A

Project Report

on

Secure Vote

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering

By

Md Armaan Ansari (2100290100096)

Sion Chowdhary (2100290100164)

Under the supervision of

Mr. Harsh Modi

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

May, 2025

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:

Name: Md Armaan Ansari, Sion Chowdhary

Roll No.: 2100290100096, 2100290100164

Date:

CERTIFICATE

This is to certify that Project Report entitled “Secure Vote” which is submitted by Md Armaan Ansari and Sion Chowdhary in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science & Engineering of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

.

Mr. Harsh Modi **Dr. Vineet Sharma**

(Assistant Professor) **(Dean CSE)**

Date:

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Mr. Harsh Modi, Department of Computer Science & Engineering, KIET, Ghaziabad, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. Vineet Sharma, Head of the Department of Computer Science & Engineering, KIET, Ghaziabad, for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all the faculty members of the department for their kind assistance and cooperation during the development of our project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members, especially faculty/industry person/any person, of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Date:

Signature:

Name : Md Armaan Ansari, Sion Chowdhary

Roll No.: 2100290100096, 2100290100164

ABSTRACT

Elections are fundamental to democratic governance, yet traditional electronic voting systems (EVMs) suffer from security vulnerabilities, lack of transparency, and susceptibility to cyber threats. This report presents a **blockchain-based e-voting framework** that enhances electoral integrity through **distributed smart contracts, cryptographic security mechanisms, and decentralized metadata storage**. Key innovations include **multi-factor authentication (MFA) for voter identity verification, master-slave architecture for system resilience, and the integration of the InterPlanetary File System (IPFS) for secure metadata storage**.

The proposed system leverages **checksum-based validation** and **automated security monitoring** to prevent vote tampering and ensure data consistency. Votes are cast through **cryptographically linked blockchain wallet addresses**, ensuring anonymity while maintaining verifiability. Smart contracts autonomously handle vote counting, reducing human intervention and potential manipulation. The framework also introduces **role-based access control (RBAC) and real-time intrusion detection mechanisms**, significantly enhancing security.

Compared to traditional e-voting systems, this blockchain-driven model offers **improved transparency, security, and trustworthiness**. Future work will explore **quantum-resistant cryptographic techniques, scalable blockchain infrastructures, and enhanced voter accessibility** for large-scale adoption in governmental elections.

TABLE OF CONTENTS

Page No.

LIST OF FIGURES

LIST OF ABBREVIATIONS

EVM	Electronic Voting Machine
IPFS	InterPlanetary File System
MFA	Multi-Factor Authentication
RBAC	Role-Based Access Control
PoW	Proof of Work
PoS	Proof of Stake
TPS	Transactions Per Second
NAM	Network Animator

1. IPFS: InterPlanetary File System
2. MFA: Multi-Factor Authentication
3. RBAC: Role-Based Access Control
4. PoW: Proof of Work
5. PoS: Proof of Stake
6. TPS: Transactions Per Second

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Elections are the cornerstone of democratic societies, enabling citizens to express their will and choose their representatives. However, the integrity of electoral processes is often questioned due to vulnerabilities in existing voting systems. Traditional paper-based voting, while historically reliable, is prone to logistical inefficiencies and potential tampering. The introduction of electronic voting machines (EVMs) promised increased efficiency and accuracy but introduced new challenges, such as susceptibility to cyber-attacks, vote manipulation, and lack of transparency. These issues have led to a growing distrust in electoral systems, undermining the legitimacy of elections.

Blockchain technology offers a promising solution to these challenges. Its inherent features—decentralization, immutability, and consensus-based validation—make it an ideal candidate for enhancing the security and reliability of e-voting systems. By leveraging blockchain, we can create a transparent, tamper-proof, and efficient voting system that addresses the vulnerabilities of traditional methods and fosters greater voter trust.

1.2 PROJECT DESCRIPTION

The proposed blockchain-based e-voting framework is a comprehensive solution designed to address the challenges of security, transparency, and efficiency in electoral processes. Below is an elaboration of the key components and innovations of the framework:

a. Distributed Smart Contracts

The system leverages multiple smart contracts to manage various stages of the voting process, such as voter registration, vote casting, and vote counting. By distributing the workload across these contracts, the framework ensures scalability and prevents bottlenecks. This approach allows the system to handle a large number of voters simultaneously without compromising performance. The distributed algorithms ensure that the computational load is evenly balanced, enabling seamless operation even during high-demand periods.

---

b. Data Integrity with Checksums and N-Checks

To maintain data integrity and prevent tampering, each vote transaction is accompanied by a checksum—a unique cryptographic hash that verifies the authenticity of the data. Additionally, the system employs **n-checks**, a redundancy mechanism where vote counts are validated across multiple smart contracts before results are finalized. This ensures that the final tally is accurate and consistent, even in the presence of potential errors or malicious attacks. The combination of checksums and n-checks provides a robust defense against data manipulation.

---

c. Master-Slave Architecture for Resilience

The framework adopts a **master-slave architecture** to enhance fault tolerance and ensure continuous operation. In this setup, a master smart contract oversees the voting process, while slave contracts act as backups. If the master contract is compromised or fails, a slave contract is automatically promoted to take over, minimizing downtime and ensuring quick recovery. This architecture enhances the system's resilience against failures and cyberattacks, making it highly reliable for critical electoral processes.

---

d. Automated Security Checks

The framework incorporates **automated security protocols** that continuously monitor the system for discrepancies, anomalies, and potential intrusions. These protocols use real-time threat detection mechanisms to identify and respond to security breaches promptly. For example, if an unauthorized attempt to alter vote data is detected, the system can automatically trigger countermeasures, such as isolating the affected node or invalidating suspicious transactions. This proactive approach ensures the integrity of the voting process and builds trust among stakeholders.

---

e. Voter Authentication and Validation

To prevent fraudulent voting, the system employs a **multi-layered voter authentication process**. Voter identities are validated by cross-referencing their government-issued voter IDs with their IP addresses and other identifying information stored within the smart contracts and blockchain. This ensures that only eligible voters can participate in the election. Additionally, the system can detect and block duplicate votes or attempts to vote from unauthorized locations, further enhancing security.

---

f. Integration of IPFS for Metadata Storage

To optimize storage and reduce the load on the blockchain, the framework integrates the **InterPlanetary File System (IPFS)** for storing election metadata, such as voter registration details and vote transactions. IPFS is a decentralized storage system that ensures data availability and redundancy. Only the cryptographic hashes of the metadata are stored on the blockchain, enabling efficient and secure data retrieval. This approach minimizes the storage requirements of the blockchain while maintaining the integrity and accessibility of election data.

---

g. Secure Admin Access Control

Admin access to the voting system is tightly controlled through **wallet address validation** and **secret key verification**. Only authorized personnel with validated wallet addresses and corresponding secret keys can manage and oversee the system. This ensures that critical functions, such as initiating the voting process or finalizing results, are performed by trusted individuals. The use of cryptographic keys adds an additional layer of security, preventing unauthorized access and potential manipulation of the system.

---

h. Mapping Voter IDs to Wallet Addresses

Each voter is assigned a **unique wallet address** that is linked to their government-issued voter ID. This mapping ensures that every vote is securely cast by an authenticated voter. The wallet addresses are preconfigured with predefined gas fees, making the voting process cost-effective for users. By linking voter IDs to wallet addresses, the system ensures accountability and traceability, while also preventing impersonation or unauthorized voting.

Additional Benefits of the Framework

1. **Transparency**: All vote transactions are recorded on the blockchain, making the process fully transparent and auditable. Anyone can verify the integrity of the election results without compromising voter privacy.
2. **Immutability**: Once a vote is recorded on the blockchain, it cannot be altered or deleted, ensuring the permanence and authenticity of the data.
3. **Decentralization**: The use of blockchain eliminates the need for a central authority, reducing the risk of manipulation and increasing trust in the electoral process.
4. **Scalability**: The distributed smart contracts and IPFS integration ensure that the system can scale to accommodate large-scale elections with millions of voters.
5. **Cost Efficiency**: By optimizing storage and using predefined gas fees, the framework minimizes operational costs while maintaining high security and performance.

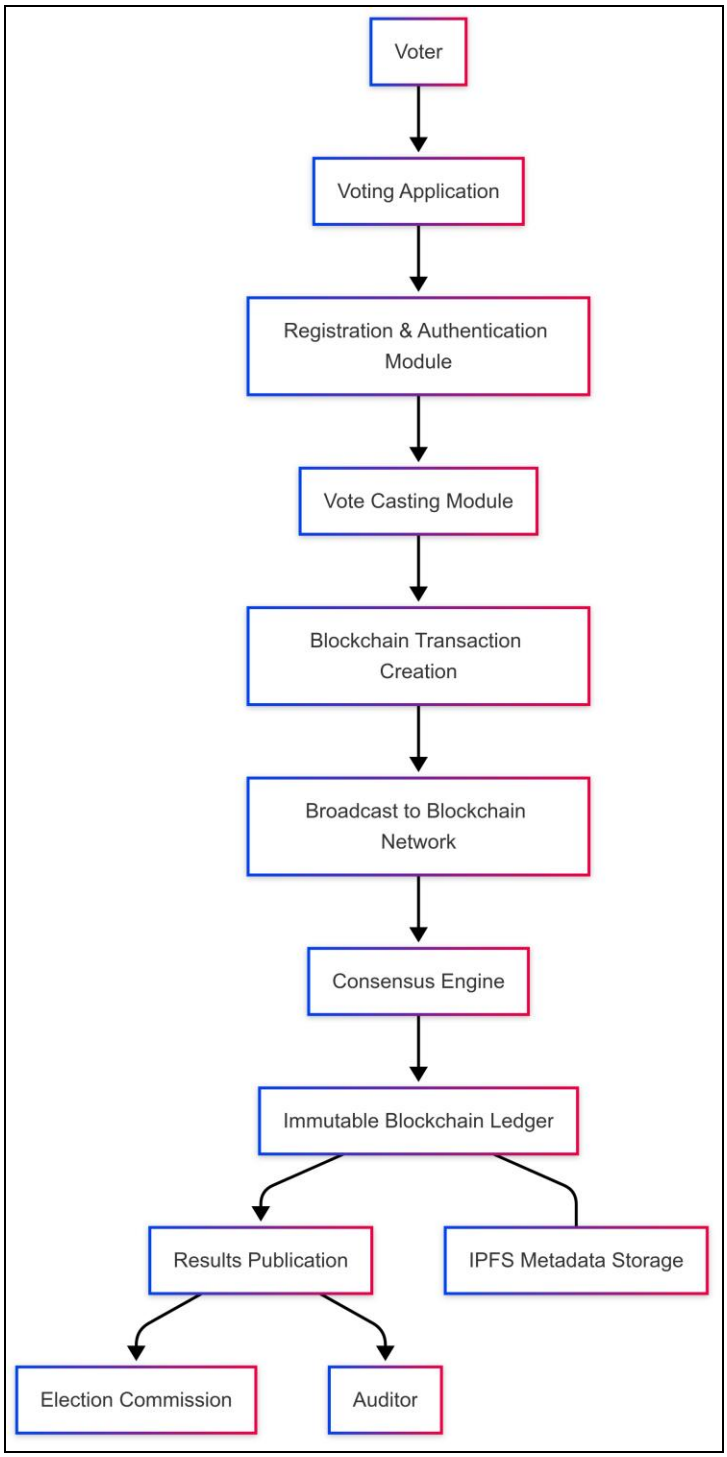


Figure 1.1 Flow chart

1.3 PROBLEM FOUNDATION

In contemporary democratic systems, ensuring secure, transparent, and reliable voting processes remains a significant challenge. Traditional electronic voting mechanisms, despite their promise of efficiency and speed, frequently encounter critical issues concerning security breaches, data tampering, voter anonymity violations, and overall transparency. These vulnerabilities undermine public confidence and can threaten the legitimacy of democratic outcomes. Moreover, centralized electronic voting systems suffer from inherent single points of failure, where corruption or technical malfunctions in central servers or databases can lead to irreparable loss or manipulation of voting data.

