A Blockchain Model for Ensuring Privacy, Trust and Dependability of Electronic Voting Systems

Sohel Ahmed Joni[1], Rabiul Rahat[1], Nishat Tasnin[1], Partho Ghose[2*], Hasan Jamil [3*]

[1]Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Rupnagar, Mirpur-2, Dhaka, 1216, Dhaka, Bangladesh.

[2*]Department of Biological and Agricultural Engineering, Texas A&M University, , College Station, 77840, Texas, United States.

[3*]Department of Computer Science, University of Idaho, 875 Perimeter Drive, Moscow, 83844-1010, Idaho, United States.

*Corresponding author(s). E-mail(s): partho.ghose@tamu.edu; jamil@uidaho.edu;
Contributing authors: sohelahmedjony@gmail.com; rabiulrahatt@gmail.com; nishattasnin02@gmail.com;

Abstract

This version of the article has been accepted for publication, after peer review and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/s10115-026-02693-6.

Voting is fundamental to exercising civic rights, and its proper administration is critical for ensuring trust in the democratic process. As the size of the voting population increases, and the on-time pronouncement of victors becomes essential, electronic voting machines and systems have emerged as crucial components of election administration. Existing security methods, such as centralized server-based authentication or traditional public-key cryptography, often suffer from major limitations, including single points of failure (SPOF) and poor scalability. Blockchain systems that rely on Proof of Work (PoW) or Proof of Stake (PoS) also introduce their own problems, such as lower throughput, high energy consumption, and increasing vulnerability to emerging post-quantum attacks. However, recent confusion and misinformation campaigns over fraudulent voting and improper counting in the USA and elsewhere point to the potential chaos that could ensue and how citizens' privacy could be threatened. In this

paper, we propose a novel blockchain-based hybrid consensus voting mechanism called the *Post Quantum Secured Hierarchical Authoritative Consensus* (PQSHAC), to improve privacy and trust in contemporary systems to improve dependability. The optimized sharding protocol we introduce utilizes the inherently distributed nature of election data to help reduce skewness and increase the cohesion of individual shards. We prevent extraneous voting and ensure voter privacy using a novel token generation protocol. The experiments demonstrate that PQSHAC achieves a throughput of $\approx$ **106** blocks/s with a block generation time of 28 seconds for 3000 blocks using a 5-shard configuration. Additionally, the proposed modular ledger architecture reduces storage requirements by approximately 60% compared to non-modular approaches, while maintaining robust security against quantum computing attacks via Dilithium-3 integration. This findings demonstrates PQSHAC's strengths over its competitors.

Keywords: EVM, Blockchain, Sharding, Multi-party computation, Security, Deep learning

1 Introduction

Election is the cornerstone of all democracies, and authentic voting systems ensure civic participation in this process. In an ideal democracy, eligible citizens must be able to effortlessly vote in private to elect their representatives and their opinions must count. These ideals are fundamental to all democracies, some more so than others. The three key ingredients of the voting systems of any election process are voter authentication, privacy during the casting of the ballots, and their accurate counting. Transparency of this process is vital for public acceptance of the results and thus the trust in the system [1]. Introduction and adoption of voting machines for efficient conduct of the elections have introduced a layer of distrust in the public [2] due to the lack of direct human involvement in the validation of the results and the potential to integrity failures of the ballots [3]. Although security challenges exist in electronic voting systems [4], they are exacerbated by the continued erosion of trust in governance and politics [5–7]. Recent incidents of e-voting vulnerabilities and hacking have further reignited concerns about the security of these systems. For example, Halderman's [8] 2011 study identified systemic flaws in India's EVMs, including a lack of verifiability and susceptibility to tampering. Similarly, the 2017 DEFCON Voting Machine Hacking Village [9] demonstrated how easily e-voting machines could be manipulated. In addition, ballot stuffing is a persistent issue in Bangladesh's elections, undermining their integrity, particularly in the 2018 and 2024 national elections. It was alleged that between 30% and 60% of votes were cast the night before the election day, undermining the credibility of the results[10, 11]. These incidents, along with Rush's [12] 2019 analysis on legal and technical safeguards, underscore the urgent need for comprehensive reforms that integrate advanced technology, public awareness, and regulatory frameworks to protect the integrity of democratic processes.

Regardless of the socio-political reasons for distrust in government-run elections[13], opportunities exist to improve security and ensure the integrity of electronic voting systems [14] and thereby remove controversies about voting machine technologies from the public conversation. Research to ensure coercion-free [15] voting is fierce to help ensure trust in the technology [16]. The immutable and decentralized data storage of blockchain technology can ensure that voting records cannot be altered or manipulated [16]. Operating on a distributed network, blockchain employs cryptographic security to ensure data integrity and auditability [16]. These features make blockchain an appealing choice for e-voting, as it provides tamper-proof ballots, independent vote tallying, and a transparent and trustworthy architecture[17]. The promise of this technology is inspiring the increased adoption even of more challenging approaches to voting – online and mobile voting without the presence of traditional polling officers for the authentication of voter eligibility [17].

1.1 Problem formulation and our contributions

The core problem addressed in this study is not merely whether blockchain can be tamper-proof, but how to design a blockchain-based voting system that simultaneously achieves high scalability and post-quantum security without compromising voter privacy. Current e-voting systems face a trilemma: centralized systems lack transparency, whereas decentralized blockchains, such as Bitcoin/Ethereum lack the throughput required for national elections and are vulnerable to future quantum attacks. The research gap lies in the absence of a hybrid consensus mechanism and system architecture that jointly integrates sharding for performance, post-quantum cryptography for long-term security, and multi party computation (MPC) for privacy-preserving voter authentication.

To address these issues, we build on blockchain technology's distributed, immutable, and tamper-resistant public ledger [18] with the

goal of improving authentication, scalability, and security in electronic voting. Incidentally, the dominant public consensus mechanisms, such as Proof of Work (PoW) [19] and Proof of Stake (PoS) [20], are primarily focused on cryptocurrency applications and are significantly inefficient for e-voting. For instance, in PoW, nodes with the most mining power can manipulate votes [21]. Similarly, in POS, any node with the highest stake can influence the ballot [21]. Additionally, these public consensus mechanisms are susceptible to vulnerabilities and are not compatible with the electoral system [22].

Moreover, while private consensus mechanisms like Proof of Authority (PoA) and Proof of Credibility (PoC) offer control over the network, they are often not recommended due to their lower immutability compared to public systems, potentially introducing single points of failure. This can compromise the integrity of the system [23, 24]. To address these concerns, we propose a hybrid consensus mechanism, PQSHAC, as shown in Table 1, which combines the strengths of public and private schemes and mitigates the security and compatibility issues present in current consensus mechanisms.

Furthermore, existing e-voting systems typically adopt a cryptocurrency oriented ledger structure that is not optimized for election workloads [25]. In e-voting, most

updates occur only upon specific actions, yet each vote can generate significant redundant data. This leads to unnecessary computation, storage, and bandwidth overhead [25]. To mitigate this issue, we propose a modular ledger architecture tailored to e-voting, that captures only election-relevant state transitions while preserving verifiability and audit-ability.

In voting systems, double or multiple votes are a serious concern. Existing e-voting solutions employ measures such as national ID (NID) verification, biometric authentication, ballot tracking, and third-party server authentication [26]. However, these approaches raise serious concerns regarding security and voter privacy, as they may expose sensitive identifiers, do not adequately protect against double voting and can introduce single points of failure [27]. In contrast, our robust multi-party token generation sy'stem securely issues unlinkable voting tokens to prevent double voting while maintaining voter privacy. By providing cryptographic proof that a vote has been cast, our approach ensures that each voter can cast at most one ballot, without revealing their identity to the tallying process..

Most existing e-voting systems also suffer from severe scalability issues. Sharding techniques are considered a simple yet powerful approach to address such limitations [24]. Although some e-voting systems have introduced blockchain sharding [22, 24], they fail to exploit the discrete and geographically partitioned nature of election data. Our solution divides data based on geographic locations, and creates data-concentrated shards aligned with election regions. This facilitates efficient election *auditing* and *data forensics*. By grouping data in this manner, we minimize the cross-node communication and enhance data accessibility and throughput.

The rapid progress in quantum-scale computing introduces a new and severe security risk to most of the existing e-voting systems. As quantum capabilities advance, quantum processor can break many deployed public key encryption mechanism and compromise ballot secrecy and integrity [28]. e-voting infrastructures could fall victim to attacks using algorithms such as Grover's or Shor's [28]. To counter these threats, we incorporate post-quantum encryption, specifically CRYSTALS-Dilithium, into our e-voting system to tackle such attacks and improve long-term security.

To enhance authentication, we adapt a deep learning (DL) based face verification model for biometric authentication. Although previous studies have used face verification, they often overlooked security concerns such as potential alterations to the captured face image during transmission or storage [29–34]. To address this, we implement a unique digital signature that cryptographically binds the verified biometric data to the corresponding government-issued ID such as a national ID card or driver's license, thereby strengthening identity assurance and resistance to tampering.

Finally, to improve overall system security and mitigate SPOF risks, we distribute control among multiple entities instead of entrusting it to a single authority[27]. The GG20 method, proposed by Gennaro and Goldfeder [35], employs threshold protocols for the elliptic curve digital signature algorithm (ECDSA), requiring a minimum of t shares among n parties to produce a valid signature. In our design, the key-generation process incorporates zero-knowledge proofs (ZKPs) to verify that the set of n parties is correctly formed, thereby eliminating SPOF risks. The private key is divided into

Table 1 Comparison of different consensus mechanisms in blockchain

Mechanism	Decentralize Control	Low Latency	Flexible Trust	Multi level Access and Permission
PoW	Yes	No	No	No
PoS	Yes	Not Specified	No	No
POA	Not Specified	Yes	Yes	Not Specified
PQSHAC	Yes	Yes	Yes	Yes

n shares, and any subset of at least t parties must collaborate to generate valid authorizations, ensuring secure, jointly controlled access to critical election operations.

We summarize the contributions of this research as follows.

- We propose a hybrid consensus mechanism called **PQSHAC** for improved public transparency and accountability and address the limitations of some of the contemporary consensus mechanisms.
- The proposed category-based sharding protocol reduces skewness and creates more data-centric shards for the voting system making the overall network faster and optimized for accessing and processing information.
- The multi-party token generation protocol prevents multiple voting while protecting voter privacy and preventing SPOF.
- The modular architecture of our distributed ledger aids the verifiability and reuse of these modules in the block as needed. This reduces the storage and bandwidth usage of the blockchain network.
- Furthermore, we integrated a DeepFace-based face verification system [29] with ZKP into our blockchain platform to develop a secure, private, and efficient facial recognition system that alleviates both security and user privacy.
- We also provide a comprehensive comparative performance evaluation, demonstrating that PQSHAC outperforms traditional PoW/PoS systems and non-modular blockchain architectures in terms of throughput, latency, and storage efficiency.

The presentation of the remainder of the article is planned as follows. Section 2 reviews existing and related work in this area. Section 3 discusses the methodology of the proposed system and its objectives. Section 4 covers the proposed blockchain system's implementation, including the block structure and pseudocode of all the algorithms used in PQSHAC. Section 5 evaluates the result and the system's performance. It also analyzes the system's security and potential vulnerabilities before concluding in Section 6.

2 Related Research

Numerous prior studies have attempted to design blockchain-based e-voting systems with varying degrees of success in the axes of authentication, security, and transparency. The work on PQSHAC Blockchain we present in this paper is motivated by recent advances such as the ones in PSC-Bchain [24] and the practical systems built [36]. Kevin et al. [36] discussed blockchain voting, and how it eliminates the need for

central authority and allows anyone to verify votes. They suggested Hyperledger Fabric as a viable blockchain solution. PSC-Bchain by Yousif et al. [24] improves energy and scalability but introduces complexity (extra computing, gas fees) and relies on a centralized voter verification mechanism, raising security concerns.

Public consensus mechanisms such as PoW or PoS are not suitable for ensuring security, accountability, and adhering to regulatory restrictions in the context of elections. The PoS mechanism can be problematic due to its tendency to favor participants with larger stakes. In a PoS system, the node with a higher stake can mint new blocks faster.[20] Similarly, in PoW[19], it favors participants with larger computing power. This concentration of power can lead to election rigging and vote manipulation, where a few or unrelated foreign stakeholders have disproportionate influence over the election. While permitted consensus mechanisms offer some advantages, they fall short in terms of transparency and vulnerability to SPOF. In a PoA system, a small group of pre-selected nodes is responsible for validating transactions and maintaining the network. This centralized structure can lead to a lack of transparency as the general public may not have insight into the decision-making process or the identities of the validating nodes. This concentration of power could be exploited, leading to potential collusion or corruption[23].

Al-Zubaidie et al. [37] showed that employing microsegmentation and yellow saddle goatfish algorithm can improve transaction security and management in blockchain-based smart contract systems for e-banking. Košťál et al. [38] proposed an Ethereum blockchain-based e-voting system, claiming it offers full transparency while securing voter privacy through homomorphic encryption. However, it is worth noting that homomorphic encryption requires significant processing power and time [39], and public blockchain systems may not be fully compliant with regulatory requirements [36].

