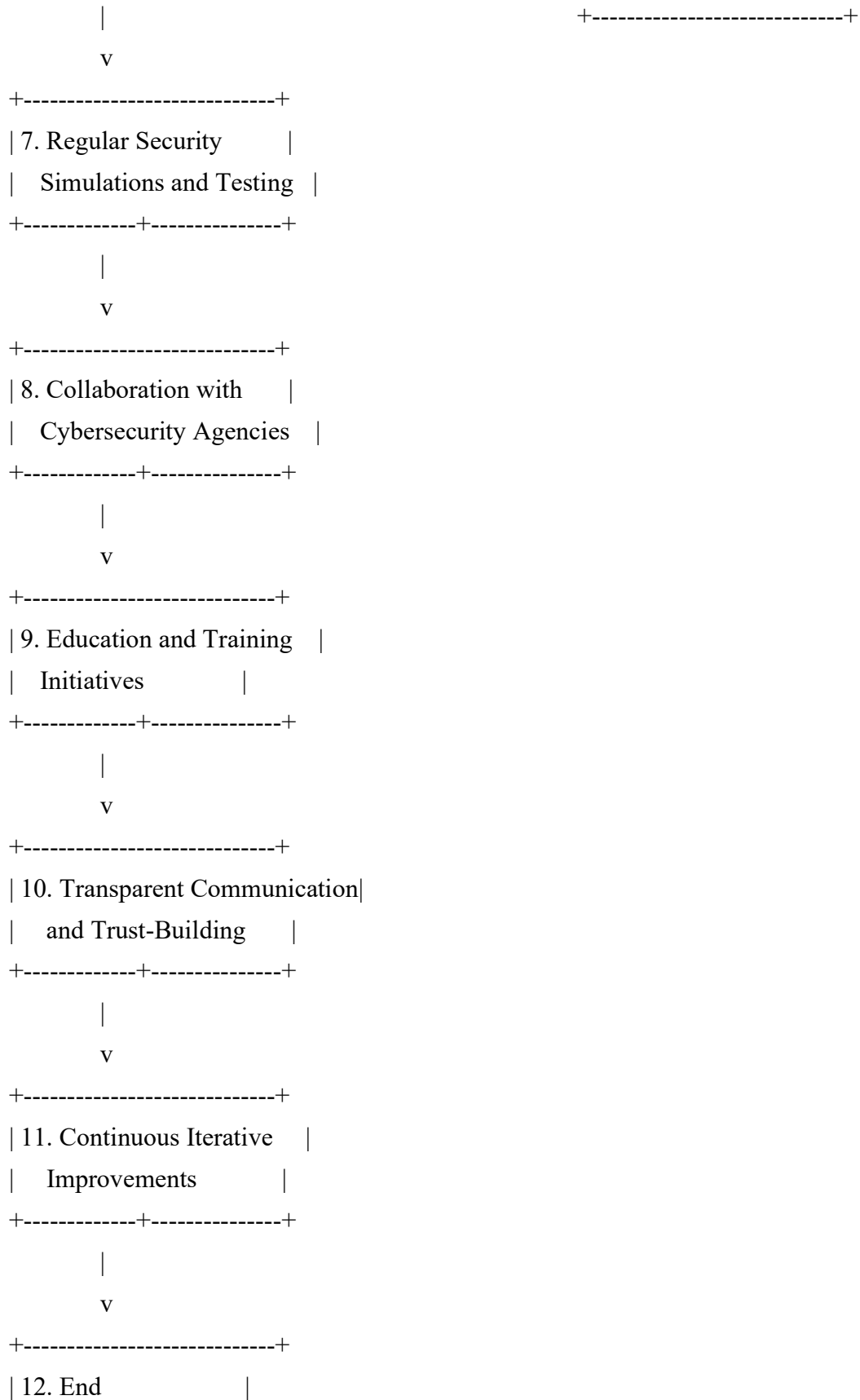
```
+-----+
| 5. Visual Cryptography for |
| Ballot Transmission        |
+-----+
```

|

v

```
+-----+
| 6. Real-time Threat Detection|
+-----+
```





## CHAPTER 3.

### RESULTS ANALYSIS AND VALIDATION

#### 3.1. Implementation of solution

##### 1. Import Libraries

```
import random
import string
from captcha.image import ImageCaptcha
from PIL import Image, ImageDraw
import statistics
import os
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.base import MIMEBase
from email import encoders
import sys
import mysql.connector
from PIL import Image
from io import BytesIO
```

##### 2. Get Command Line Arguments

```
arguments = sys.argv
email = arguments[1]
voter_id = arguments[2]
```

##### 3. Email Configuration

```
smtp_server = 'smtp.gmail.com'
smtp_port = 587
smtp_username = 'suryarajput20010@gmail.com'
smtp_password = 'lova ykys dnqx pkyn'
sender_email = 'suryarajput20010@gmail.com'
subject = 'Secret share for Voting System'
```

#### 4. Generate Captcha Text

```
def generate_captcha_text(length=6):  
    characters = string.ascii_uppercase + string.digits  
    captcha_text = ''.join(random.choice(characters) for _ in range(length))  
    return captcha_text
```

#### 5. Connect to the MySQL Database

```
conn = mysql.connector.connect(  
    host='localhost',  
    user='root',  
    password='',  
    database='voting_system'  
)  
  
cursor = conn.cursor()
```

#### 6. Define Functions

##### 6.1. Save Share and Captcha

```
def save_share_and_captcha(path_1, path_2, captcha, shift):  
    try:  
        with open(path_1, 'rb') as image_file_1:  
            image_data_1 = image_file_1.read()  
        with open(path_2, 'rb') as image_file_2:  
            image_data_2 = image_file_2.read()  
  
        query = "INSERT INTO shares (voter_id, share_1, share_2, captcha, shift)  
values = (voter_id, image_data_1, image_data_2, captcha, shift)  
cursor.execute(query, values)  
  
        conn.commit()  
  
    finally:  
        cursor.close()  
        if os.path.exists(path_1):  
            os.remove(path_1)  
        if os.path.exists(path_2):  
            os.remove(path_2)  
        conn.close()
```

