iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.80294

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Block-Chain Based Secure Voting System

Yash Thakkar[1], Saad Ansari[2], Junaid Chatabali[3], Anamika Shukla[4]
Ajeenkya DY Patil University, Pune-411062

Abstract: ***Trust in elections often wobbles when systems rely too heavily on central control. Old-style setups - like electronic machines or mainframe vote counting - can break at one weak spot, inviting hackers or insider meddling. When flaws appear, belief in results slips. A different path opens up by shifting power across a wider network, where checks happen openly, without needing to trust any single player. Step by step, code replaces closed doors: here, a program called VotingDapp.sol runs the show. Built with Hardhat, it lives on Ethereum's Sepolia playground, testing how votes might be cast beyond reach of tampering. Instead of locked rooms, rules are baked into software that cannot bend after launch. From start to finish, each choice stays sealed, visible only as proof, never altered. The goal isn't speed or novelty - it's staying unchanged. Step by step, the system checks who can vote using tight digital rules locked into code. Once someone submits a vote, their status freezes instantly - no second tries allowed. This stops fake identities from voting more than once. Candidate data lives outside the chain, stored on IPFS through Pinata so changes need no approval from one central group. How users see it? A clean front end built with Next.js connects quietly to tools like Wagmi, Viem , and RainbowKit. These handle login through crypto wallets without fuss. Behind every click, security runs deep but stays unseen. Out in the open, network checks show the system blocks intrusions without slowing down. Efficiency sticks through steady transaction fees, while results add up fast - backed by clear math proofs. A smooth front end hides complex coding tricks underneath. Security here stands far above older voting methods, not just matching them but moving past. Tampering fails every time.***

I. INTRODUCTION

Truth shapes how fair elections feel. When people believe votes might vanish or change, belief in government slips. Systems meant to count ballots safely often draw doubt because someone somewhere could twist results. Machines collecting choices electronically sometimes fail when tested by hackers or even workers inside the system. In nations with massive voter numbers such as India, stories about stolen identities or switched tallies pop up again and again. Trust drains slowly each time names disappear or totals shift oddly after polls close. Problems grow around one weak spot - too much power held in one digital vault or storage room full of paper. Hackers aim there. Insiders misuse access too. Protection crumbles if everything depends on just one hub where data piles up unprotected.

Lately, blockchain tech stepped into the spotlight by making decentralized setups feel more doable. When info gets locked in through verification and mining, changing it later becomes impossible - thanks to its shared, unchangeable record. A fresh approach here suggests building a voting platform on blockchain, aiming to wipe out ballot meddling for good. Swap out central servers. Bring in Ethereum smart contracts instead. Every step of an election then opens up - registration, choices made, count finalized - all visible, checkable, sealed tight with cryptography.

Though older methods often fail at stopping fake votes or take too long to count, new Web3 tools update records instantly, shutting down repeats right away. A setup built in Solidity, tested through Hardhat, forms the backbone here. Voting access runs through layered checks, each person confirmed directly on the ledger. Instead of confusing tech displays, users interact via secure digital wallets - powered by Wagmi and Ethers.js - that hide the heavy lifting behind smooth actions. Stored data spreads across distributed networks, keeping everything both open and locked tight.

This whole flow runs without central control, yet feels like any regular app.

All voting transactions and validations are processed over the Sepolia test network, ensuring that each cast ballot is a mathematically verifiable transaction. By recording real-time accumulated tallies directly on the blockchain, this system successfully mitigates the security flaws of traditional voting while maintaining absolute transparency for all stakeholders.

A. Motivation And Objective

What drives this project is how often election systems fail in places where fraud can spread easily. In large countries such as India, stories about rigged votes and behind-the-scenes interference keep appearing - so relying on human oversight alone makes little sense anymore. Instead of hoping officials do the right thing, a better path uses code that acts like unbreakable rules.

©IJRASET: All Rights are Reserved

This effort builds a tool based on Ethereum's network, one that runs without central control, meant only for running honest elections. Hidden inside are programs that lock down when people can vote, check identities through several stages, then add up results live - visible to everyone yet impossible to change once recorded.

