Enhancing Electoral Integrity: A Comparative Analysis and Novel Framework for Secure Electronic Voting using Biometrics, Blockchain, and Threshold Cryptography

Adrijeet Deb

M. Eng in Information Systems Security
Concordia University
Montreal, Canada
adrijeet.deb@outlook.com

***Abstract*—Traditional paper-based voting and existing electronic voting systems face significant challenges regarding security, accuracy, and integrity [1], [2]. Voter fraud and system vulnerabilities undermine trust in democratic processes. This paper proposes a novel framework to enhance electoral integrity by enhancing the security of electronic voting system. The proposed solution integrates biometrics for robust voter authentication [3], blockchain technology for creating transparent and immutable voting records [12], [13], and threshold cryptography for distributing trust and enhancing privacy [6], [7]. A comparative analysis explores different cryptographic techniques, including symmetric encryption, fuzzy commitments [4], [5], and threshold schemes for registration and authentication within the proposed architecture. The paper also reviews and contrasts existing prominent electronic voting systems such as Helios [8], Belenios [9], ElectionGuard [10], and VotingWorks. Furthermore, it details a conceptual design that integrates threshold authentication principles with the ElectionGuard framework using mechanisms such as blind BLS signatures [15] and zero knowledge proofs [16]. The objective is to establish a comprehensive system that ensures fairness, eligibility, uniqueness, privacy, accuracy, and efficiency, thus bolstering voter confidence and the security of elections.**

***Index Terms*—Blockchain, ElectionGuard, Voting, Authentication, Threshold Encryption**

I. Introduction

In our effort to find an optimal solution for voting processes, we needed to identify the problems first. The objective and the motivation to pursue this project has been outlined below.

A. Objective

In developing an enhanced voting system, our first step was to conduct a comprehensive analysis of existing electoral challenges. Our project's foundation rests on preserving and strengthening the core principles that have historically governed paper-based voting systems. Our framework adheres to six fundamental objectives that are essential for maintaining electoral integrity: [8]

- **Fairness:** The system must prevent any form of discrimination, ensuring equal access and treatment for all voters regardless of their background or circumstances.
- **Eligibility:** Only properly registered and legally qualified voters may participate in the electoral process, with robust verification mechanisms in place.
- **Uniqueness:** The system must implement sophisticated safeguards to prevent multiple voting attempts by the same individual, ensuring each eligible voter can cast exactly one vote.
- **Privacy:** Complete anonymization of all voting data must be maintained to protect voter confidentiality, making it infeasible to link specific votes to individual voters.
- **Accuracy:** The system must guarantee precise vote tabulation and recording, with mechanisms to verify that every legitimate vote is properly counted and no valid votes are lost or altered.
- **Efficiency:** While maintaining security and accuracy, the system must provide timely access to election results through optimized processing and tabulation methods.

B. Motivation

Democratic societies rely on voting as the cornerstone of citizen participation in governance [2]. While paper-based voting has been the traditional method since democracy's inception, it faces persistent challenges including lost ballots, counting errors, and invalid votes. In recent years, electronic voting systems have gained traction as an alternative, with multiple countries using them to conduct elections. Paper-based voting provides physical records that can be manually recounted, offering transparency but slower results compared to electronic systems. While paper ballots are less vulnerable to cyber attacks and technical failures, they can be compromised through physical tampering. Electronic voting delivers faster results and better accessibility features but faces risks of software manipulation and hacking. Paper systems require less

technical infrastructure but more personnel, whereas electronic systems need significant technical setup but can reduce staffing needs. However, current e-voting systems still harbor various security vulnerabilities that must be addressed [1]. For an e-voting system to gain widespread acceptance and trust, it must demonstrate both transparency and robust security measures. The right to vote represents the most essential privilege granted to citizens by democratic institutions, yet voter fraud continues to undermine electoral integrity across many jurisdictions. Our proposed solution combines smart card technology, biometric verification [3], and blockchain architecture [12], [13] to create a comprehensive system designed to eliminate voter fraud while maintaining democratic principles.

C. Background

Modern authentication systems are rapidly evolving beyond traditional passwords toward more sophisticated biometric solutions [3]. The widespread adoption of biometric authentication can be attributed to its dual advantages of enhanced security and user convenience. While today's security landscape offers an unprecedented array of protection tools, this proliferation also brings new vulnerabilities that must be carefully managed. Leading global organizations have responded by implementing multi-layered authentication frameworks to safeguard their most valuable intellectual property and data assets, often worth billions or trillions of dollars. Drawing inspiration from these enterprise security architectures, our research presents an integrated approach that combines multiple authentication mechanisms to address the persistent challenge of voter fraud. This issue has undermined democratic processes since their earliest implementations. We propose that contemporary technological advances, particularly the integration of smart card circuits and chips with blockchain technology, can finally provide a definitive solution to electoral fraud while maintaining the integrity of democratic institutions.

II. Methodology

A. Registration

Voter registration in electronic voting systems generally involves creating and maintaining a digital electoral roll that accurately identifies eligible voters. The process typically begins with the collection of citizens' personal identification information, which may include name, address, date of birth, citizenship status, and some form of unique identifier such as a social security number or national ID number. Modern e-voting registration systems often incorporate biometric data collection to enhance security and prevent fraud. This may include fingerprints, facial recognition data, iris scans, or other physiological identifiers that are unique to each individual. These biometric markers serve as a powerful authentication mechanism that is difficult to forge or duplicate. Registration data is typically stored in secure, centralized databases maintained by electoral authorities. Many systems implement a distributed database architecture with redundancies to prevent data loss and ensure system availability. Advanced encryption protocols protect this sensitive information from unauthorized access or tampering. The registration process may be conducted through various channels, including dedicated registration centers, online portals, mobile applications, or integration with existing government identification systems. Some countries have implemented automatic voter registration that activates when citizens interact with other government services, such as obtaining a driver's license. Regular audits and database maintenance help ensure the accuracy of voter rolls by removing deceased voters, updating address changes, and identifying potential duplicate registrations. This ongoing maintenance is critical to maintaining the integrity of the electoral process and preventing opportunities for fraud.

1) Utilizing standard symmetric encryption: The registration process begins at designated registration centers, where voters must appear in person. This physical presence requirement is crucial for the system's security, as it allows officials to verify the initial identity of voters and ensures that the biometric data collection and smart card programming happen in a controlled environment. The registration centers are equipped with specialized hardware for capturing biometric data and writing to the smart cards. When a voter arrives to register, they first need to select their voting area or constituency. This area selection is important because it ties into the system's area-based database structure, where voter information is organized into separate blockchain blocks for each constituency. After selecting their area, voters fill out a registration form with their personal details. The system then moves on to the critical biometric capture phase. The biometric capture process is particularly sophisticated, using a novel encryption approach that combines DNA encoding/decoding with chaotic maps. When the system captures the voter's biometric data (in this case, their facial image) [3], it first converts this data into a template. This template then goes through an intricate encryption process: the system uses chaotic systems to scramble the template, converts it using DNA encoding (based on the nucleic acids A, C, G, T), creates a complement of this encoding, and then decodes it again. The output is then hashed using SHA2 to generate an encryption key. This key is used to encrypt the actual biometric template using AES256 encryption.[Appendix A] The encrypted template is then written to the voter's smart card, while the encryption key is stored securely in the database against the voter's national ID number or social security number. The system also generates and stores a hash of the encrypted template, which will be used later for verification purposes. This creates a secure link between the physical smart card, the voter's identity, and their biometric data, while ensuring that the original biometric data cannot be reconstructed even if the card is compromised. Another interesting aspect of the registration process is how it handles multiple biometrics. The system is designed to store multiple biometric templates for each voter, providing redundancy and flexibility. This means if one biometric identifier becomes unusable (for instance, if a finger is injured), the voter can still be authenticated using

an alternative biometric. All of this registration data becomes part of the area-based blockchain, creating an immutable record of voter registration while maintaining the privacy and security of voter information. The entire registration process is designed to establish a chain of trust that begins with in-person verification and extends through the technological safeguards of encryption, hashing, and blockchain storage [13]. This comprehensive approach creates a foundation for secure voter authentication during the actual voting process while protecting against various forms of identity theft and voter fraud.

2) Using fuzzy commitments: In a fuzzy commitment-based registration system [5], we would still require voters to appear at designated registration centres, but the way we process and store their biometric data would be quite different. Instead of encrypting the actual biometric template, we would create a commitment that binds the biometric data to a randomly generated secret. Specifically, this binding can be represented by the formula:

$$H = E(F(b), s)$$

Where H is the helper data, $F(b)$ is the feature vector derived from the biometric data b, s is the randomly generated secret, and $E(\cdot, \cdot)$ represents the stream cipher encryption operation. This binding effectively "locks" the secret s with the biometric feature vector $F(b)$.

The core of this new registration process would involve converting the captured biometric data (facial image) into a binary feature vector. This is done through careful feature extraction and quantization processes that maintain the distinctive aspects of the biometric while creating a consistent binary representation. The system would then generate a random secret value and use it to create two crucial pieces of information:

1) The helper data: Created by encrypting the binary feature vector with the secret as the key stream ($H = E(F(b), s)$). We use a stream cipher rather than conventional encryption because it preserves the error tolerance properties needed for biometrics. Unlike standard encryption which requires exact inputs for decryption, stream ciphers allow for recovery of the secret even when the verification biometric sample is slightly different from the enrollment sample, as long as the difference falls within an acceptable threshold. This property is essential for biometric systems since two readings of the same biometric will never be identical.
2) The commitment: Created by hashing the secret ($C = \text{hash}(s)$). The commitment C is stored on the blockchain, while the helper data H is stored on the smart card. The actual secret s is never stored permanently; it exists only transiently during registration and verification. During verification, the system attempts to recover $s' = D(F(b'), H)$ (where b' is the freshly captured biometric and $D(\cdot, \cdot)$ is the decryption operation) and checks if $\text{hash}(s')$ matches the stored commitment C.