Liu et al. [40] proposed Cherubim, which uses 4P-2PC consensus to solve scalability challenges in sharded blockchains. The protocol boosts transaction throughput through pipelined cross-shard processing, optimized batching, and minimized communication. Duan et al. [41] developed FIN, featuring a constant-time Asynchronous Common Subset protocol with $O(n^3)$ message complexity and a novel Multi-Value Byzantine Agreement protocol. Liu et al. [42] introduced parallel CSBFT, a cross-shard Byzantine fault tolerance protocol, alongside a decentralized shard reconfiguration mechanism. Their approach reduces communication and computational overhead while maintaining security, validated through formal proofs and empirical testing. Duan et al. [43] presented DashingNGL and Dashing2, Byzantine fault tolerance protocols that balance performance and security. These protocols use weak certificates for efficiency and achieve single-phase consensus through strong certificates requiring $3f + 1$ signatures.

Lu et al.[44] proposed a privacy-preserving electronic voting scheme using Private Set Intersection (PSI), enabling voters to prove their eligibility and cast votes without disclosing their identities or vote to any party, including the voting system itself. We suggest a hybrid consensus approach to that has all predefined access controls to ensure compatibility and regulatory compliance with the electoral system[45, 46]. Also public access and Scalability and performance are major concerns in existing

blockchain technology. To tackle this, researchers have explored sharding techniques [24]. However, random shard-node assignment method in sharding does not provide an optimal shard distribution and results in increased cross-shard, intra-shard requests, and latency [47].

In contrast, we propose a novel token generation protocol computed by multi-party Computation (MPC) [48, 49] strategies to protect voter privacy and tackle issues during authentication. Existing approaches' reliance on 3rd party NID or Voter ID (VID)[1] server for voter authentication introduces SPOF, cyber threat, and risk to voter privacy. Li et al. [50] proposed a post-quantum blockchain with segregation witness, which can effectively increase the proportion of signatures in block size based on the hardness assumption of Short Integer Solution (SIS) to improve security. The challenges of current state-of-the-art post-quantum crypto-systems and their possible application to blockchains and DLTs are being investigated. Allende and his team [28] proposed a two-layer solution to secure the exchange of information between blockchain nodes over the internet and introduced a second signature in transactions using post-quantum keys that can be applied to any blockchain network. Sun et al. [51] and his companion propose a simple voting protocol based on Quantum Blockchain. They utilize sets of cryptographic primitives develop a protocol named Bit commitment and use quantum Byzantine agreement (QBA) to achieve consensus between mining nodes. Deepface [29] is a lightweight face recognition and facial attribute analysis framework based on deep learning techniques to recognize and authenticate individuals based on their facial features. This technology has been widely used in various applications such as security systems, biometric authentication, video surveillance, and social media. One approach to enhance the security and tamper-proofing face recognition systems is by using blockchain technology to store face images[52]. Blockchain's immutability and distributed storage provide a secure and accessible platform for storing large amounts of face data [53, 54].

However, these proposed methods, despite their strengths, face significant challenges in crucial areas including authentication, security, auditing, forensics, scalability, and regulatory compliance. This implies that these are not yet practical or secure enough for real-world elections. Furthermore, many researchers and civic organizations [55] raise concerns about election integrity solely based on the limitations of the blockchain and associated technologies [56]. Yet, a significant number of research suggests that approaches based on blockchain could be a breakthrough [57–59] toward restoring trust in the system.

Table 2 provides an comparative analysis of various blockchain-based e-voting systems, including existing work proposed by others and our current study. The comparison is organized into four main categories: *Security Features*, *Consensus and Architecture*, *Verification and Trust*, and *Advanced Features*.

Although most systems aim for secure e-voting based on various blockchain technologies and have achieved a level of security against electronic and paper ballot voting systems, many of them still have fatal security flaws, serious issues with scalability and efficiency, and problems with regulatory compliance. Additionally, there

[1]We use both NID and VID as generic terms for individual identification of eligible voters in a non-country-specific manner.

Table 2 Comparison with contemporary works.

Properties	[24]	[26]	[51]	[60]	[61]	[62]	[63]	[64]	**Ours**
Security Features									
Security	✓	✓	✓	✓	✓	✓	✓	✓	✓
Robustness	✓	⊛	⊛	✓	⊛	⊛	⊛	✓	✓
Confidentiality	✓	⊛	⊛	✓	⊛	⊛	✓	✓	✓
Post-Quantum Security	⊛	⊛	✓	⊛	⊛	⊛	⊛	⊛	✓
Consensus and Architecture									
Consensus Mechanism	PSC	PoS	QBA	PoS	PoS	PoS	PoS	PoW	PQSHAC
Sharding Support	✓	⊛	⊛	⊛	⊛	⊛	⊛	⊛	✓
Category-Based Sharding	⊛	⊛	⊛	⊛	⊛	⊛	⊛	⊛	✓
Block Modularity	⊛	⊛	⊛	⊛	⊛	⊛	⊛	⊛	✓
Verification and Trust									
Eligibility	✓	⊛	✓	✓	⊛	⊛	⊛	⊛	✓
Verifiability	✓	✓	✓	✓	⊛	✓	✓	✓	✓
Uniqueness	✓	✓	✓	✓	⊛	✓	✓	✓	✓
Transparency	✓	✓	✓	✓	✓	✓	✓	✓	✓
Advanced Features									
MPC Token Support	⊛	⊛	⊛	⊛	⊛	⊛	⊛	⊛	✓
Deepface Integration	✓	✓	⊛	⊛	⊛	⊛	⊛	⊛	✓
Time-based Inference	⊛	⊛	⊛	⊛	⊛	⊛	⊛	⊛	✓
Unlinkability	⊛	⊛	⊛	✓	⊛	⊛	✓	⊛	✓

[24] Abuidris et al. [26] Neloy et al. [51] Sun et al. [60] Khoury et al.
[61] Anitha et al. [62] Singh et al. [63] Ch et al. [64] Bhadoria et al.

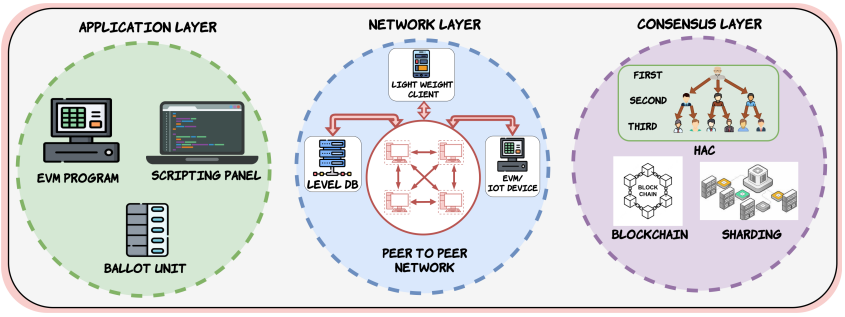

Fig. 1 Shows the three-layer node architecture of the proposed system: application, consensus, and network.

is a concerning trend to develop e-voting systems based on cryptocurrency networks, which theoretically can be influenced and are not secure enough for sensitive electoral usage. The proposed PQSHAC system bridges these gaps by integrating a hybrid consensus model for speed, post-quantum algorithms for future-proof security, and an MPC-based token system to eliminate centralized bottlenecks.

3 Methodology

The proposed voting system addresses the critical aspects of secure and fair voting. It is decentralized, thus ensuring security and cost-effectiveness, as shown in Figure 1. In addition, it is designed to be voter-verifiable, auditable, anonymous, fair, and user-friendly, thus providing all voters with an equal opportunity to cast their ballots.

In our voting system, we authenticate the identity of each voter by using biometrics and facial recognition. Furthermore, we verified the validity of the user information retrieved from the NID database through a secure cryptographic signature mechanism. Moreover, we developed a multi-party secured cryptographically secure token verification system to prevent fraudulent and double voting. This system verifies whether a valid voter has already cast a ballot while protecting their identity and privacy.

3.1 System Design

The backbone of our main voting system is a backend server application that oversees the majority of the blockchain and e-voting operations. This server is responsible for storing a copy of the blockchain or its shards in the LevelDB database. Figure 2 provides an overview of the entire voting process. The EVM system comprises several components as follows:

3.1.1 System

This system handles communication with the peer-to-peer network, manages the storage and serving of the blockchain from a local database, and performs various operations on blockchain data, including script execution. External entities such as the MPC Token generation, decentralized Key Management System (KMS) server,

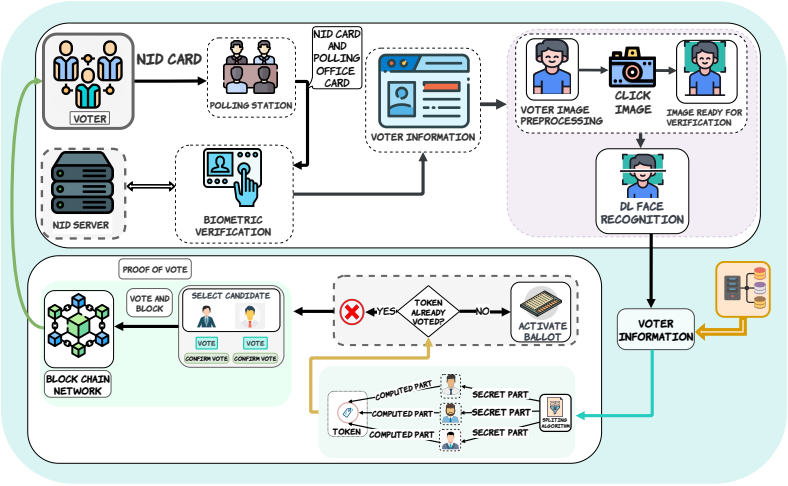

Fig. 2 Overview of the proposed e-voting system. The electronic voting process begins with voter identification using NID cards and biometric verification, including face recognition. Once verified, the ballot is activated, allowing the voters to cast their vote securely. The splitting algorithm generated unique tokens to prevent duplicate votes. Votes are securely stored in a blockchain network to ensure transparency and security. Finally, proof of vote is generated to verify the authenticity of the vote.

and NID servers provide tokens, keys, signatures, and voter information to support the election process. Users, including voters, returning officers, and polling officers, can access the system through a user interface.

3.1.2 External Entities

The proposed e-voting system integrates external services like NID servers, KMS servers, and token generation systems to ensure a secure, transparent, and reliable voting process. NID cards and servers authenticate voters and verify their eligibility, as shown in Figure 2.

3.1.3 Participants

The primary participants in the blockchain voting system are the election commission, returning officers, polling officers, and voters. Each participant, except for the candidates, has a dedicated user interface panel tailored to their access level and permissions, facilitating efficient task execution and seamless communication across various blockchain systems.

3.1.4 Election Creation

Authorized personnel use their private cryptographic keys through a secure management interface, as shown in Figure 3. The signed poll script containing crucial information is then published on the blockchain. The nodes verify the signature and script to ensure the poll's legitimacy.

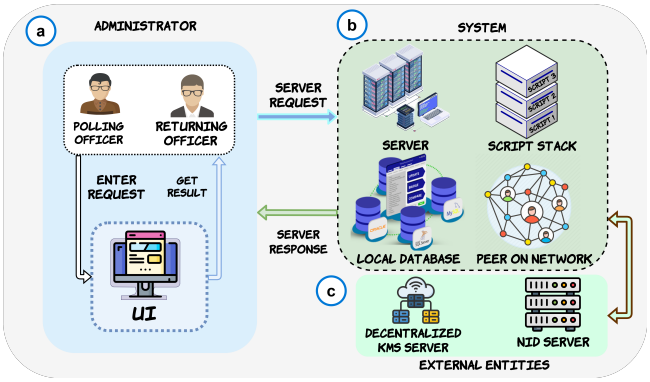

Fig. 3 Overview of Administration Panel showing (a) Administrator interface with Polling and Returning Officers accessing the system through UI, (b) System components including servers, local database, and peer network infrastructure, and (c) External entities with decentralized KMS and NID servers.

Once created, the poll can begin and polling officers can begin their duties. The hash of the poll data is stored in the blockchain, ensuring tamper-proofing and enabling accurate vote counting and legitimate election results.

3.2 Hierarchical Authoritative Consensus Model

We protect the fairness and security of our election process by setting up a hierarchical authorization system and following strict access controls and guidelines established by a higher governing body. A designated committee, similar to real-world election commissions, will assign specific roles like returning officers and panels for a particular election area using digital signatures and an access control tree. Every authorized entity within the network will adhere to the PQSHAC protocol. Any node attempting to tamper with data or deviate from the protocol will be automatically disconnected from the network, preserving the system's integrity. The PQSHAC protocol ensures consensus and a fair election process in a public network.

The protocol offers a strong framework for managing access and permissions, defining the roles and responsibilities of each authorized entity. This ensures that only those with the appropriate clearance can perform specific tasks. The hierarchical structure is similar to traditional election commissions, ensuring compliance with regulations and providing a familiar, trusted framework for secure and transparent elections. By combining digital signatures with the access control tree, we create a secure and verifiable trail for all actions within the network. This allows us to quickly identify and address any attempts to manipulate data or disrupt the process, upholding the election's integrity. Our model proposes multilevel access and authorization in the consensus mechanism, with each entity involved in the voting process having predefined access and a specific purpose within the blockchain (as shown in Figure 4).

