International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 06 | Jun 2026 www.irjet.net p-ISSN: 2395-0072

Intelligent Blockchain-Enabled E-Voting System for Secure Elections

K. V. N. Satish[1], V. Prasanna Lakshmi [2]

[1]*Final year MCA Student, Department of Computer Applications, GIET Engineering College, Andhra Pradesh, India*
[2] *Assistant Professor, Department of Computer Applications, GIET Engineering College, Andhra Pradesh, India*

Abstract – *Electronic voting has the potential to improve accessibility, speed, and administrative efficiency in elections, but it also raises serious concerns related to voter authentication, vote tampering, duplicate voting, and trust in result aggregation. This paper presents an implemented prototype of an intelligent blockchain-enabled e-voting system that combines web-based election management with multi-factor voter authentication and blockchain-backed vote logging. The system is developed using the Django framework and integrates Aadhaar-number-based voter identification, email-based one-time password verification, facial authentication, encrypted vote storage, and Ethereum smart contract support. In the proposed workflow, an election authority registers voters and political parties through a controlled administrative portal. During voting, a registered voter first enters the Aadhaar number used as the system identifier, receives an OTP through the registered email address, and then undergoes face verification before being allowed to cast a vote. To improve robustness in practical webcam conditions, the facial verification stage includes brightness and contrast enhancement before extracting facial encodings. Once the vote is cast, the selected vote is encrypted using AES, linked to the previous record through hashing, and optionally logged on an Ethereum-compatible blockchain through a Solidity smart contract and Web3 integration. This design improves confidentiality, tamper resistance, auditability, and resistance to impersonation. The implemented prototype demonstrates that a layered authentication and storage model can significantly strengthen trust in digital voting workflows. The system successfully supports voter registration, OTP delivery, facial verification, duplicate vote prevention, encrypted vote persistence, and blockchain transaction logging on a local Ethereum test network. The paper discusses the architecture, methodology, implementation details, functional results, observed limitations, and future improvements needed to transform the prototype into a large-scale production-ready election platform.*

Key Words: E-voting, blockchain, OTP authentication, facial verification, smart contracts, Django, AES encryption, Ethereum, vote security, digital elections

1. INTRODUCTION

Election systems form the foundation of democratic governance and the credibility of an election depends on the public's confidence that votes are cast only by eligible voters, counted exactly once, stored without alteration, and reported transparently. Traditional paper-based elections provide physical auditability, but they are often slow, resource-intensive, and vulnerable to logistical difficulties such as long queues, manual counting delays, ballot mishandling, and limited accessibility for some groups of voters. Electronic voting systems emerged as a response to these difficulties by offering faster result computation, simplified administration, and the possibility of remote or semi-remote participation. However, electronic voting introduces a new class of threats involving cyberattacks, unauthorized access, privacy leakage, and manipulation of centrally stored data.

A major challenge in digital voting is voter authentication. In a secure election, the system must confirm that the person attempting to vote is the legitimate registered voter and not an impersonator. At the same time, the authentication process should not become so complex that it discourages participation. Many practical e-voting systems rely on usernames, passwords, smart cards, biometric devices, or institutional voter IDs, but each approach has trade-offs. Password-based systems can be shared or stolen. Card-based systems require hardware infrastructure. Centralized biometric systems can raise privacy concerns if not carefully designed. For this reason, stronger and layered verification models are needed in high-trust election environments.

Another important issue is result integrity. In many existing digital systems, vote records are stored in centralized databases controlled by a single administrative layer. While centralized storage is operationally simple, it creates a single point of failure and a single point of trust. If the database is compromised or altered, the legitimacy of the election becomes difficult to defend. Blockchain technology addresses this concern by providing an append-only ledger in which transactions are cryptographically signed, timestamped, and linked in a tamper-evident manner. Even when blockchain is not used to store full ballot contents, it can still be used effectively to store vote hashes or integrity proofs, thereby improving transparency and auditability.

The project presented in this paper addresses these challenges by combining several security layers into one implemented web-based election platform. The system uses Aadhaar number as a voter identifier within the application, email OTP as the first authentication factor, face verification as the second authentication factor, AES encryption to protect vote content at rest, SHA-256-based

hash linkage to strengthen record integrity, and an Ethereum smart contract to store vote hashes on-chain. Administrative functions such as voter registration, party registration, and result monitoring are separated from voter-facing functions such as OTP-based login, facial verification, and vote submission.