A notable issue in existing electronic voting frameworks is the lack of robust voter authentication mechanisms. Traditional verification methods typically rely on basic credentials such as government-issued identification or PIN-based systems, which can be susceptible to forgery, identity theft, and fraudulent voting. Without comprehensive, multi-factor authentication mechanisms, it becomes challenging to ensure that votes are genuinely cast by legitimate voters, thus questioning the authenticity of the election outcomes.

Another critical concern is the balance between voter anonymity and election transparency. Traditional voting methods struggle to maintain voter anonymity while providing transparency and auditability of the voting process. Ensuring that each vote remains confidential yet verifiable by independent entities poses a complex technical challenge, and conventional solutions often compromise either transparency or anonymity to achieve the other.

Additionally, existing systems generally lack effective measures for real-time detection and mitigation of cyber threats and malicious intrusions. This gap increases the risk of sophisticated cyber-attacks capable of altering electoral data, disrupting voting processes, or even invalidating election results. Such vulnerabilities can severely damage public trust and the perceived fairness of electoral processes.

The scalability of electronic voting systems represents another pressing challenge, particularly when applied to large-scale elections involving millions of voters. Traditional systems often experience bottlenecks and performance degradation, potentially leading to delayed vote counting and result verification. Moreover, storage and management of extensive election metadata without compromising performance or security remain persistent hurdles.

Thus, the primary problem that this project aims to resolve revolves around developing a highly secure, transparent, scalable, and resilient voting framework. This framework will leverage blockchain technology, decentralized storage, advanced cryptographic protocols, and distributed smart contracts to overcome the limitations of current e-voting systems, ultimately ensuring electoral integrity and bolstering public trust in democratic processes.

Limitations of Existing E-Voting Frameworks

Despite these advancements, several limitations persist in existing e-voting frameworks:

1. Anonymous Vote-Casting: Ensuring that votes remain anonymous while allowing voters to verify their participation is a significant challenge. Many systems struggle to balance transparency with voter privacy, often sacrificing one for the other.

2. Individualized Ballot Forms: Maintaining ballot secrecy while enabling secure verification processes remains a complex task. Systems must ensure that votes are recorded accurately without revealing the voter's identity or choices.

3. Ballot Casting Verifiability: Allowing voters to confirm that their votes were accurately recorded without revealing their identities is difficult to achieve. Many systems either compromise on verifiability or anonymity, leading to potential vulnerabilities.

4. Increasing Security Issues: Cyber-attacks, hacking, and system manipulation pose significant threats to election integrity. As voting systems become more digital, they also become more susceptible to sophisticated attacks.

5. Lack of Transparency and Trust: Voters often doubt online voting results due to perceived vulnerabilities. The lack of transparency in many e-voting systems has led to a decline in public trust.

6. Voting Delays and Inefficiencies: Technical issues can cause delays and inefficiencies, especially in remote or absentee voting scenarios. These issues can undermine the credibility of the electoral process.

7. Data Storage and Retrieval: Efficient and secure storage of election metadata remains a challenge, affecting the transparency and auditability of elections. Many systems struggle to store and retrieve large volumes of data securely and efficiently.

1.4 PROPOSED SYSTEM

To effectively address the previously articulated challenges, this project proposes a comprehensive blockchain-based electronic voting (e-voting) system. The fundamental objective of this proposed system is to enhance election security, maintain voter anonymity, ensure complete transparency, and significantly improve scalability and reliability. By integrating advanced blockchain technologies, robust cryptographic protocols, and decentralized storage mechanisms, this system provides a resilient and trustworthy voting environment suitable for modern democratic processes.

At the core of the proposed system lies a decentralized blockchain infrastructure. Unlike conventional systems that rely on centralized databases, this decentralized model eliminates single points of failure, significantly reducing vulnerability to cyber-attacks and technical failures. The blockchain serves as an immutable ledger, transparently recording every vote transaction, ensuring that votes cannot be altered, deleted, or duplicated after submission. Each vote is secured through cryptographic hashes and digital signatures, thereby safeguarding the integrity and authenticity of electoral data.

An innovative aspect of the proposed system is the deployment of distributed smart contracts. These contracts autonomously handle critical operations such as voter registration, authentication, vote casting, vote counting, and result verification. By distributing these responsibilities across multiple smart contracts, the system achieves enhanced scalability and load distribution. This approach prevents bottlenecks, ensuring smooth and efficient handling of large-scale elections involving millions of voters simultaneously.

The system introduces a sophisticated multi-factor authentication (MFA) process for voter identity verification. Voters undergo multiple verification stages, including biometric authentication, government-issued identification validation, and IP address cross-verification. Each voter is assigned a unique blockchain wallet address linked directly to their verified identity. This rigorous authentication protocol effectively mitigates identity theft, voter impersonation, and fraudulent voting activities, thereby guaranteeing the legitimacy of the voting process.

To maintain voter anonymity while ensuring auditability, the system employs zero-knowledge proofs and advanced cryptographic techniques. Votes are encrypted and linked anonymously to voter identities, allowing voters to verify their own votes without compromising their privacy. Observers and independent auditors can verify election results and the integrity of the vote counts transparently, using publicly accessible blockchain data and cryptographic verifications. This dual assurance of anonymity and transparency addresses a significant limitation inherent in traditional electronic voting methods.

The proposed system also integrates the InterPlanetary File System (IPFS) for decentralized storage of election metadata, including voter registration records, ballot data, and audit logs. Only cryptographic hashes of these metadata records are stored on the blockchain, significantly reducing the storage burden on the blockchain while ensuring efficient and secure retrieval. IPFS decentralization guarantees resilience and redundancy, ensuring continuous data availability even in scenarios of partial node failure or cyber-attacks.

Furthermore, the system is equipped with automated security checks and real-time threat detection mechanisms to continually monitor and safeguard the voting infrastructure against cyber threats. Intrusion detection systems (IDS) promptly identify suspicious activities, and automated security protocols isolate and mitigate potential threats instantly. Administrators

receive real-time alerts and detailed analytics to proactively manage and respond to emerging security incidents, enhancing overall system resilience and reliability.

Lastly, to ensure cost-effectiveness and accessibility, predefined gas fees and optimized transaction protocols are embedded within the blockchain infrastructure. This approach minimizes the operational costs incurred during elections, making the system economically viable for widespread adoption. Moreover, the proposed architecture is designed to integrate seamlessly with existing electoral processes and technologies, allowing smooth transition and interoperability with traditional voting systems.

By leveraging blockchain's intrinsic strengths—decentralization, immutability, and transparency—the proposed system addresses the critical limitations of traditional electronic voting systems. The implementation of distributed smart contracts, advanced cryptographic methods, robust authentication processes, and decentralized metadata storage culminate in a secure, scalable, and transparent e-voting solution that fosters public trust and ensures electoral integrity.

Blockchain Technology as a Solution

To address these limitations, recent research has focused on integrating **blockchain technology** into e-voting systems. Blockchain's **decentralized** and **immutable ledger** provides a robust foundation for secure and transparent voting processes. Its inherent features—such as decentralization, immutability, and consensus-based validation—make it an ideal candidate for enhancing the security and reliability of e-voting systems. Blockchain technology ensures that once a vote is recorded, it cannot be altered or deleted, providing a **tamper-proof** and **transparent record** of the election results.

However, existing blockchain-based e-voting frameworks often lack comprehensive mechanisms for **load management**, **data consistency**, and **system resilience**. Many systems struggle to handle the high transaction volumes associated with large-scale elections, leading to delays and inefficiencies. Additionally, ensuring data consistency across multiple nodes in a decentralized network remains a challenge, as discrepancies can arise due to network latency or malicious actors.

Proposed Framework

This research paper aims to bridge these gaps by introducing **advanced smart contract management**, **distributed algorithms**, **automated security protocols**, and **IPFS (InterPlanetary File System) integration** into the blockchain-based e-voting system.

- **Smart Contracts**: These are used to automate the voting process, ensuring that votes are recorded and counted accurately without human intervention.

- **Distributed Algorithms**: These help balance the workload across multiple nodes, preventing bottlenecks and enhancing scalability.
- **Automated Security Protocols**: These continuously monitor the system for potential threats, ensuring that any discrepancies or intrusions are detected and addressed in real-time.
- **IPFS Integration**: This allows for the secure and efficient storage of election metadata, ensuring that data can be retrieved and verified without overburdening the blockchain.

By leveraging these innovations, the proposed framework seeks to create a **secure**, **transparent**, and **tamper-proof electronic voting system** that addresses the limitations of existing systems and fosters greater trust in electoral processes. The system's decentralized architecture ensures that there is no single point of failure, making it highly resistant to cyber-attacks and unauthorized alterations. The use of blockchain technology also ensures that the voting process is transparent and auditable, allowing voters and observers to verify the integrity of the election results.

Conclusion

In conclusion, while significant progress has been made in the development of e-voting systems, challenges related to **security**, **transparency**, and **scalability** persist. The integration of blockchain technology offers a promising solution to these challenges, paving the way for more secure and trustworthy electoral processes. This research builds on existing work by proposing a comprehensive framework that leverages blockchain's strengths to create a robust and reliable e-voting system. By addressing the limitations of existing systems, the proposed framework aims to enhance the integrity and credibility of electoral processes, ensuring that elections remain **free**, **fair**, and reflective of the will of the people.

CHAPTER 2

LITERATURE REVIEW

The evolution of electronic voting systems has been driven by the need to address persistent challenges related to **security**, **transparency**, and **efficiency** in electoral processes. Traditional voting methods, such as paper-based ballots, have historically been reliable but are prone to logistical inefficiencies, human errors, and potential tampering. These limitations have led to the development of **electronic voting machines (EVMs)**, which promised to improve accuracy, speed, and efficiency in vote counting. However, the shift to electronic voting introduced new vulnerabilities, including susceptibility to **cyber-attacks**, **vote manipulation**, and a lack of transparency in the voting process. These issues have eroded public trust in electoral systems, prompting researchers and technologists to explore innovative solutions to enhance the integrity and reliability of voting mechanisms.

Early Developments in E-Voting Systems

One of the earliest and most notable attempts to address these challenges was **Helios**, a web-based open-audit voting system developed by Adida in 2008. Helios was designed to emphasize **transparency** and **security**, allowing voters to verify that their votes were accurately recorded while maintaining ballot secrecy. The system introduced a novel approach to voter-verifiable elections, enabling voters to audit their votes without compromising anonymity. Despite its innovative design, Helios faced limitations in ensuring **complete anonymity** and **scalability**, particularly in large-scale elections where the system struggled to handle the volume of votes efficiently. Additionally, the reliance on centralized components made it vulnerable to certain types of cyber-attacks, highlighting the need for more robust and decentralized solutions.

In the same year, **Scantegrity**, proposed by Chaum et al., introduced an **end-to-end voter-verifiable optical scan voting system**. Scantegrity allowed voters to confirm that their votes were accurately recorded without compromising secrecy. The system used cryptographic techniques to ensure that votes could be verified without revealing the voter's identity. While Scantegrity improved trust in the voting process by providing a mechanism for voters to verify

their votes, it still faced challenges in ensuring **data integrity** and preventing sophisticated cyber-attacks. The system's reliance on physical ballots and optical scanning also introduced logistical complexities, making it less suitable for fully electronic voting environments.

In 2012, Dalia et al. proposed the **STAR-Vote** system, which integrated **fairness** and **robustness** through broadcast mechanisms. STAR-Vote introduced recovery and commitment rounds to ensure ballot secrecy and judgment, providing computational security proofs for the system's reliability. The system aimed to address some of the limitations of previous e-voting systems by incorporating advanced cryptographic techniques and ensuring that votes could be verified without compromising voter privacy. However, the system's complexity and reliance on centralized components made it less scalable and more vulnerable to **single points of failure**. These limitations highlighted the need for a more decentralized and resilient approach to electronic voting.