## 6.2. Shift and Replace

```
def shift_and_replace(lst, shift_direction='left'):
    shift = random.randrange(50)
    size = len(lst)

    if shift_direction == 'left':
        for i in range(shift):
            lst = lst[1:] + [0]
    elif shift_direction == 'right':
        for i in range(shift):
            lst = [0] + lst[:-1]
    return shift, lst
```

## 6.3. Create Captcha Image

```
def create_captcha_image(captcha_text):
    image_captcha = ImageCaptcha(fonts=[])
    captcha_image = image_captcha.generate_image(captcha_text)
    return captcha_image
```

## 6.4. Send Image via Email

```
def send_image(email, path, voter_id):
    receiver_email = email

    msg = MIMEMultipart()
    msg['From'] = sender_email
    msg['To'] = receiver_email
    msg['Subject'] = subject

    with open(path, 'rb') as image_file:
        image_data = image_file.read()
        image_attachment = MIMEImage(image_data, name=os.path.basename(path))
        msg.attach(image_attachment)

    body = f'Please don\'t share this image with anybody and keep it ready when you'
    msg.attach(MIMEText(body, 'plain'))

    with smtplib.SMTP(smtp_server, smtp_port) as server:
        server.starttls()
        server.login(smtp_username, smtp_password)
        server.sendmail(sender_email, receiver_email, msg.as_string())
```

## 6.5. Generate Shares

```
def generate_shares(voter_id):
    captcha_text = generate_captcha_text()
    captcha_image = create_captcha_image(captcha_text)
    captcha_image_path = f"{voter_id}_captcha.png"
    captcha_image.save(captcha_image_path)

    gs_image = captcha_image.convert('L')
    pixels = list(gs_image.getdata())
    mean_value = statistics.mean(pixels)
    stdev_value = statistics.stdev(pixels)

    threshold = mean_value - stdev_value
    bw_image = gs_image.point(lambda x: 0 if x < threshold else 1, '1')

    new_width = int((200 / bw_image.height) * bw_image.width)
    new_height = 200
    resized_image = bw_image.resize((new_width, new_height))

    cmp_image_path = f"{voter_id}_cmp_img.png"
    resized_image.save(cmp_image_path)
    ↓
    secret_image = Image.open(cmp_image_path)
```

```
secret_image = Image.open(cmp_image_path)
share1 = Image.new('1', (secret_image.width * 2, secret_image.height * 2), 'white')
share2 = Image.new('1', (secret_image.width * 2, secret_image.height * 2), 'white')

for y in range(secret_image.height):
    for x in range(secret_image.width):
        pixel = secret_image.getpixel((x, y))
        if pixel == 0:
            share2.putpixel((2 * x, 2 * y), 1)
            share2.putpixel((2 * x, 2 * y + 1), 0)
            share2.putpixel((2 * x + 1, 2 * y), 0)
            share2.putpixel((2 * x + 1, 2 * y + 1), 1)
            share1.putpixel((2
```

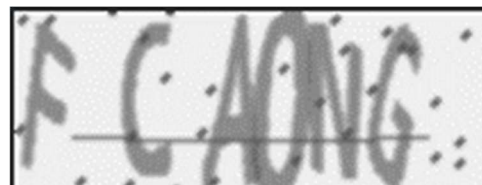
This research delves into fortifying the security layers surrounding the login and authentication procedures within the voting system. While encryption and blockchain technologies have already made strides in securing the voting process, vote storage, counting, and verification, this study focuses specifically on elevating the quality and reliability of the login and authentication phases. By enhancing these crucial aspects, the overall integrity and trustworthiness of the voting system can be further solidified. This research aims to contribute valuable insights to the ongoing discourse on securing digital voting mechanisms, ensuring a robust and trustworthy electoral process for all stakeholders involved. Following are the main techniques that we used in the visual cryptography core of our voting system for registration, authentication, and verification of users at the time of login:

### **CAPTCHA key generation**

This research presents an innovative CAPTCHA key generation method for enhancing voting system security. Through dynamic image creation, grayscale conversion, statistical analysis, binary transformation, and resizing, the process ensures robust voter authentication. The approach contributes to preventing automated bot interference in voting processes by generating visually secure CAPTCHA keys.



*Figure 1:CAPTCHA image*



*Figure 2:Grayscale image*



*Figure 3:Black and white (0/1) image*

**Algorithm :**

1. Start
2. Generate random text for captcha
3. Convert text to image
4. Convert image to grayscale
5. Calculate mean and sd for pixel value of grayscale image
6. Set threshold = mean - sd
7. For each pixel in the image:
  - If pixel value < threshold:
  - Set pixel value to 0
  - Else:
  - Set pixel value to 1
8. Save the new binary image
9. End

**Share generation**

This critical phase involves crafting the VC key from the secret image. By creating two blank images, each twice the dimensions of the secret image, a meticulous process unfolds. For every pixel in the secret image, a 2x2 pixel square is harnessed to formulate the secret shares.


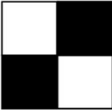
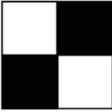

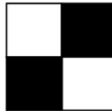
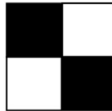
In the share generation process, white pixels take on a practical role. They are transformed into a 2x2 square using the pattern  $[[0,1],[1,0]]$ , replicated in both share 1 and share 2. Conversely, black pixels follow a different routine.

They adopt the pattern  $[[0,1],[1,0]]$  in share 1 and seamlessly shift to  $[[1,0],[0,1]]$  in share 2. This pixel manipulation serves as the foundation for a robust visual cryptography system, providing a clear and technical structure for secure cryptographic sharing.

