# Anonymous E-Voting on Hyperledger Fabric: Practical Mitigation of the Verification Paradox and Coercion Resistance

Yoon-Nyoung Jung[1], Su-Been Cho[1],
Seo-Hyun Yun[1], and Hwa-Jeong Seo[1⋆]

Department of IT Convergence Engineering,
Hansung University, Seoul (02876), South Korea
{yoonnyoung789,chosubin1208,s7815546}@gmail.com

**Abstract.** Electronic voting systems inherently encompass a structural tension among ballot secrecy, verifiability, and coercion resistance. Voters must be able to verify whether their votes have been included; however, if such verification information can serve as evidence presentable to a third party, it becomes a basis for post-election intimidation. Existing studies have focused primarily on performance evaluation or data separation, and have not comprehensively addressed the structural tension between verifiability and coercion resistance. This study defines this tension as the *verification paradox* and designs and implements an electronic voting prototype on a three-organization consortium based on Hyperledger Fabric 2.5, combining a 2-of-3 endorsement policy, nullifier-based anonymity, Exponential ElGamal homomorphic tallying, zero-knowledge proof (ZKP)-based ballot validity verification, panic password-based deniable verification, and Private Data Collection (PDC)-based coerced vote separation. Quantitative evaluation results confirm a server latency overhead of +0.9% for ElGamal relative to the AES performance baseline, statistical indistinguishability between normal and panic responses ($p > 0.05$), and a peak throughput of approximately 40.7 TPS (with an error rate of 0%) under 1,000 concurrent voters. This study demonstrates that permissioned blockchains can provide practical mitigation of the verification paradox through implementation and quantitative evaluation.

**Keywords:** Electronic voting · Hyperledger Fabric · Permissioned blockchain · Verifiability · Coercion resistance · ElGamal · Zero-knowledge proof · Private Data Collection.

## 1 Introduction

With the digital transformation of the 21st century, social demand for electronic voting systems has been rapidly increasing beyond public elections into the private sector. Despite the clear advantages of guaranteeing suffrage, transitioning

⋆ Corresponding author: hwajeong84@gmail.com.

to a contactless society, and reducing election operation costs, the successful adoption of electronic voting depends on both technical maturity and the establishment of social trust. Estonia has successfully institutionalized this approach, whereas the United Kingdom and the Netherlands withdrew their adoption efforts due to security concerns. In South Korea, the National Election Commission operates the K-Voting online voting service [21]; however, the results of the 2023 security inspection of election-related systems at the National Election Commission demonstrated that system security and trust structures must be rigorously examined in discussions regarding the introduction of electronic voting [4].

The core challenge in electronic voting design lies in the structural tension among ballot secrecy, verifiability, and coercion resistance – namely, the *verification paradox*. Voters must be able to verify that their votes have been correctly recorded; however, if such verification means are provided as evidence to third parties, they can be exploited as tools for bribery or coercion. Conversely, if verification is restricted for the sake of coercion resistance, electoral transparency is compromised.

However, existing Hyperledger Fabric-based studies have focused on performance evaluation or data separation, without directly addressing this verification paradox or examining practical deployability.

To this end, this study employs a permissioned blockchain as a foundational layer that provides multi-institutional audit logs and state transition integrity. The multi-organization endorsement policy and Private Data Collection (PDC) of Hyperledger Fabric reduce dependence on trust in a single institution and restrict the scope of disclosure of sensitive data. Building upon this foundation, this study examines whether the verification paradox can be practically mitigated by combining cryptographic mechanisms such as homomorphic encryption, zero-knowledge proofs, and deniable verification, and confirms this through system implementation and quantitative evaluation.

## 2 Background and Related Work

### 2.1 Security Requirements of Electronic Voting

The security properties that an electronic voting system must satisfy are broadly classified into seven categories [2]. **Ballot Secrecy** is the property that a voter's candidate selection must not be disclosed. **Cast-as-Intended** means that the voter must be able to verify that the system has correctly encrypted the voter's intention. **Recorded-as-Cast** is the property that it must be possible to confirm that a submitted vote has been correctly included in the ledger. **Tallied-as-Recorded** is the property that a third party must be able to independently verify that the votes on the ledger have been accurately reflected in the final results. **Universal Verifiability** is the property that the legitimacy of the voting process must be publicly verifiable by anyone. **Eligibility Verifiability** is the property that it must be possible to verify that all voters who participated in the election possessed the proper qualifications. Finally, **Coercion Resistance** is

the property that even if a coercer compels a voter to vote for a specific candidate or demands the presentation of evidence regarding the vote content, the voter must be able to effectively refuse such demands [16].

A structural tension exists among these properties. As verifiability increases, voters become more capable of presenting vote evidence to third parties, which a coercer can exploit for post-election intimidation. This study defines this as the *verification paradox* and proposes practical mitigation thereof.

### 2.2 Trends in Electronic Voting and Blockchain Application

Examining the current state of electronic voting adoption in major countries worldwide reveals that results vary depending on the level of technology and social trust. Estonia represents a leading case that successfully operates nationwide electronic voting based on a national electronic ID system [13]. In contrast, Finland experienced a case in 2008 where a court invalidated an election due to tallying discrepancies that occurred during a local government online voting experiment, and the Netherlands likewise followed a path of complete abolition after a civic organization publicly identified technical vulnerabilities in 2006 [11].

In the case of South Korea, the National Election Commission operates the K-Voting system to support online voting for institutions and organizations whose use has been approved by the commission [21]. However, a joint security inspection conducted by the National Intelligence Service, the National Election Commission, and the Korea Internet & Security Agency in 2023 uncovered numerous hacking vulnerabilities in the voting and counting management systems, raising security concerns across the entire election information system [4]. Rather than characterizing this as a vulnerability specific to a particular online voting service, it can be interpreted as a case demonstrating that both central system security and distributed trust structures must be examined in conjunction across the entire electronic voting and counting infrastructure.