Building on these developments, Bell et al. introduced **Star-vote** in 2013, focusing on creating a **secure**, **transparent**, and **auditable voting mechanism**. Star-vote used hashed tokens to maintain voter privacy while ensuring vote verifiability. The system aimed to strike a balance between transparency and anonymity, allowing voters to verify their votes without revealing their identities. Despite its advancements, Star-vote still faced challenges in handling **large-scale elections** and ensuring complete transparency without compromising voter anonymity. The system's reliance on centralized components and cryptographic techniques also introduced potential vulnerabilities, underscoring the need for a more robust and decentralized solution.

CHAPTER 3

SYSTEM REQUIREMENTS AND SPECIFICATION

The proposed blockchain-based e-voting system is designed to enhance the security, transparency, and efficiency of electoral processes. Below are the key system requirements and specifications:

3.1 Feasibility Study

The successful deployment of any digital election platform relies on a carefully conducted feasibility analysis to ensure that the proposed system is not only technically implementable but also legally compliant, economically viable, and socio-politically acceptable. SecureVote, as a blockchain-based e-voting solution, introduces novel technologies while addressing the pain points of legacy voting systems. This study offers a multidimensional analysis to evaluate whether the system can be realized and sustained under real-world conditions.

1. Technical Feasibility

The SecureVote system is grounded in advanced technologies such as Ethereum smart contracts, distributed file storage (IPFS), and cryptographic integrity protocols like SHA-256. The technical feasibility assesses whether existing hardware, software, and network infrastructure can support the proposed system.

- **Infrastructure Readiness:** Most of the system can be deployed on decentralized cloud platforms with support for Ethereum nodes and smart contract execution. Lightweight front-end clients allow compatibility with low-bandwidth devices, which is especially critical for remote or rural areas.

- **Modularity and Scalability:** The architecture follows a micro-contract-based design with sharding of vote processing, which means each voting shard can be deployed in parallel to manage voter load. This architecture supports linear scalability without overloading a single node or contract.

- **Interoperability with Existing Systems:** SecureVote can integrate with national identity databases and KYC APIs to validate Voter ID documents without needing to expose or duplicate sensitive information. This minimizes friction during adoption and avoids architectural overhauls for electoral commissions.
- **Maintenance Complexity:** While blockchain-based systems reduce the need for centralized maintenance, keeping Ethereum nodes synchronized, managing gas optimization, and updating smart contracts (via proxy upgrades) require a skilled DevOps team. These requirements are manageable but demand specialized expertise.

2. Operational Feasibility

Operational feasibility determines how well SecureVote fits into existing electoral workflows and whether it can be operated effectively by stakeholders such as election officials, technical teams, and voters.

- **User Accessibility:** With a web-based front-end and mobile responsiveness, SecureVote makes participation possible for all demographics. The user interface has been simplified for non-technical users, and language localization can further bridge accessibility gaps.
- **Administrator Control:** Role-based access and secure wallet management tools allow the electoral authority to configure elections, monitor the system's status, and perform oversight without needing blockchain proficiency.
- **Security in Practice:** Real-time alerts, multi-signature admin access, and contract-level redundancy mechanisms enhance operational security even in hostile or volatile regions.

3. Economic Feasibility

Economic feasibility analyzes whether the implementation and long-term operation of SecureVote are justifiable relative to the benefits it offers.

- **Initial Deployment Costs:** Upfront investments include setting up validator infrastructure, integrating national identity verification APIs, and training staff. While higher than paper-based systems in early stages, costs reduce drastically in subsequent elections due to reusability and automation.
- **Transaction Costs:** By optimizing contract execution and batching transactions, SecureVote reduces Ethereum gas costs. The pre-funding model ensures that voters do not bear these costs, which is a critical factor for equitable participation.
- **Return on Investment:** In the long term, the system reduces costs associated with physical ballot printing, logistics, security deployments, human counting errors, and post-election audits. Additionally, faster result publication minimizes political unrest and resource expenditure.

4. Legal Feasibility

This evaluates whether the system can operate within the legal frameworks governing elections, data privacy, and digital identity.

- **Compliance with Election Laws:** Smart contracts can be made transparent and auditable by election commissions, which supports compliance with right-to-information provisions. Immutable logs also satisfy chain-of-custody requirements mandated by most electoral boards.
- **Data Protection and Privacy:** By avoiding on-chain storage of personal identifiable information (PII) and using irreversible hash functions, SecureVote conforms to global privacy standards such as India's Digital Personal Data Protection Act and GDPR.

5. Security Feasibility

Given that election systems are often targeted by malicious actors, a security-first assessment is vital.

- **End-to-End Integrity:** From voter onboarding to vote storage and tally verification, each layer in SecureVote has built-in validation, hashing, or encryption. Votes are immutable, verifiable, and unlinkable to identities.
- **Resilience Against Attacks:** By deploying on a decentralized blockchain and adopting a master-slave failover architecture, the system can recover from DDoS attacks, node failures, or even state corruption—without human intervention.
- **Tamper Resistance:** Any attempt to alter a vote would change its SHA-256 hash, triggering mismatch flags in the system and enabling automated dispute protocols.
- **Fraud Prevention:** One-vote-per-ID enforcement, combined with gas-prepaid wallets and identity-bound registration, eliminates mass bot voting or fraudulent re-registration.

3.2 SDLC Model to be Used

The selection of an appropriate Software Development Life Cycle (SDLC) model plays a critical role in the structured and efficient delivery of a software project. Considering the scope, complexity, evolving security considerations, and stakeholder engagement required for the **SecureVote** project, the **Agile SDLC model** is the most suitable choice.

Agile is an iterative, incremental, and adaptive development methodology that enables continuous delivery and refinement of software in time-boxed cycles called sprints. Given that SecureVote involves a combination of blockchain integration, cryptographic security features, and real-time user interfaces for voters and administrators, Agile offers the flexibility and control required for managing this complexity.

Key reasons for selecting Agile for SecureVote:

1. **Incremental Development of Critical Modules**
   SecureVote comprises several modular components—such as registration, voting, verification, admin control, and blockchain integration. Agile enables the incremental development of these features in prioritized iterations, allowing each to be designed, built, tested, and refined independently.

2. **Early Feedback on Core Features**

   Since trust, usability, and transparency are essential in e-voting systems, involving stakeholders and testers early through Agile iterations ensures critical issues are caught before full-scale deployment. Prototypes of voting flow, identity validation, or smart contracts can be reviewed and adjusted quickly.

3. **Flexibility in Adapting to Security or Legal Requirements**

   Electoral systems must comply with evolving data privacy regulations, security standards, and jurisdiction-specific voting laws. Agile's adaptability supports iterative refinements to meet legal and technical updates without disrupting overall progress.

4. **Continuous Integration and Testing**

   Blockchain logic, front-end interfaces, and identity verification flows can be integrated and tested continuously within Agile cycles. This leads to faster detection of bugs, better component compatibility, and a stable system ready for user acceptance testing at the end of each sprint.

5. **Stakeholder and User Engagement**

   Agile promotes regular interaction between developers, supervisors, and end users. In the context of SecureVote, this ensures alignment with electoral goals, transparency expectations, and administrative control needs—especially important for a public trust-driven platform.

3.3 Functional Requirements:

1. Voter Registration: The system must validate voter eligibility by cross-referencing government-issued voter IDs and IP addresses.

2. Vote Casting: Voters should be able to cast votes securely using unique wallet addresses linked to their voter IDs.

3. Vote Counting: The system must aggregate votes and perform n-checks across multiple smart contracts to ensure accuracy.

4. Result Verification: Voters and observers should be able to verify vote counts independently through a transparent blockchain ledger.
5. Admin Access: Secure admin access must be ensured through wallet address validation and secret key verification.

3.4 Non-Functional Requirements:

1. Scalability: The system should handle large-scale elections with millions of voters efficiently.
2. Security: Data integrity must be maintained using checksums, n-checks, and automated intrusion detection systems.
3. Resilience: A master-slave architecture should ensure fault tolerance and continuous operation.
4. Transparency: The blockchain ledger should provide immutable and auditable records of all transactions.
5. Cost-Effectiveness: Predefined gas fees should minimize transaction costs for voters.

3.5 Technical Specifications:

- Blockchain Platform: Ethereum (supports smart contracts).
- Storage: IPFS for decentralized metadata storage.
- Hardware: Intel Quad-core 2.4 GHz processor, 16 GB RAM, 20 GB SSD.
- Software: Windows 10, Visual Studio 2022, Microsoft SQL Server 2019.

This system ensures a secure, transparent, and efficient e-voting process, addressing the limitations of traditional voting systems.

CHAPTER 4

SYSTEM DESIGN

The proposed blockchain-based e-voting system is designed to enhance the security, transparency, and efficiency of electoral processes. Below is the detailed system design, including the system architecture, use case diagram, and data flow diagram.

4.1. System Architecture

The system architecture is divided into three main layers: **Presentation Layer**, **Business Logic Layer**, and **Data Layer**. Each layer plays a critical role in ensuring the system's functionality, security, and scalability.

1 Presentation Layer

The Presentation Layer provides the user interface for voters and administrators. For voters, it offers a user-friendly interface to register, authenticate, and cast votes. The interface is responsive and compatible with various devices such as desktops, tablets, and smartphones. For administrators, it provides an interface to manage the voting process, monitor system health, and oversee security protocols. Access to the admin interface is restricted through wallet address validation and secret key verification.

2 Business Logic Layer

The Business Logic Layer is the core of the system, where all the critical operations are managed. It consists of multiple smart contracts, each responsible for specific tasks. The Registration Contract validates voter eligibility by cross-referencing government-issued voter IDs and IP addresses. The Voting Contract facilitates secure vote casting and generates checksums for each vote to ensure data integrity. The Counting Contract aggregates votes and performs n-checks across multiple smart contracts to validate vote counts. The Verification

Contract allows voters and observers to independently verify vote counts and system integrity. The Admin Control Contract manages administrative functions and ensures secure access control.

The system also employs a Master-Slave Architecture to enhance fault tolerance. If the master smart contract fails, a slave contract is automatically promoted to take over, ensuring continuous operation. Automated Security Protocols continuously monitor the system for discrepancies, intrusions, and potential threats. Automated alerts notify administrators of any security breaches, ensuring prompt action.

3 Data Layer

The Data Layer is responsible for storing and managing all system data. The Blockchain stores all vote transactions and hashes of election metadata, ensuring immutability, transparency, and data integrity. The InterPlanetary File System (IPFS) stores election metadata, such as voter registration details and vote transactions. Only the hashes of this metadata are stored on the blockchain, ensuring efficient data retrieval without overburdening the blockchain. The Database stores additional system data, such as voter IDs, wallet addresses, and admin credentials. The database is secured using encryption and access control mechanisms.

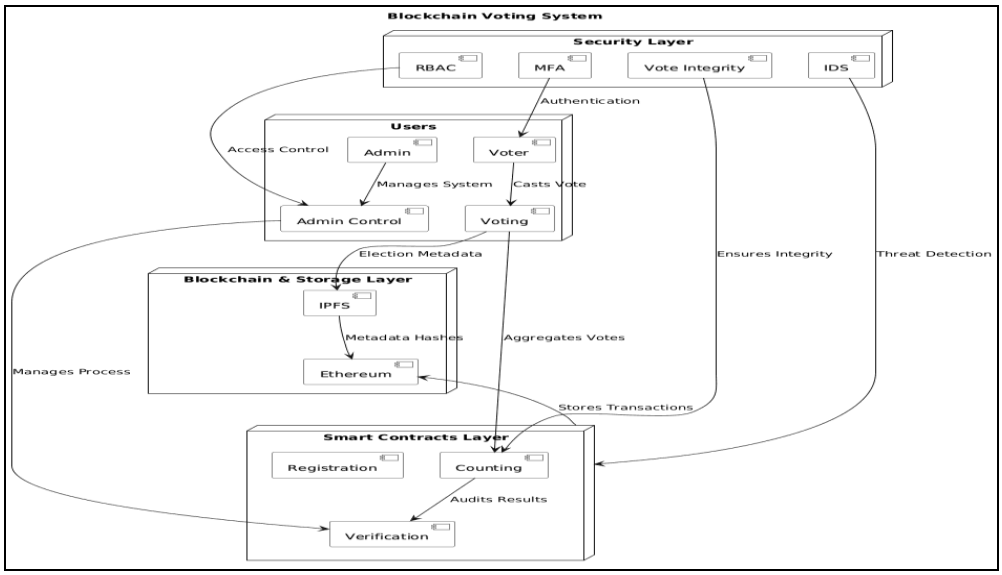


Figure 1.2 System Architecture

4.2. Use Case Diagram

The use case diagram illustrates the interactions between the system's primary actors (voters, administrators, and observers) and the system's core functionalities.

