iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** I **Month of publication:** January 2026

DOI: https://doi.org/10.22214/ijraset.2026.76825

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 I Jan 2026- Available at www.ijraset.com

A Privacy-Preserving Blockchain Voting System with Cryptographic and Steganographic Security

Prof. Ramesh Kumar K R[1], Zoya Fathima[2], Varsha Meti[3], M Venu[4]
Ballari Institute of Technology and Management, Ballari

Abstract: ***Democratic societies rely heavily on elections that citizens can trust, yet traditional voting methods—whether through paper ballots or Electronic voting machines (EVMs)—continue to struggle with issues like vote tampering, duplicate identities(unique voting ID), low transparency, and delayed results.***
Blockchain is used at the centre of this system because it functions without a single authority and keeps stored records fixed once they are written. This design removes a single point of attack and makes any hidden alteration of voting information highly unlikely.
To strengthen security, cryptography is applied to protect voter identities and to confirm that only authorised individuals can cast a ballot. Steganography adds another layer by placing sensitive voting details within ordinary digital formats, which makes unauthorised detection or interference far less likely.
In this work, we built and evaluated an e-voting model that runs on a blockchain network, maintains the principle of one vote per person, allows verification through a distributed set of nodes, and guards confidential data through multiple security measures. The outcome of this study indicates that bringing together a decentralised ledger, encryption, and concealed data techniques can support a more dependable and harder-to-tamper online voting platform, demonstrated here through a simple web-based prototype.
Keywords: E-voting, Decentralisation, Blockchain, Cryptography, Steganography.

I. INTRODUCTION

Blockchain has moved into mainstream discussion over the past few years because it offers a dependable way to store information across multiple network nodes without relying on a single authority. Its initial breakthrough came from the idea of a shared ledger that records transactions openly while preventing hidden alterations, which encouraged many researchers to look beyond its original use in digital currency [1].

Recent blockchain frameworks go beyond the idea of a basic ledger by adding computational layers that let organisations run full applications in a decentralised setup. These newer systems are built to allow different platforms to communicate and to support programmable features, which pushes blockchain past simple data recording and opens the door to more complex digital operations [3]. A key factor in this shift is the rise of smart contracts, which are self-executing pieces of code placed directly on the chain. After they are deployed, they function according to predefined rules without outside adjustment, offering consistent behaviour and long-term trust in their execution [4]. With these additions, blockchain is no longer just a method of storing records but an expanding framework that supports automated processes, secure coordination, and verifiable computation.

Blockchain has gained broad attention in recent years because it offers a secure and verifiable way to maintain records across a distributed network without relying on a single authority. Its early development showed how shared ledgers could store information with clear visibility and minimal risk of hidden alteration, which encouraged researchers to consider its use outside of cryptocurrency and financial systems [1].

What began as a trial model for handling decentralised data has now become a structural component in many digital platforms, supported by continued study and practical adaptation within the software field [2].

In terms of structure, blockchain organises information into a linked sequence of blocks, with each new block containing verified entries and a cryptographic link to the block that came before it. The initial block, identified as the genesis block, forms the base of the ledger and serves as the origin for all subsequent additions.

Within each block, transactions are compressed into a Merkle root, allowing the system to maintain a reliable and compact record of what has been stored.

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

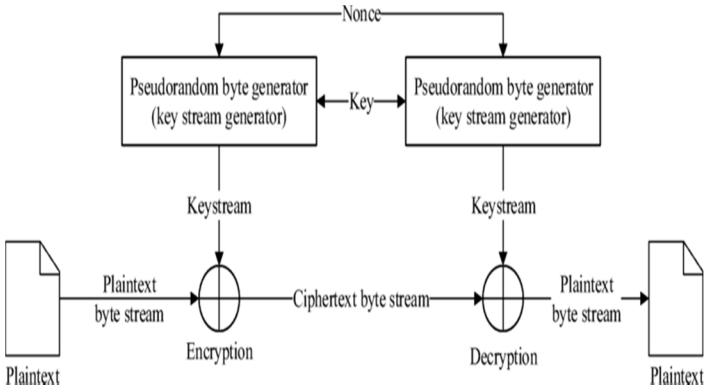


Figure 1- Hash (Sha-256) Algorithm

The system operates across a peer-to-peer network, where every participating machine or node communicates directly with others to exchange information about recent blocks and pending transactions. When a new node joins the network, it becomes aware of existing participants through peer communication, which allows it to fit naturally into the decentralised structure. Its main role is to check and confirm incoming transactions as well as newly generated blocks. During this process, the node downloads the full chain of records starting from the first block and verifies them one by one. After every block is checked and the cryptographic values align with the expected values, the node reaches full synchronisation and can function as part of the network. In this system, the Merkle root is kept in the block header and acts as a concise representation of all transactions contained within that block. Because the blocks are linked together in sequence, changing even a single transaction would require altering the headers of every subsequent block.