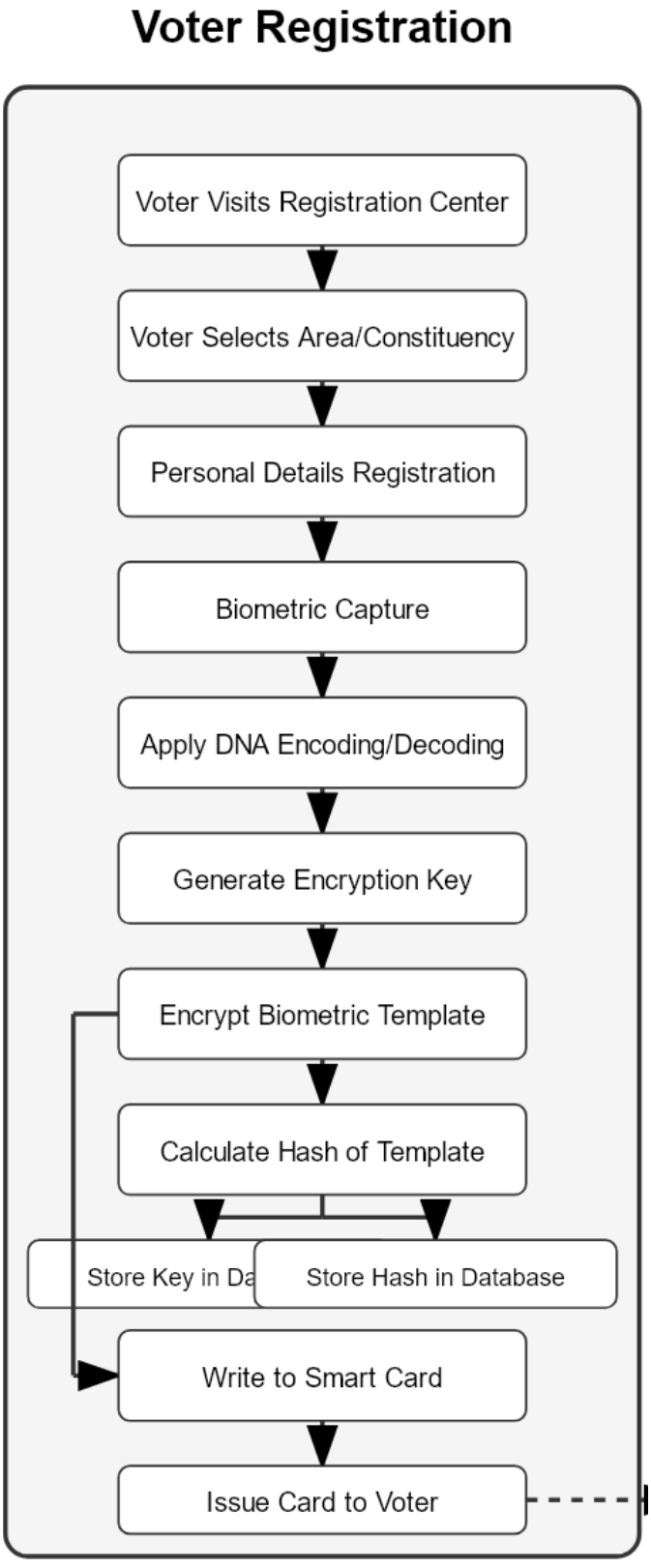

Fig. 1. Registration process while utilizing Blockchain with biometrics and smart cards

This approach offers several advantages over the current DNA encoding and AES encryption method. First, it's specifically designed to handle the natural variations in biometric readings, eliminating the need for exact matches during verification. Second, it provides strong security guarantees - even if an attacker obtains both the helper data and the commitment, they cannot feasibly recover either the original biometric data or the secret value [4], [5].

Regarding revocability, the system allows for generating new commitments for the same biometric in case the existing one is compromised. This works as follows:

1) If a user's smart card is lost or stolen, the system can generate a new random secret s_{new}
2) Create a new helper data $H_{\text{new}} = E(F(b), s_{\text{new}})$ using the stored or newly captured biometric
3) Generate a new commitment $C_{\text{new}} = \text{hash}(s_{\text{new}})$
4) Store the new commitment on the blockchain and issue a new smart card with the new helper data

This effectively invalidates the previous commitment without requiring changes to the user's biometric data itself. Since the secret is randomly generated each time, there is no correlation between the old and new commitments even though they are derived from the same biometric, providing both security and convenient revocability. The process is explained below:

1) **Initial Setup:** Voter arrives at registration centre where an official verifies their physical identification documents. During this process, the voter selects their constituency/area which will determine their voting location and eligible candidates.
2) **Biometric Processing:** The system captures the voter's biometric (facial image) and extracts robust features from this biometric data [3]. These features are then converted to a fixed-length binary string $F(b)$ that uniquely represents the voter. To handle natural variations in biometric readings, error correction coding is applied to ensure consistent verification later.
3) **Commitment Generation:** The registration system generates a random secret string s using a cryptographically secure random number generator. The system creates helper data H by performing an XOR operation between the secret and the binary features extracted earlier: $H = F(b) \oplus s$. A commitment $C = \text{hash}(s)$ is generated by hashing the secret. The random secret s itself is never stored permanently; it exists only during the registration process.
4) **Storage and Recording:** The helper data H is stored on the voter's smart card which they will use during voting. The commitment C is recorded on the area-based blockchain for verification purposes. During the verification phase, when a voter presents their smart card and provides a new biometric reading b', the system computes $s' = F(b') \oplus H$. If $F(b')$ is sufficiently similar to the original $F(b)$, then s' will be equal or very close to s, and $\text{hash}(s')$ will match the stored commitment C.
5) **Verification Testing:** Before the voter leaves the registration center, the system performs a test verification with a new biometric reading to ensure the commitment scheme works properly with the natural variations in the voter's biometric. This involves capturing a fresh biometric sample, performing the verification process described above, and confirming that the recovered secret produces a matching commitment. If verification consistently fails (which may occur with certain biometric features that show high variability), the system may adjust error correction parameters or, in rare cases, suggest an alternative biometric modality for that specific voter.
6) **Documentation:** A voter registration confirmation is issued to the voter, along with instructions on how to properly handle their smart card. The registration is recorded in an audit log for future reference, and the area-based blockchain is updated to reflect the completed registration.

This new process maintains the security benefits of the original system while adding better handling of biometric variations and stronger privacy guarantees through the fuzzy commitment scheme.

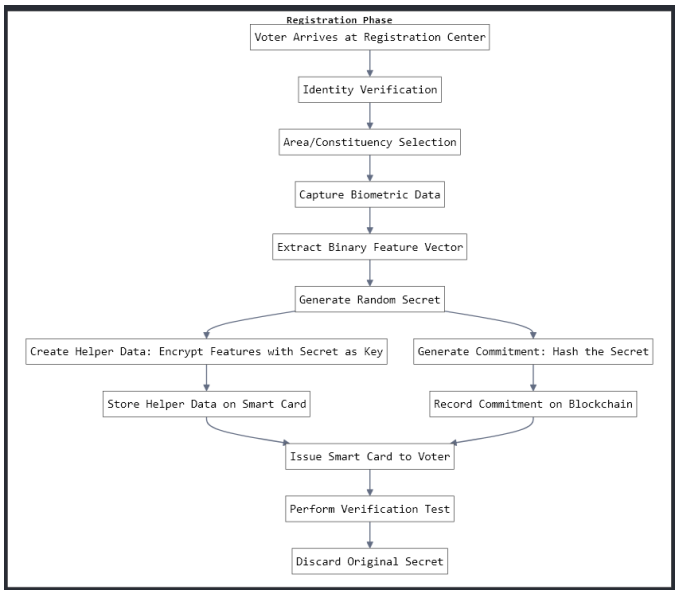

Fig. 2. Registration Phase of Voting using Fuzzy Commitments

3) Using threshold cryptosystems: In a threshold cryptosystem-based registration [7], the process would begin with setting up a network of registration authorities. Instead of having a single encryption key, the system would use a (t,n)-threshold scheme where t out of n authorities must cooperate to perform registration and verification operations. For example, we might have 5 total authorities with a threshold of 3, meaning any 3 authorities must cooperate to complete the registration process. This distributes trust and eliminates single points of failure in the registration system.

When a voter arrives for registration, their biometric data would be captured similar to the current system. However,

instead of using DNA encoding and AES encryption, the system would encrypt the biometric template using a public key associated with the threshold scheme. This public key would be known to all registration centers, but the corresponding private key would be split among the authorities using Shamir's Secret Sharing [6]. Each authority would then generate their share of the registration signature, and at least t authorities would need to contribute their shares to complete the registration process.

The appeal of this approach is that it prevents any single authority from having complete control over the registration process. Even if some authorities are compromised or unavailable, the system can continue to function as long as t honest authorities remain. Additionally, the blockchain would record which authorities participated in each registration, creating a transparent audit trail while maintaining voter privacy through the threshold encryption.

1) **System Initialization:** The system begins by generating a master key pair for the election, which serves as the cryptographic foundation for the entire voting process. The private key is then split into n shares using Shamir's Secret Sharing scheme [6], distributing security responsibility across multiple parties. These shares are distributed to designated authorities who will collectively manage the election security. Finally, the public key and verification parameters are published to enable transparent verification.
2) **Initial Voter Check-in:** A voter arrives at the registration center where an official verifies their physical identification to confirm their eligibility. Upon successful verification, the system assigns a unique voter ID that will be used throughout the process. The voter then selects their constituency or area which determines their ballot options and voting location.
3) **Biometric Capture:** The system captures the voter's biometric data, such as fingerprints or facial features [3], which is then processed and formatted into a standardized biometric template. This template is prepared specifically for threshold encryption to ensure it can be securely managed by multiple authorities. The system then generates a formal registration request containing the necessary voter information.
4) **Distributed Authority Processing:** The registration request is broadcast to all authorized election authorities. Each participating authority independently validates the request to ensure its legitimacy. A minimum threshold (t) of authorities must generate partial signatures to approve the registration, providing cryptographic redundancy [7]. These partial signatures are then mathematically combined into a complete registration credential that no single authority could produce alone.
5) **Smart Card Creation:** The voter's biometric template is encrypted with the election's public key and stored securely on a smart card that will be used for voter identification. The authority signatures validating the registration are also recorded on the card as proof of proper registration. Public parameters and verification data are included to enable authentication during voting without compromising security.
6) **Blockchain Recording:** A registration transaction is created and added to the blockchain, providing an immutable record of the registration [13]. This transaction includes identifiers for the authorities who participated in the registration process. An encrypted hash of the biometric template is recorded for future verification, and the area-based blockchain is updated to reflect the new voter registration.
7) **Verification Testing:** The system tests the authentication process using t randomly selected authorities to ensure the threshold mechanism works properly. The partial decryption process is verified to confirm that the correct number of authorities can successfully authenticate the voter. The biometric matching process is confirmed to work within acceptable parameters, and backup verification data is generated in case of technical issues during voting.
8) **Documentation and Completion:** The system issues a registration confirmation to the voter, documenting their successful enrollment. The participation of each authority in the registration process is formally recorded for auditing purposes. System audit logs are updated with the complete registration details, and the voter receives instructions on how to use their smart card during the election.

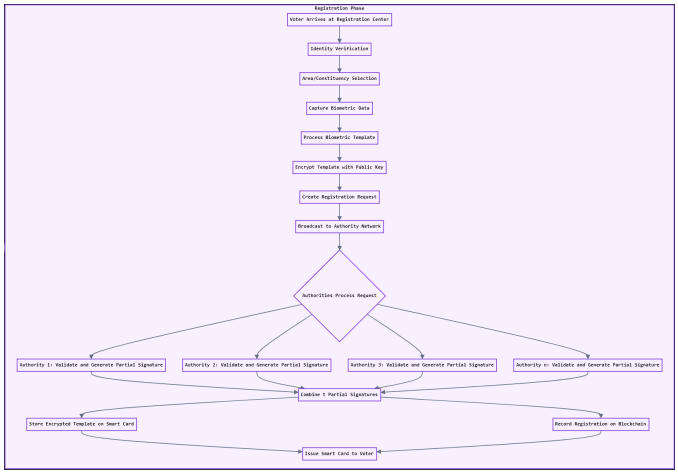

Fig. 3. Registration Phase of Voting using Threshold Encryption

B. Authentication

Authentication in electronic voting systems represents the critical process of verifying voter identity at the time of voting. This process must balance security requirements with accessibility and ease of use to ensure all eligible voters can participate. Most e-voting systems employ multi-factor authentication, requiring voters to provide something they know (such as a PIN or password), something they have (like a voter ID card or smart card), and/or something they

are (biometric identifiers). This layered approach significantly increases security compared to traditional paper-based voting methods. Upon arrival at a polling station, voters may present their identification documents or voter cards with embedded security features such as holograms, barcodes, or RFID chips. Poll workers or automated systems verify these credentials against the electoral database. More sophisticated systems incorporate real-time biometric verification, comparing the voter's live biometric data with templates stored during registration. Remote e-voting systems, including internet and mobile voting platforms, typically rely on digital certificates, one-time passwords sent via SMS, and other cryptographic protocols to authenticate voters. Some implementations use blockchain technology to create immutable records of authentication events while maintaining voter privacy. To prevent duplicate voting, many systems implement real-time voter check-in across all polling locations, marking voters as having cast their ballots immediately after authentication. This information is synchronized across the voting system to prevent multiple votes from the same individual. The authentication process must also incorporate accessibility features for voters with disabilities, such as alternative authentication methods for those unable to provide standard biometric data. Throughout the authentication process, maintaining voter privacy remains paramount, with strict separation between identity verification and the actual ballot to ensure vote secrecy.