1 Actors

The primary actors in the system are Voters, Administrators, and Observers. Voters register, authenticate, and cast votes. Administrators manage the voting process, monitor system health, and oversee security protocols. Observers verify vote counts and system integrity.

2 Use Cases

The system supports several key use cases. Voter Registration involves voters providing their government-issued voter IDs and IP addresses, which the registration contract validates. Vote Casting allows voters to cast their votes using their unique wallet addresses, with the voting contract generating a checksum for each vote to ensure data integrity. Vote Counting involves the counting contract aggregating votes and performing n-checks across multiple smart contracts to validate vote counts. Result Verification allows voters and observers to independently verify vote counts and system integrity using the verification contract. Admin Access enables administrators to manage the voting process and oversee security protocols, with access restricted through wallet address validation and secret key verification. System Monitoring involves automated security protocols continuously monitoring the system for discrepancies, intrusions, and potential threats.

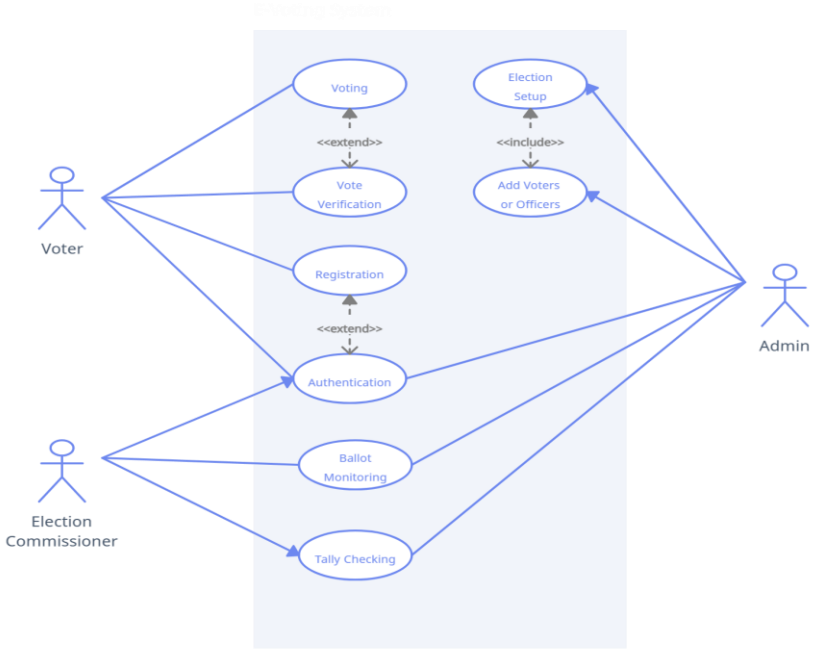


Figure 1.3 Use Case Diagram

4.3. Data Flow Diagram

The data flow diagram (DFD) illustrates how data moves through the system, from voter registration to result verification.

1 Level 0 DFD (Context Diagram)

The Level 0 DFD shows the interaction between external entities and the system. Voters provide voter ID and IP address for registration and cast votes. Administrators manage the voting process and monitor system health. Observers verify vote counts and system integrity. The system processes include Voter Registration, Vote Casting, Vote Counting, Result Verification, and Admin Access. Data stores include the Blockchain, IPFS, and Database.

2 Level 1 DFD

The Level 1 DFD provides a more detailed view of the system's processes. Voter Registration involves the registration contract validating voter eligibility and assigning a unique wallet address. Vote Casting involves the voting contract generating a checksum for the vote and storing it on the blockchain. Vote Counting involves the counting contract aggregating votes

and performing n-checks to validate vote counts. Result Verification involves the verification contract allowing voters and observers to independently verify vote counts. Admin Access involves the admin control contract validating admin credentials and granting access to administrative functions.

3 Level 2 DFD (Detailed Processes)

The Level 2 DFD provides a detailed view of each process. Voter Registration involves the voter providing voter ID and IP address, the registration contract cross-referencing government databases to validate eligibility, and assigning a unique wallet address upon validation. Vote Casting involves the voter logging in using their wallet address and multi-factor authentication (MFA), casting their vote, and the voting contract generating a checksum. Vote Counting involves the counting contract aggregating votes from multiple smart contracts, performing n-checks to validate vote counts, and recording the final vote count on the blockchain. Result Verification involves voters and observers accessing the verification contract to verify vote counts, with the verification contract retrieving vote counts from the blockchain and displaying them for verification. Admin Access involves the admin logging in using their wallet address and secret key, the admin control contract validating credentials, and granting access to administrative functions.

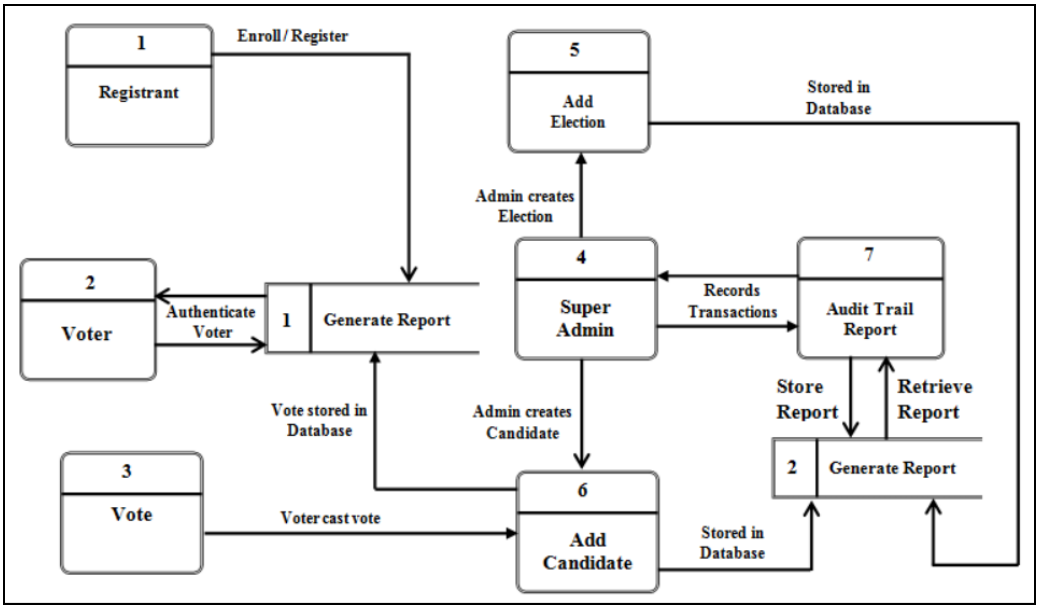


Fig 1.4 Data Flow Diagram

CHAPTER 5

PROPOSED METHODOLOGY

The proposed methodology for the blockchain-based e-voting system is designed to address the limitations of existing electronic voting frameworks by leveraging the inherent strengths of blockchain technology. The system aims to enhance security, transparency, and efficiency in the voting process while ensuring voter anonymity and data integrity. This chapter outlines the key components and architecture of the proposed system, along with the methodologies employed to achieve its objectives.

5.1 System Framework and Architecture

The proposed system is built on a decentralized blockchain architecture, which ensures that there is no single point of failure, making it highly resistant to cyber-attacks and unauthorized alterations. The system is designed to handle large-scale elections with millions of voters, ensuring scalability and reliability. The architecture incorporates distributed smart contracts, checksums, n-checks, master-slave architecture, and IPFS (InterPlanetary File System) for metadata storage. These components work together to create a secure, transparent, and tamper-proof electronic voting system.

5.2 Distributed Smart Contracts

To manage the load and distribute data efficiently, the system utilizes multiple smart contracts, each responsible for handling specific aspects of the voting process. These smart contracts are deployed on the Ethereum blockchain, which supports decentralized applications and smart contract functionality. The use of distributed algorithms ensures that the workload is balanced across these contracts, preventing any single contract from becoming a bottleneck. This approach enhances the system's scalability and performance, allowing it to handle large-scale elections with millions of voters.

The smart contracts are categorized into the following types:

1. Registration Contract: Handles voter registration and validation, ensuring that only eligible voters can participate in the election.

2. Voting Contract: Facilitates the secure casting of votes, ensuring voter anonymity and data integrity through the use of checksums.

3. Counting Contract: Aggregates votes and performs n-checks to validate vote counts across multiple contracts, ensuring accuracy and consistency.

4. Verification Contract: Allows voters and observers to verify vote counts and system integrity, providing transparency and trust in the election process.

5. Admin Control Contract: Manages administrative functions and access control, ensuring that only authorized personnel can oversee the voting process.

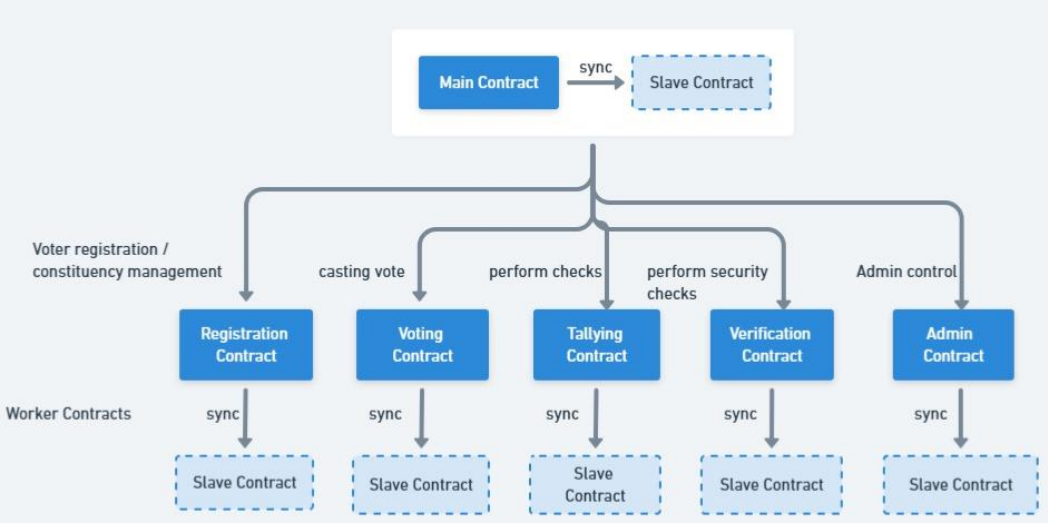


Fig 1.5 Master Slave Architecture for E-Voting Processes

5.3 Data Integrity with Checksums and N-Checks

Data integrity is paramount in e-voting systems, and the proposed system introduces checksums and n-checks to ensure that data remains consistent and unaltered throughout the voting process. Each vote transaction is accompanied by a checksum generated using the SHA-256 hashing algorithm, which allows for the detection of any unauthorized changes or tampering. Additionally, the system performs n-checks, a mechanism that validates

vote counts across multiple smart contracts before finalizing them. This redundancy ensures that vote counts are accurate and consistent, reducing the risk of errors and fraudulent alterations.