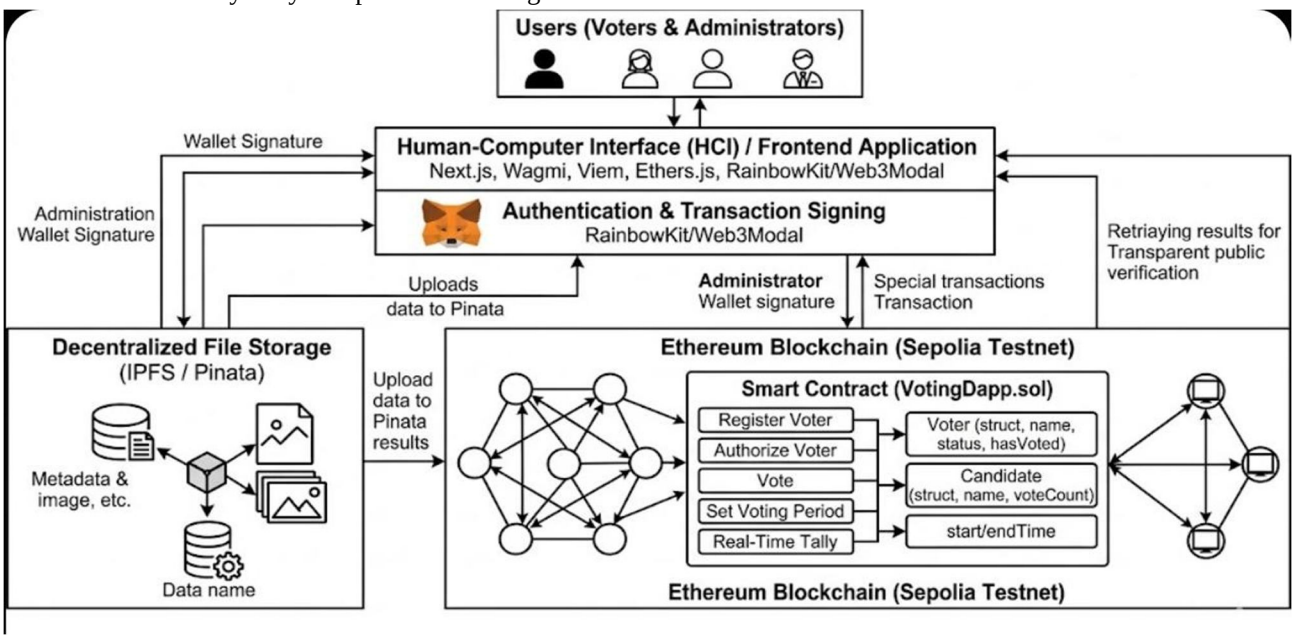


Fig. 1.1 System Architecture of the Proposed Blockchain-based Secure Voting System

II. LITERATURE REVIEW MOTIVATION AND OBJECTIVE

Getting votes counted fairly has always meant juggling ease of access against tight safeguards. Paper ballots can be checked by hand afterward yet often stumble on slow logistics, possible interference, or mistakes when adding up results [1]. To fix those weak spots, machines that record votes electronically came into play - like evms and dre setups. Still, deep reviews show many such devices carry serious risks since they run on closed software and central control points, making stealth changes harder to catch and full verification out of reach [2].

What began as a way to record online payments across separate computers soon offered something new: trust through code. Instead of one company checking each change, everyone in the system agrees together before anything updates [3]. When Ethereum arrived, it added programs - small pieces of software - that run automatically once conditions are met [4]. These programs follow strict rules built right into their structure. Soon enough, scientists realized such systems could handle elections too, step by step, locking choices securely. Rules execute exactly as written, leaving no room for late adjustments or manual steps
[5].

Some recent research looks into different setups for online voting without central control. One big issue keeps coming up - people pretending to be many voters at once. Storing details about candidates and digital items becomes tricky when it relies on regular servers [6]. Instead, using a system called IPFS might help keep things spread out and safe from tampering. That way, what users see can’t easily be faked. Other papers suggest linking this file storage with blockchain tools to strengthen trust. Here, the approach ties together current web tools like Wagmi and Ethers.js [7]. These connect to programs running on Ethereum. Files go off-chain through Pinata, part of IPFS. Together, they form one consistent setup where security comes from design, not just promises.

A. Motivation And Objective

What drives this study is the pressing demand to protect fair elections in areas often hit by organized cheating. In heavily populated countries such as India, stories about rigged ballots and behind-the-scenes interference keep surfacing, pushing the search for a method where confidence comes from numbers, not just officials' promises. When power sits in one place, risks multiply - attackers who breach central servers might change results without anyone noticing. Trust built on code beats trust based on hierarchy when stakes run high.