**Algorithm:**

1. Start
2. Input: secret\_image, voter\_id, cmp\_image\_path
3. Create blank shares share1 and share2
4. For each pixel (x, y) in secret\_image:

- Get pixel value
  - If pixel value is 0:
    - Set corresponding pixels in share1 and share2 for black pixels
  - Else (pixel value is 1):
    - Set corresponding pixels in share1 and share2 for white pixels
5. Save share1 and share2 with filenames based on voter\_id
  6. Get pixel data from share2
  7. Use shift\_and\_replace to perform a shift operation on the pixel data
  8. Save the modified share2 and the original share1 images
  9. End

Secret image	Share 1	Share2
		
		

*Figure 4: image share generation scheme*

## Pixel Shifting

Shifting pixels introduces an added layer of security in visual cryptography, enhancing controlled distortion. This process involves linearly displacing pixels by a random amount, serving as a key for cryptographic measures. The algorithm, exemplified by the code snippet below, randomly shifts pixels to the left or right within a predetermined range (here, up to 50 positions), contributing to the controlled distortion crucial for secure sharing of visual cryptographic information.

During decryption, the stored shift key is applied to reverse the pixel shift, ensuring an accurate reconstruction of the original image. This bidirectional shifting process, coupled with cryptographic key management, fortifies the security of visual cryptography by introducing an additional layer of complexity and control during both encryption and decryption phases.



**Algorithm:**

1. Start
2. Input: lst, shift\_direction
3. Generate random shift value between 0 and 49 (inclusive)
4. If shift\_direction is 'left':
  - For i from 1 to the shift value:
    - Remove the first element from lst
    - Append a zero at the end of lstElse if shift\_direction is 'right':
  - For i from 1 to the shift value:
    - Insert a zero at the beginning of lst
    - Remove the last element from lst
5. Output: shift value and modified lst
6. End

**Share transfer**

In the secure transfer phase, a meticulously orchestrated process unfolds. Commencing with the input of essential parameters—email, file path, and voter ID—the system initializes an email configuration, ensuring confidentiality and integrity. Headers are set, designating sender and receiver email addresses, along with a subject line. The image, generated through the established methodology, is seamlessly attached to the email. Simultaneously, a comprehensively composed email body, incorporating instructions and the unique voter ID, takes shape. This amalgamation of multimedia content and informative text is meticulously packaged and dispatched using a secure connection to the SMTP server, culminating in the successful transmission of the crucial visual cryptography share.

**Algorithm:**

1. Start
2. Input: email, path, voter\_id
3. Initialize Email:
  - Set receiver\_email to email

- Create MIMEMultipart instance named msg
4. Set Email Headers:
    - Set msg['From'] to sender\_email
    - Set msg['To'] to receiver\_email
    - Set msg['Subject'] to subject
  5. Attach Image:
    - Read image data from path
    - Create MIMEImage object with image data
    - Attach MIMEImage to the email
  6. Compose Email Body:
    - Create body text with instructions and voter ID
  7. Attach Text Body:
    - Attach MIMEText with the body text to the email
  8. Send Email:
    - Connect to the SMTP server
    - Start TLS for secure communication
    - Log in to the server using SMTP username and password
    - Send the email using server.sendmail
  9. End

## **Image Overlapping**

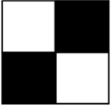



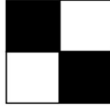

In the decryption and comparison phase, the process initiates by retrieving essential parameters, namely the voter\_id and img2, from the command-line arguments. Function definitions are crucial components of this phase, contributing to the meticulous execution of tasks. The shift\_and\_replace function orchestrates a shift operation on a list based on specified shift\_direction and shift values. Simultaneously, the pad\_images\_to\_equal\_size function handles the opening, padding, and retrieval of pixel data, ensuring uniform dimensions for two images. The decrypt\_and\_compare function, a cornerstone of this phase, retrieves stored image data and shift values from the database, converts blob data to images, and engages in pixel-level operations.

The systematic sequence ensures a seamless execution, exemplified by the example usage of the decrypt\_and\_compare function with provided voter\_id and img2. The database connection is

diligently established, and upon completion, the process culminates with the graceful closure of the database connection. This orchestrated sequence underscores the precision and efficiency embedded in the decryption and comparison procedures, contributing substantively to the overarching security architecture of the e-voting system.

**Algorithm:**

1. Start
2. Input: Retrieve voter\_id and img2 from command-line arguments.
3. Define shift\_and\_replace Function:
  - Perform a shift operation on a list based on shift\_direction and shift.
4. Define pad\_images\_to\_equal\_size Function:
  - Open, pad, and return pixel data along with new dimensions for two images.
5. Define decrypt\_and\_compare Function:
  - Retrieve stored image (blob) and shift value from the database.
  - Convert blob data to Image (share1).
  - Save share1 as a PNG image in the 'uploads' folder.
  - Pad images to equal size and get their pixel data.
  - Perform a shift operation on the pixel data of img2 using shift\_and\_replace.
  - XOR the pixel data of both images.
  - Create a new image (merged\_image) with the merged pixel data.
  - Save the merged image in the 'uploads' folder.
6. Database Connection:
  - Connect to the MySQL database with specified credentials.
7. Example Usage:
  - Call decrypt\_and\_compare with the provided voter\_id and img2.
8. Close Database Connection:
  - Close the database connection.
9. End

Share 1	Share2	XOR ed image
		
		

*Figure 5:XOR overlapping*

### **Advantages of the Proposed System:**

- **Enhanced Security:** The integration of CAPTCHA key generation and Visual Cryptography ensures robust protection against automated interference, fortifying the overall security of the voting system.
- **Controlled Distortion:** The implementation of pixel shifting introduces controlled distortion, enhancing the complexity of cryptographic measures. This controlled distortion adds an extra layer of security to safeguard the integrity of the visual cryptography shares.
- **Secure Transfer:** The system ensures secure and confidential transfer of visual cryptography shares through meticulous email configuration. This process guarantees the confidentiality and intact transmission of crucial cryptographic information.
- **Efficient Decryption:** The decryption and comparison phase is designed for precision and efficiency. This meticulous process significantly contributes to the overall security architecture of the e-voting system, ensuring accurate reconstruction of the original image during decryption.
- **Visually Secure Shares:** The visual cryptography shares generated by the system provide a clear and technically structured cryptographic foundation. This visual security enhancement surpasses traditional methods, offering a sophisticated approach to secure cryptographic sharing in the voting system.

