© FEB 2026 | IRE Journals | Volume 9 Issue 8 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I8-1714243

Secure and Unified AI – Blockchain Voting System with Real Time Verification

J. INDRA SIBI[1], B. SHANMUGA SUNDARI[2], M. ELZA MELIF[3]
[1]*PG Student, Department of CSE, PET Engineering College, Vallioor*
[2, 3]*Assistant Professor, Department of CSE, PET Engineering College, Vallioor*

Abstract- Online voting for independent elections is generally supported by trusted election providers. Typically these providers do not offer any way in which a voter can verify their vote, and hence the providers are trusted with ballot privacy and in ensuring correctness. Despite the desire to offer online voting for political elections, this lack of transparency and verifiability is often seen as a significant barrier to the large-scale adoption of online elections. Adding verifiability to an online election increases transparency and integrity, as well as allowing voters to verify that the vote they cast has been recorded correctly and included in the tally. However, replacing existing online systems with those that provide verifiable voting requires new algorithms and code to be deployed, and this presents a significant business risk to commercial election providers, as well as the societal risk for official elections selecting for public office. In this paper we present the first step in an incremental approach which minimizes the business risk but demonstrates the advantages of verifiability, by developing an implementation of key elements of a Selene-based verifiability layer and adding it to an operational online voting system. Selene is a verifiable voting protocol that publishes votes in plain text alongside a voter's tracker. These trackers enable voters to confirm that their votes have been captured correctly by the system, such that the election provider does not know which tracker has been allocated to which voter. This results in a system where even a "dishonest but cautious" election authority running the system cannot be sure of changing the result in an undetectable way, and hence gives stronger guarantees on the integrity of the election than were previously present. We explore the challenges presented by adding a verifiability layer to an operational system. The system was used in two initial trials conducted within real contested elections. We conclude by outlining the further steps in the road-map towards the deployment of a fully trustworthy online voting system.

I. INTRODUCTION

A significant problem with the traditional voting systems is the vulnerability to manipulation, security breaches, and inefficiencies that undermine the fairness, transparency, and integrity of the electoral process. One major issue is voter impersonation and fraud, where unauthorized individuals may cast votes under false identities[1] . This is a common challenge in regions with weak authentication mechanisms or poorly managed voter registration processes. Even in electronic voting systems, the lack of robust verification techniques often exposes the process to fraud and manipulation, which compromises the credibility of election results[2]. Another concern is the lack of transparency in vote counting, where manual or semi-automated systems introduce the possibility of human error or deliberate tampering. This raises doubts among voters about whether their votes are accurately counted, especially in close elections where every vote matters. Additionally, the lack of standardization across different regions and states introduces complexities and inefficiencies in managing elections, leading to inconsistencies in election procedures and results[3]. Finally, voters often have no way of verifying whether their vote has been correctly cast and recorded. Without mechanisms to allow individuals to check the status or integrity of their vote, concerns about vote tampering or errors persist. This problem is further compounded by the growing threat of cyber-attacks and unauthorized access to voting systems, which can manipulate or steal voter data, ultimately affecting the election outcome[4]. This project aims to tackle these issues by leveraging blockchain technology for secure, transparent, and immutable vote storage, along with AI-driven facial recognition for robust voter authentication. The use of blockchain ensures that once a vote is cast, it cannot be tampered with, providing an unprecedented level of trust and security. The integration of AI for face recognition adds an extra layer of protection against fraud and identity theft, while ensuring that voters can be easily and accurately verified[5]. The aim of the project is to design and implement a secure, transparent, and efficient blockchain-based electronic voting system that integrates multi-level authentication mechanisms, including QR codes and CNN-based face recognition linked to Aadhaar, to ensure accurate voter identification, tamper-proof vote storage, and real-time result verification, thereby addressing the limitations of existing voting systems[6]. The project leverages advanced technologies such as QR codes and CNN-based face recognition linked to Aadhaar for multi-level voter authentication[7]. This ensures accurate voter identification and prevents unauthorized access. Blockchain technology further enhances security by encrypting and immutably storing votes using 256-bit SHA hash codes, safeguarding the electoral process against tampering.By integrating blockchain's immutable ledger, the system ensures that votes remain unaltered and traceable[8]. A Vote Integrity Verifier Link notification system alerts voters of any tampering attempts, allowing real-time verification of their vote's authenticity[9]. The project introduces a unified electronic voting platform to standardize elections across all Indian states. and management, improving the overall efficiency of the electoral process[10].

II. LITERATURE SURVEY

Secure Online Voting System-Based on Facial Recognition by Using Deep Learning , Krishna Prakash; Nimmagaddda Vatsalya Mitra; Nallamothu Pavan Kumar; Manda Anji Babu; Shonak Bansal; Sandeep Kumar invented in 2025.The objective of this study is to design and implement a secure, reliable, and transparent online voting system for corporate elections by integrating deep learning–based facial recognition and OTP authentication to eliminate fraudulent voting and unauthorized access. The proposed system combines computer vision and deep learning techniques for identity verification and secure voting. It uses employee-specific identifiers such as employee ID and biometric credentials to authenticate users. The voting process is protected with OTP-based verification, and all votes are validated in real-time to ensure transparency and security.The system employs Haar Cascade Classifier for fast and efficient face detection and Convolutional Neural Networks (CNNs) for high-accuracy facial recognition and verification. The combination ensures quick processing with strong authentication accuracy[1].

A Comprehensive Evaluation of Secured Electronic Voting System Design Based on Face Biometric Authentication Policy, Pandarinath Potluri; R. Jayakarthik; Shivam Agarwal; Shobana S; Venkata Padmavathi S; Aarthi R invented in 2024. The main objective of this study is to design a secure, transparent, and efficient electronic voting system that leverages face biometric authentication to ensure voter identity verification and prevent electoral fraud. The proposed system integrates both hardware and software components, including cameras for facial capture, a biometric database, and encryption-based communication protocols. Deep learning-based face detection and recognition algorithms are implemented along with anti-spoofing and multi-factor authentication mechanisms. The architecture covers voter registration, ballot generation, secure vote casting, and encrypted vote counting.Deep learning algorithms for face detection and recognition, supported by anti-spoofing and encryption techniques, form the core of the system[2].