1) Using standard symmetric key encryption: This approach provides stronger security guarantees through distributed trust, while also creating a more robust and fault-tolerant registration system. The involvement of multiple authorities in each registration adds complexity but significantly enhances the system's resistance to corruption or manipulation. The core of the authentication process starts when a voter arrives at the polling station with their smart card. This smart card contains their encrypted biometric template, which was stored during the registration process using AES encryption. The encryption process is particularly interesting because it uses a novel combination of DNA encoding/decoding and chaotic maps to generate the encryption key. This method first scrambles the biometric data using chaotic systems, then applies DNA encoding, creates a complement of that encoding, and finally decodes it again. The result is then hashed using SHA2 to create the encryption key.

When authentication begins, the system first generates a hash of the encrypted data stored on the smart card. This hash is compared against a hash stored in the database to ensure the card's data hasn't been tampered with. If the hashes match, the system retrieves the key associated with the voter's NID/SSN from the database. This key is then used to decrypt the biometric template stored on the smart card. Once decrypted, the system performs a live biometric verification - in this case using facial recognition [3] - comparing the decrypted template with the voter's current biometric reading.

The system adds an extra layer of security by storing multiple biometric templates for each voter. This redundancy serves two purposes: it allows voters to use alternative biometric methods if their primary biometric isn't working (for instance, if a fingerprint is damaged), and it provides a backup authentication method. All of these operations are recorded on the blockchain, creating an immutable audit trail of authentication attempts while maintaining voter privacy through encryption and hashing [13].

What makes this approach particularly robust is its combination of physical security (the smart card), biometric verification (facial recognition), and cryptographic security (the DNA-based encryption and blockchain validation). Each layer provides distinct security guarantees, making it significantly more difficult for an attacker to compromise the system - they would need to defeat multiple security mechanisms simultaneously to successfully impersonate a voter.

2) Using fuzzy commitments: When a voter arrives at the polling station with their smart card, the authentication would begin by reading the helper data and commitment stored on the card. Unlike the current system where encrypted templates must be decrypted exactly, fuzzy commitments allow for natural variations in biometric readings [4]. The system would capture a fresh biometric sample from the voter (their face) and convert it into a binary feature vector using the same feature extraction algorithm used during registration. This consistency in feature extraction is crucial for successful authentication.

The key operation happens when the system XORs the newly captured biometric features with the helper data stored on the smart card. This operation attempts to recover the original secret that was bound to the biometric during registration. Due to the properties of fuzzy commitments, even if the new biometric reading differs slightly from the original (as biometrics naturally do), the system can still recover the secret if the difference falls within the predefined error tolerance [5]. The recovered secret is then hashed, and this hash is compared with the commitment stored on the card. If they match (or are within acceptable tolerance), the voter is authenticated.

This approach offers significant advantages over the current system. It eliminates the need to store or transmit actual biometric templates, even in encrypted form, as the helper data reveals minimal information about the original biometric [4]. It naturally handles the variations inherent in biometric readings without compromising security. And importantly, it maintains a strong binding between the biometric and the authentication process while providing cryptographic guarantees that the system cannot be fooled by similar but unauthorized biometrics.

The blockchain would still record the authentication event, but instead of storing encrypted templates or hashes of templates, it would store commitments that provide similar security guarantees while being specifically designed for biometric authentication [13]. This creates a more robust system that maintains the security benefits of blockchain while addressing the unique challenges of biometric verification.

3) Using threshold encryption: If the e-voting system were to use threshold encryption for authentication [7] instead of the current approach, it would distribute trust across multiple authorities and eliminate single points of failure in the verifica-

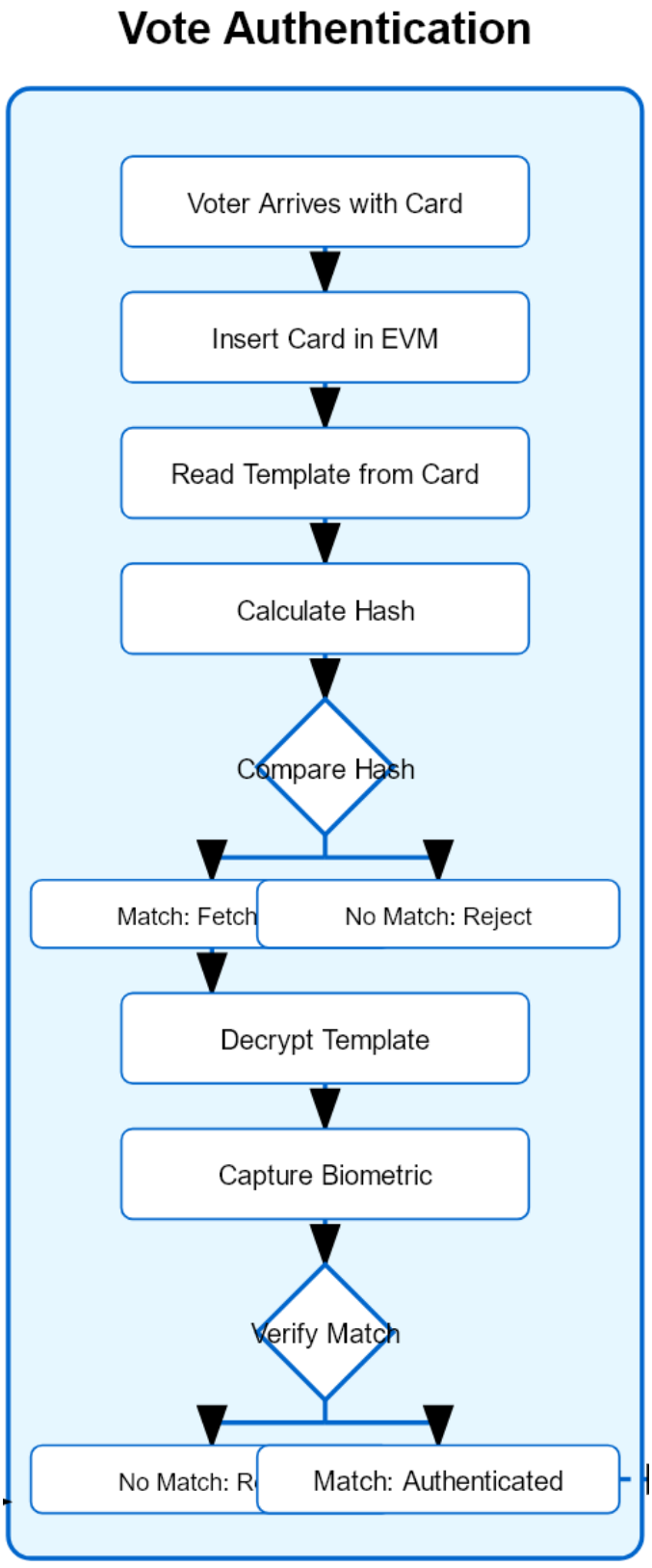

Fig. 4. Authentication Phase utilizing Blockchain with biometrics and smart cards

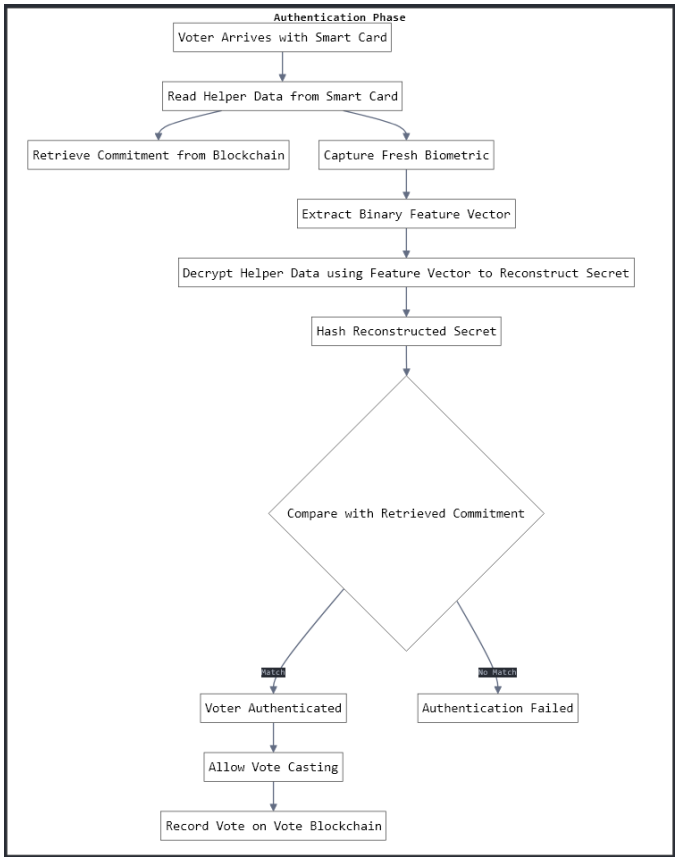

Fig. 5. Authentication Phase of Voting using Fuzzy Commitments

tion process. When a voter arrives at the polling station with their smart card, the authentication would begin differently. The smart card would contain the voter's biometric template encrypted with the election's public key. However, unlike the current system where a single authority can decrypt and verify the template, threshold encryption would require cooperation from multiple authorities [7].

The voting machine would first capture the voter's live biometric data and prepare it for comparison. It would then broadcast an authentication request to the distributed network of authentication authorities. To proceed with authentication, a minimum threshold of authorities (t out of n total) would need to participate. Each participating authority would use their private key share to generate a partial decryption of the encrypted template from the smart card [6]. These partial decryptions would then be combined at the voting machine using Lagrange interpolation to recover the fully decrypted template. Only after obtaining contributions from at least t authorities could the system complete the decryption and compare the stored template with the voter's live biometric.

This approach offers several security advantages. No single authority has enough information to decrypt the biometric template, preventing individual authorities from impersonating voters or leaking sensitive data. The system remains operational even if some authorities are unavailable or compro-

mised, as long as the threshold number of honest authorities remains accessible [7]. Additionally, the blockchain would record which authorities participated in each authentication, creating accountability without compromising voter privacy.

The threshold approach would also provide stronger auditability, as multiple independent parties must agree on each authentication decision. If an unusual pattern of authentications was occurring, multiple authorities would be able to detect and report it, rather than relying on a single authority to identify problems. This distribution of trust creates a more robust system that remains secure even if some components or authorities are compromised.