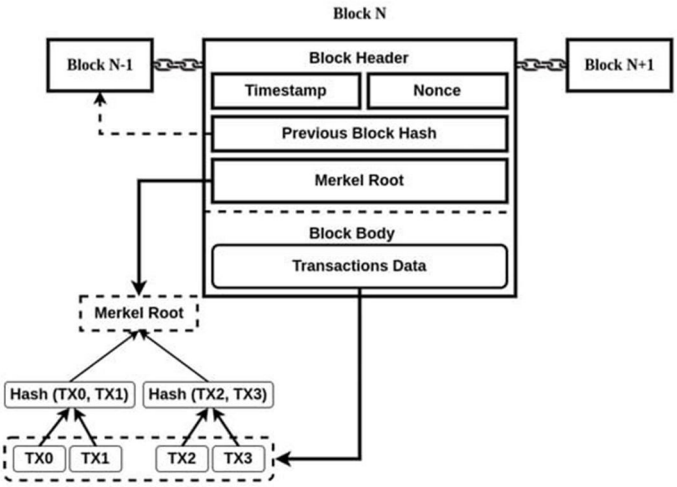


Figure 2: Blockchain Block Architecture

Blockchain for digital voting has grown mainly because many current electronic voting platforms still face practical shortcomings. Although many electronic voting models have shown acceptable results in controlled setups, their performance tends to weaken when they are introduced into larger, real-world election environments.

Issues such as confirming that each vote is correctly logged, preventing undetected changes to stored results, and sustaining public confidence in the counting process continue to present practical hurdles. These difficulties have led system architects to look beyond centralised designs and consider approaches in which responsibility is spread across multiple parties rather than concentrated in a single authority. Blockchain addresses this concern by distributing verification across independent network participants instead of relying on one of the several databases. This structure reduces the likelihood that a single malfunction or breach could disrupt the entire electoral process. Its permanent recording style also makes quiet alterations more noticeable. For these reasons, blockchain is being examined as a possible route to improve the reliability and transparency of voting systems, even though it does not yet answer every challenge associated with large-scale elections.

The reluctance toward fully digital voting often comes from the difficulty of combining technical complexity with a process that requires complete clarity and public confidence. The concern is less about replacing traditional ballots and more about proving that a digital platform can operate reliably in different environments while remaining usable for every voter. Many current online systems still face practical limits, including how well they scale, the infrastructure required to support technology, and whether they can maintain steady performance when voter traffic peaks. At the same time, election administrators now expect clearer audit paths and built-in verification options, which older technologies rarely provide. In response to these gaps, this work presents a voting framework that uses blockchain to reduce dependence on central oversight while offering straightforward verification. Smart contracts running on the chain network are used to automate essential steps in the process, reducing manual intervention and potential error points. The following section reviews earlier digital voting systems to outline the architectural and operational challenges that led to the design choices made in this model.

II. MOTIVATION AND RELATED WORKS

A central motivation for this work is to consider whether current distributed technologies can help streamline the development of dependable digital voting platforms. The intention is not to replace traditional election practices but to understand how online participation tools might widen involvement by making collective decisions easier for people to take part in. More capable digital voting systems could ease the organisational burden of large elections and provide an option for regions that face practical or financial barriers in running physical polls [6]. In this sense, the focus is on improving accessibility and reducing procedural effort rather than replicating or redefining existing democratic structures. Public institutions experimented with a voting model built on blockchain to see whether shared verification could ease administrative work. Rather than depending on a conventional validation process, the trial incorporated formal checks to assess how accurately each step of the voting works.