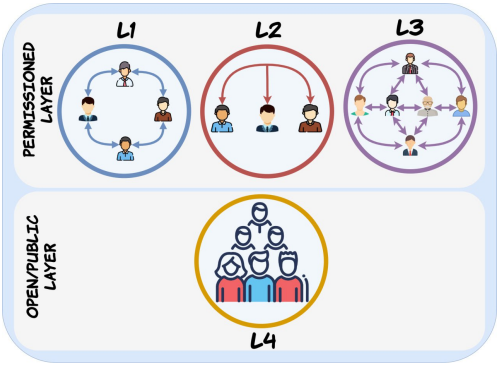

Fig. 4 This diagram visually explains the PQSHAC algorithm, which combines both public and permitted features of blockchain. It shows the various layers of permission and access for each entity in the network, making PQSHAC a hybrid consensus algorithm.

- **Election Management Layer:** $L_1 = \{a_1, a_2, ..., a_n\}$, where a_i represents an election commission responsible for election creation, administrator assignment/removal, and taking emergency actions such as halting or reorganizing elections to ensure integrity and security.
- **Administration Layer:** $L_2 = \{r_1, r_2, ..., r_m\}$, where r_i represents a returning officer or election administrator responsible for managing block validators, allowing administrators to assign or remove polling officer, publish results, and halt elections in emergencies, ensuring integrity and security.
- **Block Validation Layer:** $L_3 = \{v_1, v_2, ..., v_k\}$, where v_i represents a polling officer, allowing them to validate new blocks using encryption and access controls.
- **Public Layer:** L_4 Open to everyone, enabling data transfer and observation.

For instance, in our implementation of the consensus mechanism, each polling station intended to have multiple polling officers. These officers are defined as block validators within the system authorized by the returning officers. Returning officers, in turn, act as administrators responsible for overseeing specific areas and polling stations within them. They have the authority to assign and suspend block validators from their area, contributing to ensuring the security and fault tolerance of voter tokens. This multilayered system of permission and access helps ensure the integrity and security of the voting process. By utilizing the MPC protocol, cryptographic commitment, ZKPs, and threshold cryptography, our proposed model fortifies e-voting systems with increased reliability, security, and user confidence.

3.2.1 Authorization

The authorization process can be described as follows:

1. **First Layer (L_1) Authorization:** - Power of L_1 nodes: $P(a_i) = \{assign(r_j), remove(r_j), suspend(election), halt(election), reorganize(election)\}$, where *assign* and *remove* refer to adding

or removing returning officers, and *suspend*, *halt*, and *reorganize* are emergency powers.

2. **Second Layer (L_2) Authorization:** - Power of L_2 nodes: $P(r_i) = \{assign(v_j), remove(v_j), token\_generation\}$, where *assign* and *remove* refer to managing block validators, and *token_generation* represents their participation in token creation.
3. **Third Layer (L_3) Authorization:** - Responsibilities of L_3 nodes: $R(v_i) = \{identify\_voter(V), record\_vote(V), generate(block), validate(block)\}$, where *identify_voter* ensures proper voter identification, *record_vote* accurately records each vote, and *generate* and *validate* refer to block generation and validation, respectively.
4. **Fourth Layer (L_4) Public Accessibility:** - L_4 nodes can verify any block: $V(b) = \{verify(b) \mid b \in \text{Blocks}\}$, where *verify* ensures the integrity of each block. - Transparency: $T(L_4) = \{transparency \mid \forall b \in \text{Blocks}\}$, where *transparency* ensures public accessibility and accountability.

The entire process ensures a secure and transparent election. Votes are recorded in blocks and authorized by trusted validators from the third layer (L_3), ensuring the accuracy and validity of each vote. The signing mechanism provides traceability, allowing public nodes to confirm the authenticity of votes. The adaptability of the mechanism is achieved through the addition of layers and variation of permissions/access rights.

3.2.2 Verification

Let's consider a blockchain network with N nodes, where each node i has a unique private key K_i and a corresponding public key P_i.

1. **Proof of Validity:** For each received block, data, or script, a proof of validity is provided. This proof can be in the form of a digital signature or a cryptographic hash function.
   - Digital Signature: Each node i signs the data D using its private key K_i to create a signature $S_i = \text{Sign}(K_i, D)$. The corresponding public key P_i can then be used to verify the signature: $Verify(P_i, S_i, D) \stackrel{?}{=} \text{true}$.
   - Cryptographic Hash Function: A hash function H is used to create a unique fingerprint of the data D. The hash value $H(D)$ is then compared to a known good hash value to verify the data's integrity.
2. **Trusted Authorities and Public Keys:** Trusted authorities, denoted as $TA_1, TA_2, ..., TA_m$, possess their own private and public key pairs. These authorities share their public keys $P_{TA_1}, P_{TA_2}, ..., P_{TA_m}$ with all nodes in the network. These public keys are used to verify the validity of information accessible at authorized nodes or decentralized Key Management Server (KMS) servers.
3. **Node Verification:** When a node i receives new information (block, data, or script) from another node j, it performs the following verification steps:
   - Node i retrieves the public key P_j of node j.

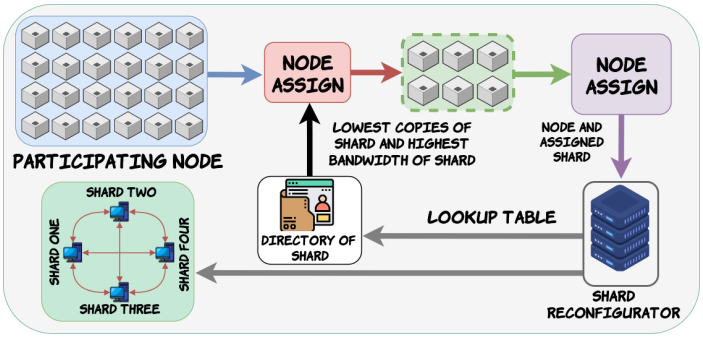

Fig. 5 This diagram shows the shard assignment process for a specific category and node. The shard re-configurator is used to find the shard for a specific category, and the coordinator is responsible for managing the shard. It assigns the node to a particular shard based on the lowest number of shard copies in the network or the highest amount of traffic on a shard.

- Node i verifies the authenticity of the received information by using the public key P_j to verify the signature or hash value associated with the data.
- If the verification fails, node i rejects the information and may flag node j as potentially malicious.

4. **Detection and Isolation of Malicious Nodes:** If a malicious node M attempts to tamper with blockchain data, its actions are detected through cryptographic verification by other nodes.

   - Let's assume node M modifies data D and creates a new signature $S'_M = \text{Sign}(K_M, D')$.
   - When other nodes receive this tampered data, they use the public key P_M of node M to verify the signature: $Verify(P_M, S'_M, D')$.
   - Since the data has been modified, the verification will fail: $Verify(P_M, S'_M, D') \neq \text{true}$.
   - Nodes detecting this discrepancy will identify node M as malicious and isolate it from the network, preventing further harm.

3.3 Incorporating Category-based Sharding in Blockchain Network

We propose a category-based sharding approach as illustrated in Figure 5.

$$\text{Shard Generation:} \quad S_i = f(PS_j) \tag{1}$$

Where S_i represents a shard, PS_j represents a polling station, and f is a function that generates shards based on polling stations.

$$\text{Shard Grouping:} \quad G_k = g(S_i, EA_l) \tag{2}$$

Where G_k represents a group of shards, EA_l represents an election area, and g is a function that groups shards within the same node according to their election area.

$$\text{Lookup Table: } LT = h(N_m, S_i) \tag{3}$$

Where LT represents the lookup table, N_m represents a node, and h is a function that maintains a record of the connected nodes and their corresponding shards.

$$\text{Data Retrieval: } D_n = r(G_k, N_m) \tag{4}$$

Where D_n represents the required data, and r is a function that retrieves data from the appropriate group of shards and nodes.

$$\text{Cross-Node Communication: } C = \sum_{i=1}^{N} \sum_{j=1}^{M} c(N_i, N_j) \tag{5}$$

Where C represents the total cross-node communication, N represents the total number of nodes, M represents the total number of connected nodes, and c is a function that calculates the communication between two nodes.

$$\text{Data Accessibility: } A = \frac{1}{N} \sum_{i=1}^{N} a(D_i) \tag{6}$$

Where A represents the average data accessibility and a is a function that calculates the accessibility of data D_i.

This strategy creates more data-concentrated shards based on their categories, facilitating more efficient election auditing and data forensics. The grouping minimizes cross-node communication and enhances data accessibility. Consequently, users can quickly locate and access the required data, leading to improved scalability, performance, and availability of blockchain networks.

3.4 Token-Based Voter Verification System

Addressing the challenge of double voting is crucial to maintaining the integrity of voting systems. Double voting occurs when an individual casts multiple votes in the same election either intentionally or accidentally. To prevent such occurrences and ensure fair elections, we implement a cryptographically secure unique token generation and verification system.

Voters and Elections: Let V denote the set of all registered voters, where each voter $v_i \in V$ has a unique identity id_i. Let E denote the set of all elections, where each election $e_j \in E$ is associated with a specific time period.

Double Voting: Double voting occurs when a voter casts multiple votes in the same election. Formally, let $B_{i,j}$ denote the set of ballots cast by voter v_i in election e_j. Then, double voting occurs if $|B_{i,j}| > 1$.

Token Generation: During an election e_j, the system generates a unique temporary token $t_{i,j}$ for each voter v_i. The token is generated using a MPC protocol involving n parties, denoted as $P = \{P_1, P_2, \ldots, P_n\}$. Each party P_k has a private input x_k, and the token $t_{i,j}$ is computed as a function f of these inputs and the voter's identity id_i:

$$t_{i,j} = f(id_i, x_1, x_2, \ldots, x_n)$$

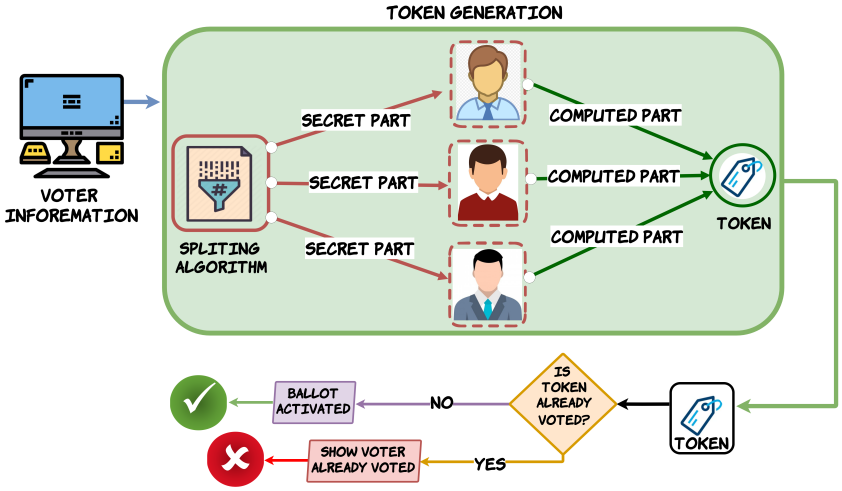

Fig. 6 The MPC token generation process is a secure and efficient way to generate tokens that can be used to verify that a voter is eligible to vote and to prevent voter fraud. The process works by taking the voter's unique identifier as input, splitting it into multiple parts, encrypting each part, and then having each encrypted part computed by a different party. The computed parts are then merged to form a token.

The MPC protocol ensures that no single party can learn the private inputs of other parties, even in collusion.

Token Verification: When a voter v_i casts a ballot $b \in B_{i,j}$ in election e_j, they submit their token $t_{i,j}$ along with the ballot. The system verifies the token by checking whether it is valid for the given voter's identity id_i and has not been previously used in the same election. This can be represented as a function g:

$$g(id_i, t_{i,j}, e_j) = \begin{cases} 1, & \text{if } t_{i,j} \text{ is valid and not previously used in } e_j \\ 0, & \text{otherwise} \end{cases}$$

Preventing Token Reuse: Once the election e_j is concluded, the system prevents the regeneration of the same token for the same voter. This can be achieved by maintaining a list L_j of used tokens for each election, and ensuring that no token appears more than once in the list:

$$\forall t_{i,j}, t'_{i,j} \in L_j : t_{i,j} \neq t'_{i,j}$$

Security and Integrity: The proposed system enhances security and effectively prevents instances of double voting, thereby safeguarding the electoral process's integrity. By using MPC protocols for token generation, the system ensures voter anonymity while preventing double voting. Additionally, the prevention of token reuse ensures that each voter can cast at most one vote per election. Figure 6 illustrates

a token-based verification system, which offers several advantages over traditional voting systems.

1. The token verification system ensures the anonymity of voters and safeguards their identities.
2. The unique token enables the verification of vote authenticity while preserving voter privacy.
3. Voters can conveniently verify cast votes using this unique token.

The token is generated exclusively using the voter's identity and the secret key during the election. Once the election is concluded, the system prevents regeneration of the same token for the same voter. This approach enhances security and effectively prevents instances of double voting, thereby safeguarding the electoral process's integrity.

3.5 Modular Ledger Structure

We propose a modular architecture for our distributed ledger, in which each module within the block is individually verifiable and reusable. This modular approach eliminates the storage of redundant information by breaking down data into smaller reusable modules.

This architecture optimizes bandwidth usage by allowing only the relevant module to be accessed for transaction verification rather than transmitting the entire block. Consequently, network traffic is minimized, thereby enhancing the transaction processing speed.

