Blockchain-Based E-Voting System

Dhruthana S ,Chandana PY ,Inchara BS ,Vandana G Raj ,Guruprasad YK

ABSTRACT

Voting is a cornerstone of democratic societies, but conventional methods like ballot papers and electronic voting machines often face challenges such as lack of transparency, vote tampering, voter impersonation, delays in results, and security vulnerabilities. Addressing these concerns is critical to building trust in electoral processes.

This project explores the use of blockchain technology to create a more secure, transparent, and reliable e-voting system. Leveraging the Ethereum blockchain, smart contracts written in Solidity manage voting logic securely and prevent duplicate votes. MetaMask is integrated to handle voter authorization and wallet management, ensuring secure participation. Supplementing the blockchain, Supabase is used as a cloud database to manage ancillary data and authentication processes. The system's front-end is built with React, providing an interactive and accessible user interface, while JavaScript handles the back-end operations that facilitate communication between the web app and blockchain smart contracts. The entire solution is implemented as a web-based application, enabling voters to participate easily from any internet-enabled device.

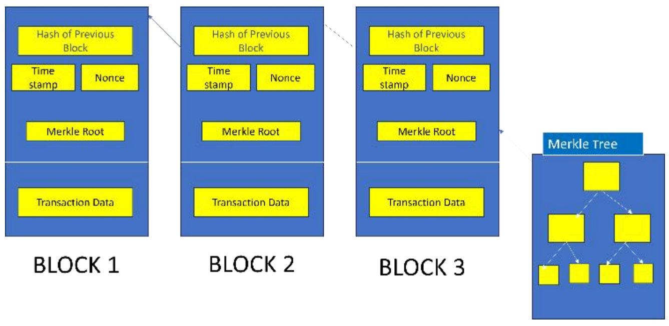
Through this architecture, the system combines blockchain's core strengths: decentralization, immutability, and transparency—with modern web technologies to deliver a practical e-voting platform. The project demonstrates both the potential benefits and current limitations of applying blockchain to electoral systems, paving the way toward more trustworthy and efficient voting in the future.

Keywords— E-voting, Smart-contracts, Blockchain, Ethereum

INTRODUCTION

Blockchain is an innovative decentralized digital ledger that records transactions securely, transparently, and immutably across a network of computers called nodes. Unlike traditional databases controlled by a central authority, blockchain distributes data across all participants, making it highly resistant to tampering or single points of failure. Transactions are grouped into blocks, each linked cryptographically to the previous block, forming a continuous chain. This chaining ensures that altering transaction history is nearly impossible, as it would require changing all subsequent blocks and gaining consensus from the majority of nodes.

Nodes in the blockchain network validate and agree on each transaction through a consensus mechanism, ensuring only legitimate data is added. The key properties of blockchain—decentralization, immutability, transparency, and security—make it a powerful technology for applications like cryptocurrencies, supply chains, and especially electronic voting systems, where trust and verifiability are critical.



This diagram shows a sequence of connected blocks containing transactions, timestamps, and cryptographic hashes linking each block to the previous one. Around the chain, multiple decentralized nodes validate transactions, representing the distributed and consensus-driven nature of blockchain technology.

Motivation

Voting forms the foundation of democracy; however, traditional voting methods—such as paper ballots and electronic voting machines—often encounter significant challenges, including vote tampering, identity fraud, lack of transparency, delayed results, and low voter turnout.. These challenges undermine public trust in elections and can threaten the legitimacy of democratic processes. While digital voting systems offer convenience, their centralized architectures expose them to security vulnerabilities and potential manipulation. Therefore, there is a strong need for a more secure, transparent, and reliable voting system that can increase voter confidence and participation while protecting the integrity of every vote cast. This motivates the exploration of new technologies to overcome the shortcomings of conventional voting methods.

Why Blockchain Helps

Blockchain technology offers a promising solution by providing a decentralized, tamper-proof ledger where votes can be recorded securely and transparently. Its core qualities—such as immutability, decentralization, and cryptographic security—prevent any single party from altering votes or manipulating outcomes without detection. By distributing voting records across numerous independent nodes, blockchain eliminates the need for a trusted central authority and ensures that all votes are permanently and publicly verifiable while maintaining voter anonymity. Moreover, blockchain enables automated vote counting and verification through smart contracts, reducing human errors and speeding up result tallying. This combination of transparency, security, privacy, and efficiency makes blockchain an ideal technology to build trustworthy and accessible electronic voting systems, addressing the critical challenges of modern elections.

LITERATURE SURVEY

1. Beulah Jayakumari, Lilly Sheeba, Maya Eapen, Jani Anbarasi, Vinayakumar Ravi, Suganya, and Malathy Jawahar (2024) [1] proposed a cloud-based hybrid blockchain e-voting system using PBFT consensus and timestamp-based authentication. Their approach improved authentication speed and reliability while ensuring transparency and tamper-resistance in elections.

2. Zunjarrao Aastha Girish, Kriti Ganeshan, Srushit More, Shrutam Tambe, and Vasavi Alumuri (2024) [2] explored blockchain and web engineering integration to develop an e-voting platform that improved security, transparency, and voter authentication by over 80%. Despite these gains, they acknowledged challenges like scalability and technical complexity.

3. Henry Ohize, Adeiza James Onumanyi, Buhari Umar, Lukman A. Ajao, Rabiu O. Isah, Eustace M. Dogo, Bello K. Nuhu, Olayemi M. Olaniyi, James G. Ambafi, Vincent B. Sheidu, and Muhammad M. Ibrahim (2024) [3] surveyed blockchain voting system architectures, cryptographic techniques, and implementations worldwide. They emphasized advances in privacy-preserving methods but pointed out infrastructural and regulatory hurdles.