Unlike purely conceptual voting models, this work is based on an implemented prototype that supports the complete workflow from registration to vote casting and result observation. At the same time, the paper avoids unsupported performance claims and describes the system as an implemented prototype rather than a nationally deployable production platform. The objective is to demonstrate a secure architecture, validate its functional workflow, and show how layered security can improve trust in digital election systems. The remainder of the paper reviews related work, explains the architecture and methodology, discusses implementation results, and identifies directions for future enhancement.

2. LITERATURE SURVEY

Traditional electronic voting systems were developed to reduce the delays and costs associated with paper ballots and manual counting. These systems improved counting speed and administrative convenience, but many of them retained centralized designs in which election records were fully controlled by a single server or authority. Such models are vulnerable to insider manipulation, system compromise, and limited transparency because ordinary voters and external auditors cannot easily verify whether stored records remain unchanged throughout the election cycle. This limitation has motivated researchers to explore cryptographic and distributed approaches for election security.

Blockchain-based voting systems have gained significant attention because distributed ledgers provide immutability, chronological ordering, and strong audit trails. In blockchain voting, ballots or ballot proofs can be stored as transactions, making later tampering significantly more difficult. Smart contracts can automate parts of the election process such as recording votes, validating phases, or exposing public counts. Researchers have shown that blockchain is especially valuable in strengthening trust, but they also note practical concerns such as privacy, transaction cost, throughput, and integration complexity. As a result, many realistic voting architectures use blockchain selectively for integrity verification rather than storing every detail of the ballot directly in plain form.

Biometric authentication has become another major research direction in secure e-voting. Fingerprints, iris scans, and facial recognition have all been studied as methods for linking a vote attempt to a real human identity. Face recognition is particularly attractive in modern web-based systems because it can operate through ordinary cameras without specialized hardware. However, biometric systems are sensitive to environmental conditions such as lighting, pose variation, image blur, and spoofing attempts. Therefore, practical implementations often combine biometrics with additional verification layers rather than using them as the sole gatekeeper.

Face recognition technologies have advanced significantly with the use of facial encodings, embedding-based matching, and deep learning models. Some research uses large convolutional neural networks, while practical applications often rely on lightweight encoding and comparison pipelines for easier deployment. A common implementation challenge is not only recognizing the face but detecting it reliably in real-world conditions. Preprocessing methods such as brightness correction, contrast enhancement, and multiple detection passes are often necessary to improve detection consistency. In applied election systems, the value of face recognition lies not in theoretical novelty alone, but in whether it helps reduce impersonation while remaining usable for real voters.

Smart contract security is also central to blockchain voting research. A smart contract must store vote-related data in a deterministic and verifiable way without exposing private ballot information. Researchers have highlighted common smart contract concerns including unauthorized function access, replay risks, poor input validation, and contract upgrade limitations. In a voting context, a contract that stores only vote hashes can reduce privacy leakage while still creating a trustworthy integrity layer. This paper follows that practical direction by using a simple Solidity contract to store vote hashes and timestamps rather than exposing plaintext vote data on-chain.

Recent work on edge and lightweight computation emphasizes that identity verification systems should remain efficient enough for deployment on ordinary consumer devices and standard web infrastructure. Although this project does not implement a full edge-computing architecture, it aligns with that broader goal by using a browser-based capture process and server-side facial encoding rather than relying on specialized devices. The literature suggests that secure voting systems are strongest when they combine usability, cryptography, biometric verification, and transparent record keeping, which is precisely the direction taken by the implemented prototype described in this paper.

3. PROBLEM STATEMENT

Conventional online voting systems often suffer from four major weaknesses: weak voter authentication, centralized

vote storage, poor transparency, and difficulty in preventing duplicate or fraudulent voting attempts. If a system depends only on a username or numeric identifier, unauthorized users may attempt impersonation. If vote data is stored only in a traditional database, malicious modification or accidental corruption may be difficult to detect. If results are announced without an auditable trail, stakeholders may question election legitimacy. Finally, if the system lacks strong duplicate-vote prevention logic, the fairness of the election is compromised.

The problem addressed in this work is therefore the design and implementation of a practical web-based voting platform that verifies voter identity more strongly than simple credential-based systems, protects vote confidentiality, preserves record integrity, and provides an auditable trail for vote submission. The proposed solution must remain deployable with accessible technologies such as web browsers, email services, standard servers, and a blockchain test network.

4. EXISTING SYSTEM

Many existing digital voting systems rely on centralized architectures in which an administrator manages voter lists, and votes are stored in a backend database. These systems can improve convenience and reduce counting time, but they often depend on single-layer authentication such as login credentials, voter IDs, or static verification codes. Such methods are not always sufficient to prevent impersonation, especially when credentials are shared, leaked, or reused. In some systems, voters may authenticate successfully even when the platform has no way to confirm that the actual person behind the device is the legitimate registered individual.