Blockchain has attracted attention as an alternative to address these limitations, and in practice, attempts to introduce blockchain into various electronic voting systems, including those at educational institutions, have been increasing [27]. The immutability, decentralization, and transparency of blockchain enhance the ability to detect post-hoc modifications and provide verifiability. However, since the transparency of public blockchains can conversely weaken vote privacy, permissioned blockchains, which enable access control and data separation, are more suitable for electronic voting.

### 2.3 Overview of Hyperledger Fabric

Hyperledger Fabric is a private blockchain platform that only permits access by authenticated participants through Membership Service Providers (MSPs) [15]. Its core components are as follows. *Peers* are nodes that host the distributed ledger and chaincode. *Chaincode* serves as smart contracts that read and write data on the ledger. *Orderers* sort transactions and generate blocks. *Channels*

**Table 1.** Comparison of related studies on e-voting systems. ✓ = supported, × = not supported, △ = partially supported.

| System | Ballot Secrecy | Cast-as-Int. | Rec.-as-Cast | Tallied-as-Rec. | Elig. Verif. | Coercion Resist. | Distributed Trust |
|---|---|---|---|---|---|---|---|
| Helios [1] | ✓ | ✓ | ✓ | ✓ | △ | × | Single server |
| Belenios [8] | ✓ | ✓ | ✓ | ✓ | ✓ | × | Trustee |
| JCJ [16] | ✓ | △ | △ | ✓ | ✓ | ✓ | Multi-authority |
| Kim [17] | △ | × | × | × | △ | × | Single org (HLF) |
| Saeed [24] | × | × | × | × | △ | × | Multi-org (HLF) |
| **This work** | ✓ | ✓ | ✓ | ✓ | ✓ | △ | 3-org 2-of-3 (HLF 2.5) |

logically partition the network so that only participating parties share the ledger. The *Certificate Authority (CA)* issues and manages MSP certificates.

The Execute-Order-Validate pipeline architecture of Fabric proceeds through four stages: client proposal submission, endorsement, ordering, and validation. Two core functionalities are relevant for electronic voting application. First, endorsement policies require the signatures of multiple organizations for the approval of specific transactions, thereby preventing manipulation by a single institution. Second, PDC separates sensitive data from the public ledger and restricts its delivery so that it is not transmitted to orderers or unauthorized peers, thereby providing restricted privacy.

### 2.4 Comparison of Related Work

Helios [1] is a web-based open-audit voting system with strong verifiability; however, it does not target coercion resistance or receipt-freeness, and it relies on a single server. Belenios [8] is a system that separates the credential issuance authority to simultaneously pursue privacy and verifiability; however, it does not target receipt-freeness or formal coercion resistance at the level of JCJ. JCJ [16] provides formal coercion resistance but is constrained in practicality due to the strong assumption of an untappable channel and the complexity of the tallying process.

Among existing Hyperledger Fabric-based studies, Kim (2019) [17] applied Fabric to a national defense personnel evaluation system, implementing channel separation and PDC-based access control; however, cryptographic vote privacy and coercion resistance were not addressed. Saeed et al. [24] systematically evaluated Fabric-based electronic voting workloads using Hyperledger Caliper; however, ZKP-based candidate value validity verification and coercion mitigation mechanisms were not included.

This study proposes a multi-layered mitigation of the verification paradox by combining a 2-of-3 endorsement policy, PDC-based data separation, Exponential ElGamal combined with ZKP, and panic passwords.

# 3 Proposed Method

## 3.1 System Architecture

This system employs a permissioned blockchain network based on Hyperledger Fabric 2.5. The network is composed of three organizations: ElectionCommission, PartyObserver, and CivilSociety, with 2, 1, and 1 peer nodes, respectively. The ordering service adopts an etcdraft 4-node Crash Fault Tolerant (CFT) architecture to eliminate single points of failure. Core transactions require a 2-of-3 endorsement policy, thereby reducing the risk that a single institution can unilaterally update voting records or tallying results. This configuration reflects actual election governance structures – the election administration authority, political parties, and civil society monitoring organizations – and reduces dependence on trust in a single institution through mutual checks among stakeholders with conflicting interests.

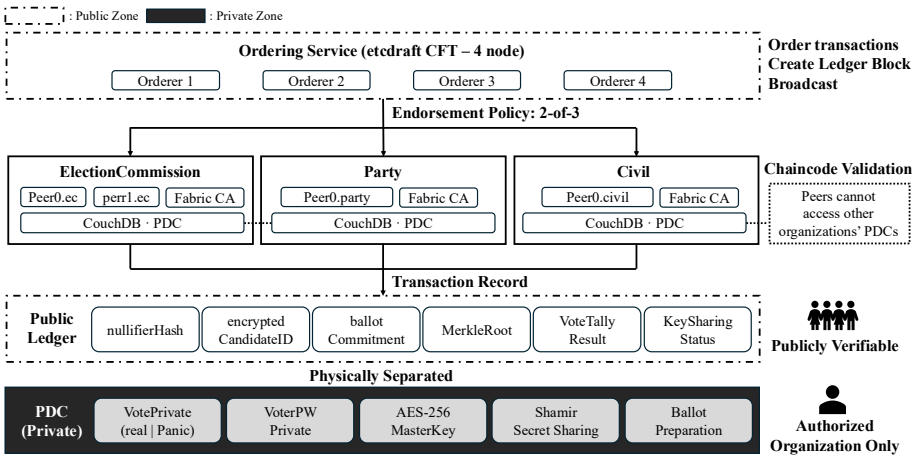



**Fig. 1.** System architecture diagram.

Data is separated into two tiers: the public ledger and PDC. The public ledger stores nullifier hashes, encrypted candidate records, candidate commitments, the Merkle tree root, and tallying results. Here, the candidate commitment is computed as SHA256($electionID \| nullifierHash \| encryptedCandidateID$), serving to detect ciphertext modifications and prevent cross-election replay attacks. PDC stores coercion-related metadata in isolation, including the credential type (normal/panic classification), panic password hashes, and encryption key shares. PDC data is not transmitted to orderers and can only be accessed by peers of authorized organizations.