Enhancing the Security of Online Voting System Using Defined Biometrics, *Devanshi Malik; Kritika Tripathi; Jyotsna invented in 2023.* The objective of this study is to develop a secure, efficient, and user-friendly online voting system that eliminates the shortcomings of manual voting by integrating biometric and multi-step verification mechanisms.The proposed web-based system employs Aadhaar verification, biometric fingerprint scanning, and two-step authentication for voter validation[3].

Coercion-Resistant E-Voting Scheme with Blind Signatures, Ahsan Aziz he found in 2019. This project is an e-voting scheme based on blind signatures which fulfils important security requirements and is efficient too.An ideal e-voting system would allow users to go online, using web-browser or a phone application, enter their credentials and vote; it would also allow voters to verify their votes after election. The properties that make e-voting such a promising technology also raise potential privacy and efficiency problems. In literature, researchers have listed the requirements which an e-voting scheme must have. Many e-voting schemes have been proposed which are based on combination of cryptographic tools, however some schemes have efficiency problems, voter or election authority does a lot of processing at their end, and some do not fulfil all security requirements[4] .

III. EXISTING SYSTEM

The existing voting systems predominantly rely on traditional methods such as paper ballots and electronic voting machines (EVMs). While these systems have served for decades, they present significant limitations that impact the integrity, efficiency, and inclusivity of the electoral process.

Paper-Based Voting
Paper ballots, one of the oldest voting methods, are still used in certain regions due to their simplicity. However, this system is prone to errors in vote counting and is labor-intensive, resulting in delays in result announcements. Furthermore, paper ballots are vulnerable to tampering, which compromises election fairness[11].

Electronic Voting Machines (EVMs) EVMs were introduced to address the inefficiencies of paper-based voting by providing faster and more accurate vote counting. However, these machines face challenges such as limited voter verification mechanisms and centralized vote storage, which can raise concerns about data security and trust[12].

DISADVANTAGES
1)Risk of tampering and data breaches in both paper ballots and EVMs.
2)Manual counting and centralized systems delay result announcements.

IV. PROPOSED SYSTEM

The proposed system is designed to overcome the limitations of traditional voting methods. By integrating cutting-edge technologies such as blockchain, facial recognition, and multi-level authentication, this system aims to ensure a more secure, efficient, and transparent electoral process.

1)Multi-Level Authentication for Voter Security
The system incorporates multi-level authentication, using QR codes and facial recognition powered by Convolutional Neural Networks (CNN). This method links voter identity with Aadhaar for precise verification, enhancing security and minimizing the possibility of identity fraud or voter impersonation.

2)Blockchain Technology Integration
The use of blockchain technology in the proposed system ensures that votes are encrypted and securely stored, guaranteeing immutability and preventing tampering. Each vote is traceable, and any unauthorized changes are immediately detected, triggering an alert system to ensure the integrity of the election process.

3)Self-Tallying Vote Mechanism
A key feature of the system is its self-tallying mechanism, which automatically counts the votes at the end of the election day. This eliminates the need for manual vote counting, resulting in faster and more accurate results. Additionally, the self-tallying system allows for same-day result announcements, improving the efficiency of the election process.

Advantages
1)Blockchain ensures immutability and protects votes from tampering.
2)Self-tallying mechanism enables same-day vote counting and results.
3)Facial recognition and Aadhaar ensure secure voter identification.

V. SYSTEM ARCHITECTURE

As described, our approach is to enhance an existing system with a verifiability layer in order to understand how such systems can be improved. Having chosen Selene as a suitable protocol to provide verifiability, in this section we describe both the existing system and how Selene is layered over it.

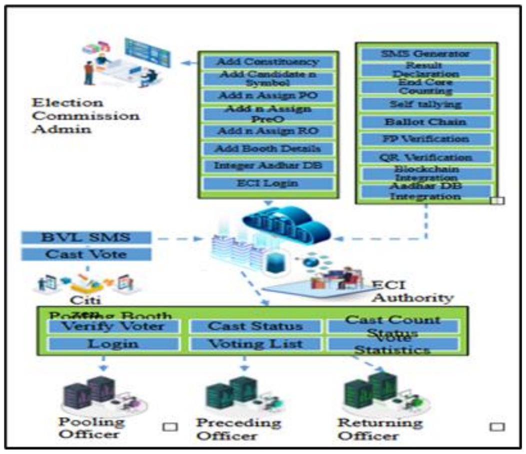


Fig 5.1 System Architecture

A.Legacy System

Our chosen commercial partners are CES, who run a signif- icant number of elections within the U.K. using a web-based voting system. Their system allows for online, postal and tele- phone voting to take place for up to hundreds of thousands of voters, with all cast ballots recorded in a relational database secure web service operating from their data centres on Microsoft Windows servers[13]. When an election is to be run, CES receive details of the election type, voters, materials needed to be released to voters, such as candidate information, and tallying rules. Each election may consist of multiple races (different questions upon which each voter may vote). These data are then used to provision a suitable section of the CES web service which can be accessed by voters using their credentials when voting is opened[14]. The database contains a list of all of the voters, their credentials and record of their vote (empty prior to voting). When voting is opened, voters receive their security creden- tials which enable them to log into the web service via any supported browser using HTTPS and cast their ballot. These credentials may be received via email or by post. Once a voter has cast their ballot, the corresponding database table is updated to contain the plaintext (machine readable) vote for every race in which the voter has voted. Postal and telephone votes may be received for certain elections, in which case the corresponding votes are recorded within the same database table by customer service representatives[15]. Furthermore, it is possible for a voter to cancel their vote, say by telephone, in order to re-vote if they wish. At the end of the election, the tallying rules are applied to the collected votes and the results released to the commissioning organisation. CES do not make the election results public as it is the responsibility of the organisation to release the results depending upon their own rules. This system therefore places complete trust within CES to run the election[16] . This includes notifying all voters, providing the election materials and credentials, maintaining the integrity of the collected votes, tallying the votes using the correct rules, and releasing the corresponding results to the commissioning organisation [17]. Since votes are held in plaintext and there is no way for a voter to verify their vote, trust is placed in the security of the system and the integrity of the staff. The use of plaintext votes is integral to the system in order to perform the tally[18].