Another limitation of conventional systems is the absence of a tamper-evident storage mechanism. If votes are inserted, altered, or deleted in a centralized database, external observers may have no independent way to detect the change. This is particularly risky in sensitive election scenarios where trust is as important as functionality. Moreover, systems that do not validate liveness or facial identity can remain vulnerable to proxy voting, fake registrations, or impersonation during remote participation.

Existing web-based systems also tend to separate user verification and data integrity as unrelated concerns. One module may manage identity, while another stores the vote, without a unified security model linking the two. As a result, a user may pass login verification but still submit a manipulated or duplicated vote record. The present project improves on this by using layered authentication and combining local encrypted storage with blockchain-backed integrity logging.

5. PROPOSED SYSTEM

The proposed system is a secure e-voting prototype designed to integrate administrative management, voter identity verification, encrypted vote recording, and blockchain transparency in a single workflow. It is implemented using Django for the web application, SQLite for the core data store, SMTP-based email delivery for OTP verification, Python-based face recognition for biometric authentication, AES encryption for vote confidentiality, and Ethereum smart contracts for blockchain logging.

The system is divided into two main roles: election authority and voter. The election authority logs into an administrative portal, registers voters, uploads facial reference images, and manages political parties. Voter registration includes capturing the voter's face image and confirming registration through an OTP sent to the voter's email. This ensures that the system creates verified voter profiles before election participation begins.

During voting, the voter first enters the Aadhaar number used as the internal voter identifier. The system sends a one-time password to the registered email address and grants further access only after correct OTP validation within a limited time period. Once OTP verification succeeds, the voter proceeds to the facial verification stage. The system captures a live image from the webcam or accepts an uploaded image, enhances the image when necessary, extracts facial encodings, and compares them against the registered voter image. Only when the face matches and no previous vote exists does the voter receive access to the party list.

After party selection, the vote is processed in multiple security layers. First, the vote content is encrypted using AES before being stored locally. Second, the vote record is linked through a previous-hash mechanism and assigned a SHA-256 hash, creating a tamper-evident chain within the application database. Third, when blockchain mode is enabled, the hash is submitted to a Solidity smart contract through Web3 and recorded on an Ethereum-compatible chain. This hybrid design gives the system both practical application-layer functionality and a separate blockchain integrity trail.

The proposed system is therefore not a pure blockchain-only voting model, but a layered secure voting architecture. It uses the application database for operational workflow and result handling, while using blockchain as an integrity assurance layer. This makes the system easier to implement and test while still demonstrating how blockchain can improve trust in vote recording.

6. METHODOLOGY

The methodology of the implemented prototype follows the full voting life cycle from voter preparation to secure vote recording. The process begins with administrative access. Election authorities authenticate through a protected admin login page and receive access to modules for adding voters, managing parties, viewing voter records, and checking election results. This administrative separation ensures that only authorized personnel can modify election configuration before the voting phase begins.

The voter registration phase is handled by the election authority. The admin enters the voter's personal details including full name, Aadhaar number, age, email, mobile number, and address. A facial reference image is captured through the browser and stored as part of the voter profile. Before the registration becomes final, the system generates a six-digit OTP and sends it to the voter's email address. Registration is completed only if the submitted OTP matches the stored OTP. This process prevents incorrect or fraudulent registration entries from being activated without email ownership confirmation.

The next phase is voter login. The voter enters the Aadhaar number into the login interface, and the system checks whether the Aadhaar number exists in the verified voter registry. If a valid record exists, the application generates a fresh OTP and sends it to the email address registered for that voter. The OTP is stored in the session along with the corresponding Aadhaar number and a timestamp. The voter must enter the correct OTP before the session expires. This step reduces the possibility of unauthorized access by requiring control over the voter's registered email inbox.

After successful OTP verification, the facial verification stage is triggered before voting access is granted. The voter captures a current image through the webcam or uploads an image. The application stores the submitted image temporarily and processes it using the face_recognition library. To improve reliability in low-light or low-contrast conditions, the system performs brightness and contrast enhancement on image variants and attempts facial detection multiple times. It then extracts a facial encoding from the temporary image and another encoding from the registered image stored during voter enrollment. These encodings are compared using a defined tolerance threshold. If no face is detected clearly or the encodings do not match, the voter is rejected and voting is blocked. If the face matches, the system proceeds to the ballot stage.

Before presenting the ballot, the application checks whether a vote already exists for the given Aadhaar number. This duplicate-vote prevention rule is enforced at the application layer and stops the same voter from casting multiple ballots. Once the voter passes this check, the system fetches the list of registered political parties and displays them for selection. The voter then chooses the intended party and submits the vote through the application interface.

