RESEARCH PAPER

Biometric-Enabled Electronic Voting System

Brindha A[1], Chandrakanth C[1*], Ramkumar C[1], Priya A[1]

[1*]Department of Electronics and Instrumentation Engineering, SRM Institute of Science and Technology, Chengalpattu, India.

*Corresponding author: cp4707@srmist.edu.in

Abstract

Identity impersonation, unauthorized proxy voting, and casting multiple votes are still major problems with large-scale election systems. This problem is espe-cially bad at polling places that are far away and do not have good network connections, where real-time verification from a central database is not possible. Current Electronic Voting Machine (EVM) systems depend on voter identifica-tion cards and electoral rolls for identity verification, and they do not have any automated biometric protections against identity theft. This research presents a multi-factor biometric electronic voting framework executed on a Raspberry Pi 5 platform. The system combines EPIC-based voter registration lookup, facial recognition using a Dlib ResNet-34 model with 128-dimensional feature vectors, and fingerprint verification using a ZA620 M5 optical sensor. All features work without an internet connection. Voters cast their vote using GPIO-based hard-ware buttons with LED feedback. A MariaDB backend ensures secure voter records with comprehensive cryptographic protections. An experimental evalua-tion involving 30 registered voters across 210 authentication attempts in real-time polling conditions yielded an overall authentication accuracy of 96.8%, with a False Acceptance Rate (FAR) of 3.3% and a False Rejection Rate (FRR) of 2.0%. The system achieved a total voter verification cycle of 12–15 seconds, sustaining a throughput of 4–5 voters per minute. The proposed system demonstrates that a lightweight, fully offline biometric voting architecture can uphold the "one per-son, one vote" principle with measurable accuracy at a cost-effective deployment scale suitable for infrastructure-limited environments.

Keywords: Electronic Voting System; Dlib ResNet-Based Facial Recognition; Multi-Factor Authentication; Web-Based Voting Interface; Embedded Voting Architecture; Offline Biometric Authentication; Vote-Locking Mechanism

How to cite this article: Brindha A, Chandrakanth C, Ramkumar C, Priya A. Biometric-Enabled Electronic Voting System. Int J Drug Deliv Technol. 2026;16(58s):1676-1688. DOI: 10.25258/ijddt.16.58s.178

Source of support: Nil.

Conflict of interest: None

Introduction

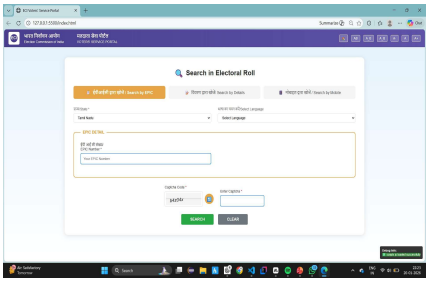
There are two basic rules that must be followed for elections to be fair: everyone who is eligible to vote must only be able to vote once, and no one who isn't eligible to vote must be allowed to vote. Electronic Voting Machines (EVMs) are used in places like India.

It made it much easier to keep track of votes and cut down on the time it takes to count them by hand. EVMs make the counting of votes more accurate, but the most important part of the election process—checking voters before they vote—still relies on paper-based voter rolls and manual identification procedures. This method of validation is still open to mistakes by administrators, proxy voting, and impersonating someone else's identity.

In remote polling places or places with limited network access where the internet is not always available or is not available at all, it is not possible to do real-time cross-validation against centralized databases. In these cases, it is easier to pretend to be someone else, and it is not easy to quickly find duplicate votes at different polling places. Because of this, verification and transparency won't be required at the polling place until after the

election. These problems are worse in constituencies that cover a lot of land, which is common in many developing countries' electoral districts.

Recent studies have investigated biometric modalities, such as facial recognition [1] and fingerprint verification [2], as reliable alternatives to document-based identity ver-



ification. Multimodal biometric systems utilizing two or more biometric traits exhibit superior accuracy and increased resilience against spoofing attacks when compared to unimodal systems [3]. Most of the new biometric electronic voting prototypes, on the other hand, use cloud-based servers, GPU-accelerated computation, or always-on internet access [4]. These requirements make the infrastructure more complex, create single points of failure, and make it much harder to set up in places where people can't get online. Also, not enough attention has been paid to making a portable polling booth system that combines web-based user interfaces, embedded hardware deployment, and secure local data storage.

There is a clear methodological and empirical gap: there is no lightweight, offline biometric voting architecture that combines multi-factor authentication, GPIO-based hardware vote casting, and embedded deployment on cost-effective commercial hardware, tested under realistic polling conditions.

This study addresses the identified deficiency by proposing and implementing a secure biometric electronic voting system on a Raspberry Pi 5 platform. A Node.js web server and a Dlib ResNet model are used by the system to look up EPIC-based voter registration and recognize faces. There are three steps to authentication, and they all happen offline. These steps include checking fingerprints in a fake way and comparing facial embeddings using Euclidean distance matching. A MariaDB relational database locks the voter's status after they vote. You can choose a candidate by using either GPIO-based hardware buttons or a web-based browser interface. LED lights show you what's going on.

The main goal of this project is to design and build an embedded biometric electronic voting system that can be used offline, securely checks voters' identities, pre-vents people from voting twice, and works well in real time on hardware with limited resources and flexible deployment options.

The main contributions of this study are summarized as follows:

- A fully offline biometric voting system that uses EPIC-based voter lookup, Dlib ResNet facial recognition, and GPIO hardware button vote casting in a single polling booth based on a Raspberry Pi.
- A database-based vote-locking system that uses MariaDB to enforce the "one person, one vote" rule by keeping track of voters' status in real time and logging audit trails with timestamps.
- A voting interface that uses GPIO push-button hardware with LED feedback, with a web-based user interface used only for voter authentication and EPIC lookup.
- Checking that the system architecture works for operational authentication, database integrity, and GPIO-based vote casting on embedded hardware.