*Threat Model and Scope.* This system considers five categories of threats. (1) API server compromise: public key signature-based credentials [3] are re-verified at

the chaincode level through four stages, rejecting any tampering. (2) Single-institution manipulation: the 2-of-3 endorsement policy enforces the consensus of at least two organizations. (3) Public ledger analysis: nullifier-based anonymization and ElGamal's probabilistic encryption sever the link between votes for the same candidate. (4) Coercer's demand for proof: this is countered through panic passwords, credential dualization, and re-voting. (5) Orderer failure: the etcdraft CFT 4-node configuration tolerates a single point of failure; however, defense against malicious orderers (BFT) has not been implemented. Furthermore, in the current prototype, the nullifier is based on a voterSecret that the voter locally generates; therefore, the scenario in which multiple nullifiers are generated using different voterSecrets with the same credential cannot be blocked at the chaincode level. Addressing this requires either credential hash-based duplicate detection or nullifier binding in the form of $nullifier = \mathrm{PRF}(credentialSecret, electionID)$, and the current implementation on the API server relies on duplicate submission prevention. Meanwhile, scenarios involving complete compromise of the voter's terminal, real-time screen surveillance, collusion of all institutions, and direct application to public office elections are not addressed in this study.

*Credential Issuance and On-Chain Verification.* The API server issues public key signature-based credentials using the voter's registered identifier, password, and election ID as inputs. Credentials are composed of the election ID, issuance and expiration timestamps, and a signature, and the voter identifier is not recorded on the ledger. When submitting a vote, the voter transmits this credential alongside the ballot, and the chaincode verifies the following in order: (1) expiration time, (2) election ID binding, (3) signature validity, and (4) whether the credential hash matches the corresponding entry in the PDC. If any verification step fails, the entire transaction is rejected, and all peers participating in the endorsement policy perform the same verification.

*Anonymous Credential Extension.* The basic credential employs a public key signature scheme, and thus a linkage between the issuer and the voter exists. To supplement this, this study implements an anonymous credential extension mode using a pairing-based randomizable signature, specifically the Pointcheval–Sanders (PS) signature [23], with selective disclosure and unlinkable proof support, as well as the BBS family of signatures [5,19]. The PS scheme verifies credentials through two bilinear pairing computations, while the BBS+ scheme enables selective disclosure of only the necessary attributes, such as eligibility status, while concealing the remaining identity information, and generates a new proof for each authentication session to enhance unlinkability between requests. Although both schemes exhibit higher authentication latency than the basic scheme, this is negligible relative to the overall Fabric pipeline latency. Detailed performance comparisons are discussed in Section 4.4.

*Nullifier-Based Last-Vote-Wins Model.* The voter's browser generates a 256-bit voterSecret locally and computes the nullifier by combining it with a per-election

blindingFactor:

$$nullifierHash = \text{SHA256}(voterSecret \| electionID \| blindingFactor) \quad (1)$$

Since the voterSecret is never transmitted outside the browser, observers of the public ledger cannot link a voter's identity to their nullifier. Because the electionID is included in the input, even with the same secret, a different nullifier is generated for each election, thereby blocking cross-election tracking. The blindingFactor is a public salt that the chaincode deterministically derives from the transaction ID and election ID through SHA256 hashing at election creation time and stores in the election state, and is distributed identically to all voters. Without the voterSecret, it is impossible to reverse-compute the nullifier. Consequently, only the voter who knows their voterSecret can regenerate their own nullifier to verify vote inclusion, thereby achieving the coexistence of anonymity and individual verifiability.

If a re-vote is cast with the same nullifier, the chaincode overwrites the existing record (last-vote-wins). This enables the voter to cast a re-vote in a free environment after coercion, thereby reflecting their final intention.

### 3.2 Privacy Assurance Mechanisms

*Vote Encryption.* The baseline implementation encrypts candidate IDs using AES-256-GCM. To satisfy the determinism requirement of Fabric chaincode, the nonce is deterministically generated as the upper 12 bytes of SHA256($key \| plaintext$). While this scheme does not leave the candidate plaintext on the ledger, it generates identical ciphertexts for the same candidate, rendering it vulnerable to frequency analysis; thus, it serves as a baseline implementation for performance comparison that does not satisfy ballot secrecy.

The extended implementation applies the Exponential variant ($g^m$ encoding) of the ElGamal cryptosystem [10]. This variant was selected because its multiplicative homomorphic property enables additive homomorphic tallying, whereby individual votes can be aggregated without decryption. A 2048-bit safe prime $p$ ($p = 2q + 1$) and a generator $g = 2$ of the subgroup of order $q$ [18] are used, and the encryption for candidate index $i$ is as follows:

$$m = B^i, \quad c_1 = g^r \bmod p, \quad c_2 = g^m \cdot y^r \bmod p \quad (2)$$

Here, $B = 10{,}000$ is a base constant to prevent collisions in vote counts between candidates, $y = g^x$ is the public key, and $r$ is a random nonce freshly generated for each vote. Since $r$ varies each time, even votes for the same candidate produce different ciphertexts. The chaincode receives only the ciphertext and the ZKP, and since it cannot ascertain the plaintext candidate ID, even endorsing peers cannot determine individual vote contents.