This research aims to build a voting system on the Ethereum blockchain, using decentralization for safety and integrity. Its main purpose takes shape through several focused goals laid out below

©IJRASET: All Rights are Reserved

1) A fresh start each time shapes how the code grows - this one builds a fixed smart contract called VotingDapp.sol through Hardhat. Step by step it runs, locking down voting periods by design instead of chance. Access rolls out across layers, permission shifting only when rules align. Behind every check sits logic carved once, never rewritten.
2) One way to stop double voting is using a clear yes-or-no marker updated all at once across the network. This flag shows if someone already cast their vote. Changes happen fully or not at all. The system records each decision exactly once. Everyone sees the same status at the same time. No gaps allow repeated entries. Updates lock in place instantly. A single source tracks every voter's action. Consistency blocks fake repeats. Each step depends on the last being finished. Nothing slips through cracks.
3) A decentralized setup kicks off by storing candidate details through Pinata on IPFS. That way, control shifts away from one central authority. Ballot data stays locked down - no room for tampering once it's set. Changes become impossible since access spreads across nodes instead of sitting in one spot. Information lives pinned, visible to everyone yet editable by none.
4) A person might explore a system built with Next.js, where React shapes the layout while RainbowKit handles connections. Instead of wrestling with complex crypto rules, someone clicks through steps on the Sepolia testnet. The interface opens up access, simplifying each move. Tools fit together quietly behind scenes. Interaction becomes straightforward, even for those unfamiliar with blockchain details. Each piece works because attention went into clarity. Navigation feels natural, almost like browsing any common website. Behind it all, structure stays strong but invisible.

Proposed Web3 / Blockchain System	Traditional EVM / Centralized System
1. User / Voter $\rightarrow$ connects securely via a cryptographic Crypto Wallet (like RainbowKit or MetaMask) which also provides secure identity verification.	**1. User / Voter** $\rightarrow$ casts vote via a physical machine or web interface.
2. Smart Contract Logic (VotingDapp.sol) $\rightarrow$ automatically enforcing all election rules, authorization, and state changes (decentralized, verifiable logic).	**2. Central Web Server** $\rightarrow$ receiving and processing votes (a single point of failure).
3. Distributed Blockchain Network (Sepolia / Immutable Ledger) $\rightarrow$ once a vote is cast, it is immediately mined into a block and permanently recorded by a consensus of nodes across the network, making it impossible to alter retroactively.	**3. Centralized Database** $\rightarrow$ storing votes (mutable data, vulnerable to unauthorized access).
Key Points (Highlight Advantages): Cryptographically secure transactions, automated real-time vote tallying through smart contract execution, high availability and resiliency due to distributed nodes (no single point of failure), and absolute transparency with mathematically verifiable integrity for all results.	**Key Points (Highlight Vulnerabilities):** Susceptible to tampering (e.g., direct database manipulation by authorized or unauthorized users), Distributed Denial of Service (DDoS) attacks against the central server, potential for significant server downtime preventing voting, and delayed or obscure manual auditing processes.

Fig. 2.1: Architectural comparison between Traditional Centralized Voting Systems and the Proposed Decentralized Blockchain System.

III. PROPOSED SYSTEM

A. *Requirements*

A setup built around new Web3 tools shaped how the decentralized voting system came together. This work leaned on a Dell Vostro 3500 running daily tasks without hiccups. Inside sat an Intel 11th Gen Core i5 chip, paired with 8 GB memory and guided by an Nvidia MX330 GPU. Coding happened locally, contracts compiled here, visuals formed on this machine.

Splitting things up, the setup had two main parts: code that runs on a blockchain and what users actually see. Instead of going with older options such as Ganache, developers picked Hardhat (v2.26.3) for handling the chain-based work. Written in Solidity (^0.8.19), the self-executing agreements leaned heavily on OpenZeppelin Contracts (@openzeppelin/contracts) to stay safe. Talking to the network? That job went to Ethers.js (v5.7.2), backed by tests built with Chai plus Waffle's special checks. Live trials happened not locally but on Sepolia, using Alchemy's RPC link to connect. When it came time to store details off-chain, Pinata stepped in - handling uploads to IPFS so data stayed distributed.