In the deployment stage, the voting system is moved from the testing setup into real use. The servers are configured, the blockchain network is set up, and all necessary nodes are connected so they can start verifying votes. During this phase, voter accounts are activated, identity login tools are enabled, and the smart-contract functions for recording and counting votes are made live. The main focus here is to make sure every voter can access the platform, cast a vote without confusion, and receive confirmation that the ballot was recorded. Network monitoring is also turned on so that any unusual activity, delays, or errors can be spotted quickly. Once everything runs smoothly and votes are being recorded correctly across the nodes, the system is considered fully deployed and ready for the actual election. After the login process, the system relied on basic blockchain ideas to keep voting clear and fair. Several nodes helped check each vote; the stored records could not be quietly changed, and new entries were added only when the network agreed. This reduced the need for one central authority to manage the election and made the process easier to verify [5], [7]. The voting steps were also checked using formal testing tools to make sure votes were stored correctly, counted without mistakes, and protected from common security threats [8]. Smart contracts were looked at as well, mainly to handle repeated tasks automatically, like confirming ballots and keeping audit records. This follows current work in blockchain systems that use simple automated scripts to support secure and reliable actions across the network [4].

Across several small trial runs, the system showed that using secure cryptographic tools together with decentralised identity checks could support a safe and easy voting experience, even when the number of voters increased. Studies on blockchain-enabled security protocols further reinforced the system's layered protection model, showing how distributed trust structures can reduce single points of failure and improve auditability across the election lifecycle [9]. When paired with basic blockchain rules that support openness and reduce extra administrative work, this system showed that large-scale digital voting could be a realistic option for wider use across the country [7]. Combined with governance-oriented blockchain practices designed to increase transparency and minimise administrative inefficiencies, this approach highlighted the potential for nationwide adoption of robust digital election systems [7].

Overall, the integration of formal verification, privacy-preserving protocols, secure identity mechanisms, and blockchain-inspired trust models provides a strong foundation for scalable, verifiable, and publicly trusted remote voting solutions suitable for future national deployments [1], [5]. These services are mostly built for quick use, letting anyone create a basic poll and share a link so others can vote without any setup. They are easy to use and don't usually require sign-up, which makes them popular, but they offer very little protection. In many cases, there is no real authentication of voters; duplicate votes can be stopped only at the level of a browser, and there is no reliable way to check whether votes were changed.

When placed next to current secure e-voting research—where identity checks, encryption, and verification steps are standard—these casual polling tools show how limited unprotected and trust-based voting systems truly are [2]. In this study, we present a voting model that uses a distributed ledger to manage ballots without relying on a single central authority.

The system is designed to handle growth and support the basic needs of modern online voting while keeping the overall structure reliable. By spreading verification across multiple participants instead of one control point, the model helps reduce errors and strengthens trust in the final result.

III. IMPLEMENTATION AND DISCUSSION

In this section, the system's workflow is presented in a concise sequence:

1) Enrollment Stage: Each participant is assigned a temporary system-generated token that confirms eligibility without requiring extensive personal information.
2) Access Validation Stage: When the user returns to vote, the platform verifies the token and ensures that only one voting action can be linked to it, avoiding the need for additional authentication layers.
3) Secure Processing Stage: The vote is handled within the application, while only a cryptographic hash of the finalised ballot is written to the distributed ledger to provide tamper evidence without exposing ballot content.
4) Data Management Stage: A lightweight SQL database retains only essential metadata, minimising privacy risks and making data retrieval more efficient.
5) Distributed Verification Stage: A network of independent nodes validates the stored hash values, preventing any single point from altering verification data.
6) Result Compilation Stage: After voting ends, the platform matches internal vote entries with their corresponding ledger hashes, allowing each participant to confirm that their submission was included by checking its assigned verification value.

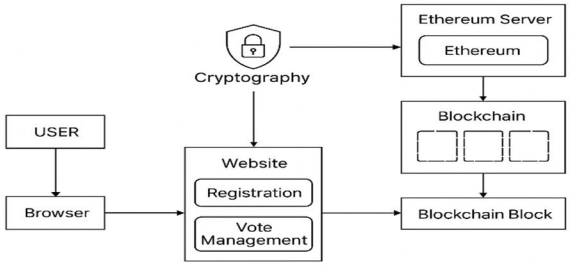


Figure 3 – System Overview

The system is organised using a layered software structure that separates user interaction, process handling, and data operations into different units. To keep the system easy to manage, it is divided into three main parts, and each one has a clear job instead of everything working all at once.