Moreover, the modular approach enhances data security and integrity through the PQSHAC consensus. Each module was individually verifiable, enabling participants to confirm their authenticity and immutability based on their respective authorized layers. This decentralized verification system mitigates the risk of tampering or manipulation, thereby strengthening the security and flexibility of the entire blockchain.

Interpretation of Modular Architecture:

Let's consider a distributed ledger with N blocks, where each block B_i contains M modules, denoted as $M_1, M_2, ..., M_M$. Each module M_j represents a subset of data within the block.

The proposed modular ledger architecture can be described as follows:

1. **Individual Verifiability:** Each module M_j within a block B_i is individually verifiable. This means that the integrity and authenticity of each module can be independently confirmed.
   - Verification function: $V(M_j) \rightarrow \{\text{true}, \text{false}\}$, where V is a verification function that checks the validity of the module's content.
   - Individual verification: $V(M_j) = \text{true} \quad \forall j \in [1, M]$.
2. **Reusability:** The modules are designed to be reusable, meaning that the same module can be used or referenced in multiple blocks.

- Reuse function: $R(M_j) \rightarrow \{0, 1\}$, where R checks if the module M_j has been reused in another block.
- Reuse condition: $R(M_j) = 1$ if M_j is reused, $R(M_j) = 0$ otherwise.

3. **Redundancy Elimination:** The modular approach eliminates the storage of redundant information.
   - Redundancy elimination: $E(B_i) = \sum_{j=1}^{M} R(M_j)$, where $E(B_i)$ represents the eliminated redundancy in block B_i.
   - Storage optimization: $S(B_i) = \sum_{j=1}^{M} |M_j| - E(B_i)$, where $S(B_i)$ is the optimized storage size of block B_i.
4. **Bandwidth Optimization:** By allowing access to specific modules for transaction verification, bandwidth usage is optimized.
   - Bandwidth usage: $BW(B_i) = \sum_{j=1}^{M} |M_j| \cdot I(M_j)$, where $BW(B_i)$ is the bandwidth required to transmit block B_i, and $I(M_j)$ is an indicator function that equals 1 if module M_j is accessed for verification.
5. **Transaction Processing Speed:** The modular architecture enhances transaction processing speed by minimizing network traffic.
   - Network traffic: $NT(B_i) = \sum_{j=1}^{M} |M_j| \cdot A(M_j)$, where $NT(B_i)$ is the network traffic generated by block B_i, and $A(M_j)$ is an indicator function that equals 1 if module M_j is accessed.
   - Processing speed: $PS(B_i) \propto \frac{1}{NT(B_i)}$, where $PS(B_i)$ is the transaction processing speed for block B_i.
6. **HAC Consensus and Security:** The modular approach enhances data security and integrity through Hierarchical Authorization and Control.
   - Individual module verification: $V_{PQHAC}(M_j) \rightarrow \{\text{true}, \text{false}\}$, where V_{HAC} is the PQSHAC-based verification function.
   - Module immutability: $I(M_j) = V_{PQHAC}(M_j) \quad \forall j \in [1, M]$, where $I(M_j)$ ensures the immutability of module M_j.
   - Security enhancement: The decentralized PQSHAC verification mitigates tampering or manipulation, strengthening the security of the blockchain.

In summary, this modular architecture optimizes storage, bandwidth usage, and transaction processing speed. The individual verifiability and reusability of modules, combined with the PQSHAC consensus, enhance data security, integrity, and flexibility within the distributed ledger system.

3.6 Incorporating Post-quantum Asymmetric Encryption Blockchain

To ensure transparency and secure authorization in our PQSHAC model, we rely heavily on public-key cryptography, digital certificates, and hash functions. Although conventional public-key cryptography has been effective, the rapid progress of quantum computing poses a potential threat in the future. Quantum algorithms, such as

Grover's and Shor's algorithms, can compromise both public-key cryptography and hash functions, necessitating a redesign of blockchains to use cryptographic systems resistant to quantum attacks.

$$\text{Public-Key Cryptography: } (PK, SK) = g(k) \tag{7}$$

Where PK represents the public key, SK represents the private key, k represents the security parameter, and g is a key generation function.

$$\text{Digital Certificate: } C = (PK, ID, sig_{CA}(PK, ID)) \tag{8}$$

Where C represents the digital certificate, ID represents the identity of the user, and sig_{CA} is a signature function of the certificate authority.

$$\text{Hash Function: } h = H(m) \tag{9}$$

Where h represents the hash value, m represents the message, and H is a hash function.

$$\text{Grover's Algorithm: } q_G(x) = U_D U_{f,x} O_0 U_{f,x} |0\rangle \tag{10}$$

Where q_G represents Grover's algorithm, x represents the input, U_D is a diffusion operator, $U_{f,x}$ is an oracle that computes the function $f(x)$, and O_0 is a reflection about the mean operator.

$$\text{Shor's Algorithm: } q_S(x) = QFT^{-1} U_f QFT |x\rangle \tag{11}$$

Where q_S represents Shor's algorithm, x represents the input, QFT is the quantum Fourier transform, U_f is a unitary operator that computes the function $f(x)$, and QFT^{-1} is the inverse quantum Fourier transform.

$$\text{Quantum-Resistant Cryptography: } (PK', SK') = g'(k') \tag{12}$$

Where PK' represents the quantum-resistant public key, SK' represents the quantum-resistant private key, k' represents the security parameter for quantum-resistant cryptography, and g' is a key generation function for quantum-resistant cryptography.

$$\text{Quantum-Resistant Signature: } sig'(m) = s'(h', SK') \tag{13}$$

Where sig' represents a quantum-resistant signature, m represents the message, h' represents the quantum-resistant hash value, and s' is a signature function for quantum-resistant cryptography.

$$\text{Quantum-Resistant Hash Function: } h' = H'(m') \tag{14}$$

Where h' represents the quantum-resistant hash value, m' represents the message, and H' is a quantum-resistant hash function.

In response, we implemented and tested Dilithium [65], a post-quantum asymmetric encryption algorithm, on our blockchain. Dilithium provides essential operations such as digital signatures, authentication, and data integrity. It is one of the algorithms selected by the National Institute of Standards and Technology (NIST) for

post-quantum proof standardization [66]. In our implementation, we tested two versions of Dilithium2 and Dilithium3. Dilithium3 offers higher security than Dilithium2, but Dilithium2 boasts faster signing speeds and produces fewer public keys and signatures [67].

To avoid key escrow risks, our system lets each polling station generate their private key locally on their own evm machine, and this key never leaves the device. For critical system keys, we use GG20 threshold cryptography, which splits a private key into n shares and requires collaboration from at least t of those shares, so no single entity can control the key alone. The EC only signs the RO public key and never has access to the private key. We reduce overhead with a hierarchical certificate structure where top-level ECs issue long-term certs to mid-level ROs, who then issue short-lived, session-specific PO certificates. Certificate revocations are recorded on the blockchain, eliminating separate lists. We use strong but smaller, quantum-resistant Dilithium-based certificates and cache frequently used public keys within shards to speed up verification.

4 Implementation

We developed a blockchain-based e-voting system from scratch, without utilizing any pre-existing blockchain libraries or frameworks. Our system comprises multiple components, including the network, execution, and consensus stacks.

The Network Stack manages the underlying infrastructure of the system, handling tasks, such as Script Fetching, Block Gossiping, and Block Propagation. It also maintains a Lookup Table for all connected intra-shard nodes and cross-shard nodes and facilitates data fetching from the P2P network.

The functionalities of the Execution Stack vary depending on client type. Upper-layer clients can create, manage, and launch elections along with additional functionalities based on their permissions. This stack also supports the EVM units and Ballot units. By contrast, public nodes are responsible for fetching, auditing, and providing data to other nodes, lightweight clients, and general users.

The Consensus Stack consists of three main components:

- Consensus mechanism: Nodes across different layers perform agreed-upon tasks according to their permission and access levels.
- Incentive rules: Nodes failing to perform designated tasks or engaging in malicious activities face penalties and potential isolation from the network. This incentivizes honest behavior and ensures network integrity.
- Validation rules: Any node can verify the validity of blocks, commands, transactions, and relevant records within the system.

We developed a system using Protocol Buffers, aiming to create a platform-neutral, extensible, and serialized data structure for the distributed ledger. To ensure security, we employed the SHA256 and Blake2b cryptographically secure hash functions. Base58, a binary encoder, was used to generate append-only timestamped blocks. For asymmetric cryptography and digital signature schemes, Cloudflare's Interoperable, Reusable Cryptographic Library (CIRCL) was employed, incorporating algorithms

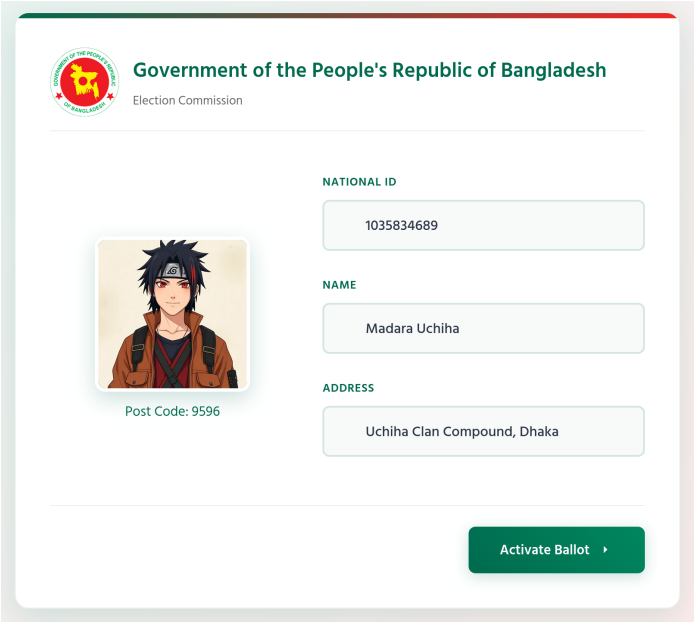

Fig. 7 The Visual representation of the EVM system Displays voter information, including names, addresses, and voter ID numbers. The system stores, manages, and serves voter data securely. It also provides Digest, QR code, and Digital Signature for tamper-proof verification.

such as Ed25519, Dilithium2, and Dilithium3, making the blockchain verifiable and immutable. The Kademlia Distributed Hash Table (DHT) subsystem in libp2p was implemented for the peer routing interface. For multi-party access and control we used t-of-n threshold ECDSA signing algorithm using GG20 with Kryptology [68] A secure token-based verification system was implemented to address the double-spending voting problem using Universally Unique Identifiers (UUID) and ZKP. This approach ensures transparency, privacy, and resistance to tampering. Level-DB was chosen for storing the blockchain, along with associated signatures, public keys, tokens, trie structures, binaries, and other relevant data, providing a fast key-value storage solution. For face verification, a deep face, hybrid face recognition, and facial attribute analysis framework for Python were utilized. The backend was built using the GIN web framework, offering a REST API to handle all requests and execute the appropriate actions. This backend seamlessly communicates with the application layer, network layer, script processing engine, and blockchain system. Additionally, a NID server was implemented using GIN and SQLite3, providing verifiable and tamper-proof data for voters and facilitating testing in a real-world scenario.

Figure 7 shows that the NID server provides the necessary infrastructure for verifying the voter identity and ensuring the integrity of the voting process.

4.1 Implementation of the Proposed Algorithm in the System

In this section, we describe the essential functions to implement our proposed algorithms.

4.1.1 Authorization Algorithm

The proposed layer-based PQSHAC consensus system requires authorization of lower-level entities by obtaining permission from higher levels. To fulfill this regulatory requirement, we implemented an authorization algorithm. Algorithm 1 facilitates multi-party signing of entities (e.g., assigning a returning officer for a specific area) using Shamir's Secret Sharing scheme with partial private keys. Each of the n parties holds a share of the private key, and at least t parties must participate to reconstruct a valid signature. The signing process involves each party generating a random nonce k_i, computing the elliptic curve point $(k_i G)_x$, and calculating their partial signature using their private key share. The coordinator then aggregates these partial signatures from at least t parties to produce the final signature (r, s), which authorizes the entity.

Algorithm 1 Multi-Party Authorization Algorithm

Input: Message to sign (msg), threshold t, total parties n, partial private keys $\{sk_1, sk_2, \ldots, sk_n\}$
Output: Signature (r, s)
$h \leftarrow \text{hash}(msg)$
for $i \leftarrow 1$ to n **do**
    $k_i \leftarrow$ party i generates random number
    $r_i \leftarrow (k_i G)_x$
    $s_i \leftarrow (h + r_i \cdot sk_i) \cdot k_i^{-1}$
    send (r_i, s_i) to coordinator
end for
▷ Coordinator reconstructs signature
$r \leftarrow \sum_{i=1}^{n} r_i$
$s \leftarrow \sum_{i=1}^{t} s_i$
return (r, s)

4.1.2 Proposed Vote Casting Algorithm

Algorithm 2 describes the vote casting process, which ensures voter authentication, prevents double voting, and securely records votes on the blockchain. The algorithm first communicates with the National Identity Database (NID) server to verify the voter's credentials using their NID and biometric data. Upon successful validation, a unique identifier is generated for the voter, from which a cryptographic token is derived through multi-party computation (MPC). This token is checked against the spent token tree a data structure maintaining records of all tokens used for voting. If the token has already been spent, indicating the voter has previously cast their

ballot, the system rejects the attempt to prevent double voting. Otherwise, the vote is encrypted, signed with the voter's private key, and appended to the blockchain as a new block, with the token marked as spent in the token tree.