Vote processing begins immediately after submission. The selected party name is first assigned as the vote content. Before it is stored in the database, the system encrypts the vote using AES in CBC mode. An initialization vector is generated, the vote is padded and encrypted, and the encrypted payload is stored in encoded form. At the same time, the system retrieves the hash of the previous vote record, if one exists, and generates a new SHA-256 hash based on the current record. This creates a linked sequence of vote records within the database and improves resistance to silent tampering.

If blockchain mode is active, the application then loads the contract ABI, establishes a connection to the Ethereum-compatible node through Web3, and uses the configured private key to sign a transaction calling the storeVote function in the Solidity contract. The transaction stores the vote hash on-chain and returns a transaction receipt containing the blockchain transaction hash. This transaction hash is saved in the vote record for later audit reference. If blockchain mode is disabled, the vote remains protected in the encrypted and hashed database layer, but no on-chain transaction is created.

The result phase aggregates votes at the administrative side. The application reads vote records, decrypts the stored vote payloads when needed, counts votes by party, and determines the majority result. When blockchain is enabled and correctly synchronized, the system also reads the vote count stored in the smart contract and compares it with the database vote count. A match indicates consistency between local records and on-chain integrity proofs. A mismatch indicates that some votes were stored only locally, the blockchain was enabled after earlier votes had already been cast, or the chain state was reset independently of the database. This behavior was observed during prototype testing and is an important practical consideration for deployment.

Overall, the methodology demonstrates a layered security workflow rather than a single-security-point model. Identity validation is strengthened through OTP and facial verification. Vote confidentiality is improved through AES encryption. Record integrity is reinforced through hash chaining. Audit transparency is enhanced through blockchain logging. Together, these mechanisms create a more trustworthy digital voting architecture than systems that rely solely on centralized credential checks and plain database storage.

7. ALGORITHMS AND TECHNOLOGIES USED

7.1 OTP AUTHENTICATION

The system uses a six-digit randomly generated OTP for both voter registration confirmation and voter login verification. OTP values are stored temporarily and validated within a fixed expiration window. This mechanism ensures that possession of the registered email account is required before access is granted.

7.2 Face Recognition and Image Enhancement

Facial verification is implemented using the Python face_recognition library. The system loads the registered face image and the live or uploaded image, enhances brightness and contrast when needed, extracts facial encodings, and compares them using a configurable tolerance threshold. This supports stronger identity verification than password-only or OTP-only approaches.

7.3 AES Encryption

Vote confidentiality is maintained using AES encryption in CBC mode before database storage. The encrypted vote is stored instead of plain vote text, which prevents direct reading of ballot content from the database by unauthorized actors.

7.4 SHA-256 Hash Chaining

Each vote record stores a hash and a reference to the previous record's hash. This creates a linked sequence of records. If a vote is altered later, the integrity of the chain is affected, making tampering easier to detect.

7.5 Ethereum Smart Contract

A Solidity smart contract stores vote hashes and timestamps on an Ethereum-compatible blockchain. The contract supports vote-hash insertion and vote-count retrieval, allowing the application to verify that blockchain and application-layer vote records are aligned.

7.6 Web3 Integration

The Django application communicates with the blockchain node using Web3. Transaction building, signing, submission, and receipt handling are all performed through this integration.

8. RESULTS AND DISCUSSION

The implemented prototype was validated functionally across the major election workflow components. The election authority module successfully supported admin login, party creation, voter registration, and voter-list management. The OTP module successfully delivered verification codes to registered email addresses after SMTP configuration was completed. The login flow correctly rejected unregistered Aadhaar numbers, expired OTP sessions, and invalid OTP submissions. This behaviour shows that the first-factor authentication layer is operating as intended.

The facial verification module also functioned successfully after improvements were made for practical webcam conditions. During early testing, low-light images and low-contrast captures occasionally caused the system to reject genuine users because the facial encoding could not be extracted reliably. To reduce this problem, the implementation was enhanced with multiple face-detection passes and image preprocessing through brightness and contrast adjustment. After these changes, facial verification became more stable under normal indoor conditions, though it still depends on reasonable camera clarity and proper user positioning. This is an important practical finding because biometric accuracy in real systems is affected not only by algorithms but also by environmental conditions.

The vote submission module demonstrated successful duplicate-vote prevention. Before storing a vote, the system checks whether a vote already exists for the same Aadhaar number. If a record is found, the voter is blocked from voting again. This simple but effective check prevents repeated submissions from the same registered identity. Once a valid vote is accepted, the encrypted vote, previous hash, and current hash are stored in the database, demonstrating that confidentiality and local integrity mechanisms operate together.