B.Design

In implementing a verifiability layer with Selene, there are two overriding requirements: 1) to provide individual and universal verifiability of the election, and 2) to ensure that the established system remains intact in case the verifiability layer fails. This latter requirement is driven by business need: when operating a large-scale, commercial election which is trialling experimental software, there must be a mechanism whereby the election can be easily recovered without loss of data. Indeed this requirement dictates that the storage of plaintext votes and existing tallying mechanism remain as-is while the software is at the experimental stage and undergoing trials[18]. Yet despite this, by adding voter verification and publishing the election results publicly, the election becomes transparent and, importantly, verification is able to expose any malicious change in the election result, thus reducing the required trust in the election provider[19]. Once the experimental software is proven and made sufficiently robust for production use, then the requirement to maintain the existing system is removed. As a consequence, in order to impact the least on the existing system, the design enforces the separation of the CES and VMV software, which is achieved simply by interfacing VMV via the relational database, which holds all of the plaintext votes, and by providing a separate user interface for vote verification and election auditing[20]. By keeping the votes in plaintext within the CES system, and then adding verifiability to the voter record and their plaintext vote, the impact on the system is minimised because neither the voter user interface or processing need to change, while the desired verifiability can be added. Nonetheless, this compromise means that the existing lack of end-to- end privacy of votes within the CES system continues at this stage. The separated VMV software architecture is shown in Fig. 2, which shows the relationship between the additional components and the CES system[21]. The components in the architecture are:Voting Web Service The existing CES e-voting system which operates without change except to provide additional infor- mation to voters to allow them to verify their vote. Vote Database The existing CES relational database holding all details about an election, voters and their plaintext vote (once a ballot has been cast). This is modified to add in the verifiability data per voter and is used as the input and output interface for VMV through the import and export of comma-separated values (CSV) data files. CES Network The secure network within which the Voting Web Service and Vote Database are held. Public access is only granted to the Voting Web Service within this network via HTTPS (and to vote only with credentials)[22]. Since the Selene Layer accesses voter and vote data, it is also run within the CES Network to ensure that all private data is kept securely within the network. Selene Layer Executes the Selene protocol by taking data from the Vote Database as CSV files, communicating with the Verificatum Nodes to perform shuffling and decryption, and with the Verification Web Service to publish verification data, including produced CSV and NIZKPoK proof files. These operations are initiated by an administrator using a computer running within the CES Network. Verificatum A series of independently-operated nodes running the Verificatum software. Two or more independent organ- isations can run a Verificatum Node which is initialised by the Selene Layer. Each Verificatum Node can communicate with each other node within the Mix-net Network[22]. Prior to a mix-net operation, such as shuffling, each node is supplied with identical CSV input and produces identical CSV output together with the corresponding proof . Mix-net Network Each Verificatum Node is run within its own secure network hosted by each independent organisation. Access to each Verificatum Node is only granted to the other Verificatum Nodes and the Selene Layer, which controls the Verificatum operations. Verification Web Service A web service with a user interface which allows administrators to publish verification data, auditors to view the published election data and voters to verify their vote[23]. This forms the public face of the VMV demonstra- tor and allows published files to be served to users. Publication requires privileged access granted to administrators via user accounts. Only administrators have accounts, while anyone can view published data. Verification Database Holds the data necessary to run the Ver- ification Web Service, including administrator user accounts and an index of each election's verification data. This includes the list of the CSV and proof files held in the Data Lake, and their corresponding contract addresses in the Quorum cluster, such that they can be retrieved via the Verification Web Service[24]. Data Lake Holds the published CSV and NIZKPoK proof files in a repository which is only accessed via the Verification Web Service. Verification Network A secure network in which the Verification Web Service and Data Lake operate. Public access is only granted to the Verification Web Service within this network via HTTPS. Quorum Node A series of independently-operated nodes run- ning the Quorum software, a particular Distributed Ledger Technology [9]. Two or more independent organisations can each run one or more Quorum Nodes. Each Quorum Node can communicate with each other node within the DLT Network. When a file is published via the Verification Web Service, it is saved to the Data Lake and a hash of the file is committed to the Quorum cluster. Periodically, the hash is verified against the file held in the Data Lake to ensure its integrity. All data held within the Vote Database, Verification Database, Data Lake, Verificatum Nodes and Quorum Nodes should be held resiliently such that they are backed up to prevent data loss. For example, each Verificatum Node holds privately within its file system its share of the election private key sk_{T_i} Verificatum was chosen as the preferred mix-net implementation because it is open source, has worked successfully in a number of large-scale elections, has a proven cryptographic protocol, works well with ElGamal encryption and produces the desired NIZKPoK for each operation. The Verificatum Nodes within the mix-net require a threshold number of the nodes to perform cryptographic operations. The number of nodes within the mix-net and the number required for a threshold is configurable when the mix-net is created. For example, four Verificatum Nodes can be run such that a threshold of three of them is needed to operate[25]. This allows for one (and only one) node to be removed from the mix-net for it to still be able to operate. If less than the threshold number of Verificatum Nodes is available, then no mix-net operations can take place. Nodes may be removed from the mix-net through failure or if the operator of the node is thought to be compromised or malicious. Similarly for the Quorum cluster. Quorum was chosen as it is open source (based upon Ethereum [19]) and being developed as an enterprise-ready DLT solution by J.P. Morgan [9], ensuring that it is sufficiently robust for commercial operation. The sequence of interactions between the components is shown in Fig. 3, and the following provides detail on each stage of operation, describing the various design choices associated with each component, while relating to the Selene protocol in Section II.