Algorithm 2 Vote Casting Algorithm

Input: Voter NID ($VoterNID$), Biometric data ($BiometricData$), Voter's private key ($PrivateKey$)
Output: Vote casting status (Success/AlreadyCast/Invalid)
Block ← new BlockStruct
VoterData ← NIDServer.VerifyVoter(VoterNID, BiometricData)
if VoterData ≠ Null **then** ▷ Voter is validated
    UniqueID ← GenerateUniqueID(VoterData)
    Token ← MPC.GenerateToken(UniqueID)
    if TokenTree.IsTokenSpent(Token) = False **then**
        Block.vote ← GetVote(seed, keyp, Randp)
        Block.Signature ← SignBlock(Block, PrivateKey)
        TokenTree.MarkTokenAsSpent(Token)
        AppendBlockToChain(Block)
        return VoteCastSuccess
    else
        return VoteAlreadyCast
    end if
else
    return VoterNotValid
end if

4.1.3 Ballot Tallying

After the election concludes, votes must be tallied securely while maintaining transparency. Algorithm 3 describes the vote tallying process, which iterates through all blocks in the blockchain to extract and decrypt votes using multiple decryption layers. Each vote undergoes multiple decryption operations to reverse the layered encryption applied during casting. The algorithm then matches each decrypted vote against the candidate list to identify the selected candidate and increments their vote count accordingly. The `GetVotes` function extracts votes from each block and performs the necessary decryptions, while the `GetCandidate` function maps each vote to the corresponding candidate. All cryptographic proofs, random functions, and encryption keys are made public after the election, enabling independent verification by any interested party. This transparent and verifiable process ensures trust in the election results while maintaining voter anonymity through cryptographic protection.

The proposed system leverages ZKPs for transparency and voter anonymity. ZKPs allow the verification of individual votes without revealing their content, ensuring that everyone can confirm the election's fairness. Additionally, cryptographic methods, such

Algorithm 3 Ballot Tallying Algorithm

Input: Blockchain (*blockchain*), Candidate list (*CandidateList*)
Output: Vote tally results (*result*)
function TallyVotes(*blockchain*, *CandidateList*)
    $votes \leftarrow \emptyset$
    $result \leftarrow \emptyset$
    for all $block \in blockchain$ **do**
        $votes \leftarrow votes \cup GetVotes(block)$
    end for
    for all $vote \in votes$ **do**
        $candidate \leftarrow GetCandidate(vote, CandidateList)$
        $result[candidate] \leftarrow result.get(candidate, 0) + 1$
    end for
    return *result*
end function
function GetVotes(*block*)
    $votes \leftarrow block.vote$ ▷ Extract encrypted votes
    for all $vote \in votes$ **do**
        $vote \leftarrow DecryptMultipleTimes(vote)$ ▷ Reverse layered encryption
    end for
    return *votes*
end function
function GetCandidate(*vote*, *CandidateList*)
    for all $candidate \in CandidateList$ **do**
        if $vote = candidate$ **then return** *candidate*
        end if
    end for
    return *null* ▷ Handle invalid votes
end function

as hashing, encryption, and digital signatures, safeguard vote integrity and prevent tampering.

4.1.4 Token Generation

Algorithm 5 implements secure token generation using zero-knowledge multi-party computation (zk-MPC). This approach enables multiple parties to collaboratively compute a token based on the voter's unique identifier without revealing the voter's identity to any individual party. The algorithm first partitions the voter's secret identifier into n parts using the `SliceFunction`. Each participating party then independently computes their share through the `MultyPartyComputation` function by generating a random number, XORing it with the voter's seed (if within valid election time), and combining it with their assigned secret part. Each computed share is hashed and digitally signed with the party's private key to ensure integrity. The `MergeShares` function verifies the signature of each share using the corresponding

Algorithm 4 Detailed Vote Tallying with Encryption Layers

Input: Blockchain (*blockchain*), Candidate list (*CandidateList*), Decryption keys (*BallotKey*, *keyp*, *keyr*, *keye*)
Output: Vote count per candidate (*result*)
function TallyingVotes(*blockchain*, *CandidateList*)
    votes ← empty list
    for each *block* in *blockchain* **do**
        blockVotes ← GetVotes(*block*)
        votes ← *votes* ∪ *blockVotes*
    end for
    for *i* ← 0 to *len*(*votes*) **do**
        for *j* ← 0 to *len*(*CandidateList*) **do**
            if *votes*[*i*] = *CandidateList*[*j*] **then**
                votes[*i*] ← *CandidateList*[*j*]
            end if
        end for
    end for
    result ← CountVotes(*votes*)
    return *result*
end function
function GetVotes(block)
    votes ← getVotesFromBlock(*block*)
    for *i* ← 0 to *len*(*votes*) **do**
        votes[*i*] ← decrypt(*votes*[*i*], *keye*) ▷ Layer 4 decryption
        votes[*i*] ← decrypt(*votes*[*i*], *keye*) ▷ Layer 3 decryption
        votes[*i*] ← decrypt(*votes*[*i*], *keyr*) ▷ Layer 2 decryption
        votes[*i*] ← decrypt(*votes*[*i*], *keyp*) ▷ Layer 1 decryption
        votes[*i*] ← decrypt(*votes*[*i*], *BallotKey*) ▷ Base decryption
    end for
    return *votes*
end function
function getVotesFromBlock(block)
    return *block.vote*
end function
function CountVotes(*votes*)
    result ← empty map
    for *i* ← 0 to *len*(*votes*) **do**
        if *votes*[*i*] in *result* **then**
            result[*votes*[*i*]] ← *result*[*votes*[*i*]] + 1
        else
            result[*votes*[*i*]] ← 1
        end if
    end for
    return *result*
end function

public key and reconstructs the token by XORing all valid shares together. Finally, the `SignToken` function produces the final cryptographic token by hashing and signing it with the system's private key. This decentralized approach ensures that no single party can reconstruct the voter's identity, while the time-based seed verification restricts token generation to valid election periods, preventing unauthorized pre-generation or post-election token creation.

4.1.5 Peer Routing Algorithm

Peer discovery and routing are fundamental to establishing a decentralized network topology. Our implementation utilizes the Kademlia Distributed Hash Table (DHT) subsystem within the libp2p framework, incorporating enhancements from S/Kademlia, Coral, and the BitTorrent DHT specifications. Algorithm 6 details the peer routing implementation, which begins by retrieving the latest blockchain block and initializing a libp2p host with the provided configuration. The algorithm then creates a Kademlia DHT instance and bootstraps it to discover initial peers in the network. For each bootstrap peer address in the configuration, the algorithm attempts to establish a connection. Upon successful connection, the peer's information is stored in the local peer store with appropriate address time-to-live (TTL) values. The algorithm then attempts to share the latest block data with the connected peer. Connection failures are handled gracefully with appropriate error logging. This distributed peer discovery mechanism ensures that nodes can dynamically join the network and synchronize blockchain data without relying on centralized coordination.

4.1.6 Integrating DeepFace with Blockchain Using Zero-Knowledge Proofs

Biometric verification enhances the security of voter authentication while maintaining privacy through cryptographic proofs. Algorithm 7 integrates the DeepFace facial recognition model with zero-knowledge proof mechanisms on the blockchain. The algorithm retrieves the voter's registered facial image ($KnownIMG$) from the database using their NID and compares it against the captured biometric data ($UnknownIMG$). The DeepFace verification function computes a distance metric d representing the facial similarity between the two images. Through extensive empirical testing, we established a recognition threshold of 0.70 to optimize the balance between false acceptance rate (FAR) and false rejection rate (FRR). If the computed distance $d \leq 0.70$, the faces match with sufficient confidence. Upon successful verification, a zero-knowledge proof is generated to attest to the verification without revealing the biometric data itself. This proof is then stored on the blockchain, creating an immutable audit trail while preserving voter privacy. If verification fails, the algorithm returns False, preventing unauthorized access.

4.1.7 Shard Management

Efficient data distribution across a decentralized network requires strategic partitioning of polling stations into shards. Algorithm 8 organizes the shard management

Algorithm 5 Token Generation using Zero-Knowledge Multi-Party Computation

Input: Voter's unique identifier (*secret*), voter's seed (*seed*), number of parties (n), party private keys, valid election time range
Output: Signed cryptographic token (*token*)
function SliceFunction($secret, n$)
    $parts \leftarrow$ empty list
    $partSize \leftarrow$ length of secret$/n$
    for $i \leftarrow 0$ to n **do**
        $parts[i] \leftarrow$ secret[i * partSize : (i + 1) * partSize]
    end for
    return *parts*
end function
function MultyPartyComputation($parts, n, seed$)
    $shares \leftarrow$ empty list
    for $i \leftarrow 0$ to n **do**
        $shares[i] \leftarrow$ generateRandomNumber()
        if $Time()$ is in range of Valid Election Time **then**
            $shares[i] \leftarrow$ shares[i] $\oplus$ seed
        else
            $shares[i] \leftarrow$ shares[i] $\oplus$ 0
        end if
        $shares[i] \leftarrow$ shares[i] $\oplus$ parts[i]
        $shares[i] \leftarrow$ Hash($shares[i]$)
        $shares[i] \leftarrow$ Sign($shares[i]$, party's private key)
    end for
    return *shares*
end function
function MergeShares($shares, n$)
    $token \leftarrow$ empty string
    for $i \leftarrow 0$ to n **do**
        if $Verify(shares[i].signature$, party's public key) is Valid **then**
            $token \leftarrow$ token $\oplus$ shares[i]
        else
            return *invalid*
        end if
    end for
    return *token*
end function
function SignToken($token, privateKey$)
    $token \leftarrow$ Hash($token$)
    $token \leftarrow$ Sign($token, privateKey$)
    return *token*
end function

Algorithm 6 Peer Routing Algorithm

Input: Configuration information (cfg), context (ctx), bootstrap peer addresses
Output: Connection status and shared block statistics
$bit \leftarrow$ Retrieve the latest block
$host \leftarrow$ Create a node host using cfg
$kademliaDHT \leftarrow$ Initialize Kademlia DHT with ctx and $host$
$kademliaDHT.Bootstrap(ctx)$
for all $peerAddr \in config.BootstrapPeers$ **do**
    $peer \leftarrow$ Wait for peer connection
    try:
    $success \leftarrow host.Connect(ctx, peerAddr)$
    if *success* **then**
        print "Connected to: *peerAddr*"
        $host.Peerstore().AddAddrs(ID, Addrs, AddrTTL)$
        if $host.SendBlockData(bit, peer)$ **then**
            print "Block *bit* shared successfully"
        else
            print "Block sharing failed"
        end if
    else
        print "Connection failed to: *peerAddr*"
    end if
    except Exception as e:
    print "Connection failed with exception: e"
end for
return Connection and block sharing statistics

Algorithm 7 ZKP-based Facial Recognition with DeepFace

Input: Voter NID ($VoterNid$), captured biometric data ($BiometricData$)
Output: Verification status (Boolean)
function VERIFYFACE($VoterNid, BiometricData$)
    $KnownIMG \leftarrow$ GetImageFromDatabase($VoterNid$)
    $UnknownIMG \leftarrow$ ImportBiometricData($BiometricData$)
    $R \leftarrow$ DeepFace.Verify($KnownIMG, UnknownIMG$)
    if $R.d \leq 0.70$ **then**
        $Proof \leftarrow$ GenerateZKP(R)
        Blockchain.StoreFacialProof($Proof$)
        return True
    else
        return False
    end if
end function

process in three phases. First, for each polling station PS_j, a unique shard S_i is generated using a deterministic function $f(PS_j)$, which might use geographical coordinates, administrative codes, or consistent hashing. Second, shards are grouped into election areas based on administrative boundaries all shards belonging to the same election area EA_l are collected into a group G_k. This hierarchical organization reflects the real-world administrative structure of elections. Third, a lookup table LT is constructed to map each network node N_m to its assigned shards, enabling efficient query routing. This mapping distributes the computational and storage load across the network while maintaining data locality for election areas. The resulting structure supports parallel vote processing across shards while enabling area-level aggregation for tallying results.

Algorithm 8 Shard Management Algorithm

Input: Polling stations $PS = \{PS_1, PS_2, \ldots, PS_j\}$, election areas $EA = \{EA_1, EA_2, \ldots, EA_l\}$, nodes $N = \{N_1, N_2, \ldots, N_m\}$
Output: Groups $G = \{G_1, G_2, \ldots, G_k\}$, lookup table LT
for each polling station PS_j in PS **do**
    Generate shard $S_i = f(PS_j)$
    Add S_i to set S
end for
for each election area EA_l in EA **do**
    Initialize group G_k as empty
    for each shard S_i in S **do**
        if S_i belongs to EA_l **then**
            Add S_i to group G_k
        end if
    end for
    Add group G_k to set G
end for
Initialize LT as empty dictionary
for each node N_m in N **do**
    Determine shards S_i assigned to N_m
    $LT[N_m] = S_i$
end for
return G, LT

4.1.8 Cross-Shard Data Retrieval

In a sharded blockchain architecture, retrieving voting data may require cross-shard communication when a query originates from a node not responsible for the target shard. Algorithm 9 addresses this challenge by implementing intelligent peer discovery and data retrieval. The algorithm accepts a vote identifier V_{id}, polling station code P_{code}, lookup table LT, and maximum neighbor search depth D. It first identifies the shard S_{vote} responsible for the specified polling station using the lookup table. If the shard's PeerId is available in the lookup table, the algorithm directly retrieves the

data. Otherwise, it invokes the `findPeerIdInNeighbors` function, which performs a breadth-first search through the node's neighbors up to depth D to locate a peer managing the target shard. This neighbor discovery mechanism provides resilience against node failures and network partitions. Once the appropriate PeerId is identified, the algorithm fetches the voting data using the vote identifier. If any step fails shard not found, peer unreachable, or data unavailable the algorithm returns null. This design ensures reliable data retrieval in a distributed, sharded environment while minimizing network overhead through efficient peer discovery.

