ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 12, Issue 06, June 2025

A Secure and Scalable Online Voting System: Leveraging Smart Technologies for Digital Governance and Enhanced Privacy

[1] K. Shalini, [2] R. Rishi Sai Teja, [3] M. Ajith Kumar, [4] A. Keneeth

[1] Assistant Professor, Department of C.S.E, Matrusri Engineering College, Saidabad, Hyderabad, Telangana, India
[2] [3] [4] UG Scholar, Department of C.S.E, Matrusri Engineering College, Saidabad, Hyderabad, Telangana, India

Abstract*— The demand for reliable, transparent, and scalable online voting platforms have grown significantly, especially in democratic nations where traditional voting methods face logistical challenges, voter fraud, and limited accessibility. This work presents a secure online voting system built using the MERN stack (MongoDB, Express.js, ReactJS, and Node.js), designed to address these issues through an integrated role-based architecture. The platform accommodates two user roles: administrators, who manage elections, candidates, and parties, and voters, who securely log in and cast a single vote per election. To ensure end-to-end security and system integrity, the platform employs JSON Web Tokens (JWT) for authentication, bcrypt for password encryption, AES for vote data encryption, and CAPTCHA to prevent automated login attempts. With an intuitive user interface and advanced backend validations, the system enforces access control and voting limitations. This prototype demonstrates the feasibility of secure digital elections and provides a practical foundation for real-world implementation and future enhancements.

Index Terms*— Online voting system, secure e-voting, MERN stack, role-based access control, JWT authentication, AES encryption, CAPTCHA verification, bcrypt hashing, digital elections, voting platform security.

I. INTRODUCTION

Traditional voting methods frequently encounter obstacles like logistical delays, voter impersonation, and restricted accessibility. The increasing prevalence of digital technologies has led to the emergence of online voting as a potential solution to enhance efficiency, transparency, and convenience in electoral processes.

Nevertheless, implementing secure online voting systems necessitates resolving significant concerns such as data protection, authentication, and resilience against automated attacks. This paper introduces a secure online voting system that utilizes the MERN stack (MongoDB, Express.js, ReactJS, Node.js) and follows a role-based architecture to cater to the requirements of administrators and voters.

Authentication is ensured through the use of JSON web tokens (JWT), password hashing with bcrypt, vote data encryption with AES, and captcha for bot prevention. The system is designed with a user-friendly interface and advanced backend validations, aiming to provide a dependable model for secure digital elections.

II. OBJECTIVES AND SCOPE

A. Objectives

The goal of this project is to create a safe and user-friendly online voting system that tackles the problems associated with traditional voting methods. It guarantees secure authentication using JWT, bcrypt, and AES, and prevents unauthorized access through captcha. The system follows a role-based structure for administrators and voters, enforces restrictions on multiple votes, and offers a scalable, modular platform that can be adapted for practical elections.

B. Scope

This project encompasses the creation and execution of a secure online voting system, utilizing the MERN stack. It supports two distinct roles administrators and voters each with specific permissions and capabilities. Administrators have the responsibility of overseeing elections, managing candidates and parties, while voters can securely log in and cast a single vote per election. The system guarantees data security and reliable operations by utilizing JWT-based authentication, bcrypt password hashing, AES encryption, and captcha verification. Although the system is currently being developed as a prototype, it is designed to be scalable and adaptable for larger, real-world election environments.

III. LITERATURE REVIEW

In the last two decades, various models have been suggested to construct secure and user-friendly online voting systems. The Estonian i-voting system, one of the earliest approaches, enabled citizens to cast their votes remotely by utilizing their national identification cards. Despite being an early adopter of digital voting, concerns were raised regarding centralized control, potential client-side vulnerabilities, and the lack of transparency in the auditing process.

The helios voting system, an online voting platform that allows for open audits, employs end-to-end verifiability through the use of homomorphic encryption and zero-knowledge proofs. While helios is suitable for low-pressure situations like university elections, it is not well-suited for large-scale political elections due to its

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 12, Issue 06, June 2025

dependence on trust in the server and limited safeguards against coercion or vote buying.

In [1], researchers proposed a blockchain-based e-voting system that ensured vote immutability and transparency using Ethereum smart contracts. However, despite its robustness against tampering, the system suffered from high computational overhead, scalability limitations, and usability issues, especially for non-technical users. Similarly, the work by Kshetri and Voas [2] highlighted blockchain's potential but emphasized implementation hurdles, including regulatory uncertainty and the digital divide.