©IJRASET: All Rights are Reserved

Starting off, the front end ran on Next.js version 13.2.4, shaped visually through Tailwind CSS at v3.3.1. Instead of coding web3.js by hand, blockchain talks happened via Wagmi 2.12.17 paired with Viem 2.21.19. Security during login came from RainbowKit working alongside WalletConnectCloud. Voter ID checks stayed protected thanks to that setup.

B. Methodology

1) Smart Contract Architecture:Inside a single smart contract called VotingDapp.sol, the main rules of the election system were built. Control flows like stages in a machine, guided by one admin - the person who owns the contract. Setting up begins with configuring key details: when things start, when they end, plus listing those running. Time boundaries are locked using two values - startTime and endTime - that mark the exact period people can vote. Any move made before or after gets blocked automatically thanks to a security check named onlyDuringVoting. This rule shuts down invalid attempts through code enforcement, not human oversight.

2) Voter Authorization Protocol: For solid election security, a tight three-step check process was built. Starting off, each voter gets entered through registerVoter, which drops their public address into the database as Pending. Only after that can an admin step in - using approveVoter - to shift their status to Approved. When casting votes, the vote function runs live checks: it confirms registration, approval, and whether timing falls inside the open period. Built-in safeguards only let valid participants proceed when conditions align just right. Every time someone votes, the system marks their spot with a permanent yes-or-no tag. If that tag already says yes, the rules stop anything else from happening. This check runs every single time, no exceptions. Once the vote goes through, the tag flips right away, locked into place. After that change, trying again does nothing at all. The whole process ties each move to one digital identity only. Nothing slips past once the update takes effect. A second attempt finds the door shut before it even knocks.

3) How Transactions Change System States: One step at a time, voting moves through a secure digital path. From the start, the person picks a choice on a web screen built with Next.js. After that comes forming the data package - Wagmi and Ethers.js handle it together. Signing happens next, only when the user approves with their secret key inside a tool like MetaMask. With approval locked in, the network pushes the result out through Alchemy's gate to Sepolia. Once inside the VotingDapp.sol contract, checks confirm if access is allowed. When approved, the current status shifts without delay. The chosen candidate's count moves up by one - c.voteCount increases directly - and right after, a Voted signal writes into chain records. Since each addition locks instantly across nodes, the total stands fixed once voting ends, based only on what voteCount holds at that moment, making outcomes clear and instant.

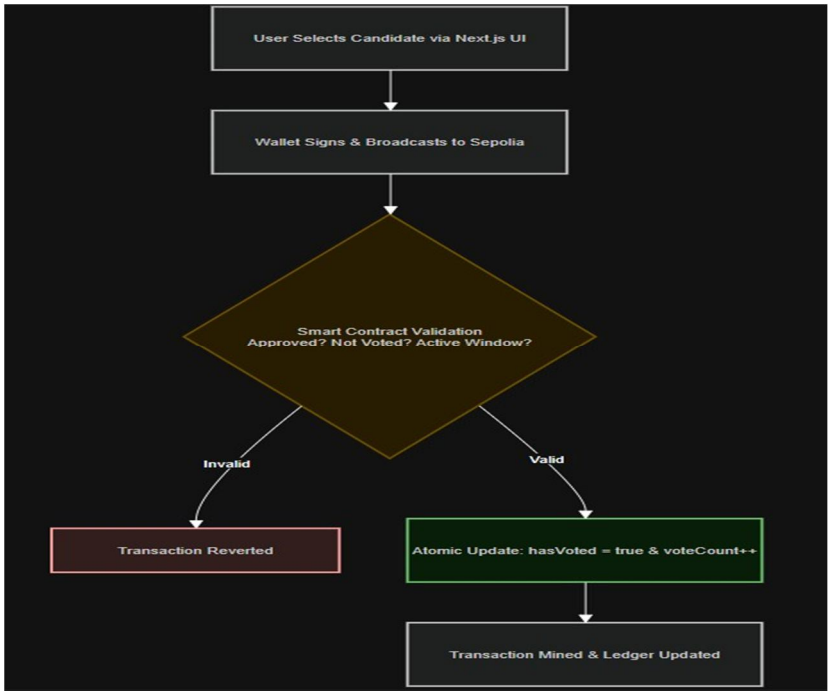


Fig. 3.1: Transaction workflow for the proposed voting system (a) User interface interaction and transaction broadcasting (b) Smart contract logic, state updating, and real-time tallying.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

IV. RESULTS AND DISCUSSION

A. Smart Contract Deployment and Gas Optimization

Out of the gate, the VotingDapp.sol code transformed into machine-readable format through Hardhat without errors. With optimization cranked to 200 runs, the compiler trimmed down the output just enough to save on gas. A live connection to Alchemy's API carried the deployment package onto the Sepolia testing ground. Once confirmed, the system locked in the creator's wallet as sole controller of contract settings.