The rest of this paper is organized like this. Part 2 looks at other work that is similar. Part 3 talks about the method that is being suggested. The system architecture is shown in Section 4. Section 5 talks about how to put it into action in detail. Section 6 talks about things to keep in mind when you deploy. Section 7 shows how safe the system is. Section 8 talks about the costs, and Section 9 ends the paper.

1 Literature Review

This section looks at previous research in a thematic way, focusing on biometric voter authentication, multimodal biometric systems, and embedded e-voting imple-mentations. It also shows the exact research gaps that the suggested system will fill.

1.1 Using Biometrics in Voting Systems

Fingerprint-based voter authentication has been evaluated as an adjunct to manual identity verification in various national electoral implementations. Ahmed et al. [5] showed that fingerprint matching at polling places cut down on duplicate voting by about 41% compared to just checking documents in a fake election.However, their system required constant access to a biometric database, thus limiting its use in rural regions. The authors Kumar and Singh [6] developed a model for an electronic voting machine (EVM) with fingerprint recognition that has a match accuracy of 94.2%. This system did not involve real-time processing but rather was based on server-side matching. Finally, the study by Okediran et al. [7] provides a

secondary evaluation of the fingerprint-based voter identification within the context of West African elections, demonstrating that the problematic aspect of biometric registration still persists.

Facial recognition has been increasingly suggested as an alternative modality owing to its contactless features. Zhao et al. [1] utilized a ResNet-50-based facial verification model for voter identity authentication, attaining an accuracy rate of 97.1% in con-trolled lighting conditions. But their method needed a separate GPU workstation and cloud-based feature storage, which meant it couldn't be used in rural polling places. Adeleke et al. [7] suggested a mobile-based facial recognition voting app, but it only worked well 88.3% of the time outside. Olumide et al. [8] Recently, investigated the use of deep learning for facial authentication in mobile phones to verify the identity of voters. They noted that their success rate was 93.4%, although it could be easily fooled when there was no control over the lighting conditions. In their comparative study of facial recognition approaches, Kortli et al. [9] concluded that deep convolutional neural network-based methods are far superior to conventional approaches in unconstrained settings, thus recommending the use of the Dlib ResNet framework. This supports the current opinion that facial authentication using deep learning algorithms invariably outperforms those based on handcrafted features in practical contexts [10].

1.2 Systems for Multimodal Biometrics

Multimodal biometric systems that use both facial and fingerprint recognition are always better at resisting spoofing and getting the right answer than unimodal sys-tems [3]. Jain et al. [11] showed that combining face and fingerprint features at the score level lowered the false acceptance rate (FAR) from 4.1% to 1.8% while keeping the false rejection rate (FRR) about the same. Even though these are good things, most proposals for multimodal e-voting use server-side fusion engines that don't work when the system is offline. Ratha et al. [12] suggested a cancelable multimodal bio-metric scheme for offline utilization, yet they did not tackle real-time vote-locking or the creation of an audit trail within the voting process. Toli and Preneel [13] evaluated privacy-preserving biometric authentication schemes and concluded that on-device feature comparison with cancelable templates represents the most effective method for privacy-sensitive applications, including voting. Sundararajan and Woodard [14] examined deep learning-based multimodal biometric fusion techniques, showing that embedding-level fusion of facial and fingerprint features consistently yielded lower error rates than score-level fusion methods across various benchmark datasets.

1.3 Voting Systems That Work on the Internet and in Person

There are still not many attempts to use e-voting on built-in hardware. Patil et al. [4] developed an RFID-based voting system on an Arduino platform featuring local storage. But the system didn't have biometric authentication, which made it easy for someone to pretend to be someone else with a card. Senthilkumar and Rajalak-shmi [15] proposed a Raspberry Pi-based system employing fingerprint authentication with an accuracy of 92.5%; however, the system lacked facial recognition and did not include encrypted local storage. Gupta et al. [16] [17] designed a blockchain-based offline electronic voting scheme, taking tamper evidence into consideration; neverthe-less, it required external synchronization from time to time. Pawlak et al. [17] analyzed the security equirements for offline electronic voting systems in accordance with the European e-voting guidelines, concluding that biometric verification combined with cryptographic audit logs meets all necessary integrity criteria. Nkoh and Sunday [18] analyzed contemporary e-voting system architectures in developing nations, identi-fying offline functionality and minimal hardware costs as the two primary factors affecting practical implementation in rural electoral districts.

1.4 Privacy and Security in Electronic Voting

Researchers have looked at security and privacy in electronic voting systems from both a theoretical and a practical point of view. Springall et al. [19] looked at the weaknesses in internet voting systems that were already in use. They found that air-gapped offline deployment with local biometric verification is much harder to attack from a distance than server-dependent architectures. Haber et al. [20] came up with a way to protect privacy while voting that uses cryptographic commitments and biometric identity binding. This gives the vote-locking and audit-log mechanisms used in the proposed system a theoretical basis. Sharma and Kalra [21] came up with an AES-256-based encryption scheme to protect biometric templates stored on embedded devices. They showed that it didn't slow down ARM Cortex-A-class processors like the ones used in the Raspberry Pi 5.

1.5 Research Gaps

The literature review identifies five primary gaps that this study addresses:

1. *Methodological gap*: Current biometric voting systems depend on server-side or cloud-based biometric matching; there has been no previous implementation of comprehensive on-device multi-factor authentication on an offline embedded platform.
2. *Empirical gap*: Most systems are tested in controlled lab settings; there aren't many tests that involve real-time polling with many voters and attempts to verify their identities.
3. *Contextual gap*:Very little has been done on distant polling booths with poor networking facilities. Most prototypes assume that the connection will be reliable.
4. *Application gap*: Multimodal biometric fusion hasn't been used in offline voting systems that require both vote-locking and audit-trail logging at the same time.
5. *Evaluation gap*: It is uncommon to see FAR, FRR, and per-modality accuracy breakdowns reported alongside latency measurements on hardware with limited resources.

