DESIGNING AN EFFICIENT BLOCKCHAIN BASED INTERNET OF THINGS IN SMART FARMING

TSEGAZEWOLD, KINFU

BAHIR DAR UNIVERSITY

BAHIR DAR INSTITUTE OF TECHNOLOGY

SCHOOL OF RESEARCH AND GRADUATE STUDIES

FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING

COMPUTER ENGINEERING

MSc. THESIS ON

DESIGNING AN EFFICIENT BLOCKCHAIN BASED INTERNET OF THINGS IN SMART FARMING

BY

TSEGAZEWOLD KINFU

July 17, 2023

Bahir Dar, Ethiopia

BAHIR DAR UNIVERSITY

BAHIR DAR INSTITUTE OF TECHNOLOGY

FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING

Designing an Efficient Blockchain-Based Internet of Things (IoT) in Smart Farming

BY

TSEGAZEWOLD KINFU

A thesis submitted

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Engineering

Advisor: Yirga Yayeh (PhD)

July 17, 2023

Bahir Dar, Ethiopia

BAHIR DAR UNIVERSITY

BAHIR DAR INSTITUTE OF TECHNOLOGY

SCHOOL OF RESEARCH AND GRADUATE STUDIES

FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING

THESIS APPROVAL SHEET

I hereby confirm that the changes required by the examiners have been carried out and incorporated in the final thesis.

Name of Student: Tsegazewold Kinfu Signature _____ Date Aug 02, 2023

As members of the board of examiners, we examined this thesis entitled "*Designing an Efficient Blockchain Based IoT in Smart Farming*" by Tsegazewold Kinfu. We hereby certify that the thesis is accepted for fulfilling the requirements for the award of the degree of Masters of science in Computer Engineering.

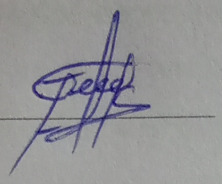
Approved:

Yirga Yayeh (PhD)		July 20, 2023
Name of Advisor:	Signature	Date
Sosina Mengistu (PhD)		July 19, 2023
Name of External Examiner:	Signature	Date
Abebe Tesfahun (PhD)		July 11, 2023
Name of Internal Examiner:	Signature	Date
Wubie Engdaw		21-08-2023
Name of Chair Holder:	Signature	Date
Wubie Engdaw		21-08-2023
Name of Chair Person:	Signature	Date
Teketay Mulu (PhD)		23-08-2023
Name of Faculty Dean:	Signature	Date

Getinet Yenealem (PhD)
Assistant Professor
Post Graduate Coordinator

Declaration

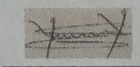
This is to certify that the thesis entitled **"Designing an Efficient Blockchain-Based IoT System in Smart Farming"**, submitted in partial fulfillment of the requirements for the degree of Master of Science in **Computer Engineering** under faculty of **Electrical and Computer Engineering**, Bahir Dar Institute of Technology, is a record of original work carried out by me and has never been submitted to this or any other institution to get any other degree or certificates. The assistance and help I received during the course of this investigation have been duly acknowledged.

Name of the student: Tsegazewold Kinfu Signature: 

Date of submission: June 20, 2023

Place: Bahir Dar

This thesis has been submitted for examination with my approval as a university advisor.

Name of Advisor: Yirga Yayeh (PhD) Signature: 

Acknowledgment

First and foremost, I would like to express my gratitude to the almighty God and the holy saint virgin Mary which have allowed me to achieve this significant milestone in my life. I am truly grateful for their kind-hearted help in all aspects of my journey. I would also like to extend my deepest thanks and heartfelt gratitude to my advisor, **Yirga Yayeh(PhD)**. His exceptional support, generosity, guidance, encouragement, and valuable feedback have been instrumental throughout every stage of my thesis work. His expertise and dedication have greatly contributed to the successful completion of this thesis.

I am also grateful to my lovely wife, Mrs. Samrawit Ereta, for her constant support and encouragement during my years of study and throughout the thesis process. Furthermore, I express my deepest gratitude to my parents and all my best friends who have provided assistance in various ways, contributing to the successful completion of this work. Their support, both emotionally and practically, has been invaluable, and I am truly thankful for their presence in my life.

I would like to express my gratitude to Hawassa University for granting and sponsoring me for this scholarship program. I am truly grateful for the opportunity provided to me. Additionally, I would like to extend my thanks to all the individuals who have supported me, directly or indirectly, in completing my thesis. Their support has been invaluable throughout my academic journey.