Later checks on the contract tracked how much gas each transaction used. Since spending less gas means lower fees, this matters most when running code live on Ethereum. Instead of guessing, real tests ran multiple times on actions that alter data - like signing up voters, allowing them, or casting ballots - to pin down typical computing cost.

B. Functional Security and Sybil Resistance Testing

Midway through the test, every try by unknown addresses to cast votes got blocked - solid proof the safeguards worked. Something interesting happened when approved wallets tried voting twice: the system shut it down instantly. Each rejection came straight from a core rule check failing inside the code. That failure made the EVM stop everything before changes could stick. Layered permissions held firm throughout, just like they were built to do.

Midway through testing, several transactions launched at once from a wallet that already voted. Each time, the system saw the vote record and stopped the process cold - message popped up: "Already voted." From another angle, when someone tried voting too early or too late, based on the coded timeline, math itself shut it down. Timing rules held firm, no exceptions made.

C. User Interface and Real-Time State Synchronization

A web app built with Next.js talked to a live smart contract, thanks to tools called Wagmi and Viem. Thanks to RainbowKit joining the mix, people could link their Web3 wallets without hassle. On-screen details about candidates came straight from blockchain storage, pulled using React Query. Files tied to each candidate lived on Pinata's IPFS system, loaded cleanly into view.

After the vote went through, a short pause followed - about as long as it takes Sepolia to seal a block, so roughly 12 to 15 seconds. Only when the block arrived did the system log the Voted event from the smart contract. That signal got picked up by Wagmi tools right away, which then nudged the interface to redraw itself instantly. Instead of waiting or reloading, the updated tally appeared on screen straightaway, showing exactly how many votes the choice now held. Everything stayed visible, nothing hidden, all flowing without needing a push.

Smart Contract Function	Computational Action	Average Gas Used	Avg. Confirmation Time
Contract Deployment	Instantiation of `VotingDapp.sol`	~1,450,000 Gas	~14 Seconds
registerVoter()	Adding address to struct (Pending)	~65,000 Gas	~14 Seconds
approveVoter()	State change (Pending $\rightarrow$ Approved)	~45,000 Gas	~14 Seconds
vote()	State change (hasVoted = true, voteCount++)	~115,000 Gas	~14 Seconds

Table 4.1: Transaction Performance and Gas Utilization on Sepolia Testnet

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

V. CONCLUSION AND FUTURE WORK

A. Conclusion

A blockchain-powered voting platform shows how distributed ledgers can fix deep flaws in standard election systems. Instead of relying on central servers that can be altered, votes now live inside an unchangeable Ethereum program. This shift removes chances for ballot changes, insider manipulation, or collapse from one broken component. The code behind the vote tracker - VotingDapp.sol - uses layered checks to confirm voters, while locking down duplicate submissions using instant data updates. On screen, tools like Next.js paired with Wagmi and RainbowKit reveal complex encryption in ways people actually understand. After a choice gets digitally sealed and added to the Sepolia chain, it stays visible forever, open for checking by anyone at any time.