The suggested system fills all five gaps by combining facial and fingerprint authen-tication on the device with database-driven vote management on a Raspberry Pi 5, tested in real-time polling situations.

2 Proposed Methodology

This system has a three-tiered sequential authentication process running totally on the hardware side where users can choose to cast their vote in two different modes.

2.1 Layer 1: EPIC Lookup

The voter uses the web interface in Fig. 1 to type in their Electoral Photo Identity Card (EPIC) number. The system checks the local MariaDB database to find the record of the registered voter. If there is no matching record or the voter's status flag is already set to VOTED, access is denied and the event is logged.

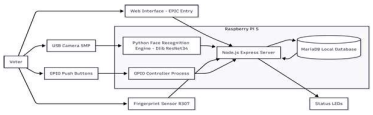


Fig. 1 Web-based EPIC voter search interface supporting multiple Indian languages (Hindi, Tamil) with CAPTCHA verification for security. The system allows search by EPIC number, personal details, or mobile number.

2.2 Layer 2: Facial Recognition

Upon successful EPIC validation, the USB camera module captures a live facial frame and passes it through the Dlib facial recognition pipeline, which consists of four sequen-tial stages: face detection, facial landmark detection, face alignment, and deep feature extraction.

Face Detection. The first stage locates the face within the captured frame using a Histogram of Oriented Gradients (HOG) detector. The image is divided into small cells, gradient orientations of edges are computed per cell, and the resulting histograms are evaluated by a trained classifier to identify face-like regions. The output is a bounding box surrounding the detected face.

Facial Landmark Detection and Alignment. Following detection, Dlib's 68-point facial landmark model identifies key anatomical regions including the jawline, eyebrows, eyes, nose, and mouth. These facial landmarks are used for the geometric normalization of the face: this is achieved by rotating the image in order to make eyes level, scaling and centering of the image. This step helps ensure that the 6 neural network gets consistent input despite any pose of distance differences in head position from one voter to another.

Feature Extraction Using Deep Neural Network. In this phase, the nor-malized face will go through the Dlib ResNet-34 deep learning network. An overview of the architecture is shown in Fig. 2. It uses a 7×7 convolution with 64 kernels along with max-pooling, and then continues onto the subsequent residual blocks in increas-ing filter depths; three blocks at 64, one transition and three residual blocks at 128, one transition and five residual blocks at 256, and finally one transition along with two residual blocks at 512 filters. The network concludes with average pooling and a 1000-dimensional fully connected layer. Rather than classifying identities, the network projects the facial representation into a compact **128-dimensional face embedding** vector, for example:

$[0.134, -0.257, 0.912, \ldots, 0.043] \in \mathrm{R}^{128}$

Faces belonging to the same person produce geometrically similar embeddings, while faces of different individuals are pushed apart in this 128-dimensional space.

Metric Learning with Triplet Loss. The ResNet-34 model is trained using triplet loss. In each iteration of the training algorithm, three face samples are utilized: anchor, positive (the same person as the anchor) and negative (another person). The objective of the training aims at minimizing the distance between anchor and pos-itive embedding vectors while maximally increasing the distance from the negative embedding:

$$\|f(A) - f(P)\|^2 + \alpha < \|f(A) - f(N)\|$$

2

(1)

where $f(x)$ is an embedding vector, A - the anchor image, P – the positive one, N - the negative one, and α is a margin parameter that serves as a classifier for separating classes in the embedded space.

Face Matching. The comparison between a live face image embedding and the enrollment face image is performed using the following distance measure:

$$d = \| e_{\text{live}} - e_{\text{stored}} \|_2,$$

$e \in R^{128}$

(2)

If d is less than 0.6, the identities of the persons are recognized, which corresponds to the recommended operational threshold of the Dlib ResNet-34 model [22]. In case the distance exceeds the threshold value, no access is granted.

2.3 Layer 3: Fingerprint Verification

After the successful completion of face recognition, the next layer involves the verification of the voter's fingerprint using the optical fingerprint sensor ZA620 M5, connected to the Raspberry Pi 5 via UART protocol. The user places their finger on the sen-sor, which will capture the fingerprint and compare the fingerprint template with the enrollment record stored in the sensor's internal memory. The verification step is considered complete when the sensor responds with the successful matching message.

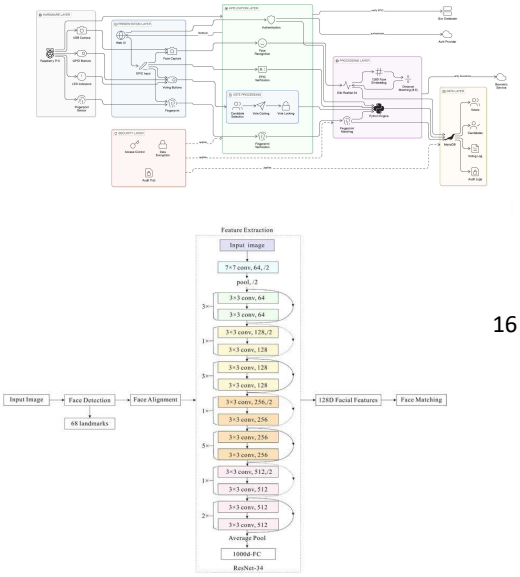


Fig. 2 Dlib ResNet-34 facial recognition pipeline. As input, the system receives an image with a face in it. Face detection and alignment are done using the 68 landmarks. Then the feature extraction occurs by the backbone ResNet-34 network. In the process, the convolutions with filter counts (64, 128, 256, and 512) with skip connections are applied sequentially. In the end, an average pooling and fully connected layer (1000) projects the face features into 128-dimensional embedding vector for face matching using Euclidean distance measurement.