4. Said El Kafhali (2024) [4] studied blockchain-based e- voting system requirements, presenting a permissioned blockchain framework integrating zero-knowledge proofs for privacy and verifiability. The study addressed design challenges, advocating scalable and user- accessible solutions.

5. Tulegen Aidynov, Nikolaj Goranin, Dina Satybaldina, and Assel Nurusheva (2024) [5] conducted a systematic review of cryptographic protocols and blockchain applications in e-voting. Their analysis highlighted improved security and trustworthiness with blockchain but underscored scalability and legal obstacles as major concerns.

6. Singh, Ganesh, Patil, Kumar, Rani, and Pippal (2023) [6] developed a secure voting platform using Ethereum smart contracts aimed at increasing election security, transparency, and efficiency. The integration of AI-based facial recognition improved voter authentication and system performance compared to traditional methods.

7. Afar, Ab Aziz, Shukur, and Hussain (2022) [7] performed a systematic literature review and meta-analysis on scalable blockchain-based e-voting systems. They analyzed cryptographic techniques, consensus protocols, and scalability methods, highlighting current challenges and proposing future research directions.

8. Yadav, Thombare, Urade, and Patil (2020) [8] designed an immutable blockchain voting system to ensure secure, transparent elections. Their model leveraged blockchain to improve trust and prevent vote tampering, demonstrating enhanced election result verifiability.

PROPOSED SYSTEM

The proposed blockchain-based e-voting system is designed to ensure secure, transparent, and tamper-proof elections. The system is divided into three major components: voter registration, vote casting, and vote tallying. Each component integrates blockchain and cryptographic techniques to maintain integrity, anonymity, and reliability.

In the registration phase, eligible voters are verified by an authorized government body using a digital identity system (e.g., Aadhaar, passport, or biometric data). Once verified, each voter is assigned a unique blockchain address and private key, which are required to participate in the election.

During the vote casting phase, voters use a secure interface (web or mobile application) to select their candidate. The vote is encrypted and submitted as a blockchain transaction. This ensures that votes cannot be altered or deleted once recorded. A smart contract is deployed to validate voter eligibility, prevent duplicate voting, and securely record each vote on the blockchain.

Finally, in the result declaration phase, the blockchain's immutable ledger allows for transparent vote counting. The smart contract automatically tallies the votes and publishes the results without the need for manual intervention. Since all transactions are stored on the blockchain, the results can be independently verified by auditors, stakeholders, or even the public, ensuring maximum trust in the electoral process.

The main features of the proposed system include:

Immutability – votes cannot be tampered with once recorded.

Decentralization – no single authority controls the system.

Anonymity – voters' identities remain hidden through cryptographic techniques.

Auditability – every vote can be traced and verified without compromising voter privacy.

SYSTEM DESIGN

1. Voter Registration: Each voter undergoes identity verification using national ID (such as Aadhaar, passport, or biometric data).

Once verified, a unique blockchain account (public/private key pair) is generated for the voter.

The public key acts as the voter's identifier, while the private key is used for secure vote casting.

A smart contract ensures that each voter can be registered only once.

2. Vote Casting: The voter logs into the voting application (web or mobile) and selects their candidate.

The vote is encrypted with cryptographic algorithms (e.g., RSA, ElGamal, or Homomorphic Encryption) to ensure voter privacy.

The encrypted vote is then submitted to the blockchain as a transaction.

A smart contract validates the transaction, checks voter eligibility, and prevents double voting.

3. Vote Storage: Once the vote is validated, it is stored in the blockchain's immutable ledger.

Since blockchain is decentralized, all nodes in the network hold a copy of the vote records, making tampering impossible.

4. Vote Tallying & Result Declaration: After voting ends, a smart contract automatically tallies votes recorded in the ledger.

Results are published directly on the blockchain for transparency.

Independent auditors can verify the results by reviewing blockchain transactions.

METHODOLOGY

The methodology follows a step-by-step approach to ensure security and usability:

1. Authentication & Authorization: Voters authenticate themselves using a government-issued ID and biometrics.

Once authenticated, the system issues a unique blockchain wallet for each voter.

2. Smart Contract Deployment: Smart contracts define rules for registration, vote casting, and tallying.

They are deployed on the blockchain before the election begins.

3. Vote Casting Process**:** Voter selects a candidate. The vote is encrypted with the election authority's public key to maintain privacy.

The smart contract verifies that the voter is valid and hasn't voted before. The vote transaction is then recorded on the blockchain.

4. Consensus Mechanism**:** The blockchain uses a consensus algorithm (such as Proof of Authority (PoA) in permissioned systems, **or** Proof of Work (PoW)/Proof of Stake (PoS) in public systems) to validate and store transactions.

This ensures that no unauthorized node can manipulate vote records.

5. Result Calculation: When the election ends, the smart contract executes a secure tallying function.

The contract decrypts votes (using private keys held by the election commission) and publishes the results.

ALGORITHM

Homomorphic encryption with 256 SHA hash algorithm for cloud security:

Homomorphic encryption (HE) and the SHA-256 hash algorithm are two distinct cryptographic tools that serve different purposes in cloud security. They are not used together in a direct or combined way to encrypt data. HE is for performing computations on encrypted data, while SHA-256 is for verifying data integrity and authenticity.

Homomorphic encryption is a powerful, yet computationally expensive, type of encryption that allows a third party (like a cloud provider) to perform calculations on encrypted data without ever having to decrypt it. The result of these operations is also an encrypted value which, when decrypted by the data owner, yields the same result as if the operations were performed on the original, unencrypted data.

Types of Homomorphic Encryption

Partially Homomorphic Encryption (PHE): Supports one type of operation (e.g., addition or multiplication) an unlimited number of times. The RSA cryptosystem is a well-known example that is multiplicatively homomorphic.