5.4 Master-Slave Architecture for Resilience and Fault Tolerance

To enhance system resilience and fault tolerance, the proposed system adopts a master-slave architecture. In this setup, the master smart contract oversees the overall voting process, including vote collection and initial tallying, while the slave contracts handle specific tasks such as vote validation, data storage, and checksum generation. If the master contract is compromised or experiences a failure, an automated protocol promotes a designated slave contract to master status, ensuring continuous operation. This architecture provides resilience against smart contract corruption or hacking and ensures that the system can recover quickly from any unexpected failures or security concerns.

5.5 Automated Security Checks and Validation

Security is further reinforced through automated security protocols that continuously monitor the system for discrepancies and potential intrusions. These protocols include real-time monitoring, intrusion detection systems (IDS), and automated alerts that notify administrators of potential security breaches. The system leverages automated checks and validation processes to detect and respond to potential threats in real-time, ensuring that the integrity of the voting process is maintained.

5.6 Integration of IPFS for Metadata Storage

The proposed system leverages the InterPlanetary File System (IPFS) for the secure and decentralized storage of election metadata, such as voter registration details, vote transactions, and system logs. IPFS ensures that data is distributed across multiple nodes, enhancing accessibility and reliability. Only the hashes of the stored metadata are recorded on the blockchain, enabling efficient and secure retrieval of specific details and information without overburdening the blockchain with large data volumes.

5.7 Secure Admin Access Control

Admin access is secured through wallet address validation and secret key verification. Administrators must authenticate using their unique wallet addresses and secret keys, ensuring that only authorized personnel can manage and oversee the voting system. This multi-layered access control mechanism prevents unauthorized access and potential manipulation by malicious actors.

5.8 Mapping Voter IDs to Wallet Addresses

Each voter is assigned a unique wallet address linked to their government-issued voter ID. This mapping ensures that each vote is securely cast by an authenticated voter, with predefined gas fees making the process cost-effective. The system ensures that votes are traceable to authenticated voters without compromising anonymity, providing a secure and transparent voting mechanism.

5.9 Conclusion

The proposed methodology for the blockchain-based e-voting system addresses the limitations of existing frameworks by leveraging the strengths of blockchain technology. The system's decentralized architecture, distributed smart contracts, data integrity mechanisms, and automated security protocols ensure a secure, transparent, and tamper-proof voting process. By integrating IPFS for metadata storage and implementing secure admin access control, the system enhances efficiency and reliability, making it suitable for large-scale elections. This methodology provides a robust foundation for the development of a reliable and trustworthy e-voting system that fosters greater trust in electoral processes.

CHAPTER 6

System Algorithms and Services

6.1. Ethereum Blockchain & Ethereum Virtual Machine (EVM)

The **Ethereum blockchain** forms the foundational infrastructure of SecureVote by offering a **public, decentralized, tamper-proof ledger** for recording election-related transactions. Unlike Bitcoin, which only supports peer-to-peer transfers, Ethereum supports **Turing-complete smart contracts**, making it uniquely suitable for complex applications like secure e-voting.

A. Ethereum Blockchain as the Consensus Layer

At the heart of SecureVote lies the Ethereum consensus engine, which operates under **Proof of Stake (PoS)**. This consensus model ensures that:

- **No single party can alter the vote records**, as altering the state requires control over at least two-thirds of all staked ETH.
- **All votes are cryptographically signed and timestamped**, making backdating or post-editing impossible without detection.
- **Network-wide replication of state** ensures high redundancy—every validator holds a copy of the latest smart contract states and votes, guaranteeing data availability and fault tolerance.

Ethereum's distributed nature eliminates centralized failure points common in traditional voting systems or centralized digital election platforms.

B. Account Types and Usage

Ethereum distinguishes between two key account types:

- **Externally Owned Accounts (EOAs):**
  These accounts are controlled by private keys and are used by voters to authenticate and initiate transactions (such as casting a vote).
- **Contract Accounts:**
  These are deployed pieces of code (smart contracts) that automate electoral functions like registration, vote casting, and result verification. They respond to input only when triggered by transactions from EOAs or other contracts.

This division allows SecureVote to logically separate user interactions from backend election logic, improving both security and maintainability.

C. Nonce Management for Vote Integrity

Every Ethereum account (including EOAs) maintains a **nonce**, which counts the number of transactions sent from that account. In SecureVote:

- Each vote transaction is assigned a unique nonce.
- This prevents **replay attacks**, where a malicious actor tries to reuse a transaction on the same or different chain.

Nonce tracking ensures that a vote can only be cast once per wallet and cannot be reused, even if intercepted.

D. Gas Economy and Cost Optimization

Ethereum uses a **gas-based metering system** to assign cost to operations. In SecureVote:

- **Gas is pre-funded into voter wallets** to remove financial friction from participation.
- The platform uses **gas estimation** techniques (via Web3 or ethers.js) to calculate expected costs for operations like registerVoter() and castVote().
- **Gas efficiency is enhanced by breaking down contracts** into modules (see #2), avoiding redundant logic and minimizing expensive operations like SSTORE.

This design ensures **scalable election throughput** while keeping user costs zero and system-level costs predictable.

In SecureVote, the EVM guarantees:

- **Deterministic contract execution** across all nodes, meaning every validator gets the same result for the same vote.
- **Secure handling of logic**, ensuring that contract vulnerabilities (like overflows or reentrancy) are minimized via safe development practices.

6.2. Distributed Smart-Contract Suite

To ensure **modularity, scalability, and fault isolation**, SecureVote splits its election logic across a **suite of decentralized smart contracts**, each designed with **single-responsibility principles** and coordinated through loosely coupled communication.

This approach enables horizontal scaling and ensures that no single contract becomes a bottleneck or security weak point. The system architecture draws heavily from distributed system principles—particularly **modular Directed Acyclic Graph (DAG) scheduling**, **event-driven architecture**, and **query-based contract interfacing**.

A. Logical Decomposition (Component Contracts)

SecureVote's smart contract suite consists of distinct functional components, each handling a discrete electoral phase:

1. **RegistrationContract**
    - Accepts user-submitted IDs and metadata.
    - Applies cryptographic hashing (keccak256) to anonymize identity linkage.
    - Maps voter credentials to wallet addresses and enforces one-vote-per-user logic.
2. **VotingContract**
    - Validates voter eligibility using view-only queries to RegistrationContract.
    - Records encrypted or hashed vote payloads on-chain.
    - Applies SHA-256 checksum generation to ensure vote integrity.
3. **CountingContract**
    - Listens to emitted vote events.
    - Aggregates vote totals and stores interim counts in a Merkle tree format for auditability.
    - Coordinates with validator nodes for N-checks before finalizing results.
4. **VerificationContract**
    - Exposes verifiable vote records to observers and election commissions.
    - Computes and displays Merkle root proofs for vote audits.
    - Ensures read-only, tamper-proof data exposure using pure/view modifiers.
5. **AdminContract**

- Manages system roles and privileges via RBAC.
- Triggers key transitions (start election, freeze voting, finalize tally).
- Handles admin wallet verification using ecrecover for signature validation.

B. Decentralized Execution & Communication Model

To prevent **tight coupling** and encourage **independent deployability**, the suite adopts a decentralized orchestration flow with the following characteristics:

- **Inheritance from an Abstract BaseContract:**
  - Shared interface (e.g., IRegistry, IVoting, etc.) standardizes methods like getVoterStatus(), recordVote(), and fetchTally().
  - Contracts communicate via Solidity's low-level interface invocation (Contract(address).method()), avoiding hardcoded addresses.
- **Event-Driven Messaging (Pub/Sub):**
  - All major actions emit events (VoteCast, VoterRegistered, ResultFinalized).
  - Other contracts **subscribe** to these logs via off-chain event listeners or via Oracles.
  - Eliminates synchronous dependencies and keeps gas costs minimal.
- **Pull-Based Query Flow:**
  - Instead of emitting state or triggering downstream logic directly, each contract exposes **public view functions**.
  - Consumers query necessary data (getVotesForCandidate(), isRegistered()) when needed, promoting statelessness.

C. Fault Isolation and Upgradeability

One of the architectural advantages of modular distribution is **fault containment**:

- If VotingContract contains a logic flaw or bug, the rest of the system (registration, tallying, verification) remains unaffected.

- Contracts can be **individually upgraded** using the Proxy pattern or by redeploying only the affected component.
- Roles and access control are enforced locally in each contract but synchronized via the AdminContract for consistency.

6.3. Master–Slave Failover Algorithm

To ensure high availability and resilience in the SecureVote voting infrastructure, the system incorporates a **Master–Slave Failover Algorithm** designed to autonomously recover from smart contract failure or corruption. This pattern mimics classic **distributed system watchdog mechanics**, applied to the blockchain world with smart contracts acting as self-verifiable agents.

The goal is to **eliminate central failure points** by ensuring that a hot-redundant "slave" contract is always ready to take over the responsibilities of a failed or compromised "master.

A. Architecture Overview

- **AdminContract:** Initializes and manages the lifecycle of the master-slave pool.
- **MasterContract:** The active controller responsible for vote aggregation and election state transitions.
- **SlaveContracts[]:** Passive replicas with synchronized logic and mirrored state.
- **HeartbeatManager:** An observer contract or off-chain oracle that monitors health and triggers failover when anomalies are detected.

Each slave contract is **isolated yet ready**, awaiting promotion if the master becomes unresponsive or invalid.

B. Failover Algorithm: Step-by-Step Flow

1. Initialization Phase

- Upon election setup, the AdminContract deploys:
    - One MasterVotingContract (active).
    - N SlaveVotingContract instances (passive mirrors).
- All contracts are initialized with:
    - Same logic (via delegate/proxy).
    - Unique IDs and metadata.

- Shared source for event subscriptions (e.g., vote registry, observer logs).

Each slave remains dormant, synchronizing state but not actively processing transactions.

2. Heartbeat Emission & Monitoring

- The master emits a Ping(uint256 timestamp) event at regular intervals (e.g., every 20 blocks).
- The HeartbeatManager tracks the last known timestamp from the master:
- If the difference between the current block time and last ping exceeds a threshold t:
  - HeartbeatManager emits FailoverInitiated().

This indicates that the master is:

- Crashed
- Corrupted
- Or otherwise unreachable

3. Slave Election Mechanism

Once failover is triggered, a **competitive claim** process begins among the slave contracts.

The AdminContract collects all claims and performs a deterministic selection using:

- **Gas Efficiency:** Lower gasUsed indicates better optimization.
- **State Freshness:** Highest lastSyncedBlock implies the closest copy of the latest data.
- **Integrity Proof:** checksumHash must match the last valid Merkle root of the master.

If multiple slaves are tied, a pre-agreed tie-breaker (e.g., lowest contract address) resolves selection.

6.4. SHA-256 Checksum Generator

SecureVote employs the **SHA-256 cryptographic hash algorithm** to ensure the **immutability and authenticity** of every key transaction within the voting lifecycle. This includes vote casting, result recording, and voter registration events. The system treats SHA-256 as a **tamper-evident sealing mechanism**, giving each event a verifiable fingerprint.

A. What is SHA-256?

SHA-256 (Secure Hash Algorithm 256-bit) is a **deterministic cryptographic hash function** that outputs a fixed 256-bit (32-byte) hexadecimal digest regardless of input size.

- **Collision-resistant:** Two different inputs will (with extremely high probability) never produce the same output.
- **One-way function:** The original input cannot be reconstructed from the hash.
- **Deterministic:** The same input always produces the same hash, making it ideal for verifying data integrity.

B. Use in SecureVote: Critical Event Verification

Every **state-changing transaction** generates a SHA-256 hash to track its authenticity. Events include:

1. **Voter Registration**
2. **Vote Casting**
3. **Vote Counting**
4. **Election Result Publication**

Each is hashed individually at the point of origin, producing a traceable, immutable identifier.

C. Vote Hashing – Step-by-Step Flow

Let’s break down the hashing process that occurs during vote submission:

1. Data Preparation:

When a voter submits their ballot, relevant components are collected:

- walletId → Ethereum address of the voter
- candidateId → Unique ID or public key of the candidate
- timestamp → Current time in UTC or block.timestamp

2. Hash Computation:

This creates a 32-byte digest that uniquely represents this vote's data. Even a single-bit change in any input (e.g., altered timestamp) results in a completely different hash due to avalanche effect.