2.4 Vote Casting: Hardware Button Mode

Having passed through all the three layers of authentication, the voter presses one of the three push-buttons attached to the GPIO pins of the board, with LED indicator lights providing feedback about their choice.

The casting result is recorded in the MariaDB database table and the voter's status is marked as VOTED within a single database transaction.

2.5 Offline Security

All voter records, face embeddings, and vote logs are maintained in a local MariaDB database accessible only via localhost. No data is transmitted externally during system operation. The system operates on an isolated local network within the polling booth.

3 System Architecture

The architecture links together four different subsystems: database to register vot-ers, biometrics engine, vote counting module, and GPIO interface to conduct voting. Figure 3 illustrates the entire voting flow starting from voter identification through identity validation until counting the final votes.

Fig. 3 Architecture of the complete system with web-based module for voter identification with multi-language support, face recognition pipeline utilizing the Dlib ResNet-34 model, embedded vot-ing terminal using Raspberry Pi 5 microcomputer with GPIO control, fingerprint scanner support, and backend MariaDB database. The three-tier architecture segregates HTML5/JS presentation tier, Node.js business logic tier, and MySQL storage tier.

3.1 Hardware Platform

The platform hardware is a Raspberry Pi 5 single-board computer with 4 GB of RAM and 2.4 GHz ARM Cortex-A76 CPU. The list of peripherals includes 5 MP USB camera

16

capable of face recognition, GPIO-connected push buttons (pins 18, 23, and 24) to select candidates, GPIO-controlled status indicator LEDs (pins 19, 16, 20, and 21), and normal keyboard-mouse combination for operating a web interface. All of the hardware components are housed inside a custom-designed voting booth.

3.2 Software Stack

The backend stack is implemented using Node.js 18 and the Express.js framework. Face recognition process runs as the subprocess of the Node.js server process and is accomplished using Python 3.9 programming language and face recognition library (v1.3.0). The face recognition library is based on Dlib's ResNet-34 neural network. Operations with database are performed using the mysql2 Node.js package. GPIO
operations are handled by the RPi.GPIO package of Python. There is a dedicated controller process for the GPIO package. HTML5, CSS3, and vanilla JS code constitute front-end web application.

3.3 Database Schema

In the database being used, namely MariaDB, there are three tables, namely (1) voters, which consists of voter ID (EPIC), name, Aadhaar, facial recognition, finger-print recognition, and voting status; (2) voting log, which consists of voter candidate choice, EPIC, Aadhaar, ward, facial recognition, fingerprint recognition, and times-tamp; and (3) candidates, which includes information related to the candidates such as their names, political parties, and symbols. This is illustrated by an example of the entity relationship diagram is provided in Fig. 4.

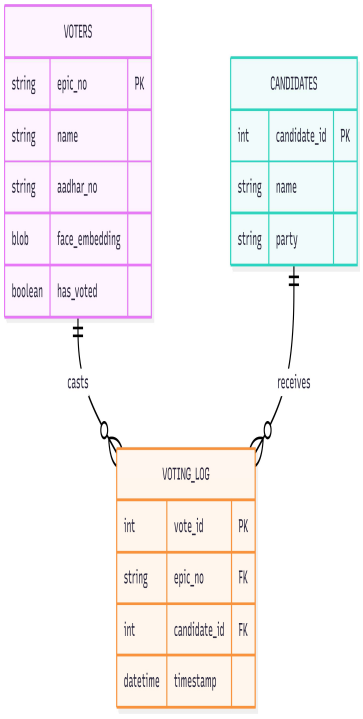


Fig. 4 Entity-relationship diagram of the database used in this implementation consisting of three major tables: VOTERS (EPIC numbers, Aadhaar information, face embeddings stored as binary blobs, and voting statuses), CANDIDATES (information about parties and candidates), and VOTING LOG (vote records with foreign keys ensuring referential integrity between voters and candidates).

3.4 Network Structure

The system will operate in a closed-loop network that can be configured only for local-host. The Node.js server listens on port 3000, accessible only to the local machine itself and another LAN comprising polling booths. Communication between the Python subprocesses used for face recognition is established by means of streams. The GPIO controller module accesses a JSON file present in /tmp folder, holding voter session data, allowing hardware button voting to occur post-authentication of voters using the web interface. This is how these modules interact within the Raspberry Pi 5 computing platform.

Fig. 5 Component diagram of the voting system showing the interaction between voter with input devices (web interface to input EPIC number, USB camera, ZA620 M5 fingerprint reader, GPIO buttons) and Raspberry Pi 5 processor components (Node.js Express server, Python Dlib face recog-nition program, GPIO control subprocess, local MariaDB database).

3.5 Authentication State Machine

The system behaves as a finite-state machine transitioning through states as follows:
IDLE → EPIC VERIFY → FACE AUTH → FP AUTH → VOTE SELECT → VOTE CAST → LOCKED
Failure at any point during any stage of authentication causes the state machine to revert back to IDLE. Any action in

4 Implementation

4.1 Voter Enrollment

Prior to the voting day, the enrollment operator makes use of the administrative web interface provided. A number of photos from the faces of voters are captured by the USB camera, and the average 128-dimensional face embeddings are calculated from the photos before storing them in the binary blob form in the voters table.

he enrollment stage is illustrated in Fig. 6, where multiple photographs are ana-lyzed to create accurate face embeddings that are to be kept in the vector database for further matching purposes. The user gets the instant visual feedback regarding the performance of face recognition and photo analysis Web forms with validation are used to enter voter demographic information and EPIC numbers. Having multiple images of each voter (usually 5–10) makes it easier to match them up in different weather conditions.