7) Controller: This is the part the user sees on the screen. It includes pages, buttons, text fields, and menus. It does not do any calculations or store information. Its only role is to show the interface properly and collect what the user enters. Once the user clicks or selects something, it sends that action to the next layer.
8) View: This section handles what happens after a user interacts with the system. It checks login details, runs verification steps, manages voting actions, and communicates with the blockchain when required. It doesn't keep data itself. Instead, it makes sure all steps follow the rules and manages how different parts respond.
9) Model: This is where the stored information is kept. It includes login details, verification records, and voting data. When the logic layer needs to check a voter or save a new record, it sends a request here. This layer then updates or returns the correct information.

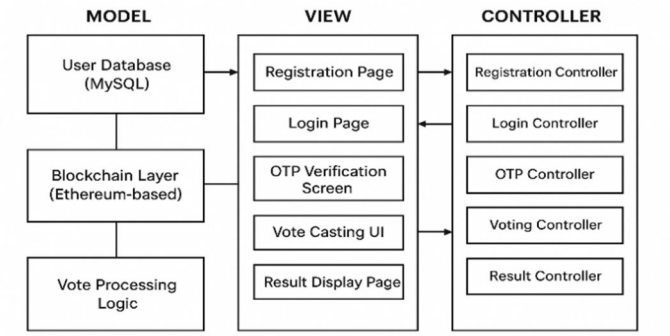


Figure 4: MVC Architecture of layers in Blockchain

In this system, users participate in the voting process without the need to create a blockchain. Instead of generating a cryptographic profile for every voter, the platform focuses on verifying user eligibility through conventional authentication steps. Once logged in, the system ensures that each ballot is recorded accurately and associated with the correct session, preventing duplicate submissions. The security of voting actions is maintained through controlled access rather than distributed ledger processes. As a result, voters can browse candidates and cast their choices without any form of computational cost.

In developing this system, different security mechanisms are used for distinct purposes. In this architecture, cryptographic functions guide the decision-making rules of the application, while steganographic methods act as auxiliary safeguards that obscure sensitive operational details from unauthorised users.

At the beginning of the development process, the system is set up by configuring the required environment and establishing the foundational modules that handle the application's security operations. As the implementation progresses, the program logic responsible for processing votes and managing secure transactions is added, and supportive functions are created to ensure that data is formatted and handled consistently. Figure 5 illustrates how these components are organised, the key variables involved, and the sequence in which information flows through the secure processing pipeline.

```
contract VotingUnit {

    struct Entry {
        uint code;
        string title;
        uint total;
    }

    constructor() {
        addEntry ("Option X");
        addEntry ("Option Y");
    }
   }
```

Fig 5 – Code block to define a struct variable and contract

We have specified that the struct candidate has an ID of unsigned integer type, a name of string type, and a vote count of unsigned integer type. To store these structs, we use solidity mapping, which is like an associative array or a hash that associates key-value pairs.

```
mapping(UID => Voter) public
```

Here, the key to mapping is an unsigned integer, the value is Candidate structure type, and the mapping's visibility is set to public to get a getter function.

```
contract PollSystem {

    struct Option
     {
        uint ref;
        string title;
        uint tally;
      }
     mapping(uint => Option)
     public optionList;
     uint public optionTotal;
    function PollSystem()
     public
     {
        addOption ("Choice A");
        addOption ("Choice B");}
```

```
function pushItem(string memory
text) private {
          itemCount++;
          items[itemCount] =
      Item(itemCount, text, 0);
      }
```

Fig. 6. Code block of complete contract code

After developing a lightweight user interface was later developed to interact with the system's core services, focusing on providing a smooth and secure experience for participants. Instead of relying on traditional OTP-based checks, the client application uses a temporary session token generated by the backend to confirm a user's eligibility before granting access. The interface, created with a minimal combination of web technologies, handles only basic display and input tasks while delegating all verification routines to the backend. Once the session token is validated, the client forwards the voter's selection to the processing layer through protected communication routes, ensuring that no sensitive information is exposed during transmission. This design enables users to engage with the platform confidently, while the underlying architecture maintains data integrity, and ensures that each submitted vote is registered accurately.

