blockchains

MDPI

Article

Hybrid-Blockchain-Based Electronic Voting Machine System Embedded with Deepface, Sharding, and Post-Quantum Techniques

Sohel Ahmed Joni [1,†], Rabiul Rahat [1,†], Nishat Tasnin [1,†], Partho Ghose [1,*,†], Md. Ashraf Uddin [2,*] and John Ayoade [3]

[1] Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka 1216, Bangladesh; 20215103018@cse.bubt.edu.bd (S.A.J.); 20215103017@cse.bubt.edu.bd (R.R.); 20215103008@cse.bubt.edu.bd (N.T.)
[2] School of Information Technology, Deakin University, Melbourne, VIC 3125, Australia
[3] School of Information Technology, Crown Institute of Higher Education, North Sydney, NSW 2060, Australia; john.ayoade@cihe.edu.au
* Correspondence: partho@bubt.edu.bd (P.G.); ashraf.uddin@deakin.edu.au (M.A.U.)
† These authors contributed equally to this work.

Abstract: The integrity of democratic processes relies on secure and reliable election systems, yet achieving this reliability is challenging. This paper introduces the Post-Quantum Secured Multiparty Computed Hierarchical Authoritative Consensus Blockchain (PQMPCHAC-Bchain), a novel e-voting system designed to overcome the limitations of current Biometric Electronic Voting Machine (EVM) systems, which suffer from trust issues due to closed-source designs, cyber vulnerabilities, and regulatory concerns. Our primary objective is to develop a robust, scalable, and secure e-voting framework that enhances transparency and trust in electoral outcomes. Key contributions include integrating hierarchical authorization and access control with a novel consensus mechanism for proper electoral governance. We implement blockchain sharding techniques to improve scalability and propose a multiparty computed token generation system to prevent fraudulent voting and secure voter privacy. Post-quantum cryptography is incorporated to safeguard against potential quantum computing threats, future-proofing the system. Additionally, we enhance authentication through a deep learning-based face verification model for biometric validation. Our performance analysis indicates that the PQMPCHAC-Bchain e-voting system offers a promising solution for secure elections. By addressing critical aspects of security, scalability, and trust, our proposed system aims to advance the field of electronic voting. This research contributes to ongoing efforts to strengthen the integrity of democratic processes through technological innovation.

Keywords: EVM; sharding; post-quantum attacks; deep learning; security; scalability

Citation: 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* **2024**, 2, 366–423. https://doi.org/10.3390/blockchains2040017

Academic Editors: Liehuang Zhu and Keke Gai

Received: 12 July 2024
Revised: 4 September 2024
Accepted: 11 September 2024
Published: 30 September 2024

1. Introduction

An election system is a crucial part of democracy and a catalyst for the progression of a country. Conducting a fair and equitable election in a country in which democracy has not yet been firmly established poses significant challenges, particularly in the absence of a reliable voting system [1]. The vulnerability of paper-based voting systems to tampering and fraud by influential groups or individuals poses a significant threat. Newer Electronic Voting Machines (EVMs) that use biometrics have addressed some of these concerns. However, they often operate as "black boxes", meaning that voters cannot accurately verify their votes. This lack of transparency undermines trust in the voting process. Furthermore, current EVM voting systems are prone to manipulation by a single authority [1].

In response to this issue, researchers have turned their attention to emerging technologies, such as blockchain, as a potential way to enhance transparency and security in

voting systems. Blockchain [2] is a distributed, immutable, and tamper-resistant public ledger that offers several important characteristics. Notably, one of its key features is immutability, which ensures that once data are recorded on the blockchain, they cannot be altered or deleted. This is achieved by requiring each generated block to include the digest of the previous block, forming an unbreakable chain that guarantees the integrity and permanence of the blockchain. Any attempt to tamper with existing blocks disrupts the integrity of the chain, rendering the blockchain obsolete and unreliable [3].

To operate a blockchain, an authoritative consensus is essential, and it is one of the most critical features of this technology. In many countries, electoral systems are legally required to enforce specific regulations to prevent unauthorized activities and misconduct during elections. But consensus mechanisms that are specially designed to manage cryptocurrency operation or general-purpose blockchain are just too unsophisticated to handle the complex permission and access systems required for electoral processes [4].

Therefore, this paper proposes a consensus mechanism that employs a controlled level of authority, akin to an authorized block, to include or exclude specific entities based on their authorized permissions within the electoral system. This approach ensures transparency, ballot integrity, and immutability while maintaining necessary regulation without granting any entity excessive or unconventional permissions.

Blockchain technology, due to its transparency, is considered a promising tool for implementing modern and innovative voting processes [5,6]. However, scalability and performance are significant concerns in existing proposed blockchain-based e-voting systems. To address this issue, researchers have explored sharding techniques, such as [7,8], which involve partitioning the blockchain into smaller shards. Nevertheless, most of their proposed methods uses shuffle sharding that involves selecting shards and nodes randomly, failing to create data-contracted shards and introducing new latency-related issues discussed in Section 2.3. In our study, we adopt a category-based sharding approach, in which the blockchain is divided into shards based on the category or type of data stored.

Moreover, preserving the confidentiality of voter privacy while maintaining transparency in a blockchain is crucial. Several studies have employed various encryption techniques to ensure user privacy. Some proposals involve using a third-party server for verification, while others suggest storing voter information directly on the blockchain to validate voter identity. However, using a third-party server does not provide adequate security and immutability, and storing voter information on the blockchain can potentially breach voter privacy and anonymity [9–12].

As a solution, we propose a novel multiparty computed [13] token generation system that enables secure voter verification, checks for fraudulent and double votes, and keeps voter identities anonymous. For verification, it uses zero-knowledge proofs, where the system validates against the securely generated token instead of directly validating the voter. In our work, we adopt post-quantum secured asymmetric cryptography. This encryption algorithm can provide security from attacks not only by general computers but also by quantum computers [14].

Furthermore, tackling fraudulent voting is another challenge in electronic voting systems. As an extra layer of security, we have adopted biometric AI face verification, also known as DeepFace, which is a deep learning facial recognition system. This method employs a convolutional neural network architecture to identify faces in digital images with high accuracy [15,16]. By implementing these features, our proposed system aims to provide a secure, transparent, and efficient electronic voting process.

The contributions of this study are summarized as follows.

- To address the limitations of existing consensus mechanisms, this research proposes a novel hybrid consensus model (HAC) that provides the controlled level of permission and access required for the electoral process.
- The proposed category-based novel sharding mechanism leverages the inherent discrete nature of election data, such as divisions by polling station and election area.

This significantly enhances the e-voting system's scalability and data concentration compared to existing sharding mechanisms for e-voting.

- We incorporate post-quantum cryptography and perform a comprehensive analysis of Dilithium 2- and Dilithium 3-based implementations compared to Ed25519-based blockchains. This analysis aims to determine whether post-quantum asymmetric cryptography can be a viable alternative to traditional cryptography.
- As an extra layer of security to prevent fraudulent voting, we incorporated DeepFace to enhance the scalability, security, and performance of the system.
- We propose a novel multiparty computation (MPC) protocol to generate a secure and verifiable token for each voter. For verification, it uses zero-knowledge proofs, where the system validates against the securely generated token instead of directly validating the voter. This solves coerced and double voting while protecting their identity.
- Finally, we conduct a performance analysis using synthetic election data similar to real-world data, and the results are promising. Our findings demonstrate that the proposed system can potentially handle large-scale elections without sacrificing security or performance.

Our research makes several contributions to the domains of e-voting and blockchains. We propose a novel hybrid consensus model called Hierarchical Authoritative Consensus (HAC) along with the integration of sharding, post-quantum cryptography, and deep learning facial recognition models. This study aims to enhance the security, scalability, and efficiency of e-voting systems based on blockchain technology.

The remainder of this paper is organized as follows. In Section 2, we review past research on blockchain-based e-voting systems and highlight major deployment challenges. Section 3 introduces our e-voting system and its objectives. Section 4 details our hybrid consensus model and its design. Section 5 describes the implementation of the blockchain system, including the block structure and pseudocode. Section 6 evaluates the system performance and compares it with other technologies. Section 7 analyzes the system security and potential vulnerabilities. Finally, Section 8 concludes the paper.

2. Related Work

In this section, we introduce previous work related to this research in the context of blockchain-based e-voting systems and consider three significant issues associated with this research: security, consensus and verification, and sharding.

2.1. Previous Work

Many previous studies have attempted to develop protocols for blockchain-based e-voting systems and create incentive schemes for cryptocurrencies. Our work is motivated by recent advances [12,17–19].

Kevin et al. [12] discussed blockchain voting, highlighting its elimination of central authority and enabling of vote verification by anyone. They suggested Hyperledger Fabric as a viable blockchain solution. Das et al. [17] proposed a blockchain-based voting system integrated with face recognition. However, their method and others have limitations, including scalability and extensive computational requirements. Additionally, their use of an msp system to validate voters undermines privacy, and the Raft consensus mechanism used by their system is not feasible for electoral system permissions and access. Khoury et al. [18] proposed a decentralized trustless voting platform using Ethereum and mobile numbers to register each voter in the system. Hasan et al. [20] also proposed an Ethereum-based voting system using smart contracts. Neloy et al. [10] suggested an Ethereum network-based system with user verification via Metamask wallet, face recognition, and deepface analysis. Singh et al. [21] proposed a system using unique identification like Aadhar Card numbers or OTPs for authentication and Ethereum blockchain, integrating with traditional Electronic Voting Machines. Although researchers from [10,17,18,20,21] proposed Ethereum-blockchain-based e-voting systems aiming for secure voting, performance and scalability issues were not adequately addressed. Many of these proposals use

OTPs or third-party servers, which do not provide adequate security and immutability. Additionally, storing voters' unique identifiers on solidity smart contracts can potentially breach voter privacy and anonymity. Neloy et al. [10] efficiently utilized solidity smart contracts to reduce transaction costs, but relying on an Ethereum-based network could lead to centralization and high gas costs for large-scale elections. Yousif et al. [9] proposed a hybrid blockchain (PSC-Bchain) combining Proof of Credibility and Proof of Stake to address energy consumption and scalability issues in blockchain-based e-voting systems. They employed Ethereum smart contracts and a sharding mechanism to enhance security and performance. However, this approach introduces additional computational complexity and increased gas fees. Moreover, their approach does not fully address voter privacy, as it uses a third-party server called a managed server to store node and voter information. This poses a severe risk to system integrity and voter privacy, as the data stored on the managed server are not part of an immutable ledger. Additionally, their proposal does not adequately address the potential risk of single points of failure that could occur from this server.

2.2. Consensus and Verification

Li et al. [22] proposed a consensus algorithm, POV, for decentralized arbitration through voting. POV theoretically achieves transaction finality with one confirmation. However, the study lacks experimental evaluation, and multilevel access control is unclear. Wang et al. [19] introduced CW-DPoS, a strategy to improve node activity and fairness in the DPoS consensus algorithm, resulting in performance enhancements. However, security risks and real-world election complexities are not addressed. Yuanyuan et al. [23] proposed DT-DPoS, enhancing the DPoS consensus algorithm's security and scalability with an Eigen Trust model and ring signatures against DoS and collusion attacks. A theoretical analysis supported its effectiveness.

2.3. Sharding

Sharding enhances blockchain scalability by dividing it into smaller, independent pieces called shards. These shards process transactions simultaneously, enabling parallel processing and boosting throughput [24]. Tao et al. [25] proposed a new distributed and dynamic sharding system to significantly increase the throughput of smart-contract-based blockchain systems while minimizing cross-shard communication. The proposed algorithms were analyzed using a game-theoretic approach. Ren et al. [26] analyzed the high cost of cross-shard transactions and found that most Bitcoin transactions have simple dependencies and can become single shards with a dependency-aware placement algorithm.

Sharding methods such as those proposed by Tao et al [25] and Ren et al. [26] focus on minimizing cross-shard communication and managing transaction dependencies, which are effective for general transactions but fall short in handling the complex permissions and strict regulatory requirements of e-voting. The lack of data concentration in these methods can lead to latency issues and inefficiencies in processing election data, which are often divided by polling station, area, and election category.

Li et al. [27] A sharding-based blockchain breaks shard isolation by managing state and logic storage and smart contract execution. Nodes share contract logic, while shards store distinct states. A cross-shard consensus algorithm ensures transaction security and efficiency. Wang et al. [28] proposed a distributed blockchain storage scheme using sharding and secret-sharing without a trusted third party. It reduces storage and communication costs with polynomial commitment and homomorphic properties. Yousif et al. [9] proposed sharding in an e-voting system, demonstrating improvements over non-sharding techniques. However, their model assigns nodes randomly to shards using a Verifiable Random Function (VRF) and Verifiable Delay Function (VDF), failing to leverage the inherent discrete categorical nature of election data. Such data can be divided by polling station, area, and election category.

2.4. Post-Quantum Cryptography

Li and his companions [29] proposed a new lattice-based signature scheme based on the Short Integer Solution (SIS) problem. In addition, its efficacy against the Shor and Grover algorithms was analyzed. Y. Gao et al. [30] proposed a secure cryptocurrency scheme based on PQB to resist quantum computing attacks. In addition, the improved correctness and unforgeability of the signature under the lattice SIS assumption were analyzed. Li et al. [31], as mentioned in this paper, proposed a post-quantum blockchain with a segregation witness, increasing signature proportions in block size based on SIS hardness. They also studied post-quantum crypto-systems and their challenges for blockchain and DLT application. Ref. [32] proposed a layer-two solution for secure blockchain node communication and introduced post-quantum key transactions. Sun et al. [33] proposed a quantum-blockchain-based voting protocol using cryptographic primitives and a quantum Byzantine agreement for consensus.

The existing proposed systems, while advancing post-quantum security, tend to prioritize cryptographic hardness and resistance to quantum attacks. However, they may fall short in addressing the unique and critical needs of e-voting systems, such as efficiency, scalability, data management, voter privacy, and the ability to handle complex, real-time election data securely and transparently.

2.5. Deepface

Deepface [34] is a deep learning system used for face recognition and authentication. Blockchain enhances security and tamper-proof nature by storing face images, providing secure and accessible large-scale face data storage [35]. Deep learning parallelization enhances face recognition accuracy and efficiency by converting facial features into binary hash codes for faster recognition [36].

2.6. Problem Formulation

After analyzing the related work on blockchain-based e-voting, consensus, post-quantum cryptography, and DeepFace in blockchain-based e-voting systems, here are the formulated problems and challenges we need to address in the proposed system:

1. Many e-voting systems rely on third-party blockchain systems like Bitcoin and Ethereum. However, the gas fees or transaction costs may become infeasible for hosting large-scale, nationwide elections.
2. Consensus mechanisms are one of the most critical features of blockchain and decentralized networks. Most existing blockchain EVM systems use consensus mechanisms designed for managing cryptocurrency operations. These general-purpose blockchains do not provide the required various levels of access and control for different entities, and they fail to meet legally required regulations to prevent unauthorized activities.
3. Existing systems rely on third-party wallets and managed servers for voter validation, potentially compromising system integrity and voter privacy, as data stored on these servers are not part of an immutable ledger. Moreover, single-entity management introduces single points of failure. To address these issues, a secure voter validation system is required, one that references the distributed ledger, protects voter privacy, and employs a multiparty security model.
4. In the age of quantum computing, traditional asymmetric encryption can be broken via quantum computers using algorithms like Shor's and Grover's. We need our system to be a deterrent against such attacks.
5. Existing blockchain sharding approaches like VRF and VRD have failed to utilize the discrete categorical nature of election data and failed to create data-concentrated shards. E-voting systems require a high level of data concentration and integrity, which is often not achievable by the random sharding approach.
6. We need a way to prevent identity theft, impersonation, or other fraudulent activities. Biometric facial recognition adds an extra layer of security by ensuring that the person casting the vote matches the registered voter's identity.

To address the challenges in election systems, this study proposes a *Post-quantum Secured Multiparty Computed Hierarchical Authoritative Consensus Blockchain* (PQMPCHAC-Bchain) to make a secure, transparent, and scalable e-voting system, eliminating doubts about election outcomes and ensuring a fair and accurate voting process.

2.7. Comparative Analysis of Existing Methods and Proposed Approach

In this section, we present a comparative analysis with existing e-voting systems to highlight the novelty and advantages of our proposed approach. This analysis focuses on key features such as consensus mechanisms, performance, sharding, MPC token generation systems, post-quantum cryptography, immutable zk-voter verification, block modularity, access and control, single points of failure, and limitations, as shown in Table 1.

Table 1. Comparative analysis of existing methods and our proposed approach.

Features	Previous Study [10]	Previous Study [9]	Previous Study [20]	Previous Study [37]	Our Proposed
Consensus Mechanism	PoS	PoS & PoC	PoS	PBFT	**HAC Hybrid Consensus**
Performance	25 TPS	25 TPS	25 TPS	–	*181 TPS*
Sharding	VRD (Random)	No	No	No	*Category-based Sharding*
Shard Data Concentration	Low	N/A	N/A	N/A	*High*
MPC Token Generation System	No	No	No	No	*Yes*
Post Quantum Cryptography	No	No	No	No	*Dilithium*
Immutable zk-voter Verification	No	No	No	No	*Yes*
Block Modularity	No	No	No	No	*Yes*
Single Point of Failure	Potentially	Potentially	Yes	Yes	*No*
Access & Control	Hybrid	Public	Public	Public	*Hierarchical Layer-Based Access & Control*
Limitations	Sharding, Compatibility	VRD Sharding & Complexity	Privacy, Managed Server	Latency	*None*

The comparison demonstrates how our proposed method addresses the shortcomings of previous studies and offers significant improvements in various aspects. More comparisons on various requirements can be found in Section 7.3.

3. Hybrid Consensus Model

This paper presents a novel approach to consensus called the Hierarchical Authoritative Consensus (HAC) model. In this section, we discuss the details of the proposed hybrid consensus model and provide a clear perspective of its design. The consensus model integrates a dependable signing mechanism that ensures the authenticity of each participant involved in voting. This feature is crucial for maintaining the system's integrity.

Furthermore, our hybrid consensus model employs a hierarchical authorization and access-control system. This innovative approach allows us to capitalize on the beneficial features of both public and private blockchains while simultaneously mitigating their drawbacks. This balance contributes significantly to the efficiency and security of the consensus model.

3.1. Structure of the Blockchain Network in the Framework

The current structure of e-voting systems presents numerous challenges related to credibility and verification, and this study aims to identify and address these by proposing the PQMPCHAC-Bchain model, designed to be compatible with the existing electoral system while enhancing its security, transparency, reliability, and integrity.

To achieve this, the architecture of a blockchain network for voting applications must be defined. In our proposed model, each entity involved in the voting process is authorized by a higher authority. This hierarchical authorization and predefined access control ensure that each entity has the necessary permissions to facilitate hybrid consensus and e-voting functionality. By implementing the PQMPCHAC-Bchain model, we can address the challenges presented by the existing e-voting system structure and improve the overall electoral process.

The proposed model in Figure 1 features multilevel access and authorization in its consensus mechanism, with each voting process entity having predefined access and a specific role within the blockchain. The hybrid blockchain system proposed differs from combining two separate blockchains, using a single blockchain with hierarchical and separate layers for permission and access based on the user within the consensus mechanism. This approach offers the benefits of public and private chains without compromising security. By maintaining sensitive election authoritative access and control of the public network, the model ensures secure, fair voting as per law regulations while preserving the transparency and accountability of a public blockchain by sharing all ledgers publicly. Operating on a public network also ensures greater immutability.

- *Public Layer*: Open to everyone, allowing data transfer and value observation. Anyone can join, read, and monitor transactions, but adding new or appended data is restricted.
- *Private Layer:* Accessible only to authorized entities like polling officers, enabling them to validate new blocks using encryption and access controls.
- *Administration Layer:* Manages blockchain validators, allowing administrators to assign/remove validators, publish results, and halt elections in emergencies, ensuring integrity and security.
- *Election Management Layer:* Oversees the election process, enabling election creation, administrator assignment/removal, and emergency actions like halting or reorganizing elections to safeguard integrity and security.

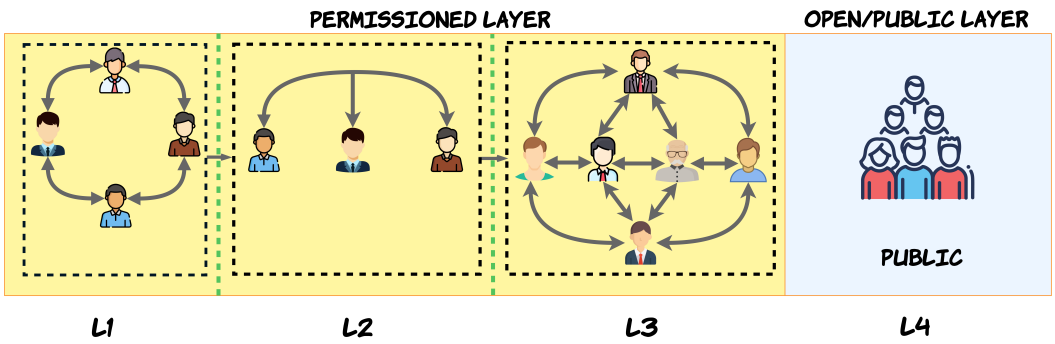

Figure 1. This diagram visually explains the Hierarchical Authoritative Consensus algorithm, which combines both public and permitted features of blockchain. It shows the multiple layers of permission and access for each entity in the network, making HAC a hybrid consensus algorithm.

Overall, this approach enhances the security and reliability of e-voting systems by combining the advantages of public and private blockchains while addressing their respective limitations. By implementing hierarchical and separate layers for permission and access, we can ensure that sensitive information remains confidential while maintaining the transparency and accountability of the voting process.

3.2. Comparison with Other Consensus Mechanisms

To better understand the advantages of our proposed HAC model, we compare it with other popular consensus mechanisms :

- *Proof of Work (PoW):* Used by Bitcoin, PoW requires miners to solve complex mathematical puzzles to validate transactions. While it is secure, it is energy-intensive and slow.
- *Proof of Stake (PoS):* Used by Ethereum 2.0, PoS selects validators based on their staked cryptocurrency. It is more energy-efficient than PoW but can lead to centralization if large stakeholders control the network.
- *Practical Byzantine Fault Tolerance (PBFT):* Used by Hyperledger Fabric, PBFT ensures consensus through a voting process among nodes. It is efficient but can be slow with a large number of nodes.
- *Delegated Proof of Stake (DPoS):*Used by EOS, DPoS allows token holders to elect delegates who validate transactions. It is fast and scalable but can lead to centralization if a small number of delegates control the network.

Table 2 summarizes the comparison of these consensus mechanisms.

Table 2. Comparison of consensus mechanisms.

Consensus Mechanism	Energy Efficiency	Scalability	Security	Decentralization
PoW	Low	Low	High	High
PoS	High	Medium	High	Medium
PBFT	Medium	Low	High	Medium
DPoS	High	High	Medium	Low
PQMPCHAC	*High*	*High*	*High*	*High*

The HAC model combines the best features of these consensus mechanisms while addressing their limitations. It is energy-efficient, scalable, secure, and decentralized, making it suitable for e-voting systems.

3.3. Real-Life Hybrid Consensus Model

Figure 1 illustrates our multi-layer hybrid consensus mechanism, which utilizes multiple layers of nodes to ensure fair and accurate voting. Our system features threshold cryptography and distributed authorization power, mitigating the risks of single-point failures and enhancing security over the network, thereby improving voters' trust and confidence in the election process.

An ideal example of how the HAC mechanism can be utilized in an e-voting scenario is as follows:

1. *Layer 1:* Holds significant power, managing election administrators/returning officers in layer 2, and can suspend, halt, or reorganize elections in emergencies.
2. *Layer 2:* Multiple returning officers/election administrators assign/remove block validators in layer 3 and participate in token generation.
3. *Layer 3:* Multiple polling officer nodes ensure proper voter identification, accurate vote recording, block generation, validation, and propagation and contribute to token generation.
4. *Layer 4:* Public layer with open access for block verification, promoting transparency and accountability. Public nodes cannot validate blocks but can access all network blocks.

In the HAC model, votes are recorded in blocks and authorized by trusted validators from the third layer, ensuring that only valid votes are counted. To enhance transparency and accountability, a trusted signing mechanism traces each vote to its validator, allowing every public node to confirm the authenticity and accuracy of each vote. This ensures a fair and transparent election process, with the mechanism adaptable to various election requirements through additional layers and permission/access variations.

3.4. Method for Verifying the Credibility of a Block

Maintaining block credibility is vital in decentralized and secure blockchain networks. In our proposed HAC mechanism, blocks require multiple authorization levels due to its layered permission and access structure. All nodes have secret public–private key sets, while authorized nodes from layers 1, 2, and 3 share their public keys with public nodes for block verification. Public nodes obtain these keys from trusted sources like the decentralized KMS server, Election Commission server, or other trusted network nodes.

Before accepting blocks, each node verifies every section, including the corresponding public key. Malicious nodes attempting to tamper with blockchain data are detected and isolated, ensuring network integrity and security. The HAC hybrid consensus model validates each block per layer and ensures public verifiability of each block section, fostering multiparty confidence and enabling voter verification from the public layer.

4. The Proposed Blockchain-Based E-Voting System

Our proposed voting system is designed to ensure a secure and fair election process. It encompasses several key features such as decentralization, security, cost-effectiveness, voter verifiability, auditability, anonymity, fairness, and ease of use. These features guarantee that all voters have an equal opportunity to cast their ballots, which are then accurately and securely counted. To verify each voter's information, we use biometric and facial recognition systems. These systems authenticate user information retrieved from the National Identification Database (NID) through a secure cryptographic signature mechanism. To prevent fraud and double voting, we've developed a multiparty cryptographically secure token verification system. This system ensures the protection of the voter's identity and privacy while checking whether a valid voter has already cast their vote.

The system architecture (Figure 2) consists of three main layers:

1. *Application Layer:* Includes components like the EVM unit, Ballot unit, and Script panel.
2. *Network Layer:* Consists of elements such as the P2P network, IoT devices, lightweight devices, and database.
3. *Consensus Layer:* Incorporates components like the lookup table, shard management system, blockchain, Script execution, and proof of Hierarchical Access and Control (HAC) for each block.

The system is designed to accommodate real-world electoral systems, incorporating roles such as returning officers (election administrators) who oversee polling stations in specific areas. Polling officers, acting as block validators or miners, are authorized by returning officers, creating a multi-layer system of permission and access. This structure ensures the integrity and security of the voting process, establish voter trust and confidence. The following section discusses various users and their roles in the electronic voting system.

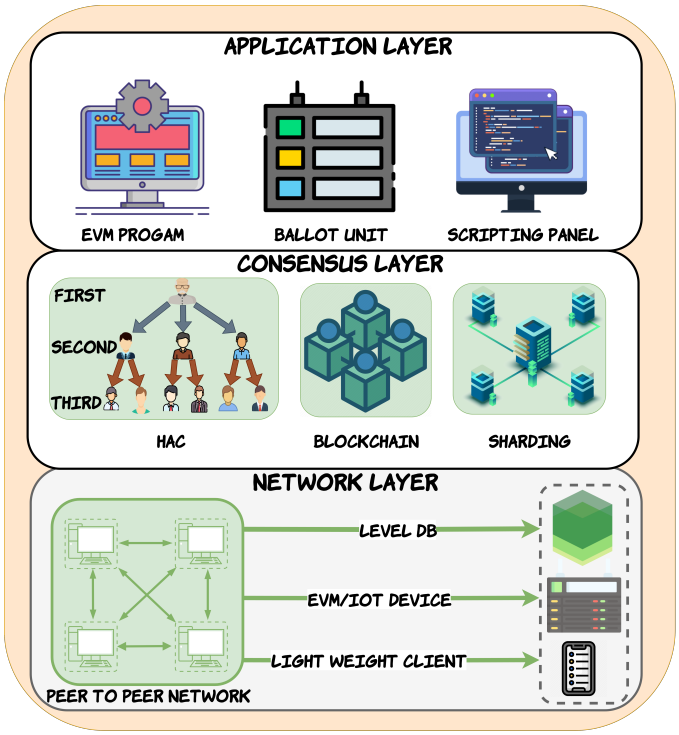

Figure 2. Node architecture of the proposed system, featuring three key layers: Application Layer for user interaction and services, Consensus Layer for maintaining node agreement, and Network Layer for managing internode communication.

4.1. Roles of Participants

To organize a national-level presidential election, certain rules and regulations must be enforced. Here is a basic roadmap outlining how various entities can interact, participate, and fulfill their roles and duties in a blockchain-based e-voting system to ensure compliance with electoral regulations.

- *Election Commission (EC):* The EC is a powerful entity that oversees the election process. They can assign or remove election administrators, suspend or halt elections, and reorganize elections in the case of emergencies or corruption. The EC can be composed of multiple entities to reduce the risk of a single point of failure, and we have used threshold cryptography to provide more robust security over the system.
- *Returning Officer (RO):* The RO assigns or removes block validators (e.g., polling officers), participates in token generation, and oversees elections in their authorized areas, ensuring compliance with rules and regulations.
- *Polling Officer (PO):* The PO is responsible for accurate voter identification, recording votes, and maintaining ballot records, effectively making them block validators or miners in the system.
- *Candidate:* A candidate is a person seeking election to a position of authority.
- *Voter:* A voter is a registered individual who can cast a vote for their chosen candidate. Upon successfully casting their vote, they receive a secret token to verify if their vote has been accurately stored in the system.

4.2. Entities

In a real-world scenario, an election needs to be supported by multiple external entities such as the National Identity card system (NID), NID server, blockchain network, scripting mechanism, key management server, and token verification. Some of these entities are briefly described below.

- *NID system:* A unique government-issued ID for each voter is essential to perform election tasks accurately and prevent fraudulent or double voting. The NID server securely manages voter information, participates in token generation, and uses cryptographic signatures to prevent data tampering, ensuring accurate voting.
- *Blockchain Scripting:* A transparent and efficient scripting mechanism enables predefined operations for authorized personnel, ensuring fair elections and surpassing smart contracts in speed and security.
- *Multiparty Token Generation System:* A secure multiparty computation system generates unique tokens for voters to cast, preventing double voting and protecting voter identity.

4.3. System Design

To ensure the smooth functioning of the main voting system, a backend server acts as the primary controller for most functions. Specifically, a local copy of the blockchain or its shards is stored in a key-value pair database, called LevelDB, on the backend server. This server, built on Gin, a web framework written in Go, manages the local database, communicates with the blockchain network, and connects with a decentralized key management server (KMS). Moreover, the server handles communication with the NID server, block generation, token generation, and script execution mechanisms. To provide a clearer understanding of the system, a detailed overview is presented in Figure 3.

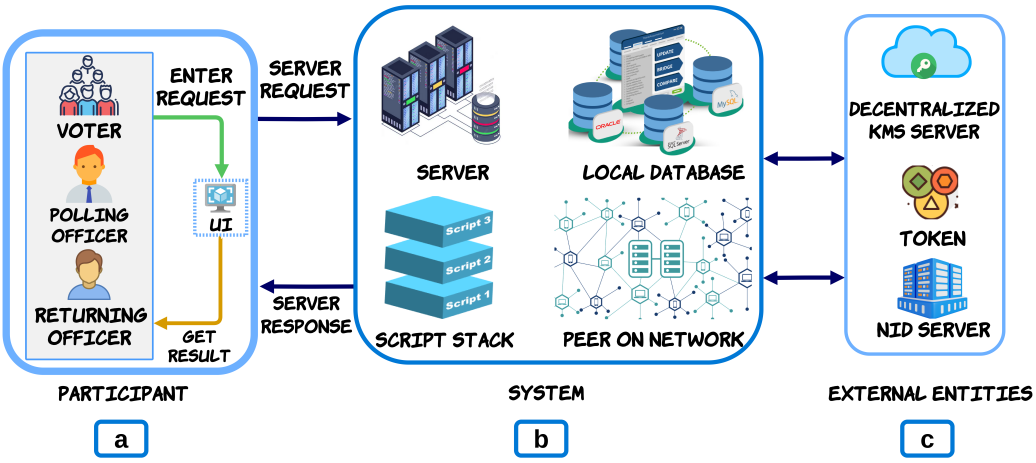

Figure 3. This diagram visually illustrates the proposed election process, highlighting interactions among various participants, system components, and external entities. (**a**) Participants include voters, polling officers, and returning officers. (**b**) System components encompass the server, local database, scripting system, and peers on the network. (**c**) External entities involve the decentralized key management server (KMS), token generation system, and NID server.

4.3.1. Initialization Phase

Before being deployed for an election, our system must undergo several preprocessing steps. These steps include retrieving various public keys from the KMS server, initiating a connection between the NID server and relevant nodes, and retrieving the election configuration, voter list, candidate list, and all required commands with proof. After retrieving and validating all the required data and scripts, the EVM system executes all scripts and updates the system accordingly.

4.3.2. Registration Phase

Before being deployed for an election, our system must undergo several preprocessing steps. These steps include retrieving various public keys from the KMS server, initiating a connection between the NID server and relevant nodes, and retrieving the election configuration, voter list, candidate list, and all required commands with proof. After retrieving

and validating all the required data and scripts, the EVM system executes all scripts and updates the system accordingly. Users must register with their NID, institutional ID, public key, and unique Digital ID for the hierarchical-layer-based consensus mechanism. Registration includes real-time deep learning face recognition and verification for participant identification in blockchain scripting commands.

1. *Returning Officer (RO):* An administrator overseeing a specific area and its polling stations, assigned by the Election Commission (EC). ROs register with credentials and can access their panels to create and publish commands in the script stack to manage the election process.
2. *Polling Officer (PO):* A person appointed by the RO to oversee the voting process at a polling station. A PO registers with their credentials and can access their panel to activate the voting machine and ensure a secure and orderly voting process for all eligible voters.
3. *Voter:* An individual eligible to vote in an election. Voters register their information, pictures, and biometrics, which are stored in the NID server. They have a smart NID card containing their NID number, photos, and biometrics to verify their identity at the polling station.

DeepFace Algorithm: The face recognition feature is an additional layer of authentication in the system, and it is used to verify users before they can access their respective systems as shown in Figure 4 with an algorithm. The system is built using DeepFace, a face recognition and facial attribute analysis library for Python, OpenCV, and TensorFlow.

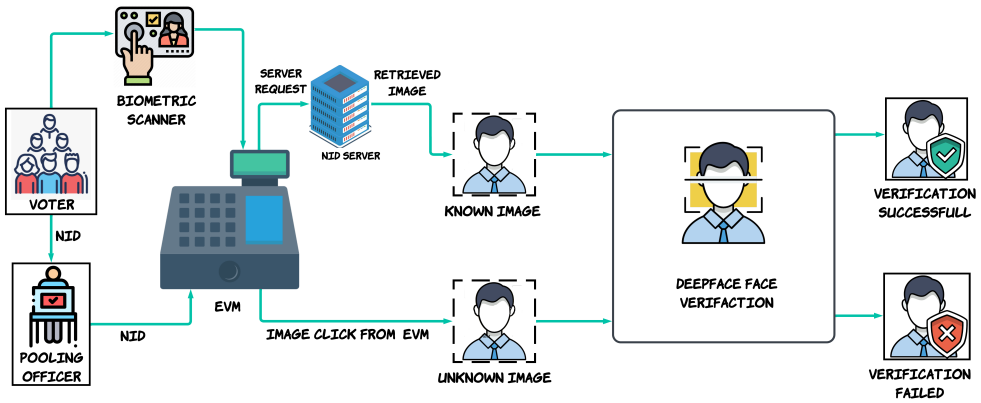

Figure 4. The overview of the biometric face verification system. From the figure, we can see how the system scans the voter's face in front of the EVM unit. DeepFace, a facial recognition and facial attribute analysis deep learning model, has been employed to compare the image of the voter with the database image to ensure voter integrity.

After conducting rigorous tests, we determined the threshold to be approximately 0.10, as it showed optimal performance in face recognition and facial attribute analysis. Algorithm 1 shows the use of face recognition using the DeepFace package. *KnownIMG* is the known image obtained from the system database, and *UnknownIMG* is the image clicked through the system. "R" is the details of the face match records between two different images, and "d" is the distance of difference.

Algorithm 1 Proposed face recognition algorithm using DeepFace package

function VERIFYFACE(*VoterNid*, *BiometricData*)
    KnownIMG ← import image from database using *VoterNid*
    UnknownIMG ← import recently clicked image from *BiometricData*
    R ← DeepFace.verify(KnownIMG, UnknownIMG)
    return *R.dist* $\leq$ 0.10
end function

4.3.3. Blockchain Command (Scripting System)

Smart contracts are self-executing digital agreements that utilize computer codes to enforce the terms of an agreement without intermediaries. However, smart contracts face issues such as coding vulnerabilities, complex designs, high power consumption, high transaction costs, reliance on external data, and excessive user authority. Regulatory uncertainty and a lack of confidential execution make them incompatible with electoral rules in many countries [38]. This study proposes a stack-based scripting mechanism that is compatible with the existing regulatory system, is simpler, and provides users with a controlled level of access and permission with less computational overhead. As shown in Figure 5, this scripting command allows the Election Commission and Returning Officer to have appropriate control over the blockchain and the panel to perform their duties.

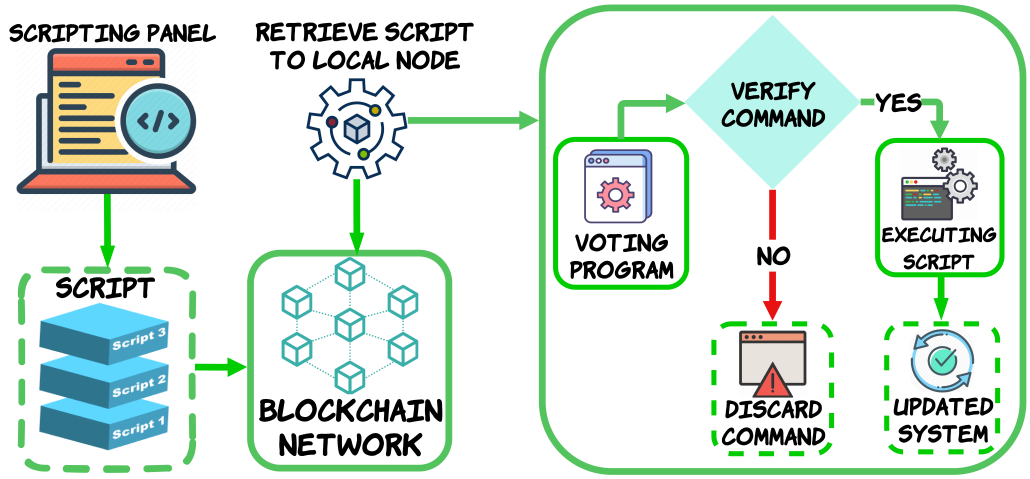

Figure 5. The diagram visually explains the proposed scripting mechanism. It shows how a scripting panel pushes commands in the script stack and publishes them on blockchain networks. Other nodes retrieve the script from the network to their local machines, verify its authenticity, execute the command, and update the system accordingly.

4.3.4. Authorization and Verification Process

In order to maintain the security and authenticity of each record in the blockchain, such as a "polling officer" record, certain fields like name, ID, designation, assigned polling station, and Digital ID are hashed to create a unique digest of the record. This digest is then signed by an authorized entity, such as a returning officer, using its private key to generate a verifiable certificate as proof of authenticity. This digest can also be used by the returning officer to create a command on the blockchain, authorizing the individual as a polling officer. Moreover, the record verification process involves the use of an authorized entity's public key to decrypt the signature. Successful decryption confirms the authenticity and integrity of the record, as shown in Figure 6. This verification process is essential for ensuring transparent verification and preventing fraud in blockchain systems.

Authorization algorithm: In our layer-based hierarchical consensus system, we often need to obtain authorization from a higher level before authorizing a lower-level entity. To support this bureaucratic system, we have implemented an authorization algorithm.

A detailed algorithmic representation of the peer-routing algorithm is provided in Appendix C.4 of Appendix C.

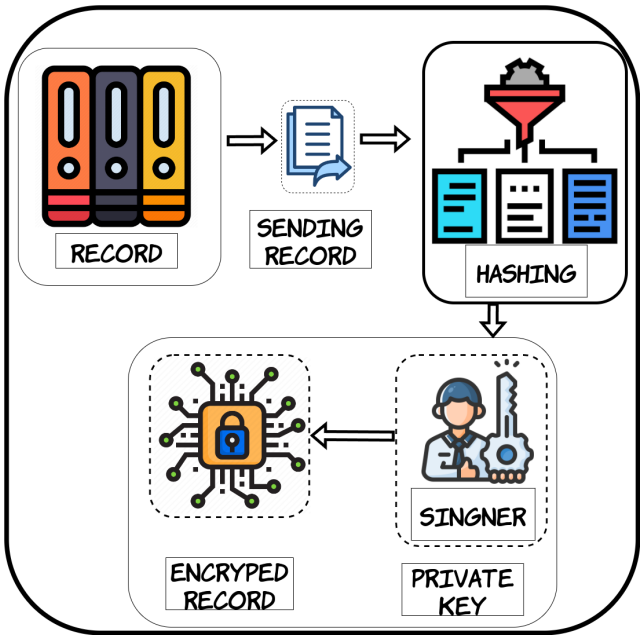

Figure 6. This diagram presents a lucid explanation of the authorization process. It shows how every entity is hashed, signed, and generates a proof of record.

4.3.5. Peer Routing Algorithm

Peer routing is a key process for discovering nodes within a network and building a decentralized structure. A detailed algorithmic representation of the peer-routing algorithm shown in Algorithm 2.

Algorithm 2 Peer Routing Algorithm

```
bit ← getLatestBlock()
cfg ← getConfig()
host ← makeNodeHost(cfg)
kademliaDHT ← dht.New(ctx, host)
kademliaDHT.Bootstrap(ctx)
for peerAddr in cfg.BootstrapPeers do
    peer ← peerChan()
    if error in host.Connect(ctx, peer) then
        print "Connection failed: <error message>"
        continue
    end if
    if host.SendBlockData(bit, peer) is successful then
        print "Connected to: <peer>"
        host.Peerstore().AddAddrs(peer.ID, peer.Addrs, peerstore.PermanentAddrTTL)
    else
        print "Connection failed"
    end if
end for
```

4.3.6. Election Process

To ensure the integrity of the election, only authorized individuals such as election administrators or returning officers can create polls. They use their private keys to sign the poll data, which is then added to the poll script published on the blockchain network. This script contains all the necessary information to verify the authenticity of the poll, including the returning officer's identity, polling station details, candidate details, and election time. Any computer in the network can verify the signature and script to ensure that the poll is valid. Once a poll is created, it can start, and polling officers can begin their work. The poll

data are stored on the blockchain in a way that cannot be altered, giving voters confidence that their votes are being counted correctly and that the election results are accurate.

Voters present their smart NID cards to the polling officer, who scans them and verifies voter information, shown in Figure 7. Biometric facial recognition further authenticates voters against the NID database before proceeding with the blockchain-based election.

A unique token is generated for each voter to ensure one-time participation. The system verifies whether the tokens have been used previously. Otherwise, the ballot is activated, allowing the voter to vote. After the voter casts a vote, it is encrypted and stored in a block on the blockchain network. A receipt is provided to the voter. This ensures that votes are secure and that election results are accurate. As shown in Figure 8, the voting process is automated and transparent. Voters can verify that their votes have been cast and counted accurately by checking the blockchain. This process enhances the security and transparency of the election, thereby increasing public trust in the electoral system.

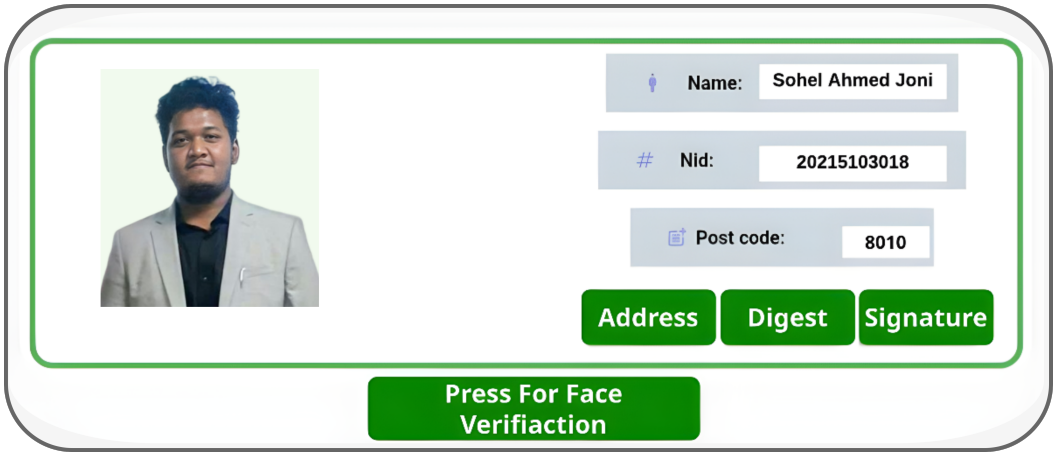

Figure 7. A visual representation of the EVM system that displays voter information, which includes names, addresses, and voter ID numbers.

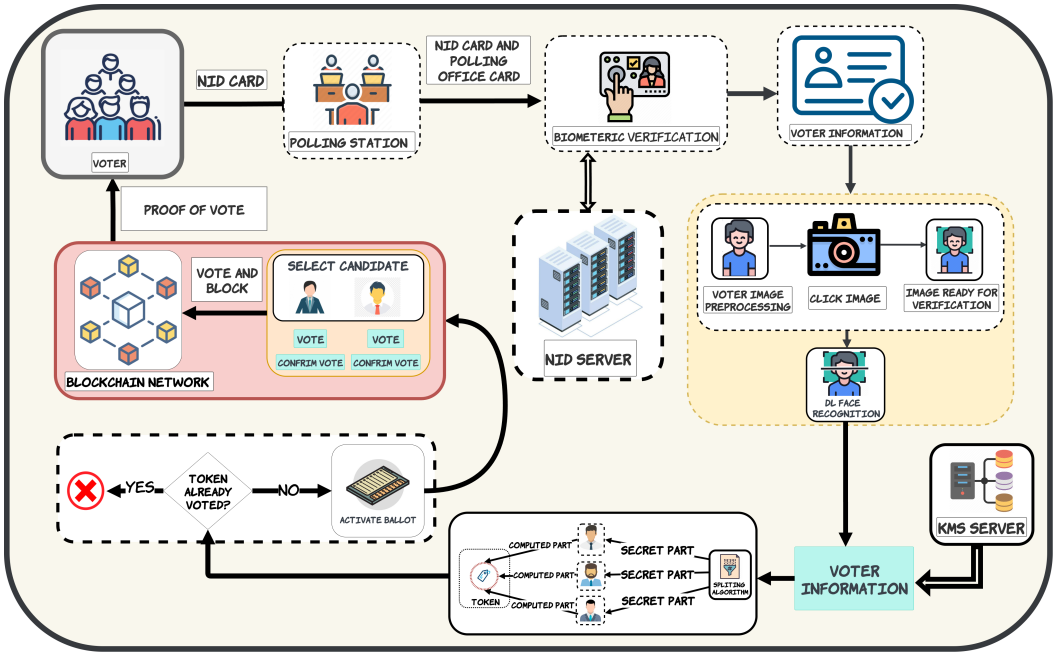

Figure 8. Voters present smart NID cards and undergo biometric facial verification for identification, as illustrated in Figure 4. A unique token is generated as mentioned in Section 4.6, which prevents multiple voting. Verified voters cast encrypted votes stored securely on the blockchain, ensuring integrity. Post-voting verification is also available.

Proposed Vote Casting Algorithm: The algorithm for casting votes is illustrated in Algorithm 3. To cast a vote, a voter is required to provide his or her NID number and biometric information. Next, this information is sent to the NID server, which can verify the voter's identity. If the verification is successful, the server provides the voter with a unique token that they can use to cast their vote.

Algorithm 3 Proposed Vote Casting Algorithm

```
Declare a block
Block ← new BlockStruct
Get voter information
function GetVoter(VoterNid, BiometricData)
    VoterStruct ← NID, Name, PSCODE, Address, AvatarDigest, Signature
    return VoterStruct
end function
CheckVoter
function CheckVoter(NID, Name, PSCODE, Address, AvatarDigest, Signature)
    digest ← Hash(NID, Name, PSCODE, Address, AvatarDigest)
    if Check(Signature, publicKey, digest) ≠ true then
        return Invalid
    end if
    if digest ∉ ApprovedVoterList then
        return Invalid
    end if
    token ← Hash(Rsig_seed + Psig_seed + EID + VoterStruct.digest + Rnd())
    if token exists in spendedVoters then
        return DoubleVote
    end if
    return ValidVoter
end function
Getting Vote from Ballot unit
Block.vote ← GetVote(seed, keyp, Randp)
Sign the block
BlockDigest ← generateBlockSignature(Block)
Block.Signature ← generateBlockSignature(BlockDigest, Privatekey)
Append Block to The chain
Nextblock ← ...                                   ▷ Next block in the chain
```

Ballot Tallying: After the election, votes are tallied using cryptographic proofs, ensuring only valid votes are counted. All the pertinent election data, including cryptographic proof, random functions, and encryption keys, are made public for everyone to check their results themselves. This open and verifiable process ensures that everyone can trust the election results. The detailed algorithmic representation of the ballot tallying is available in Algorithm 4.

4.4. Incorporating Category-Based Sharding in Blockchain Network

As blockchain networks grow, nodes struggle with data transmission, reception, management, and storage. Sharding addresses these issues by dividing the blockchain into smaller subnetworks (shards) that handle distinct data subsets. Parallel transaction processing across shards improves scalability and performance. However, sharding can introduce challenges like increased latency and reduced throughput due to cross-shard communication. To tackle these challenges, we propose category-based sharding (Figure 9). This solution groups voting data by election area and polling station, generating shards based on polling stations and categorizing them by election area shown in Figure 10. A distributed lookup table efficiently redirects requests to appropriate nodes, minimizing internode communication and enhancing data accessibility. This improves overall scalability, per-

formance, and availability of blockchain networks, overcoming challenges introduced by traditional sharding techniques.

Algorithm 4 Tallying Votes

```
function TALLYINGVOTES(blockchain, CanidatelistVotes)
    votes ← empty map
    for each block in blockchain do
        votes ← getVotesFromBlock(block)
        for each vote in votes do
            decryptedVote ← decrypt(vote, Ballotkey)
            if decryptedVote is in CanidatelistVotes then
                votes[decryptedVote] ← votes[decryptedVote] + 1
            end if
        end for
    end for
    return votes
end function
function GETVOTESFROMBLOCK(block)
    return block.vote
end function
```

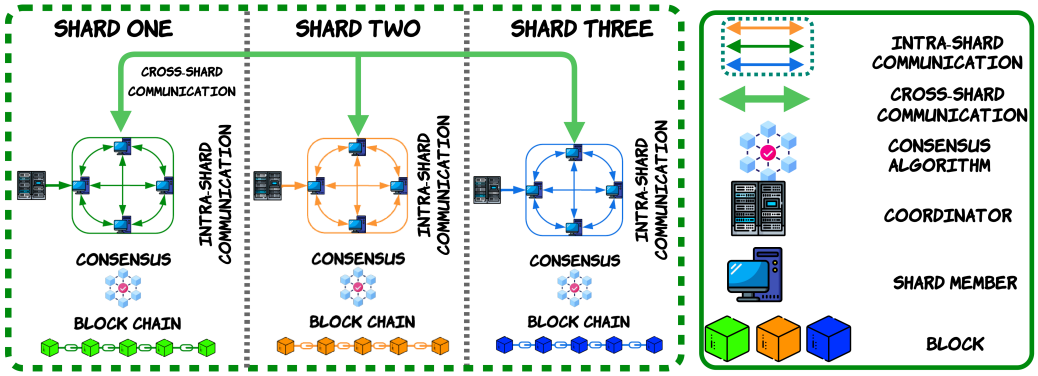

Figure 9. The diagram illustrates category-based sharding, dividing a blockchain into categories with dedicated shards for data storage. A coordinator manages shards, and a lookup table directs category queries. Intra-shard and cross-shard communication occurs via a peer-to-peer network.

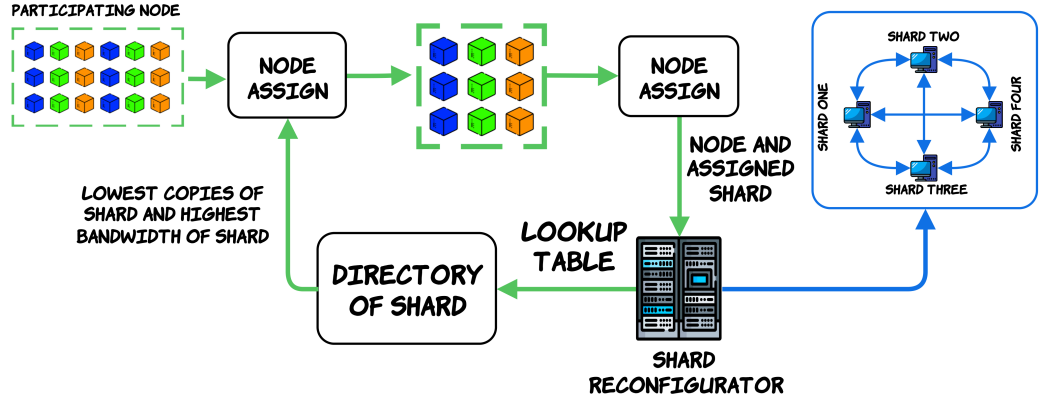

Figure 10. The diagram illustrates shard assignment for a specific category and node. The shard reconfigurator and coordinator manage the process, optimizing shard assignments based on network traffic and copy count.

4.5. Incorporating Post-Quantum Asymmetric Encryption in Blockchain

To ensure transparency and secure authorization, our PQMPCHAC-Bchain model uses public key cryptography, digital certificates, and hash functions for secure authorization and transparency. However, with the advancement of quantum computing, potential attacks based on Grover's and Shor's algorithms pose a threat to these cryptographic systems. To address this, we implemented and tested Dilithium, a post-quantum asymmetric encryption algorithm selected by NIST for standardization, on our blockchain. We tested two versions, Dilithium2 and Dilithium3, with the latter offering higher security but slower signing speed and larger keys. Incorporating quantum-resistant cryptography ensures the long-term security and reliability of our blockchain system.

4.6. Token-Based Voter Verification System

Double voting is a serious issue in voting systems, occurring when someone votes twice in the same election, either accidentally or intentionally. To ensure a fair election, it is crucial to prevent double voting. To address this, we introduced a cryptographically secure unique token generation and verification system to defend against fraud and double-voting activities.

As shown in Figure 11, the proposed system generates a unique temporary token for each voter by using MPC. MPC is a cryptographic technique that allows multiple parties to jointly compute a token over their inputs while protecting the voter's identity. This means that no single party can learn anything about other parties' inputs, even if they collude. The token can only be generated using the voter's identity and a secret key during the election, and cannot be regenerated for the same voter once the election is over. Because the token is generated using MPC, core encryption is secure even if some parties are compromised, ensuring that the voter's privacy remains intact. For a detailed overview of the time inference logic in multi-party computation, please see Figure A25 in Appendix B. Our system provides a secure and reliable solution to prevent double voting in elections.

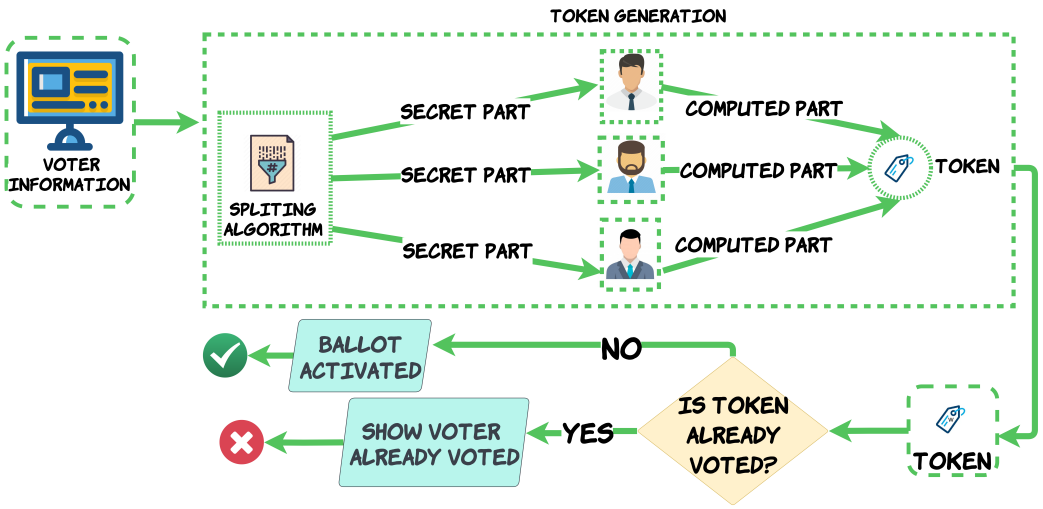

Figure 11. The high-level representation of the structure of token-based verification in the system shows how voter information is split and computed into a token, which is then used to verify double voting without revealing the voter's identity.

Figure 11 illustrates the token verification system. This method has several advantages over the traditional voting systems. First, MPC is more secure because it is using MPC to protect voter identities. Second, the process is more transparent. Thus, the uniqueness of the vote can be verified while protecting the voter information. Third, the token system is more convenient because voters can cast their votes anywhere.

Instead of directly storing or checking the voter's identity, we verified the voter's identity using a privacy-preserving token. This token is cryptographically linked to the

voter's identity, but cannot be traced back to the voter's identity. We then checked this token against a list of spent tokens to ensure that the voter had not already voted. This ensures that voter's privacy is preserved. The working procedure of this system can be found in Appendix C.1 of Appendix C for a detailed overview of the time-inference logic in multi-arty computation.

Token generation: This system enables multiple parties to collaborate in computing a function based on a voter's unique identifier without disclosing the voter's identity to any of the parties involved. A detailed algorithmic representation of the token generation system is available in Algorithm 5.

Algorithm 5 Formula of Token Generation

```
function SliceSecret(secret, n)
    parts ← empty list
    partSize ← length of secret/n
    for i ← 0 to n do
        parts[i] ← secret[i * partSize : (i + 1) * partSize]
    end for
    return parts
end function
function MultyPartyComputation(secret, n, seed, validTime)
    parts ← SliceSecret(secret, n)
    shares ← empty list
    for i ← 0 to n do
        if validTime.contains(Time()) then
            shares[i] ← parts[i] ⊕ seed
        else
            shares[i] ← parts[i] ⊕ 0
        end if
        shares[i] ← Hash(shares[i])
        shares[i] ← Sign(shares[i], parties[i].privatekey)
    end for
    return shares
end function
function MergeShares(shares, n, parties)
    token ← empty string
    for i ← 0 to n do
        if Verify(shares[i].signature, parties[i].publickey) is Valid then
            token ← token ⊕ shares[i]
        else
            return invalid
        end if
    end for
    return token
end function
function GenerateToken(secret, n, seed, validTime, privatekey, parties)
    shares ← MultyPartyComputation(secret, n, seed, validTime)
    token ← MergeShares(shares, n, parties)
    token ← Hash(token)
    token ← Sign(token, privatekey)
    return token
end function
```

4.7. Block Structure: A Modular Approach toward Efficiency

In the context of election data, the existing ledger structure and management system cannot fully reduce redundant data while accommodating the HAC. To address the storage capacity and layer-based authorization and verification issues in blockchain, we propose

a novel modular approach. Block modularity stores data as independently verifiable modules, links them in the blockchain, stores a copy of these modules in a local database, and stores their Merkle hash in a block. We store, hash, and sign one or more fields in each record as separate units, ensuring their modularity and independent verification. An e-voting platform optimizes this modular approach by processing only the updated section and referencing previously validated data, thereby minimizing duplicate processing and reducing unnecessary verification. Figure A25 in Appendix B provides a visual representation of our block and its units. The proposed modular approach not only addresses the storage capacity issue but also enhances the layer-based authorization and verification process in blockchain.

Block Digest Generation: Each field of the block generates a hash code. We combine these hash codes to produce the digest of the entire block. During the propagation of the block, each record possesses a verifiable certificate, providing proof of authenticity. Appendix A Figure A24 provides an additional visual representation.

4.8. Voter Data Generation

To test the scalability of our proposed system, we simulated it with a large number of voters across various regions. Since our proposed system leverages the discrete nature of election data and the hierarchical structure of each electoral region, we required a substantial amount of voter and election data. However, sensitive voter datasets are scarce and often lack the necessary features we were looking for. Therefore, we had to create our own dataset targeted at Bangladesh. Using geolocation data, Bengali name datasets, and the Faker library, we generated 113,500 synthetic voter records through subsampling and combinatorial approaches. This dataset [39] closely resembles real-world voter data and includes a digest field for immutability testing and blockchain applications. The voters are well distributed across the regions, which helps in creating an ideal environment for testing and simulation, as shown in Figure 12.

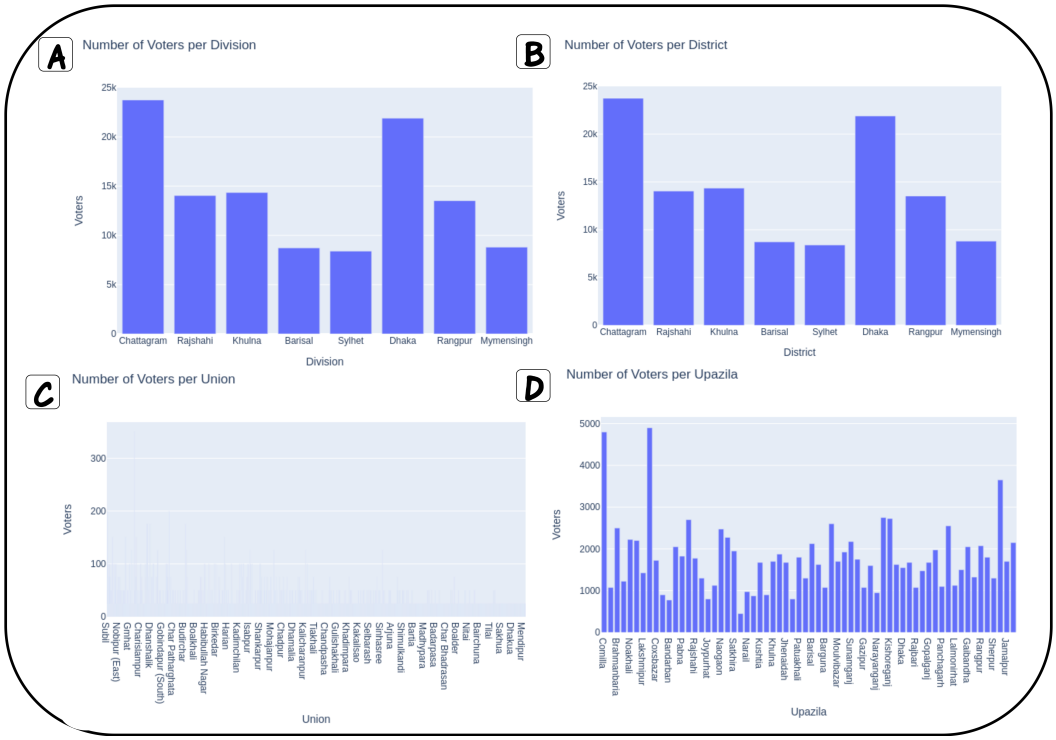

Figure 12. Data generation process for voter distribution. The bar graph illustrates the hierarchical breakdown of voter counts across different administrative divisions: (**A**) number of voters per division, (**B**) number of voters per district, (**C**) number of voters per union, and (**D**) number of voters per upazila.

5. Implementation

We built a custom blockchain-based e-voting system from scratch, without relying on existing libraries or frameworks. This approach allowed us to focus on essential features and security measures. The system comprises three main components: the Network Stack, the Execution Stack, and the Consensus Stack.

1. *Network Stack:* This component is mainly responsible for establishing communication with various internal and external entities, such as the NID server, KMS server, MPC network, and handling peer routing for p2p tasks.
2. *Execution Stack:* The primary role of this stack is to execute various operations, including block generation, e-voting operations, shard management, face verification, token generation, scripting execution, and more.
3. *Consensus Stack:* This component focuses on various authorization and verification processes, including but not limited to new block verification, script verification, HAC tree verification, and malicious node isolation. Only after successful verification can data be appended to the blockchain, ensuring the integrity and security of the system.

To create a platform-neutral and extensible serialized data structure for blocks, we used protocol buffers. We also employed cryptographically secure hash functions (SHA256, Blake2b) and binary encoders (Base58) to generate secure, append-only timestamped blocks, ensuring the blockchain's security and immutability. For asymmetric cryptography and digital signatures, we integrated Cloudflare's CIRCL library, including algorithms like Ed25519, Dilithium2, and Dilithium3. This enhances the blockchain's verifiability and immutability. We used the Kademlia DHT subsystem within libp2p for peer routing, ensuring efficient and secure communication between network nodes. To prevent double-spending, we implemented a secure token-based verification system using Universally Unique Identifiers (UUIDs) and zero-knowledge proofs, prioritizing transparency, privacy, and tamper-resistance. For blockchain data storage, we chose LevelDB, a fast key-value storage solution offering efficient data management. Our secure token generation system verifies whether voters have already cast ballots without revealing their identities, preventing double voting. We optimized the search process within spent tokens using a red-black tree-supported tree-set data structure. For face verification, we employed the Deepface framework, which compares facial images using Euclidean similarity metrics to determine if they belong to the same individual. We developed a backend using the GIN web framework, providing a REST API for managing requests and actions and ensuring smooth communication between system layers. Additionally, we created a user-friendly graphical user interface (GUI) frontend using Flutter, with seamless communication with the backend via REST APIs. To enhance the EVM system, we implemented an NID server using GIN and SQLite 3, providing verifiable and tamper-proof voter data, ensuring voting process integrity. The NID server demonstrates the system's practicality and effectiveness in real-world contexts. It provides the infrastructure for verifying voter identity and maintaining voting process integrity, offering a secure, transparent, and efficient e-voting solution.

6. Performance Analysis

We conducted blockchain system experiments, focusing on block generation rate, throughput, and memory usage based on sharding, post-quantum cryptography, and block modularity. We integrated advanced post-quantum cryptographic algorithms (Dilithium-2 and Dilithium-3) with general elliptic curve cryptography for robust security. The experiments explored the benefits and challenges of sharding without block modularity, sharding with block modularity, and system performance without sharding. We also investigated the system's behavior without sharding and block modularity. The aim was to assess the impact of sharding and block modularity on system performance and efficiency.

6.1. Experimental Setup

To evaluate the performance and effectiveness of our proposed system, we conducted a series of experiments. The experimental setup included the following components:

- *Hardware*: The experiments were conducted on a Desktop with AMD 5600 g processors, 8 GB of RAM, and 1 TB of SSD storage.
- *Software*: The backend server was built on Gin, a web framework written in Go. The blockchain network was implemented using a custom blockchain framework.
- *Datasets*: We generated synthetic voter data using geolocation data, Bengali name datasets, and the Faker library. The Bangladesh-Voter-Synthetic-Dataset [39] dataset included 113,500 synthetic voter records, distributed across various regions.

6.2. Experiment without Sharding and Post-Quantum

The block modularity graph (a) shows that the block modularity approach outperforms the non-modular approach in block generation time as the number of voters increases beyond 1000. This is because block modularity segments system data into smaller categories, preventing data duplication and enabling faster processing and validation of new data.

The throughput bar chart (b) illustrates that the block modularity approach surpasses the non-modular approach in network performance over time as the number of voters increases. Initially, the non-modular approach showed higher throughput, but the block modularity approach achieved 2 times of throughput compared to the non-modular approach's for 3000 nodes.

The line graph in Figure 13a and bar chart (b) depict the block generation rate and throughput of a blockchain network with and without block modularity, respectively. Table A1 in the Appendix D provides detailed experiment data.

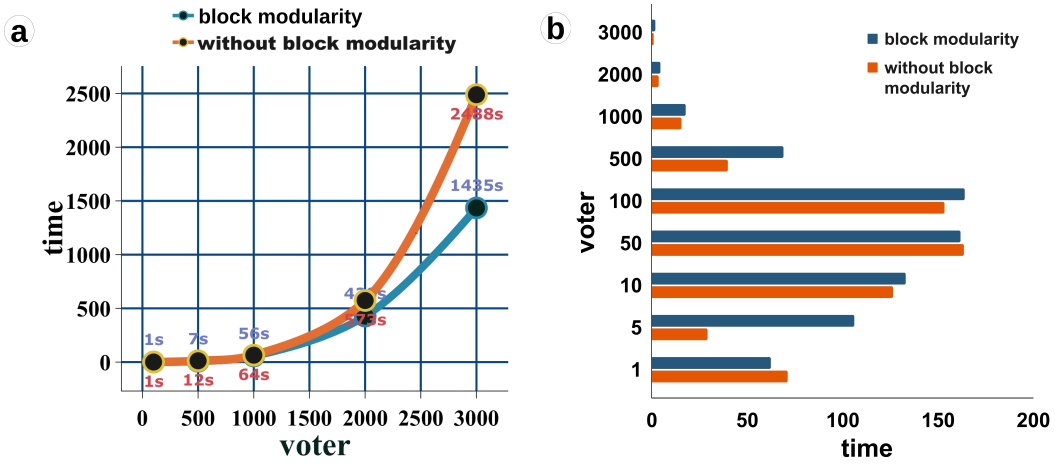

Figure 13. Comparison of Block Generation Time (**a**) and Throughput (**b**) with and without Block Modularity. (**a**) The blue line represents block modularity, while the orange line depicts the scenario without it. The horizontal axis shows voter count, and the vertical axis displays time in seconds. Block modularity demonstrates lower block generation times as voter count increases due to efficient data segmentation. (**b**) The vertical axis represents voter count, and the horizontal axis denotes time in seconds. Block modularity (blue line) shows significantly better throughput compared to the scenario without it (orange line), highlighting enhanced performance.

Block modularity enables each node to process and validate only relevant blocks, while the non-modular approach duplicates all data, leading to increased memory consumption, slower block generation, and lower throughput.

Therefore, block modularity significantly improves block generation rate and throughput as the number of voters increases in a blockchain network.

6.3. Results after Adding Post-Quantum

To provide a comprehensive analysis of post-quantum cryptography, we conducted experiments on four variants of two post-quantum algorithms. Specifically, we evaluated Dilithium-2 with block modularity, Dilithium-2 without block modularity, Dilithium-3 with

block modularity, and Dilithium-3 without block modularity. Furthermore, we compared these post-quantum algorithms across various criteria to gain a better understanding of their performance and effectiveness.

1. First, we compared Dilithium-2 with block modularity to Dilithium-2 without block modularity. The results are shown in Appendix A.1.
2. Second, we compared Dilithium-3 with block modularity to Dilithium-3 without block modularity. The results are shown in Appendix A.2.
3. Third, we compared Dilithium-2 with block modularity to block modularity by itself. We also compared Dilithium-2 without block modularity to block modularity by itself. These results are shown in Appendix A.3 and Appendix A.4, respectively.
4. Similarly, we compared Dilithium-3 with block modularity to block modularity by itself. We also compared Dilithium-3 without block modularity to block modularity by itself. These results are shown in Appendix A.5 and Appendix A.6, respectively.
5. We compared Dilithium-2 vs. Sharding and Dilithium-3 vs Sharding. The results are shown in Appendix A.7.

In summary, Figures 14 and 15 show that Dilithium-3 with block modularity outperformed the other algorithms in terms of block generation time and throughput.

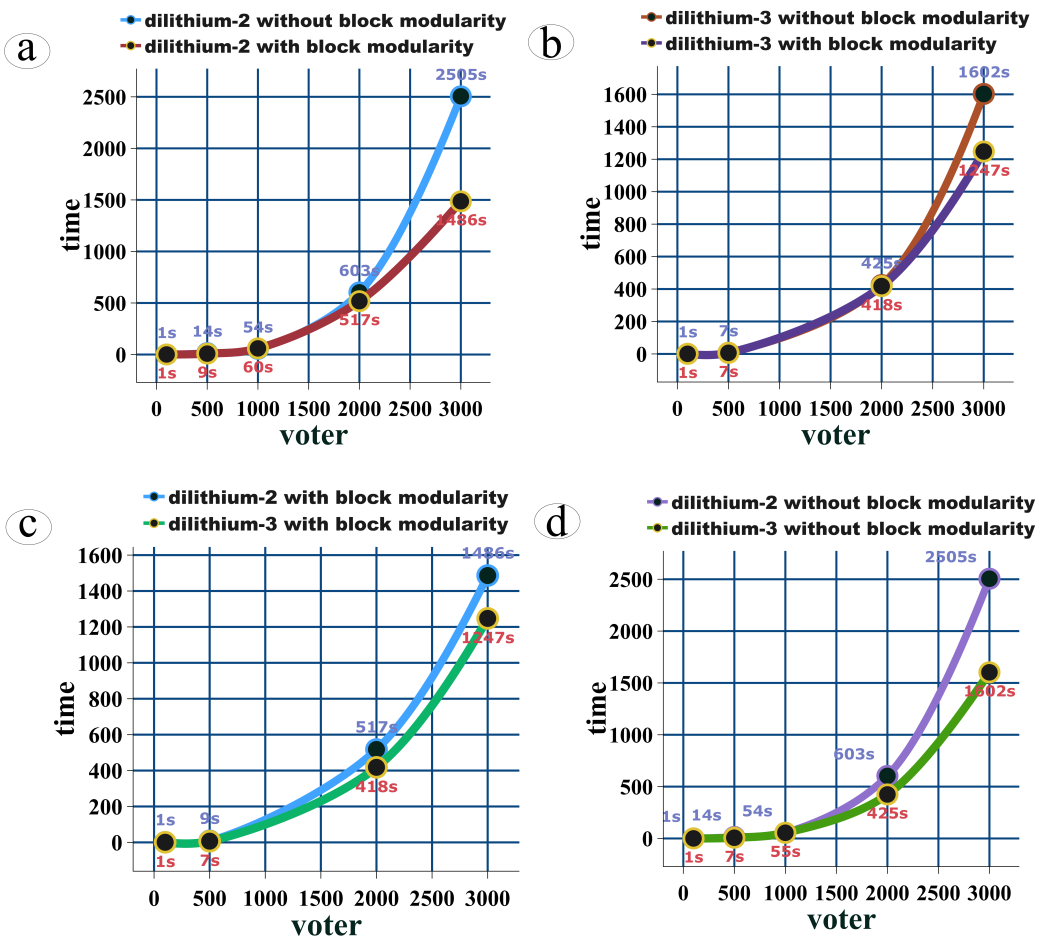

Figure 14. This graph compares Dilithium-2 and Dilithium-3 performances. Here, Figure (**a**) shows Dilithium-2 without (sky blue) and with (dark red) block modularity, and Figure (**b**) represents Dilithium-3 without (dark red) and with (purple) block modularity. Also, (**c**,**d**) compare both with block modularity—Dilithium-2 (sky blue) and Dilithium-3 (green)—and Dilithium-2 (purple) and Dilithium-3 (green) without block modularity. The horizontal axis depicts voters, while the vertical axis shows time. Dilithium-3 with block modularity generates blocks faster, enhancing performance and security, especially with more voters.

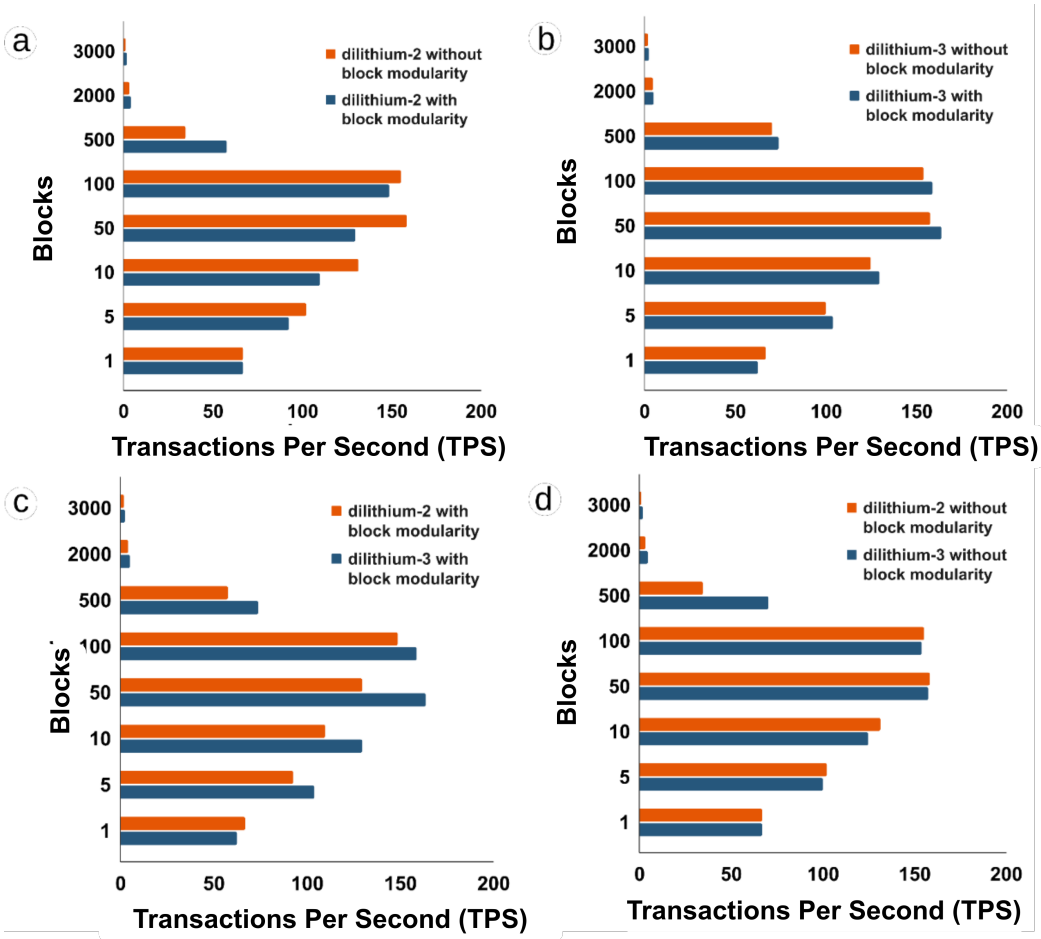

Figure 15. This graph compares Dilithium-2 and Dilithium-3 throughput. Here, Figure (**a**) shows Dilithium-2 without (orange) and with (dark blue) block modularity, and Figure (**b**) represents Dilithium-3 without (orange) and with (dark blue) block modularity. Also, (**c**,**d**) compare both with block modularity— Dilithium-2 (orange) and Dilithium-3 (dark blue)—and Dilithium-2 (orange) and Dilithium-3 (dark blue) without block modularity. The horizontal axis depicts voters, while the vertical axis shows time. Dilithium-3 with block modularity generates blocks faster, enhancing performance and security, especially with more voters.

6.4. Performance after Incorporating Sharding

Sharding is a technique in distributed systems that divides data into smaller parts for better scalability and performance. The comparison data is shown in Table A6 in the Appendix D.

Sharding is a technique for dividing data into smaller parts to improve scalability and performance in distributed systems. The line graph in Figure 16a compares sharding with two, three, and five shards, showing that sharding with five shards provides the best performance as the number of voters increases. At its peak, the green line takes only 28 s to generate 3000 blocks, while the blue and orange lines take 430 and 153 s, respectively. In terms of throughput, the bar graph in Figure 16b shows that sharding with five shards (light green line) outperforms sharding with two and three shards as the number of voters increases, reaching approximately 106 TPS at 3000 voters. Breaking data into more shards can lead to higher performance and throughput.

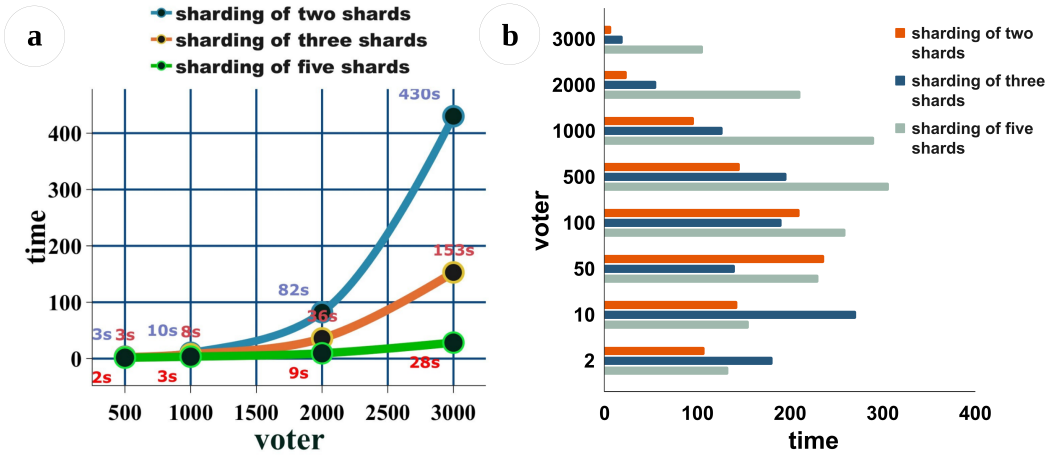

Figure 16. The graph shows the block generation time (**a**) and throughput (**b**) for sharding with two, three, and five shards. Sharding with five shards provides the best performance, both in terms of block generation time and throughput. This is because sharding with five shards distributes the workload more evenly across the shards, resulting in less work for each shard and faster block generation.

6.5. Performance Comparison: Without Sharding vs. with Sharding

In this section, we analyze the performance of our blockchain system with and without sharding, focusing on block generation time and throughput. The results are presented in Figure 17A,B respectively.

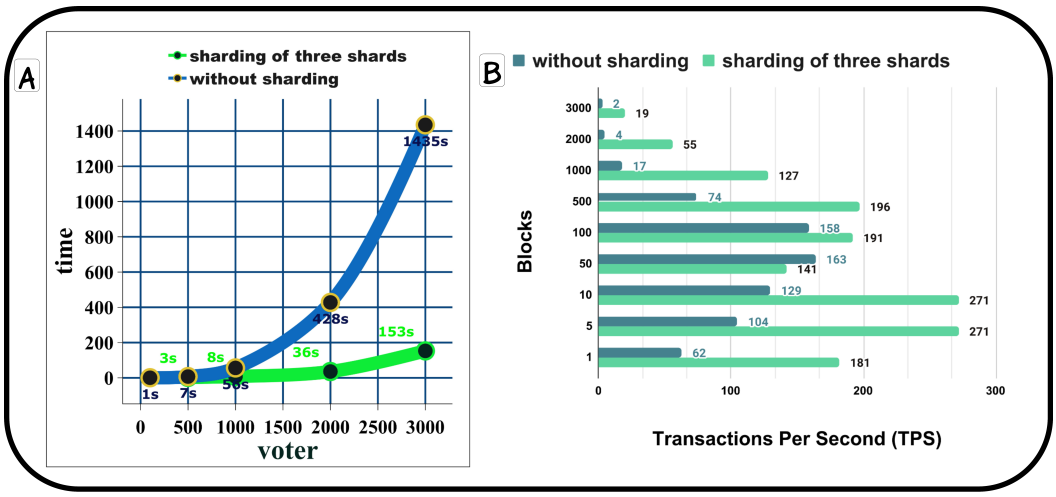

Figure 17. This figure presents a comparative analysis of block generation time and throughput for systems with and without sharding. The line graph (**A**) illustrates the block generation time, where the blue line represents the system without sharding, and the green line represents the system with sharding, demonstrating a significant reduction in block generation time when sharding is implemented. The bar graph (**B**) shows the throughput, with the dark green bar representing the system without sharding and the green bar representing the system with sharding. The results indicate that sharding not only reduces block generation time, but also significantly enhances throughput.

Figure 17A illustrates the block generation time. The blue line represents the system's performance without sharding, while the green line demonstrates the impact of implementing sharding. It is evident that the block generation time without sharding (blue line) is significantly higher than with sharding (green line). This observation underscores the effectiveness of sharding in reducing block generation time.

Figure 17B presents the throughput performance. The dark green line represents the throughput without sharding, and the green line represents the throughput with sharding.

The bar graph clearly shows that sharding outperforms the system without sharding. Initially, the throughput with sharding is 181 TPS (Transactions Per Second), whereas without sharding, it is 62 TPS. In comparison, other blockchains like Ethereum and Bitcoin have throughput's of 25 TPS and 7 TPS. Our system achieves a throughput of 181 TPS, which is higher than both Ethereum and Bitcoin. While there are other blockchains with much higher transaction rates, these are typically multi-transactional blockchains designed for monetary purposes. Our system, however, is developed for a voting system, which does not require multi-transaction capabilities.

In the context of voting, the number of voters is not constant, which can introduce complexity in multi-transactional blockchains. Our system employs a mono-transactional blockchain, meaning a block will immediately process a transaction upon generation. This design choice simplifies the system and aligns well with the requirements of a voting system.

In summary, our system demonstrates superior performance with sharding, achieving higher throughput and faster block generation times compared to systems without sharding. Additionally, our throughput outperforms that of Ethereum and Bitcoin, making it well suited for a voting system.

6.6. Analysis Based on Memory Consumption

The bar graph in Figure 18 shows how block modularity and sharding affect storage consumption in the blockchain network.

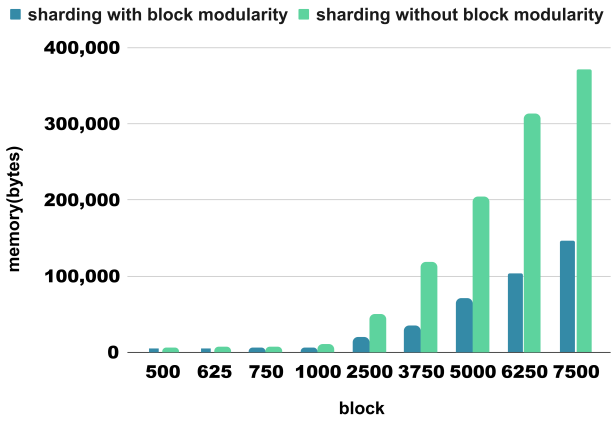

Figure 18. The bar graph shows the storage consumption with and without sharding for different numbers of blocks. The x-axis represents the number of blocks, and the y-axis represents the total storage consumption in bytes. Sharding significantly reduces storage consumption in blockchain networks, especially for large numbers of blocks.

Without sharding, each block requires 26 bytes of storage, and as the number of blocks increases to 7500, the total storage consumption significantly rises to 371,072 bytes. However, when sharding is implemented, the storage requirement for each block decreases to 16 bytes, and the overall storage consumption reduces to 146,082.4 bytes even with 7500 blocks. This approach results in reduced data replication, faster and more efficient data processing and validation, and decreased computational requirements. These findings demonstrate the significant storage efficiency achieved through sharding, which has important implications for scalability and resource management in blockchain systems. The insights presented in this analysis are important for researchers and practitioners designing and implementing blockchain architectures. Sharding is a powerful technique that can be used to improve the efficiency and scalability of blockchain networks.

7. Discussion and Analysis

In this section, we discuss the attack analysis and security analysis.

7.1. Attack Analysis

Hybrid blockchains are considered to be more secure than public blockchains, as they are only validated by their respective layers. However, the block is as accessible and transparent as a public blockchain. Even though they are not immune to every attack, some of the most common attack vectors on private blockchains include the following:

- *Vote Tampering:* One of the most significant concerns in electronic voting systems is vote tampering, where an attacker modifies the votes to favor a particular candidate. Our system uses a blockchain-based secure ledger architecture, in which each vote is stored as a ledger in a block, and once a block is added to the chain, it cannot be modified. This ensures that the votes are immutable and cannot be tampered with.
- *Double Voting:* Double voting occurs when a voter casts more than one vote in the same election. Our system uses a multiparty computed cryptographically secure token verification system to prevent double voting. Each voter is issued with a unique token that is used to cast their vote, and the system checks whether the token has already been used before allowing the vote to be cast.
- *Denial of Service (DoS) Attacks:* DoS attacks are designed to make the voting system unavailable to users by overwhelming it with traffic or disrupting its infrastructure. Our system uses sharding to distribute the load across multiple nodes, thereby making it more resilient to DoS attacks. In addition, the system uses a proof of Hierarchical Access and Control (HAC) mechanism, which ensures that only authorized nodes can add blocks to the chain, preventing attackers from flooding the network with invalid blocks.
- *Sybil Attacks:* Sybil attacks involve creating multiple fake identities to gain an unfair advantage in the voting process. Our system uses biometric and facial recognition to verify the identity of each voter and employs an HAC authorization tree to create valid roles and identities. This makes it difficult for attackers to create fake identities and to ensure the integrity of the voting process.
- *Man-in-the-Middle (MitM) Attacks:* MitM attacks involve intercepting and modifying communication between two parties. Our system uses secure cryptographic mechanisms such as end-to-end encrypted communication protocols, public-key cryptography, and digital signatures to ensure that all communication between nodes is secure and cannot be intercepted or modified by unauthorized parties. This ensures the confidentiality, integrity, and authenticity of all data transmitted in the system.
- *Quantum Computing Attacks:* With the advent of quantum computing, there is a risk that traditional cryptographic mechanisms could be broken, making the voting system vulnerable to attacks. Our system uses post-quantum-secured cryptographic algorithms that are designed to be resistant to attacks by quantum computers, ensuring that the system remains secure even in the face of advances in quantum computing.

7.2. Security Analysis

Hybrid blockchains are considered more secure than public blockchains, as only authorized personnel validate them. They offer accessibility and transparency comparable to private blockchains, yet are not entirely immune to attacks. The proposed PQMPCHAC-Bchain model incorporates various security measures to prevent the attacks mentioned in Section 7.1, including the following:

- *Confidentiality:* Our system utilizes encrypted, authenticated communication channels between peers. Each peer is uniquely identified by a Peer ID derived from a private cryptographic key. Zero-Knowledge Multiparty Computation (zk-MPC) enables computations on encrypted data without revealing individual inputs, ensuring voter information remains confidential.
- *Integrity:* We guarantee vote integrity and authenticity through an Immutable Distributed Ledger Technology (IDLT) built with post-quantum secure Dilithium-based

digital signatures. A multiparty cryptographically secure token verification system prevents double voting while protecting voter identity and privacy.

- *Availability:* A category-based sharding mechanism distributes workload across multiple nodes, enhancing scalability and resilience against Denial of Service (DoS) attacks, thus ensuring high availability and reliability.
- *Authentication:* Voter identity is authenticated using MPC token generation and authentication mechanisms, along with biometric and facial recognition technologies. A Hierarchical Access Control (HAC) authorization tree creates valid roles and identities, mitigating Sybil attack risks.
- *Non-repudiation:* Append-only digital signatures, MPC tokens, and the HAC authorization tree ensure non-repudiation in the voting process, preventing voters from denying or disputing their votes.
- *Post-Quantum Security:* Post-quantum cryptographic algorithms, such as lattice-based Dilithium, are employed for authentication, MPC operations, and authorization tasks. This approach ensures robust security even in the face of quantum computing advancements.
- *Randomness:* Cryptographically secure randomness facilitates the generation of zero-knowledge proofs during computation and validation phases, ensuring reliable and verifiable proofs without compromising sensitive information.
- *Malicious Node Isolation:* The system verifies data sources using the HAC-tree and isolates nodes that fail to comply with HAC or provide incorrect proofs or data. Node actions are shared within the network, and data are replicated across the closest peers, effectively isolating malicious nodes. This approach secures the system against Sybil and eclipse attacks.
- *Threshold Cryptography:* To mitigate single points of failure, threshold cryptography distributes control over certificate authorization, script creation, and other critical operations among multiple entities.
- *Secure Communication Protocol:* End-to-end encrypted communication protocols like TLS 1.3 and Noise protect against eavesdropping and Man-in-the-Middle (MitM) attacks, ensuring secure data transmission between nodes.
- *Secure KMS Server:* A decentralized Key Management Server (KMS) provides relevant public keys and certificates for verifying votes, tokens, and blocks, preventing fraud and ensuring the authenticity of all transactions in the system.

In summary, our proposed system employs a multi-layered security approach that includes blockchain-based immutability, unique voter identification, sharding, proof-of-stake consensus, end-to-end encryption, and post-quantum cryptography to protect against various potential attacks and ensure the integrity and security of the voting process.

7.3. Requirement Analysis

The proposed blockchain-based e-voting system is designed to provide both accuracy and security, meeting the following requirements:

- *Voter eligibility:* Only verified voters are allowed to cast their votes.
- *Verifiability:* Voters can verify that their votes have been correctly counted and included in the final tally.
- *Robustness:* The voting results and associated data are stored on the blockchain, making it highly resistant to tampering.
- *Uniqueness:* The voting process is verified twice, once on the blockchain and once on the server, to ensure that each vote is unique.
- *Ballot receipt:* Upon submitting their vote, voters receive a block ID as confirmation that their vote has been successfully recorded on the blockchain.
- *Transparency:* The blockchain's transparent nature allows anyone on the network to view all voting procedures, ensuring that the process is fair and open.
- *Trustworthiness:* The blockchain's scripting provides various security features to the e-voting system, such as security, autonomy, and transparency, making it resistant to manipulation and errors.

- *Scalability:* The proposed PQMPCHAC-Bchain, combined with sharding, block modularity, and post-quantum technologies, provides high levels of security and throughput, making the blockchain-based e-voting system scalable.

As we can see in comparison Table 3, our proposed blockchain-based e-voting system is a secure and scalable solution that can meet the requirements of modern elections.

Table 3. Comparison of the proposed e-voting system based on a blockchain and previous related work.

	Reference								
Properties	[9]	[10]	[33]	[18]	[5]	[40]	[41]	[21]	*Our Proposed*
Security	✓	✓	✓	✓	✓	✓	✓	✓	✓
Robustness	✓	✗	✗	✓	✗	✗	✗	✓	✓
Consensus	PSC	POS	QBA	POS	POS	POS	POS	POW	HAC
Sharding	✓	✗	✗	✗	✗	✗	✗	✗	✓
Eligibility 5	✓	✗	✓	✓	✗	✗	✗	✗	✓
Verifiability	✓	✓	✓	✓	✗	✓	✓	✓	✓
Uniqueness	✓	✓	✓	✓	✗	✓	✓	✓	✓
Ballot receipt	✓	✗	✓	✓	✗	✓	✓	✓	✓
Transparency	✓	✓	✓	✓	✓	✓	✓	✓	✓
Embed Trust	✓	✓	✓	✓	✗	✓	✓	✓	✓
Scalability	✓	✗	✗	✓	✓	✓	✓	✗	✓
Post Quantum	✗	✗	✓	✗	✗	✗	✗	✗	✓
Deepface	✓	✓	✗	✗	✗	✗	✗	✗	✓
Time-based inference	✗	✗	✗	✗	✗	✗	✗	✗	✓
Unlinkability	✗	✗	✗	✓	✗	✗	✓	✗	✓
Confidentiality	✓	✗	✗	✓	✗	✗	✓	✓	✓

8. Conclusions

E-voting can address the shortcomings of traditional voting systems, but current solutions, including Biometric EVM, face challenges like reduced voter trust, cyber threats, and scalability issues. To overcome these, we present PQMPCHAC-Bchain, an e-voting system that uses the HAC model and combines public and private blockchains for transparency, immutability, and efficiency. The proposed system represents a groundbreaking hybrid-blockchain-based e-voting solution, prioritizing post-quantum safety and scalability. It offers a robust blockchain programming environment, user-friendly scripting mechanisms, and adherence to relevant regulatory frameworks. To protect voter identities while checking for double voting, we propose a multiparty computed token generation and verification mechanism. Our system employs biometric and facial recognition technologies for voter authentication and a secure cryptographic signature process for validating voter information. To evaluate the system's efficacy, we conducted tests using high-quality, synthetically generated voter records and election data, demonstrating its scalability, security, and effectiveness. Our system achieved 181 TPS and is compatible with existing electoral laws and regulations, allowing seamless integration into the current e-voting framework. For the test simulation, we set up entities such as the Election Commission,

returning officer, and polling officer. However, certain limitations must be addressed in future research. The system's reliance on a predetermined independent party for token computations introduces a potential centralization of power. To counteract this, we intend to explore decentralized and open MPC participation systems. Additionally, we aim to incorporate secure post-quantum communication protocols and smart adaptive sharding technology to enhance the security, scalability, and overall performance of our system.

Author Contributions: Conceptualization, S.A.J., R.R. and P.G.; methodology, S.A.J., R.R., N.T. and P.G.; software, S.A.J., R.R., N.T. and P.G.; validation, P.G., M.A.U. and J.A.; formal analysis, S.A.J. and M.A.U.; investigation, N.T., P.G. and M.A.U.; resources, S.A.J., R.R., N.T., P.G. and M.A.U.; writing—original draft preparation, S.A.J., R.R., N.T. and P.G.; writing—review and editing, P.G., M.A.U. and J.A.; visualization, S.A.J., R.R., N.T. and P.G.; supervision, P.G., M.A.U. and J.A.; project administration, P.G. and S.A.J. All authors have read and agreed to the published version of the manuscript.

Funding: his research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset used to test performance and scalability has been uploaded to the Hugging Face dataset repository under the name "Bangladesh-Voter-Synthetic-Dataset (Revision 958f5b4)", available at the following DOI: https://doi.org/10.57967/hf/2932.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Appendix A.1. Dilithium-2

We conducted an analysis to compare the performance of Dilithium-2 in a blockchain network with and without block modularity, focusing on efficiency and transaction capacity. We represented the results using a line graph shown in Figure A1 to show the overall time and a bar graph shown in Figure A2 to represent the throughput. Also, our comparison data is provided in Table A2.

In the line graph, the horizontal axis represents the number of voters and the vertical axis represents the time. For throughput, the horizontal axis represents the time and the vertical axis represents the number of voters.

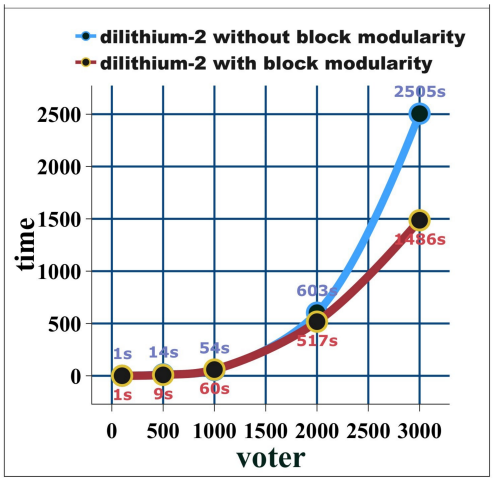

Figure A1. Performance of Dilithium-2 with block modularity (dark red line) vs. without block modularity (sky-blue line). The horizontal axis represents the number of voters, and the vertical axis represents the time. Block modularity improves the performance of Dilithium-2, especially as the number of voters increases.

In Figure A1, the sky-blue line represents Dilithium-2 without block modularity and the dark red line represents Dilithium-2 with block modularity. In terms of throughput shown in Figure A2, the green line represents Dilithium-2 without block modularity and the sky-blue line represents Dilithium-2 with block modularity.

For Dilithium-2 with and without block modularity, the sky-blue line and red line remain the same in Figure A1 for generating a single block for a single voter. However, as the number of voters increases, the time taken to generate blocks also increases. Interestingly, the red line remains lower than the sky-blue line, which means that Dilithium-2 with block modularity takes less time to generate blocks than Dilithium-2 without block modularity when the number of voters increases.

As a result, we can say that Dilithium-2 with block modularity is faster than Dilithium-2 without block modularity. This is because it can perform computations in parallel, which reduces the overall time required. It is also more efficient in terms of memory usage, which further reduces the time required.

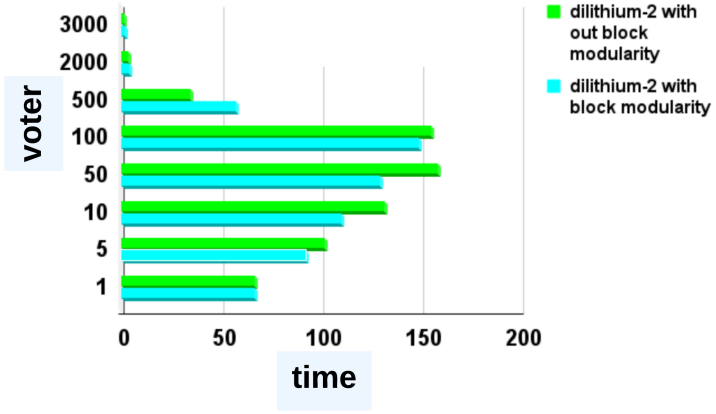

Figure A2. Throughput of Dilithium-2 with block modularity vs. without block modularity. The horizontal axis represents the number of voters, and the vertical axis represents the throughput in transaction per second (tps). Block modularity improves throughput, especially with the high number of voters.

In terms of throughput shown in Figure A2, Dilithium-2 with block modularity also shows overall better performance than Dilithium-2 without block modularity. Although Dilithium-2 without block modularity performs slightly better before 100 voters, Dilithium-2 with block modularity performs well at peak, giving approximately the same throughput of 1 tps.

In conclusion, Dilithium-2 with block modularity overall performs better than Dilithium-2 without block modularity.

Appendix A.2. Dilithium-3

We also analyzed to compare the performance of Dilithium-3 in a blockchain network with and without block modularity, focusing on efficiency and transaction capacity. We represented the results using a line graph shown in Figure A3 to show the overall time and a bar graph shown in Figure A4 to represent the throughput. Additionally, comparison data is shown in Table A3.

In the line graph, the horizontal axis represents the number of voters and the vertical axis represents the time. For throughput, the horizontal axis represents the time and the vertical axis represents the number of voters.

In Figure A3, the violet line represents Dilithium-3 with block modularity and the maroon line represents Dilithium-3 without block modularity. In Figure A4, the sky blue line represents Dilithium-3 with block modularity and the green line represents Dilithium-3 without block modularity.

For Dilithium-3 with and without block modularity, the dilithium-3 without block modularity (violet line) and dilithium-3 with block modularity (chocolate colored line) remain the same in Figure A3 for generating a single block. However, as the number of voters increases, the time taken to generate blocks also increases. Interestingly, the maroon line remains lower than the violet line, which means that Dilithium-3 with block modularity takes less time to generate blocks than Dilithium-3 without block modularity when the number of voters increases.

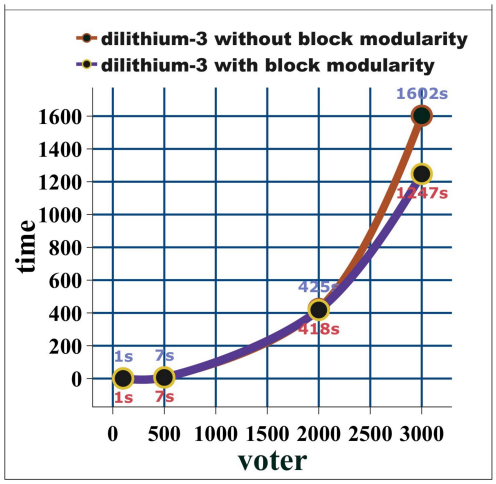

Figure A3. Performance of Dilithium-3 with block modularity (violet line) vs. without block modularity (maroon line). The horizontal axis represents the number of voters, and the vertical axis represents the time. Block modularity improves performance, especially with the high number of voters, because it can perform computations in parallel and is more efficient in terms of memory usage.

As a result, we can say that Dilithium-3 with block modularity is faster than Dilithium-3 without block modularity. This is because it can perform computations in parallel, which reduces the overall time required. It is also more efficient in terms of memory usage, which further reduces the time required. As a result, Dilithium-3 with block modularity is typically much faster and more efficient than Dilithium-3 without block modularity.

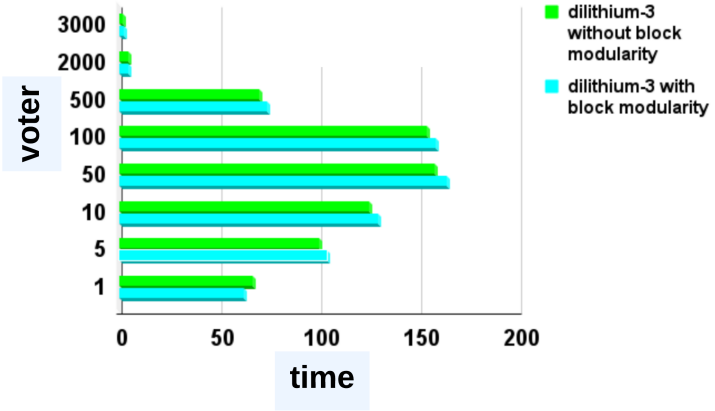

Figure A4. The bar graph shows the throughput of Dilithium-3 with and without block modularity in a blockchain network. The vertical axis represents the number of voters, and the horizontal axis represents time. The bar graph shows that Dilithium-3 with block modularity (sky-blue line) has a higher throughput than Dilithium-3 without block modularity (green line), especially when the number of voters is high. Throughput is measured in blocks per second (bps). This is because block modularity allows Dilithium-3 to process more transactions per second.

In terms of throughput shown in Figure A4, Dilithium-3 with block modularity also shows overall better performance than Dilithium-3 without block modularity. Although

Dilithium-3 performs slightly better before 200 voters, Dilithium-3 with block modularity performs better at peak, giving approximately the same throughput of 2 bps.

In conclusion, Dilithium-3 with block modularity overall performs better than Dilithium-3 without block modularity.

Appendix A.3. Block Modularity vs. Dilithium-2 with Block Modularity

Here, we compare block modularity and Dilithium-2 with block modularity.

The line graph Figure A5 shows the time it takes to generate a block, with the number of voters on the horizontal axis and the time in seconds on the vertical axis. The blue line represents block modularity with Dilithium-2, and the green line represents block modularity alone.

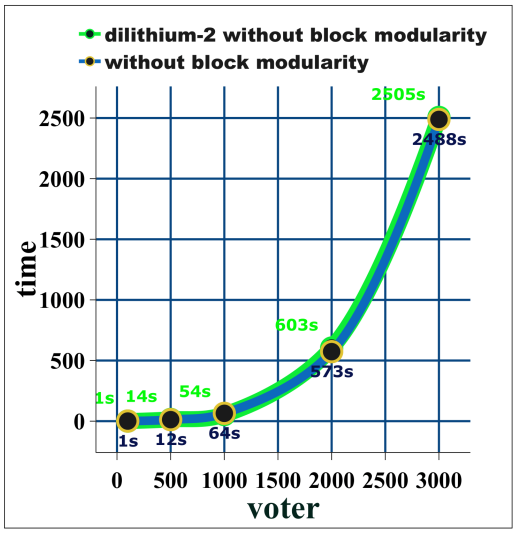

Figure A5. Comparison block generation time of block modularity and Dilithium-2 with block modularity. The blue line represents block modularity with Dilithium-2, and the green line represents block modularity alone. The number of voters is on the horizontal axis, and the time in seconds to generate a block is on the vertical axis. In conclusion, block modularity shows better performance than Dilithium-2 with block modularity.

Initially, the block generation time is almost the same for both configurations. However, after 1000 voters, the block generation time for Dilithium-2 with block modularity is slightly higher than the block generation time for block modularity alone.

This is because Dilithium-2 is a more secure cryptographic algorithm than the general elliptic curve algorithm. As a result, it takes slightly longer to generate blocks using Dilithium-2.

Overall, block modularity shows better performance than Dilithium-2 with block modularity.This is because block modularity systems separate all data by category. When new data arrives, it does not duplicate the existing data. Instead, it only updates the new data. This results in slightly faster block generation times and higher throughput.

In addition, block modularity can reduce bandwidth usage by up to 70 percent, storage usage by up to 60 percent, and data usage by up to 40 percent.

In terms of throughput shown in Figure A6, the green line represents block modularity and the sky-blue line represents Dilithium-2 with block modularity. The bar graph shows that Dilithium-2 with block modularity initially has better throughput than block modularity. However, at their peak, they both have the same throughput.

In conclusion, merging Dilithium-2 with block modularity provides extra security at the cost of slightly slower block generation times and throughput.

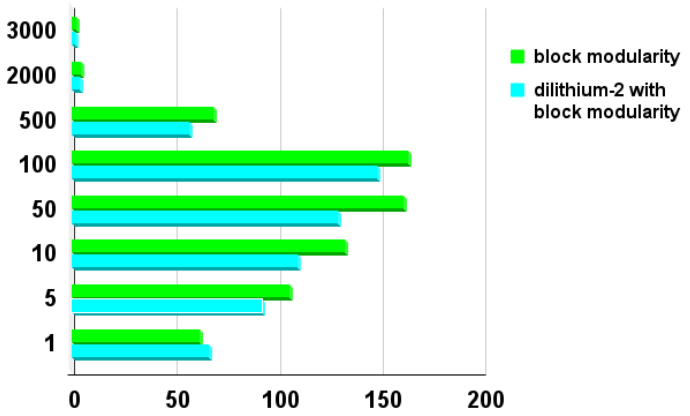

Figure A6. Block Modularity vs. Dilithium-2 with Block Modularity Throughput. This bar graph compares the throughput of block modularity and Dilithium-2 with block modularity. The x-axis shows the number of voters, and the y-axis shows the time. The blue bars represent block modularity with Dilithium-2, and the green bars represent block modularity alone. Block modularity initially has better throughput than Dilithium-2 with block modularity. However, at their peak, they both have the same throughput.

Appendix A.4. Without Block Modularity vs. Dilithium-2 without Block Modularity

We compare Dilithium-2 without block modularity and without block modularity.

The line graph Figure A7 shows the time it takes to generate a block, with the number of voters on the horizontal axis and the time in seconds on the vertical axis. The green line represents Dilithium-2 without block modularity, and the blue line represents without block modularity.

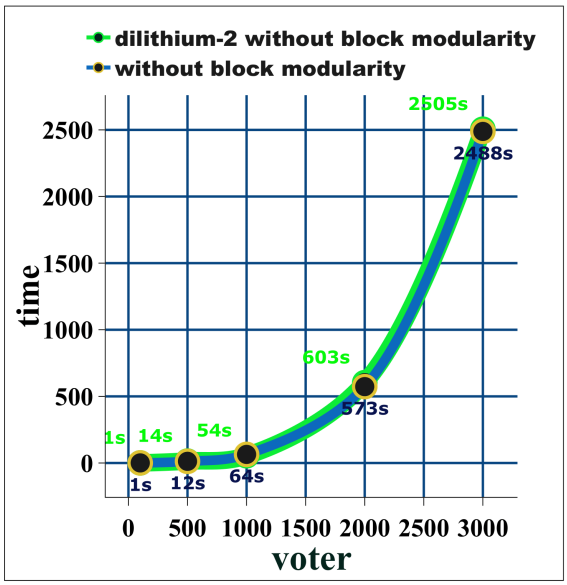

Figure A7. Comparison block generation time of Dilithium-2 without block modularity and without block modularity. The green line represents Dilithium-2 without block modularity, and the blue line represents without block modularity. The number of voters is on the horizontal axis, and the time in seconds to generate a block is on the vertical axis. The block generation time is almost the same for both configurations.

The line graph shows that the block generation time is almost the same for both configurations. This is because both configurations do not use block modularity. However, using Dilithium-2 without block modularity took slightly longer.

In terms of throughput shown in Figure A8, the sky-blue line represents Dilithium-2 without block modularity and the green line represents without block modularity. The bar graph shows that Dilithium-2 without block modularity initially has better throughput than without block modularity. However, at their peak, they both have the same throughput.

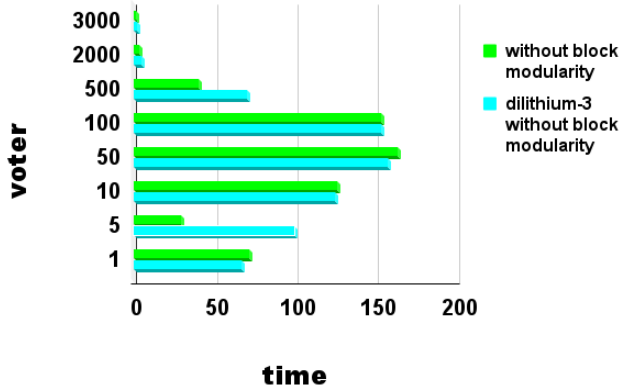

Figure A8. Comparison of the throughput of Dilithium-2 without block modularity and without block modularity. The sky-blue line represents Dilithium-2 without block modularity, and the green line represents without block modularity. The number of voters is on the horizontal axis and the number of blocks generated per second is on the vertical axis. Dilithium-2 without block modularity initially has better throughput than without block modularity. However, at their peak, they both have the same throughput.

In conclusion, merging Dilithium-2 without block modularity provides extra security with almost no impact on block generation and throughput.

Appendix A.5. Block Modularity vs. Dilithium-3 with Block Modularity

Block modularity and Dilithium-3 with block modularity are two blockchain designs that offer different trade-offs between performance and security.

Block modularity is a blockchain design that separates all data into categories. When new data arrives, it does not duplicate the existing data. Instead, it only updates the new data. This results in slightly faster block generation times and higher throughput. In addition, block modularity can reduce bandwidth usage by up to 70 percent, storage usage by up to 60 percent, and data usage by up to 40 percent.

Dilithium-3 with block modularity is a blockchain design that uses the Dilithium-3 cryptographic algorithm to improve the security of block modularity. Dilithium-3 is considered to be one of the most secure cryptographic algorithms available today. However, it is also more computationally expensive than other cryptographic algorithms, which can lead to slower block generation times.

The line graph in Figure A9 shows the time it takes to generate a block, with the number of voters on the horizontal axis and the time in seconds on the vertical axis. The blue line represents block modularity with Dilithium-3, and the green line represents block modularity alone.

Initially, the block generation time is almost the same for both configurations. However, after a certain number of voters, the block generation time for Dilithium-3 with block modularity is slightly lower than the block generation time for block modularity alone.

Dilithium-3 with block modularity offers better security than block modularity alone. This is because Dilithium-3 is a more secure cryptographic algorithm. Dilithium-3 is considered to be one of the most secure cryptographic algorithms available today and can protect against quantum computers.

In terms of throughput, the bar graph in Figure A10 compares the throughput of block modularity and Dilithium-3 with block modularity. The x-axis shows the number of voters, and the y-axis shows the time. The blue bars represent block modularity with Dilithium-3, and the green bars represent block modularity alone.

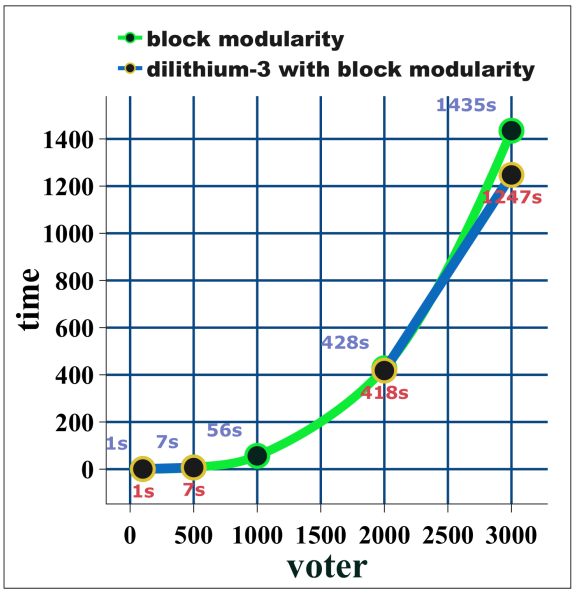

Figure A9. Comparison of the block generation time of block modularity and Dilithium-3 with block modularity. The blue line represents block modularity with Dilithium-3, and the green line represents block modularity alone. The number of voters is on the horizontal axis, and the time in seconds to generate a block is on the vertical axis. Dilithium-3 with block modularity offers faster block generation times than block modularity alone.

Block modularity initially has better throughput than Dilithium-3 with block modularity. However, at their peak, they both have the same throughput.

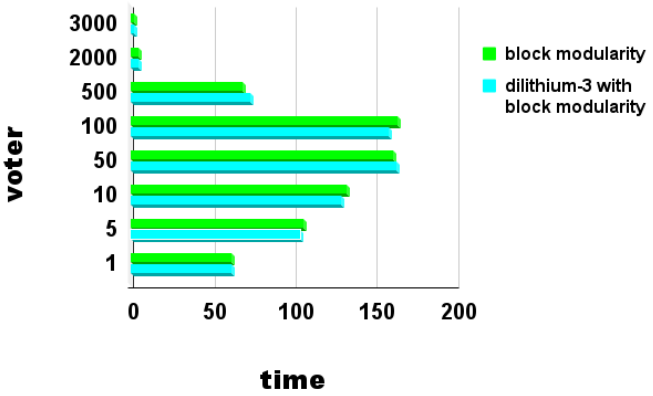

Figure A10. Comparison of the throughput of block modularity and Dilithium-3 with block modularity. The blue bars represent block modularity with Dilithium-3, and the green bars represent block modularity alone. The number of voters is on the horizontal axis and time is on the vertical axis. Block modularity initially has better throughput than Dilithium-3 with block modularity. However, at their peak, they both have the same throughput.

In conclusion, block modularity and Dilithium-3 with block modularity are two blockchain designs that offer different trade-offs between performance and security. Block modularity offers slightly faster block generation times and higher throughput, while Dilithium-3 with block modularity offers better security. The best choice for a particular blockchain application will depend on the specific requirements of that application.

Appendix A.6. Without Block Modularity vs. Dilithium-3 without Block Modularity

We compare Dilithium-3 without block modularity and without block modularity.

The line graph in Figure A11 shows the time it takes to generate a block, with the number of voters on the horizontal axis and the time in seconds on the vertical axis. The

green line represents Dilithium-3 without block modularity, and the blue line represents without block modularity.

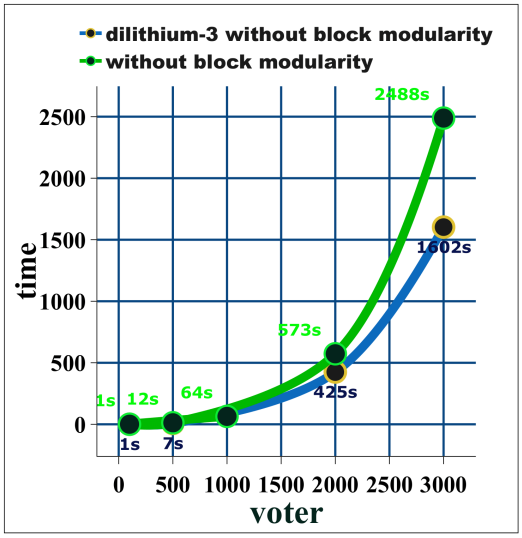

Figure A11. Block Generation Time Comparison of Dilithium-3 without Block Modularity and Without Block Modularity. The green line represents Dilithium-3 without block modularity, and the blue line represents without block modularity. The number of voters is on the horizontal axis and the time in seconds is on the vertical axis. Dilithium-3 without block modularity took less time to generate a block.

The line graph shows that the initial block generation time is almost the same for both configurations. However, after 1000 voters, Dilithium-3 without block modularity took a lower time than without block modularity. This is strange because Dilithium-3 is a more computationally expensive cryptographic algorithm than the one used without block modularity.

In terms of throughput, the bar graph in Figure A12 shows that without block modularity initially has better throughput than Dilithium-3 without block modularity. However, at their peak, they both have the same throughput.

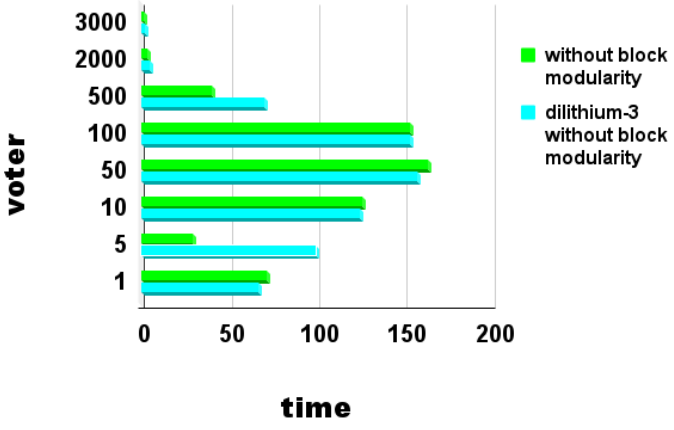

Figure A12. Throughput Comparison of Dilithium-3 without Block Modularity and Without Block Modularity. The blue-sky bars represent without block modularity with Dilithium-3, and the green bars represent without block modularity alone. The number of voters is on the horizontal axis and the time is on the vertical axis. The bar graph shows that without block modularity initially has better throughput than Dilithium-3 without block modularity. However, at their peak, they both have the same throughput.

In conclusion, merging Dilithium-3 without block modularity provides extra security and slightly faster block generation time. However, it also has slightly lower throughput than without block modularity at lower voter counts.

Appendix A.7. Dilithium-2 vs. Dilithium-3

In comparison to Dilithium-2 without block modularity, Dilithium-3 without block modularity demonstrates improved performance. The comparison data is shown in Tables A4 and A5.

The line graph in Figure A13 shows that Dilithium-3 without block modularity (olive line) achieves faster transaction processing than Dilithium-2 without block modularity (violet line). The time required to generate 3000 blocks is approximately 1602.457 s in Dilithium-3, whereas Dilithium-2 takes around 2505.086 s.

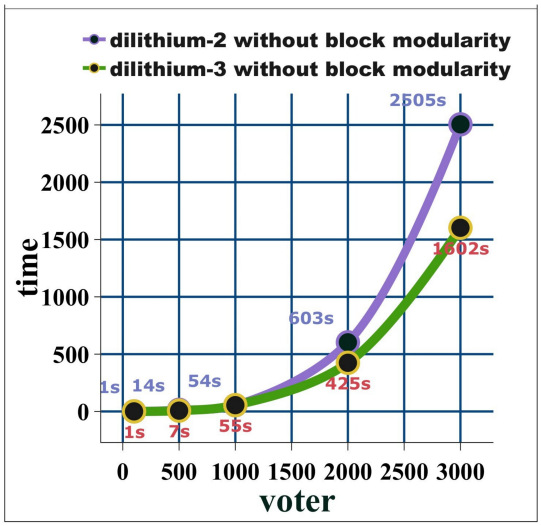

Figure A13. Performance Comparison of Dilithium-3 without block modularity and Dilithium-2 without block modularity. The olive-colored line represents Dilithium-3 without block modularity, and the violet line represents Dilithium-2 without block modularity. The horizontal axis represents the number of blocks generated, and the vertical axis represents the time in seconds. In conclusion, Dilithium-3 without block modularity gave better performance than Dilithium-2 without block modularity.

Regarding throughput, the bar graph Figure A14 shows that Dilithium-3 without block modularity (sky-blue line) performs slightly better than Dilithium-2 without block modularity (green line). The throughput for the first voter is the same for both, at 66 bps. However, Dilithium-3 without block modularity achieves a slightly better throughput after the 100th voter, peaking at 1.8 bps, while Dilithium-2 without block modularity peaks at 1.19 bps.

In summary, Dilithium-3 without block modularity achieves faster transaction processing and higher security compared to Dilithium-2 without block modularity.

These findings emphasize the advantages of Dilithium-3, indicating its superior efficiency and higher security in the blockchain network. It is faster, more efficient, has higher throughput, and is more secure. It is also future-proofed against quantum computers.

Dilithium-3 with block modularity demonstrates improved performance over Dilithium-2 with block modularity.

The line graph Figure A15 shows that Dilithium-3 with block modularity (green line) achieves faster transaction processing than Dilithium-2 with block modularity (sky-blue line). Dilithium-3 takes approximately 1247 s to generate 3000 blocks, while Dilithium-2 takes around 1486 s.

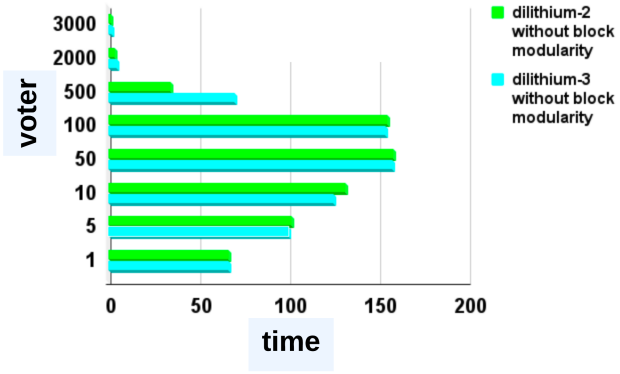

Figure A14. Comparing the performance of Dilithium-3 and Dilithium-2 without block modularity in terms of throughput. In the bar graph, the horizontal axis represents the time, and the vertical axis represents the voter. Additionally, the sky-blue line represents Dilithium-3 without block modularity and the green line represents Dilithium-2 without block modularity. In conclusion, Dilithium-3 without block modularity performs slightly better than Dilithium-2 without block modularity.

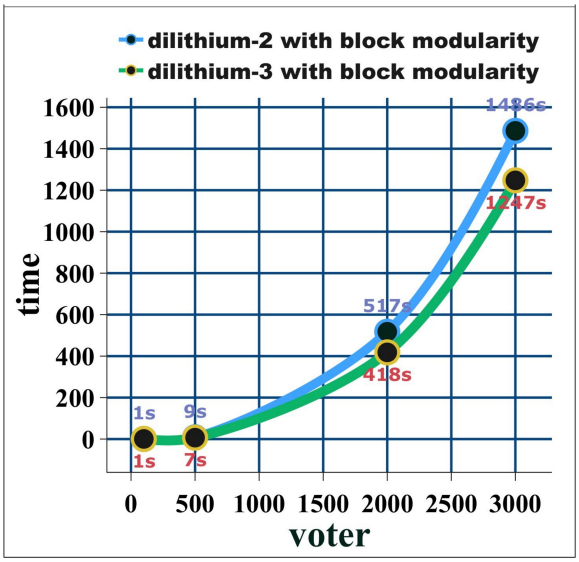

Figure A15. Performance comparison of Dilithium-3 with block modularity and Dilithium-2 with block modularity. The green line represents Dilithium-3 with block modularity, and the sky-blue line represents Dilithium-2 with block modularity. The horizontal axis represents the number of blocks generated, and the vertical axis represents the time in seconds. Here, Dilithium-3 with block modularity is faster than Dilithium-2 with block modularity.

Regarding throughput, the bar graph Figure A16 shows that Dilithium-3 with block modularity (green line) performs slightly better than Dilithium-2 with block modularity (sky-blue line). The throughput for the first voter is the same for both, at 66 bits per second (bps). However, Dilithium-3 with block modularity achieves a slightly better throughput after the 100th voter, peaking at 2.4 bps, while Dilithium-2 with block modularity peaks at 2 bps.

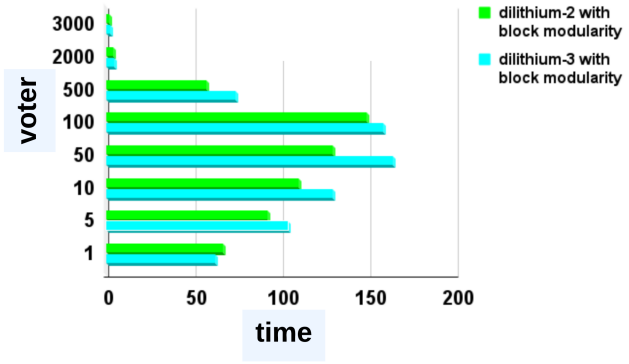

Figure A16. Comparing the throughput of Dilithium-3 with block modularity and Dilithium-2 with block modularity over time. The graph shows the voter on the vertical axis and time on the horizontal axis. The green line represents Dilithium-3 with block modularity, and the sky-blue line represents Dilithium-2 with block modularity. Dilithium-3 with block modularity performs better than Dilithium-2 with block modularity in terms of throughput.

In summary, Dilithium-3 with block modularity is a more secure, efficient, and easier-to-implement signature scheme than Dilithium-2 with block modularity. It has security proof against quantum attacks up to NIST security level 5, while Dilithium-2 has security proof up to NIST security level 3. It is also more efficient in terms of both public key size and signature size, and it is more resilient to errors. Finally, it is easier to implement than Dilithium-2.

In short, Dilithium-3 with block modularity is the better choice for applications that require strong security against quantum attacks.

These findings emphasize the advantages of Dilithium-3, indicating its superior efficiency and higher security in blockchain networks. It is faster, more efficient, has higher throughput, and is more secure. It is also future-proofed against quantum computers.

Appendix A.8. Sharding vs. Dilithium-2

Dilithium-2 with Block Modularity:

Dilithium-2 with block modularity emphasizes cryptographic security. It is designed to protect data from attacks and ensure data integrity. The focus is on optimizing the efficiency of signature generation and verification processes. Its security strength is measured by its public key length, which offers a level of security up to NIST security level 3. The architecture is simple, making it easier to implement and manage. However, its scalability may be limited, which can hinder performance in scenarios with rapid growth and increased workloads.

Sharding with Two Shards:

Sharding with two shards emphasizes performance and scalability. By splitting data into two shards, it allows for parallel processing and faster data access. The goal is to improve system performance, especially in scenarios where numerous transactions need to be processed simultaneously. While sharding with two shards offers performance benefits, it may require more complex management due to challenges with data distribution and load balancing. Additionally, the approach does not inherently address cryptographic security concerns.

The line graph shown in Figure A17 illustrates the time it takes to generate a block for different numbers of voters. The orange line represents Dilithium-2 with block modularity, and the sky-blue line represents sharding with two shards. As observed, sharding with two shards consistently outperforms Dilithium-2 with block modularity, especially for large numbers of voters. The comparison data is shown in Tables A7 and A8.

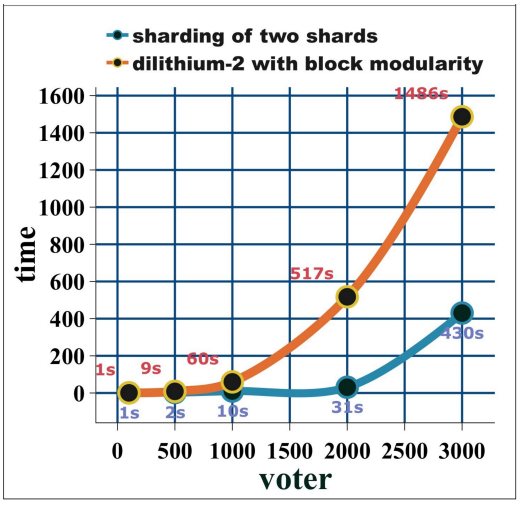

Figure A17. Time to generate a block for different numbers of voters using Dilithium-2 with block modularity (orange line) and sharding with two shards (sky-blue line). The horizontal axis represents the number of voters, and the vertical axis represents the time in milliseconds. Sharding with two shards consistently outperforms Dilithium-2 with block modularity, especially for large numbers of voters.

The bar graph shown in Figure A18 compares the throughput of the two approaches for different numbers of voters. The green line represents sharding, and the sky-blue line represents Dilithium-2 with block modularity. Again, sharding outperforms Dilithium-2 with block modularity, especially for large numbers of voters.

In conclusion, the graphs (Figure A17) and (Figure A18) show that sharding with two shards provides better performance than Dilithium-2 with block modularity for both block generation and throughput. This is because sharding allows for parallel processing and faster data access. While sharding with two shards may require more complex management, it is a better choice for blockchain networks that need to scale to large numbers of users and transactions.

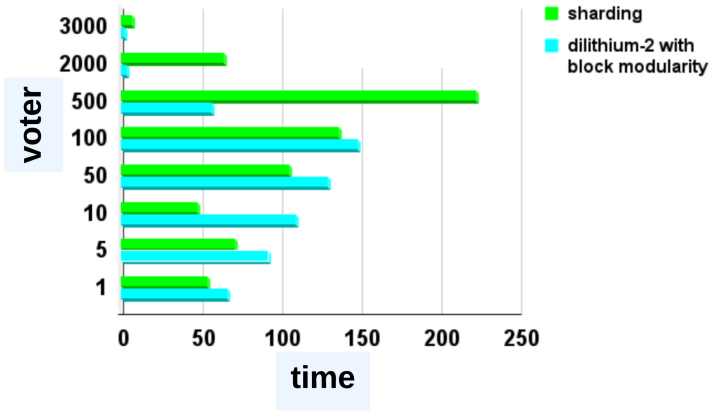

Figure A18. Throughput comparison of sharding with two shards and Dilithium-2 with block modularity for different numbers of voters. The vertical axis represents the number of voters, and the horizontal axis represents the time in seconds. Sharding consistently outperforms Dilithium-2 with block modularity, especially for large numbers of voters. This suggests that sharding is a better choice for blockchain networks that need to scale to large numbers of users and transactions.

Dilithium-2 without Block Modularity:

Dilithium-2 without block modularity is a cryptographic signature scheme that emphasizes performance. It is designed to be fast and efficient while still providing a high

level of security. Its security level is up to NIST security level 3. The architecture is simple, making it easier to implement and manage. However, it may not be as secure as other signature schemes that use block modularity.

Sharding with Two Shards:

Sharding with two shards emphasizes scalability. By splitting data into two shards, it allows for parallel processing and faster data access. The goal is to improve system performance, especially in scenarios where many transactions need to be processed simultaneously. While sharding with two shards offers performance benefits, it may require more complex management due to challenges with data distribution and load balancing.

The line graph shown in Figure A19 illustrates the time it takes to generate a block for different numbers of voters. The red line represents Dilithium-2 without block modularity, and the green line represents sharding with two shards. Initially, both approaches take the same amount of time, but after 1000 voters, sharding with two shards consistently outperforms Dilithium-2 without block modularity, especially for large numbers of voters. The comparison data is shown in Table A8.

The bar graph shown in Figure A20 illustrates the throughput of the two approaches for different numbers of voters. The green line represents sharding, and the sky-blue line represents Dilithium-2 without block modularity. Again, sharding outperforms Dilithium-2 without block modularity, especially for large numbers of voters.

In conclusion, the graphs (Figure A19) and (Figure A20) show that sharding with two shards provides better performance than Dilithium-2 without block modularity for both block generation and throughput. This is because sharding allows for parallel processing and faster data access. While sharding with two shards may require more complex management, it is a better choice for blockchain networks that need to scale to large numbers of users and transactions.

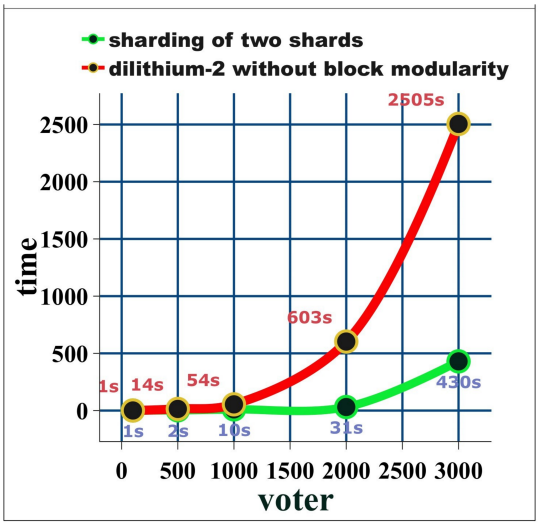

Figure A19. Time to generate a block for different numbers of voters, comparing Dilithium-2 without block modularity and sharding with two shards. The x-axis represents the number of voters, and the y-axis represents the time in seconds. The red line represents Dilithium-2 without block modularity, and the green line represents sharding with two shards. Sharding with two shards consistently outperforms Dilithium-2 without block modularity.

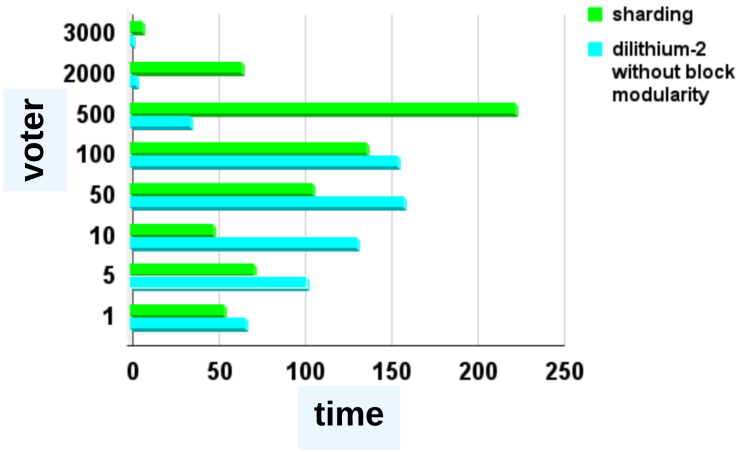

Figure A20. Throughput of Dilithium-2 without block modularity and sharding with two shards. The bar graph shows the throughput of Dilithium-2 without block modularity and sharding with two shards. The horizontal axis represents time, and the vertical axis represents time. These results demonstrate the significant throughput improvements achieved through sharding.

Appendix A.9. Sharding vs. Dilithium-3

Dilithium-3 with Block Modularity:

Dilithium-3 with block modularity is a cryptographic signature scheme that emphasizes security. It is designed to protect data from attacks and ensure data integrity. The focus is on optimizing the efficiency of signature generation and verification processes. Its security strength is measured by its public key length, which offers a level of security up to NIST security level 4. The architecture is simple, making it easier to implement and manage. However, its scalability may be limited, which can hinder performance in scenarios with rapid growth and increased workloads.

Sharding with Two Shards:

Sharding with two shards emphasizes performance and scalability. By splitting data into two shards, it allows for parallel processing and faster data access. The goal is to improve system performance, especially in scenarios where numerous transactions need to be processed simultaneously. While sharding with two shards offers performance benefits, it may require more complex management due to challenges with data distribution and load balancing. Additionally, the approach does not inherently address cryptographic security concerns.

The line graph shown in Figure A21 illustrates the time it takes to generate a block for different numbers of voters. The violet line represents Dilithium-3 with block modularity, and the green line represents sharding with two shards. Initially, both the violet and green lines stayed almost the same, but when the number of voters increased, the violet line went higher than the green line. This means the green line (sharding) performed better than the violet line (Dilithium-3 with block modularity). The comparison data is shown in Tables A9 and A10.

The bar graph shown in Figure A22 illustrates the throughput of the two approaches for different numbers of voters. The green line represents sharding, and the sky-blue line represents Dilithium-3 with block modularity. Again, sharding outperforms Dilithium-3 with block modularity, especially for large numbers of voters.

In conclusion, the graphs (Figure A21) and (Figure A22) show that sharding with two shards provides better performance than Dilithium-3 with block modularity for both block generation and throughput. This is because sharding allows for parallel processing and faster data access. While sharding with two shards may require more complex management, it is a better choice for blockchain networks that need to scale to large numbers of users and transactions.

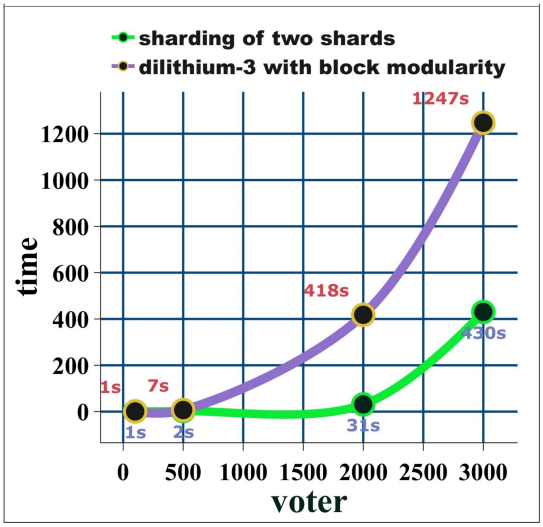

Figure A21. Block generation time for Dilithium-3 with block modularity and sharding with two shards. The horizontal axis represents the number of voters, and the vertical axis represents the time in seconds. Here, sharding with two shards gave better performance than Dilithium-3 with block modularity.

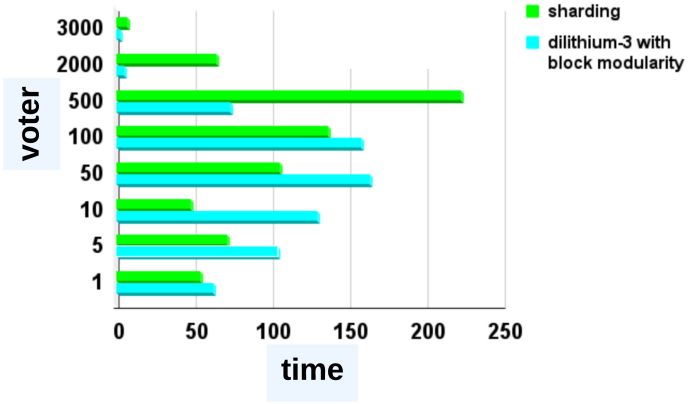

Figure A22. Throughput of Dilithium-3 with block modularity and sharding with two shards. The horizontal axis represents the time, the vertical axis represents voters, and the green line represents sharding, and the sky-blue line represents Dilithium-3 with block modularity. Here, sharding with two shards gave better performance than Dilithium-3 with block modularity.

Dilithium-3 without Block Modularity:

Dilithium-3 without block modularity is a cryptographic signature scheme that emphasizes performance. It is designed to be fast and efficient while still providing a high level of security. Its security level is up to NIST security level 4. The architecture is simple, making it easier to implement and manage. However, it may not be as secure as other signature schemes that use block modularity.

Sharding with Two Shards:

Sharding with two shards emphasizes scalability. By splitting data into two shards, it allows for parallel processing and faster data access. The goal is to improve system performance, especially in scenarios where numerous transactions need to be processed simultaneously. While sharding with two shards offers performance benefits, it may require more complex management due to challenges with data distribution and load balancing.

The line graph shown in Figure A23 illustrates the time it takes to generate a block for different numbers of voters. The blue line represents Dilithium-3 without block modularity, and the green line represents sharding with two shards. The blue line and green line stayed almost the same, but when the number of voters increased, the blue line (Dilithium-3

without block modularity) took more time than the green line (sharding). This makes the conclusion that sharding performs better than Dilithium-3 without block modularity.

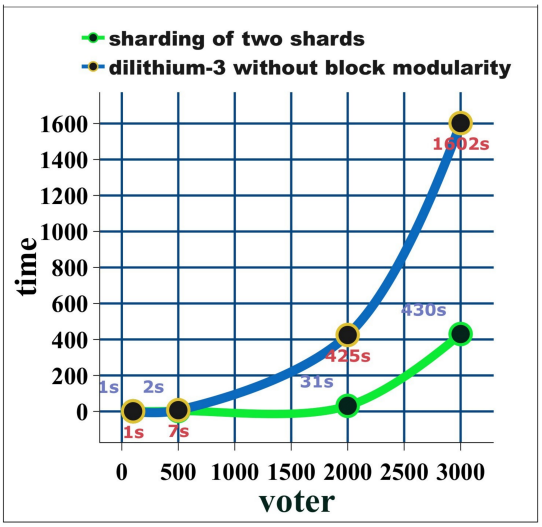

Figure A23. Block generation time for Dilithium-3 without block modularity and sharding. The horizontal axis represents the time, and the vertical axis represents the number of voters. As the number of voters increases, sharding with two shards (green line) is able to maintain a consistent block generation time. In contrast, Dilithium-3 without block modularity (blue line) experiences significant performance degradation. This comparison highlights the efficiency of sharding in maintaining consistent performance under increasing voter loads.

The bar graph shown in Figure A24 illustrates the throughput of the two approaches for different numbers of voters. The green line represents sharding, and the sky-blue line represents Dilithium-3 without block modularity. Again, sharding outperforms Dilithium-3 without block modularity, especially for large numbers of voters.

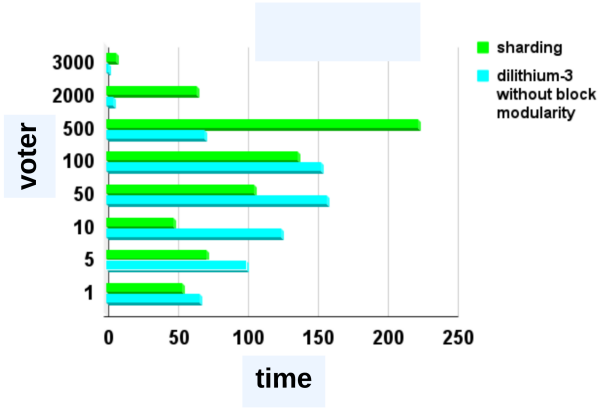

Figure A24. Throughput of Dilithium-3 without block modularity and sharding with two shards. The bar graph shows the throughput of the two approaches for different numbers of voters. The horizontal axis represents the number of voters, and the vertical axis represents the throughput in transactions per second. Here, sharding with two shards gave better performance than Dilithium-3 without block modularity.

In conclusion, the graphs (Figure A23) and (Figure A24) show that sharding with two shards provides better performance than Dilithium-3 without block modularity for both block generation and throughput. This is because sharding allows for parallel processing and faster data access. While sharding with two shards may require more complex management, it is a better choice for blockchain networks that need to scale to large numbers of users and transactions. The comparison data is shown in Table A10.

Appendix B. Diagram

Appendix B.1. Block Header

High level Unit					
PK	**Block**				
Unit 1		Unit 2		Unit 3	
Field	**Value**	**Field**	**Value**	**Field**	**Value**
abc...	123...	abc...	123...	abc...	123...
abc...	123...	abc...	123...	abc...	123...
abc...	123...	abc...	123...	abc...	123...
Hash	12312031...	Hash	12312031...	Hash	12312031...

Figure A25. This is a visual representation of a block and its units. The block header stores the metadata of the block, and the block body stores the data of the block. The block is divided into different units, each of which has a hash. The block hash is made by combining the hashes of all the units of the block.

Appendix B.2. Merkel Tree

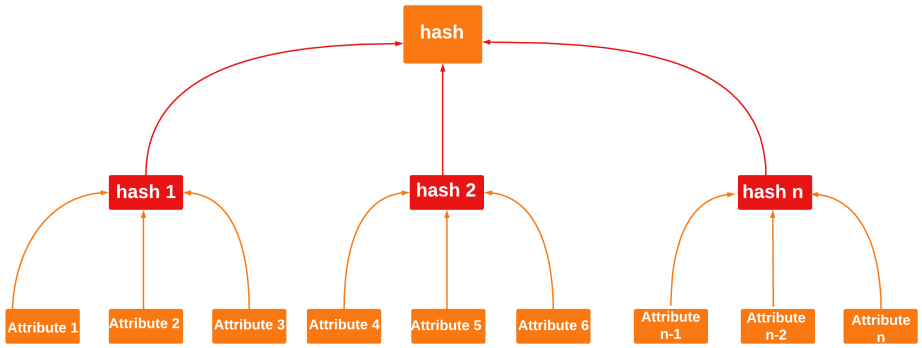

Figure A26. The visual representation of the Merkle tree shows how it is constructed from the bottom up. The leaves of the tree are hashes of the data blocks, and the non-leaf nodes are hashes of their child nodes. The root of the tree is called the Merkle root. The block record is made up of the Merkle root and other metadata, and the Merkle root is stored in the block header.

Appendix B.3. Authorization

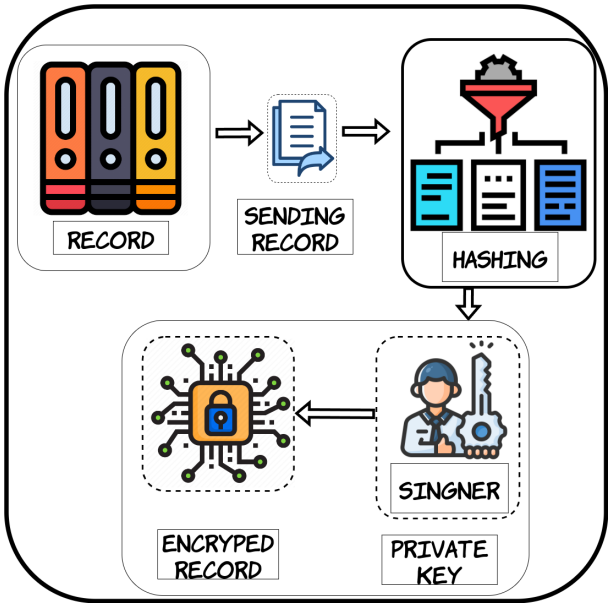

Figure A27. This diagram presents a lucid explanation of the authorization process. It shows how every entity is hashed, signed, and generates proof of record.

Appendix B.4. Biometric Verification

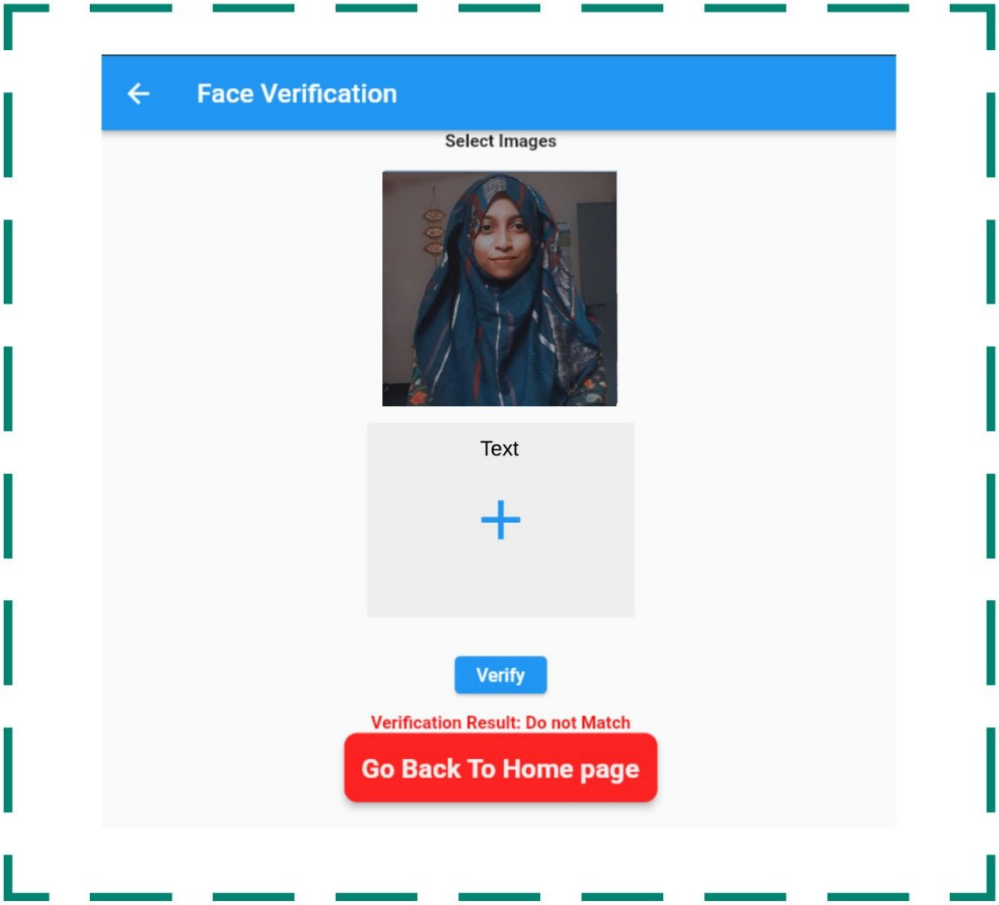

Figure A28. The representation of a user capturing an image in front of a system for verification prior to accessing their respective systems is a common security measure employed to prevent unauthorized

access. The user is typically required to maintain a steady pose while the system captures a high-resolution image. The captured image is then compared to a stored image of the user's face to validate their identity.

Appendix B.5. Verification Process

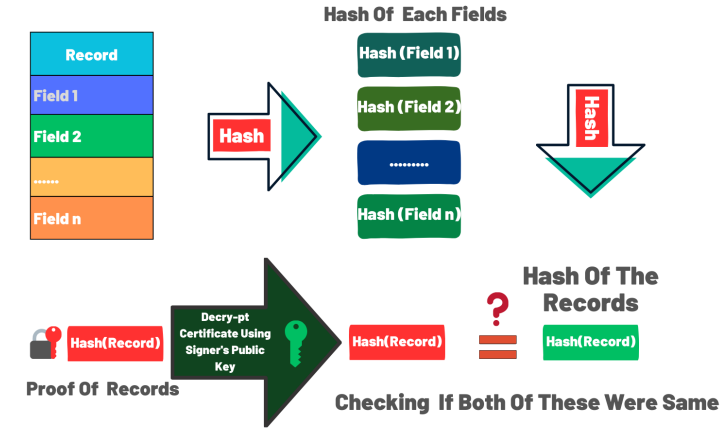

Figure A29. This diagram provides a clear explanation of the verification process. The intended record is hashed, signed, and generates into a proof. This newly generated proof is then compared to the existing proof to ensure validity.

Appendix B.6. Time-Based Inference

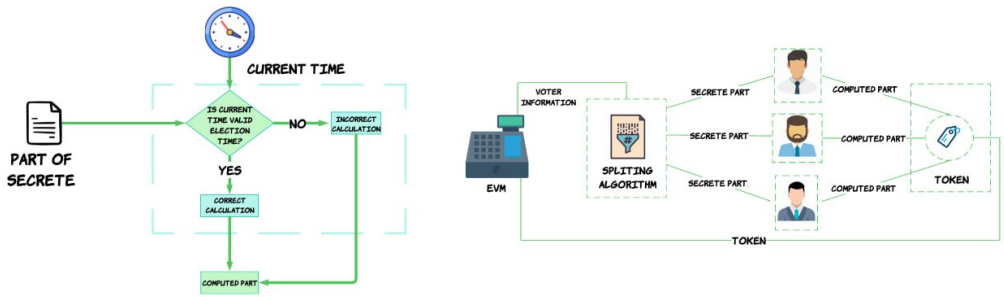

Figure A30. The MPC token generation process is a secure and efficient way to generate tokens that can be used to verify that a voter is eligible to vote and to prevent voter fraud. The process works by taking the voter's unique identifier as input, splitting it into multiple parts, encrypting each part, and then having each encrypted part computed by a different party. The computed parts are then merged to form a token.

Appendix C. Algorithm

Appendix C.1. Token-Based Voter Verification System Working Procedure

We describe the working procedure of the token-based voter verification system below.

- A cryptographically secure string is generated from the voter's unique identifier.
- The string is then encrypted and hashed into a secret.
- The secret string is split into multiple pieces and sent to different parties for further computation.
- Each party has its own random function, seed, and encryption key, which are updated before and after the election.
- After each party receives its piece of the secret string, they compute their part of the token.

- Once all the parts of the token have been computed, they are merged together to form a single token.
- The token is then checked to see if it has already been spent.
- If the token is not spent, the token is then used to cast a vote.
- The token is then stored as a spent token in the blockchain after the voter casts their vote.

Appendix C.2. Algorithmic Representation of Token Generation

Algorithm A1 Formulae of Token Generation

```
Generate a token
secret ← Voter's unique identifier
seed ← Voter's seed
Slice the secret into n parts
function SliceFunction(secret, n)
    parts ← empty list
    partSize ← length of secret/n
    for i ← 0 to n do
        parts[i] ← secret[i * partSize : (i + 1) * partSize]
    end for
    return parts
end function
Compute the parts of the secret in each party
function MultyPartyComputation(parts, n)
    parts ← SliceFunction(secret, n)
    shares ← empty list
    for i ← 0 to n do
        if Time() is in the range of ValidElectionTime then
            shares[i] ← shares[i] ⊕ seed
        else
            shares[i] ← shares[i] ⊕ 0
        end if
        shares[i] ← generateRandomNumber()
        shares[i] ← shares[i] ⊕ parts[i]
        shares[i] ← Hash(shares[i])
        shares[i] ← Sign(shares[i], partiesprivatekey)
    end for
    return shares
end function
Merge the shares of secret into a token
function MergeShares(shares, n)
    shares ← MultyPartyComputation(parts, n)
    token ← empty string
    for i ← 0 to n do
        if Verify(shares[i].signature, partiespublicKey) is Valid then
            token ← token ⊕ shares[i]
        else
            return invalid
        end if
    end for
    return token
end function
Sign the token
function SignToken(token, privatekey)
    token ← MergeShares(shares, n)
    token ← Hash(token)
    token ← Sign(token, privatekey)
    return token
end function
Return token
```

Appendix C.3. Algorithmic Representation of Tallying Votes

Algorithm A2 Tallying Votes

```
generate a token
secret ← Voter unique identifier
seed ← Voter's seed
Slice the secret into n parts
function TALLYINGVOTES(blockchain, lengthOfchain, CanidatelistVotes)
    votes ← empty map
    votes ← getVotes
    for each block in blockchain do
        votes ← getVotes(block)
    end for
    for i ← 0 to len(votes) do
        for j ← 0 to len(CanidatelistVotes) do
            if votes[i] is equal to CanidatelistVotes[j] then
                votes[i] ← Canidatelist[j]
            end if
        end for
    end for
    result ← countVotes(votes)
    return result
end function
get votes from block
function GETVOTES(block)
    votes ← empty list
    votes ← getVotesFromBlock(block)
    votes[i] ← encrypt(votes[i], Ballotkey)
    L0 ← Hash(votes[i], seed)
    votes[i] ← encrypt(L0,keyp)
    L1 ← Hash(votes[i],Rp)
    votes[i] ← encrypt(L1,keyr)
    L2 ← Hash(votes[i],Rr)
    votes[i] ← encrypt(L2,keye)
    n ← Hash(votes[i],Re)
    votes[i] ← encrypt(n,keye)
    return votes
end function
get votes from block
function GETVOTESFROMBLOCK(block)
    votes ← block.vote
    return votes
end function
count votes
function COUNTVOTES(votes)
    result ← empty map
    for i ← 0 to len(votes) do
        result[votes[i]] ← result[votes[i]] + 1
    end for
    return result
end function
Return result
```

Appendix C.4. Algorithmic Representation of Authorization Mechanism

Algorithm A3 Proposed Authorization Algorithm

```
function GETAUTHORIZE (userdata, authorizer)
    if authorizer = "ElectionCommission" then
        signature = GetAuthorizeFromEC(userdata)
        if signature ≠ nil then
            return signature
        else
            return nil
        end if
    else if authorizer = "StateElectionCommission" then
        signature = GetAuthorizeFromSEC(userdata)
        if signature ≠ nil then
            return signature
        else
            return nil
        end if
    else if authorizer = "DistrictElectionCommission" then
        signature = GetAuthorizeFromDEC(userdata)
        if signature ≠ nil then
            return signature
        else
            return nil
        end if
    else if authorizer = "ReturningOfficer" then
        signature = GetAuthorizeFromRO(userdata)
        if signature ≠ nil then
            return signature
        else
            return nil
        end if
    else if authorizer = "PresidingOfficer" then
        signature = GetAuthorizeFromPO(userdata)
        if signature ≠ nil then
            return signature
        else
            return nil
        end if
    else
        return nil
    end if
    return nil
end function
```

Appendix C.5. Algorithmic Representation of Peer Routing

1. Get the latest block using `getLatestBlock()` function and store it in `bit`.
2. Get the configuration using `getConfig()` function and store it in `cfg`.
3. Create a node host using `makeNodeHost(cfg)` function and store it in `host`.
4. Create a new Kademlia DHT using `dht.New(ctx, host)` function and store it in `kademliaDHT`.
5. Bootstrap the Kademlia DHT using `kademliaDHT.Bootstrap(ctx)`.
6. For each peer address in `config.BootstrapPeers`:
    a. Get the peer using `peerChan()` function and hold until peer is found.

b. If there is an error while connecting to the peer using `host.Connect(ctx, peer)`, print "Connection failed: <error message>" and continue to the next peer address.
c. If the block data is successfully sent to the peer using `host.SendBlockData(bit, peer)`, print "Connected to: <peer>" and add the peer's addresses to the peerstore using `host.Peerstore().AddAddrs(peer.ID, peer.Addrs, peerstore.PermanentAddrTTL)`.
d. If the block data is not successfully sent to the peer, print "Connection failed".

Appendix D. Raw Data

Appendix D.1. Block Modularity vs. without Block Modularity

Table A1. Comparison of with block modularity and without block modularity.

With Block Modularity			Without Block Modularity		
Voter	**Time (s)**	**Throughput**	**Voter**	**Time (s)**	**Throughput**
1	0.016	62.5	1	0.014	71.42857143
5	0.047	106.3829787	5	0.17	29.41176471
10	0.075	133.3333333	10	0.079	126.5822785
50	0.309	161.8122977	50	0.305	163.9344262
100	0.609	164.2036125	100	0.651	153.609831
500	7.22	69.25207756	500	12.476	40.07694774
1000	56.288	17.76577601	1000	64.455	15.51470018
2000	427.722	4.675934369	2000	573.184	3.489280929
3000	1434.873	2.090777372	3000	2488.459	1.205565372

Appendix D.2. Dilithium-2 with Block Modularity vs. Dilithium-2 without Block Modularity

Table A2. Comparison of Block Generation Time and Throughput without Block Modularity Combined with Dilithium-2 and with Block Modularity Combined with Dilithium-2.

Dilithium-2 without Block Modularity			Dilithium-2 with Block Modularity		
Voter	**Time (s)**	**Throughput**	**Voter**	**Time (s)**	**Throughput**
1	0.015	66.66666667	1	0.015	66.66666667
5	0.049	102.0408163	5	0.054	92.59259259
10	0.076	131.5789474	10	0.091	109.8901099
50	0.316	158.2278481	50	0.385	129.8701299
100	0.644	155.2795031	100	0.672	148.8095238
500	14.4	34.72222222	500	8.713	57.3855159
1000	54.308	18.41349341	1000	59.61	16.77570877
2000	603.193	3.315688345	2000	517.474	3.864928479
3000	2505.086	1.197563676	3000	1486.277	2.018466275

Appendix D.3. Dilithium-3 with Block Modularity vs. Dilithium-3 without Block Modularity

Table A3. Comparison of Block Generation Time and Throughput without Block Modularity Combined with Dilithium-3 and with Block Modularity Combined with Dilithium-3.

Dilithium-3 without Block Modularity			Dilithium-3 with Block Modularity		
Voter	**Time (s)**	**Throughput**	**Voter**	**Time (s)**	**Throughput**
1	0.015	66.66666667	1	0.016	62.5
5	0.05	100	5	0.048	104.1666667
10	0.08	125	10	0.077	129.8701299
50	0.317	157.7287066	50	0.305	163.9344262
100	0.65	153.8461538	100	0.63	158.7301587
500	7.121	70.21485746	500	6.772	73.83343178
2000	424.67	4.709539172	2000	417.999	4.784700442
3000	1602.457	1.872125118	3000	1247.433	2.404938782

Appendix D.4. Dilithium-2 vs. Dilithium-3 without Block Modularity

Table A4. Comparison of Block Generation Time and Throughput without Block Modularity Combined with Dilithium-2 and Dilithium-3.

Dilithium-2 and without Block Modularity			Dilithium-3 and without Block Modularity		
Voter	**Time (s)**	**Throughput**	**Voter**	**Time (s)**	**Throughput**
1	0.015	66.66666667	1	0.015	66.66666667
5	0.049	102.0408163	5	0.05	100
10	0.076	131.5789474	10	0.08	125
50	0.316	158.2278481	50	0.317	157.7287066
100	0.644	155.2795031	100	0.65	153.8461538
500	14.4	34.72222222	500	7.121	70.21485746
1000	54.308	18.41349341	1000	55.033	18.17091563
2000	603.193	3.315688345	2000	424.67	4.709539172
3000	2505.086	1.197563676	3000	1602.457	1.872125118

Appendix D.5. Dilithium-2 vs. Dilithium-3 with Block Modularity

Table A5. Comparison of Block Generation Time and Throughput without Block Modularity Combined with Dilithium-2 and with Block Modularity Combined with Dilithium-3.

Dilithium-2 and with Block Modularity			Dilithium-3 and with Block Modularity		
Voter	**Time (s)**	**Throughput**	**Voter**	**Time (s)**	**Throughput**
1	0.015	66.66666667	1	0.016	62.5
5	0.054	92.59259259	5	0.048	104.1666667
10	0.091	109.8901099	10	0.077	129.8701299
50	0.385	129.8701299	50	0.305	163.9344262
100	0.672	148.8095238	100	0.63	158.7301587
500	8.713	57.3855159	500	6.772	73.83343178

Table A5. *Cont.*

Dilithium-2 and with Block Modularity			Dilithium-3 and with Block Modularity		
Voter	**Time (s)**	**Throughput**	**Voter**	**Time (s)**	**Throughput**
2000	517.474	3.864928479	2000	417.999	4.784700442
3000	1486.277	2.018466275	3000	1247.433	2.404938782

Appendix D.6. Comparison between Sharding

Table A6. Comparison between two, three, and five shards in sharding.

Sharding with Two Shards			Sharding with Three Shards			Sharding with Five Shards		
Block	**Time (s)**	**Throughput**	**Block**	**Time (s)**	**Throughput**	**Block**	**Time (s)**	**Throughput**
2	0.0185	108.1081081	2	0.011	181.8181818	2	0.015	133.3333333
5	0.347	14.4092219	5	0.0184	271.7391304	5	0.0224	223.2142857
10	0.0695	143.8848921	10	0.0368	271.7391304	10	0.0643	155.5209953
50	0.211	236.9668246	50	0.354	141.2429379	50	0.2166	230.8402585
100	0.474	210.9704641	100	0.522	191.5708812	100	0.3848	259.8752599
400	2.2425	178.3723523	400	1.344	297.6190476	400	1.345	297.3977695
500	3.426	145.9427904	500	2.543	196.6181675	500	1.6264	307.4274471
1000	10.356	96.5623793	1000	7.844	127.4859765	1000	3.437	290.9514111
2000	82.2415	24.31862259	2000	35.897	55.71496225	2000	9.434	211.999152
3000	430.055	6.975851926	3000	152.885	19.62259214	3000	28.323	105.9209829

Appendix D.7. Without Block Modularity Dilithium-2 vs. Sharding

Table A7. Comparison of without block modularity combined with Dilithium-2 and sharding.

Sharding			Dilithium-2 and without Block Modularity		
Voter	**Time (s)**	**Throughput**	**Voter**	**Time (s)**	**Throughput**
1	0.0185	54.05405405	1	0.015	66.66666667
5	0.0695	71.94244604	5	0.049	102.0408163
10	0.2075	48.19277108	10	0.076	131.5789474
50	0.474	105.4852321	50	0.316	158.2278481
100	0.7305	136.8925394	100	0.644	155.2795031
500	2.2425	222.9654404	500	14.4	34.72222222
1000	10.356	96.5623793	1000	54.308	18.41349341
2000	30.9185	64.68619112	2000	603.193	3.315688345
3000	430.055	6.975851926	3000	2505.086	1.197563676

Appendix D.8. With Block Modularity Dilithium-2 vs. Sharding

Table A8. Comparison of with block modularity combined with Dilithium-2 and sharding.

Sharding			Dilithium-2 and with Block Modularity		
Voter	**Time (s)**	**Throughput**	**Voter**	**Time (s)**	**Throughput**
1	0.0185	54.05405405	1	0.015	66.66666667
5	0.0695	71.94244604	5	0.054	92.59259259
10	0.2075	48.19277108	10	0.091	109.8901099
50	0.474	105.4852321	50	0.385	129.8701299
100	0.7305	136.8925394	100	0.672	148.8095238
500	2.2425	222.9654404	500	8.713	57.3855159
1000	10.356	96.5623793	1000	59.61	16.77570877
2000	30.9185	64.68619112	2000	517.474	3.864928479
3000	430.055	6.975851926	3000	1486.277	2.018466275

Appendix D.9. Without Block Modularity: Dilithium-3 vs. Sharding

Table A9. Comparison of without block modularity combined with Dilithium-3 and sharding.

Sharding			Dilithium-3 and without Block Modularity		
Voter	**Time (s)**	**Throughput**	**Voter**	**Time (s)**	**Throughput**
1	0.0185	54.05405405	1	0.015	66.66666667
5	0.0695	71.94244604	5	0.05	100
10	0.2075	48.19277108	10	0.08	125
50	0.474	105.4852321	50	0.317	157.7287066
100	0.7305	136.8925394	100	0.65	153.8461538
500	2.2425	222.9654404	500	7.121	70.21485746
2000	30.9185	64.68619112	2000	424.67	4.709539172
3000	430.055	6.975851926	3000	1602.457	1.872125118

Appendix D.10. Without Block Modularity: Dilithium-3 vs. Sharding

Table A10. Comparison of with block modularity combined with Dilithium-3 and sharding.

Sharding			Dilithium-3 and with Block Modularity		
Voter	**Time (s)**	**Throughput**	**Voter**	**Time (s)**	**Throughput**
1	0.0185	54.05405405	1	0.016	62.5
5	0.0695	71.94244604	5	0.048	104.1666667
10	0.2075	48.19277108	10	0.077	129.8701299
50	0.474	105.4852321	50	0.305	163.9344262
100	0.7305	136.8925394	100	0.63	158.7301587
500	2.2425	222.9654404	500	6.772	73.83343178
2000	30.9185	64.68619112	2000	417.999	4.784700442
3000	430.055	6.975851926	3000	1247.433	2.404938782

Appendix D.11. Storage Consumetion

Table A11. Storage consumption of without block modularity with sharding and with block modularity with sharding.

Without Block Modularity and Sharding		Block Modularity Sharding	
Block	**Memory (byte)**	**Block**	**Memory (byte)**
1	26	1	16
5	52	5	80
25	76	25	97.6
50	96.8	50	100.8
100	181.6	100	134.4
250	533.6	250	248
375	964.8	375	404.8
500	1453.6	500	582.4
625	2190.4	625	775.2
750	3016.8	750	1152.8
1000	6319.2	1000	1816
2500	45,272	2500	15,762.4
3750	113,824.8	3750	30,360
5000	200,077.6	5000	66,399.2
6250	308,527.2	6250	103,209.6
7500	371,072	7500	146,082.4

References

1. USAID. *Supporting Free and Fair Elections*; United States Agency for International Development (USAID): Washington, DC, USA. Available online: https://www.usaid.gov/democracy/supporting-free-and-fair-elections (accessed on 13 November 2023).
2. Nasrullah, T.M.; Islam, M.M.; Uddin, M.A.; Khan, M.A.; Layek, M.A.; Stranieri, A.; Huh, E.N. Device Agent Assisted Blockchain Leveraged Framework for Internet of Things. *IEEE Access* **2022**, *11*, 1254–1268. [CrossRef]
3. Sallal, M.; de Fréin, R.; Malik, A. PVPBC: Privacy and Verifiability Preserving E-Voting Based on Permissioned Blockchain. *Future Internet* **2023**, *15*, 121. [CrossRef]
4. Du, Z.; Li, Y.; Fu, Y.; Zheng, X. Blockchain-based access control architecture for multi-domain environments. *Pervasive Mob. Comput.* **2024**, *98*, 101878. [CrossRef]
5. Anitha, V.; Caro, O.J.M.; Sudharsan, R.; Yoganandan, S.; Vimal, M. Transparent voting system using blockchain. *Meas. Sens.* **2023**, *25*, 100620. [CrossRef]
6. Bajpai, M.; Haider, A.; Mishra, A.; Perwej, Y.; Rastogi, N. A novel vote counting system based on secure blockchain. *Int. J. Sci. Res. Sci. Eng. Technol* **2022**, *9*, 69–79. [CrossRef]
7. Stančíková, I.; Homoliak, I. SBvote: Scalable Self-Tallying Blockchain-Based Voting. In Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, Tallinn, Estonia, 27–31 March 2023; pp. 203–211.
8. Kohad, H.; Kumar, S.; Ambhaikar, A. Scalability of Blockchain based E-voting system using Multiobjective Genetic Algorithm with Sharding. In Proceedings of the 2022 IEEE Delhi Section Conference (DELCON), New Delhi, India, 11–13 February 2022; pp. 1–4.
9. 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 J.* **2021**, *43*, 357–370. [CrossRef]
10. 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* **2023**, *3*, 1–17. [CrossRef]
11. Vaidya, C.; Kirnapure, C.; Rithe, J.; Sonkusare, D.; Khade, P.; Kharche, K. An Approach Towards Decentralized E-Voting. In Proceedings of the IEEE 2023 11th International Conference on Emerging Trends in Engineering & Technology-Signal and Information Processing (ICETET-SIP), Nagpur, India, 28–29 April 2023; pp. 1–6.
12. Curran, K. E-Voting on the Blockchain. *J. Br. Blockchain Assoc.* **2018**, *1*, 1–6. [CrossRef]

13. Peralta, R.; Brandão, L.T.A.N. *NIST First Call for Multi-Party Threshold Schemes*; National Institute of Standards and Technology, Gaithersburg, MD, USA, January 2023. [CrossRef]
14. 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; IACR: Bochum, Germany, 2018; pp. 238–268.
15. Ghose, P.; Sharmin, S.; Gaur, L.; Zhao, Z. Grid-search integrated optimized support vector machine model for breast cancer detection. In Proceedings of the 2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Las Vegas, NV, USA, 6–8 December 2022; pp. 2846–2852.
16. 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. *Intell. Syst. Appl.* **2022**, *16*, 200130. [CrossRef]
17. Das, S.K.; Saha, S.; DasGupta, S. Decentralized Voting: A Blockchain-Based Voting System. In *Proceedings of the Applications of Networks, Sensors and Autonomous Systems Analytics: Proceedings of ICANSAA 2020*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 33–45.
18. Khoury, D.; Kfoury, E.F.; Kassem, A.; Harb, H. Decentralized voting platform based on ethereum blockchain. In Proceedings of the 2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET), Beirut, Lebanon, 14–16 November 2018; pp. 1–6.
19. Bing, W.; Hui-ling, L.; Li, P. Optimized DPoS consensus strategy: Credit-weighted comprehensive election. *Ain Shams Eng. J.* **2023**, *14*, 101874. [CrossRef]
20. Hassan, H.S.; Hassan, R.; Gbashi, E.K. E-voting System Based on Ethereum Blockchain Technology Using Ganache and Remix Environments. *Eng. Technol. J.* **2023**, *41*, 1–16. [CrossRef]
21. Bhadoria, R.S.; Das, A.P.; Bashar, A.; Zikria, M. Implementing Blockchain-Based Traceable Certificates as Sustainable Technology in Democratic Elections. *Electronics* **2022**, *11*, 3359. [CrossRef]
22. Li, K.; Li, H.; Wang, H.; An, H.; Lu, P.; Yi, P.; Zhu, F. PoV: An efficient voting-based consensus algorithm for consortium blockchains. *Front. Blockchain* **2020**, *3*, 11. [CrossRef]
23. Sun, Y.; Yan, B.; Yao, Y.; Yu, J. DT-DPoS: A delegated proof of stake consensus algorithm with dynamic trust. *Procedia Comput. Sci.* **2021**, *187*, 371–376. [CrossRef]
24. Liu, Y.; Liu, J.; Salles, M.A.V.; Zhang, Z.; Li, T.; Hu, B.; Henglein, F.; Lu, R. Building blocks of sharding blockchain systems: Concepts, approaches, and open problems. *Comput. Sci. Rev.* **2022**, *46*, 100513. [CrossRef]
25. Tao, Y.; Li, B.; Jiang, J.; Ng, H.C.; Wang, C.; Li, B. On sharding open blockchains with smart contracts. In Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE), Dallas, TX, USA, 20–24 April 2020; pp. 1357–1368.
26. Ren, L.; Ward, P.A. Transaction Placement in Sharded Blockchains. *arXiv* **2021**, arXiv:2109.07670.
27. Li, M.; Lin, Y.; Zhang, J.; Wang, W. Jenga: Orchestrating smart contracts in sharding-based blockchain for efficient processing. In Proceedings of the 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), Bologna, Italy, 10–13 July 2022; pp. 133–143.
28. Wang, J.; Chenchen, H.; Xiaofeng, Y.; Yongjun, R.; Sherratt, S. Distributed secure storage scheme based on sharding blockchain. *Comput. Mater. Contin.* **2022**, *70*, 4485–4502. [CrossRef]
29. Li, C.Y.; Chen, X.B.; Chen, Y.L.; Hou, Y.Y.; Li, J. A new lattice-based signature scheme in post-quantum blockchain network. *IEEE Access* **2018**, *7*, 2026–2033. [CrossRef]
30. Gao, Y.L.; Chen, X.B.; Chen, Y.L.; Sun, Y.; Niu, X.X.; Yang, Y.X. A secure cryptocurrency scheme based on post-quantum blockchain. *IEEE Access* **2018**, *6*, 27205–27213. [CrossRef]
31. Li, B.; Wu, F. Post Quantum Blockchain with Segregation Witness. In Proceedings of the 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS), Chengdu, China, 23–26 April 2021; pp. 522–527.
32. 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. *Sci. Rep.* **2023**, *13*, 5664. [CrossRef]
33. Sun, X.; Wang, Q.; Kulicki, P.; Sopek, M. A simple voting protocol on quantum blockchain. *Int. J. Theor. Phys.* **2019**, *58*, 275–281. [CrossRef]
34. Serengil, S.I.; Ozpinar, A. LightFace: A Hybrid Deep Face Recognition Framework. In Proceedings of the 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), Istanbul, Turkey, 15–17 October 2020; pp. 23–27. [CrossRef]
35. Shankar, S.; Madarkar, J.; Sharma, P. Securing face recognition system using blockchain technology. In *Proceedings of the International Conference on Machine Learning, Image Processing, Network Security and Data Sciences*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 449–460.
36. Pandey, R.; Zhou, Y.; Govindaraju, V. Deep secure encoding: An application to face recognition. *arXiv* **2015**, arXiv:1506.04340.
37. Jayakumari, B.; Sheeba, S.L.; Eapen, M.; Anbarasi, J.; Ravi, V.; Suganya, A.; Jawahar, M. E-voting system using cloud-based hybrid blockchain technology. *J. Saf. Sci. Resil.* **2024**, *5*, 102–109. [CrossRef]
38. Gilcrest, J.; Carvalho, A. Smart contracts: Legal considerations. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 3277–3281.
39. Sohel Ahmed Joni, Bangladesh-Voter-Synthetic-Dataset (Revision 958f5b4), Hugging Face. 2024. Available online: https://huggingface.co/datasets/jonybepary/Bangladesh-Voter-Synthetic-Dataset (accessed on 3 September 2024),

40. Singh, J.; Rastogi, U.; Goel, Y.; Gupta, B.; Utkarsh. Blockchain-based decentralized voting system security Perspective: Safe and secure for digital voting system. *arXiv* **2023**, arXiv:2303.06306.
41. Ch, R.; Kumari D, J.; Gadekallu, T.R.; Iwendi, C. Distributed-ledger-based blockchain technology for reliable electronic voting system with statistical analysis. *Electronics* **2022**, *11*, 3308. [CrossRef]