Other systems, such as the sensus protocol [3], included blind signatures to protect voter privacy, but they faced challenges in terms of performance and scalability. Biometric authentication-based systems, like those used in [4], provided more secure identity verification but raised concerns about data privacy and the expenses associated with setting up the necessary infrastructure.

Additionally, captcha integration has been studied in systems like [5] to combat bot-based attacks, but these measures were often not part of a comprehensive security framework that encompassed encrypted storage and secure authentication layers.

Most of these approaches contributed significantly to the field but either lacked user-friendliness, had complex implementation requirements, or didn't integrate a full-stack architecture suitable for public deployment.

In contrast, the proposed system presents a full-stack MERN-based solution combining proven technologies such as JWT for session security, bcrypt for password hashing, AES encryption for vote confidentiality, and CAPTCHA for bot protection. This design balances practicality, usability, and strong security while offering a modular and scalable foundation adaptable to different election scenarios.

IV. SYSTEM ARCHITECTURE

The proposed Secure Online Voting System's architecture is designed to support secure, scalable, and role-based digital elections. It is implemented using the MERN stack MongoDB, Express.js, ReactJS, and Node.js enabling full-stack JavaScript development and seamless integration between frontend and backend components. The architecture follows a multi-tier model, emphasizing modularity, data protection, and user role segregation.

A. Core Architectural Layers:

1. Presentation Layer (ReactJS):

This layer provides the interface for both administrators and voters. It delivers a user-friendly, responsive design where users can interact with the system based on their authenticated role. Voters can register, log in, and view or participate in elections, whereas administrators manage elections, parties, and candidates through secure panels.

2. Application Layer (Node.js with Express.js):

The server-side logic is handled in this layer. It validates inputs, manages authentication and authorization, processes voting operations, handles OTP verifications, and communicates with the database. APIs are protected by middleware to ensure that access is granted only to verified users with valid tokens and appropriate roles.

3. Data Layer (MongoDB):

MongoDB stores collections for users, elections, parties, candidates, and encrypted vote records. The NoSQL structure enables flexible and scalable data handling. Sensitive user credentials and vote data are stored securely using hashing and encryption techniques.

B. Security Framework

Security is a central focus of the system, implemented through multiple integrated techniques:

CAPTCHA: Prevents automated or bot-based login attempts.

Email-based OTP Verification: Adds a second layer of authentication to validate the user's identity during registration and login.

JWT (JSON Web Tokens): Ensures secure, stateless session management, allowing only authenticated users to access protected routes.

bcrypt.js: Hashes and securely stores user passwords to protect against data breaches.

AES (Advanced Encryption Standard): Encrypts vote data before storing it in the database to maintain the confidentiality and integrity of user choices.

C. Role-Based Access and Flow

1. Voter Workflow:

A voter signs up by completing CAPTCHA and email OTP verification. Upon successful login, they are authenticated using JWT and are allowed to view ongoing elections. Each voter can cast only one vote per election, which is encrypted using AES and recorded securely.

2. Administrator Workflow:

Administrators log in securely and gain access to election management modules. They can create, update, or delete elections, candidates, and parties. Administrators do not have access to voter credentials or vote content, maintaining privacy and role isolation.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 12, Issue 06, June 2025

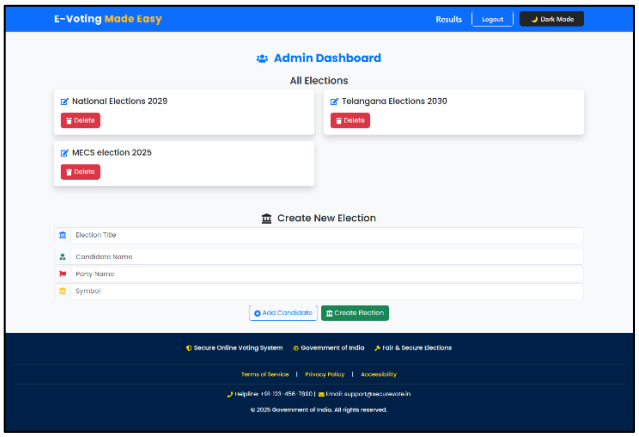

Figure 1. Admin Dashboard Showing Election & Candidate Management

A. Interaction Flow and Communication

All frontend-backend interactions are routed through secure RESTful APIs. The client sends requests with attached JWTs, and the server validates each token before processing. Real-time status updates, such as vote success or login failures, are handled through dynamic frontend rendering and backend responses.