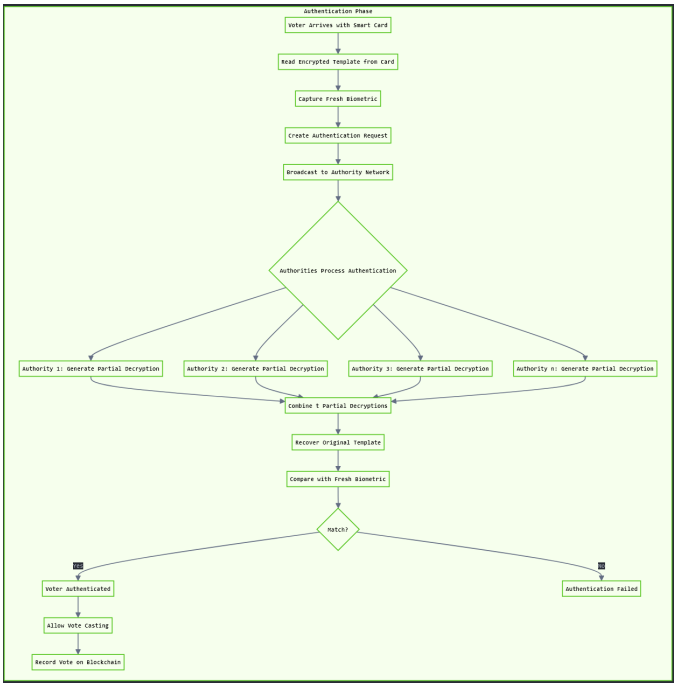

Fig. 6. Authentication Phase of Voting using Threshold Encryption

Below is a comprehensive comparison of the three approaches—AES Encryption (Current), Fuzzy Commitments, and Threshold Cryptography—across various security aspects in both registration and authentication phases.

Each approach offers distinct advantages: the current system provides simplicity and efficiency, fuzzy commitments offer superior biometric handling and template protection, while threshold cryptography provides the strongest protections against insider threats and authority compromises. The optimal choice depends on which security properties are most important for the specific e-voting implementation.

III. Existing Technologies

There are a lot of technologies that have been developed for voting however, all of them are not popular among governments and voters as each of them have them have their own pros and cons. Comparisons have been done between these technologies to better understand why some are in use and some have not been implemented for practical use yet.

A. Helios and Belenios: Comparison and Analysis

1) Helios Voting System: Helios is one of the first widely-deployed web-based, end-to-end verifiable (E2E) voting systems designed for low-coercion environments like university and organizational elections [8].

Key Features of Helios

- **Open Source:** Fully transparent code available for public scrutiny.
- **End-to-End Verifiability:** Voters can verify that their votes were correctly recorded and counted [11].
- **Web-Based Interface:** Accessible from any modern web browser, making it user-friendly and widely available.
- **Homomorphic Encryption:** Enables tallying of encrypted votes without the need to decrypt individual ballots, ensuring vote privacy.
- **Bulletin Board:** A public record of all encrypted votes, ensuring transparency and enabling independent verification.
- **Cast or Audit Mechanism:** Voters can choose to either cast their vote or audit the ballot to verify that encryption was correctly applied.

Security Aspects: Helios represents a pioneering approach to verifiable electronic voting with several notable security characteristics [8]. It offers universal verifiability where anyone can verify that all votes were properly counted, while also providing strong voter privacy through encryption that masks individual voting choices. The system employs cryptographic techniques to maintain end-to-end verification throughout the entire voting process. However, Helios has recognized limitations, particularly in its vulnerability to sophisticated coercion attacks where voters might be forced to reveal their voting choices [20]. It also faces challenges with potential client-side compromises that could undermine security, and the system requires trusted setup assumptions that introduce some security dependencies. Despite these constraints, Helios has established itself as a foundational protocol in academic research on secure electronic voting systems.

2) Belenios Voting System: Belenios is considered an evolution of Helios, developed by researchers at INRIA (France). It maintains core verification properties while addressing some of Helios's limitations [9].

Key Features of Belenios

- **Credential-Based Authentication:** Provides stronger voter authentication using individual credentials.
- **Improved Cryptographic Foundations:** Offers more formal security guarantees based on well-established cryptographic principles [21].
- **Flexible Tallying Methods:** Supports various election formats such as approval voting, ranked-choice voting, and more.
- **Modular Architecture:** Designed with modular components, allowing parts of the system to be replaced or upgraded independently.
- **Credential Authority:** Utilizes a separate entity to manage and distribute voter credentials, enhancing security

TABLE I
REGISTRATION PHASE SECURITY FEATURES

Feature	AES	Fuzzy Commitments [4], [5]	Threshold Crypto [6], [7]
Biometric Storage	Encrypted templates	No template stored	Encrypted with shared key
Key Management	Single key in DB	Random secrets	Private key split
Revocability	Re-encrypt template	New secrets, same data	Reissue with new authority
Trust Model	Centralized	Centralized, safer storage	Distributed trust
Fault Tolerance	Single point failure	Same	Tolerates authority failure
Template Privacy	Encrypted	Strong – no template stored	Encrypted + distributed
Hardware Req.	Standard sensors	Sensors + extractor	Infra for authorities
Speed	Fast	Fast	Slower (multi-party)

TABLE II
AUTHENTICATION PHASE SECURITY FEATURES

Feature	AES Encryption	Fuzzy Commitments [4], [5]	Threshold Cryptography [6], [7]
Biometric Matching	Exact match after decryption	Tolerates natural variations	Exact match after distributed decryption
Auth. Control	Single authority can authenticate	Single authority can authenticate	Requires t authorities to cooperate
Impersonation Resistance	Compromise of encryption key	Matching biometric within tolerance	Compromise of t authorities
Template Exposure Risk	Exists in memory during verification	No exposure; match in commitment space	Exists after combining partial decryptions
Offline Authentication	Possible with local keys	Possible with stored commitments	Not possible; needs communication
Auth. Speed	Fast decryption	Fast XOR and hashing	Slower; needs communication
Audit Trail	Blockchain logs auth events	Blockchain logs verifications	Blockchain logs authority participation
Backup Auth.	Alt. biometrics	Handles variation naturally	Any t of n authorities can proceed

and trust.

- **Support for Mixnets:** Offers mixnet-based tallying as an alternative to homomorphic tallying, improving vote anonymity [14].

a) Security Aspects: Belenios offers enhanced security through several significant improvements to electronic voting systems [9]. It provides stronger resistance to ballot stuffing through its sophisticated credential management system, while simultaneously implementing better safeguards against various server-side attacks that have plagued earlier e-voting platforms. The system is built upon more rigorous cryptographic proofs of its security properties, giving it a more solid theoretical foundation than many alternatives [21]. Despite these advances, Belenios remains vulnerable to client-side attacks, which continue to represent a significant challenge in the e-voting security landscape. While the system has made notable progress in addressing coercion resistance, this protection remains incomplete, representing an area where further development would be beneficial.

b) Key Differences in Security Properties:

- **Ballot Stuffing:** Belenios's credential system makes it more difficult for malicious servers to inject fraudulent votes [9].
- **Authentication:** Belenios enhances security by separating the credential authority from the voting server [9].
- **Verification:** Both systems support individual vote verification, but Belenios strengthens the verification chain through its credential system [9].
- **Trust Distribution:** Belenios distributes trust across multiple entities, minimizing single points of failure [9].
- **Cryptographic Foundation:** Belenios is based on more rigorous and formally proven cryptographic security models [9], [21].

Neither system fully solves the issues of client-side security and sophisticated coercion attacks, which remain challenges for remote electronic voting in general [1], [20].

B. ElectionGuard and VotingWorks: Comparison and Analysis

1) ElectionGuard: ElectionGuard is an open-source software development kit (SDK) created by Microsoft to make voting more secure, verifiable, and transparent [10].

a) Key Features of ElectionGuard:

- **Open Source:** Freely available on GitHub under the MIT license, promoting transparency and collaboration.
- **End-to-End Verifiability:** Voters can confirm that their votes were accurately recorded and counted [11].
- **Homomorphic Encryption:** Allows for tallying of encrypted votes without decrypting individual ballots, preserving voter privacy.
- **Hardware Agnostic:** Designed to integrate with existing voting systems and hardware, offering flexibility in deployment.
- **Tracking Codes:** Provides each voter with a unique tracking code to verify their vote's inclusion in the final tally.
- **Verifiable Tallying:** Enables independent verification of the election results by any observer.
- **Risk-Limiting Audit Support:** Facilitates statistical audits to verify the accuracy of election outcomes.

b) Security Aspects: ElectionGuard incorporates several robust security features that significantly enhance the integrity of electronic voting systems [10]. It employs advanced homomorphic encryption, specifically the ElGamal cryptosystem, combined with threshold decryption to protect ballot contents while enabling verifiable tallying [7]. The system's multi-trustee design effectively prevents single points of failure

TABLE III
TABULAR COMPARISON OF HELIOS AND BELENIOS

Feature/Aspect	Helios [8]	Belenios [9]
Core Design	First-gen E2E system	Evolution with enhanced security
Voting Credentials	Email-based, optional password	Blinded tokens from credential authority
Ballot Casting	Cast-or-audit paradigm	Similar, improved verification
Tallying Method	Homomorphic encryption	Homomorphic + Mixnet options
Authentication	Basic (email/password)	Advanced credential management
Vote Verification	Voter-verifiable	Enhanced with credentials
Coercion Resistance	Limited	Improved, not complete
Ballot Secrecy	Crypto-based; browser risks	Enhanced via credential separation
Trust Model	Trustees decrypt	Credential authority + trustees
User Experience	Simple, streamlined	Extra steps for credentials
Deployment Complexity	Relatively simple	More complex
Formal Security Proofs	Limited	More comprehensive [21]
Implementation	JavaScript, Python	OCaml + formal verification
Suitable Elections	Low-stakes, org. use	Medium-stakes, stronger security

by distributing cryptographic authority across multiple independent parties. Every ballot cast through ElectionGuard is accompanied by strong cryptographic proofs that ensure ballot correctness without revealing voter choices. The implementation of zero-knowledge proofs throughout the system provides mathematical guarantees of voter privacy while still allowing for comprehensive verification [16]. While ElectionGuard was designed with formal security verification principles to mathematically prove its cryptographic properties, it's important to note that the system cannot prevent all forms of physical tampering with voting machines, which remains a challenge for any electronic voting solution.

2) VotingWorks: VotingWorks is a non-profit organization that builds open-source voting equipment and software focused on security, usability, and cost-effectiveness.

Key Features of VotingWorks

- **Non-Profit Structure:** Mission-driven rather than profit-driven.
- **Open Source Hardware and Software:** Provides complete transparency, allowing public review and verification of the system's integrity.
- **Paper-Based:** Creates physical paper ballots for all votes to ensure verifiability and auditability. (Similar principle to Prêt à Voter [17]–[19])
- **Risk-Limiting Audits:** Includes built-in tools for statistical verification of election outcomes.
- **Simplified User Interface:** Designed for accessibility and ease of use for voters and election workers alike.
- **Low Cost:** Offers a more affordable alternative compared to traditional commercial voting systems.
- **Modular Design:** Enables individual components to be updated or replaced without overhauling the entire system.