5 Experiments, Results, and Discussions

In this section, we will provide a detailed analysis of the experimental results in terms of throughput, memory consumption, and block generation time.

The experiments were conducted on a host machine equipped with an AMD Ryzen 5 5600G CPU, 8GB of RAM, 512GB SSD, and running Debian 11 OS. A cluster of Go-based Docker containers was set up with nodes communicating over a custom Docker network. The number of nodes was varied to assess the systems scalability.

Performance metrics, including execution time, CPU and memory usage, and network latency, were collected using Go's built-in benchmarking tool. The tests encompassed single-node performance, multinode performance, and sharding scenarios. In addition, the Docker daemon was configured with performance-optimized settings, and the swap space was disabled to prevent disk swapping.

5.1 Results

Figure 8 illustrates the block generation rate (labeled as a, b, c) and its corresponding throughput (labeled as d, e, f) with and without block modularity. In standard experiments labeled as "a," block modularity consistently outperforms the non-modular approach. Initially, both methods exhibit similar times to generate a block for a single voter. However, beyond 1000 voters, the non-modular approach experiences significantly longer generation times.

This difference in performance stems from how data is managed. Block modularity intelligently divides data into smaller, categorized segments. When a new vote is added to the network, only the relevant, updated portions are merged, reducing unnecessary data duplication. This streamlined process enables faster block processing and validation. On the contrary, the non-modular method replicates all data with each operation, resulting in significant memory consumption and a gradual slowdown in block generation as the network scales. This inefficiency can lead to bottlenecks and hinder performance, particularly as the number of voters increases substantially. Moreover, in terms of throughput labeled as "d," block modularity demonstrates better performance compared to the non-modular approach.

5.1.1 Dilithium (Post-quantum) vs EdDSA (Ed25519)

The target of this research is to compare the performance of post-quantum asymmetric encryption to that of general-purpose elliptic curve encryption. To this end, the

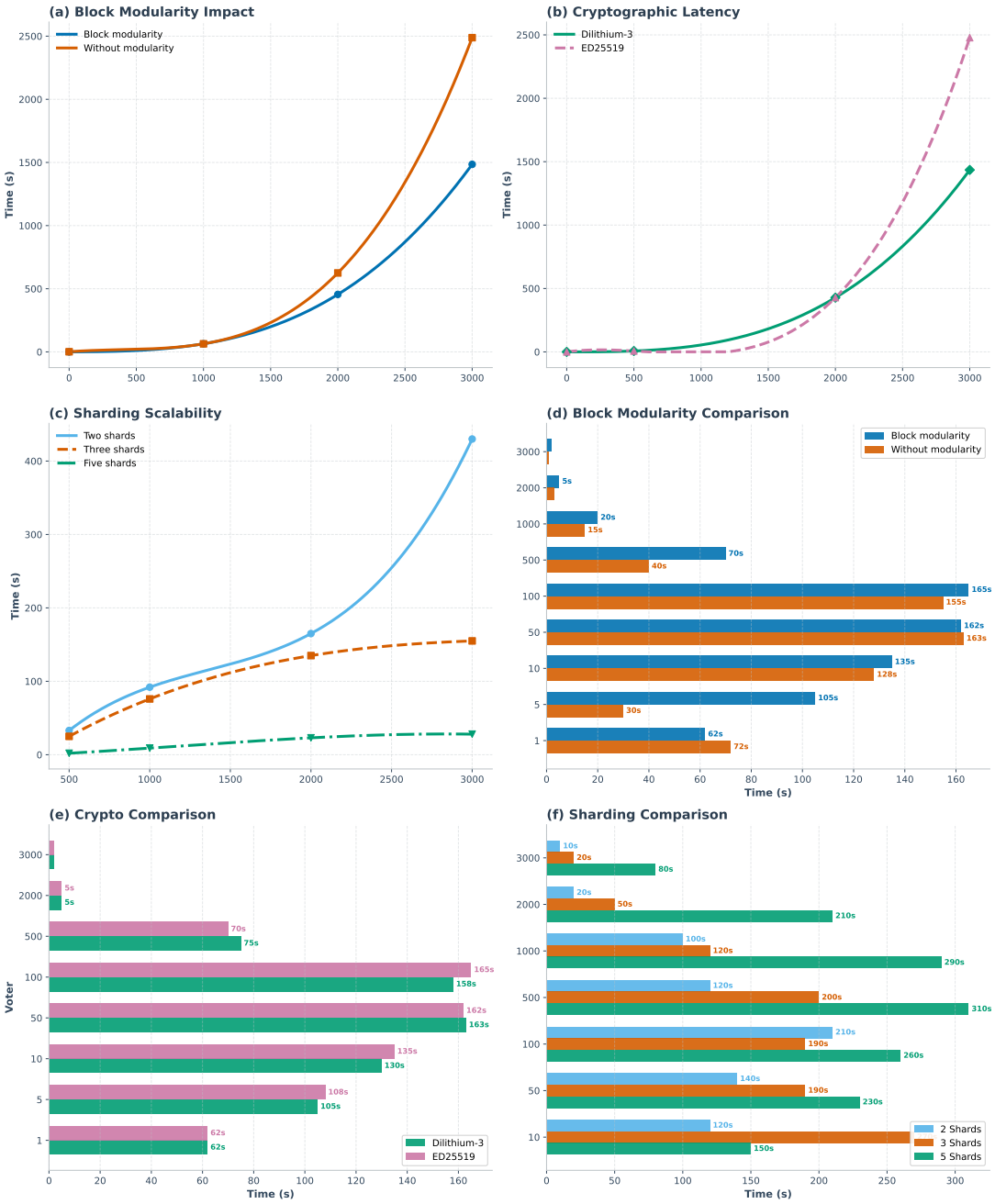

Fig. 8 The figure illustrates the visual representation of block generation time and throughput of the blockchain networks. In the representation, 'a' signifies the block generation time with and without block modularity, 'b' denotes the block generation time of Dilithium-3 and Ed25519, and 'c' represents the block generation time of sharding with two, three, and five shards. Additionally, 'd,' 'e,' and 'f' indicate the respective throughputs.

Algorithm 9 Cross-Shard Voting Data Retrieval Algorithm

Input: Lookup table LT, vote identifier V_{id}, polling station code P_{code}, max neighbor depth D
Output: Voting data V_{data} or null
Initialize $V_{data} \leftarrow$ null
function FINDPEERIDINNEIGHBORS(node, D)
    Initialize queue with node
    Initialize visited set with node
    Initialize depth $\leftarrow 1$
    while queue is not empty **and** depth $\leq$ D **do**
        Dequeue current node
        if current node has PeerId **then**
            return PeerId
        end if
        for each neighbor of current node **do**
            if neighbor not in visited **then**
                Add neighbor to visited
                Enqueue neighbor
            end if
        end for
        Increment depth
    end while
    return null
end function
▷ Identify shard from polling station code
if P_{code} is in LT **then**
    $S_{vote} \leftarrow LT[P_{code}]$
else
    return null
end if
▷ Retrieve PeerId for the shard
if S_{vote} is in LT **then**
    $PeerId_{vote} \leftarrow LT[S_{vote}]$
else
    $PeerId_{vote} \leftarrow$ findPeerIdInNeighbors(S_{vote}, D)
    if $PeerId_{vote}$ is null **then**
        return null
    end if
end if
$V_{data} \leftarrow$ retrieveVoteData($PeerId_{vote}$, V_{id})
return V_{data}

line graph in Figure 8 illustrates a performance comparison between the Dilithium-3 post-quantum algorithm and the general-purpose elliptic curve signing algorithm Ed25519 (labeled as "b"). Initially, both algorithms exhibit similar performance, but

as the number of voters increases, Ed25519 takes slightly less time than Dilithium-3. However, the difference is negligible and the lead time is almost comparable. This indicates that Dilithium-3 can be a viable drop-in replacement for general-purpose elliptic curve encryption (Ed25519) in terms of block generation time and security against quantum computer attacks. Additionally, with regards to throughput (labeled as "e"), Dilithium-3 and Ed25519 perform almost identically.

5.1.2 Sharding

Sharding, a technique for dividing data into smaller manageable parts, enhances scalability and performance in distributed systems by enabling parallel processing. Figure 8(c) illustrates how sharding with various shard counts impacts block generation time. The blue, orange, and green lines represent sharding with two, three, and five shards, respectively. The horizontal axis shows the number of voters, and the vertical axis represents the time in seconds. The graph shows that five-shard sharding offers the best performance. At its peak, the green line takes only 28 s to generate 3000 blocks, compared with 430 s for the two-shard option and 153 s for the three-shard option. Figure 8(f) presents a bar graph showing how sharding with different numbers of shards affects throughput. The orange, blue, and light green lines represent two, three, and five shards, respectively. The vertical axis represents the number of blocks processed, and the horizontal axis represents the time in seconds. Similar to block generation time, five-shard sharding demonstrated the highest measured throughput. At its peak, it processes approximately 106 blocks per second, exceeding the throughput of popular networks such as Bitcoin and Ethereum.

In conclusion, dividing the data into more shards results in significantly improved performance and throughput, thereby demonstrating the effectiveness of this approach in distributed systems.

5.1.3 Memory Consumption

Figure 9, illustrates the impact of sharding with block modularity on storage consumption within the blockchain network. These findings hold significant value for researchers and practitioners seeking to understand the storage efficiency benefits of sharding. In the first scenario, without sharding and block modularity, increased storage consumption is observed. Each block consumes 26 bytes, resulting in 371,072 bytes for 7500 blocks. This represents the storage requirements of a non sharded blockchain network.

In contrast, implementing sharding reduces individual block storage requirements to 16 bytes. Consequently, even with 7500 blocks, the overall storage usage decreases to 146,082.4 bytes. This reduction demonstrates the improvement in storage efficiency achieved through sharding and block modularity.

5.2 Security Analysis and Threat Model

In this section, we delve into the Security Analysis and Threat Model, outlining potential vulnerabilities, assessing risks, and discussing mitigation strategies for ensuring system robustness.

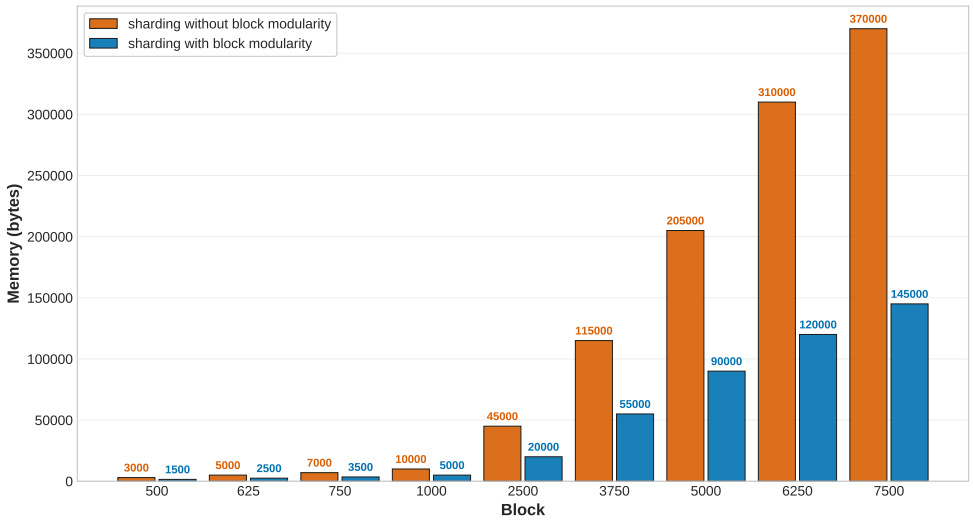

Fig. 9 The bar graph illustrates that the memory consumption with sharding (blue bar) is lower than without sharding (green bar), indicating that sharding offers a memory-saving advantage.

5.2.1 Security Requirements

To conduct a safe and secure election, the system must satisfy the following fundamental security properties:

- **Vote Privacy:** No party should be able to determine how an individual has voted. Votes must remain encrypted and unlinked to their identities. Even system administrators cannot correlate votes to voters. Vote privacy must be maintained even if some system components are compromised.
- **Vote Integrity:** Only eligible voters can cast a single vote. Votes cannot be modified, deleted, or fabricated once cast. The final tally must accurately reflect all legitimate votes.
- **Verifiability:** The system must maintain both individual and universal verifiability throughout the voting process. Voters should be able to verify their vote was correctly recorded, while anyone can independently verify the correctness of the final tally without compromising security or privacy.
- **Availability:** The system must remain fully operational throughout the election period, ensuring that voters can cast their ballots without unreasonable delays. Additionally, the system must be robust and resilient against denial-of-service attacks.
- **Coercion Resistance:** ensures that voters are unable to show or prove to anyone how they vote. This prevents vote selling and protects voters from being coerced into revealing their choices.