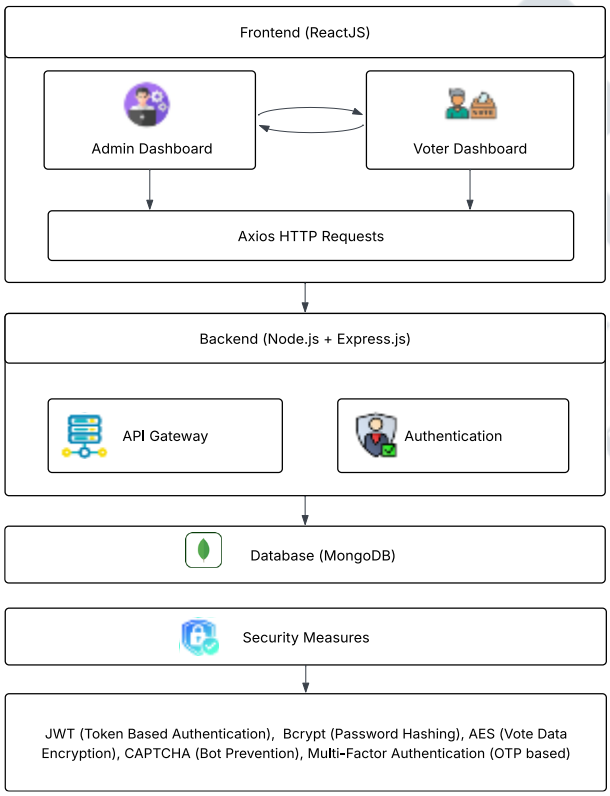

Figure 2. System Architecture of Secure Online Voting System

V. PROPOSED METHODOLOGY

The proposed system is a secure, role-based online voting platform designed to offer a scalable and tamper-proof digital alternative to conventional elections. Developed using the MERN stack (MongoDB, Express.js, ReactJS, Node.js), it integrates multiple layers of security, including token-based authentication, encryption protocols, human verification tools, and multi-factor authentication through email-based One-Time Passwords (OTPs). This section outlines the system's architectural methodology and workflows that ensure confidentiality, integrity, authenticity, and usability. By combining modern cryptographic standards, layered authentication, and human-verification mechanisms, the platform achieves trustworthiness and reliability in digital elections.

A. Role-Based Access and User Workflow

The system design includes role-based access control to distinguish between administrators and voters. Administrators have the authority to establish and manage elections, determine eligible candidates and parties, and oversee the election timetable. Nevertheless, they are prohibited from accessing voter credentials or vote data to uphold impartiality and protect privacy.

Users are obligated to register on the platform and, after verifying their identity, are shown a compilation of ongoing elections. A voter is permitted to cast a single vote per election. This limitation is implemented through both frontend validations and backend database checks. Access to restricted routes is safeguarded through access-control logic and protected API endpoints, ensuring that only authenticated users with the appropriate role can interact with the system.

Figure 3. Secure Vote Casting Page with Candidate List

B. Registration and Authentication Mechanism

The system utilizes a secure, two-step user registration and authentication process. During the registration process, the user provides their personal information, such as their name, email address, and password. The password is transformed using the bcrypt hashing algorithm before being saved in the database, making it impossible to decipher in its original form even if the data is accessed.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 12, Issue 06, June 2025

The process of verifying one's identity when logging in requires two steps. Initially, the system checks the email-password combination using bcrypt hash comparison to ensure its accuracy. After confirming the voter's identity, a six-digit OTP (one-time password) is generated and delivered to their registered email address. The user needs to input this one-time password to successfully complete the authentication process. Only after successfully verifying the user's identity, the backend generates a secure token (JSON web token), which is used to maintain the user's session during subsequent interactions.