Somewhat Homomorphic Encryption (SHE): Supports a limited number of both additions and multiplications.

Fully Homomorphic Encryption (FHE): Supports an unlimited number of both additions and multiplications on the ciphertext. This is the most powerful and flexible type, allowing for any arbitrary computation.

How It Works (General Flow)

Key Generation: A user generates a public key and a private key. The public key is used for encryption, while the private key is used for decryption.

Encryption: The user encrypts their sensitive data using the public key before sending it to the cloud.

Cloud Computation: The cloud provider receives the encrypted data and an encrypted program or function. Using a special "evaluation key," the cloud performs the requested computations (e.g., adding two encrypted numbers) directly on the ciphertext.

Result Return: The cloud returns the encrypted result to the user.

Decryption: The user decrypts the result using their private key to get the final plaintext answer. The cloud provider never sees the unencrypted data.

Example: Secure Medical Data Analytics jfg

Imagine a hospital wants to run an AI algorithm on patient data to predict disease risk, but strict privacy regulations (like GDPR) prevent them from sharing unencrypted data with a third-party cloud analytics service.

Client (Hospital): Encrypts patient records (e.g., age, blood pressure, cholesterol levels) using an FHE scheme.

Cloud Server: Receives the encrypted data. It does not have the private key, so it can't see the individual patient information.

Computation: The cloud server runs the AI algorithm on the encrypted data. For instance, it might perform a series of encrypted additions and multiplications to calculate a risk score. The result is an encrypted risk score for each patient.

Return: The cloud sends the encrypted risk scores back to the hospital.

Decryption: The hospital decrypts the scores to see the plaintext results, like "Patient A: High Risk," without the cloud ever having seen the original, sensitive data.

SHA-256 Hash Algorithm

The SHA-256 (Secure Hash Algorithm 256-bit) is a cryptographic hash function, not an encryption algorithm. A hash function takes an input of any size and produces a fixed-length, unique string of characters called a hash value or digest.

Key Properties of SHA-256

One-Way: It's computationally infeasible to reverse the hashing process to get the original data from the hash value.

Deterministic: The same input will always produce the same hash.

Avalanche Effect: A tiny change in the input data results in a drastically different hash value.

Collision Resistance: It's extremely difficult to find two different inputs that produce the same hash.

How It Works (Simplified)

The SHA-256 process involves several steps:

Padding: The input message is padded to a specific length.

Initialization: The algorithm starts with a set of 8 predefined 32-bit words.

Processing: The padded message is processed in 512-bit chunks through a series of complex bitwise operations, logical functions, and additions.

Final Hash: After all chunks are processed, the final 256-bit hash value is produced.

Example: Verifying File Integrity t_

When a user downloads a large file from a cloud server, they can use its SHA-256 hash to ensure the file hasn't been corrupted or tampered with.

Cloud Server: The server calculates the SHA-256 hash of the original, uncorrupted file and publishes it on the website.

Client: A user downloads the file.

Integrity Check: The user calculates the SHA-256 hash of the downloaded file on their local machine.

Comparison: The user compares the hash they computed with the one published by the server. If the hashes match, the user can be sure the file is identical to the one on the server and has not been altered during download.

The Role of HE and SHA-256 in Cloud Security

Homomorphic encryption and SHA-256 are not interchangeable; they solve different security problems:

Homomorphic Encryption addresses the problem of data confidentiality during computation. It ensures that data remains private even while being processed by an untrusted third party.

SHA-256 addresses the problem of data integrity and authenticity. It ensures that data hasn't been changed or corrupted and can be used to verify the identity of the sender.

For example, a cloud security system could use both. The system could store sensitive data using HE to enable secure analytics, while also storing the SHA-256 hash of a file's metadata to quickly verify its integrity. Using both techniques provides a comprehensive approach to securing different aspects of data in the cloud.

Detailed FHE Example: Collaborative Medical Research

Imagine multiple hospitals (Clients A, B, C) want to collaborate on a research project to identify new disease biomarkers by analyzing their combined patient data. However, they cannot directly share patient data due to strict privacy regulations. FHE provides a solution:

Key Generation (Each Client):

Each hospital (Client A, B, C) generates its own Public Key (PK), Secret Key (SK), and a special Evaluation Key (EK).

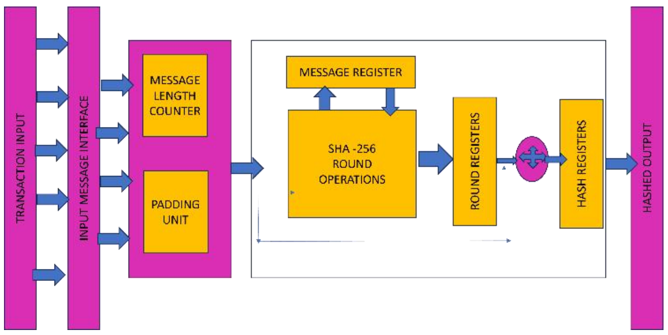
They would share their PKs with each other (for encrypting data that others might want to compute on) and their EKs with the cloud (for enabling the cloud to compute). They keep their SKs private. Encryption (Each Client):

Client A has patient data Data_A = {Age: 45, BMI: 28, MarkerX: 1.2}.

Client B has patient data Data_B = {Age: 52, BMI: 31, MarkerX: 0.9}.

Client C has patient data Data_C = {Age: 38, BMI: 25, MarkerX: 1.5}.

Each client encrypts their respective data using their own PK (or a shared PK if they agree on one for the research) and uploads the Ciphertext_A, Ciphertext_B, Ciphertext_C to a secure cloud server. The cloud server now holds encrypted patient records.