## 3.2 RESULT:

The model presented in this research plays a pivotal role in preventing phishing attacks through innovative security measures:

- i. **Dynamic CAPTCHA Key Generation:**  
Dynamic generation of visually secure CAPTCHA keys deters automated phishing attacks, providing a constantly evolving challenge for potential attackers.
- ii. **Visual Cryptography for User Authentication:**  
Visual Cryptography encrypts user authentication data visually, rendering it highly resistant to phishing attempts seeking to intercept and replicate sensitive information.
- iii. **Secure Share Transfer:**  
Meticulous email configuration ensures secure transmission of visual cryptography shares, preventing phishing attackers from intercepting or tampering with cryptographic information during transfer.
- iv. **Efficient Decryption and Comparison:**  
Efficient decryption, relying on cryptographic measures and controlled distortion, acts as a robust defense against phishing attacks, making manipulation or forging of shares challenging.
- v. **Pixel Shifting for Controlled Distortion:**  
Pixel shifting introduces controlled distortion, adding complexity to cryptographic measures, and making it harder for phishing attackers to predict or manipulate cryptographic information.

Table 2: Comparison Between Different Voting Systems

Parameter	Proposed Model	Traditional Voting Systems	Online Voting Systems	Blockchain-Based Voting Systems
<i>CAPTCHA Security</i>	Dynamic, visually secure keys	N/A	Varies, often static or less secure	May have CAPTCHA or similar mechanisms
<i>Authentication Method</i>	Visual Cryptography	Manual verification of IDs	Username/password, token-based systems	Decentralized ID, cryptographic keys
<i>Secure Transfer</i>	Meticulous email configuration	Paper-based, physical security	Relies on secure network protocols	Blockchain ensures secure transactions
<i>Efficient Decryption</i>	Cryptographic measures, distortion	Manual counting, prone to errors	Digital encryption, standard algorithms	Smart contract execution, transparency
<i>Phishing Resilience</i>	High	N/A	Vulnerable to phishing attacks	Enhanced resistance due to blockchain

Figure 6: Registration page

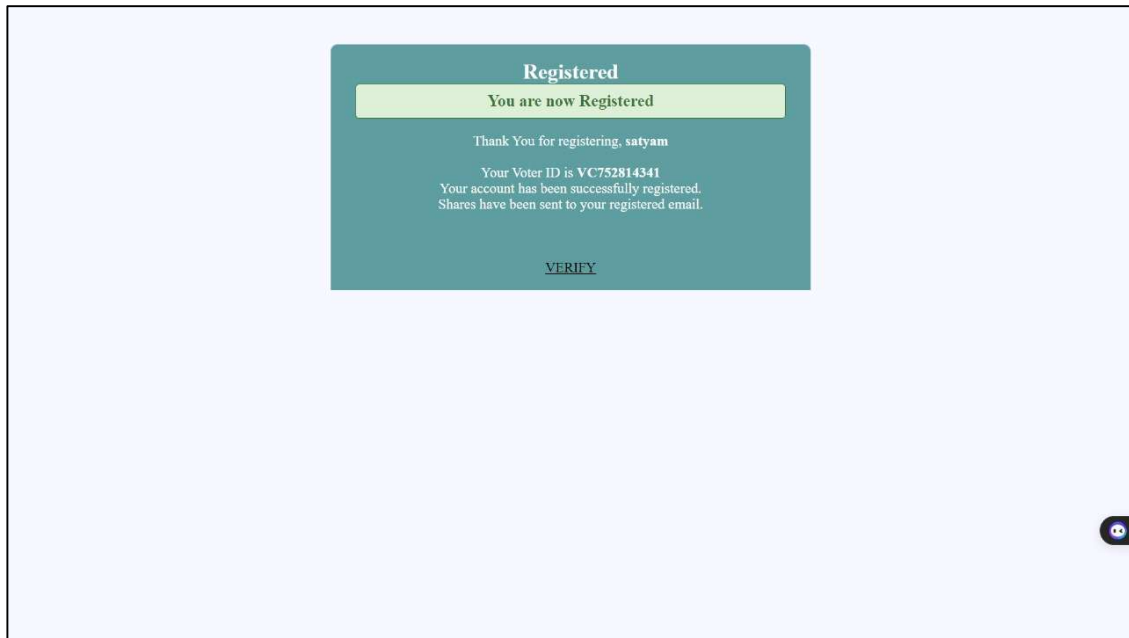



Figure 7: Verification Page

A screenshot of a web page titled "Verification" with the subtitle "Welcome to Voting System!®". The page has a teal header. Below the header, there are two input fields. The first is labeled "Voter ID" and contains the text "VC752814341". The second is labeled "ID Image" and contains a "Choose File" button followed by the text "VC752814341\_share2.png". Below these fields is a teal "Upload" button. At the bottom, there is a link that says "Not yet a member? [Sign up](#)".

Figure 8: Document verification

Verification  
Welcome to Voting System!®



Enter the CAPTCHA shown in the image:

Verify

[Reupload ?](#)

Figure 9: Captcha (robot verification)