Blockchain integration was also validated on a local Ganache-based Ethereum test network. After contract compilation, deployment, and environment configuration, the application successfully submitted vote hashes to the smart contract and received transaction hashes in return. These transaction hashes were stored alongside the vote records, allowing later audit reference. This confirms that the blockchain logging layer is operational when the blockchain environment is active and properly synchronized with the application.

An important practical observation emerged during result verification. When some votes existed in the database before blockchain logging was enabled, or when the blockchain workspace had been restarted independently, the vote count stored in the database and the count stored in the smart contract did not match. In such cases, the result page displayed blockchain verification as false. This is not necessarily a system failure; rather, it reflects real state inconsistency between local storage and blockchain history. The observation highlights that blockchain-backed election systems require consistent chain lifecycle management. Once an election begins, the blockchain environment should remain stable and vote logging should

stay enabled for all ballots if strict one-to-one verification is expected.

9. CONCLUSIONS

This paper presented an implemented prototype of a secure e-voting system that combines web-based election management with multi-layer voter authentication and blockchain-assisted vote integrity. The system was designed to address some of the most important weaknesses of conventional online voting platforms, including weak user authentication, centralized trust dependency, duplicate vote risk, and poor result auditability. By integrating Aadhaar-number-based identification, email OTP verification, facial authentication, encrypted vote storage, hash-based linking, and blockchain vote-hash logging, the proposed system establishes a stronger security posture than ordinary credential-driven election portals.

The implemented workflow demonstrates that security in digital elections should not depend on a single control point. OTP alone can verify access to an email account, but it cannot confirm physical identity. Face verification alone can confirm biometric similarity, but it should not operate without session-level or identity-level controls. Database storage alone may support convenient result management, but it does not provide an independent tamper-evident audit layer. The proposed architecture combines these mechanisms so that each module supports the others. This layered design is the central strength of the system.

Prototype testing showed that the modules function correctly when configured and used within expected conditions. OTP-based registration and login were successful, facial verification became more reliable after image-enhancement improvements, duplicate voting checks were enforced correctly, encrypted vote storage worked as intended, and blockchain transaction logging was successfully demonstrated on an Ethereum-compatible test network. The observation of blockchain/database mismatch in unsynchronized scenarios also provided a realistic operational lesson: blockchain-enabled elections require stable infrastructure and consistent transaction logging across the entire voting period.

In summary, the system demonstrates a practical and meaningful direction for secure digital elections. It improves transparency, strengthens authentication, reduces fraudulent voting opportunities, and introduces a verifiable transaction layer for election records. While the prototype is not yet a nationwide production-ready platform, it provides a strong foundation for future secure election systems and shows that combining web technologies, biometrics, cryptography, and blockchain can significantly improve trust in electronic voting.

11. FUTURE SCOPE

Several improvements can extend this prototype into a more scalable and production-oriented system. A stronger administrative authentication model should replace the current static admin credentials. A dedicated liveness-detection module can be added to reduce spoofing risks from printed photos or replayed video. The blockchain layer can be migrated from a local Ganache environment to a permissioned consortium blockchain or a managed Ethereum-compatible deployment for more realistic multi-node operation. Database design can also be improved to support larger voter populations and more formal audit reporting.

Future work may also include multilingual interfaces, mobile-friendly voting workflows, accessibility features for elderly or differently abled users, and stronger election analytics dashboards. Additional experimental work should measure authentication time, face verification reliability under varied conditions, blockchain latency, and usability performance with larger user groups. With these enhancements, the system could evolve from an academic prototype into a more mature and field-ready secure election platform.

REFERENCES

1) S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008.
2) National Institute of Standards and Technology, “FIPS PUB 197: Advanced Encryption Standard (AES),” 2001.
3) N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” Proceedings of CVPR, 2005.
4) F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering,” Proceedings of CVPR, 2015.
5) G. Wood, “Ethereum: A Secure Decentralised Generalised Transaction Ledger,” Ethereum Yellow Paper, 2014.
6) D. Chaum, “Secret-Ballot Receipts: True Voter-Verifiable Elections,” IEEE Security & Privacy, vol. 2, no. 1, pp. 38–47, 2004.
7) M. Bellare, D. Canetti, and H. Krawczyk, “Keying Hash Functions for Message Authentication,” Advances in Cryptology, 1996.
8) Django Software Foundation, “Django Documentation,” available through the official Django project documentation.
9) Web3.py Documentation, official documentation for Ethereum client interaction in Python.
10) Python face_recognition and underlying dlib-based facial encoding documentation for implementation reference.