Cloud Computation:

The research team decides to compute the average MarkerX value across all patients and identify patients with MarkerX above a certain threshold, all while the data remains encrypted.

The cloud server, using the Evaluation Key (which allows homomorphic operations), performs the following:

Encrypted Sum = Encrypt(MarkerX_A) + Encrypt(MarkerX_B) + Encrypt(MarkerX_C)

Encrypted Count = Encrypt(1) + Encrypt(1) + Encrypt(1) (to get the count of patients)

Encrypted Average = Encrypted Sum / Encrypted Count (division might be approximated via multiplication by inverse, depending on the scheme)

Encrypted Threshold Check = Encrypt(MarkerX_i) > Encrypt(Threshold_Value) (this would involve more complex comparisons using encrypted gates)

All these operations are done on the ciphertexts. The cloud never sees the actual MarkerX values.

Result Return:

The cloud returns the Encrypted Average MarkerX and Encrypted Patient IDs Meeting Threshold to a designated researcher (e.g., Client A).

Decryption:

Client A uses their SK to decrypt the Encrypted Average MarkerX to get the plaintext average (e.g., 1.2).

They also decrypt the Encrypted Patient IDs to identify which patients meet the criteria, still without exposing individual MarkerX values to the cloud.

This example showcases how FHE enables collaborative data analysis without compromising individual data privacy, a critical requirement in sensitive fields like healthcare.

Detailed SHA-256 Example: Secure Software Distribution and Update

Consider a software company distributing updates for its application through a cloud content delivery network (CDN). It's crucial to ensure that users receive authentic and untampered software.

Software Company (Source):

Develops a new software update file (e.g., AppUpdate_v2.0.exe).

Calculates the SHA-256 hash of this exact file.

SHA256(AppUpdate_v2.0.exe) = "a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6q7r8s9t0u1v2w3x4y5z6a7b8c9d0e1f"

Publishes this SHA-256 hash on its official website or includes it in a signed manifest file.

Uploads AppUpdate_v2.0.exe to the cloud CDN for distribution.

User (Client):

Wants to update their software and downloads AppUpdate_v2.0.exe from the cloud CDN.

Also retrieves the published SHA-256 hash from the software company's official website.

Verification (Client):
The user's system (or a dedicated tool) calculates theSHA-256 hash of the downloaded AppUpdate_v2.0.exe file.

Scenario 1 (No Tampering): If the downloaded file is identical to the original, the calculated hash will be:

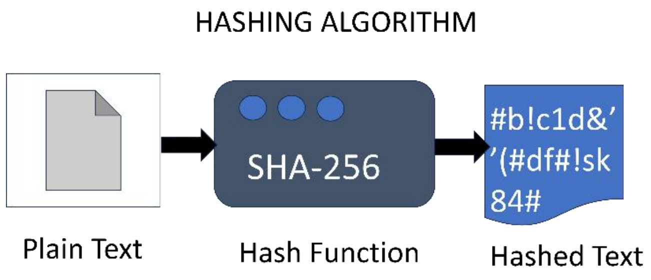
Calculated SHA256 = "a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6q7r8s9t0u1v2w3x4y5z6a7b8c9d0e1f"

Since Calculated SHA256 matches Published SHA256, the user knows the file is authentic and hasn't been corrupted or maliciously altered. They can proceed with the installation.

Scenario 2 (Tampering/Corruption): If a malicious actor intercepted the file on the CDN and injected malware, or if the download was corrupted, even a single bit change would result in a completely different hash:

Calculated SHA256 = "x9y8z7a6b5c4d3e2f1g0h9i8j7k6l5m4n3o2p1q0r9s8t7u6v5w4x3y3z2a1b0" (Example of a completely different hash)

Since Calculated SHA256 does NOT match Published SHA256, the user is immediately alerted that the file is compromised or corrupted and should not be trusted or installed.



This example clearly shows how SHA-256 is vital for ensuring the integrity and authenticity of data, especially in cloud environments where data might pass through many untrusted intermediaries.

By understanding these detailed flows and examples, you can appreciate how Homomorphic Encryption and SHA-256, while different in their mechanisms, both contribute significantly to building robust and secure cloud systems.

IMPLEMENTATION AND SIMULATION

The proposed blockchain-based e-voting system is implemented and tested on blockchain platforms such as Ethereum and Hyperledger Fabric, which provide the necessary infrastructure for decentralized, secure, and transparent applications. The implementation involves the creation of smart contracts to manage election processes, deployment of blockchain nodes for consensus, and development of a front-end application to enable voter interaction.

1. Ethereum Blockchain:

Solidity: For writing smart contracts that define voting rules (voter registration, vote casting, and tallying).

Ganache: A local Ethereum blockchain used for simulation and testing of smart contracts without spending real cryptocurrency.

MetaMask: A browser wallet extension that allows voters to manage private keys and interact with the blockchain.

Web3.js: A JavaScript library to connect the front-end voting application with the Ethereum blockchain.

2. Hyperledger Fabric:

Permissioned Blockchain suitable for government elections.

Chaincode (similar to smart contracts) written in Go/Node.js to enforce voting logic.

Fabric CA (Certificate Authority) for managing voter identities.

Docker Containers for deploying multiple blockchain nodes in a controlled environment.

3. Blockchain Platform:

Ethereum (Public Blockchain): Provides decentralization and immutability for small-scale elections (e.g., student unions, organizations).

Hyperledger Fabric (Permissioned Blockchain): Suitable for large-scale government elections requiring strict control over participants.

4. Smart Contracts**:**

Written in Solidity (Ethereum) or Chaincode (Hyperledger)**.**

Define election rules: voter registration, validation, vote casting, and tallying.

Automatically execute voting logic without human intervention.

5. Front-End Application:

A web or mobile interface for voters to interact with the system.