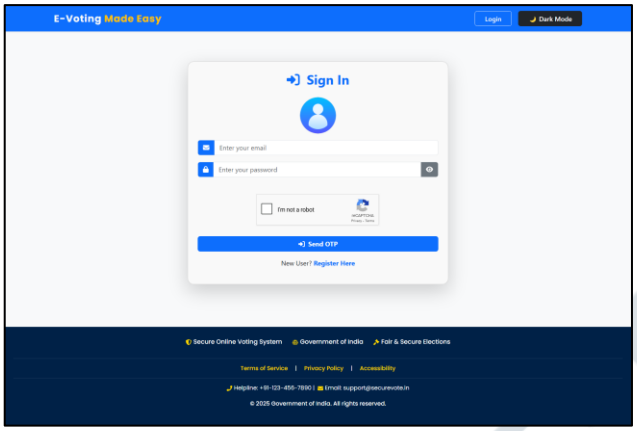

Figure 4. Voter Login Interface with Google reCAPTCHA Integration

By incorporating email-based OTP as an additional security measure, account protection is strengthened as it verifies that access is only granted to users who possess the registered email account, effectively reducing the likelihood of credential stuffing or brute-force attacks.

C. Secure Voting via AES Encryption

Once a voter is authenticated, they are presented with a real-time list of ongoing elections. Before allowing access to the voting interface, the system checks if the voter has already cast their vote in the selected election. If the check passes, the voting UI is rendered.

When a vote is cast, the chosen candidate's identifier along with relevant election metadata is encrypted using the Advanced Encryption Standard (AES) algorithm. The AES-encrypted vote is then transmitted and stored securely in the database. This ensures vote confidentiality throughout its lifecycle, even in the case of backend compromise. The decryption logic is kept isolated and is only invoked during the final result tallying, performed under administrative control.

Using AES provides symmetric encryption efficiency while guaranteeing the privacy and confidentiality of sensitive voting data.

D. Human Verification with CAPTCHA

To protect the system from automated bots and spam registrations, a CAPTCHA module is embedded into both the registration and login interfaces. The CAPTCHA challenge ensures that only real human users are allowed to proceed with sensitive operations. This addition safeguards the platform against brute-force attacks and malicious automation scripts, further bolstering the integrity of the system.

The CAPTCHA solution is lightweight and seamlessly integrated into the user interface, striking a balance between usability and security without compromising user experience.

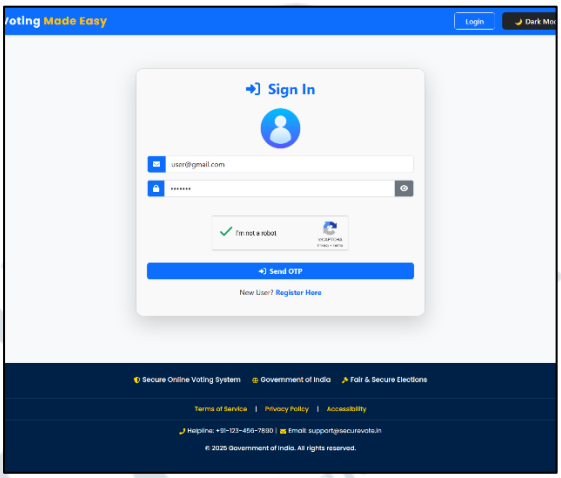

Figure 5. Google reCAPTCHA

E. Frontend-Backend Communication and Session Control

The frontend application, developed using ReactJS, provides an intuitive and responsive interface tailored for different user roles. All API calls to the backend server are facilitated through Axios, with headers carrying JWT tokens to ensure secure, stateless authentication.

The backend, developed in Node.js using the Express.js framework, exposes role-specific endpoints that are guarded by middleware to validate JWTs and enforce access control. The system also maintains audit logs of key operations, providing traceability and support for any future forensic analysis if needed.

To further enhance the robustness of session control, tokens are invalidated upon logout, and expiration times are defined to limit their validity, reducing the risk of session hijacking.

VI. IMPLEMENTATION AND RESULTS

The secure online voting system was developed using the MERN (MongoDB, Express.js, ReactJS, and Node.js) technology stack, chosen for its flexibility, scalability, and support for modern web application development. The implementation of advanced security

A. Frontend Development

The frontend of the application was built using ReactJS, enabling a dynamic, component-based structure and enhancing maintainability and scalability. The user interface

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 12, Issue 06, June 2025

(UI) was designed to be intuitive, responsive, and visually aligned with the Indian voting system's AESthetic, providing clear navigation for both voters and administrators.