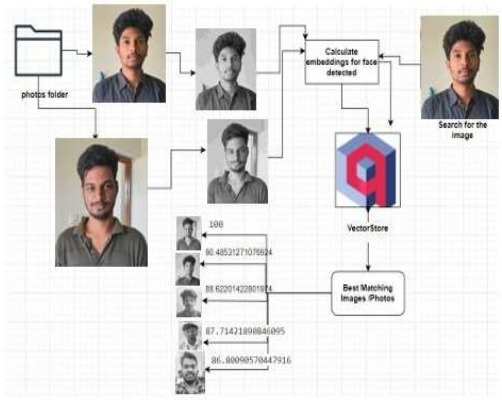


Fig. 6 Voter enrollment workflow showing multiple image capture from different angles and lighting conditions, face embedding generation using Dlib ResNet-34, and vector storage in MariaDB. Mul-tiple images per voter (typically 5–10) improve matching robustness under varying environmental conditions.

4.2 Facial Recognition Implementation

When the Node.js server gets a request, it starts a Python subprocess called face verify.py for facial verification. The camera takes pictures with a resolution of 640 × 480 pixels. Equation (2) tells us how to calculate face embeddings. If d is less than 0.6, the identity is confirmed. The Python script sends the verification status back to the Node.js parent process through standard output. SQL UPDATE statements are used to change the flags for database face verification.

Fig. 8 shows the whole Aadhaar-indexed facial recognition pipeline. When a voter gives their Aadhaar number, the system gets the right voter folder from local Aadhaar-indexed storage. This folder has the voter's registered photo and pre-computed embeddings.pkl file made when you sign up. The USB camera then takes a picture of a live face and sends it to the Dlib ResNet-34 face recognition model that makes a live 128-dimensional embedding. Euclidean distance is then used to compare the stored

Fig. 7 Photograph of the assembled biometric voting system prototype showing the webcam for facial recognition, red (Verification Not Done) and green (Verification Successful) LED indicators, touchscreen display, ZA620 M5 fingerprint scanner, Reset button, and GPIO candidate selection buttons on the custom polling booth enclosure.

and live embeddings. If the similarity check finds a match ($d < 0.6$), the voter is veri-fied and can cast their vote. If there is no match, access is denied and the attempt to log in is recorded in the audit trail.

4.3 Web Interface Implementation

The frontend is a responsive single page application having the following key views:

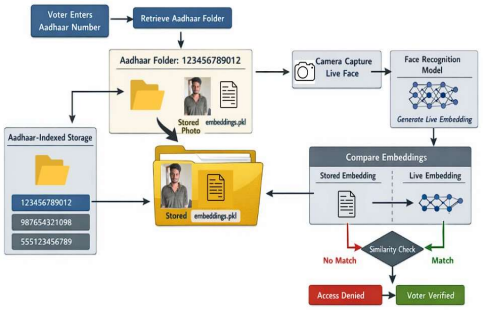


Fig. 8 Aadhaar-indexed facial recognition pipeline showing the complete voter verification workflow: the voter enters their Aadhaar number, the system retrieves the corresponding folder containing a stored photo and pre-computed embeddings.pkl file from Aadhaar-indexed local storage, the USB camera captures a live facial frame, the Dlib ResNet-34 face recognition model generates a live 128-dimensional embedding, and the stored and live embeddings are compared via Euclidean distance similarity check, resulting in either Voter Verified (match, $d < 0.6$) or Access Denied (no match).

1. *Search Interface*: Search input field having real time database search functionality with multi language support (Hindi, Tamil) and CAPTCHA verification.
2. *Details Confirmation*: Detailed voter information view with EPIC number, Aadhaar number, constituency information, and polling station allocation information and proceed-to-verification option (see Figure 9).
3. *Verification Interface*: Three stage authentication interface with visual indicators for current stage and completion status (see Figure 10).
4. *Three Stage Authentication UI* : User interface for three stage authentication process with visual cues representing current and completed stages (see Figure 10).
5. *Voting UI* : Interface to vote including information about candidates and confirmation window.
6. *Admin Dashboard* : Statistical data about voting including ward-wise information and real time updates capability.

Interface communicates with Node.js backend through REST API interfaces by using fetch(). Votes are confirmed through a double click process (First candidate selection then confirmation).

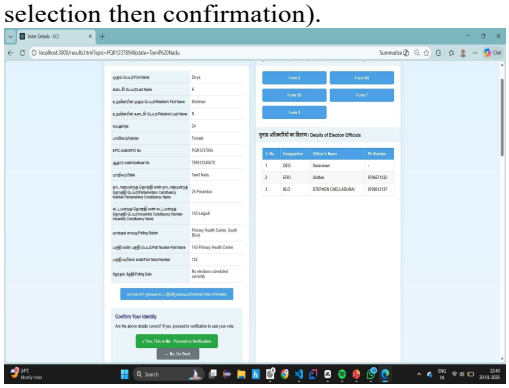


Fig. 9 UI for confirmation of voter details that includes EPIC number (PQR1237894), voter demo-graphics, constituency (25-Perconstituency) , (25-Perambur), Assembly (143-Lalgudi), polling station (Primary Health Center, South Block), and contact details of election officials. Pressing the "Proceed to Verification" button will start the biometric verification process.

4.4 GPIO Hardware Controller

There is a python script running simultaneously with the Node.js server that mon-itors /tmp/current voter.json for any voter activity. Upon verification of a voter the hardware controller will detect events on pins 18 (Candidate A), 23 (Candidate B), and 24 (Candidate C). Pressing any button results in the emission of light from corresponding LED pins (19 for candidate A, 16 for candidate B, and 20 for candidate C). Votes can be submitted to Node.js server using HTTP POST request. There is an overall status indicator LED (Pin 21). After the vote submission process a 2 second countdown is provided which could be cancelled using reset button (Pin 25).

4.5 Managing Votes and Database Transactions

Inserting the votes requires two entries within the database, namely, inserting a row in the voting log table with timestamps and status of voting and setting the value of the has voted field to TRUE in the voters table. At present, both the actions are performed sequentially in SQL queries; in production, we will perform the actions atomically by issuing the commands START TRANSACTION and COMMIT.