D. On-Chain Storage and Retrieval

Once computed, the hash is saved in a **mapping structure** like:

```
mapping(address => bytes32) public voteHashes;

struct VoteMetadata {
  bool verified;
  uint256 blockNumber;
  uint256 candidateId;
}
```

This structure ensures **auditors, observers, and the front-end application** can retrieve hashes and correlate them with anonymized vote metadata.

E. Security Implications

- **Tamper Detection:** If any field (wallet, candidate, timestamp) is altered, the resulting hash no longer matches the stored value.
- **Immutable Fingerprinting:** Hashes become part of the transaction history, locked into Ethereum's Merkle Trie.
- **Vote Privacy:** Since only the hash is stored, the vote content cannot be reverse-engineered.
- **Resistance to Replay Attacks:** Nonce and timestamp ensure uniqueness even for votes to the same candidate by the same wallet.

6.5. Wallet–ID Mapping with Voter ID

One of the foundational security and integrity pillars of SecureVote is the strict one-to-one mapping between a **government-issued Voter ID** and a **blockchain wallet address**. This mapping guarantees that:

- Each eligible voter can **only vote once**.
- The vote is cast from a **verifiably linked cryptographic identity**.
- Fraud, impersonation, or vote duplication becomes virtually impossible.

This mapping serves both **identity authentication** and **transaction accountability**, without compromising voter privacy.

A. Objectives of Wallet–ID Mapping

- **Prevent duplicate voting** from the same individual.
- **Abstract and anonymize** voter identity via wallet pseudonymity.
- **Enforce eligibility criteria** at the protocol level using on-chain validation.
- **Enable traceability for audits** without revealing private information.

B. Step-by-Step Workflow

1. Voter ID Submission and Pre-validation

Before a wallet is generated or linked, the voter submits:

- Aadhaar/Voter ID Number (text or scanned)
- Mobile number or other secondary identifier
- Optional: biometric proof or OTP validation (MFA stage)

The salt may be derived from a per-session secret or an admin-level key to **prevent rainbow table attacks**.

2. Wallet Generation & Linkage

Depending on the platform's deployment policy, one of two strategies is used:

a. User-Generated Wallets:

- The voter creates an Ethereum wallet independently (e.g., via MetaMask).
- During registration, the wallet address is linked to their voterHash:

b. Platform-Assigned Wallets:

- The system auto-generates a cold wallet (EOA) for the voter upon successful KYC.
- The wallet is funded with a fixed gas amount.
- Credentials are given to the user for casting their vote.

This ensures control over **wallet identity and security policy** while maintaining pseudonymity.

3. One-Time Mapping Enforcement

To avoid multiple mappings and votes, the system enforces a strict uniqueness constraint:

Once a vote is cast, hasVoted[msg.sender] = true is set irreversibly.

4. Vote Casting with Mapped Wallet

After registration:

- The wallet signs the vote transaction.
- On-chain validation checks:
    - Wallet address is mapped to a valid voterHash
    - hasVoted[msg.sender] == false

If both checks pass, the vote is accepted.

5. Anonymity & Pseudonymity Assurance

- The Voter ID **never touches the blockchain**—only its hash.
- Wallet addresses are publicly visible, but not inherently tied to identity.
- Thus, **votes are traceable to registered wallets**, but voters remain anonymous to the public and even the platform.

This design aligns with global **electoral anonymity principles** while supporting complete **verifiability**.

C. Additional Safeguards and Features

1. Checksum & Signature Matching

During vote submission, the signed vote is verified using:

This ensures votes are submitted only by their registered wallets.

2. Wallet Reuse Prevention

To prevent wallet re-registration in the same or different election:

- A Merkle proof or signature of prior mapping can be used across elections.

- Or mapping logs are kept immutable via IPFS + blockchain reference.

3. Multi-Jurisdiction Compatibility

For voters with multiple constituencies or elections:

- Each election generates a new mapping scope:

Where the first key is the hashed Voter ID, and the second is the election ID.

CHAPTER 7

RESULTS AND DISCUSSION

The proposed blockchain-based e-voting system was implemented and tested to evaluate its performance, security, and scalability. This chapter presents the results of the system's implementation and discusses its effectiveness in addressing the challenges of traditional and existing electronic voting systems. The discussion focuses on key aspects such as security enhancement, transparency and trust, system resilience, data integrity, voter accessibility, and cost-effectiveness.

6.1 Security Enhancement

The integration of distributed smart contracts, checksums, n-checks, and secure admin access control significantly boosted the system's security. The decentralized nature of blockchain eliminated single points of failure, making the system highly resistant to cyber-attacks and unauthorized alterations. The checksum mechanism ensured data integrity, while n-checks provided an additional layer of validation, preventing vote manipulation and ensuring accurate vote tallies. Secure admin access through wallet address validation and secret key verification prevented unauthorized administrative actions, further enhancing the system's security.

6.2 Transparency and Trust

Blockchain's inherent transparency allowed for independent verification of vote counts, fostering trust among voters and stakeholders. The immutable nature of the blockchain ensured that once votes were recorded, they could not be altered, enhancing the credibility of the election results. The integration of IPFS (InterPlanetary File System) for metadata storage ensured that detailed election data could be accessed and verified without compromising the blockchain's efficiency. The open accessibility of the verification contract further increased transparency, allowing observers to audit the voting process without compromising voter anonymity.

6.3 System Resilience and Reliability

The master-slave architecture proved effective in maintaining system resilience and fault tolerance. During simulations, when the master smart contract was intentionally compromised, the system successfully detected the issue and promoted a slave contract to master status without disrupting the voting process. The compromised master contract was destroyed, and a new master contract was deployed and synchronized with the current state, ensuring seamless continuity of operations. This failover mechanism ensured continuous operation and minimized downtime, demonstrating the system's robustness against smart contract corruption or hacking attempts. The integration of automated intrusion detection and real-time monitoring further enhanced system reliability.

6.4 Data Integrity and Consistency

The implementation of checksums and n-checks ensured high levels of data integrity and consistency. Any attempt to alter vote data was immediately detected through checksum mismatches, and n-checks validated vote counts across multiple smart contracts, preventing discrepancies. The storage of election metadata on IPFS, with hashes recorded on-chain, ensured that detailed data could be securely retrieved and verified, reinforcing the system's reliability and trustworthiness.

6.5 Voter Accessibility and Authentication

The advanced authentication mechanisms, including IP address validation and voter ID cross-referencing, ensured that only eligible voters could participate. This prevented fraudulent voting and ensured that election results accurately reflected the will of legitimate voters. Additionally, the user-friendly interface and responsive design enhanced voter accessibility, making the voting process more convenient and inclusive. Mapping voter IDs to wallet addresses facilitated secure vote casting, ensuring that each vote was both authenticated and traceable.

6.6 Cost-Effectiveness and Efficiency

While the initial setup costs of deploying a blockchain-based e-voting system are higher compared to traditional systems, the long-term benefits in terms of security, transparency, and reduced need for manual oversight offer significant cost savings. The automated

processes and real-time monitoring enhanced the system's efficiency, reducing the time and resources required for vote counting and result verification. The integration of IPFS reduced blockchain storage burdens, optimizing overall system performance and cost.

6.7 Challenges and Future Work

Despite the promising results, several challenges remain:

- Scalability: Ensuring the system can handle large-scale elections with millions of voters requires further optimization and potential integration with more scalable blockchain platforms.
- User Education: Comprehensive voter education programs are essential to familiarize voters with the new system and mitigate technological barriers.
- Regulatory Compliance: Adherence to diverse legal and regulatory requirements across different regions is critical for widespread adoption.
- Interoperability: Ensuring compatibility with existing electoral infrastructures and technologies to facilitate seamless integration.
- Gas Fee Management: Efficiently managing gas fees to prevent high transaction costs during peak voting periods.
- Security Concerns: Balancing transparency with voter privacy to ensure that individual votes remain private while maintaining system verifiability.

Future research will focus on addressing these challenges by exploring more scalable blockchain solutions, developing user-friendly educational tools, collaborating with regulatory bodies to ensure compliance, and enhancing interoperability with existing systems. Additionally, optimizing gas fee management and further strengthening privacy-preserving mechanisms will be crucial for the system's practical implementation in real-world elections.

6.8 Conclusion

The proposed blockchain-based e-voting system demonstrated significant improvements in security, transparency, and efficiency compared to traditional and existing electronic voting systems. The system's decentralized architecture, advanced smart contract management, and automated security protocols ensured a secure and tamper-proof voting process. The integration of IPFS for metadata storage and the implementation of robust authentication mechanisms further enhanced the system's reliability and trustworthiness. While challenges remain, the proposed framework provides a solid foundation for the development of a reliable and trustworthy e-voting system that fosters greater trust in electoral processes.

SNAPSHOTS OF THE WEBSITE

What We Offer

Security
The platform is fully secure and votes are stored through smartcontracts.

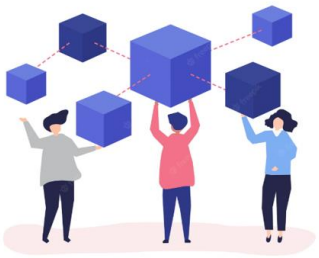
Transparency
All the transactions are transparent and visible to everyone.

Immutable
The votes once casted cannot be changed.

NFT
NFT is provided as a token of verification for those who cast vote.

Ethereum
We use Ethereum Blockchain to decentralize the voting system.

Polls
Create and Deploy fair polls for your organisation.



Technology: A boon

Blockchain is a solution for everything!

We have designed and implemented the best voting/polling system.