Security Aspects:

VotingWorks implements a comprehensive security approach centered on verifiability and transparency in electronic voting. Its use of paper ballots provides a crucial physical audit trail that can be manually verified independently of any software systems. The platform's commitment to open-source development means all code is available for public scrutiny, allowing independent security researchers to identify and address potential vulnerabilities. This transparency is complemented by support for rigorous post-election auditing protocols such as risk-limiting audits that statistically verify results. VotingWorks employs a deliberately simplified design philosophy that reduces the overall attack surface, making the system easier to secure and analyze. To protect against network-based threats, the platform utilizes air-gapped tallying systems that physically isolate vote counting from external networks. While these technical measures provide significant protection, VotingWorks acknowledges that physical security ultimately relies on proper election procedures implemented by election officials.

a) Key Differences in Security Properties:

- **Security Foundation:** ElectionGuard relies primarily on cryptographic security [10], while VotingWorks emphasizes physical paper ballots as the security foundation.
- **Verification Approach:** ElectionGuard provides mathematical verification through encryption [11], while VotingWorks focuses on paper-based verification with statistical auditing.
- **Integration Model:** ElectionGuard is designed to enhance existing systems, while VotingWorks offers complete end-to-end voting systems.
- **Trust Distribution:** ElectionGuard distributes trust through cryptographic means and multiple trustees [7], while VotingWorks relies more on procedural controls and physical evidence.
- **Modernization Approach:** ElectionGuard brings cryptographic verification to voting, while VotingWorks modernizes the traditional paper-ballot approach.
- **Transparency:** Both are open source, but they emphasize different aspects — ElectionGuard focuses on cryptographic transparency, while VotingWorks emphasizes hardware and software transparency.

Both systems represent different philosophies in election security - ElectionGuard bringing advanced cryptography to make digital voting more secure, while VotingWorks focuses on making traditional paper-based voting more efficient and verifiable.

TABLE IV
COMPARISON OF ELECTIONGUARD AND VOTINGWORKS

Feature/Aspect	ElectionGuard [10]	VotingWorks
Core Design	Crypto SDK for verifiable elections	Full voting system: hardware + software
Organization Type	Microsoft-initiated (tech company)	Non-profit organization
Primary Focus	E2E verifiability via cryptography [11]	Usable, secure paper-based voting
Software Model	SDK for integration	Complete software stack
Hardware Approach	Vendor-agnostic integration	Custom hardware and peripherals
Ballot Format	Paper/digital; crypto-verifiable	Paper-based with digital marking
Verification Method	Math proofs, encryption	Paper trail + risk-limiting audits
Tallying Method	Homomorphic + threshold decrypt.	Scan paper ballots with audits
Trust Model	Crypto verification, trustees	Paper checks + procedural control
Language	C, C#, Python, TypeScript	Python and JavaScript
Deployment Status	SDK available	Deployed in jurisdictions
Cost Model	Free SDK; varied costs	Lower cost than commercial vendors
Auditability	Crypto + risk-limiting audits	Paper audit with stats
Accessibility	Varies by integration	High focus on usability/access

C. Open.Vote and Blockchain Voting: Comparison and Analysis

1) Open Vote System: Open.Vote refers to a class of open-source voting platforms focused on transparency and participation. While not a specific proprietary system, these platforms share common principles and approaches.

a) Key Features of Open.Vote Systems:

- **Open Source:** Complete transparency of code and processes.
- **Community Governance:** Often managed by non-profits or community organizations.
- **Flexible Implementation:** Can be adapted for various election types.
- **Participation Focus:** Emphasis on increasing voter engagement.
- **Auditable Trail:** Records of election procedures and results.
- **Accessible Design:** Attention to usability and accessibility standards.
- **Transparent Counting:** Open algorithms for vote tabulation.

Security Aspects: Open.vote takes an approach to security that emphasizes transparency and public code review, making its implementation visible and accessible to independent verification. Rather than employing complex cryptographic techniques, the system relies primarily on conventional encryption methods to protect data integrity and confidentiality. This approach means that Open.vote's security is heavily dependent on proper implementation and deployment practices by election administrators. The platform typically employs standard authentication methods for voter verification, which can be effective but may not offer the advanced protections of specialized voting protocols. Like other web applications, Open.vote remains vulnerable to common web security threats including injection attacks, cross-site scripting, and server-side vulnerabilities. To address these limitations, the security model often incorporates procedural controls and administrative safeguards that complement the technical protections to maintain the overall integrity of the voting process.

2) Blockchain Voting: Blockchain voting leverages distributed ledger technology to create immutable records of votes and election processes [12], [13].

Key Features of Blockchain Voting

- **Distributed Ledger:** Decentralized record of all transactions [13].
- **Immutability:** Once recorded, votes cannot be altered.
- **Consensus Mechanisms:** Multiple parties verify election integrity.
- **Smart Contracts:** Automated election rules and procedures [12].
- **Cryptographic Verification:** Mathematical proofs of vote integrity.
- **Tokenization:** Votes represented as digital assets.
- **Public Verifiability:** Anyone can verify the blockchain's integrity.

a) Security Aspects: Blockchain voting systems employ cryptographic security founded on consensus algorithms that distribute trust across multiple nodes, providing robust protection against single points of failure in the voting infrastructure [13]. However, some implementations remain vulnerable to 51% attacks where a majority of computing power could potentially manipulate results, particularly in smaller or permissioned blockchains. These systems present significant key management challenges for voters, who must securely maintain private keys without technical support infrastructure. When implemented on public blockchains, potential privacy concerns arise as transaction patterns may reveal voting behavior despite encryption [12]. The overall security of blockchain voting fundamentally depends on implementation quality, with security guarantees varying widely between different projects and approaches. Additionally, blockchain voting systems offer varying levels of coercion resistance based on their specific design choices, with some systems prioritizing verifiability at the potential expense of vote privacy [12], [20].

b) Key Differences in Security Properties:

- **Security Foundation:** Open.Vote relies on conventional security practices and transparency, while blockchain voting uses cryptographic consensus mechanisms [13].

TABLE V
COMPARISON OF OPEN.VOTE SYSTEMS AND BLOCKCHAIN VOTING

Feature/Aspect	Open.Vote Systems	Blockchain Voting [12], [13]
Underlying Tech	Encrypted conventional databases	Distributed ledger technology
Verification Method	Procedural audits, open-source code	Crypto consensus, immutability
Trust Model	Trust in org/code implementation	Distributed trust via nodes
Transparency	Code/process transparency	Algorithm + transaction transparency
Privacy Approach	Standard DB security	Pseudonymity or ZK proofs [16]
Centralization	Centralized servers	Decentralized nodes
Immutability	Based on audit logs	Built-in blockchain structure
Cost	Lower implementation cost	Higher infra/computational cost
Environmental Impact	Normal energy usage	High (esp. PoW mechanisms)
Scalability	More scalable for large votes	Throughput limitations
Auditability	Manual + code review	Automatic via consensus
Voter Experience	Simple interfaces	May need wallets or key mgmt
Coercion Resistance	Depends on design	Varies by implementation [20]
Key Management	Centralized key control	Distributed/user-managed keys
Maturity	Practically established	Emerging with limited deployment [13]

- **Threat Model:** Open.Vote systems typically defend against internal corruption through transparency, while blockchain systems focus on distributed verification to prevent manipulation.
- **Failure Modes:** Open.Vote systems may have centralized points of failure, while blockchain systems risk consensus attacks or smart contract vulnerabilities [12].
- **Transparency vs. Privacy:** Open.Vote emphasizes process transparency, while blockchain must balance transaction transparency with vote privacy [12].
- **Verification Approach:** Open.Vote verification is typically human-auditable, while blockchain provides automated cryptographic verification [13].
- **Trust Requirements:** Open.Vote requires trust in the implementing organization, while blockchain distributes trust across multiple participants [13].
- **Long-term Security:** Open.Vote systems can be updated more easily for security, while blockchain's immutability makes security updates challenging after deployment.

c) Implementation Challenges: Both approaches face significant implementation challenges:

- **Open.Vote Systems:** Ensuring proper security controls, preventing insider threats, and maintaining public confidence.
- **Blockchain Voting:** Addressing scalability limitations, managing user keys securely, and balancing transparency with privacy [12], [13].

Neither approach completely solves all electronic voting security challenges [1], and hybrid approaches that combine elements of both may provide more comprehensive solutions. The most appropriate system depends on the specific election context, threat model, and resources available.

IV. NOVEL APPROACH: CONCEPTUAL DESIGN

Integrating ElectionGuard with Threshold Authentication: ElectionGuard already uses threshold encryption in its core design [7], [10], which creates an excellent foundation for integrating it with a threshold authentication protocol. Let me walk through how these systems could be effectively combined while preserving their security properties.

1) Understanding the Components: ElectionGuard uses a threshold ElGamal encryption scheme where multiple trustees share the ability to decrypt all encrypted values related to the election [10]. No single trustee can decrypt votes, requiring cooperation of a threshold number of trustees (e.g., k of n) to complete the tallying process [7].

2) ElectionGuard's Threshold Encryption: Why It Matters: The Core Concept of Threshold Encryption ElectionGuard uses a threshold ElGamal encryption scheme as a foundational security mechanism [10]. Let us elaborate on how this works and, importantly, why this approach is critical for election security.

a) Technical Process: In ElectionGuard's threshold encryption system, the key generation process distributes security responsibility across multiple parties. Instead of generating a single private key, the system uses a distributed key generation protocol where n trustees each receive a unique key share [7].

For encryption, all ballots are encrypted with a single public key that corresponds to all the trustees' shares combined. This allows anyone to encrypt ballots without needing access to the decryption keys.

Decryption requires cooperation. To decrypt the election results, at least k out of n trustees must participate by applying their key shares to the encrypted tally [6], [7]. Any smaller number of trustees cannot decrypt anything.

The reconstruction phase combines these partial decryptions from k trustees to reveal the final election results while maintaining the secrecy of individual ballots.

A. Why This Matters: Security Rationale

The threshold approach addresses several critical security concerns:

1) Defense Against Collusion: In a single-key system, whoever holds the key can unilaterally decrypt individual votes, compromising ballot secrecy. This creates a significant security vulnerability and concentration of power.

By requiring k trustees to cooperate for decryption, ElectionGuard ensures that no small group can decrypt ballots

on their own [7]. Even if some trustees are compromised or collude with adversaries, as long as their number is less than k, ballot secrecy is maintained. For example, in a system with 7 trustees where 4 are needed for decryption (k=4, n=7), adversaries would need to compromise 4 trustees to break encryption security—a much higher bar than compromising a single entity.

2) Trustee Independence and Diversity: The system's security depends on trustees being selected from different stakeholders with opposing interests. These trustees might include representatives from different political parties, election officials from different jurisdictions, independent election monitoring organizations, technical experts from different institutions, and representatives from different branches of government.

This diversity makes large-scale collusion impractical because these entities have competing interests and would expose each other's malfeasance. A Republican trustee would be incentivized to report if Democratic trustees were attempting to manipulate the system, and vice versa. This creates a natural check-and-balance within the cryptographic system itself.

B. Balance Between Security and Availability

The threshold approach creates a carefully designed balance between security and system availability. Setting k too low (e.g., k=2 in a system with n=10) would make collusion easier and reduce security. Setting k too high (e.g., k=10 in a system with n=10) would mean any single trustee could prevent decryption by withholding their share, threatening availability.

By tuning these parameters appropriately (e.g., k=7 in a system with n=10), ElectionGuard creates a system that is both secure against reasonable collusion scenarios and resilient against trustees who might become unavailable or attempt to disrupt the process [7]. This parametric flexibility allows the system to be configured based on the specific trust model of each election.

C. Separation of Duties

This approach enforces a critical security principle: separation of duties. No single entity can both authorize voters (authentication) and see how they voted (decryption). This separation is fundamental to maintaining ballot secrecy in digital voting systems [20]. It mirrors traditional election controls where different people verify voter eligibility and count ballots.

D. Practical Implementation

In practice, trustee key shares are typically stored in hardware security modules (HSMs) that prevent the extraction of the key material. The trustees participate in formal, often public ceremonies for key generation before the election and decryption after polls close.