C.Setup

Once the election has been defined by CES, and the number of voters quantified, then the Selene Layer can be used to set up the election verifiability parameters. The first stage is to initialise the cryptographic parameters (Selene: Create Params) which will be used to create the corresponding encryption and signing keys (Section II-A). For example, this includes the cyclic group G with order p, and the associated generator g. Once this has been done, the election key pair can be created. Recall that in Selene, the election encryption key pair is created by the mix-net so that shares of the private key sk_{T_i} are distributed across the mix-net nodes i, while the public key pk_T is available for third-parties to encrypt data. In Verificatum, each node in the mix-net is initialised with data relevant to the election, including the cryptographic parameters, election name, and the IP address of each Verificatum Node in the mix-net (Selene: Create Mix-net). For convenience, a copy of the Selene Layer software is used on each Verificatum Node to run the Verificatum commands necessary to initialise the local files (Mix-net: Create)[26]. When this has been completed, the Selene Layer is used (Selene: Election Key) to create the election key pair (Mix-net: Create Key). During key pair creation, each Verificatum node communicates with each other node to form a consensus on the key and ensure that each has a share of the private key (with each share remaining private to the node). We now turn to our first challenge in the Selene proto- col (Section II-F1) where Selene assumes that all voters have their own encryption and signing keys initialised from suitable cryptographic parameters[27]. In VMV voters do not have their own keys, even though the keys are assumed to be available to complete the Selene protocol. In an ideal scenario, each voter would have their own keys either allocated to them via organisational or national infrastructure, or they would be able to generate and store them securely themselves. This is not the case for the CES system, and hence during the setup of the election, the encryption (sk_i, pk_i) and signing keys for all of the voters are generated for them (Selene: Voters Keys) by taking from the CES system the number of voters in the election (CES: Number of Voters). This is an unavoidable compromise in the protocol which means that the Selene Layer holds all of the keys for the voters and is therefore a trusted party. However, while the Selene Layer holds both the private and public keys for the voters, the private keys are not shared with the CES system so that it is not possible for CES staff to encrypt and sign votes only the Selene Layer can do this. Once all of the keys have been generated, the next stage is to create the trackers (Selene: Create Trackers). Random trackers are created (enough for every voter), which are then mapped to a number in the cyclic group G and encrypted using the election public key pk_T (Section II-A). These encrypted values are then shuffled (Selene: Shuffle Trackers) using the mix-net (Mix- net: Shuffle) so that there is no correspondence between the input and output encrypted trackers from the mix-net. A longside the trackers, the first part of their commitments, β, (Selene: Create β) are then generated (Section II-B) by first re- questing that every Verificatum Node generates a random value r for each voter (Mix-net: Randomise)[28]. This random number generation is not part of Verificatum and is instead completed by the controlling Selene Layer on each node. These random values are then combined in encrypted form and transformed to be one half of the ciphertext of the tracker encrypted under each voter's public key pk_i, as described in Equation 2. The value g^r is also held privately by the node. This process also involves the decryption of data by the mix-net but for simplicity, we omit the details here as the process is fully described in [29].The last stage of the election setup is to allocate each voter their encryption and signature key pairs, an encrypted tracker and the corresponding β. CES is then provided with this data (Selene: Save Data) to store in the Vote Database (CES:

Keys and β) against each voter (without the private keys). All of the public data is then sent to the Verification Web Service for publication (Verification: Publish). During publication, the Verification Web Service saves each of the supplied CSV and proof files to the Data Lake (Data Lake: Add Files) then calculates a SHA-256 hash of each file and then commits this hash to the Quorum cluster (Quorum: Add Hashes)[30]. Once committed, the files are made available for public viewing by the Verification Web Service (Verification: Verify Election and Vote Status), which allows files to be retrieved (Data Lake: Hold and View Files) and periodically checks the correspond- ing hashes against the contract in Quorum (Quorum: Verify Hashes). Our choice of using a Data Lake is motivated by the size of the files that are generated. With, for example, keys with 3027 bits, a voter record (3) consists of approximately 4000 bytes, so that with 1000 voters, the corresponding data file is just under 4 MB and 100,000 voters 400 MB. While Quorum is designed to immutably store data, the larger the amount of data that needs storing, the longer the required consensus protocols take to run across the Quorum Nodes. Consequently, a Data Lake is used to hold the files, which can then be of arbitrary size, while the Quorum cluster only holds a hash of each file. If any file is changed, its hash will therefore not match to that which is stored in Quorum. While this is not ideal since it requires the hash to be checked when retrieving any file to ensure its integrity, it is more practical without compromising on the immutability of the files and allowing independent verification of the hashes. Data is committed to the Quorum cluster using the notion of a ‘contract’. A contract is similar in concept to an object in an object-oriented programming language. Each contract template is written using the Solidity [31] language, encapsulating data and methods which operate on the data. To commit data to the Quorum cluster, the compiled contract is loaded and a new instance created with the required data and/or methods executed. The contract is then committed to all nodes in the cluster through a consensus protocol so that it is written to the blockchain. Once committed, the address of the contract is returned, and this can be used to retrieve the contact and its data. Once all of the data for this stage has been published, anyone can view the public data for the election via the Verification Web Service (Verification: Verify Election and Vote Status). As part of the email sent out by CES to all voters with their security credentials, each voter is also sent their β value. This can be used by the voter to verify that their β exists within the verification data (their ‘Vote Status’). This therefore enables any interested third-party to verify, for example, the number of voters, that every voter has a unique β, and to also independently verify all of the NIZKPoK.

D. Voting

Once the setup is complete, CES may open the election for voting. Here, the CES system remains unchanged in that voters use their security credentials to login to the Voting Web Service (via HTTPS) and submit their vote (CES: Voting). Each vote is recorded in plaintext within the Vote Database. CES also allow voters to cancel their votes via telephone, and then to re-vote. Here we face two challenges presented by Selene which assumes the end-to-end encryption of votes (Section II-C). First, votes are not end-to-end encrypted since voters submit their vote in plaintext which is then recorded in the database (Section II-F2). As discussed, this means that CES are trusted to maintain the privacy of votes, but this was required to allow the CES system to remain intact and recoverable in the event of a failure in VMV. However it would be a straightforward adaptation of the system to receive and manage only encrypted votes. Second, votes are not recorded in real-time to the Quorum cluster (Section II-F3).