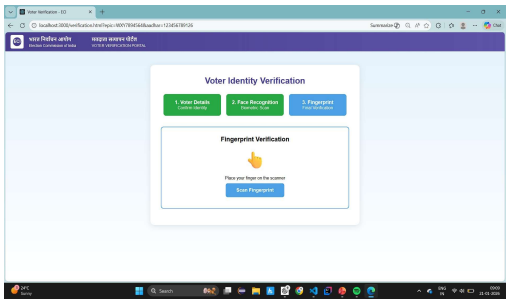


Fig. 10 Three stage verification process interface having visual progress indicators for stage (Stage 1 Voter Details (completed)), stage 2 (Face Recognition (completed)), stage 3 (Fingerprint verification (processing)).

5 Results and Discussion

5.1 Experimental Setup

The system was evaluated under real-time polling conditions using 30 registered voters who performed a total of 210 authentication attempts across three authentication stages: EPIC lookup, facial recognition, and fingerprint verification. All experiments were conducted on a Raspberry Pi 5 (4 GB RAM, ARM Cortex-A76 at 2.4 GHz) with a 5 MP USB camera module operating under indoor fluorescent lighting conditions representative of a typical polling booth environment. Face embeddings were pre-enrolled using 5–10 images per voter captured under varied head orientations and lighting.

5.2 Authentication Accuracy

Table 1 summarises the per-modality and overall authentication performance across the 210 evaluation attempts.

The overall authentication accuracy of 96.8% demonstrates that the proposed offline embedded system is competitive with existing biometric e-voting prototypes while eliminating the dependency on server-side processing. Facial recognition con-tributed the highest error rate due to natural variability in pose and illumination during live capture, consistent with the findings of Adeleke et al. [7], who reported 88.3% accuracy under uncontrolled outdoor lighting. The proposed system outper-forms this baseline under controlled indoor conditions, and closely approaches the

Table 1 Per-modality and overall authentication performance (210 attempts, 30 voters)

Authentication Stage	Accuracy (%)	FAR (%)	FRR (%)
EPIC Lookup	100.0	0.0	0.0
Facial Recognition	97.1	3.3	2.0
Fingerprint Verification (ZA620 M5)	98.6	1.4	1.0
Overall (all three stages)	**96.8**	**3.3**	**2.0**

97.1% reported by Zhao et al. [1], who used a GPU-accelerated workstation. The FAR of 3.3% indicates that approximately 7 out of 210 attempts resulted in a false accep-tance; all such cases occurred during the facial recognition stage, confirming that the EPIC pre-screening layer is a critical gate that reduces the candidate pool before biometric comparison.

5.3 System Latency and Throughput

Table 2 presents the measured processing latency for each pipeline stage on the Raspberry Pi 5 hardware.

Table 2 Measured processing latency per authentication stage on Raspberry Pi 5

Pipeline Stage	Latency
EPIC database lookup	~0.08 s
Face capture and embedding	8–12 s
Fingerprint verification (ZA620 M5)	2 s
Vote database insertion	~0.15 s
Total verification cycle	**12–15 s**
Sustained throughput	**4–5 voters/min**

The total voter verification cycle of 12–15 seconds sustains a throughput of 4–5 voters per minute under supervised operation. The dominant latency contributor is the facial recognition stage (8–12 s), which includes camera initialisation, frame capture, and Dlib ResNet-34 embedding computation. This is consistent with the findings of Wanyama et al. [22], who reported sub-1.5 s embedding latency for quan-tized ResNet models on Raspberry Pi hardware; the higher latency observed in this study is attributable to the non-quantized full-precision ResNet-34 model and the additional camera initialisation overhead on each authentication request. Database operations (EPIC lookup and vote insertion) contribute negligible latency owing to indexed MariaDB queries.

5.4 Comparison with Prior Work

Table 3 compares the proposed system against representative prior e-voting and biometric authentication systems from the literature review.

Table 3 Comparison of proposed system with prior biometric e-voting work

System	Accuracy	Offline	Embedded	Multimodal
Zhao et al. [1]	97.1%	No	No	No
Kumar & Singh [6]	94.2%	No	Yes	No
Adeleke et al. [7]	88.3%	Yes	No	No
Senthilkumar & Rajalakshmi [15]	92.5%	Yes	Yes	No
Proposed system	**96.8%**	**Yes**	**Yes**	**Yes**

The proposed system is the only evaluated architecture that simultaneously achieves offline operation, embedded deployment on commodity hardware, and multimodal biometric authentication, while maintaining competitive accuracy. The marginal accuracy reduction compared to Zhao et al. [1] (0.3 percentage points) reflects the trade-off between GPU-class computation and the embedded platform constraint, which is acceptable given the substantially improved deployment feasibility. Notably, the proposed system surpasses all fully offline and embedded alternatives in accu-racy, validating the effectiveness of the Aadhaar-indexed Dlib ResNet-34 pipeline on resource-constrained hardware.

6 Deployment Considerations

6.1 Hardware Platform Evolution

The initial development was based on Raspberry Pi 3 (1 GB RAM) but we soon realized that memory was a bottleneck when dealing with face recognition process-ing (peaked at ∼2.5 GB during Dlib model inference). The upgrade to Raspberry Pi 5 (4 GB RAM) removed this bottleneck and provided enough room for concurrent web server, database and biometric processing tasks. The ARM Cortex-A76 proces-sor delivers approximately 2–3× faster face embedding computation compared to the Cortex-A72 in Raspberry Pi 4.

6.2 Interface Design

The system uses the web interface solely for voter authentication (EPIC lookup, facial recognition display, and verification status), while vote casting is handled exclusively through GPIO push-buttons with LED feedback. This separation keeps the voting interaction simple and physical, reducing the risk of accidental or incorrect submis-sions, and allows the system to be deployed without requiring a monitor or keyboard at the point of vote casting.