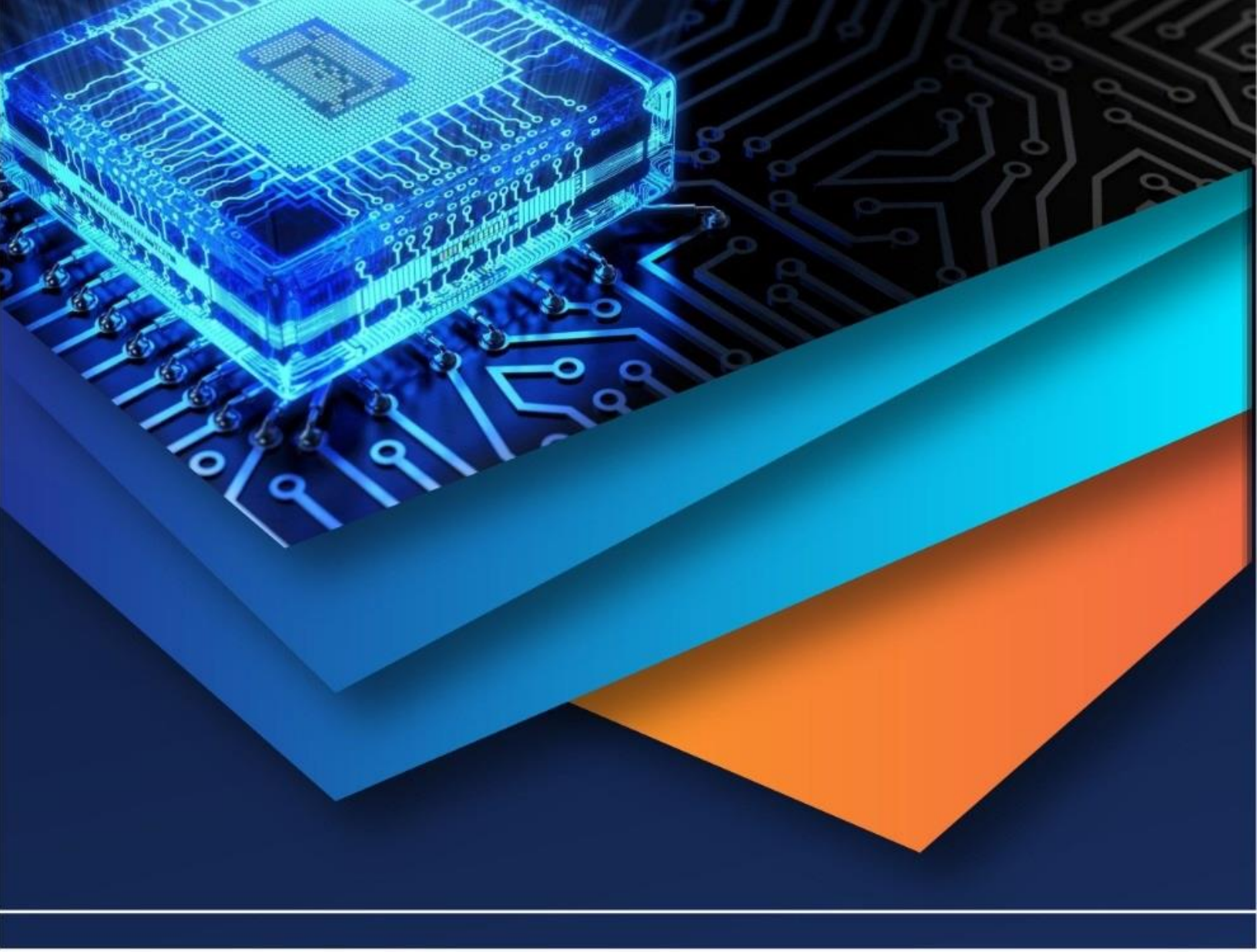
B. Future Work

Right now, the setup keeps votes secure across every step. Still, there are clear ways it could improve down the line. One idea: using zk-SNARKs so people stay fully anonymous even though their eligibility gets checked. That way, the system knows someone can vote and hasn't already - without seeing who they are or what they picked. Another path opens up when shifting away from Ethereum's busy main network. Moving to something like Arbitrum or Polygon cuts fee costs sharply, yet still leans on strong underlying protection. On top of that, logging in through the app might get safer with decentralized ID tools or built-in device scans, linking real-world identity more tightly to digital keys.

REFERENCES

[1] A. Halderman, "Practical attacks on real-world e-voting," in Proceedings of the IEEE Conference on Privacy, Security and Trust, 2018, pp. 145-152.
[2] S. Springall et al., "Security analysis of the Estonian internet voting system," in Proceedings of the 21st ACM Conference on Computer and Communications Security, 2014, pp. 703-715.
[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Decentralized Business Review, p. 21260, 2008.
[4] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum project yellow paper, vol. 151, no. 2014, pp. 1-32, 2014.
[5] F. Hao, P. Y. Ryan, and P. Zieliński, "Anonymous voting by two-round public discussion," IET Information Security, vol. 4, no. 2, pp. 62-67, 2010.
[6] J. Douceur, "The Sybil attack," in International workshop on peer-to-peer systems, Springer, Berlin, Heidelberg, 2002, pp. 251-260.
[7] J. Benet, "IPFS-content addressed, versioned, P2P file system," arXiv preprint arXiv:1407.3561, 2014.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)