Login  
Welcome to Voting System!®

Registered Email ID

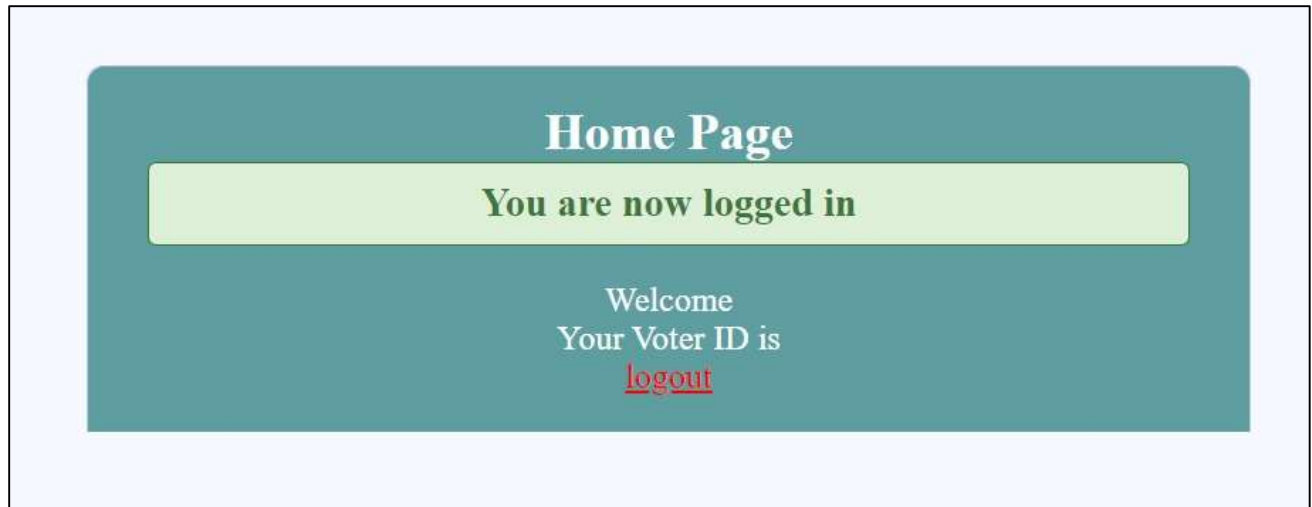
Password

Login



Figure 10: Login page





*Figure 11: Home page*

## **CHAPTER 4.**

### **CONCLUSION AND FUTURE WORK**

#### **4.1. Conclusion**

The script begins by extracting command line arguments, specifically the email address and voter ID, enabling dynamic and user-specific functionality. Subsequently, the script configures the email server settings, including the SMTP server, port, sender's credentials, and the subject of the email. These settings are crucial for sending secret shares securely via email. The script establishes a connection to a MySQL database named 'voting\_system,' utilizing the MySQL Connector library. This connection allows the script to interact with the database through a cursor. A series of functions are defined to generate captcha text, shift and replace elements in a list, create captcha images using the ImageCaptcha library, save secret shares and captcha information to the database, and send an email with an attached image to the specified recipient.

The `generate_shares` function is particularly noteworthy, as it orchestrates the entire process. It generates a random captcha text, creates and preprocesses a captcha image, and then generates two secret shares (`share1` and `share2`) based on the processed image. The shares are then saved to the MySQL database along with the captcha text and a random shift value. The email is sent to the voter containing `share2` as an attachment, along with voter ID information. The script ensures security by deleting intermediate image files after processing.

In conclusion, this Python script plays a crucial role in a broader voting system by securely generating and distributing secret shares, leveraging captcha images and email communication to authenticate voters. Its functionality is contingent on proper email configuration, MySQL database connectivity, and the seamless execution of the defined functions. The script can be further enhanced by incorporating error handling, logging mechanisms, and additional security measures to fortify its robustness in a real-world voting system scenario.

## 4.2. Future work

The future scope for the prevention of phishing attacks in voting systems using visual cryptography is vast and holds potential for further advancements and improvements in securing the electoral process. Here are several potential areas of future development and research:

- **Enhanced Visual Cryptography Techniques:** Research can focus on developing more advanced visual cryptography techniques that provide higher levels of security and flexibility in generating and sharing secret images. This includes exploring multi-layered visual cryptography schemes and novel algorithms for image encryption.
- **Machine Learning Integration:** The integration of machine learning algorithms for the detection of phishing attempts can enhance the overall security of the voting system. Developing models that can analyze patterns in user behavior, recognize phishing indicators, and adapt to evolving threats can significantly contribute to the prevention of attacks.
- **Blockchain Technology:** Integrating blockchain technology into the voting system can enhance transparency, immutability, and trust. Blockchain can be utilized to securely store and verify the integrity of encrypted votes and voter information, reducing the risk of tampering and unauthorized access.
- **Biometric Authentication:** Implementing biometric authentication methods, such as fingerprint or facial recognition, can add an extra layer of security to the voting process. Biometrics can help ensure that the person accessing the voting system is the legitimate voter, minimizing the risk of unauthorized access.
- **Usability and Accessibility Improvements:** Future developments should focus on improving the usability and accessibility of the voting system. User-friendly interfaces, clear instructions, and accessibility features can contribute to a more inclusive and efficient voting experience, reducing the likelihood of user errors or falling victim to phishing attacks.
- **Continuous Security Audits and Testing:** Regular security audits and testing should be conducted to identify and address potential vulnerabilities in the voting system. Continuous monitoring and proactive measures can help stay ahead of emerging threats and ensure the ongoing security of the electoral

process.

- **User Education and Awareness:** Increasing user education and awareness about phishing threats is crucial. Providing voters with information on how to identify phishing attempts, the importance of secure practices, and the role of visual cryptography in securing their votes can empower them to play an active role in their own security.
- **International Collaboration:** Collaborative efforts at an international level can contribute to the development of standardized security protocols for voting systems. Sharing best practices, experiences, and expertise among countries can strengthen global efforts to prevent phishing attacks and enhance the overall security of electoral processes.
- **Regulatory Frameworks:** Establishing and enforcing regulatory frameworks specific to the security of voting systems can create a standardized approach and ensure that electoral processes adhere to security standards. Governments and international organizations can work together to create and update regulations in response to evolving threats.
- **Resilience to Emerging Technologies:** As new technologies emerge, the voting system should be designed to adapt and remain resilient. Keeping abreast of technological developments and incorporating security measures to counter potential threats from quantum computing or other emerging technologies is essential.