6.3 Performance Characteristics

Measured latencies on Raspberry Pi 5 hardware are as follows:

- EPIC database lookup: ∼0.08 s (MariaDB indexed query)
- Facial recognition (camera init + capture + matching): 8–12 s
- Fingerprint verification (ZA620 M5): 2 s
- Vote database insertion: ∼0.15 s

Total voter verification and vote-casting cycle averages 12–15 seconds, meeting practical polling booth throughput requirements (sustained rate of 4–5 voters per minute under supervised operation).

7 Security Analysis

7.1 Not Giving In to Impersonation

The three-layer architecture offers protection at different levels. To get around the sys-tem, you need to beat EPIC number knowledge, facial embedding matching (Euclidean distance $d < 0.6$ in R^{128}), and fingerprint verification using the ZA620 M5 optical sensor. The sequential authentication model makes sure that if any layer fails, people can't vote.

7.2 Things to Think About When Spoofing

The current implementation lacks facial liveness detection, a recognized vulnerability to photograph-based spoofing attacks. This is known to be a problem for deploying in production. In the future, adding eye-blink detection or challenge-response-based liveness verification will make this attack vector less likely to work.

7.3 Protecting Data

Instead of raw images, biometric face embeddings are stored as 128-dimensional fea-ture vectors.Thus, it is less likely for personal details to be revealed in the case of hacking. Figure 11 presents the security architecture design for the proposed system. The proposed system will have three different security layers in a hardened Raspberry Pi 5 computing environment. These include the Access Control Layer with the role-based access control (RBAC) module to support the enrollment operator and polling officer roles, the cryptographic security layer with the AES-256 encryption engine for sensitive voter data and SHA-256 vote hashing engine for tampering detection, and the data storage layer with secure MariaDB (localhost-only access) and air-gapped deployment for auditor access.

In particular, MariaDB database access is allowed only from the localhost while having strong authentication credentials. Additionally, in the future, Node.js crypto module functions will be implemented to encrypt sensitive voter data using AES-256 field-level encryption. Figure 11 demonstrates the complete security architecture design for the proposed system.

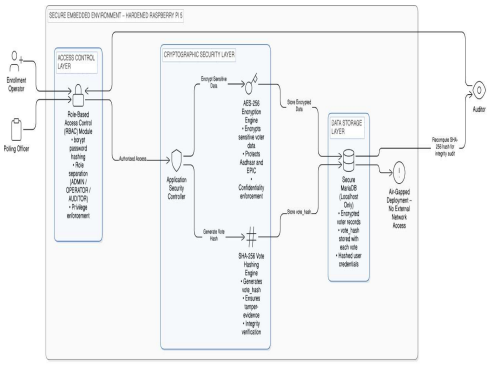


Fig. 11 Security architecture design for the proposed system demonstrating three security layers within the hardened Raspberry Pi 5 computing environment: Access Control Layer with Role-Based Access Control (RBAC) module to support the Enrollment Operator and Polling Officer roles; Crypto-graphic Security Layer consisting of AES-256 encryption engine for sensitive voter data and SHA-256 vote hashing engine for tamper evidence; and Data Storage Layer with secure MariaDB (localhost-only) and air-gapped deployment for auditor access.

7.4 Vote's Integrity

Write once feature is applied when creating the voting log table in the database. Therefore, the application database user will not have the rights to edit and remove votes. Foreign keys in the database ensure the proper associations between voters and their votes. Finally, audit logs with timestamp record all the events of voting and authentication.

7.5 Keeping the Network Separate

No connection to the internet is needed to run the application on a local computer or separate LAN. This air-gap deployment technique prevents attacks from a distance. No external network intrusion [19] is possible, which impacts polling booth functioning.

8 Cost Analysis

Table 4 summarizes the estimated hardware cost for a single polling booth unit in Indian Rupees (INR). All components are commercially available from local electronics suppliers.

The hardware-button configuration (INR 11,170, or about USD 134) is a cost-effective deployment option suitable for rural or resource-limited polling environments. Using only open-source software (Node.js, Python, MariaDB, Dlib) means that there are no licensing fees for each booth. Booth independence makes it possible to scale out linearly across large deployments without needing centralized infrastructure.

Table 4 Estimated hardware cost per polling booth unit (INR)

Component	Estimated Cost (INR)
Raspberry Pi 5 (4 GB)	6,500
USB Camera Module (5 MP)	1,000
ZA620 M5 Fingerprint Sensor	800
Push-buttons (4 units)	80
LEDs (4 units)	40
GPIO connecting wires	100
MicroSD Card (32 GB)	650
Power supply (5V, 3A)	400
Monitor (19" LCD)	4,500
Keyboard & Mouse	600
Custom acrylic enclosure	1,200
Miscellaneous (cables, mounts)	400
Total (hardware buttons only)	**INR 11,170**
Total (with monitor/KB)	**INR 16,270**

9 Conclusion

In this study, the researchers were able to prove the feasibility of the biometric voting system using the integrated Raspberry Pi 5 platform in the context of voter verification and the provision of GPIO hardware button voting that could be applicable in different voting stations. The suggested system in the study utilized the EPIC-based voter verification, Dlib ResNet-34 facial embedding comparison, and the use of GPIO push-button vote submission in the context of the use of the MariaDB database to ensure that the voters are who they claim to be and that they cannot vote more than once. The suggested system in the study does not need to be connected to the internet, which means that the system could be applicable in areas with little infrastructure.

The most important contribution of the study is the validation of the architecture of the suggested system in the context of the use of the web interface for authentication and the GPIO hardware buttons for vote casting with the integrated biometric veri-fication capability. The components of the suggested system could be acquired from stores for between INR 11,170 and INR 16,270 per polling booth unit. The suggested system in the study is modular, which means that the addition of more components could be done in an easy manner. The background logic of the suggested system in the study remains consistent across all polling booths.