*Ballot Validity Zero-Knowledge Proof.* The voter must prove that the vote is a valid ciphertext encoding exactly one of the $k$ permitted candidates, without revealing the candidate selection. To this end, a disjunctive OR-proof based on

partial knowledge proof [9] is applied. The key idea is that a legitimate proof is generated for the actually selected candidate, while simulated proofs are constructed for the remaining $k-1$ candidates, such that the verifier cannot distinguish between the two. In the simulation branch, the challenge $e_j$ and response $z_j$ are randomly selected, and $a_{1,j} = g^{z_j} \cdot c_1^{-e_j}$, $a_{2,j} = y^{z_j} \cdot (c_2/g^{m_i})^{-e_j}$ are back-computed. In the actual branch, a random nonce $w$ is selected, and $a_{1,i} = g^w$, $a_{2,i} = y^w$ are computed. Subsequently, the Fiat–Shamir transform is applied to convert the interactive OR-proof into a non-interactive proof by hashing all commitments, ciphertexts, and public parameters to generate the challenge [12]:

$$e = \mathrm{SHA256}(c_1 \| c_2 \| a_{1,0} \| a_{2,0} \| \cdots \| a_{1,k-1} \| a_{2,k-1}) \bmod q \tag{3}$$

The challenge for the actual branch is computed as $e_i = e - \sum_{j \neq i} e_j \bmod q$, and the response is $z_i = w + e_i \cdot r \bmod q$. The chaincode verifies for each candidate $j$ that $g^{z_j} = a_{1,j} \cdot c_1^{e_j}$ and $y^{z_j} = a_{2,j} \cdot (c_2/g^{m_j})^{e_j}$, and confirms that $\sum e_j$ matches the hash. This verification does not decrypt the candidate, and therefore the voter can prove the validity of the ballot without exposing their selection, while the chaincode can cryptographically reject invalid ballots.

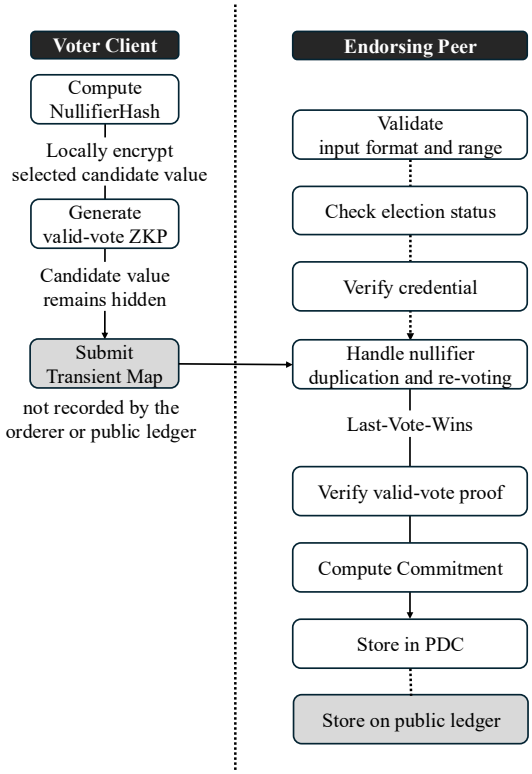



**Fig. 2.** Vote transaction processing flow.

*Homomorphic Tallying and Distributed Key Management.* Individual ciphertexts are tallied without decryption using ElGamal's multiplicative homomorphic property. When multiplying $n$ vote ciphertexts $(c_{1,i}, c_{2,i})$ component-wise:

$$\prod E(m_i) = \Big(\prod c_{1,i},\ \prod c_{2,i}\Big) = E\Big(\sum m_i\Big) \tag{4}$$

Here, $E(m_i)$ denotes the ciphertext of the $i$-th vote. To recover the exponent value $\sum m_i$ from $g^{\sum m_i}$ obtained after decryption, a baby-step giant-step algorithm with time and space complexity $O(\sqrt{N})$ is applied for a search upper bound $N$ [26]. Here, $N$ is the search upper bound of the aggregated exponent value, determined by the number of candidates $k$, the base $B$, and the maximum number of votes. In the conditions of this experiment, the search upper bound was sufficiently small that computation was completed within a few milliseconds; however, as the number of candidates and the maximum number of votes increase, the cost of discrete logarithm recovery also increases, necessitating separate performance evaluation and encoding optimization for large-scale elections. The recovered value is decomposed in base-$B$ to compute per-candidate vote counts, and in this process, only the aggregate value is decrypted without decrypting individual vote ciphertexts.

In AES mode, the encryption key is distributed to three organizations using Shamir's $(2, 3)$ threshold secret sharing scheme after the election concludes, and subsequently deleted from PDC; at least two organizations must submit their shares to reconstruct the key for tallying. In ElGamal mode, however, the private key is stored directly in PDC and is not currently subject to Shamir distribution; replacing this with Distributed Key Generation (DKG) is identified as a future task [25]:

$$f(x) = s + a \cdot x \bmod q, \quad \textit{threshold} = 2, \quad \textit{shares} = 3 \tag{5}$$

Here, $s$ is the private key, $a$ is a randomly chosen polynomial coefficient (distinct from the ElGamal encryption nonce $r$ in Equation 2), and $q$ is the order of the ElGamal subgroup.

The original key is deleted from PDC after distribution, ensuring that no single organization can independently decrypt. However, during the election, the original private key exists in PDC, and thus there is a potential risk that a peer operator with PDC access rights could access the private key; this must be addressed through the future introduction of Distributed Key Generation (DKG). When any two organizations submit their shares, the private key is reconstructed via Lagrange interpolation, forming a trust structure consistent with the 2-of-3 endorsement policy.