Table of Contents

List of Figures

List of Tables

List of Abbrevations and Symbols

List of Abbrevations

AMQP	Advanced Message Queuing Protocol
CFT	Crash Fault Tolerant
CoAP	Constrained Application Protocol
DDS	Data Distribution Service
GDP	Gross Domestic Product
HLF	Hyperledger Fabric
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IoT	Internet of Things
IP	Internet Protocol
LoRaWAN	Long Range Wide Area Network
MQTT	Message Queuing Telemetry Transport
MSP	Membership Service Provider
OS	Ordering Service
PaaS	Platform as a Service
PBFT	Practical Byzantine Fault Tolerance
PoA	Proof of Authority
PoS	Proof of Stake
PoW	Proof of Work
PTM	Peer Transaction Manager
RBAC	Role-Based Access Control
SaaS	Software as a Service
TLS	Transport Layer Security

Abstract

With the increasing adoption of emerging technologies in developing countries, such as smart farming and precision agriculture, the integration of blockchain-based Internet of Things (IoT) systems has gained significant attention. In this study, we propose a four-layer architecture for designing a blockchain-based IoT system in the domain of smart farming. The architecture comprises device nodes, fog nodes, a cloud server, and end users. Hyperledger Fabric is employed as the underlying blockchain framework, ensuring permissioned access and secure data management. Additionally, we incorporate Role-Based Access Control (RBAC) to further enhance security. Through performance analysis, including measures such as throughput and latency, we evaluate the system's efficiency and responsiveness. This study explores the benefits of utilizing Hyperledger Fabric and a permissioned blockchain approach in the context of smart farming. It offers valuable insights for stakeholders and farmers who are seeking secure and reliable IoT solutions for agricultural practices, emphasizing improved security and trust in the field of smart farming.

Key words: **Blockchain, Cloud Computing, Consensus Algorithm, Cryptography Algorithm, Fog Computing.**

Chapter 1

Introduction

1.1 Background

The agriculture sector represents a turning point for the country's global gross domestic product (GDP), sustainable growth, and economic development. According to the World Bank's agriculture and rural development report, approximately 65 percent of the world's population is involved in farming and agriculture, and it is predicted that global agricultural productivity will increase by 1.75 percent annually by 2050 to accommodate the needs of the estimated 10 billion people[1]. Emerging technologies must be applied in the agriculture sector in order to meet productivity and growth. Agriculture is currently ongoing by integrating Wireless sensor networks,Artificial Intelligence (AI), Robotics, Unmanned Aerial Vehicles (UAV), Big Data Analysis (BDA), and Machine Learning[2]. The Internet of Things (IoT) is playing a crucial role in agricultural parameters to increase crop yields, decrease costs, and optimize sensor input data for activities like crop growth status, irrigation, monitoring livestock, soil condition, pesticide and fertilizer, weed control, and so on[2, 3].

In precision agriculture, Artificial Intelligence and machine learning are adopted to predict soil content, crop maturity rotation, optimal planting time, and harvesting time based on agricultural sensor data. IoT-based smart irrigation is also crucial to overcome the scarcity of clean water resources required for many plant varieties and achieve optimal water utilization in smart farming. While there are several challenges with IoT and WSN in terms of security and privacy, recently, Blockchain technology has been introduced to mitigate the challenges and issues in smart farming and precision agriculture[4].

Blockchain technology, created by "Satoshi Nakamoto" in 2008, revolutionized peer-to-peer transactions by introducing a public distributed ledger. This innovative technology eliminated the need for traditional intermediaries like banks, making it possible to conduct secure and transparent transactions. Initially, blockchain found widespread use in the realm of financial transactions, serving as the foundation for decentralized cryptocurrencies like Bitcoin[5]. Due to its interesting features like decentralization, reliability, zero exchange transaction fees, and secure data storage systems, blockchain

technology is being utilized in a variety of industries (such as agriculture, health centers, power grids, electronic voting, and so on) [5, 6].

Every user has their unique address which is protected with the help of public-key cryptography [5]. IoT plays a pivotal role in social and national development, and for interconnecting between humans, computers, and smart devices. This interconnected network enables ubiquitous access to information and services, revolutionizing various industries such as transportation, agriculture, home automation, and health monitoring through wearable technology. By enhancing quality of life, ensuring safety, reducing workload, and enabling effective monitoring and management of agricultural fields, IoT devices have become indispensable. The key components of IoT devices encompass sensors, devices, gateways, and cloud infrastructure. Their integration is instrumental in driving the progress of IoT-enabled solutions[7]. However, maintaining and securing IoT deployments in centralized cloud server farms can be challenging. Integrating blockchain technology with IoT offers a promising solution to address the privacy and security challenges associated with IoT. In addition, Role-Based Access Control (RBAC) plays a crucial role in enhancing security and access control. By assigning specific roles to actors such as farm admins, agronomists, veterinarians, and farmers, RBAC ensures that each user has the appropriate level of access and permissions based on their responsibilities. This structured approach minimizes the risk of unauthorized access, data breaches, and tampering.