E. Verification

Once the election period has ended and voting closed, the final set of verification data may be generated and the tally performed. First, all of the plaintext votes are exported from the Vote Database (CES: Plaintext Votes). This is only done within the CES Network so that the plaintext votes are never compromised. Alongside the plaintext votes, the public keys and encrypted trackers stored within the Vote Database for each voter are also exported within the CSV file. This enables the Selene Layer to find the corresponding private signature key for each voter. In order to encrypt all of the plaintext votes, we must now overcome the limitation of ElGamal which can only be used to encrypt numbers within the cyclic group G (Section II-F4). This can either be overcome by first supplying a complete list of all possible plaintext votes, or by finding all distinct votes which have been cast. The former is possible where production of the list of distinct plaintext votes is tractable, such as for yes/no votes or similar, but which becomes too time-consuming in preferential voting with lots of candidates. The latter can then be used to find all distinct votes from those that have been cast, which at the worst case will be as many as there are ballots cast. Once a list of distinct votes has been generated, each can be mapped to a unique number in the cyclic group for encryption. For example, for every vote option, a unique random number v_i is generated which can be mapped into a number within the cyclic group G with generator g to yield

$V_i = g^{v_i} \mod p$

where p is the prime order of the group. With all of the votes mapped, the Selene Layer is then used to encrypt and sign the votes for each voter (Selene: Encrypt and Sign Votes) to produce the completed vote record (3). This includes the proof of correct encryption. Votes are signed using the Digital Signature Algorithm (DSA) [32]. The encrypted tracker and encrypted vote tuples for each voter are then extracted for mixing (Section II-D). Each Verificatum Node receives the list of tuples (Selene: Mix Votes) and is in- structed to mix them (Mix-net: Mix). This performs a shuffle of the tuples before decryption. The result is a list of tuples holding the plaintext tracker in the cyclic group and the corresponding plaintext vote. The trackers in the cyclic group are then mapped back to the trackers (5). The final stage of the Selene protocol is to enable voters to calculate their tracker by supplying them each with their ran dom values $g^{r_{i,j}}$ held by the Verificatum Nodes. Unfortunately, providing each independent Verificatum Node with a means to form a secure channel to every voter is not practical, as this would require some form of identifying information to be given to each node (such as an email address or credentials for a voter to access the node securely). Here then we address this challenge (Section II-F5) by focusing instead on the end result, namely providing each voter with their α_i commitment. In Selene, the concept is that each voter must generate their α_i from the random values provided to them. In some way this allows the voter to avoid coercion supposedly because they cannot be forced to reveal their true α_i, and can instead generate an alternative α^t which points to a different vote record. To achieve this, the values must be sent securely, and the voter must have the necessary software needed to calculate their α_i or an alternative. Both of these impose constraints on the voter which detract from their experience and make verification more difficult. Our approach is to assume that the voter can receive their α_i directly without having access to their random values. This does increase the risk of coercion, but prevents the distribution of personal data to the Verificatum Nodes[33]. Here then instead, each Verificatum Node shares the random values with the Selene Layer running within the CES Network (Mix-net: Randoms), which then calculates the α values (Selene: Create α). The α values can then be distributed by CES to the voters without revealing any personal data to VMV. The encrypted vote, corresponding signature and α are pro- vided to CES (Selene: Save Data) to store in the Vote Database (CES: Encrypted Votes and α), and so that the α can be provided to each voter once the tally has been completed (CES: Tally). This is achieved by sending them an email with their α and β embedded within it, together with instructions on how to verify their vote. Here also, all of the public data is then sent to the Verifica- tion Web Service for publication (Verification: Publish). Once committed, the files are made available for public viewing by the Verification Web Service (Verification: Verify Election and Vote Status/Vote).Software Environment The implementation of the VMV demon- strator consists of five software environments:CES The CES software runs intact on Microsoft Windows servers in their data centres.The interface with VMV is through the import and export of CSV files from the Vote Database. Selene and Verificatum For portability, the Selene Layer was built in Java, using the Bouncy Castle cryptographic API . Verificatum is also built using Java . By using Java, both pieces of software can be run on Windows or Linux servers providing flexibility for deployment, while Java also provides strong support for cryptography. Quorum Quorum is written in Go and can be deployed to Linux servers. Verification The Verification Web Service was built using Ruby- on-Rails to support rapid application development of a web service which requires both a user interface and an underlying infrastructure that supports the Data Lake and access to Quorum. A web service can therefore be accessed by any internet-enabled computer and a suitable web browser, and promotes the use of best-practice guidelines for the de- velopment of user interfaces to promote a good voter experi- ence. The web service can be run on Linux servers with the Data

Lake provided by suitable resilient storage (for example Amazon Web Services S3 [4]). The source code for the VMV software [10] is open source under the MIT licence, and is available at https://github.com/saschneider/VMV. Version 1.0, used in the system described in this paper, is archived at 10.5281/zenodo.3695909.

VI. SYSTEM IMPLEMENTATION

The implementation of the project involves a combination of multiple technologies and methodologies that work together to create a transparent, secure, and efficient voting environment. Below is a breakdown of the system implementation:

1. Front-End Development
The front-end of the system is built using HTML, CSS, JavaScript, and React.js to deliver a responsive and user-friendly interface. Bootstrap are integrated to ensure seamless user interactions. The front-end design is optimized for accessibility across various devices, allowing citizens and election officials to easily navigate the platform.

2. Back-End Development
The backend of the system leverages Python (Flask) to handle complex data processing, manage user requests, and facilitate blockchain integration. MySQL serves as the database to securely store critical election-related information such as voter details, vote logs, and candidate data.

3. User Authentication and Access Control
To ensure the integrity and security of the voting process, the system utilizes Aadhar-based QR code verification and facial recognition to authenticate voters. For election officials, multi-factor authentication is used to ensure secure access.

4. Blockchain Integration for Secure Voting
Blockchain technology, specifically Hyperledger fabric, is at the core of this system, ensuring that votes are securely stored as transactions on an immutable ledger. Smart contracts verify the authenticity of votes, prevent tampering, and facilitate automatic vote tallying. The decentralized nature of the blockchain ensures that no single entity can alter or manipulate election results.

5. Voting Process and Ballot Management
The voting process begins with voter authentication, where users are verified using Aadhaar details and facial recognition. Once authenticated, voters can select their preferred candidates from an electronic ballot, cast their vote, and the vote is securely encrypted and recorded on the blockchain. This process guarantees confidentiality and integrity of votes.