### 3.3 Verifiability Assurance

*Cast-as-Intended Verification.* The method for verifying that the voter's intention has been correctly encrypted differs depending on the encryption mode. In the AES baseline implementation, a commit-then-audit protocol [2] is adapted

and applied. (1) When the voter selects a candidate, the chaincode generates and returns an encrypted provisional ballot (commitment). (2) The voter chooses between auditing and submitting. If auditing is selected, the chaincode discloses the encryption key of the corresponding ballot, enabling the voter to independently verify the re-encryption result; the audited ballot is discarded and not included in the tally. (3) If submission is selected, the corresponding ciphertext is recorded on the ledger as the final vote. In this case, since the chaincode is aware of the candidate plaintext, ballot secrecy depends on trust in the chaincode. In ElGamal mode, the voter directly encrypts in the browser and attaches a disjunctive OR-proof; therefore, the chaincode verifies only validity without knowledge of the candidate plaintext. In this case, cast-as-intended assurance on the correctness of the client-side encryption implementation and the soundness of the ZKP.

*Record Integrity Verification.* After the election concludes, the chaincode constructs a binary Merkle tree over all nullifiers, using SHA-256 for each node hash in this implementation. The root hash is recorded on the public ledger and subsequently used for inclusion proof verification [20]. Each leaf is composed of a hash combining the nullifier hash, encrypted candidate ID, and candidate commitment, thereby enabling the detection of post-hoc ciphertext replacement. The voter recomputes the nullifier using their voterSecret, requests an inclusion proof, and recomputes the path from the leaf to the root to independently confirm that their vote has been recorded without modification.

*Tally Accuracy Verification.* To verify the accuracy of the tally in ElGamal mode, the chaincode decrypts the homomorphically aggregated ciphertext $(C_1, C_2)$ using the private key $x$, and generates a non-interactive zero-knowledge proof [6] for the discrete logarithm equivalence. This proof guarantees that $\log_g y = \log_{C_1}(C_2/g^m)$, i.e., that the public key $y = g^x$ was computed with the same private key $x$ used to compute the decryption factor $C_1^x$. Specifically, a random nonce $k$ is selected to compute $a_1 = g^k$, $a_2 = C_1^k$, and the challenge $e = \text{SHA256}(g\|y\|C_1\|C_2\|a_1\|a_2) \bmod q$ is derived, after which the response $z = k + e \cdot x \bmod q$ is computed. An external auditor can verify decryption accuracy without the private key by checking that $g^z = a_1 \cdot y^e$ and $C_1^z = a_2 \cdot (C_2/g^m)^e$.

The above three mechanisms constitute an end-to-end (E2E) verification chain from vote generation to tallying for normal votes recorded on the public ledger. However, since the exclusion of panic votes from the tally depends on verification by organizations with PDC access rights, complete public universal verifiability is not provided.

### 3.4 Coercion Resistance Layers

*Panic Password-Based Deniable Verification.* At election creation time, the chaincode generates dummy nullifiers for each candidate, including them in the Merkle tree with a structure identical to that of actual votes. Dummy nullifiers are generated in a fixed quantity per candidate (three in the prototype) as

SHA256("DUMMY"$\|electionID\|candidateID\|index\|txID$), and dummy ciphertexts and commitments are also recorded on the ledger with a structure identical to that of actual records. During tallying, dummy records are explicitly excluded and do not affect the results.

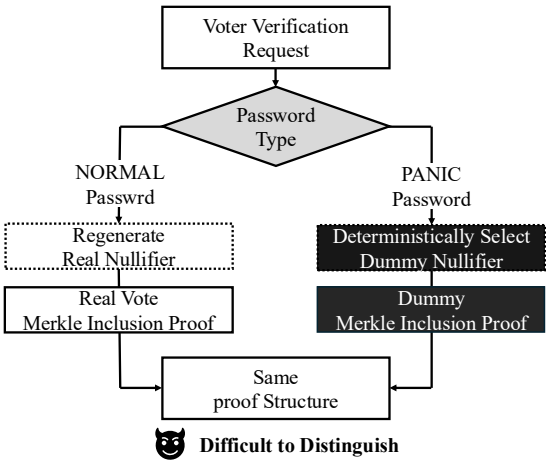



**Fig. 3.** Deniable verification flow based on panic passwords.

Voters set both a normal password and a panic password at the time of voting, and the hash of each is stored in PDC:

$$normalPWHash = \text{SHA256}(normalPW \| nullifierHash) \tag{6}$$

$$panicPWHash = \text{SHA256}(panicPW \| nullifierHash) \tag{7}$$

When the panic password is entered in the presence of a coercer, the chaincode deterministically selects one of the dummy nullifiers based on the corresponding hash and returns an inclusion proof. Since the response has the same structure as a normal response, the coercer cannot distinguish between the two modes based on response alone, and statistical verification thereof is presented in Section 4.3.

*Credential Dualization and Record Shuffle.* When the credential type is set to "panic" under coercion and a vote is cast, the corresponding information is recorded only in PDC and is indistinguishable from a normal vote on the public ledger. During tallying, the chaincode queries the PDC to exclude panic votes from the results. This is an adaptation of the coerced vote filtering concept [16] using Fabric PDC. However, since PDC is not an untappable channel but rather a private data separation layer, this study positions it as a limited data separation layer.