Material-UI and Bootstrap were utilized for responsive design elements, ensuring cross-platform compatibility. Routing between views was handled using React Router, allowing for seamless transitions between login, registration, elections, and voting screens.

B. Backend and API Layer

The backend of the application was implemented using Node.js and Express.js, providing a lightweight and efficient server-side framework for handling API requests, session management, and database interactions. RESTful APIs were developed to facilitate secure communication between the frontend and backend, enabling functionalities such as user registration, login, vote casting, and election management.

A comprehensive middleware architecture was developed to validate JSON Web Tokens (JWT), ensuring that only authenticated users could access protected routes. Route-specific access was further segregated based on user roles voter or administrator enforcing role-based access control (RBAC).

C. Database and Data Security

The system uses MongoDB, a document-oriented NoSQL database selected for its scalability, performance, and compatibility with JavaScript-based frameworks. It stores key entities such as user credentials, election metadata, candidate lists, and voting records. Sensitive information, including passwords, is never stored in plain text; instead, bcrypt.js hashes passwords before storage, strengthening authentication security against brute-force and dictionary attacks.

To protect vote confidentiality, each vote is encrypted using AES before storage, ensuring even administrators cannot access raw vote data. JSON Web Tokens (JWT) maintain secure, stateless user sessions post-authentication, minimizing the risk of session hijacking and identity spoofing. These combined mechanisms create a robust security framework to safeguard all critical system components.

Figure 6. MongoDB Snapshot Showing Encrypted Vote and Hashed User Password

D. Multi-Factor Authentication

To safeguard user accounts and ensure that only legitimate voters access the system, a robust Multi-Factor Authentication (MFA) mechanism was implemented. The process begins with CAPTCHA verification using Google reCAPTCHA, which effectively filters out bots and automated scripts attempting to access the system. CAPTCHA acts as the first line of defense, ensuring that only real users proceed to the next authentication step.

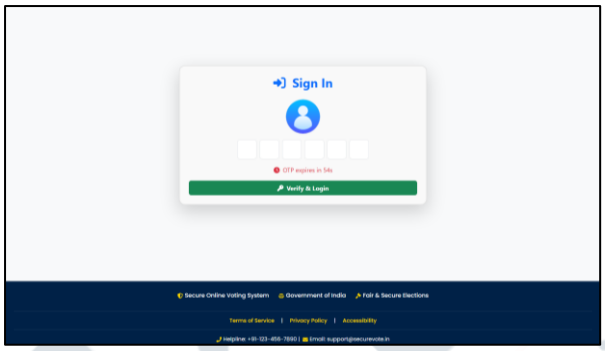

Figure 7. Email-Based OTP Verification Screen for Secure Login

Following a successful CAPTCHA validation, users are prompted to verify their identity via a One-Time Password (OTP) sent to their registered email address. This OTP is generated server-side and securely delivered using the Nodemailer library. The OTP is time-sensitive and must be entered within a limited time frame, adding a second layer of protection. This dual-check mechanism CAPTCHA followed by email OTP minimizes the risk of unauthorized access, credential theft, and brute-force attacks. By requiring something the user knows (credentials) and something the user has (email access), the system adheres to modern best practices in secure authentication.

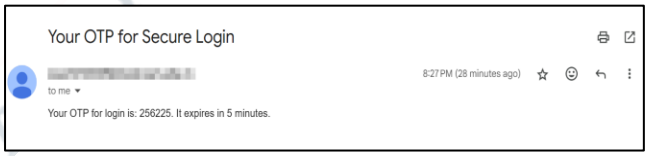

Figure 8. User Email Inbox with Received One-Time Password (OTP)

The OTP verification process also enhances user accountability and reinforces trust in the platform. Only after successful completion of both CAPTCHA and OTP steps is a session token (JWT) generated and access to the voting dashboard granted. This secure sequence of verifications makes it exceedingly difficult for attackers to bypass authentication, ensuring a more resilient voting environment.

E. Vote Casting Logic

The vote-casting mechanism prioritizes both confidentiality and integrity. After authenticating and authorizing a voter, the backend verifies whether the user has already voted in the specific election using a combination of the voter's unique ID and election ID. If no prior vote exists, the system allows submission.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 12, Issue 06, June 2025

The backend strictly enforces one vote per user per election. Any duplicate voting attempt is rejected, and the user is notified. This validation within the controller logic ensures fairness and prevents multiple votes from the same user across any device or interface.