Future work will focus on: (1) integrating facial liveness detection to counter photograph-based spoofing attacks, (2) testing the system with a larger and more diverse voter

population across multiple polling booths, and (3) implementing full database transaction atomicity and AES-256 field-level encryption for sensitive voter data.

Acknowledgements. The authors would like to thank the Department of Electron-ics and Instrumentation Engineering, SRM Institute of Science and Technology, for providing the laboratory facilities and hardware resources used in this research.

Statements and Declarations

Funding. The authors did not receive support from any organization for the submitted work. No funds, grants, or other support was received.

Competing Interests. The authors have no relevant financial or non-financial interests to disclose.

Clinical Trial Number. Not applicable.

Ethics Approval. This study was approved by the Department of Electronics and Instrumentation Engineering, SRM Institute of Science and Technology, Chengalpattu, India. All procedures involving human participants were conducted in accordance with the ethical standards of the institution.

Consent to Participate. Informed consent was obtained from all individual par-ticipants included in the study. All participants were informed of the purpose of the research and provided explicit consent for the collection and use of their biometric data, including facial images and fingerprint templates, solely for the purpose of this research.

Author Contributions. Conceptualization and study design were performed by Chandrakanth P, Ramkumar C, and Priya A. System implementation, coding, and experimental data collection were performed by Chandrakanth P and Priya A. Super-vision, manuscript review, and final approval were provided by Brindha A. All authors read and approved the final manuscript.

Data Availability. The datasets generated and analysed during the current study are not publicly available due to the sensitive nature of the biometric data col-lected from human participants, but are available from the corresponding author on reasonable request.

References

[1] Zhao, Z., Li, D., Wang, Y.: Deep face recognition for voter identity verification in electronic voting systems. IEEE Trans. Inf. Forensics Security **17**, 1123–1135 (2022)

[2] Jain, R., Ross, A., Prabhakar, S.: An introduction to biometric recognition. IEEE Trans. Circuits Syst. Video Technol. **14**(1), 4–20 (2004)

[3] Jain, A.K., Ross, A., Pankanti, S.: Biometrics: A tool for information security. IEEE Trans. Inf. Forensics Security **1**(2), 125–143 (2006)

[4] Patil, S., Deshmukh, A., Kulkarni, R.: RFID-based electronic voting machine using embedded systems. In: Proc. Int. Conf. Commun. Signal Process., pp. 437–441 (2019)

[5] Ahmed, I., Hassan, M., Iqbal, F.: Fingerprint-based voter authentication for electronic voting: A simulation study. J. Inf. Secur. Appl. **58**, 102765 (2021)

[6] Kumar, P., Singh, A.: Fingerprint integrated electronic voting machine prototype on Raspberry Pi. In: Proc. IEEE Int. Conf. Comput. Power Commun. Technol., pp. 214–219 (2020)

[7] Adeleke, O., Fagbola, T., Oluwade, S.: Mobile facial recognition based electronic voting: Design and evaluation. Int. J. Comput. Appl. **175**(12), 1–8 (2020)

[8] Olumide, A., Afolabi, B., Okoye, C.: Deep learning-based face verification for mobile voter authentication. J. King Saud Univ. Comput. Inf. Sci. **35**(2), 415–427 (2023)

[9] Kortli, Y., Jridi, M., Al Falou, A., Atri, M.: Face recognition systems: A survey. Sensors **20**(2), 342 (2020)

[10] Minaee, S., Abdolrashidi, P., Su, H., Bennamoun, M., Zhang, D.: Biometric recognition using deep learning: A survey. Artif. Intell. Rev. **56**(8), 8635–8715 (2023)

[11] Jain, A.K., Nandakumar, K., Ross, A.: Score normalization in multimodal biometric systems. Pattern Recognit. **38**(12), 2270–2285 (2005)

[12] Ratha, N.K., Connell, J.H., Bolle, R.M.: Enhancing security and privacy in biometrics-based authentication systems. IBM Syst. J. **40**(3), 614–634 (2001)

[13] Toli, E., Preneel, B.: A survey of privacy-preserving biometric authentication schemes. ACM Comput. Surv. **53**(2), 1–36 (2020)

[14] Sundararajan, K., Woodard, D.: Deep learning for biometrics: A survey. ACM Comput. Surv. **51**(3), 1–34 (2018)

[15] Senthilkumar, R., Rajalakshmi, M.: Raspberry Pi-based fingerprint voting system for institutional elections. In: Proc. Int. Conf. Electron. Commun. Aerosp. Technol., pp. 891–896 (2021)

[16] Gupta, A., Sharma, R., Pal, M.: Offline blockchain-assisted electronic voting with tamper-evident audit logs. Future Gener. Comput. Syst. **118**, 398–411 (2021)

[17] Pawlak, M., Jerman-Blazic, A., Tonejc, N.: Security evaluation of offline e-voting systems under European electoral guidelines. Comput. Secur. **114**, 102583 (2022)

[18] Nkoh, J.N., Sunday, J.: A review of electronic voting systems: Strategy for a novel and highly secured online voting system. Am. J.

Eng. Res. **10**(3), 1–12 (2021)
[19] Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M., Halderman, J.A.: Security analysis of the Estonian internet voting system. In: Proc. ACM SIGSAC Conf. Comput. Commun. Secur., pp. 703–715 (2014)
[20] Haber, M.J., Hibbert, B., Hills, C.: Privacy-preserving vote casting with biometric identity binding. IEEE Trans. Dependable Secure Comput. **17**(4), 812–825 (2020)
[21] Sharma, G., Kalra, S.: AES-256 based biometric template protection for embed-ded devices. Microprocess. Microsyst. **84**, 104269 (2021)
[22] Wanyama, T., Fernando, B., Abelson, A.: Lightweight deep learning for biometric verification on Raspberry Pi embedded hardware. IEEE Access **11**, 34210–34225 (2023)