**Table 2.** Experimental environment.

| Item | Setting |
|---|---|
| Blockchain platform | Hyperledger Fabric 2.5 (CCAAS) |
| Organization configuration | ElectionCommission, PartyObserver, CivilSociety |
| Consensus / Ordering | etcdraft 4-node CFT, 2-of-3 endorsement policy |
| State DB | CouchDB |
| Batch Timeout | 2 s (default) |
| Encryption mode | AES-256 (baseline), ElGamal |
| Credential | Ed25519 (baseline), PS-BN254, BBS+BLS12381 |
| Hardware | Apple M2, 8 GB RAM, macOS Darwin 24.6.0, Docker |

Meanwhile, in a system that permits re-voting, timing analysis poses a threat. An attacker who observes the return order of vote records can potentially infer the timing of re-votes. To prevent this, the chaincode applies a deterministic shuffle to randomize the order upon record return. The shuffle seed is generated by combining the encryption key, election ID, and tally proof hash, thereby satisfying Fabric's determinism requirement while making external temporal analysis difficult.

The above four mechanisms – panic passwords, credential dualization, re-voting, and record shuffle – constitute a multi-layered mitigation of the verification paradox. While maintaining verifiability, they ensure that voters can present false evidence or retroactively replace coerced votes, and that coercion status cannot be identified from the public ledger, thereby restraining the appropriation of verifiability as a tool for coercion.

## 4 Performance Evaluation

### 4.1 Experimental Environment

The experimental environment is summarized in Table 2. The AES/ElGamal core performance comparison was measured with $N = 100$, excluding 10 warmup iterations. The credential scheme-specific authentication performance was measured with $N = 50$, and the normal/panic indistinguishability evaluation was measured with $N = 90$ for each mode. All experiments were conducted in the Ed25519 credential verification mode with actual credential verification enabled.

### 4.2 Core Performance Measurement Results

Table 3 compares the performance of the AES baseline implementation and the ElGamal extension. Approximately 92% of the vote confirmation latency is attributable to the Fabric block generation cycle (BatchTimeout 2 s), and security logic such as credential verification, ZKP verification, and nullifier processing amounts to only a few milliseconds in total. Therefore, even when transitioning from AES to ElGamal, the server-side overhead is limited to an average

**Table 3.** Performance comparison between AES baseline implementation and Exponential ElGamal ($N = 100$).

| Metric | AES | ElGamal | Overhead |
|---|---|---|---|
| Mean vote finalization (ms) | 2,165 | 2,184 | +19 (+0.9%) |
| Vote finalization P50 (ms) | 2,157 | 2,178 | +21 (+1.0%) |
| Vote finalization P95 (ms) | 2,245 | 2,263 | +18 (+0.8%) |
| Client-side ZKP generation (ms) | – | 50.5 | – |
| Payload size (bytes) | ∼200 | 4,043 | ∼20× |

**Table 4.** Statistical test for indistinguishability of normal/panic verification responses ($N = 90 + 90$).

| Metric | Normal | Panic |
|---|---|---|
| Mean latency (ms) | 372.8 | 375.9 |
| P50 latency (ms) | 358.2 | 359.1 |
| Standard deviation (ms) | 41.7 | 43.6 |
| Mean response size (bytes) | 1,168.8 | 1,168.2 |
| Sample size ($N$) | 90 | 90 |
| Welch's $t$-test: $t \approx -0.49$, $df \approx 178$, $p > 0.05$ | | |

of +0.9% and +0.8% at P95, with the additional cost primarily arising from client-side ZKP generation (50.5 ms) and payload size increase (∼20×). This implies that even strengthening the encryption scheme has no practical impact on user-perceived latency.

### 4.3 Indistinguishability Evaluation of Deniable Verification

The core requirement for coercion resistance is that normal and panic verification responses must be indistinguishable to an external observer. Table 4 presents the results of measuring both modes 90 times each in the same election and performing Welch's $t$-test.

With $p > 0.05$ ($t \approx -0.49$, $df \approx 178$), the mean latency difference between the two modes (3.1 ms) is statistically insignificant. This difference is less than 1% of the mean latency and sufficiently small relative to the standard deviation of approximately 42 ms, and the response size difference of 0.6 bytes is virtually identical. Therefore, it is practically infeasible for a coercer to distinguish between the two modes through statistical analysis of response time or size. However, these results are limited to experiments based on mean latency and response size, and evaluation of distinguishability based on distribution shape or classifiers remains a future task.

**Table 5.** Authentication performance comparison by credential scheme ($N = 50$, single thread).

| Metric | Ed25519 | PS-BN254 | BBS+ |
|---|---|---|---|
| Authentication latency avg (ms) | 0.35 | 59.0 | 48.4 |
| Authentication latency P50 (ms) | 0.33 | 56.8 | 47.3 |
| API TPS ($c$=20) | 6,632 | ∼18 | ∼21 |
| Credential size (bytes) | 238 | 359 | 326 |

### 4.4 Credential Scheme-Specific Authentication Performance Comparison

Table 5 compares authentication performance according to the credential scheme. The significant increase in authentication latency for PS and BBS+ relative to Ed25519 is attributable to the intrinsic cost of bilinear pairing computations. However, as confirmed in Section 4.2, the vote confirmation latency is predominantly composed of Fabric pipeline costs; therefore, authentication delays of less than 60 ms have a limited impact on user experience. However, the API throughput of PS/BBS+ is significantly lower than that of Ed25519, and thus parallelization of the authentication layer must be separately considered for high-concurrency operations. BBS+ exhibits an 18% lower latency than PS while also providing the selective attribute disclosure and unlinkability described in Section 3.1.

### 4.5 Performance Comparison with Related Work

Table 6 compares the performance of this system with related systems. Since the experimental environments differ, caution is required for direct numerical comparisons.