6. Result Computation and Transparency
Election results are computed in real-time and stored on the blockchain for transparent, tamper-proof result tallying. As votes are cast, they are automatically aggregated, and results are displayed on the admin dashboard. Blockchain allows for real-time audits, offering transparency to stakeholders and eliminating concerns over electoral fraud.

7. Notification System
The Notification System ensures that voters and election officials are kept informed throughout the election cycle. Voters receive notifications about election dates, voting status, and result announcements via email or SMS. All notifications are tracked and logged to ensure that users receive timely
updates throughout the process.

8. Security Measures
Security is a core focus of the system, utilizing end-to-end encryption to protect the integrity and confidentiality of voter data and votes. Hashing algorithms ensure that voter identities and vote data remain secure. Additionally, distributed ledger technology ensures there is no centralized control, reducing the risk of vote tampering.

9. Testing and Validation
Rigorous system testing is conducted to ensure the platform's functionality, security, and performance. This includes unit testing, penetration testing, and blockchain validation. User feedback during testing helps to optimize the system, address potential vulnerabilities, and improve the user experience.

10. Deployment
Once the system is fully developed and tested, it is deployed to a cloud-based server to ensure scalability and high availability during elections. Post-

deployment, security audits and regular updates are performed to maintain the system's compliance with election laws and regulations.

VII. RESULT AND DISCUSSION

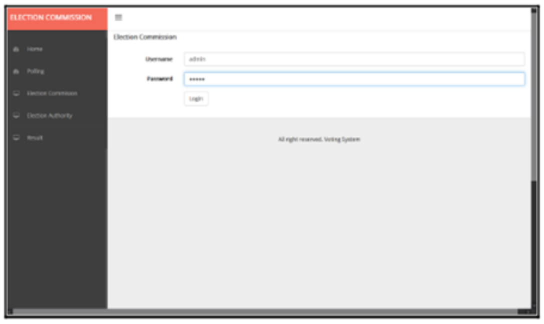


Fig 7.1 Home page

The home page of the voting system where the Election Commission can log in using a user name and password. It serves as the main access point for authorized users to manage election-related operations securely.

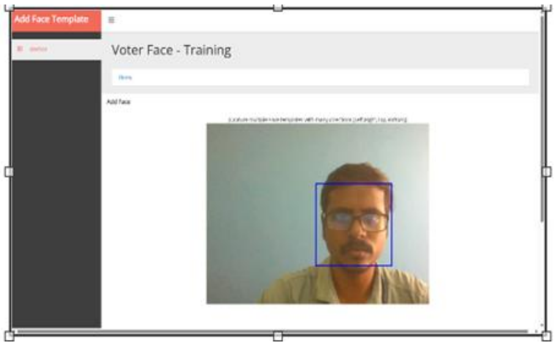


Fig 7.2 Training Module

The facial recognition training module for newly registered voters. The system captures facial features using an image processing algorithm and generates a unique face template. It uses bounding box detection to identify and extract the voter's facial region for training. The trained facial data is encoded and stored in the system for future identification and verification. This process enhances security and automation in the voter authentication workflow.

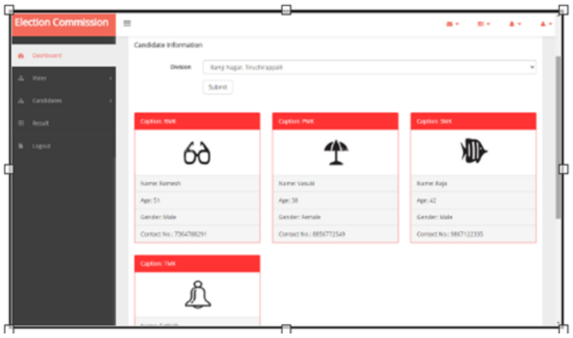


Fig 7.3 Candidate information page

The Candidate Information page of the Election Commission system. It presents the details of registered candidates, including their name, age, gender, and contact number. Each candidate is represented with a unique election symbol, and the interface provides options for viewing or submitting candidate data. The layout follows a structured, user-friendly design suitable for digital voting or candidate management applications.

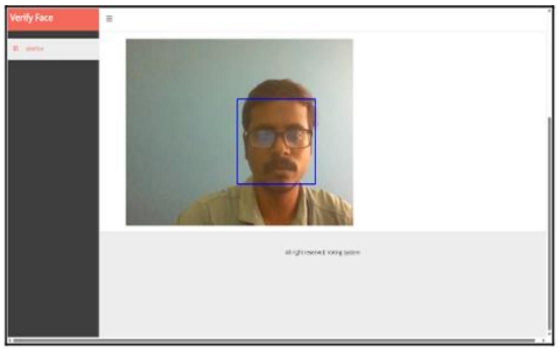
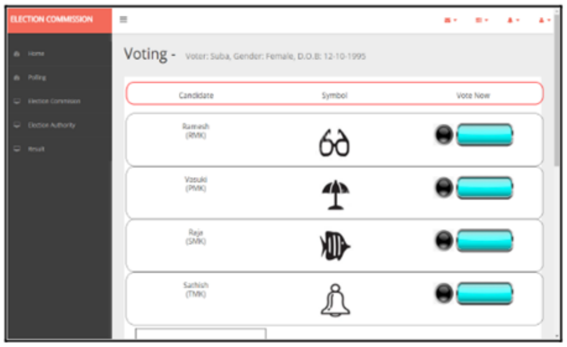


Fig 7.4 Face verification module

The Face Verification module of the Election Commission Voting System. The interface captures the user's live image through a connected camera and identifies facial regions using a bounding box for recognition and verification. This process ensures biometric authentication of the voter before granting access to the voting portal. The system utilizes computer vision and AI-based face detection algorithms to compare live facial data with stored voter records, enhancing security, transparency, and prevention of impersonation in electronic voting. The module forms a key component of the digital identity verification framework integrated within the election system.