These ceremonies are structured, documented processes that can be observed by the public, media, and other stakeholders to ensure compliance with procedures. They create tangible, observable security events rather than relying on abstract trust.

The parameters k and n can be adjusted based on the security requirements and trust model of the specific election [7]. For a national election, you might see higher values (e.g., k=7, n=10) while a smaller organization might use lower values (e.g., k=3, n=5).

This threshold cryptographic approach is one of the most important innovations in election security [7], providing strong cryptographic guarantees rather than depending solely on procedural controls or trust in individuals. It transforms security from a matter of trust into a matter of mathematical certainty, where breaking the system requires an impractically large conspiracy across opposing interests. Threshold Authentication Integration Considerations

For integrating a threshold authentication protocol, we need to consider:

- The existing ElectionGuard architecture [10]
- How voter authentication happens before vote casting
- Where threshold authentication would fit in this workflow
- How to maintain security properties of both systems

E. Conceptual Integration Design

a) Phase 1: Voter Registration and Credential Distribution:

1) Threshold Credential Authority Setup:: A set of m credential authorities is established for the system. Each authority independently generates a share of a master secret key using a distributed key generation protocol [7]. This distributed approach ensures no single authority possesses the complete master key. The system requires that at least t authorities (where $t \leq m$) must cooperate to create valid credentials, providing security through distribution of trust.

2) Voter Registration Process: During registration, the voter's identity is verified through traditional authentication means such as government-issued ID documents, biometrics [3], or other established verification processes. Following verification, each credential authority independently creates a partial credential for the voter using its secret key share. The voter then combines these partial credentials mathematically to form their complete voting credential. This approach ensures the full credential is never known to any single authority, preventing any one entity from tracking voter participation.

3) Credential Privacy: To preserve privacy, voters apply a cryptographic blinding operation to their credential. This mathematical transformation changes the appearance of the credential while preserving its validity properties. The blinding effectively prevents credential authorities from tracking specific voters through their credentials, as the blinded version used during voting cannot be linked back to the original issued credential [14].

Two ways to blinding the credentials

a) Approach 1: Blind Signature Model: In the blind signature approach [14], the privacy protection happens during credential issuance. The voter applies a blinding transformation to their identity or credential request before submitting it to the credential authorities. Each authority signs this blinded value without seeing the actual identity or credential being

authorized. The voter can then unblind the signature to obtain a valid credential that cannot be linked back to their identity by the authorities. When later casting a vote, the voter can directly present this credential without additional privacy measures, since the credential authorities never saw the unblinded version and thus cannot connect it to a specific voter. The credential itself becomes a private, anonymous token that proves eligibility without revealing identity. This approach simplifies the voting process since the voter can simply present their credential directly. The computational complexity is front-loaded to the registration phase rather than the voting phase.

b) Approach 2: Zero-Knowledge Proof Model: In the zero-knowledge approach [16], the credential authorities issue credentials directly, seeing the full credential value and knowing which voter received which credential. However, when voting, the voter uses zero-knowledge proofs to demonstrate possession of a valid credential without revealing which specific credential they're using. This approach shifts the privacy protection to the voting phase. The voter creates a cryptographic proof that they possess a credential that was properly issued by the authorities, without disclosing which specific credential it is. This prevents the authorities from linking the vote back to the voter's identity, even though they know all the credentials they issued. This method typically requires more complex computation during the voting process but can simplify the registration phase.

c) Phase 2: Integration with ElectionGuard:

4) Authentication During Ballot Creation: When a voter initiates the voting process in ElectionGuard [10], they first authenticate using their threshold credential. Rather than revealing the credential directly, the voter produces a zero-knowledge proof demonstrating possession of a valid credential without disclosing the credential itself [16]. This cryptographic proof is verified against the public parameters of the credential system, confirming the voter's eligibility without compromising privacy.

5) Ballot Preparation with ElectionGuard: Once authenticated, ElectionGuard's existing ballot encryption process begins [10]. The voter's choices are encrypted using the election's public key, creating a mathematical representation that hides the actual vote selections. Simultaneously, zero-knowledge proofs are generated to prove the ballot is well-formed—containing valid selections without revealing what those selections are [16].

6) Credential Binding: When a voter casts their ballot, two critical cryptographic elements exist separately: the encrypted ballot containing their vote choices and their credential proof establishing their eligibility to vote. These must be bound together in a way that prevents various attacks while preserving anonymity.

The binding process typically employs a cryptographic technique called a zero-knowledge proof of knowledge [16]. In particular, the system would use what cryptographers call a "proof of knowledge of a signature on a committed value." Here's how this works: First, the voter encrypts their ballot selections using ElectionGuard's homomorphic encryption with the election public key [10]. This creates a ciphertext that hides the actual vote choices.

Next, the voter creates a cryptographic commitment to their credential. A commitment in cryptography is similar to placing something in a sealed, opaque envelope—it hides the value (the credential) while preventing it from being changed later.

The voter then constructs a zero-knowledge proof that demonstrates several critical facts simultaneously: that they possess a valid credential properly signed by the threshold of authorities, that the committed value is that credential, and that this credential has not been used before in this election [16].

This proof is constructed using mathematical operations on elliptic curves, typically employing a technique like Schnorr proofs or sigma protocols [16] that can be made non-interactive using the Fiat-Shamir heuristic.

The binding itself consists of including the hash of the encrypted ballot within this zero-knowledge proof. This creates a cryptographic link between the specific ballot and the credential proof without revealing the credential.

The system validates this proof when the ballot is submitted. If valid, it records a cryptographic hash of the credential (or a token derived from it) in a public list of "used credentials" without recording which ballot it was used with. This prevents the same credential from being used twice.

a) Phase 3: Vote Casting and Verification:

7) Ballot Submission:: The encrypted ballot with its attached credential proof is submitted to the voting system. Upon receipt, the system verifies the credential proof is valid by checking its cryptographic properties against the system's public parameters. Importantly, this verification happens without the system learning the credential itself. The system also checks that this credential hasn't been used before in this election, preventing double voting.

a) Digital Submission via Personal Devices: If voters use their own devices (computers, smartphones, tablets) to prepare and submit ballots:

Many voters from lower socioeconomic backgrounds may lack reliable internet access or appropriate devices. The digital divide remains significant in many communities, particularly affecting rural, elderly, low-income, and some minority populations.

Voters with disabilities might struggle with inaccessible interfaces. Screen readers, voice navigation, and other assistive technologies must be fully supported, which is often overlooked in implementation.

Language minorities may face barriers if the interface isn't available in their primary language or uses technical terminology that doesn't translate well.

b) QR Code Systems: QR codes offer a bridge between digital encryption and physical voting (similar ideas explored in Prêt à Voter [17]):

This approach requires voters to either print QR codes at home (raising the same access issues as personal devices) or use on-site equipment that generates them.

Voters with visual impairments may be unable to verify what's encoded in a QR code, creating a trust gap in the verification process.

Elderly voters or those unfamiliar with technology might find the concept confusing or intimidating, potentially discouraging participation.

c) Kiosk-Based Systems at Polling Places: Dedicated voting machines at polling locations:

While potentially more accessible, these still require digital literacy to navigate complex authentication and encryption interfaces. Wait times could increase significantly due to the additional steps involved, disproportionately affecting hourly workers who cannot afford extended time away from work. The complexity of the interface might require additional poll worker assistance, potentially compromising the privacy of voters who need help.

d) Broader Systemic Barriers: The introduction of cryptographic credentials and proofs creates systemic barriers beyond the physical submission method:

e) Complexity Barrier:: The cognitive load of understanding cryptographic concepts, managing credentials, and verifying ballots may be prohibitive for voters with limited education, cognitive disabilities, or language barriers.

f) Trust Gap:: Communities with historical reasons to distrust government or technology may be skeptical of a system they cannot fully understand, potentially reducing participation.

g) Geographic Limitations:: Rural voters with limited connectivity may struggle with systems that require internet access for any part of the credential management or verification process.

h) Documentation Requirements:: If credential issuance requires specific identification documents, this could disproportionately impact communities that face barriers to obtaining such documentation, including indigenous populations, transgender voters, housing-insecure individuals, and certain immigrant communities.

i) Mitigating These Barriers: To address these concerns, several approaches should be considered:

j) Multiple Submission Pathways:: Offer various methods for credential management and ballot submission, ensuring no single approach becomes a bottleneck for participation.

k) Assistive Technology Integration:: Design systems from the ground up to work with screen readers, voice controls, and other assistive technologies.

l) Plain Language Design:: Present complex cryptographic concepts using simple, clear language that's accessible across educational levels and available in multiple languages.

m) Proxy Assistance Protocols:: Develop secure methods for voters who need assistance to receive help without compromising their vote privacy or security.

n) Extended Voting Periods:: Provide longer voting windows to accommodate those who need more time to navigate the technical requirements.

o) Community-Based Training:: Partner with trusted community organizations to provide hands-on training and support, particularly in marginalized communities.

The implementation of advanced cryptographic voting systems must carefully balance security enhancements against the fundamental democratic principle of broad and equal access to the ballot. Technology that increases security while decreasing accessibility may ultimately undermine the democratic process it aims to protect.

8) Ballot Tracking: ElectionGuard's existing tracking code system is extended to include verification that the credential was accepted [10]. This allows voters to confirm their participation was recorded correctly. Voters can verify their ballot was included in the final tally without revealing their specific vote choices, maintaining the secret ballot principle while providing transparency [11].

9) Tallying Process: ElectionGuard's tallying process remains largely unchanged in this integrated system [10]. The threshold trustees cooperate to decrypt and tally the results using their shares of the election secret key [7]. Before tallying, the system verifies all ballots came from valid credentials, ensuring only legitimate votes are counted. This separation between authentication and tallying authorities provides important security boundaries.

F. e. Technical Implementation Details

a) Cryptographic Construction: To integrate these systems, we would use:

1) Threshold BLS Signatures for Authentication:: To integrate these systems effectively, we would use Threshold BLS Signatures for authentication [15], where credential authorities generate BLS signature shares and voters combine these signature shares to create their complete credential. This approach enables efficient verification and compact proofs.

2) Zero-Knowledge Proofs for Credential Verification:: For credential verification, the system employs zero-knowledge proofs such as Schnorr protocols or zk-SNARKs [16] that demonstrate possession of valid credentials without revealing the credentials themselves. These cryptographic techniques reveal nothing about the credential while proving its validity.

3) Double-Encryption for Privacy:: Privacy is further enhanced through double-encryption approaches. The ballot itself is encrypted with ElectionGuard's election key [10], while the credential proof remains separate but cryptographically linked. This separation prevents correlation between voter identity and specific ballot contents.

G. Data Flow Architecture

1) System Setup Phase: The election authority establishes system parameters by selecting appropriate cryptographic primitives. For ballot encryption, the system uses ElGamal encryption over an elliptic curve group G with generator g. For credential signatures, the system uses BLS signatures over a pairing-friendly curve [15]. The parameters include group descriptions, hash functions, and other public values needed by all participants.

For the trustee setup, n trustees participate in a distributed key generation protocol [7]. Each trustee i generates a secret key share sk_i and a corresponding public key share pk_i. The election public key PK is computed as the product of all trustee public key shares:

$$PK = pk_1 \cdot pk_2 \cdot \ldots \cdot pk_n$$

The secret key SK is implicitly defined as:

$$SK = sk_1 + sk_2 + \ldots + sk_n$$

though no party ever computes it directly [6].