Saeed et al. [24] reported approximately 106 TPS for a 1-org configuration, approximately 61 TPS for a 2-org configuration, and approximately 40.3 TPS for a 3-org configuration in a Fabric-based electronic voting workload. Clarke et

**Table 6.** Performance comparison with related studies under different experimental environments.

cccc

| System | TPS | Distributed Trust | ZKP Cost |
|---|---|---|---|
| Saeed et al. [24] | 106 (1-org), 61 (2-org), 40.3 (3-org) | Multi-org HLF | None |
| Clarke et al. [7] | ∼75 (CastVote) | IBM Cloud HLF | None |
| Pan et al. [22] | 8.0–9.9 (ZKP) | 2-org HLF | 341 ms (add ZKP tx) |
| **This work** | ∼40.7 (3-org, peak) | 3-org HLF | OR-proof 50.5 ms ($k = 2$) |

al. [7] reported that the write function CastVote achieved a maximum throughput of approximately 75 TPS on IBM Cloud-based Hyperledger Fabric. These studies do not include ZKP-based candidate value validity verification or homomorphic tally decryption proof costs in their performance evaluations. This study, in a 3-org 2-of-3 configuration, includes additional security computations such as ZKP verification, PDC storage, and credential verification, yet achieved a peak of approximately 40.7 TPS, suggesting that the cost of additional security logic was not dominant relative to the overall Fabric transaction pipeline cost in this experimental environment.

Pan et al. [22] reported that when ZKP functionality was integrated into Hyperledger Fabric, throughput for query and add operations decreased to approximately 8.0 TPS and 9.9 TPS, respectively. In contrast, the OR-proof in this study is generated on the client side and the chaincode performs only verification; therefore, the computational burden on the chaincode side is lower than configurations where ZKP verification logic is included in the Fabric transaction path.

Additionally, from the perspective of zero-knowledge proofs, the disjunctive OR-proof in this study is applicable without a trusted setup, and under the simple selection conditions of this experiment, it demonstrated sufficiently low costs of 50.5 ms for generation and approximately 5 ms for verification. In contrast, Huber et al. [14] demonstrated that General-Purpose ZKP (GPZKP) is practically feasible for ballot validity proofs based on Exponential ElGamal by applying Groth16 zk-SNARKs for various voting schemes and complex ballot formats. While Groth16 requires per-circuit setup, it offers the advantages of small proof size and near-constant verification time; therefore, it can be considered as an alternative for large-scale elections with many candidates or complex ballot validity conditions.

### 4.6 Scalability Evaluation

Table 7 presents the results of measuring throughput, latency, and error rate while increasing the number of concurrent voters ($C$) from 1 to 3,000.

In the range of $C = 1$–1,000, an error rate of 0% was maintained, and TPS of approximately 35–41 demonstrated stable throughput in the range of $C =$ 300–1,000. The maximum throughput was observed at approximately 40.7 TPS at $C = 1{,}000$. The average latency of approximately 2.2–2.9 seconds in the range of $C = 1$–100 indicates that the block generation wait cost due to Fabric BatchTimeout (2 s) is dominant.

At $C \geq 1{,}500$, the error rate surged to 17.6% or higher, and the average latency exceeded 20 seconds, indicating that the system entered a saturation state. This is attributable to the I/O contention of the CouchDB state database and the concurrent transaction processing limits of the Fabric ordering pipeline. Therefore, the practical saturation point of this system is $C \approx 1{,}000$, and the stable operation range is judged to be $C = 300$–1,000 (TPS $\approx$ 35–41, error rate 0%).

**Table 7.** Throughput by number of concurrent voters and system limits.

| Concurrent voters ($C$) | TPS | Mean latency (ms) | P95 (ms) | Error rate |
|---|---|---|---|---|
| 1 | 0.5 | 2,164 | 2,243 | 0% |
| 10 | 4.2 | 2,367 | 2,406 | 0% |
| 50 | 11.1 | 2,858 | 2,949 | 0% |
| 100 | 21.0 | 2,848 | 2,856 | 0% |
| 300 | 36.5 | 6,920 | 7,467 | 0% |
| 500 | 34.7 | 6,745 | 7,635 | 0% |
| 1,000 | 40.7 | 6,608 | 7,238 | 0% |
| 1,500 | 15.2 | 22,242 | 26,976 | 17.6% |
| 2,000 | 13.0 | 20,518 | 27,800 | 27.2% |
| 3,000 | 23.2 | 12,387 | 21,182 | 50% |

Based on $C = 100$ (21.0 TPS), approximately 151,000 votes can be processed arithmetically over a 2-hour period, and based on $C = 1{,}000$ (40.7 TPS), approximately 293,000 votes over a 2-hour period. However, vote processing at this throughput is arithmetically feasible, and separate verification is required for sustained load over extended periods.

However, the scalability evaluation of this experiment is limited to loads of up to 3,000 concurrent users, and the feasibility of longer-duration, larger-scale operations requires separate verification. Furthermore, real-time terminal surveillance, malicious orderers, and other exceptional scenarios necessitate the future integration of BFT orderers, TEE-based client protection, and game-based formal proof introduction. The current prototype's private key PDC storage structure must be replaced with Distributed Key Generation (DKG), and strengthening the cryptographic binding between nullifiers and credentials is also a future task. From a cryptographic security and efficiency perspective, the extension of on-chain ZKP re-verification at the chaincode level and the transition to elliptic curve ElGamal can be enhanced.

## 5 Conclusion

This study defined the verification paradox – the structural tension among ballot secrecy, verifiability, and coercion resistance in electronic voting systems – and designed and evaluated a method for its practical mitigation in a permissioned blockchain environment based on Hyperledger Fabric 2.5.