6. Wallet & Key Management:

Each voter is issued a blockchain wallet (MetaMask for Ethereum or Fabric CA for Hyperledger).

Keys are generated using elliptic curve cryptography to secure transactions.

IMPLMENTATION

1. System Setup: Deploy a local blockchain using Ganache (Ethereum) or Dockerized Fabric network (Hyperledger). Configure multiple nodes to simulate decentralization.

2. Smart Contract Deployment**:** Develop a contract to register voters and candidates. Contract prevents double voting and enforces election deadlines. Deploy contract on blockchain network.

3. Voter Registration**:** Voter identity verified by admin. Smart contract stores voter's public key in blockchain registry.

4. Vote Casting: Voter logs in using private key/wallet. Voter selects candidate → vote encrypted with election authority's public key. Transaction submitted to blockchain and validated by smart contract.

5. Consensus & Storage: Blockchain network validates the transaction using consensus (PoA/PoS for Ethereum, Raft for Hyperledger).Valid vote is permanently recorded in the ledger.

6. Vote Tallying: At election closure, smart contract automatically decrypts and counts votes .Results are stored on blockchain and published transparently.

SIMULATION

1. Objectives: The simulation of the blockchain-based e-voting system was carried out to assess its feasibility, security, and efficiency in a controlled setting. Since conducting a real-world election is not practical at the research stage, blockchain environments such as Ethereum and Hyperledger Fabric were used to replicate the voting process. The key aim of this simulation was to verify that blockchain can ensure immutability of votes, enable transparent auditing, and provide automated verification and result declaration through smart contracts.

2. Setup: The simulation was conducted in two blockchain environments. On Ethereum, a local test network was created using Ganache, with multiple simulated accounts assigned to represent voters. Smart contracts were developed in Solidity and deployed on this network, while MetaMask was used to manage cryptographic keys and Web3.js served as the interface for voter interaction. In the case of Hyperledger Fabric, a permissioned blockchain was established with several validating nodes acting as election

authorities. Chaincode was deployed to manage functions such as voter registration, vote casting, and vote counting, while the Fabric certificate authority was responsible for assigning unique digital identities to voters.

3. Voting process: The voting process in the simulation followed a series of secure steps. First, the election authority deployed the smart contract and registered eligible voters, providing each with a unique blockchain wallet containing a public and private key. When a voter logged in, they selected their preferred candidate and cast their vote through the application. Each vote was encrypted for privacy and digitally signed for authenticity before being transmitted as a transaction to the blockchain. The smart contract verified the voter's eligibility and ensured that no double voting occurred. Once validated, the vote was permanently stored in the blockchain ledger, making it immutable and auditable. At the conclusion of the election, the smart contract automatically tallied all votes and published the results, eliminating the risk of manipulation.

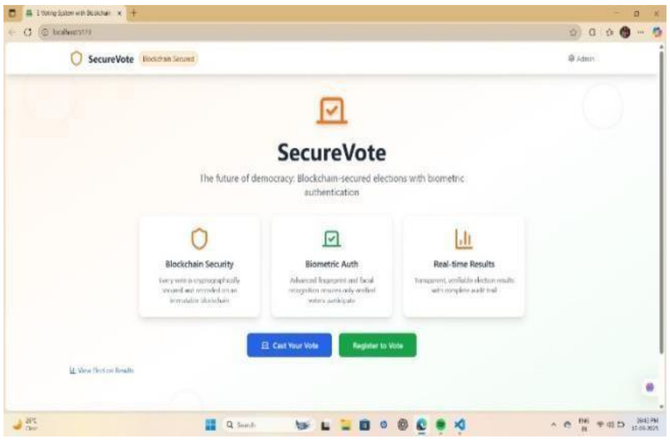
4. Scenarios: Different scenarios were simulated to test the performance of the proposed system. A small-scale election with around one hundred voters was conducted on Ethereum, demonstrating the system's suitability for organizational or local-level elections. A medium-scale election with over one thousand voters was simulated on Hyperledger Fabric, highlighting its ability to handle larger volumes of transactions efficiently. Stress testing was also performed by simulating multiple voters casting their ballots simultaneously, which confirmed that the blockchain could process concurrent transactions without data duplication or loss.

5. Observation and outcome: The results of the simulation validated the system's expected advantages. Once recorded, votes could not be altered or removed, ensuring immutability. The distributed ledger provided transparency, as results could be independently verified by authorized entities. Voter anonymity was maintained through cryptographic techniques while still enforcing eligibility. The comparison of platforms revealed that Ethereum offers strong decentralization but suffers from latency and higher transaction costs, whereas Hyperledger Fabric delivers better scalability and performance but relies on permissioned authorities. Overall, the simulation demonstrated that the proposed blockchain-based e-voting system can effectively secure voting processes, prevent fraud, and provide trustworthy results, particularly for small- and medium-scale elections.