For credential authorities, m separate entities are designated. Each credential authority j generates a secret signing key ca_j and publishes the corresponding verification key vk_j. These keys are used for the BLS signature scheme [15]. The system defines a threshold t where at least t out of m authorities must participate to create a valid credential.

All public parameters, verification keys, and the election public key are published on a public bulletin board accessible to all parties.

2) Voter Registration: The voter first establishes their identity through traditional verification methods at a registration authority. After identity confirmation, the voter is assigned a unique identifier ID.

The voter generates a random blinding factor r and computes a blinded version of their identity:

$$\text{BlindedID} = H(ID)^r$$

where H is a hash function mapping identities to points on the BLS curve. This uses the concept of blind signatures [14].

The voter sends BlindedID to each credential authority j, along with proof of registration from the registration authority. Each authority verifies this proof, then computes a partial credential as:

$$\text{PartialCred}_j = \text{BlindedID}^{ca_j}$$

which is a BLS partial signature on the blinded identity [15].

Each authority returns PartialCred_j to the voter, who verifies its correctness using the authority's public verification key vk_j by checking the pairing equation:

$$e(\text{PartialCred}_j, g) = e(\text{BlindedID}, vk_j)$$

After collecting at least t valid partial credentials, the voter combines them using Lagrange interpolation to create a complete blinded credential:

$$\text{BlindedCred} = \prod \text{PartialCred}_j^{\lambda_j}$$

where λ_j are the Lagrange coefficients [6].

The voter then unblinds this credential by computing:

$$\text{Cred} = \text{BlindedCred}^{1/r}$$

resulting in a BLS signature on $H(ID)$ under the collective authority signing key. This credential serves as the voter's anonymous voting token.

3) Vote Creation and Encryption: When ready to vote, the voter accesses the ElectionGuard system and makes their ballot selections through the user interface [10]. Let $\mathbf{m}$ be the vector representing these selections.

The system encrypts each selection m_i using ElGamal encryption with the election public key PK:

$$\text{Enc}(m_i) = (g^{r_i},\ m_i \cdot PK^{r_i})$$

where r_i is a random value chosen for each selection.

For each encrypted selection, the system generates a zero-knowledge proof (ZKP) demonstrating that the encrypted value is a valid ballot option (typically 0 or 1 for each option) without revealing which value was encrypted [16]. These are Chaum-Pedersen proofs adapted for the ElGamal encryption system.

Together, these encrypted selections and their associated ZKPs form the encrypted ballot EB.

4) Credential Binding and Ballot Submission: The voter creates a Pedersen commitment to their credential:

$$\text{CommCred} = g_1^s \cdot h_1^{\text{Cred}}$$

where g_1 and h_1 are distinct generators, and s is a random blinding factor.

The voter constructs a zero-knowledge proof π_{cred} demonstrating: (1) they know a credential Cred and randomness s that opens commitment CommCred, and (2) Cred is a valid BLS signature from the credential authorities on some identity that was properly registered [16].

This proof uses Schnorr-type protocols adapted for BLS signatures [15]. The challenge in this proof incorporates $H(EB)$ – the hash of the encrypted ballot – binding the credential proof to this specific ballot.

The voter submits the package $(EB, \text{CommCred}, \pi_{\text{cred}})$ to the voting system. The system verifies all ZKPs in EB and verifies π_{cred} against CommCred and the system parameters.

If the proofs verify correctly, the system checks that CommCred (or a deterministic transformation of it) does not appear in the list of already-used credentials. If this check passes, the system adds this commitment to the list of used credentials and accepts the ballot.

The system generates a tracking code:

$$TC = H(EB \| \text{CommCred})$$

and returns it to the voter as a receipt. The ballot and credential commitment are published on the public bulletin board, enabling universal verification [11].

5) Verification and Tallying: Any observer can verify that each ballot on the bulletin board contains valid ZKPs for the encrypted selections and a valid credential proof [11]. This verification confirms that all accepted ballots came from eligible voters who each voted only once.

After the voting period ends, the trustees commence the tallying procedure. First, the system homomorphically combines all encrypted ballots to produce encrypted tallies for each option.

Each trustee i generates a partial decryption of the encrypted tallies using their secret key share sk_i, along with a proof that they performed this decryption correctly [7].

These partial decryptions and proofs are published on the bulletin board. Anyone can verify these proofs against the trustees' public key shares.

The final tally is computed by combining the partial decryptions from at least k trustees [6]. The result reveals the election outcome without decrypting individual ballots.

6) Voter Verification: A voter can verify their ballot was included in the tally by locating their tracking code TC on the public bulletin board and confirming the associated encrypted ballot matches their selections [11].

For each selection, the voter can recompute the encryption using the randomness r_i they used during ballot creation and verify it matches what appears on the bulletin board.

The voter can also verify the proofs attached to their ballot are accepted as valid by the system, confirming their vote will be counted.

The voter can perform these verification steps without revealing their identity or how they voted, preserving ballot secrecy [11].

This protocol achieves end-to-end verification while maintaining strong privacy guarantees through the separation of authentication (credential authorities) and vote counting (trustees). The binding between credentials and ballots ensures only eligible voters participate, while the zero-knowledge proofs [16] and homomorphic tallying ensure no one can determine how any individual voted.

a) Protocol Security Considerations:

b) 1. Separation of Authorities:: Credential authorities should be established as separate entities from ElectionGuard trustees to maintain proper security boundaries [9]. This separation prevents collusion that could break voter privacy by connecting identities to votes. Different organizations should control each authority type, ideally representing diverse interests such as different political parties, branches of government, and civil society organizations.

c) 2. Coercion Resistance Enhancement:: The threshold credential can include coercion resistance mechanisms for high-stakes elections [20]. For instance, voters under coercion could use an alternate form of their credential that produces a seemingly valid but actually discarded ballot. Implementing this feature requires careful cryptographic design to preserve the one-voter-one-vote property while giving voters a way to protect themselves from coercion.

d) 3. Key Management:: Both systems require secure key management practices. Hardware security modules (HSMs) should store all authority and trustee key shares to prevent extraction or misuse. The system should implement rigorous ceremony protocols for key generation and storage, with appropriate witnesses and documentation to ensure trust in these critical processes.

e) Implementation Challenges:

f) 1. Performance Considerations:: The integration introduces additional cryptographic operations that impact performance. For large-scale elections, specific optimizations would be needed, including batched verification techniques that can validate multiple proofs simultaneously to improve throughput.

g) 2. System Complexity:: The integrated system is substantially more complex than either system alone. This complexity requires careful formal verification of the cryptographic protocols and their composition [21]. Clear security arguments must be developed and documented about how the systems interact to maintain their security properties when combined.

h) 3. Usability Issues:: The credential management adds complexity for voters compared to traditional voting systems. Thoughtfully designed user interfaces must guide voters through the process of managing their credentials and casting their votes. Additionally, secure recovery mechanisms for lost credentials must be established without introducing vulnerabilities into the system.

H. Deployment Strategy

a) Phase 1: Proof of Concept Development: The initial implementation focuses on creating a working prototype in a controlled environment. Software developers build the core cryptographic components including the threshold BLS credential system [15], zero-knowledge proof mechanisms [16], and integration points with ElectionGuard [10]. This phase includes extensive testing with simulated voters and elections to validate the mathematical correctness of the protocol. The deliverable is a functioning prototype demonstrating end-to-end operation of the integrated system with documentation of the cryptographic constructions and security assumptions.

b) Phase 2: Security Analysis and Auditing: Independent cryptographers and security researchers conduct thorough security analyses of the system design. This includes formal verification of the cryptographic protocols [21], threat modeling to identify potential vulnerabilities, and code auditing to find implementation flaws. The security team creates adversarial scenarios to test against various attack vectors, with particular attention to credential forgery attempts, double-voting attacks, and privacy leakages. Results from this phase inform necessary refinements to both the protocol design and implementation before proceeding to real-world testing.

c) Phase 3: Limited Field Trials: The system is deployed in small, low-stakes elections such as university student government or community organization votes (similar environments where Helios was initially tested [8]). These controlled deployments involve real voters but maintain parallel paper systems as backups. Administrators collect comprehensive metrics on performance, usability, and any technical issues encountered. Voter experience surveys capture feedback on the credential management process and verification steps. Each trial is followed by a thorough post-mortem analysis to identify

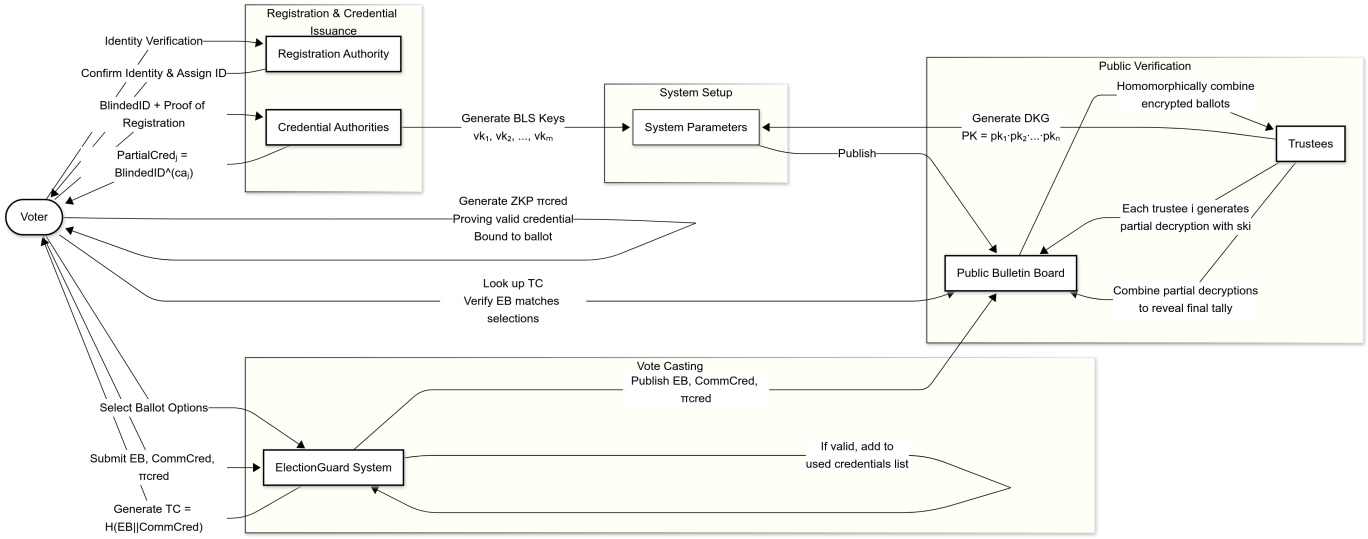

Fig. 7. Data Flow Architecture

areas for improvement in both the technology and supporting procedures.

d) Phase 4: Incremental Scale Expansion: Building on successful limited trials, the system gradually expands to larger elections with more diverse voter populations. This phase introduces the system to municipal elections or organizational votes with several thousand participants. Deployment includes comprehensive voter education campaigns, poll worker training programs, and technical support infrastructure. Each deployment maintains careful monitoring of system performance under increased load and more diverse usage conditions. Security monitoring tools watch for any unexpected behaviors or potential compromise attempts.

e) Phase 5: Certification and Full Implementation: The system undergoes formal certification processes applicable to the jurisdiction where it will be used. This includes compliance verification with relevant voting system standards, accessibility requirements, and security regulations. A full risk-limiting audit framework is established to validate election results post-deployment. The final roll-out includes creation of detailed contingency plans, backup procedures, and disaster recovery protocols. Long-term maintenance strategies ensure the system remains secure against evolving threats through regular updates and security reviews. By combining ElectionGuard's ballot encryption with threshold authentication [7], [10], we create a system that ensures only eligible voters can vote, each voter can vote only once, and no one—not even the system administrators—can connect voters to their specific votes. This maintains the end-to-end verifiability of ElectionGuard [11] while adding strong authentication guarantees from the threshold credential system.