The results of the implementation-based quantitative evaluation are as follows. First, through the 2-of-3 endorsement policy among three organizations (election commission, political party, and civil society organization), the risk of a single institution unilaterally updating the ledger state was reduced. Second, the server-side overhead upon introducing Exponential ElGamal homomorphic tallying was confirmed to be less than 1% relative to the AES performance

comparison baseline, confirming that the additional cost associated with security enhancement is negligible. Third, the latency difference (3.1 ms) between normal verification responses and panic verification responses was statistically insignificant ($p > 0.05$), confirming that it is difficult for a coercer to identify the mode through response analysis. Fourth, a peak of approximately 40.7 TPS (error rate 0%) was confirmed under 1,000 concurrent users, and at 1,500 or more concurrent users, the error rate surged due to state DB I/O contention, demonstrating that the practical saturation point exists at $C \approx 1{,}000$.

The E2E verification chain, composed of cast-as-intended verification, record integrity verification, and tally accuracy verification, supports independent verification of the public ledger. Simultaneously, the four-layer coercion mitigation mechanism, composed of panic passwords, credential dualization, re-voting, and record shuffle, restrains the appropriation of verification means as tools for coercion. This demonstrates that through the combination of cryptographic design and permissioned blockchain, verifiability can be maintained while the risk of evidence-based coercion can be practically mitigated.

As social demand for electronic voting expands broadly into the private sector, this study has demonstrated at the current juncture that permissioned blockchain serves as an effective foundation for practically mitigating the verification paradox. Future tasks remain in narrowing the gap between the technical prototype and real-world governance by pursuing legal and institutional compatibility analysis in parallel with election audit standardization.

## References

1. Adida, B.: Helios: Web-based open-audit voting. In: Proceedings of the 17th USENIX Security Symposium. pp. 335–348. USENIX Association (2008)
2. Benaloh, J.: Simple verifiable elections. In: Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop (EVT). USENIX Association (2006)
3. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. Journal of Cryptographic Engineering **2**(2), 77–89 (2012)
4. Boannews: National election commission hacking controversy: Comprehensive cybersecurity deficiencies confirmed, including hacking vulnerabilities in voting and counting systems. `https://www.boannews.com/media/view.asp?direct=mobile&idx=122505` (2023), accessed: 2026-05-20
5. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Proceedings of CRYPTO 2004. Lecture Notes in Computer Science, vol. 3152, pp. 41–55. Springer (2004)
6. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Proceedings of CRYPTO 1992. Lecture Notes in Computer Science, vol. 740, pp. 89–105. Springer (1992)
7. Clarke, R., McGuire, L., Baza, M., Rasheed, A., Alsabaan, M.: Online voting scheme using IBM cloud-based hyperledger Fabric with privacy-preservation. Applied Sciences **13**(13), 7905 (2023)
8. Cortier, V., Gaudry, P., Glondu, S.: Belenios: A simple private and verifiable electronic voting system. In: Foundations of Security, Protocols, and Equational

Reasoning. Lecture Notes in Computer Science, vol. 11565, pp. 214–238. Springer (2019)
9. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Proceedings of CRYPTO 1994. Lecture Notes in Computer Science, vol. 839, pp. 174–187. Springer (1994)
10. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985)
11. i Esteve, J.B.: Online voting: Current and future practices. Tech. rep., International IDEA (2025), iSBN 978-91-8137-024-9 (HTML), 978-91-8137-023-2 (PDF)
12. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Proceedings of CRYPTO 1986. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer (1986)
13. Hong, S.P.: A study on the applicability of online voting systems using blockchain technology. Tech. rep., National Election Commission (2017)
14. Huber, N., Küsters, R., Liedtke, J., Rausch, D.: ZK-SNARKs for ballot validity: A feasibility study. In: E-Vote-ID 2024. Lecture Notes in Computer Science, vol. 15014, pp. 107–123. Springer (2024)
15. Hyperledger Foundation: Hyperledger Fabric documentation v2.5. `https://hyperledger-fabric.readthedocs.io/en/release-2.5/` (2024)
16. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES). pp. 61–70. ACM (2005)
17. Kim, K.H.: Design and Implementation of a Participatory National Defense Personnel Voting System Using Hyperledger. Master's thesis, Ajou University Graduate School of Information and Communication Technology (2019)
18. Kivinen, T., Kojo, M.: RFC 3526: More modular exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE). IETF RFC 3526 (2003)
19. Looker, T., Kalos, V., Whitehead, A., Lodder, M.: The BBS signature scheme. Internet-Draft draft-irtf-cfrg-bbs-signatures-10, IETF Crypto Forum Research Group (2026), work in progress
20. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Proceedings of CRYPTO 1987. Lecture Notes in Computer Science, vol. 293, pp. 369–378. Springer (1987)
21. National Election Commission: Introduction to the online voting system (K-Voting) service. `https://vote.kvoting.go.kr/serviceIntroduce/onlineVote.do` (2026), accessed: 2026-05-20
22. Pan, R., Shi, Z., Belloum, A., Zhao, Z.: Operating ZKPs on blockchain: A performance analysis based on hyperledger Fabric. In: Proceedings of the 2024 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS). pp. 69–78. IEEE (2024)
23. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Proceedings of CT-RSA 2016. Lecture Notes in Computer Science, vol. 9610, pp. 111–126. Springer (2016)
24. Saeed, S.H., Hadi, S.M., Hamad, A.H.: Performance evaluation of e-voting based on hyperledger Fabric blockchain platform. Revue d'Intelligence Artificielle (IIETA) **36**(4), 581–587 (2022)
25. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)
26. Shanks, D.: Class number, a theory of factorization, and genera. In: Proceedings of Symposia in Pure Mathematics. vol. 20, pp. 415–440. American Mathematical Society (1971)

27. Sung, K.J., Jung, C.R., Cho, E.A., Lee, J.H., Kim, H.Y., Kim, Y.W., Lee, K.H.: A blockchain-based campus electronic voting system. Journal of the Korea Institute of Information Security and Cryptology **28**(4), 779–787 (2018)