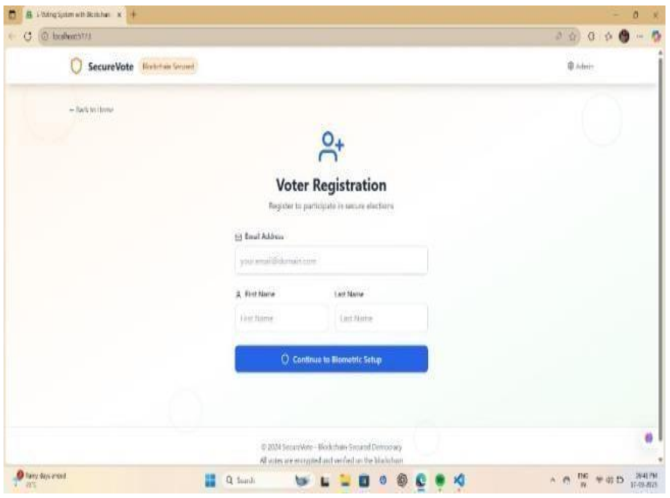
RESULTS AND DISCUSSION

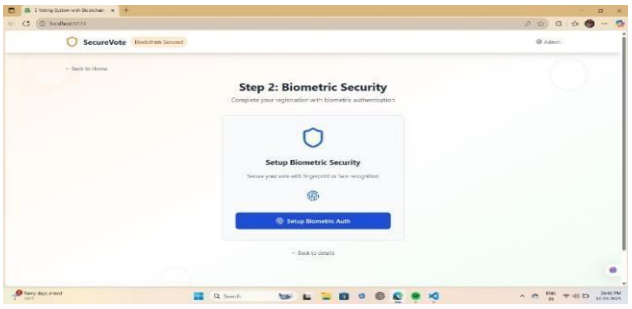
A. Security

The proposed system significantly enhances election security by leveraging blockchain's immutability and smart contract automation. Each vote is stored as a unique, tamper-proof transaction on the Ethereum blockchain, ensuring that it cannot be altered or deleted after submission. Multifactor authentication (MFA), combined with MetaMask wallet verification, prevents unauthorized access and identity theft. Homomorphic encryption further ensures vote confidentiality even during the counting process. Unlike traditional centralized systems, this design eliminates single points of failure, reducing risks of insider threats and cyberattacks.

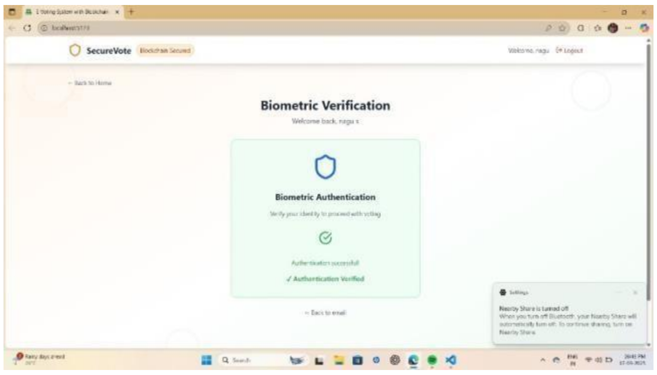


This is the main landing page of the "SecureVote" system. It explicitly highlights its three core features: "Blockchain Security," "Biometric Auth," and "Real-time Results." This serves as the system's mission statement, emphasizing its commitment to security through cryptographic protection and an immutable blockchain.

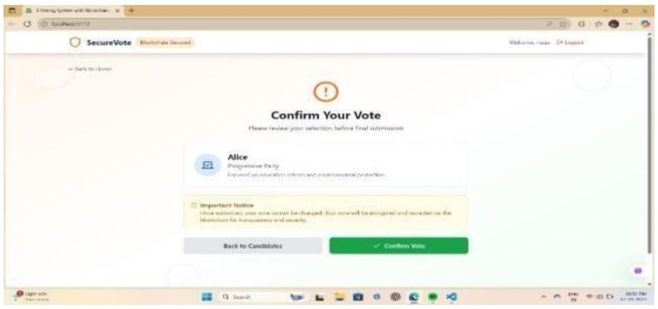

SecureVote
Blockchain Secured
Admin
← Back to Home
Voter Registration
Register to participate in secure elections
Email Address
your.email@domain.com
First Name
Last Name
First Name
Last Name
Continue to Biometric Setup
© 2024 SecureVote - Blockchain Secured Democracy
All votes are encrypted and verified on the blockchain


SecureVote
Blockchain Secured
Admin
← Back to Home
Step 2: Biometric Security
Complete your registration with biometric authentication
Setup Biometric Security
Secure your vote with fingerprint or face recognition
Setup Biometric Auth
← Back to details

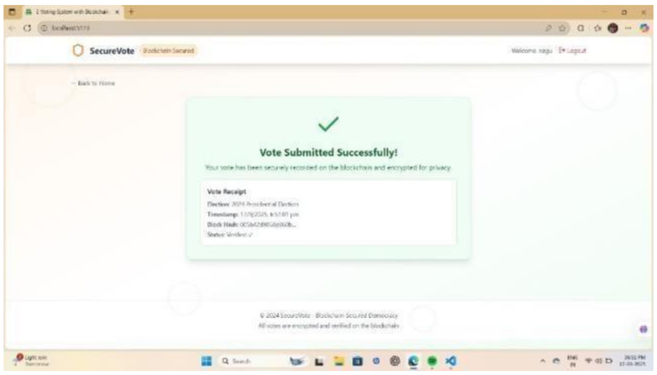
show the voter registration process. They detail a two-step process: first, the voter enters their email and name, and then they are prompted to set up "Biometric Security." This demonstrates a multi-factor authentication approach, combining a digital identity with a physical biometric one (fingerprint or face recognition), which is a key security measure to prevent unauthorized voting.



It highlights the use of biometric authentication to verify the identity of voters before they can proceed with casting their votes, ensuring security and preventing unauthorized access. The system confirms successful authentication with a green checkmark and a message stating "Authentication Verified," demonstrating how blockchain technology can be integrated with biometric security to provide transparency, trust, and tamper-proof digital voting.



This image depicts a secure voting interface for the "SecureVote" system, which utilizes blockchain technology for enhanced security and transparency. The user is prompted to confirm their vote for "Alice" of the Progressive Party, with a clear notice stating that once submitted, the vote cannot be changed and will be encrypted and recorded on the blockchain.



Your vote has been securely recorded on the blockchain and encrypted for privacy.