After the user's credentials are confirmed, the interface interacts with the system's internal services through controlled request channels that restrict unauthorised access. Instead of directly handling sensitive data, the client forwards only the necessary information, and the backend manages all verification and processing tasks. This layered communication model ensures that vote submissions remain accurate and protected, allowing participants to use the platform with confidence while the system quietly maintains the security and consistency of every recorded vote.

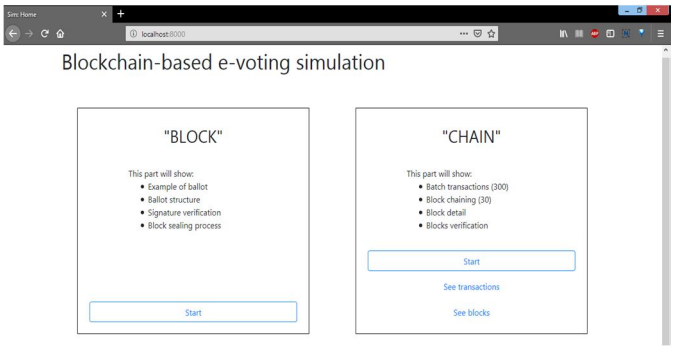


Fig 8 – Screenshot of the application during the mining process

To manage participation, the system keeps a small record linking each user to their voting status. A simple function receives an option identifier, verifies and marks their choice as completed, and then increases the tally for the chosen option. Figure 9 shows the compact logic used for registering a user's selection.

```
mapping(address => bool) public flagged;
function cast(uint _count) public {
require(!flagged[msg.sender]);}
 require(key > 0 && key <= itemCount);
 used[msg.sender] = true;
  items[key].score++;
  emit log Vote (key);
```

Fig 9 – Code Block for casting of vote/ vote process

When a vote is submitted, the transaction fee compensates the node (miner) that processes it, and once all entries are recorded, the system compiles the totals. The option with the largest count is then presented as the outcome.

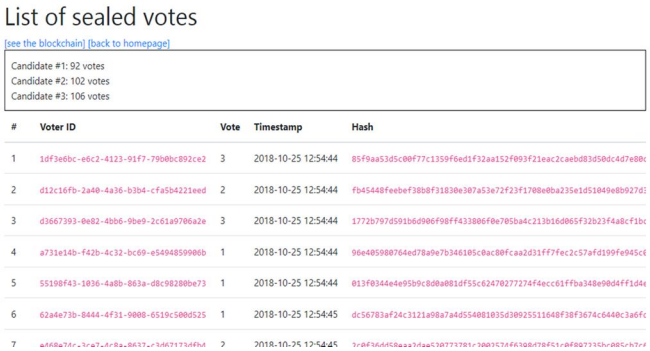


Fig. 10. Screenshot of election results

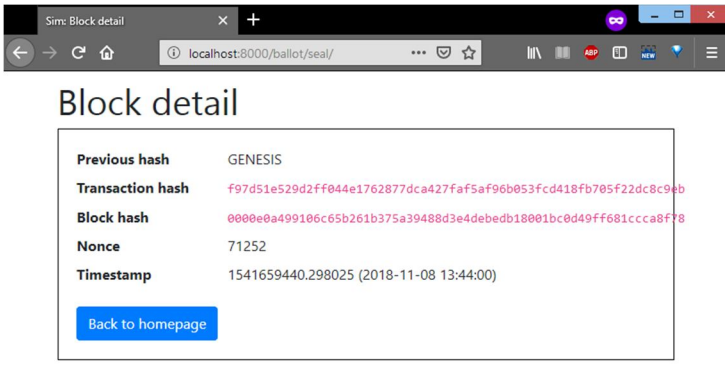


Fig. 11. Screenshot of a vote-casting entry in the chain.

Figure 10 illustrates the block updates that occur once a vote is recorded, while Figure 11 presents the transaction details, including the hash, block height, contract reference, and time of submission, account used, and overall resource cost. This system is designed for small, controlled elections such as campus-level polls, and although it functions reliably on the Ethereum network, scaling to nationwide participation would introduce additional challenges beyond the scope of this project. This system is intended only for compact voting environments, such as departmental or campus-level polls, where participation is relatively small and manageable.