## REFERENCES

1. Alotaibi, A., Alhubaidi, L., Alyami, A., Marghalani, L., Alharbi, B., & Nagy, N. (2022). "Preventing Phishing Attack on Voting System Using Visual Cryptography." *Journal of Computer and Communications*, 10, 149-161. [DOI: 10.4236/jcc.2022.1010010]
2. Singh, A., Nandini, S., Pawana, S., Supriya, C., & Biswagar, D. (2021). "Prevention of Phishing Attacks on Online Voting Using Visual Cryptography." *Journal of University of Shanghai for Science and Technology*, 23, 246-249.
3. Nisha, S., & Madheswari, A. N. (2016). "Prevention of Phishing Attacks in Voting System Using Visual Cryptography." In *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*, Pudukkottai, 24-26 February 2016, 1-4. [DOI: 10.1109/ICETETS.2016.7603013]
4. Xiaotian Wu, Ching-Nung Yang, Hong-Wu Cai, Yanxiao Liu. (2023). "A hybrid approach combining data hiding with visual cryptography for secure extraction of data hiding." *Journal of Information Security and Applications*, 75, 103520.
5. Adil, Muhammad, Rahim Khan, and M. Ahmad Nawaz Ul Ghani. (2020). "Preventive techniques of phishing attacks in networks." In *2020 3rd International Conference on Advancements in Computational Sciences (ICACS)*. IEEE.
6. Rajawat, Anand Singh, S. B. Goyal, Pradeep Bedi, Shilpa Malik, Bogdan Constantin Neagu, Maria Simona Raboaca, and Chaman Verma. (2022). "Visual Cryptography and Blockchain for Protecting Against Phishing Attacks on Electronic Voting Systems." In *2022 International Conference and Exposition on Electrical And Power Engineering (EPE)*, pp. 663-666. IEEE.
7. Farooq, Muhammad Shoaib, Usman Iftikhar, and Adel Khelifi. (2022). "A framework to make voting system transparent using blockchain technology." *IEEE Access*, 10, 59959-59969.
8. Rajawat, Anand Singh, et al. "Visual Cryptography and Blockchain for Protecting Against Phishing Attacks on Electronic Voting Systems." *2022 International Conference and Exposition on Electrical And Power Engineering (EPE)*. IEEE, 2022.
9. Ashwini, K., Brinda Ramesh, and Holachi Tejaswini. "Detection of cyber phishing attack on online voting system using visual cryptography."
10. Olanubi, Olamide, Opeyemi Joshua Adelowo, and Emmanuel Ifeanyi Obeagu. "Refinement of Voting System through Visual Cryptography and Multi-factor Authentication to Further Mitigate Clone Phishing Attack." *Asian Journal of Research in Computer Science* 16.4 (2023): 145-160.

## APPENDIX

Generate\_shares.py:

```
import random
import string
from captcha.image import ImageCaptcha
from PIL import Image, ImageDraw
import statistics
import os
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.base import MIMEBase
from email import encoders
import sys
import mysql.connector
from PIL import Image
from io import BytesIO

#to client - img 2 - risk
#to server -

arguments = sys.argv
email = arguments[1]
voter_id = arguments[2]

# Email configuration
smtp_server = 'smtp.gmail.com'
smtp_port = 587
smtp_username = 'suryarajput20010@gmail.com'
smtp_password = 'lova ykys dnqx pkyn'
sender_email = 'suryarajput20010@gmail.com'
subject = 'Secret share for Voting System'

def generate_captcha_text(length=6):
    characters = string.ascii_uppercase + string.digits
    captcha_text = ''.join(random.choice(characters) for _ in range(length))
    return captcha_text

# Connect to the MySQL database
conn = mysql.connector.connect(
    host='localhost',
    user='root',
```

```

        password='',
        database='voting_system'
    )

# Create a cursor object to interact with the database
cursor = conn.cursor()

def save_share_and_captcha(path_1,path_2, captcha,shift):
    try:
        # Open the image file
        with open(path_1, 'rb') as image_file_1:
            image_data_1 = image_file_1.read()
        with open(path_2, 'rb') as image_file_2:
            image_data_2 = image_file_2.read()

        # Insert the data into the database
        query = "INSERT INTO shares (voter_id,share_1,share_2, captcha, shift)
VALUES (%s, %s, %s, %s, %s)"
        values = (voter_id,image_data_1,image_data_2,captcha,shift)
        cursor.execute(query, values)

        # Commit the transaction
        conn.commit()

    finally:
        # Close the cursor and connection
        cursor.close()
        if os.path.exists(path_1):
            # Delete the file
            os.remove(path_1)
        if os.path.exists(path_2):
            # Delete the file
            os.remove(path_2)
        conn.close()

def shift_and_replace(lst, shift_direction='left'):
    shift=random.randrange(50)
    size = len(lst)

    if shift_direction == 'left':
        # Shift left
        for i in range(shift):
            lst = lst[1:] + [0]
    elif shift_direction == 'right':
        # Shift right
        for i in range(shift):
            lst = [0] + lst[:-1]

```

```

    return shift, lst

def create_captcha_image(captcha_text):
    image_captcha = ImageCaptcha(fonts=[])
    captcha_image = image_captcha.generate_image(captcha_text)
    return captcha_image

def send_image(email, path, voter_id):
    receiver_email = email

    msg = MIMEMultipart()
    msg['From'] = sender_email
    msg['To'] = receiver_email
    msg['Subject'] = subject

    with open(path, 'rb') as image_file:
        image_data = image_file.read()
        image_attachment = MIMEImage(image_data, name=os.path.basename(path))
        msg.attach(image_attachment)

    body = f'Please don\'t share this image with anybody and keep it ready when you want to log in to the page. \n Note :- The first 11 characters starting with `VCXXXXXXXX` is your VOTER ID . That is {voter_id}'
    msg.attach(MIMEText(body, 'plain'))

    with smtplib.SMTP(smtp_server, smtp_port) as server:
        server.starttls()
        server.login(smtp_username, smtp_password)
        server.sendmail(sender_email, receiver_email, msg.as_string())

def generate_shares(voter_id):
    captcha_text = generate_captcha_text()
    captcha_image = create_captcha_image(captcha_text)
    captcha_image_path = f"{voter_id}_captcha.png"
    captcha_image.save(captcha_image_path)

    gs_image = captcha_image.convert('L')
    pixels = list(gs_image.getdata())
    mean_value = statistics.mean(pixels)
    stdev_value = statistics.stdev(pixels)

    threshold = mean_value - stdev_value
    bw_image = gs_image.point(lambda x: 0 if x < threshold else 1, '1')

    new_width = int((200 / bw_image.height) * bw_image.width)
    new_height = 200
    resized_image = bw_image.resize((new_width, new_height))