Vote Receipt
Election: 2024 Presidential Election
Timestamp: 17/9/2025, 6:51:01 pm
Block Hash: 0c5b42d90d0e0b...
Status: Verified ✓

© 2024 SecureVote · Blockchain Secured Democracy
All votes are encrypted and verified on the blockchain

The image depicts a confirmation page of an e-voting system named SecureVote, which utilizes blockchain technology for secure and transparent voting. The system has successfully recorded a vote for the 2024 Presidential Election on the blockchain, ensuring the vote's privacy through encryption. A vote receipt is provided with details including the election name, timestamp, block hash, and verification status. The interface indicates the vote has been "Verified," emphasizing the security and integrity of the blockchain-secured voting process.

B. Scalability

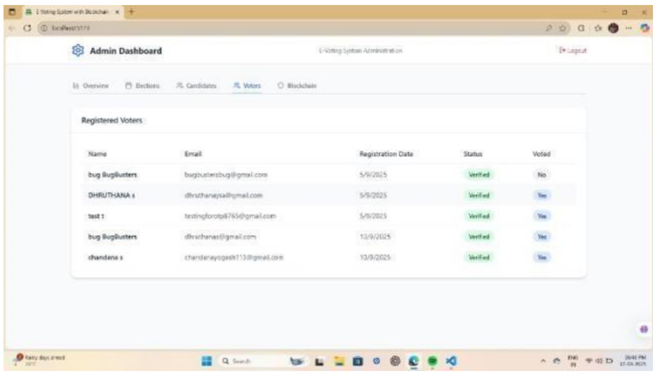
Scalability remains a significant challenge for blockchain applications, and this system is no exception.

On-Chain Limitations: The current implementation operates on the Ethereum testnet. Ethereum's public mainnet, while secure, has inherent limitations in transaction throughput and cost (gas fees). Conducting a large-scale national election with millions of voters would be prohibitively expensive and slow with the current architecture, as each vote requires a gas fee and must be processed by the entire network.

Off-Chain Scalability: the use of Supabase for non-critical data is a strategic decision that enhances scalability. Superbase, built on PostgreSQL, can efficiently handle high volumes of read/write operations for user interfaces and metadata, scaling horizontally as needed. This hybrid approach (on-chain for votes, off-chain for UI/UX) effectively offloads scalable operations from the constrained blockchain layer.

Potential Solutions: Future iterations could explore Layer-2 scaling solutions (e.g., Polygon, Arbitrum) or alternative consensus mechanisms (e.g., Proof-of-Stake sidechains) designed for high-throughput applications.

This would maintain security guarantees while drastically reducing costs and increasing transaction speed



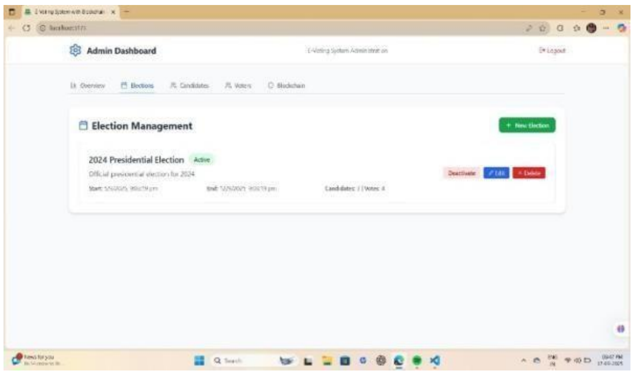
Admin Dashboard

E-Voting System Administration

Registered Voters

Name	Email	Registration Date	Status	Voted
bug BugBusters	bugbustersbug@gmail.com	5/9/2025	Verified	No
DHRUTHANA s	dhruthanaysai@gmail.com	5/9/2025	Verified	Yes
test t	testingforotp6765@gmail.com	5/9/2025	Verified	Yes
bug BugBusters	dhruthanas@gmail.com	10/9/2025	Verified	Yes
chandana s	chandanayogesh113@gmail.com	10/9/2025	Verified	Yes

This image depicts an Admin Dashboard for an E-Voting System utilizing Blockchain technology. The dashboard displays a list of "Registered Voters" with details including Name, Email, Registration Date, Status, and Voted status. All listed voters are "Verified," indicating successful registration validation. The integration of Blockchain in this E- Voting System likely ensures transparency, security, and immutability of voting data, as Blockchain technology is known for its ability to provide a tamper-proof and auditable record of transactions or, in this case, votes.



Admin Dashboard

Election Management

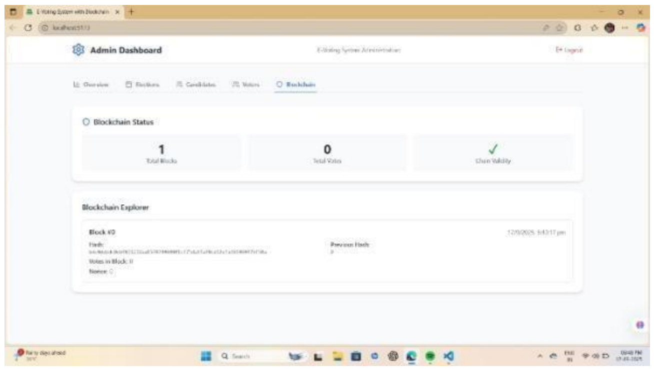
2024 Presidential Election — Active

Official presidential election for 2024

This displays the Admin Dashboard of a Blockchain-based E-Voting System, showcasing the Election Management interface. Here, the administrator can manage elections, with the example showing the "2024 Presidential Election" marked as active. It provides essential details such as the start and end times, number of candidates, and votes cast. The dashboard also offers options to create a new election, deactivate, edit, or delete existing ones. This demonstrates how blockchain technology ensures transparency, security, and efficient management of digital elections while giving administrators full control over the election process.