V. Conclusion:

This paper undertook a comprehensive exploration of secure electronic voting systems, driven by the need to address persistent challenges in traditional and existing digital methods [1], [2] while upholding core democratic principles of fairness, eligibility, uniqueness, privacy, accuracy, and efficiency. An analysis of various cryptographic approaches for robust voter registration and authentication compared standard symmetric encryption, fuzzy commitments [4], [5], and threshold cryptosystems [6], [7], highlighting their respective strengths and weaknesses, particularly concerning biometric data handling and trust distribution. A review of prominent existing technologies, including Helios [8], Belenios [9], ElectionGuard [10], VotingWorks, Open.Vote, and blockchain-based systems [12], [13], provided context by examining their capabilities and limitations regarding end-to-end verifiability [11], privacy protection, trust models, and coercion resistance [20].

Building upon this foundation, the paper proposed a novel integrated framework combining the end-to-end verifiability and threshold encryption features of ElectionGuard [10], [11] with a robust threshold authentication system utilizing techniques like BLS signatures [15] and zero-knowledge proofs [16]. This hybrid design aims to achieve strong separation between authentication and tallying authorities, enhance resistance against single points of failure and certain attack vectors, and cryptographically ensure that only eligible, authenticated voters can cast ballots anonymously and verifiably, with each vote accurately counted. The detailed exploration covered conceptual design, data flow architecture, security considerations, and a phased deployment strategy. While this integrated approach offers significant theoretical advantages for enhancing electoral integrity and voter confidence, the paper also acknowledges substantial practical challenges. These include increased system complexity, potential performance overhead from advanced cryptographic operations, critical usability and accessibility concerns for diverse voter populations, the need for rigorous key management ceremonies, and the complexities of secure deployment and auditing. Further research,

formal security analysis [21], careful implementation focusing on user experience, pilot testing in controlled environments, and addressing accessibility barriers are crucial steps required before such advanced cryptographic voting systems can be considered for widespread, reliable adoption in critical elections.

REFERENCES

[1] A. D. Rubin, "Security Considerations for Remote Electronic Voting," *Commun. ACM*, vol. 45, no. 12, pp. 39–44, Dec. 2002. doi: 10.1145/585597.585602.

[2] R. L. Rivest, "Voting," in *Secure Elections: Perceptions and Realities*, D. Jefferson, Ed. Stanford, CA: Stanford University Press, 2008.

[3] A. K. Jain, A. Ross, and S. Prabhakar, "An Introduction to Biometric Recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 1, pp. 4–20, Jan. 2004. doi: 10.1109/TCSVT.2003.818349.

[4] A. Juels and M. Sudan, "A Fuzzy Vault Scheme," in *Proc. IEEE Int. Symp. Information Theory*, Lausanne, Switzerland, Jun./Jul. 2002, p. 408. doi: 10.1109/ISIT.2002.1023680.

[5] A. Juels and M. Wattenberg, "A Fuzzy Commitment Scheme," in *Proc. 6th ACM Conf. Computer and Communications Security (CCS '99)*, Singapore, Nov. 1999, pp. 28–36. doi: 10.1145/319709.319714.

[6] A. Shamir, "How to Share a Secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979. doi: 10.1145/359168.359176.

[7] Y. Desmedt, "Threshold Cryptography," *European Trans. Telecommunications*, vol. 5, no. 4, pp. 449–457, Jul./Aug. 1994. doi: 10.1002/ett.4460050407.

[8] B. Adida, "Helios: Web-based Open-Audit Voting," in *Proc. 17th USENIX Security Symposium*, San Jose, CA, USA, Jul./Aug. 2008, pp. 335–348.

[9] S. Cortier *et al.*, "Belenios: A Simple and Secure E-Voting System," *ACM Trans. Inf. Syst. Secur.*, vol. 19, no. 2, Article 7, Sep. 2016. doi: 10.1145/2975906.

[10] Microsoft, "ElectionGuard Specification." [Online]. Available: https://github.com/microsoft/electionguard.

[11] J. A. Halderman and A. Teague, "The Future of Voting: End-to-End Verifiable Elections," *Election Law Journal*, vol. 14, no. 4, pp. 346–358, Dec. 2015. doi: 10.1089/elj.2015.0327.

[12] P. McCorry, S. F. Shahandashti, and F. Hao, "A Smart Contract for Boardroom Voting with Maximum Voter Privacy," in *Proc. Int. Conf. Financial Cryptography and Data Security (FC)*, Malta, Apr. 2017, pp. 357–375. doi: 10.1007/978-3-319-70278-0_22.

[13] K. Kshetri and J. Voas, "Blockchain in Developing Countries," *IT Professional*, vol. 20, no. 2, pp. 11–14, Mar./Apr. 2018. doi: 10.1109/MITP.2018.021921645.

[14] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981. doi: 10.1145/358549.358563.

[15] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *J. Cryptology*, vol. 17, no. 4, pp. 297–319, Sep. 2004. doi: 10.1007/s00145-004-0314-9.

[16] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols," in *Proc. CRYPTO '94*, Santa Barbara, CA, USA, Aug. 1994, pp. 174–187. doi: 10.1007/3-540-48658-5_17.

[17] P. Y. A. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia, "Prêt à Voter: A Verifiable Voting System," *IEEE Trans. Inf. Forensics Secur.*, vol. 4, no. 4, pp. 662–673, Dec. 2009. doi: 10.1109/TIFS.2009.2033233.

[18] P. Y. A. Ryan and S. Schneider, "Prêt à Voter with Re-encryption Mixes," in *Proc. 11th Int. Conf. Research in Security (ESORICS)*, Hamburg, Germany, Sep. 2006, pp. 329–344. doi: 10.1007/11863908_21.

[19] J. Heather and P. Y. A. Ryan, "Experiences with Prêt à Voter," in *Proc. 2nd Int. Conf. E-voting and Identity (Vote-ID)*, Bregenz, Austria, Aug. 2009, pp. 1–12. doi: 10.1007/978-3-642-03997-3_1.

[20] P. Y. A. Ryan, "Security Requirements of Remote Electronic Voting," in *Towards Trustworthy Elections: New Directions in Electronic Voting*, D. Chaum *et al.*, Eds. Berlin, Heidelberg: Springer, 2010, pp. 131–146. doi: 10.1007/978-3-642-12980-3_8.

[21] V. Cortier, P. Gaudry, and P. Y. A. Ryan, Eds., *Proc. 6th Int. Conf. E-Voting and Identity (Vote-ID 2017)*, Bregenz, Austria, Oct. 2017, vol. 10615, Lecture Notes in Computer Science. Springer, Cham. doi: 10.1007/978-3-319-68687-5.

APPENDIX

This appendix describes our secure electronic voting system that integrates blockchain technology with a novel DNA-encoded biometric authentication mechanism to ensure electoral integrity while maintaining voter privacy.

DNA-Encoded Biometric Processing

The core innovation of our system lies in the biometric template protection using DNA encoding/decoding coupled with chaotic maps. The process secures voter biometric data as follows:

Biometric Capture and Template Creation: When a voter registers, their biometric data (facial structure, fingerprint, or retinal pattern) is captured via high-resolution sensors. The system extracts feature vectors from raw biometric data to generate a compact template. This template is then divided into multiple binary blocks of equivalent size (denoted as A1, A2, and A3) to prepare for the encryption process.

Chaotic Map Scrambling: The binary blocks undergo scrambling using chaotic systems characterized by extreme sensitivity to initial conditions. The system employs the following transformation:

$$\begin{aligned} A1(i,j) &\leftrightarrow A1(lx(i), ly(j)) \\ A2(i,j) &\leftrightarrow A2(lx(i), lz(j)) \\ A3(i,j) &\leftrightarrow A3(ly(i), lz(j)) \end{aligned} \tag{1}$$

Where lx, ly, lz, and lq represent chaotic sequence values generated with carefully selected initial conditions. This transformation creates substantial disorganization in the biometric template structure, making pattern recognition computationally infeasible without knowledge of the exact chaotic parameters.

DNA Encoding and Complementation: After chaotic scrambling, the system applies DNA encoding to the scrambled blocks. This process maps binary data to DNA nucleotide sequences (A, C, G, T) according to one of eight Watson-Crick complementary rule systems:

(AC)(TC)(CG)(GA)	(AT)(TG)(GC)(CA)
(AC)(CT)(TG)(GA)	(AC)(CG)(GT)(TA)
(AG)(GT)(TC)(CA)	(AG)(GC)(CT)(TA)

These encoded blocks undergo DNA complementation where each nucleotide is replaced with its complementary nucleotide (A↔T, C↔G). For example, a sequence "AACGT" becomes "TTGCA". This step introduces another layer of complexity to the encryption process.

DNA Decoding and Key Generation: The complemented DNA sequences are then decoded back to binary form, but now with significantly altered structure. The resulting binary stream undergoes SHA-256 hashing to generate a 256-bit encryption key. A representative output of this process produces a binary key: 0011010101000001010001010100000100110010001100110100010100110110...

This key is used with AES-256 to encrypt the original biometric template. The hash of the encrypted template is stored in the central database, while the encrypted template

itself is written to the voter's smart card. The decryption key is securely stored against the voter's national ID in the database and is only retrieved after successful hash verification.

Security Analysis

The security of this approach derives from multiple factors:

- **Chaotic Sensitivity**: The chaotic maps exhibit butterfly effect properties where minimal variations in input parameters lead to drastically different outputs, making reverse-engineering computationally infeasible.
- **DNA Encoding Complexity**: The utilization of DNA encoding creates 24 possible encoding schemes, with 8 complementary systems, substantially increasing the encryption complexity.
- **Template Diversity**: Any two biometric samples, even from the same individual, produce entirely different encrypted templates due to the chaotic nature of the encryption process, thereby preventing cross-matching attacks.
- **Cryptographic Strength**: The resulting AES-256 encryption combined with the dynamically generated keys provides cryptographic protection requiring approximately $2^{254.3}$ operations for the most efficient known attack, making brute-force approaches impractical.

Voting Architecture Integration

When a voter arrives at a polling station, they present their smart card containing the encrypted biometric template. The system:

1) Reads the encrypted template from the smart card
2) Computes its hash value and verifies against the database
3) If matched, retrieves the corresponding decryption key
4) Decrypts the template and captures the voter's live biometric data
5) Compares the decrypted template with the live biometric sample
6) Upon successful verification, authorizes the voter to cast their ballot
7) Records the vote on a separate blockchain with machine ID signature

This multi-layered verification process ensures that biometric data remains secure while enabling reliable voter authentication. The blockchain component maintains an immutable record of votes without compromising voter anonymity, as only the transaction ID and encrypted vote information are publicly visible.

The architecture successfully addresses limitations in current voting systems by preventing duplicate votes, eliminating ballot tampering, providing immediate results, and creating an auditable trail while maintaining voter privacy through the innovative application of DNA encoding and chaotic systems for biometric template protection.