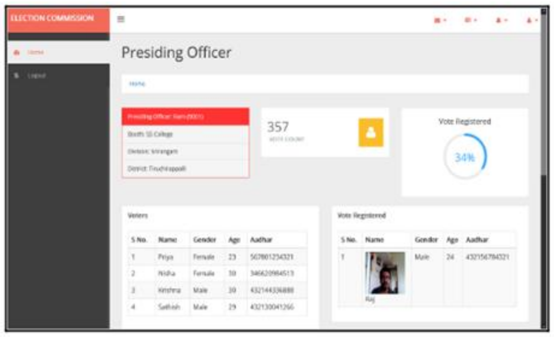


Voting - Voter: Suba, Gender: Female, D.O.B: 12-10-1995

Candidate	Symbol	Vote Now
Ramesh (RMK)		
Vasuki (PMK)		
Raja (SMK)		
Sathish (TMK)		

Fig 7.5 Voting interface

The voting interface of an Electronic Voting System (EVS) designed for secure and user-friendly election processes. The interface displays a structured layout under the "Election Commission" module, showing the voter's details (name, gender, date of birth) along with a list of candidates participating in the election.Each candidate entry includes the candidate name, party abbreviation, and corresponding party symbol for easy recognition. On the right side, a "Vote Now" button (or indicator) allows the voter to cast their vote electronically for the chosen candidate.



357

Voters

S.No.	Name	Gender	Age	Aadhar
1	Priya	Female	23	567801234321
2	Nisha	Female	30	346620984513
3	Krishna	Male	30	432144336888
4	Sathish	Male	29	432130041266

Vote Registered

S.No.	Name	Gender	Age	Aadhar
1	Raj	Male	24	432156784321

Fig 7.6 Presiding officer's dashboard

The Presiding Officer's dashboard used to monitor election activities.It displays voter details, booth and district information, and total vote count.The interface tracks registered voters and shows live voting progress using visual charts.It helps ensure secure, transparent, and efficient election management. The Presiding Officer's dashboard used to monitor election activities.It displays voter details, booth and district information, and total vote count.The interface tracks registered voters and shows live voting progress using visual charts.It helps ensure secure, transparent, and efficient election management.

VIII. CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION

In conclusion, the project offers a secure, transparent, and efficient alternative to traditional voting systems. The system leverages blockchain technology to ensure the integrity of votes and prevent any tampering or fraud. By integrating Aadhar authentication for secure user verification and OTP for transaction validation, the system guarantees that only eligible voters participate in the election process. Additionally, the use of blockchain ensures that all votes are immutable, providing transparency and accountability in the voting process. The system is designed with several key modules, including the Voter Registration System, Vote Casting System, and Vote Verification System. Each module is carefully crafted to ensure smooth operation and secure functionality. The user-friendly interface and seamless integration with blockchain ensure a smooth experience for both voters and administrators. Through thorough testing, the system demonstrated robust performance under high traffic and successfully passed all functionality, security, and performance tests. The system's scalability allows it to handle elections of varying sizes, making it suitable for both small organizations and large-scale governmental elections.

FUTURE ENHANCEMENT

Future enhancements for the Blockchain-Based Online Voting System could focus on increasing accessibility, enhancing security, and broadening its functionality. Some potential improvements include:
Integration with Multi-Factor Authentication (MFA): Incorporating multi-factor authentication mechanisms such as one-time passwords (OTPs) or mobile app-based authentication will provide an extra layer of security to ensure voter identity verification.

Support for Global Elections: The system could be expanded to support international elections by incorporating different languages, electoral laws, and cultural contexts, thus broadening its potential usage for global elections.

Integration with Digital Identity Systems: Integrating with decentralized identity systems, such as Decentralized Identifiers (DIDs), can offer a more secure, user-controlled way to verify voter identities while respecting privacy and autonomy.

Integration with Election Commission Systems: The system can be integrated with national and local election commission systems to streamline election data management, reporting, and compliance with electoral regulations.

REFERENCES

[1] Secure Online Voting System-Based on Facial Recognition by Using Deep Learning , Krishna Prakash; Nimmagaddda Vatsalya Mitra; Nallamothu Pavan Kumar; Manda Anji Babu; Shonak Bansal; Sandeep Kumar 2025

[2] A Comprehensive Evaluation of Secured Electronic Voting System Design Based on Face Biometric Authentication Policy, Pandarinath Potluri; R. Jayakarthik; Shivam Agarwal; Shobana S; Venkata Padmavathi S; Aarthi R 2024.

[3] Enhancing the Security of Online Voting System Using Defined Biometrics, *Devanshi Malik; Kritika Tripathi; Jyotsna 2023*

[4] Enhancing the Security of Online Voting System Using Defined Biometrics, Devanshi Malik; Kritika Tripathi; Jyotsna invented in 2023.

[5] M. Arnaud, V. Cortier, and C. Wiedling, "Analysis of an electronic board- room voting system," in Proc. 4th Int. Conf., Springer, 2013, pp. 109–126.

[6] J. Ben-Nun et al., "A new implementation of a dual (paper and crypto- graphic) voting system," in Proc. 5th Int. Conf. Electron. Voting, 2012,pp. 315–329.

[7] J. Benaloh, "Simple verifiable elections," in Proc. USENIX/ACCURATE Electron. Voting Technol. Workshop, 2006, pp. 5–5.

[8] J. Benaloh, R. L. Rivest, P. Y. A. Ryan, P. B. Stark, V. Teague, and P. L. Vora, "End-to-end verifiability," 2015, arXiv:1504.03778.

[9] Blockchain Solutions Group, "Quorum whitepaper," 2017. Accessed: Nov 14, 2021. [Online]. Available: https://www.blocksg.com/single-post/ 2017/12/27/Quorum-Whitepaper

[10] M. Casey, "VMV: Verify my vote software," 2020. Accessed: Nov. 8, 2023. doi: 10.5281/zenodo.3695909.

[11] P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo, "BeleniosRF: A non-interactive receipt-free electronic voting scheme," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2016, pp. 1614–1625.

[12] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," Commun. ACM, vol. 24, no. 2, pp. 84–90, 1981.

[13] D. Chaum et al., "Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes," in Proc. USENIX/ACCURATE Electron. Voting Workshop, 2008, Art. no. 13.

[14] D. Chaum, P. Y. A. Ryan, and S. A. Schneider, "A practical voter-verifiable election scheme," in Proc. 10th Eur. Symp. Res. Comput. Secur., 2005, pp. 118–139.