Maintaining privacy in a blockchain voting environment is difficult because the technology is naturally designed for openness rather than secrecy. Most distributed ledgers expose transaction activity to all participants, which makes it challenging to hide the link between a voter and the recorded action. This lack of separation between identity and activity is unacceptable in formal elections where ballot secrecy must be guaranteed. Although several researchers, including F Hao et al., have explored cryptographic constructions that obscure voter information through key-based protocols, these approaches still leave practical gaps. As a result, creating a fully private yet verifiable blockchain voting model continues to be an unresolved technical challenge.

IV. CONCLUSION

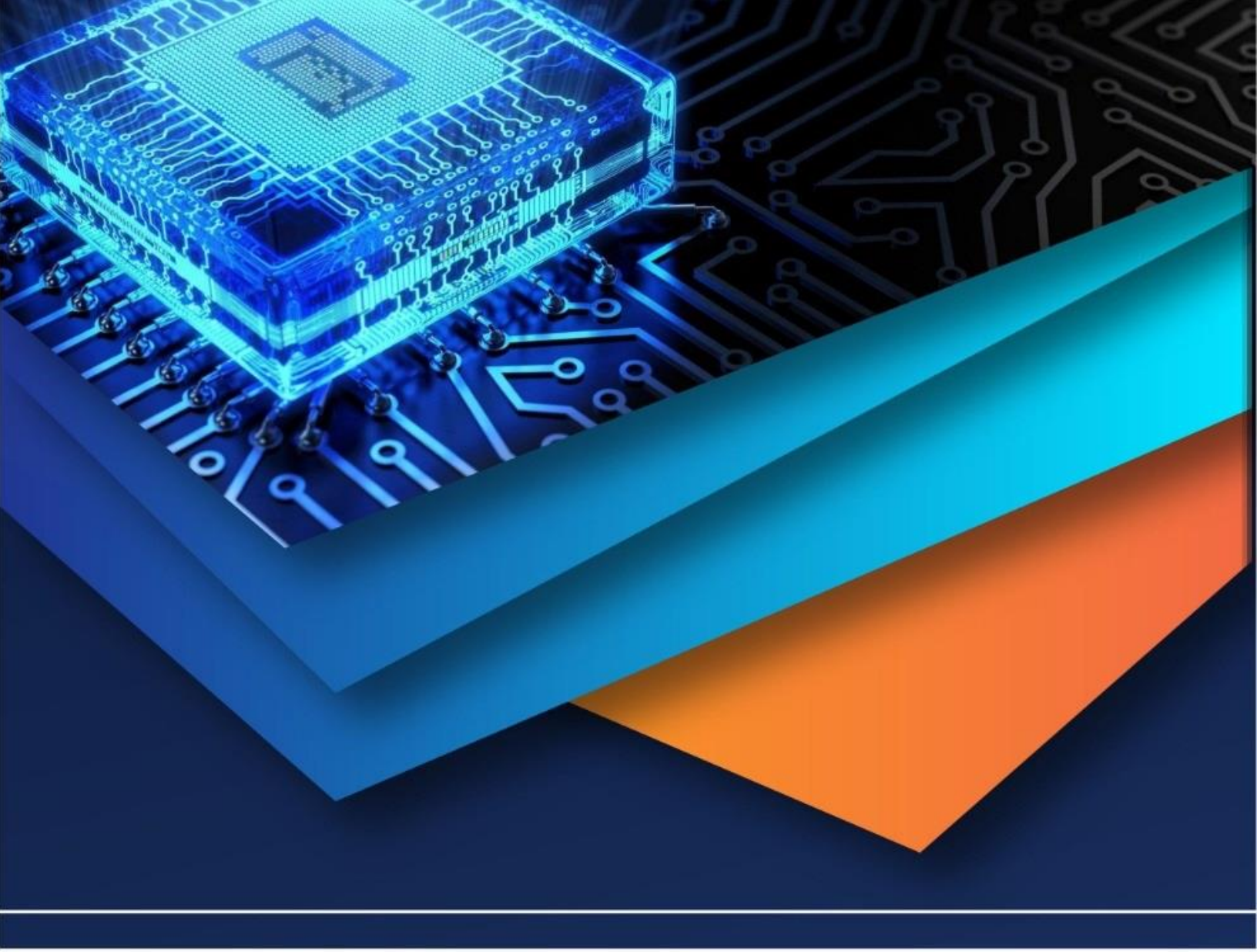
In this paper, we presented a conceptual voting platform that incorporates distributed-ledger concepts to reorganise the way digital ballots are handled and verified. Rather than positioning blockchain as a replacement for existing election infrastructures, the system demonstrates how decentralised record-validation can complement traditional digital workflows by offering a tamper-resistant method for tracking vote activity, consistent with established blockchain architectural principles [1]. For environments where administrative oversight is limited, such an approach can reduce dependency on central authorities and support more accountable governance practices [7]. Over the years, numerous electronic voting prototypes have attempted to modernise decision-making processes, but many fell short due to weaknesses in process validation, limited transparency, or operational rigidity. These challenges have motivated a shift toward voting frameworks that incorporate formal verification and traceable event structures, allowing system correctness to be evaluated more rigorously—an approach supported by findings in recent verification-focused research [8].