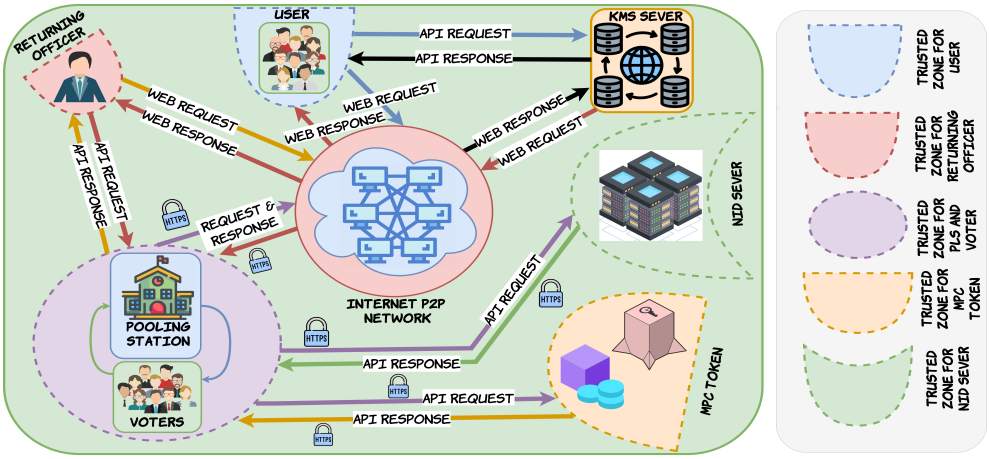

Fig. 10 STRIDE Threat Model Diagram illustrating the system components, data flow, trust boundaries, and potential threats in The PQSHAC-Bchain system with p2p network, NID server, MPC network, KMS server, and database interaction.

5.2.2 Threat Categorization and Mitigation Using STRIDE

In the context of blockchain and distributed networks, an adversary can attempt the following attacks:

- Control multiple participants in a threshold cryptography scheme.
- Monitor network traffic between components.
- Attempt to compromise system components.
- Launch denial-of-service (DoS) attacks.
- Attempt to manipulate or forge votes.
- Try to break voter privacy through various attack vectors.

Our security analysis employs the STRIDE threat modeling framework to systematically identify and categorize potential vulnerabilities. STRIDE encompassing Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege provides a comprehensive approach to security assessment. Figure 10 illustrates the interrelationships between these threat categories. In the following sections, we examine each threat vector in detail and present corresponding defensive strategies and countermeasures.

Spoofing: To prevent an attacker from impersonating a legitimate voter or election official, thereby gaining unauthorized access or casting fraudulent votes, we employ unique privacy-preserving tokens for voters, and secure KMS server-backed node authentication. Additionally, election officials are required for multilevel authorization via threshold cryptography, which ensures that even if an attacker attempts to impersonate, it cannot gain any control over the system.

Tampering: Data integrity is safeguarded through a PQHAC hybrid consensus mechanism that authorizes and verifies the data on the blockchain. The ledger is designed as an append-only structure, meaning that once data are recorded, they

cannot be tampered with or deleted. Each data iteration is visible in the public layer, and the use of ZKPs for ballots ensures voter privacy and result integrity.

Repudiation: The system implements comprehensive non-repudiation measures through end-to-end encrypted communication protocols, effectively neutralizing common attack vectors, including eavesdropping, man-in-the-middle (MITM), data tampering, and unauthorized traffic analysis. A cryptographically secured tamper-evident activity log facilitates forensic auditing and ensures action attribution.

Information Disclosure: To protect sensitive information such as voter identities and voting patterns, we use zk-MPC. This ensures privacy by preventing participants from knowing who they might collaborate with thanks to a secure participant assignment system.

Denial of Service: To prevent system disruption from flooding or DDoS attacks, category-based sharding is implemented. This method distributes traffic across similar shards, balancing the load to handle high volumes efficiently and ensuring uninterrupted voting.

Elevation of Privilege: To prevent unauthorized access to administrative functions, we employed a hierarchical access control system that ensures that users are granted only the minimum necessary privileges in accordance with electoral laws and regulations. Furthermore, the independent monitoring of system activity is utilized to detect and flag anomalous behavior, thereby enhancing the overall security of the system.

5.2.3 Comparative Analysis of Security Measures in existing system

The table 3 presents a comparative analysis of security measures, through the STRIDE framework. Each study is assessed based on its effectiveness in addressing these security aspects mentioned in the table. We believe that our proposed system could be an comparable alternative approach, incorporating multilevel authorization, PQHAC, ZKPs, zk-MPC tokens, category-based sharding, and layer-based HAC. The proposed system can effectively addresses the identified security challenges, offering a robust framework.

Although some studies [24, 26, 44] effectively tackle specific security threats, common weaknesses such as limited logging and inadequate DoS mitigation [70] persist. The proposed system provides a more robust and comprehensive solution, addressing these shortcomings effectively as shown in the table 3.

5.3 Comparative Analysis with Existing Works

In the *Security Features section*, all solutions generally provide some level of security, but there are variations in robustness and confidentiality. Notably, only a few solutions, including ours, incorporate post-quantum security[71, 72], which is crucial for future-proofing against quantum computing threats[28].

The *Consensus and Architecture* section summarizes the consensus mechanisms used in prior work, including PoS, PoW, and related variants [24, 26, 51, 60]. In contrast, we introduce a new consensus mechanism, PQSHAC. Our system achieves approximately 106 blocks per second, significantly outperforming Ethereum-based

Table 3 Comparative Analysis of Security Measures **(STRIDE)** in Blockchain-Based e-voting Systems

Study	**Spoofing**	**Tampering**	**Repudiation**	**Information Disclosure**	**Denial of Service**	**Privilege Elevation**
Curran et al. [36]	Basic digital signatures†	PoS consensus	Limited logging†	Basic access controls†	Not addressed	Role-based access
Abuidris et al. [24]	Centralized verification†	Hybrid PSC-Bchain consensus	Centralized verification†	Third-party server management†	Sharding mechanism	Server-based roles†
Neloy et al. [26]	Multi-factor auth	PoS consensus	Limited logging†	SQL data storage†	Not addressed	Basic role management
Wang et al. [69]	SLRS auth & distributed keys	Immutable ledger	Transparent audit trail	ElGamal encryption	Centralized compute†	Multi-tier access control
Lu et al. [44]	PSI-based eligibility	Immutable ledger	PSI non-repudiation	PSI privacy protection	Network resilience	Restricted access
Allende et al. [28]	Post-quantum dual-layer	PQ signature verification	PQ TLS & X.509	PQ cryptography	Not addressed	Granular permissions
Sun et al. [51]	Quantum bit commitment	Quantum Byzantine agreement	Not addressed	Quantum security	Not addressed	Not addressed
Our Proposed	multilevel authorization and threshold cryptography	PQHAC hybrid consensus mechanism & ZKPs ballot	E2EE protocols & tamper-evident activity log	zk-MPC token generation systems	Category-based sharding mechanism & shard balancing	Layer based HAC

† Indicates potential security vulnerability

implementations ($\sim$ 15–30 TPS) and Bitcoin-based approaches ($\sim$ 7 TPS). Most existing e-voting studies do not report throughput metrics and many rely directly on Ethereum- or Bitcoin-based systems [60], which limits precise performance comparison. A key differentiator of our design is native sharding support unlike most prior solutions, our system implements category-based sharding and a modular block structure.

In the *Verification and Trust category*, most solutions offer verifiability, transparency, and uniqueness, but there are differences in how they handle eligibility. Our solution ensures eligibility, verifiability, and uniqueness, aligning with high standards of trust and transparency.

Finally, the *Advanced Features section* reveals that many existing solutions do not support advanced features such as MPC token support, deepface integration[30, 73], time-based inference, and unlinkability. Our solution stands out by incorporating all these advanced features, making it a more comprehensive and versatile blockchain-based system compared to the others reviewed.

Overall,the table 2 demonstrates that our proposed solution provides notable improvements in robustness, security, and functionality compared to existing blockchain frameworks.

5.4 Current Known Limitation

To boost scalability, our architecture applies category-based sharding, significantly reducing latency and enhancing performance; however, it currently relies on random node assignment, lacking the ability to prioritize nodes by computing power or adapt shard distribution under varying load conditions. Meanwhile, although government-issued voter IDs are paired with a zk-MPC token system that mitigates double-voting risks and preserves user privacy, we have not yet incorporated alternate identity solution, such as wallet-based methods or Decentralized Identifiers (DIDs), thereby constraining flexibility and interoperability. To defend against post-quantum threats, the blockchain uses post-quantum cryptography for user data and identity, yet communication over the current internet infrastructure is still protected via standard asymmetric encryption rather than a fully post-quantum protocol.

6 Conclusion

In conclusion, this study underscores the potential of hybrid, open-source blockchain e-voting systems, enhanced with secure protocols, biometric authentication, sharding, and post-quantum cryptography, to address the limitations of traditional voting and bolster voter trust. The key lesson is that combining MPC protocols with biometric authentication preserves anonymity and ensures security, sharding techniques alleviate congestion and boost scalability, and post-quantum cryptography protects against emerging threats. The implication of our findings is that if widely adopted, our approach can significantly enhance transparency, inclusivity, and trust in digital elections, paving the way for more robust democratic processes.

Chief among the research contributions we make is the PQSHAC model, which integrates multiple cryptographic and scaling strategies into a holistic e-voting framework. This architecture fortifies security, ensures anonymity, and efficiently handles large-scale data processing. However, we observed practical hurdles during implementation, including the requirement for a high-availability internet infrastructure in remote polling stations, which was a challenge. Future research will focus on optimizing sharding node selection, exploring lattice-based cryptography for homomorphic encryption to allow tallying of encrypted votes directly, and conducting large-scale pilot tests to evaluate user usability.

We will continue to explore the application of Deep Reinforcement Learning to optimize shard resource management, integrating Decentralized Identities (DIs) to maintain compliance with legal regulations, and developing user-friendly ballot interfaces with machine translation features for greater accessibility in our future research. Finally, we invite researchers and practitioners to refine and implement the PQSHAC model, investigate these advanced techniques to create secure, scalable, and inclusive e-voting solutions that uphold public confidence in electoral processes.

Acknowledgment

Hasan Jamil's research was supported in part by the National Institutes of Health IDeA grant P20GM103408, the National Science Foundation CSSI grant OAC 2410668, and the US Department of Energy grant DE-0011014.

References

[1] Sherlock, J., Grainger, R., McDonald, S.-A., Daly, M.: Transparency: A Tool to Build Election Trust. http://tinyurl.com/muwce7fu. Accessed: 2/18/2024 (2022)

[2] Garnett, H.A., Simpson, P.: American Trust in Voting Technology. http://tinyurl.com/yxfd3bwd. Accessed: 2/18/2024 (2019)

[3] Aranha, D.F., Barbosa, P., Cardoso, T.N.C., Araújo, C.L., Matias, P.: The return of software vulnerabilities in the brazilian voting machine. Comput. Secur. **86**, 335–349 (2019)

[4] Chaeikar, S.S., Jolfaei, A., Mohammad, N., Ostovari, P.: Security principles and challenges in electronic voting. In: 2021 IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW), pp. 38–45 (2021). IEEE

[5] Pew Research Center: Public Trust in Government: 1958-2025. Accessed: 2025-12-08. https://www.pewresearch.org/politics/2025/12/04/public-trust-in-government-1958-2025/

[6] Brady, H.E., Kent, T.B.: Fifty years of declining confidence & increasing polarization in trust in american institutions. Daedalus **151**(4), 43–66 (2022)

[7] McKay, L., Jennings, W., Stoker, G.: Political trust in the "places that don't matter". Frontiers in political science **3**, 642236 (2021)

[8] Wolchok, S., Wustrow, E., Halderman, J., Prasad, H., Kankipati, A., Sakhamuri, S., Yagati, V., Gonggrijp, R.: Security analysis of india's electronic voting machines, pp. 1–14 (2010). https://doi.org/10.1145/1866307.1866309

[9] Blaze, M., Braun, J., Advisors, C.G.: Defcon 25 voting machine hacking village. Proceedings of DEFCON, Washington DC, 1–18 (2017)

[10] Riaz, A., Parvez, S.: Anatomy of a rigged election in a hybrid regime: The lessons from bangladesh. Democratization **28**(4), 801–820 (2021)

[11] Alamgir, M.: Last 3 national polls: Election commissioners breached oath of office. https://www.thedailystar.net/news/bangladesh/news/last-3-national-polls-election-commissioners-breached-oath-office-3785941 (2024)

[12] Rush, J.: Hacking the right to vote. Va. L. Rev. Online **105**, 67 (2019)

[13] Jones, B., Matsa, K.E.: Why Americans are losing trust in elections and the media. http://tinyurl.com/2xcrmf8w. Accessed: 2/18/2024 (2022)

[14] Bergeron-Boutin, O., Clayton, K., Kousser, T., Nyhan, B., Prather, L.: Communicating With Voters to Build Trust In the u.s. Election System: Best Practices and New Areas for Research. http://tinyurl.com/2v8m66dp. Accessed: 2/18/2024 (2023)