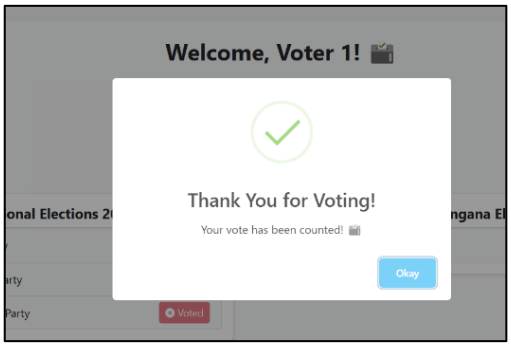

Figure 9. Confirmation Screen After Successful Vote Submission

To preserve ballot secrecy, each vote is encrypted using the Advanced Encryption Standard (AES) before storage in MongoDB, ensuring that even in the event of a database breach, votes remain unreadable without the decryption key. Encryption is performed server-side to maintain system integrity.

The backend strictly enforces one vote per user per election. Any duplicate voting attempt is rejected, and the user is notified. This validation within the controller logic ensures fairness and prevents multiple votes from the same user across any device or interface.

F. Results and Observations

Extensive testing validated both the functionality and security of the proposed system. A key achievement was the successful implementation of multi-factor authentication, requiring users to complete CAPTCHA and OTP verification before accessing any election interface. This ensured only legitimate voters participated and blocked bots and malicious users.

Vote confidentiality was preserved through AES encryption, with encrypted votes stored securely in MongoDB and no mechanism to trace votes back to voters, ensuring full anonymity and preventing administrative manipulation. The backend reliably detected and blocked duplicate voting attempts, preventing users from voting multiple times in the same election. Admin users operated from a separate dashboard, with no access to voter identities or vote contents.

Performance testing under simulated load showed fast response times for vote submission and prompt OTP delivery, typically within seconds. The system also demonstrated strong resilience against vulnerabilities like brute-force attacks, CSRF, and session hijacking, supported by secure JWT sessions and bcrypt-hashed credentials. These results confirm the platform's robustness and readiness for real-world deployment.

VII. CONCLUSION AND FUTURE SCOPE

This paper presents a secure, scalable, and user-friendly online voting system aimed at overcoming the limitations of traditional voting methods in large-scale democratic processes. Built with the MERN stack (MongoDB, Express.js, ReactJS, Node.js), the system ensures secure authentication, vote confidentiality, and defense against malicious attacks using technologies such as JSON Web Tokens (JWT), bcrypt is used for password hashing, AES encryption is employed for vote data, CAPTCHA verification is implemented for CAPTCHA, and email-based One-Time Password (OTP) is utilized for multi-factor authentication. Role-based access control further enhances security by clearly separating administrative and voter privileges.

Our implementation demonstrates that secure digital elections are feasible and can improve participation, transparency, accessibility, and efficiency while reducing logistical challenges and costs. The modular and extensible architecture also allows for easy integration of advanced features in the future.

Future enhancements could include biometric verification for stronger identity validation, real-time vote tracking dashboards for election officials, multilingual support, and integration with national databases for voter verification. The system would also benefit from cybersecurity audits and legal reviews to ensure compliance with electoral laws. With these developments, the platform holds strong potential for deployment at municipal, state, or national levels, paving the way for modernized electoral systems.

REFERENCES

[1] A. Vassil, M. Solvak, K. Vinkel, T. Alnek, A. Trechsel, and R. Alvarez, "The Estonian e-voting system: Security, usability and experience," Internet Voting, 2016.

[2] B. Adida, "Helios: Web-based Open-Audit Voting," in Proceedings of the 17th USENIX Security Symposium, 2008.

[3] M. S. Hwang and C. C. Lee, "A Study of Electronic Voting System Using Biometric Authentication," International Journal of Network Security, vol. 10, no. 1, pp. 1–10, Jan. 2010.

[4] P. Kumar, P. K. Tiwari, and D. C. Verma, "Online voting system using biometric verification and CAPTCHA for secure authentication," International Journal of Computer Applications, vol. 180, no. 32, pp. 17–22, Apr. 2018.

[5] D. Gritzalis, "Secure Electronic Voting: New Trends," The Future of Identity in the Information Society, IFIP Advances in Information and Communication Technology, vol. 298, pp. 137–148, 2009.