```



```

cmp_image_path = f"{voter_id}_cmp_img.png"
resized_image.save(cmp_image_path)

secret_image = Image.open(cmp_image_path)
share1 = Image.new('1', (secret_image.width * 2, secret_image.height * 2),
'white')
share2 = Image.new('1', (secret_image.width * 2, secret_image.height * 2),
'white')

for y in range(secret_image.height):
    for x in range(secret_image.width):
        pixel = secret_image.getpixel((x, y))
        if pixel == 0:
            share2.putpixel((2 * x, 2 * y), 1)
            share2.putpixel((2 * x, 2 * y + 1), 0)
            share2.putpixel((2 * x + 1, 2 * y), 0)
            share2.putpixel((2 * x + 1, 2 * y + 1), 1)
            share1.putpixel((2 * x, 2 * y), 1)
            share1.putpixel((2 * x, 2 * y + 1), 0)
            share1.putpixel((2 * x + 1, 2 * y), 0)
            share1.putpixel((2 * x + 1, 2 * y + 1), 1)
        else:
            share2.putpixel((2 * x, 2 * y), 0)
            share2.putpixel((2 * x, 2 * y + 1), 1)
            share2.putpixel((2 * x + 1, 2 * y), 1)
            share2.putpixel((2 * x + 1, 2 * y + 1), 0)
            share1.putpixel((2 * x, 2 * y), 1)
            share1.putpixel((2 * x, 2 * y + 1), 0)
            share1.putpixel((2 * x + 1, 2 * y), 0)
            share1.putpixel((2 * x + 1, 2 * y + 1), 1)

share1_path = os.path.abspath(f"{voter_id}_share1.png")
share2_path = os.path.abspath(f"{voter_id}_share2.png")

data_2=list(share2.getdata())

shift,data_2=shift_and_replace(data_2)

share2.putdata(data_2)
share2.save(share2_path)
share1.save(share1_path)

if os.path.exists(cmp_image_path):
    # Delete the file
    os.remove(cmp_image_path)
if os.path.exists(captcha_image_path):

```

```

        # Delete the file
        os.remove(captcha_image_path)
    secret_image.close()

    return captcha_text, f"{voter_id}_share1.png", f"{voter_id}_share2.png", shift,

captcha, path_1, path_2, shift = generate_shares(voter_id)
send_image(email, path_2, voter_id)
save_share_and_captcha(path_1, path_2, captcha, shift)

```

### Decrypt.py

```

from PIL import Image
import mysql.connector
from io import BytesIO
import os
import sys

arguments = sys.argv
voter_id = arguments[1]
img2 = arguments[2]

def shift_and_replace( lst, shift_direction='right', shift=()):
    n = list(shift)[0]
    if shift_direction == 'left':
        # Shift left
        for i in range(n):
            lst = lst[1:] + [0]
    elif shift_direction == 'right':
        # Shift right
        for i in range(n-1):
            lst = [0] + lst[:-1]
    return lst

def pad_images_to_equal_size(image1_path, image2_path):
    # Open the images
    image1 = Image.open(image1_path)
    image2 = Image.open(image2_path)

    # Get the dimensions of the images
    width1, height1 = image1.size
    width2, height2 = image2.size

```

```

# Find the maximum width and height
max_width = max(width1, width2)
max_height = max(height1, height2)

# Create a new image with the maximum dimensions
padded_image1 = Image.new('1', (max_width, max_height), color='white')
padded_image2 = Image.new('1', (max_width, max_height), color='white')

# Paste the original images onto the padded images
padded_image1.paste(image1, ((max_width - width1) // 2, (max_height - height1)
// 2))
padded_image2.paste(image2, ((max_width - width2) // 2, (max_height - height2)
// 2))

# Convert images to bytes
padded_image1.save(image1_path)
#padded_image1.show()
image_data1 = list(padded_image1.getdata())

padded_image2.save(image2_path)
#padded_image2.show()
image_data2 = list(padded_image2.getdata())

# Close the images
image1.close()
image2.close()

return image_data1, image_data2,max_width,max_height

def decrypt_and_compare(voter_id, img2):
# Retrieve the stored image (blob) from the database
shift=0
cursor = db_connection.cursor()
cursor.execute("SELECT share_1 FROM shares WHERE voter_id = %s", (voter_id,))
stored_image_blob = cursor.fetchone()[0]
cursor.execute("SELECT shift FROM shares WHERE voter_id = %s", (voter_id,))
shift=cursor.fetchone()

# Convert blob data to Image
share1 = Image.open(BytesIO(stored_image_blob))

# Get the path to the 'uploads' folder in the same directory as the script
script_directory = os.path.dirname(os.path.abspath(__file__))
uploads_folder = os.path.join(script_directory, "uploads")

# Ensure the 'uploads' folder exists

```

```

if not os.path.exists(uploads_folder):
    os.makedirs(uploads_folder)

# Save the image as PNG in the 'uploads' folder
img1_path = os.path.join(uploads_folder, f"{voter_id}_share1.png")
img2_path = img2
share1.save(img1_path, format="PNG")
share1.close()

# Get pixel data from shares
p1, p2, new_width, new_height = pad_images_to_equal_size(img1_path, img2_path)

p2=shift_and_replace(p2,'right',shift)

# XOR the pixel data
overlay = [pixel1 ^ pixel2 for pixel1, pixel2 in zip(p1, p2)]

# Create a new image with the merged data
merged_image = Image.new('1', (new_width, new_height), 'white')
merged_image.putdata(overlay)

# Save the merged image in the 'uploads' folder
merged_image_path = os.path.join(uploads_folder, f"{voter_id}_merged.png")
merged_image.save(merged_image_path, format="PNG")
merged_image.close()

# Optionally return the paths for further usage
print(merged_image_path)

# Example usage:
db_connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="voting_system"
)

decrypt_and_compare(voter_id,img2)

# Don't forget to close the database connection when done
db_connection.close()