[15] Haines, T., Müller, J., Querejeta-Azurmendi, I.: Scalable coercion-resistant e-voting under weaker trust assumptions. In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, pp. 1576–1584 (2023)

[16] Diaconita, V., Belciu, A., Stoica, M.G.: Trustful blockchain-based framework for privacy enabling voting in a university. J. Theor. Appl. Electron. Commer. Res. **18**(1), 150–169 (2022)

[17] Golnarian, D., Saedi, K., Bahrak, B.: A decentralized and trustless e-voting system based on blockchain technology. In: 2022 27th International Computer Conference, Computer Society of Iran (CSICC), pp. 1–7 (2022). IEEE

[18] Buchanan, W.J.: Introduction to Blockchain. https://asecuritysite.com/blockchain/chapter91. Accessed: May 01, 2024 (2024). https://asecuritysite.com/blockchain/chapter91

[19] Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security,

pp. 3–16 (2016)

[20] Nguyen, C.T., Hoang, D.T., Nguyen, D.N., Niyato, D., Nguyen, H.T., Dutkiewicz, E.: Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. IEEE access **7**, 85727–85745 (2019)

[21] Chen, Y., Chen, H., Zhang, Y., Han, M., Siddula, M., Cai, Z.: A survey on blockchain systems: Attacks, defenses, and privacy preservation. High-Confidence Computing **2**(2), 100048 (2022) https://doi.org/10.1016/j.hcc.2021.100048

[22] Joni, S.A., Rahat, R., Tasnin, N., Ghose, P., Gaur, L.: Hac-bchain: A secure and scalable blockchain-shard based e-voting system. In: 2023 IEEE Technology & Engineering Management Conference-Asia Pacific (TEMSCON-ASPAC), pp. 1–6 (2023). IEEE

[23] Joshi, S.: Feasibility of proof of authority as a consensus protocol model. arXiv preprint arXiv:2109.02480 (2021)

[24] Abuidris, Y., Kumar, R., Yang, T., Onginjo, J.: Secure large-scale e-voting system based on blockchain contract using a hybrid consensus model combined with sharding. Etri Journal **43**(2), 357–370 (2021)

[25] Alzhrani, F., Saeedi, K., Zhao, L.: Architectural patterns for blockchain systems and application design. Applied Sciences **13**(20), 11533 (2023)

[26] Neloy, M.N., Wahab, M.A., Wasif, S., All Noman, A., Rahaman, M., Pranto, T.H., Haque, A.B., Rahman, R.M.: A remote and cost-optimized voting system using blockchain and smart contract. IET Blockchain **3**(1), 1–17 (2023)

[27] Dooley, K.: Designing Large Scale LANs: Help for Network Designers, pp. 26–36. "O'Reilly Media, Inc.", Sebastopol, CA (2001)

[28] Allende, M., León, D.L., Cerón, S., Pareja, A., Pacheco, E., Leal, A., Da Silva, M., Pardo, A., Jones, D., Worrall, D.J., *et al.*: Quantum-resistance in blockchain networks. Scientific Reports **13**(1), 5664 (2023)

[29] Serengil, S.I., Ozpinar, A.: Lightface: A hybrid deep face recognition framework. In: 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), pp. 23–27 (2020). https://doi.org/10.1109/ASYU50717.2020.9259802 . IEEE. https://doi.org/10.1109/ASYU50717.2020.9259802

[30] Serengil, S.I., Ozpinar, A.: An Evaluation of SQL and NoSQL Databases for Facial Recognition Pipelines. Cambridge Open Engage. Preprint (2023). https://doi.org/10.33774/coe-2023-18rcn . https://doi.org/10.33774/coe-2023-18rcn

[31] Ghose, P., Uddin, M.A., Acharjee, U.K., Sharmin, S.: Deep viewing for the identification of covid-19 infection status from chest x-ray image using cnn based

architecture. Intelligent Systems with Applications **16**, 200130 (2022)

[32] Ghose, P., Biswas, M., Gaur, L.: Brainsegnet: a lightweight brain tumor segmentation model based on u-net and progressive neuron expansion. In: International Conference on Brain Informatics, pp. 249–260 (2023). Springer

[33] Ghose, P., Sharmin, S., Gaur, L., Zhao, Z.: Grid-search integrated optimized support vector machine model for breast cancer detection. In: 2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 2846–2852 (2022). IEEE

[34] Ghose, P., Joni, S.A., Rahat, R., Tasnin, N., Biswas, M.: An effective combination of deep and machine learning models for monkeypox detection from dermatographic image. In: International Conference on Trends in Electronics and Health Informatics, pp. 33–44 (2023). Springer Nature Singapore Singapore

[35] Gennaro, R., Goldfeder, S.: One Round Threshold ECDSA with Identifiable Abort. Cryptology ePrint Archive, Paper 2020/540 (2020). https://eprint.iacr.org/2020/540

[36] Curran, K.: E-voting on the blockchain. The Journal of the British Blockchain Association **1**(2) (2018)

[37] Al-Zubaidie, M., Jebbar, W.: Transaction security and management of blockchain-based smart contracts in e-banking-employing microsegmentation and yellow saddle goatfish. Mesopotamian Journal of CyberSecurity **4**(2), 1–19 (2024)

[38] Košt'ál, K., Bencel, R., Ries, M., Kotuliak, I.: Blockchain e-voting done right: Privacy and transparency with public blockchain. In: 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), pp. 592–595 (2019). IEEE

[39] Yousiff, S.A., Muhajjar, R.A., Al-Zubaidie, M.H.: Designing a blockchain approach to secure firefighting stations based internet of things. Informatica **47**(10) (2023)

[40] Liu, A., Liu, Y., Wu, Q., Zhao, B., Li, D., Lu, Y., Lu, R., Susilo, W.: Cherubim: A secure and highly parallel cross-shard consensus using quadruple pipelined two-phase commit for sharding blockchains. IEEE Transactions on Information Forensics and Security **19**, 3178–3193 (2024) https://doi.org/10.1109/TIFS.2024.3358990

[41] Duan, S., Wang, X., Zhang, H.: Fin: Practical signature-free asynchronous common subset in constant time. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. CCS '23, pp. 815–829. Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3576915.3616633 . https://doi.org/10.1145/3576915.3616633

[42] Liu, Y., Xing, X., Cheng, H., Li, D., Guan, Z., Liu, J., Wu, Q.: A flexible sharding blockchain protocol based on cross-shard byzantine fault tolerance. IEEE Transactions on Information Forensics and Security **18**, 2276–2291 (2023)

[43] Duan, S., Zhang, H., Sui, X., Huang, B., Mu, C., Di, G., Wang, X.: Dashing and star: Byzantine fault tolerance with weak certificates. In: Proceedings of the Nineteenth European Conference on Computer Systems, pp. 250–264 (2024)

[44] Lu, S., Li, Z., Miao, X., Han, Q., Zheng, J.: Piws: private intersection weighted sum protocol for privacy-preserving score-based voting with perfect ballot secrecy. IEEE Transactions on Computational Social Systems **10**(3), 1039–1056 (2022)

[45] Joni, S.A., Rahat, R., Tasnin, N., Ghose, P., Uddin, M.A., Ayoade, J.: Hybrid-blockchain-based electronic voting machine system embedded with deepface, sharding, and post-quantum techniques. Blockchains **2**(4), 366–423 (2024)

[46] Rahat, R., Joni, S.A., Tasnin, N., Ghose, P., Gaur, L.: Towards secure democracy: a hybrid blockchain-enabled secure and scalable e-voting system with sharding and post-quantum cryptography. International Journal of System Assurance Engineering and Management, 1–11 (2025)

[47] Liu, J., Shen, X., Xie, M., Zhang, Q.: Research on sharding strategy of blockchain based on topsis. In: International Conference on Smart Computing and Communication, pp. 247–257 (2022). Springer

[48] Garg, S., Sahai, A.: Adaptively secure multi-party computation with dishonest majority. In: Annual Cryptology Conference, pp. 105–123 (2012). Springer

[49] Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, pp. 639–648 (1996)

[50] Li, B., Wu, F.: Post quantum blockchain with segregation witness. In: 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS), pp. 522–527 (2021). IEEE

[51] Sun, X., Wang, Q., Kulicki, P., Sopek, M.: A simple voting protocol on quantum blockchain. International Journal of Theoretical Physics **58**(1), 275–281 (2019) https://doi.org/10.1007/s10773-018-3929-6

[52] Ahmed Joni, S., Rahat, R., Tasnin, N., Ghose, P., Biswas, M.: Enhancing electoral integrity: A hybrid blockchain-based e-voting system with deep learning and post-quantum cryptography. In: International Conference on Trends in Electronics and Health Informatics, pp. 687–698 (2023). Springer

[53] Ghose, P., Uddin, M.A., Islam, M.M., Islam, M., Acharjee, U.K.: A breast cancer

detection model using a tuned svm classifier. In: 2022 25th International Conference on Computer and Information Technology (ICCIT), pp. 102–107 (2022). IEEE

[54] Ghose, P., Alavi, M., Tabassum, M., Ashraf Uddin, M., Biswas, M., Mahbub, K., Gaur, L., Mallik, S., Zhao, Z.: Detecting covid-19 infection status from chest x-ray and ct scan via single transfer learning-driven approach. Frontiers in Genetics **13**, 980338 (2022)

[55] Jefferson, D.: The Myth of "Secure" Blockchain Voting. http://tinyurl.com/4evkx3eh. Accessed: 2/18/2024 (2023)

[56] Park, S., Specter, M., Narula, N., Rivest, R.L.: Going from bad to worse: from internet voting to blockchain voting. Journal of Cybersecurity **7**(1), 025 (2021)

[57] Kumar, R., Badwal, L., Avasthi, S., Prakash, A.: A secure decentralized e-voting with blockchain & smart contracts. In: 2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence), pp. 419–424 (2023)

[58] Hajian Berenjestanaki, M., Barzegar, H.R., El Ioini, N., Pahl, C.: Blockchain-based e-voting systems: A technology review. Electronics **13**(1) (2024)

[59] Bhadoria, R.S., Das, A.P., Bashar, A., Zikria, M.: Implementing blockchain-based traceable certificates as sustainable technology in democratic elections. Electronics **11**(20), 3359 (2022)

[60] Khoury, D., Kfoury, E.F., Kassem, A., Harb, H.: Decentralized voting platform based on ethereum blockchain. In: 2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET), pp. 1–6 (2018). IEEE

[61] Anitha, V., Caro, O.J.M., Sudharsan, R., Yoganandan, S., Vimal, M.: Transparent voting system using blockchain. Measurement: Sensors **25**, 100620 (2023)

[62] Singh, J., Rastogi, U., Goel, Y., Gupta, B., et al.: Blockchain-based decentralized voting system security perspective: Safe and secure for digital voting system. arXiv preprint arXiv:2303.06306 (2023)

[63] Ch, R., Kumari D, J., Gadekallu, T.R., Iwendi, C.: Distributed-ledger-based blockchain technology for reliable electronic voting system with statistical analysis. Electronics **11**(20), 3308 (2022)

[64] Bhadoria, R.S., Das, A.P., Bashar, A., Zikria, M.: Implementing blockchain-based traceable certificates as sustainable technology in democratic elections. Electronics **11**(20), 3359 (2022)

[65] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR

Transactions on Cryptographic Hardware and Embedded Systems, 238–268 (2018)

[66] Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., et al.: Status report on the third round of the nist post-quantum cryptography standardization process. US Department of Commerce, NIST (2022)

[67] Greconici, D.O., Kannwischer, M.J., Sprenkels, A.: Compact dilithium implementations on cortex-m3 and cortex-m4. IACR Transactions on Cryptographic Hardware and Embedded Systems, 1–24 (2021)

[68] Buchanan, W.J.: Any t-of-n threshold ECDSA signing algorithm using GG20 with Kryptology. https://asecuritysite.com/ecdsa/sss_gg03. Accessed: April 20, 2024 (2024). https://asecuritysite.com/ecdsa/sss_gg03

[69] Wang, P.-L., Yang, S.-H., Hsiao, H.-C.: Hybrid-voting: A hybrid structured electronic voting system. In: Companion Proceedings of the Web Conference 2020, pp. 83–84 (2020)

[70] Mahjabin, T., Xiao, Y., Sun, G., Jiang, W.: A survey of distributed denial-of-service attack, prevention, and mitigation techniques. International Journal of Distributed Sensor Networks **13**(12), 1550147717741463 (2017)

[71] Gupta, S., Gupta, K.K., Shukla, P.K., Shrivas, M.K.: Blockchain-based voting system powered by post-quantum cryptography (bbvsp-pqc). In: 2022 Second International Conference on Power, Control and Computing Technologies (ICPC2T), pp. 1–8 (2022). IEEE

[72] Gupta, S., Gupta, A., Pandya, I.Y., Bhatt, A., Mehta, K.: End to end secure e-voting using blockchain & quantum key distribution. Materials Today: Proceedings **80**, 3363–3370 (2023)

[73] Serengil, S.I., Ozpinar, A.: Hyperextended lightface: A facial attribute analysis framework. In: 2021 International Conference on Engineering and Emerging Technologies (ICEET), pp. 1–4 (2021). https://doi.org/10.1109/ICEET53442.2021.9659697 . IEEE. https://doi.org/10.1109/ICEET53442.2021.9659697