C. Efficiency

Efficiency is achieved through automation, optimized workflows, and user-friendly design. Voter authentication using MetaMask is quick and eliminates dependence on centralized login systems. Once verified, voters can cast their votes through an intuitive web interface, reducing time and complexity compared to manual methods. Smart contracts automate vote counting and result publication, significantly speeding up the process while minimizing errors. This automation reduces operational costs, minimizes manual delays, and ensures near-instant availability of election results.



Admin Dashboard

Blockchain Status

1 Total Blocks — 0 Total Votes — ✓ Chain Validity

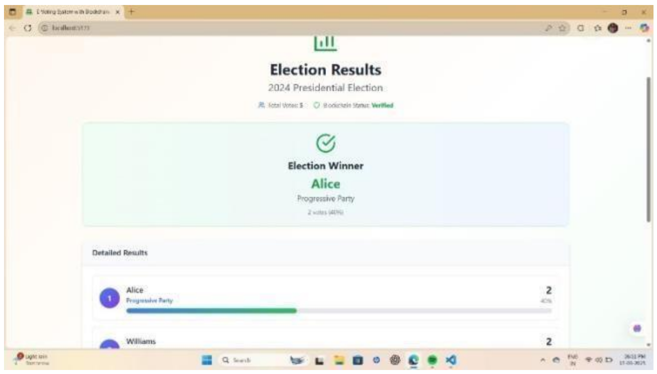
Blockchain Explorer

Block #0

Votes in Block: 0

Nonce: 0

This "Blockchain" tab on the "Admin Dashboard" is a "Blockchain Explorer." It provides a snapshot of the blockchain's current state, showing the "Total Blocks," "Total Votes," and "Chain Validity." This real-time visibility is crucial for efficiency, allowing administrators to quickly verify the system's integrity and monitor the voting process.



Election Results

2024 Presidential Election

Total Votes: 5 — Blockchain Status: Verified

Election Winner

Alice

Progressive Party

Detailed Results

Alice, Progressive Party — 2

Williams — 2

This shows the "Election Results." The system has efficiently calculated and displayed the results in a clear and easy-to-read format, including a breakdown of votes and percentages for each candidate. The fact that the "Blockchain Status" is "Verified" and the "Total Votes" are displayed in real-time demonstrates the system's efficiency in processing and tallying votes almost instantaneously after they are cast.

DISCUSSION

The system successfully demonstrates that blockchain-based voting can improve election security, scalability, and efficiency. While Ethereum’s transaction speed and cost remain limitations for nationwide deployment, the integration of off-chain storage, decentralized identity verification, and smart contract automation provides a robust framework for secure institutional and organizational elections.

CONCLUSION AND FUTURE SCOPE

The proposed ‘Remote Secure Voting System Using Ethereum and Smart Contract’ successfully demonstrates a robust and modern framework for secure digital voting. It effectively leverages decentralized blockchain technology to address critical flaws in traditional electoral methods such as lack of transparency, vulnerability to tampering, and limited accessibility. By strategically integrating Ethereum smart contracts, MetaMask for decentralized authentication, superbase for scalable off-chain data management, and a React-based frontend, the system ensures end-to-end verifiability, immutability, and ease of use. This makes it a viable solution for institutional, organizational, and potentially governmental elections.

Looking ahead, the system’s future scope is broad and offers significant opportunities for enhancement, These include improving scalability through Layer 2 solutions or sidechains, strengthening voter anonymity and privacy via advanced cryptographic techniques such as zero-knowledge proofs (zk-SNARKs/zk-STARKs), and enhancing mobile accessibility through dedicated applications. Future developments may explore cross-chain interoperability, AI-driven fraud detection, regulatory compliance integrations, decentralized storage solutions such as IPFS, and advanced voting models like quadratic or ranked-choice voting. By pursuing these innovations, the system can evolve from a foundational academic prototype into a comprehensive, universally adaptable platform capable of restoring trust and integrity in electoral processes worldwide.

REFERENCES

1. Singh, A., Ganesh, A., Patil, R. R., Kumar, S., Rani, R., & Pippal, S. K. (2023). Secure voting website using Ethereum and smart contracts.

2. afar, U., Ab Aziz, M. J., Shukur, Z., & Hussain, H. A. (2022). A systematic literature review and meta-analysis on scalable blockchain-based electronic voting systems.

3. Yadav, A. S., Thombare, A. U., Urade, Y. V., & Patil, A (2020). E-Voting using Blockchain Technology .International Journal of Engineering Research & Technology

4. Beulah Jayakumari , S Lilly Sheeba , Maya Eapen , Jani Anbarasi, Vinayakumar Ravi , A. Suganya,Malathy Jawahar .(2024). E-voting system using cloud-based hybrid blockchain technology

5. Zunjarrao Aastha Girish, Kirtik Ganeshan, Srushti More, Shrutam Tambe, Prof. Vasavi Alamuri .(2024). E-Voting System Using Blockchain and Web Engineering.

6. Henry O. Ohize, Adeiza James Onumanyi, Buhari U. Umar, Lukman A. Ajao, Rabiu O. Isah, Eustace M. Dogo, Bello K. Nuhu, Olayemi M. Olaniyi, James G. Ambafi, Vincent B. Sheidu, Muhammad M. Ibrahim. (2024). Blockchain for Securing Electronic Voting Systems: A Survey of Architectures, Trends, Solutions, and Challenges.

7. Said El Kafhali .(2024). Blockchain-Based Electronic Voting System: Significance and Requirements

8. Tolegen Aidynov, Nikolaj Gramin, Dina Satybaldina, and Assel Nurusheva .(2024). : A Systematic Literature Review of Current Trends in Electronic Voting System Protection Using Mode.