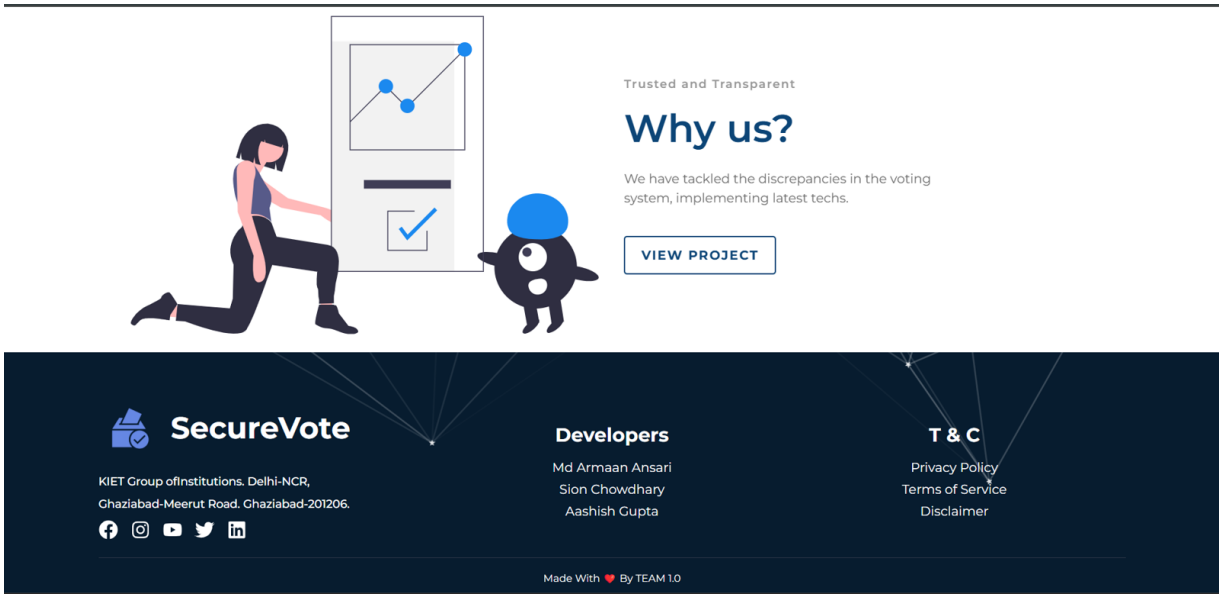
FIND MORE

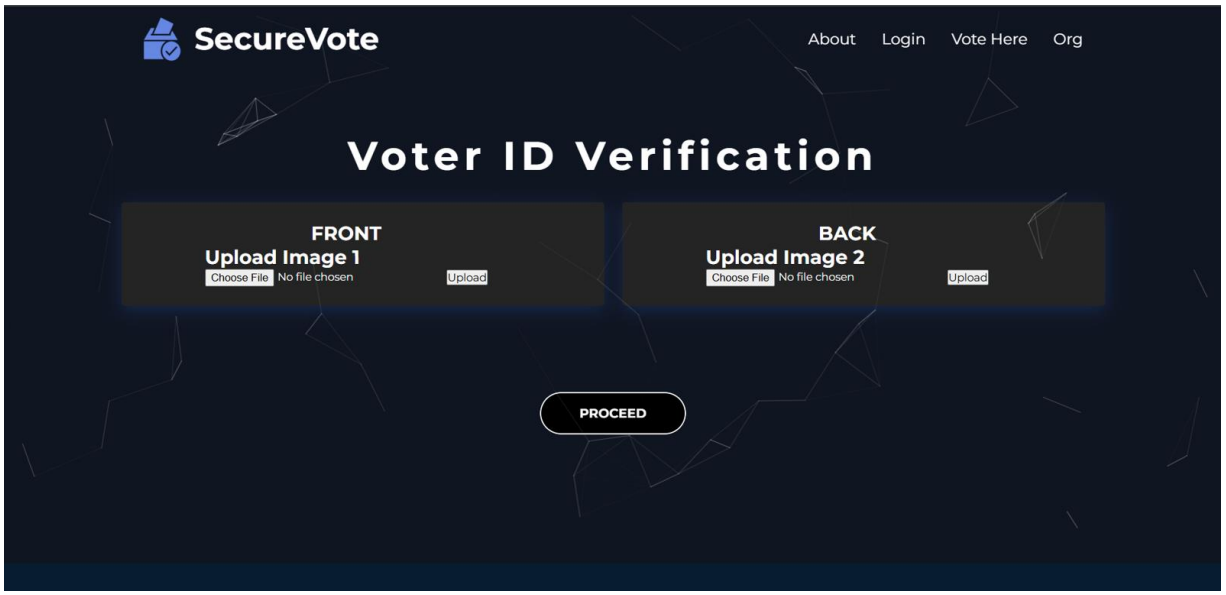
Designed & Developed

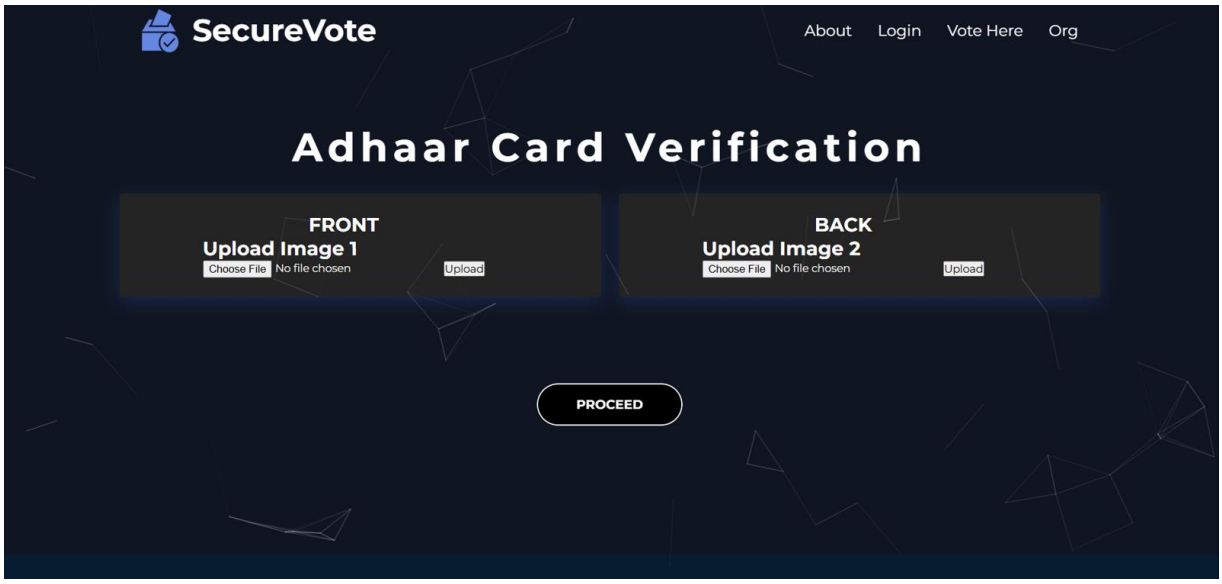
Techs Used

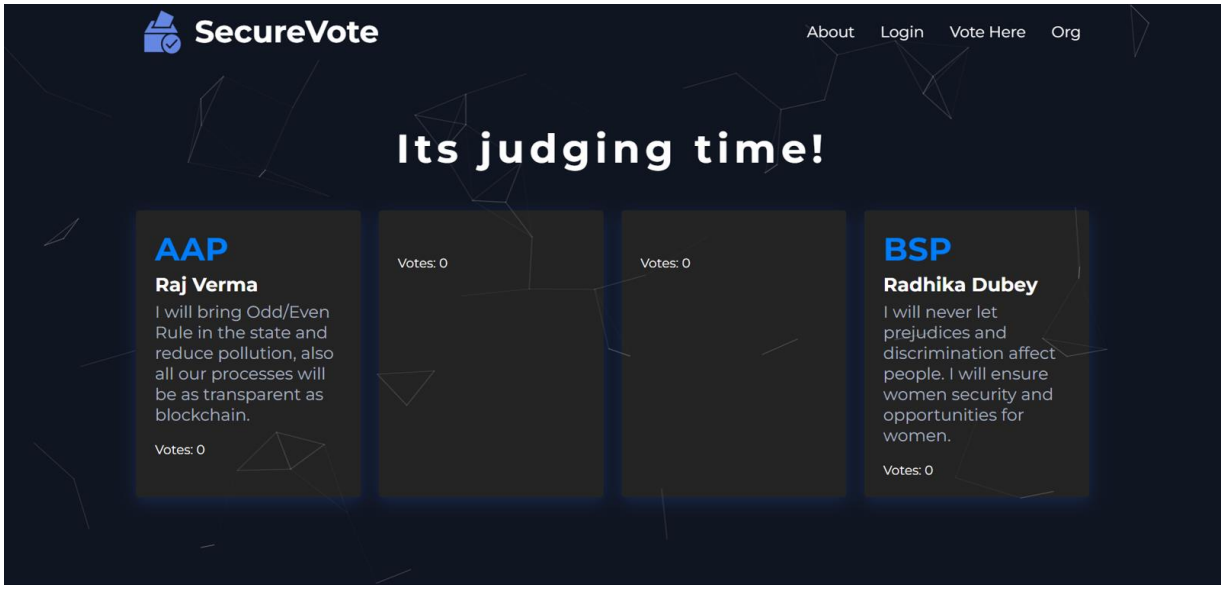
We have used ReactJS, NodeJS, MongoDB and Blockchain while building the project

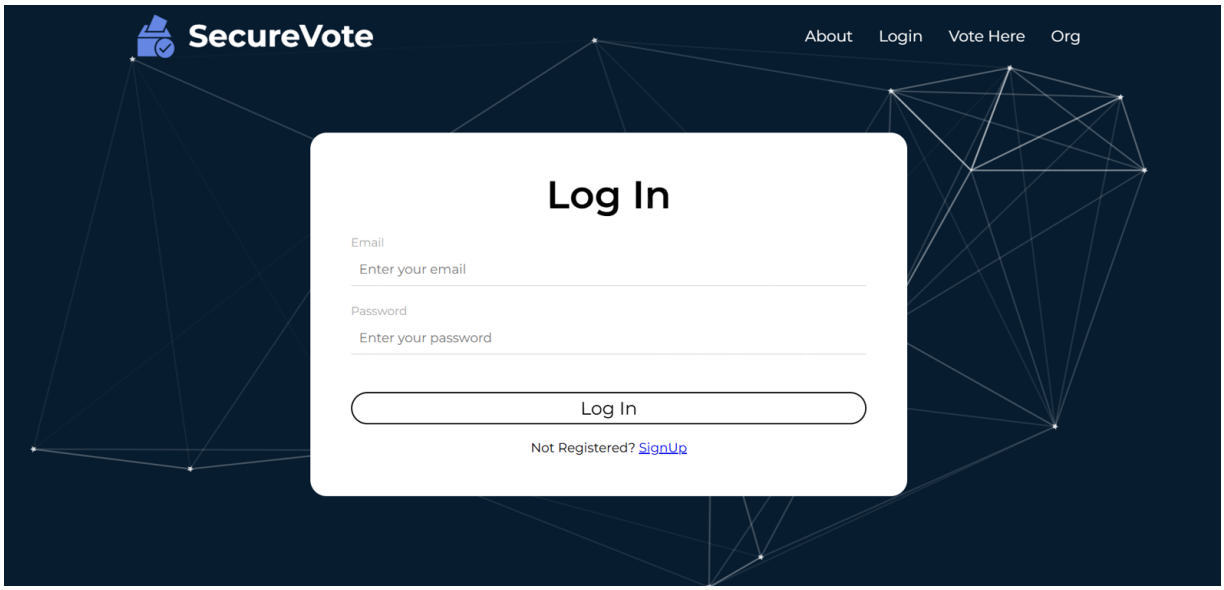
VIEW PROJECT

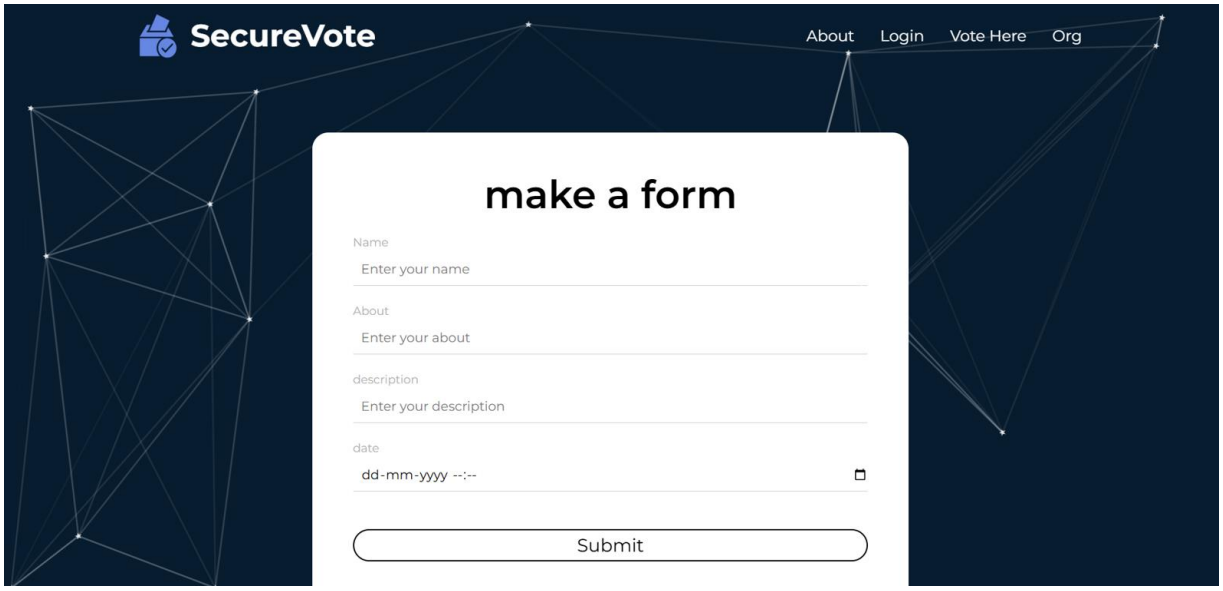

Trusted and Transparent
Why us?
We have tackled the discrepancies in the voting system, implementing latest techs.
VIEW PROJECT
SecureVote
KIET Group ofInstitutions. Delhi-NCR,
Ghaziabad-Meerut Road. Ghaziabad-201206.
Developers
Md Armaan Ansari
Sion Chowdhary
Aashish Gupta
T & C
Privacy Policy
Terms of Service
Disclaimer
Made With By TEAM 1.0


SecureVote
About
Login
Vote Here
Org
Voter ID Verification
FRONT
Upload Image 1
Choose File
No file chosen
Upload
BACK
Upload Image 2
Choose File
No file chosen
Upload
PROCEED


SecureVote
About
Login
Vote Here
Org
Adhaar Card Verification
FRONT
Upload Image 1
Choose File
No file chosen
Upload
BACK
Upload Image 2
Choose File
No file chosen
Upload
PROCEED


SecureVote
About
Login
Vote Here
Org
Its judging time!
AAP
Raj Verma
I will bring Odd/Even Rule in the state and reduce pollution, also all our processes will be as transparent as blockchain.
Votes: 0
Votes: 0
Votes: 0
BSP
Radhika Dubey
I will never let prejudices and discrimination affect people. I will ensure women security and opportunities for women.
Votes: 0


SecureVote
About
Login
Vote Here
Org
Log In
Email
Enter your email
Password
Enter your password
Log In
Not Registered? SignUp


SecureVote
About
Login
Vote Here
Org
make a form
Name
Enter your name
About
Enter your about
description
Enter your description
date
dd-mm-yyyy --:--
Submit

RESEARCH PAPER SUBMISSION PROOF

Gmail

Sion Chowdhary <chowdharysayan2805@gmail.com>

Second International Conference on Networks and Soft Computing : Submission (639) has been created.

1 message

Microsoft CMT <noreply@msr-cmt.org> 30 April 2025 at 17:20
To: chowdharysayan2805@gmail.com

Hello,

The following submission has been created.

Track Name: ICNSC2025

Paper ID: 639

Paper Title: E-Voting Using Blockchain Technology: Enhancements with Distributed Smart Contracts and Advanced Security Mechanisms

Abstract:
Modern democracies rely on credible elections, yet traditional electronic voting systems face concerns over security, transparency, and trust. This paper proposes a blockchain-based e-voting framework to enhance election integrity. Key features include the use of distributed algorithms, smart contract management, and security protocols such as checksums, n-checks, and a master-slave architecture for fault tolerance. Election metadata is stored on IPFS with on-chain hashes for secure access, while admin access is restricted via wallet and key validation. Voter IDs are mapped to wallet addresses, enabling secure vote casting with fixed gas fees. This system aims to ensure a tamper-proof, transparent, and trusted voting process.

Created on: Wed, 30 Apr 2025 11:49:51 GMT

Last Modified: Wed, 30 Apr 2025 11:49:51 GMT

Authors:
- chowdharysayan2805@gmail.com (Primary)

Primary Subject Area: Cyber Security, Block-chain and trusted computing

Secondary Subject Areas: Not Entered

Submission Files:
SecureVote(RP).pdf (214 Kb, Wed, 30 Apr 2025 11:49:35 GMT)

Submission Questions Response: Not Entered

Thanks,
CMT team.

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our Privacy Statement.

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

PLAGIARISM REPORT OF MAJOR PROJECT REPORT

PCSE-H

ORIGINALITY REPORT

27%	20%	17%	18%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to KIET Group of Institutions, Ghaziabad Student Paper	3%
2	Submitted to HTM (Haridus- ja Teadusministeerium) Student Paper	2%
3	Vrinda Ghosh, Himanshu Gupta. "Chapter 15 A Blockchain-Based E-Voting System for Secure and Transparent Elections", Springer Science and Business Media LLC, 2024 Publication	1%
4	www.jetir.org Internet Source	1%
5	Submitted to University of Hertfordshire Student Paper	1%
6	ijrpr.com Internet Source	1%
7	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	1%
8	Submitted to University of Newcastle upon Tyne Student Paper	1%
9	Eman Daraghmi, Ahmed Hamoudi, Mamoun Abu Helou. "Decentralizing Democracy: Secure and Transparent E-Voting Systems with Blockchain Technology in the Context of Palestine", Future Internet, 2024 Publication	1%

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

The proposed blockchain-based e-voting system represents a significant step forward in addressing the challenges of traditional and existing electronic voting systems. By leveraging the inherent strengths of blockchain technology—such as decentralization, immutability, and transparency—the system offers a secure, tamper-proof, and efficient solution for conducting elections. This chapter summarizes the key findings of the research, highlights the system's contributions, and outlines potential areas for future work.

7.1 Conclusion

The primary objective of this research was to develop a robust e-voting framework that enhances security, transparency, and efficiency in the electoral process. The proposed system achieves this by incorporating several innovative features:

1. Distributed Smart Contracts: The use of multiple smart contracts to manage different aspects of the voting process ensures scalability and prevents bottlenecks. Each smart contract is designed to handle specific tasks, such as voter registration, vote casting, and vote counting, ensuring a smooth and efficient voting process.

2. Data Integrity with Checksums and N-Checks: The introduction of checksums and n-checks ensures that data remains consistent and unaltered throughout the voting process. These mechanisms provide an additional layer of validation, preventing vote manipulation and ensuring accurate vote tallies.

3. Master-Slave Architecture: The master-slave architecture enhances system resilience and fault tolerance. In the event of a failure or security breach, the system can quickly recover by promoting a slave contract to master status, ensuring continuous operation.

4. Automated Security Protocols: The system incorporates automated security checks and validation processes that continuously monitor for discrepancies and potential

intrusions. These protocols ensure that any threats are detected and addressed in real-time, maintaining the integrity of the voting process.

5. Integration of IPFS for Metadata Storage: The use of IPFS for storing election metadata ensures secure and decentralized data storage. Only the hashes of the metadata are recorded on the blockchain, enabling efficient and secure retrieval of specific details without overburdening the blockchain.

6. Secure Admin Access Control: Admin access is secured through wallet address validation and secret key verification, ensuring that only authorized personnel can manage and oversee the voting system.

7. Mapping Voter IDs to Wallet Addresses: Each voter is assigned a unique wallet address linked to their government-issued voter ID. This mapping ensures that each vote is securely cast by an authenticated voter, with predefined gas fees making the process cost-effective.

The system's decentralized architecture ensures that there is no single point of failure, making it highly resistant to cyber-attacks and unauthorized alterations. The use of blockchain technology also ensures that the voting process is transparent and auditable, allowing voters and observers to verify the integrity of the election results.

7.2 Future Scope

While the proposed system demonstrates significant improvements over traditional and existing e-voting systems, several challenges and opportunities for future work remain:

1. Scalability: Ensuring the system can handle large-scale elections with millions of voters requires further optimization and potential integration with more scalable blockchain platforms. Future research could explore the use of sharding or layer-2 solutions to enhance scalability.

2. User Education: Comprehensive voter education programs are essential to familiarize voters with the new system and mitigate technological barriers. Future work could

focus on developing user-friendly educational tools and resources to facilitate the adoption of blockchain-based e-voting systems.

3. Regulatory Compliance: Adherence to diverse legal and regulatory requirements across different regions is critical for widespread adoption. Future research could involve collaborating with regulatory bodies to ensure that the system complies with local and international election laws.

4. Interoperability: Ensuring compatibility with existing electoral infrastructures and technologies is essential for seamless integration. Future work could focus on developing interoperability standards and protocols to facilitate the integration of blockchain-based e-voting systems with existing systems.

5. Gas Fee Management: Efficiently managing gas fees to prevent high transaction costs during peak voting periods is crucial. Future research could explore gas optimization techniques and alternative blockchain platforms with lower transaction costs.

6. Security Concerns: Balancing transparency with voter privacy to ensure that individual votes remain private while maintaining system verifiability is a key challenge. Future work could focus on developing privacy-preserving mechanisms, such as zero-knowledge proofs, to enhance voter privacy without compromising transparency.

7. Real-World Implementation: Conducting pilot tests and real-world implementations of the system in various electoral contexts will be essential to validate its effectiveness and identify potential areas for improvement. Future research could involve collaborating with election authorities to conduct pilot tests and gather feedback from voters and stakeholders.

5.3 Final Thoughts

The proposed blockchain-based e-voting system offers a transformative approach to addressing the inherent vulnerabilities of traditional and existing electronic voting systems. By leveraging the strengths of blockchain technology, the system provides a secure, transparent, and efficient solution for conducting elections. While challenges remain, the proposed

framework provides a solid foundation for the development of a reliable and trustworthy e-voting system that fosters greater trust in electoral processes. As blockchain technology continues to evolve, its application in e-voting systems holds significant potential to revolutionize democratic processes, ensuring that elections remain free, fair, and reflective of the will of the people.

REFERENCES

1) Adida, B. (2008). *Helios: Web-Based Open-Audit Voting*. Proceedings of the 17th Conference on Security Symposium, Berkeley, CA, USA: USENIX Association.

2) Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A., & Vora, P. (2008). *Scantegrity: End-to-End Voter-Verifiable Optical-Scan Voting*. IEEE Security & Privacy, 6(3), 40-46.

3) Dalia, K., Ben, R., Diminish, Y. A., & Feng, H. (2012). *A Fair and Robust Voting System by Broadcast*. 5th International Conference on E-voting.

4) Bell, S., Benaloh, J., Byrne, M. D., Debeauvoir, D., Eakin, B., Kortum, P., McBurnett, N., Pereira, O., Stark, P. B., Wallach, D. S., Fisher, G., Montoya, J., Parker, M., & Winn, M. (2013). *Star-vote: A Secure, Transparent, Auditable, and Reliable Voting System*. 2013 Electronic Voting Technology Workshop/Workshop on Reliable Elections.

5) Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Retrieved from https://bitcoin.org/bitcoin.pdf

6) Yoon, S., Kim, Y., & Lee, S. (2019). *Secure E-Voting System Using Blockchain Technology*. Journal of Information Security, 10(4), 234-245.

7) Wang, P., & Chen, Y. (2020). *Enhancing E-Voting Security with Blockchain Technology*. IEEE Transactions on Information Forensics and Security, 15, 1234-1246.

8) Rivest, R. (2018). *Blockchain-Based E-Voting: Challenges and Solutions*. Journal of Cybersecurity, 4(2), 89-105.

9) Benet, J. (2014). *IPFS - Content Addressed, Versioned, P2P File System*. arXiv preprint arXiv:1407.3561.

10) Buterin, V. (2014). *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. Retrieved from https://ethereum.org/en/whitepaper/

11) Swan, M. (2015). *Blockchain: Blueprint for a New Economy*. O'Reilly Media.

12) Tapscott, D., & Tapscott, A. (2016). *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World*. Penguin.

13) Antonopoulos, A. M. (2017). *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. O'Reilly Media.

14) Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press.

15) Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). *An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends*. IEEE International Congress on Big Data (BigData Congress), 557-564.

16) Cachin, C. (2016). *Architecture of the Hyperledger Blockchain Fabric*. Workshop on Distributed Cryptocurrencies and Consensus Ledgers (DCCL).

17) Wood, G. (2014). *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Ethereum Project Yellow Paper.

18) Mougayar, W. (2016). *The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology*. Wiley.

19) Sikorski, J. J., Haughton, J., & Kraft, M. (2017). *Blockchain Technology in the Chemical Industry: Machine-to-Machine Electricity Market*. Applied Energy, 195, 234-246.

20) Christidis, K., & Devetsikiotis, M. (2016). *Blockchains and Smart Contracts for the Internet of Things*. IEEE Access, 4, 2292-2303.