```

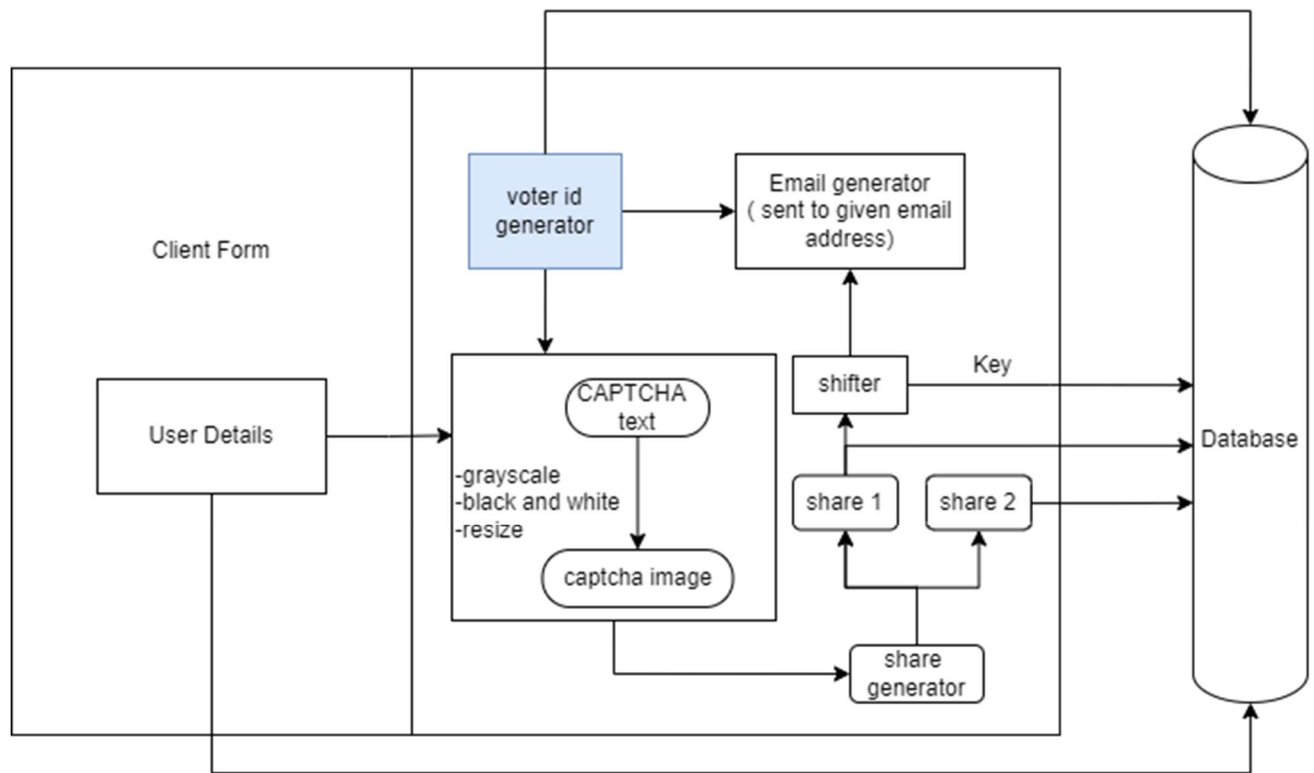


Figure 12: Architecture for registration phase

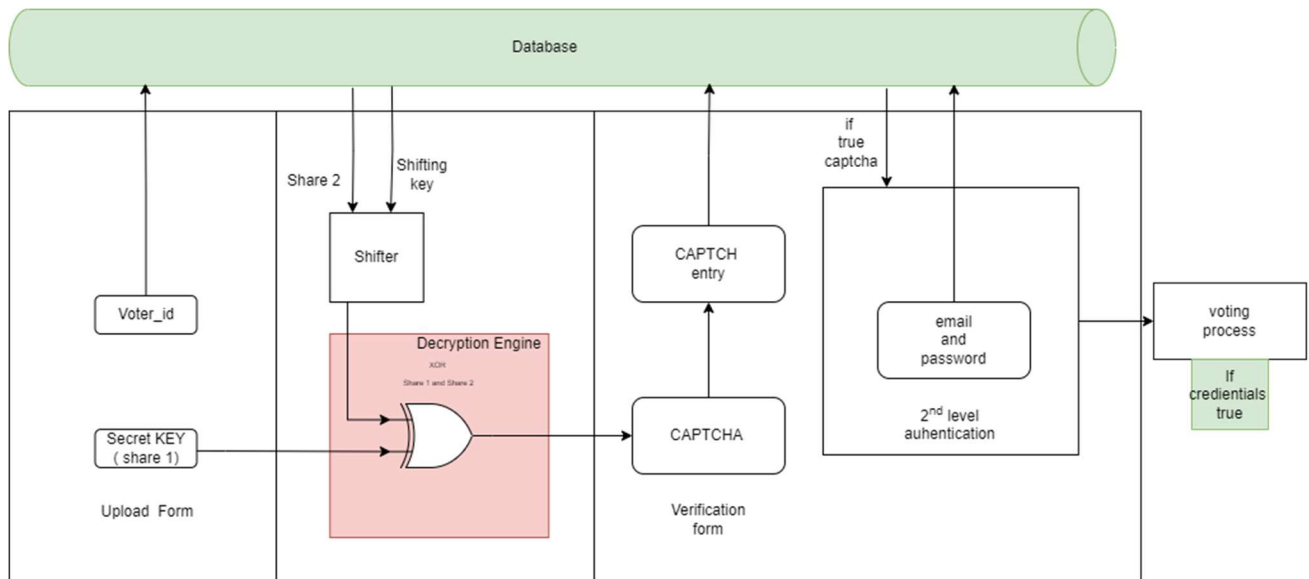


Figure 13: Architecture for Login Phase