Likewise, advances in blockchain-enabled security models indicate that decentralised data protection mechanisms can significantly enhance auditability and reduce the risks posed by centralised vulnerabilities [9].

Unlike many earlier digital voting approaches that relied heavily on central servers, systems built on distributed-ledger principles—such as the prototype developed in this study—offer a more resilient foundation by decentralising verification and reducing points of failure. Even so, blockchain cannot independently satisfy every requirement of a real election. Critical tasks, particularly confirming the true identity of each participant, still depend on trusted external methods such as biometric checks or government-issued credentials, which must operate alongside the ledger to ensure complete system reliability [12]. Although blockchain introduces new opportunities for improving transparency and strengthening audit trails, its current form is not yet equipped to handle the scale, diversity, and operational demands of nationwide elections. Meaningful progress in areas like network capacity will be necessary before blockchain can fully supported.

REFERENCES

[1] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, “An extensive overview of blockchain technology, its structural components, and security mechanisms,” Technical Report, 2016.
[2] R. Krimmer and M. Volkamer, “A comparative study of modern electronic voting systems focusing on system requirements, operational risks, and evaluation criteria,” Proc. Int. Conf. on E-Governance Technologies, pp. 45–57, 2014.
[3] V. Buterin, “Design principles and conceptual architecture of a next-generation decentralised platform supporting smart contracts and distributed applications,” Whitepaper, 2015.
[4] J. Rodrigues and S. B. Cruz, “Foundational principles and deployment considerations for secure smart contract systems across distributed ledgers,” Journal of Distributed Computing Systems, vol. 12, no. 3, pp. 221–233, 2018.
[5] K. Sako and M. Kilian, “Cryptographic frameworks for protecting electronic ballots and ensuring confidentiality in remote digital elections,” Journal of Cryptographic Engineering, vol. 9, no. 2, pp. 89–102, 2017.
[6] T. Hardjono and N. Smith, “A decentralised identity model integrating trust, authentication, and blockchain-based verification,” IEEE Communications Standards Magazine, vol. 3, no. 4, pp. 28–37, 2019.
[7] H. Gupta and R. Sandhu, “Blockchain-enabled public governance models aimed at improving transparency and reducing administrative complexities,” Government Information Quarterly, vol. 36, no. 1, pp. 112–123, 2019.
[8] P. Y. A. Ryan and S. Schneider, “Formal analysis models and verification strategies for practical electronic voting systems,” International Journal of Information Security, vol. 16, no. 2, pp. 123–136, 2017.
[9] L. Chen, Z. Xu, and W. Shi, “A comprehensive survey of blockchain-supported security protocols and their applications in modern digital ecosystems,” IEEE Access, vol. 7, pp. 12345–12367, 2019.
[10] F. Hao, K. Kogias, and S. Patel, “Architectural considerations and common design patterns for privacy-preserving e-voting platforms,” Proc. Int. Workshop on Secure Computing Systems, pp. 77–88, 2018.
[11] T. Okamoto and A. Fujioka, “A cryptographic analysis of electronic voting systems supporting unconditional privacy, integrity, and end-to-end verifiability,” IEICE Transactions on Fundamentals, vol. E99-A, no. 5, pp. 1001–1010, 2016.
[12] Y. Zhang and J. Wang, “Applications, challenges, and emerging opportunities of distributed ledger technologies for securing public digital infrastructures,” Journal of Information Systems Security, vol. 14, no. 4, pp. 201–214, 2020.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY