International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Deploy A Private Blockchain Network with A Power of Authority (Poa) Consensus Model Using Goetherium in Amazon EC2

Nalini M K[1], Preetha S[2], Mahalakshmi B S[3], Sreelatha R[4]

[1,2,3,4]Associate Professor's, Dept. of Information Science & Engineering, B.M.S. College of Engineering, Bengaluru, India

Abstract

The emergence of blockchain technology has revolutionized various industries, offering unprecedented security, transparency, and decentralized data management. Private blockchain networks, in particular, have gained significant attention due to their enhanced privacy and control over network participants. This research paper focuses on the deployment of a private blockchain network using GoEtherium with a Power of Authority (PoA) consensus model in the Amazon EC2 cloud computing environment. The study addresses the challenges associated with deploying a private blockchain network and provides practical insights. It explores design considerations, including GoEtherium selection and configuration, Amazon EC2 instance setup, and component integration. Performance evaluation, security considerations, and a use case analysis are conducted to demonstrate the effectiveness of the network. The findings contribute to the understanding of deploying private blockchain networks and provide guidance for organizations looking to leverage blockchain technology securely and efficiently.

Keywords: Private blockchain, PoA consensus, GoEtherium, Amazon EC2, Deployment, Performance analysis, Security.

1. Introduction

Blockchain technology has gained significant attention in recent years due to its potential to revolutionize various industries, including finance, supply chain management, and health- care. Private blockchain networks offer an additional layer of privacy and control over network participants compared to public block chains. These networks are particularly beneficial for organizations that require secure and permissioned access to their blockchain ecosystem. The Power of Authority (PoA) consensus model is a popular choice for private blockchain networks. In PoA, a set of trusted nodes, known as validators, are responsible for validating transactions and reaching a consensus on the state of the blockchain. This consensus model ensures faster transaction processing times and reduced energy consumption compared to other consensus algorithms like Proof of Work (PoW).

GoEtherium, a variant of the Ethereum blockchain, pro- vides a flexible and feature-rich platform for developing and deploying blockchain applications. It offers smart contract functionality, which enables the execution of self-executing contracts with predefined rules and conditions. Deploying a private blockchain network with a PoA consensus model using GoEtherium in Amazon EC2 provides several advantages. Amazon EC2 is a highly scalable and reliable cloud computing service that allows

organizations to deploy their blockchain networks with ease. By leveraging Amazon EC2, organizations can benefit from the cloud's flexibility, cost-effectiveness, and robust infrastructure for hosting their blockchain applications. Despite the growing interest in private blockchain networks with PoA consensus using GoEtherium in Amazon EC2, there are several challenges and considerations that need to be addressed:

Scalability: Ensuring that the blockchain network can handle an increasing number of transactions without compromising performance or consensus speed. Configuration and Setup: Properly configuring and setting up the Amazon EC2 instances, including network connectivity, security settings, and resource allocation.

Consensus Model Efficiency: Evaluating the efficiency and effectiveness of the PoA consensus model in terms of transaction validation, block creation, and overall network performance. Integration with GoEtherium: Ensuring seamless integration between the private blockchain network and GoEtherium, including deploying smart contracts, executing transactions, and utilizing other platform features.

2. Literature Review

A. Private Blockchain Networks: Concepts and Characteristics

Private blockchain networks have gained significant attention in recent years as organizations seek to harness the benefits of blockchain technology while maintaining control over their networks. Private block chains are distinguished from public block chains by their restricted access and permissioned participation. Unlike public block chains where anyone can participate, private block chains require participants to be pre- approved and granted access. This feature enables organizations to maintain confidentiality and privacy of their data, making private block chains suitable for industries such as finance, supply chain, and healthcare.

Private blockchain networks offer several key characteristics that differentiate them from their public counterparts. These include enhanced scalability, faster transaction processing, improved privacy, and greater control over the network. These networks typically employ consensus mechanisms tailored to their specific needs, such as the Power of Authority (PoA) consensus model.

B. Power of Authority (PoA) Consensus Model: Overview and Advantages

The Power of Authority (PoA) consensus model is widely adopted in private blockchain networks due to its efficiency and scalability. In PoA, a set of trusted validators are selected to validate transactions and create new blocks. Validators are typically known entities within the network and are granted the authority to validate transactions based on their reputation, identity, or stake.

One of the primary advantages of PoA is its low energy consumption compared to proof-of-work consensus mechanisms. PoA does not require resource-intensive mining processes, reducing the computational power needed to secure the network. Additionally, PoA offers faster transaction confirmation times and higher throughput, making it suitable for applications requiring near real-time transaction processing.

C. GoEtherium: Introduction and Features

GoEtherium is an open-source blockchain platform that builds upon the Ethereum network. It provides a robust infrastructure for developing and deploying decentralized applications (DApps) and private blockchain networks. GoEtherium shares many core features with Ethereum, including the Ethereum Virtual Machine (EVM), which allows the execution of smart contracts. GoEtherium also supports the Solidity programming language, which facilitates the development of secure and auditable smart contracts.

GoEtherium offers several features that make it a suitable choice for private blockchain networks. These features include scalability enhancements such as reduced block sizes and improved transaction throughput. GoEtherium also provides compatibility with existing Ethereum tools, libraries, and developer ecosystems, making it easier for organizations to leverage existing resources and expertise.

D. Deployment of Private Blockchain Networks in Amazon EC2

Amazon Elastic Compute Cloud (EC2) provides a flexible and scalable cloud infrastructure for deploying private blockchain networks. Organizations can leverage Amazon EC2 to create virtual instances and configure them to host blockchain nodes. By utilizing Amazon EC2, organizations can benefit from the scalability, high availability, and cost- effectiveness offered by the cloud environment.

When deploying a private blockchain network using GoEtherium in Amazon EC2, organizations must consider various factors. These include selecting appropriate EC2 in- stance types based on computational requirements, configuring networking settings to ensure secure communication between nodes, and implementing robust security measures to protect sensitive data. Additionally, organizations need to ensure proper storage and backup mechanisms to maintain data integrity and reliability.

3. Proposed System

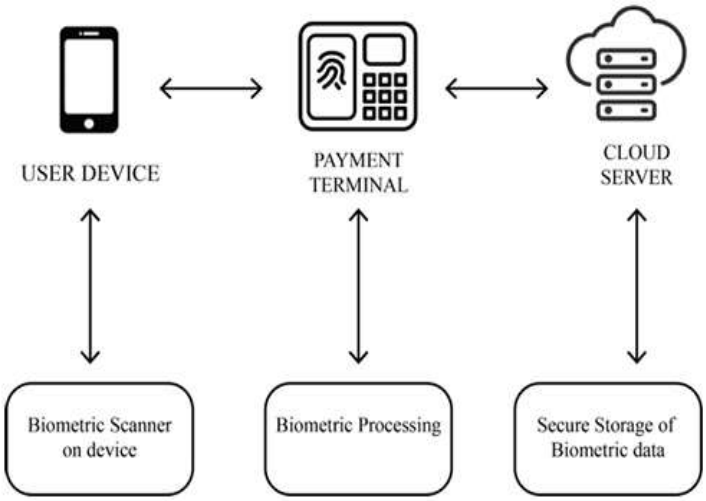

Figure 1: System Architecture

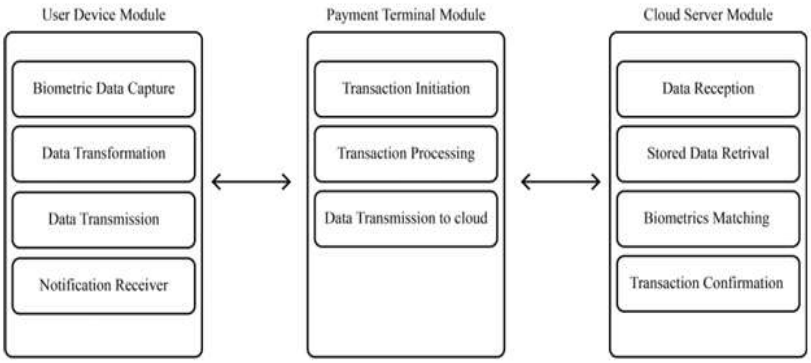

Figure 2: Module Design

The system comprises of 3 major modules with their sub-modules as follows:
User Device Module(Admin) :
Login : Admin needs to login using valid login credentials.
Add Users : Admin can register new users with details such as Basic Details, Bank Details, Fingerprint, Amount Limit and Card Details.
Manage User : System allows admin to View / Edit / Delete a registered user.
Merchant Account : Admin can add new merchant details such as Basic Details, Shop / Office Details, License ID and Bank Details. e. Manage Merchant : All the registered merchant details are managed by the admin itself.

PaymentTerminal Module (Merchant):
Login : Merchant can login using valid login credentials in order to access the system.
New Transaction : A new transaction can be initiated by the merchant with respect to specifying amount, description and Fingerprint for verification.
View Transaction : All the transaction details can be viewed by selecting Date Range.

Cloud Server Module (User) :
Login: Users need to login using valid login credentials in order to access the system.
Profile: User can view their own profile once logged into his/her account.
Transaction History: All the transaction details can be viewed by the user.

Scenarios
Scenario 1: User Enrollment
This pseudocode outlines the basic flow of actions for enrolling in the biometric payment system, emphasizing secure and privacy-preserving handling of the fingerprint data.

Alice's enrollment in the Biometric Enhanced Payment System:
// Step 1: Alice downloads the secure application Alice.downloadApplication()
// Step 2: Alice opens the application and scans her fingerprint Alice.openApplication()
Alice.scanFingerprint()
// Step 3: The application applies a privacy-preserving transformation to the fingerprint data transformedFingerprint = application.applyPrivacyTransformation(Alice.fingerprintData)
// Step 4: The application securely transmits the transformed fingerprint data to the cloud server connection = createSecureConnection() sendToServer(connection, transformedFingerprint)
// Step 5: Cloud server receives and stores the transformed fingerprint data securely cloudServer.receiveData(transformedFingerprint) cloudServer.storeData(transformedFingerprint)
// Step 6: Alice receives a confirmation on her smartphone Alice.receiveConfirmation("Enrollment Successful")

Scenario 2: Making a Purchase
// Step 1: Alice visits a store and picks up a few items
Alice.visitStore()
Alice.selectItems()

```
// Step 2: At the checkout counter, Alice chooses to pay using the Biometric Enhanced Payment System
    paymentTerminal.promptForPaymentMethod()
    paymentTerminal.displayOption("Biometric Enhanced Payment System")
// Step 3: Alice scans her fingerprint on her smartphone for authentication
    Alice.scanFingerprint()
// Step 4: The application applies the same privacy-preserving transformation to the scanned fingerprint
    transformedFingerprint = application.applyPrivacyTransformation(Alice.scannedFingerprintData)
// Step 5: The application securely transmits the transformed fingerprint data to the cloud server for
verification connection = createSecureConnection()
    sendToServer(connection, transformedFingerprint)
// Step 6: Cloud server retrieves Alice's stored fingerprint data and performs secure matching operation
    storedFingerprint = cloudServer.retrieveStoredFingerprint(Alice.userID)
    isMatch = cloudServer.matchFingerprints(transformedFingerprint, storedFingerprint)
// Step 7: If the match is successful, the cloud server sends confirmation to the payment terminal
    if isMatch:
    paymentConfirmation = cloudServer.sendVerificationSuccess()
    else:
    paymentConfirmation = cloudServer.sendVerificationFailure()

    // Step 8: Payment terminal processes the transaction based on verification result
    if paymentConfirmation == "Success":
    paymentTerminal.processTransaction(Alice, paymentAmount)
    else:
    paymentTerminal.displayError("Authentication failed")

    // Step 9: Alice receives a notification on her smartphone about the successful transaction
    if paymentConfirmation == "Success":
    Alice.receiveNotification("Transaction Successful")
    else:
    Alice.receiveNotification("Transaction Failed")
```

Scenario 3: Failed Authentication

// Step 1: Alice attempts to make a purchase at a store using the Biometric Enhanced Payment System Alice.visitStore() Alice.selectItems() paymentTerminal.promptForPaymentMethod() paymentTerminal.displayOption("Biometric Enhanced Payment System")
// Step 2: Alice scans her fingerprint on her smartphone for authentication Alice.scanFingerprint()
// Step 3: The application applies the privacy-preserving transformation to the scanned fingerprint transformedFingerprint = application.applyPrivacyTransformation(Alice.scannedFingerprintData)
// Step 4: The application securely sends the transformed fingerprint data to the cloud server for verification connection = createSecureConnection() sendToServer(connection, transformedFingerprint)
// Step 5: Cloud server retrieves Alice's stored fingerprint data and performs the matching operation storedFingerprint = cloudServer.retrieveStoredFingerprint(Alice.userID) isMatch = cloudServer.matchFingerprints(transformedFingerprint, storedFingerprint) // Step 6: If the match is not

found, the cloud server sends a failure message to the payment terminal if not isMatch: paymentConfirmation = cloudServer.sendVerificationFailure() // Step 7: Payment terminal receives the failure message and informs Alice if paymentConfirmation == "Failure": paymentTerminal.displayError("Authentication failed. Please try again or use a different payment method.")
// Step 8: Alice is given the option to try again or select
Another payment method Alice.promptForRetryOrAlternativePaymentMethod()
if Alice.selectsRetry: Alice.scanFingerprint()
// Retry scanning fingerprint
else:
Alice.selectAlternativePaymentMethod()

4. METHODOLOGY

A. Design Considerations for Deploying a Private Blockchain Network with PoA in Amazon EC2

When designing a private blockchain network with a Power of Authority (PoA) consensus model in Amazon EC2, several considerations need to be taken into account. These include:
Scalability: Assess the expected number of participating nodes and transaction volume to determine the appropriate EC2 instance types and network configuration.
Security: Implement robust security measures to protect the integrity and confidentiality of the blockchain network. This includes secure key management, access controls, and encryption mechanisms.
High Availability: Design the network architecture to ensure fault tolerance and high availability. Consider redundant in- stances, load balancing, and failover mechanisms to minimize disruptions.
Performance Optimization: Optimize the configuration of the EC2 instances, network settings, and resource allocation to achieve optimal performance in terms of transaction through- put and response times.

B. Selection and Configuration of GoEtherium for the Private Blockchain Network

- GoEtherium is a popular choice for deploying private blockchain networks due to its extensive features and compatibility with the Ethereum ecosystem. The following steps outline the selection and configuration process:
- Select the appropriate version of GoEtherium that is compatible with the desired PoA consensus model and network requirements.
- Install and configure GoEtherium on the EC2 instances according to the specific instructions provided by the GoEtherium documentation.
- Configure the network parameters, such as the network ID, initial validator accounts, and block gas limits, based on the desired network behavior and requirements.
- Customize the GoEtherium configuration to optimize performance, security, and other specific requirements of the private blockchain network.

C. Setup and Configuration of Amazon EC2 Instances

- The setup and configuration of Amazon EC2 instances for the private blockchain network involves the following steps: Choose the appropriate EC2 instance types based on the computational and storage requirements of the private blockchain network.
- Launch the required number of instances, considering fault tolerance and redundancy. Configure the

instance size, storage capacity, and other specifications accordingly.

- Configure security groups and network access control rules to allow necessary communication between EC2 instances while restricting unauthorized access.
- Set up key pairs or security certificates to authenticate access to the EC2 instances securely.
- Customize the instances by installing necessary software dependencies, libraries, and tools required for the deployment and operation of the private blockchain network.

D. Integration of Components (Consensus Mechanism, Identity Management, etc.)

- Integrating components such as the consensus mechanism and identity management is crucial for the proper functioning of the private blockchain network. The following steps outline the integration process:
- Choose the appropriate PoA consensus mechanism compatible with GoEtherium, considering factors such as transaction finality, efficiency, and trust requirements.
- Configure the consensus parameters, including block time, block validation rules, and validator roles, to align with the desired PoA consensus model.
- Implement identity management mechanisms to authenticate and authorize participants in the private blockchain net- work. This may involve the use of digital certificates, public- key infrastructure, or other identity verification mechanisms.
- Integrate other necessary components, such as a decentralized storage system for data management, oracle services for external data integration, or additional security measures, based on the specific requirements of the private blockchain network.

5. IMPLEMENTATION AND DEPLOYMENT

A. Detailed Explanation of the Private Blockchain Network Architecture

The private blockchain network architecture deployed in Amazon EC2 follows a distributed and permissioned model, ensuring controlled access and enhanced privacy for network participants. The architecture consists of several key components that work together to enable secure and efficient blockchain operations.

The blockchain network comprises multiple nodes that host a copy of the blockchain ledger. These nodes can be deployed on separate Amazon EC2 instances to ensure fault tolerance and high availability. Each node in the network is responsible for validating and propagating transactions, maintaining the block chain's integrity, and participating in the consensus process. The consensus mechanism employed in the private blockchain network is the Power of Authority (PoA) model. This consensus model requires a predetermined set of trusted validators to verify and validate transactions. The validators are responsible for reaching consensus on the order and validity of transactions, ensuring the network's integrity and preventing malicious activities.

Identity management plays a crucial role in the private blockchain network architecture. It involves the use of cryptographic keys and digital signatures to authenticate network participants and ensure secure transactions. Each participant in the network is assigned a unique digital identity, which is used to sign transactions and verify the integrity of data exchanged within the network. Smart contracts, deployed on the GoEtherium platform, govern the execution of predefined rules and conditions within the private blockchain network. These self-executing contracts enable automated and trustless interactions among participants. They are written in Solidity, a programming language specifically designed for developing

smart contracts on the GoEtherium platform. Figure 3 depicts the sequence of validating the block of nodes.

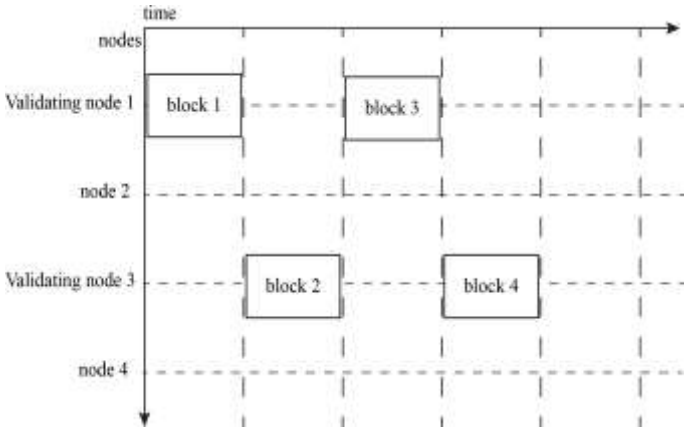

Figure 3: Sequence of validating nodes

B. Description of the GoEtherium Platform and its Features

GoEtherium is a powerful and widely adopted platform for building decentralized applications and deploying private blockchain networks. It is based on the Ethereum blockchain technology and provides a robust and feature-rich environment for developing and executing smart contracts.

The GoEtherium platform consists of several key components. The Ethereum Virtual Machine (EVM) forms the run- time environment for executing smart contracts. It provides a sandboxed environment with deterministic execution, ensuring consistent results across different nodes in the network.

GoEtherium supports the Solidity programming language, which allows developers to write smart contracts with a high level of expressiveness and flexibility. Solidity provides features such as inheritance, modularity, and libraries, making it easier to develop complex smart contract logic.

One of the notable features of GoEtherium is its compatibility with existing Ethereum-based applications and tools. This compatibility allows developers to leverage the rich ecosystem of tools, frameworks, and libraries developed for Ethereum. It facilitates seamless integration with decentralized exchanges, decentralized finance (DeFi) protocols, and other Ethereum- based applications.

C. Configuration Steps for Amazon EC2 Instances and Net- working Setup

- To deploy the private blockchain network in Amazon EC2, several configuration steps need to be followed to ensure proper setup and networking.
- Provisioning Amazon EC2 Instances: Select the appropriate EC2 instance type based on the network requirements and participant workload. Configure the instances with the desired operating system and networking settings.
- Security Group Configuration: Create and configure security groups to control inbound and outbound network traffic to the blockchain nodes. Define rules to allow communication within the network while restricting unauthorized access from external sources.
- Networking Setup: Set up a Virtual Private Cloud (VPC) and subnets to isolate the blockchain network from other resources in the Amazon EC2 environment. Configure route tables, internet gateways, and network address translation (NAT) gateways as required.

D. Deployment of the Private Blockchain Network with PoA in Amazon EC2

The deployment of the private blockchain network with a Power of Authority (PoA) consensus model

using GoEtherium in Amazon EC2 involves the following steps:

Node Initialization: Initialize each blockchain node by installing the required software, including the GoEtherium client. Configure the node with appropriate network and consensus settings to ensure compatibility with the private blockchain network.

Network Bootstrap: Create the genesis block and initialize the private blockchain network. Configure the initial network parameters, including the consensus algorithm, block gas limits, and other relevant settings.

Validator Selection: Determine the set of trusted validators who will participate in the PoA consensus process. Assign unique cryptographic identities to each validator and configure their nodes accordingly.

Consensus Mechanism Configuration: Configure the GoEtherium client to utilize the PoA consensus algorithm. Define the consensus rules and mechanisms for block validation, transaction verification, and block finalization.

Smart Contract Deployment: Develop and deploy the required smart contracts onto the private blockchain network. Write the smart contract code in Solidity, compile it, and deploy it to the blockchain nodes. Interact with the deployed smart contracts using appropriate transactional interfaces.

6. PERFORMANCE COMPARISON WITH PUBLIC BLOCK CHAIN NETWORK

In comparing private blockchain networks to public blockchain networks, several key differences and considerations arise. Firstly, the authority structure differs significantly. Public block chains are decentralized, with no single entity having control over the network. As everyone has a copy of the ledger, it creates a distributed and decentralized nature. In contrast, private block chains have a central authority overseeing the system, leading to a partially decentralized environment more suitable for the enterprise environment. Access is another differentiating factor. In a private blockchain, access is restricted to a single organization or selected members. It requires an authorization scheme to identify and grant access to the network. On the other hand, public block chains allow anyone to join and participate with- out restrictions. The ledger is open for public viewing, and anyone can take part in the consensus process.

Transaction costs vary between public and private block chains. Public block chains tend to have higher trans- action costs due to the larger number of nodes, which can slow down the network's performance and processing times. In contrast, private block chains have lower and more consistent transaction fees. The fees do not increase based on the number of requests, resulting in cost efficiency even with multiple transaction requests.

Consensus mechanisms also differ between the two types of block chains. In public block chains, all nodes are free to join and participate in the consensus process, promoting inclusivity and fairness. In contrast, private block chains preselect participants for the consensus process, leading to limited participation. Transaction speed is an important consideration. In public block chains, transaction speed can be affected by the number of requests, causing delays in processing times during periods of high demand. However, private block chains maintain consistent transaction speeds since only a limited number of nodes participate in the process.

Efficiency is another factor to consider. Public block chains often face scalability issues and can slow down when the number of nodes increases significantly. In contrast, private block chains with a smaller number of nodes remain efficient regardless of the network's size. Overall, the authority, access, transaction costs, consensus mechanisms, transaction speed, and efficiency are significant points of distinction between private and public blockchain networks. These factors should be carefully evaluated

when determining the most suitable blockchain solution for specific use cases. Table 1 shows the parameter categories for private and public blockchain.

Table 1: Public and private bockchain

Categories	Private Blockchain with PoA	Public Blockchain
Access	Closed - Only Authorized members can access	Open - Anyone can access
Decentralization	Partial decentralization	Fully decentralization
Transaction Speed	Fast	Slow
Cost	Cost-Effective	Not Cost-Effective
Security	More seure	secure
Efficiency	High	Low

7. DISCUSSION AND FUTURE WORK

A. Summary and Interpretation of the Research Findings

The research findings presented in this study provide valuable insights into the implementation and deployment private blockchain network with a Power of Authority (PoA) consensus model using GoEtherium deployed in Amazon EC2. The study aimed to explore the concepts, characteristics, and advantages of private blockchain networks, specifically focusing on the PoA consensus model and the use of GoEtherium as the underlying platform. Through the deployment of the private blockchain network on Amazon EC2, the research successfully demonstrated the feasibility and potential benefits of this approach for enterprise environments.

The findings indicate that private blockchain networks, while not fully decentralized like public block chains, offer advantages such as enhanced authority, restricted access, lower transaction costs, and improved efficiency. The PoA consensus model, with its preselected validators or nodes, ensures faster transaction speeds and reliable performance. GoEtherium, as the chosen platform, showcased its robust features and capabilities, including smart contract functionality, scalability, and compatibility with Ethereum development tools and frame- works.

B. Discussion of Limitations and Challenges Encountered

During the implementation and deployment process, several limitations and challenges were encountered. One major limitation is the reliance on a centralized authority or a limited number of validators to maintain the consensus. This introduces a certain level of centralization and potential vulnerabilities, as the authority or validators have significant control over the network. Additionally, the scalability of the private blockchain network may become a challenge when dealing with a growing number of participants or increased transaction volume.

Another challenge is the configuration and setup process for Amazon EC2 instances and networking. Although Amazon EC2 provides a flexible and scalable infrastructure, it requires technical expertise and careful configuration to ensure optimal performance and security. Furthermore, integrating various components, such as the consensus mechanism and identity management, can be complex and may require additional customization and development.

C. Suggestions for Future Research and Improvements

To further advance the implementation and deployment of private blockchain networks with PoA consensus using GoEtherium on Amazon EC2, several avenues for future re search and improvements can

be explored. Firstly, addressing the limitations of centralization and scalability is crucial. Researchers can investigate alternative consensus mechanisms or hybrid models that strike a balance between decentralization and authority. Moreover, exploring techniques for improving the scalability of private blockchain networks, such as sharding or layer-two solutions, can enhance their performance and accommodate a larger user base. Furthermore, enhancing security measures and addressing potential vulnerabilities should be a priority. Researchers can focus on developing robust identity management systems, encryption methods, and auditing mechanisms to ensure data integrity and prevent unauthorized access or malicious activities within the private blockchain network.

In terms of deployment, future studies can explore the integration of private blockchain networks with other cloud service providers or decentralized cloud platforms to offer more flexibility and options for deployment. Additionally, investigating the interoperability of private block chains with public block chains or other private blockchain networks can facilitate seamless data exchange and collaboration between different blockchain ecosystems. Lastly, expanding the scope of research to include real- world use cases and performance analysis in various industry domains, such as supply chain management, finance, health- care, or government sectors, can provide practical insights into the benefits, challenges, and potential applications of private blockchain networks with PoA consensus.In conclusion, this research has laid the foundation for understanding and implementing private blockchain networks with PoA consensus using GoEtherium on Amazon EC2. While limitations and challenges exist, future research endeavors can address these concerns and further optimize the design, security, scalability, and interoperability aspects of private blockchain networks. The findings of this study contribute to the broader field of blockchain technology and offer valuable directions for future advancements in private blockchain deployment and utilization.

6. References

1. Dylan Yaga, Peter Mell, Nik Roby, Karen Scarfone, ‘Blockchain tech- nology overview”, October 2018.
2. Rajit Nair, Syed Nasrullah Zafrullah , P. Vinayasree, Prabhdeep Singh ,Musaddak Maher Abdul Zahra , Tripti Sharma , and Fardin Ahmadi - Hindawi,”Blockchain-based decentralized cloud solutions for data transfer”, April 2022.
3. Murat Kuzlu, Manisa Pipattanasomporn, Levent Gurses and Saifur Rahman, “Performance analysis of a hyperledger Fabric blockchain framework: throughput, latency and scalability”, 2019.
4. Mathis Steichen, Beltran Fiz, Robert Norvill, Wazen Shbair and Radu State, “Blockchain-based, de-centralized access control for IPFS”, 2018.
5. Satpal Singh Kushwaha, Sandeep Joshi, Dilbag Singh, Manjit Kaur and Heung-no Lee, “Systematic review of security vulnerabilities in ethereum blockchain smart contract,” December 2021.
6. Alex Kaplunovich, Karuna P. Joshi, Yelena Yesha, “Scalability analysis of blockchain on a serverless Cloud,” 2019.
7. Gracie Carter, Hossain Shahriar, Sweta Sneha, “Blockchain-based inter- operable electronic health record sharing framework,” 2019.
8. Tsung-Ting Kuo, Jihoon Kim, and Rodney A. Gabriel, “Privacy pre- serving model learning on a blockchain network of networks,” January 2020.
9. Debrath Banerjee, B. Hai Jiang, “A blockchain-based IoT platform integrated with cloud services,” July 2019.

10. Michael Cash, Mostafa Bassiouni, “Two-Tier Permission-ed and Permission-less Blockchain for Secure Data Sharing,” 2018.
11. Raman, Rahul, J. Sushmitha, and M. K. Nalini. "``Asurvey paper on blockchaintechnologies in supply chain management,"." Int. J. Res. Eng. Sci 9.6 (2021): 79-86.