[15] N. Chondros et al., "D-DEMOS: A distributed, end-to-end verifiable, internet voting system," in Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst., 2016, pp. 711–720.

[16] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a secure voting system," in Proc. IEEE Symp. Secur. Privacy, 2008, pp. 354–368.

[17] C. Culnane and S. A. Schneider, "A peered bulletin board for robust use in verifiable voting systems," in Proc. IEEE 27th Comput. Secur. Found. Symp., 2014, pp. 169–183.

[18] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. Inf. Theory, vol. 31, no. 4, pp. 469–472, Jul. 1985. Ethereum, "Home | ethereum," 2019. Accessed: Nov. 14, 2021. [Online]. Available: https://ethereum.org/ Ethereum, "Solidity-Solidity 0.4.21 documentation," 2019. Accessed: Nov. 14,2021. https://github.com/ethereum/solidity NIST, "Digital signature standard (DSS)," 2023. Accessed: Nov. 8, 2023. [Online].Available: https://csrc.nist.gov/pubs/fips/186-5/final

[19] K. Gjøsteen, "The Norwegian internet voting protocol," in Proc. 3rd Int. Conf., Springer, 2011, pp. 1–18. K. Gjøsteen and A. S. Lund, "An

experiment on the security of the Norwegian electronic voting protocol," Ann. des Télécommunications, vol. 71, no. 7/8, pp. 299–307, 2016. Google, "The go programming language," 2019. Accessed: Nov. 14, 2021. [Online]. Available: https://golang.org/ GoQuorum, "Home | quorum," 2021. Accessed: Nov. 14, 2021. [Online]. Available: https://www.goquorum.com/

[20] D. H. Hansson, "Ruby on rails | a web-application framework that includes everything needed to create database-backed web applications according to the model-view-controller (MVC) pattern," 2019. Accessed: Nov. 14, 2021. [Online]. Available: https://rubyonrails.org/

[21] S. Heiberg, T. Martens, P. Vinkel, and J. Willemson, "Improving the verifiability of the Estonian internet voting scheme," in Proc. 1st Int. Joint Conf. Electron. Voting, Springer, 2016, pp. 92–107.

[22] S. Heiberg and J. Willemson, "Verifiable internet voting in Estonia," in Proc. 6th Int. Conf. Electron. Voting, 2014, pp. 1–8.

[23] J. Helbach and J. Schwenk, "Secure internet voting with code sheets," in Proc. 1st Int. Conf. E-Voting Identity, 2007, pp. 166–177. Information Commissioner's Office, "Guide to the general data pro- tection regulation-GOV.UK," 2018. Accessed: Nov. 14, 2021. [Online]. Available: https://www.gov.uk/government/publications/guide-to- the-general-data-protection-regulation V. Iovino, A. Rial, P. B. Rønne, and P. Y. A. Ryan, "Using Selene to verify your vote in JCJ," in Proc. Int. Workshops Financial Cryptogr. Data Secur., Springer, 2017, pp. 385–403.

[24] R. Joaquim, P. Ferreira, and C. Ribeiro, "EVIV: An end-to-end verifiable internet voting system," Comput. Secur., vol. 32, pp. 170–191, 2013.

[25] S. Khazaei and D. Wikström, "Return code schemes for electronic voting systems," in Proc. 2nd Int. Joint Conf. Electron. Voting, Springer, 2017, pp. 198–209.

[26] O. Kulyk, J. Henzel, K. Renaud, and M. Volkamer, "Comparing "challenge-based" and "code-based" internet voting verification implementations," in Proc. 17th IFIP TC 13 Int. Conf. Hum.-Comput. Interac- tion, Springer, 2019, pp. 519–538.

[27] R. Küsters, J. Müller, E. Scapin, and T. Truderung, "sElect: A lightweight verifiable remote voting system," in Proc. IEEE 29th Comput. Secur. Found. Symp., 2016, pp. 341–354.

[28] B. Laurie, A. Langley, and E. Käsper, "Certificate transparency," RFC, vol. 6962, pp. 1–27, 2013. "Legion of the Bouncy Castle Inc. The legion of the bouncy castle java cryptography APIs," 2019. Accessed Nov. 14, 2021. [Online]. Available: https://www.bouncycastle.org/ Oracle, "Java | Oracle," 2019. Accessed: Nov. 21, 2021. [Online]. Avail- able: https://www.java.com/

[29] P. Roenne, P. Y. Ryan, and M.-L. Zollinger, "Electryo, in-person voting with transparent voter verifiability and eligibility verifiability," in Proc. 3rd Int. Joint Conf. Electron. Voting E-Vote-ID: TUT Press Proc., 2018,pp. 147–164.

[30] P. Y. A. Ryan, P. B. Rønne, and V. Iovino, "Selene: Voting with transparent verifiability and coercion-mitigation," in Proc. Int. Workshops Financial Cryptogr. Data Secur., 2016, pp. 176–192.

[31] P. Y. A. Ryan and V. Teague, "Pretty good democracy," in Proc. 17th Int. Workshop Secur. Protoc., 2009, pp. 111–130.[41] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme," in Proc. Conf. Adv. Cryptol., 1995, pp. 393–403.[42] M. Sallal et al., "Augmenting an internet voting system with selene verifiability using permissioned distributed ledger," in Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst., 2020, pp. 1167–1168.

[32] M. Sallal et al., "VMV: Augmenting an internet voting system with selene verifiability," 2019, arXiv: 1912.00288.Swiss Post, E-voting: Online voting and elections, 2023. Accessed: Nov. 8, 2023.[Online]Available: https://digital-solutions.post.ch/en/e- government/digitization-solutions/evoting/publications-and-source-code The Political and Constitutional Reform Committee, "Voter engagement in the UK: Follow up - political and constitutional reform," 2015. Accessed:
Nov.14,2021.[Online].Available:https://publicati

ons.parliament.uk/pa/cm201415/cmselect/cmpolcon/938/93802.htm

[33] D. Wikström, "Verificatum," 2021. Accessed: Nov. 14, 2021. [Online]. Available: https://www.verificatum.com/