In the global context, farmers worldwide often rely on traditional farming techniques, which may hinder their ability to achieve higher crop yields and improve overall quality of life. To tackle this challenge and unlock the potential of smart farming, the integration of blockchain-based IoT technology becomes paramount. By implementing secure, real-time, and efficient monitoring systems, blockchain-based IoT has the potential to enhance productivity and revolutionize the agricultural industry on a global scale. Therefore, the design of a blockchain-based IoT system for smart farming becomes imperative in order to harness these benefits and address the challenges faced by farmers worldwide.

1.2 Motivation of the Study

The motivation behind designing an effective blockchain-based IoT system in smart farming is to address the challenges and limitations of traditional farming practices. The traditional farming methods rely heavily on manual labor and guesswork, which can lead to inefficient use of resources and reduced yields. With the help of IoT technology, farmers can gather real-time data on crop and livestock conditions, weather patterns, and soil moisture levels. This data can then be used to make informed decisions about crop management, irrigation, and fertilization.

However, storing and managing this large volume of data in a secure and transparent manner can be a challenge. This is where blockchain technology comes into play. By using a decentralized and tamper-proof ledger, farmers can store and share data in a secure and transparent manner. This can help to increase trust and accountability among stakeholders, improve supply chain traceability, and reduce fraud and waste.

Therefore, the motivation behind designing an effective blockchain-based IoT system in smart farming is to leverage the benefits of IoT and blockchain technology to improve the efficiency, productivity, and sustainability of the agriculture sector.

1.3 Statement of the Problem

In agriculture, traditional farming practices have been widely adopted for a long period of time, as farmers depend on manual labor and natural techniques to care for their crops and livestock. However, with the advancements in technology, the agricultural sector has also embraced the digital revolution. Smart farming, which involves the use of various technologies such as IoT, has emerged as a solution to address the challenges of traditional farming practices. IoT enables farmers to collect real-time data on crop growth and livestock behavior, monitor environmental conditions such as temperature, humidity, and soil moisture, and automate irrigation and fertilization systems[4]. This technology has the potential to transform the agricultural sector by increasing productivity, and improving the quality of produce.

However, the integration of IoT in smart farming has also posed several challenges. One of the biggest challenges is the limited data storage capacity of IoT devices[3]. This creates a significant impact when integrating with a blockchain-based system, which

requires a large amount of data storage. Additionally, IoT devices have limited computing power, making it difficult to process large amounts of data quickly. The security of IoT devices is another issue that needs to be addressed. These drawbacks limit the scalability and effectiveness of IoT-based smart farming solutions.

To address these challenges, a proposed system is introduced that integrates IoT with blockchain technology in smart farming. The proposed system aims to provide a secure, transparent, and efficient platform for collecting, storing, and processing data. This system includes a permissioned blockchain, Hyperledger Fabric, which is optimized for enterprise use cases. Additionally, the proposed system implements a distributed architecture with multiple nodes to improve data processing speed and ensure high availability. Role-based access control is also implemented to ensure that only authorized actors can access and modify the data.

While, deploying blockchain in smart farming IoT devices is very challenging due to the scarcity of data storage. One of the main concerns of integrating blockchain with IoT is that IoT devices have limited storage capabilities, which makes it difficult to store and manage large amounts of data. To mitigate this challenge, we propose a four-layer architecture that includes a device node layer, a fog node layer, a cloud node layer, and an end-user layer. The device node layer consists of IoT devices that collect data from the farming environment, while the fog node layer is responsible for storing and processing the data in real-time. The cloud node layer provides additional storage and processing capabilities, and the end-user layer allows farmers to access and interact with the system. By using this architecture, we aim to enhance the storage scarcity of IoT devices and provide real-time response from the fog node layer, which can help to reduce latency and improve the performance of the system.

In this study, the following research question are raised:

- What is the understandable consensus algorithm for implementing blockchain-based IoT in smart farming?
- How can the performance of the proposed smart farming system be assessed and measured using appropriate approaches and metrics?
- How does the implementation of RBAC improve security and access control in the smart farming environment?

1.4 Objectives of the Study

1.4.1 General Objective

The general objective of this paper is to design an effective blockchain-based IoT for a smart farming environment.

1.4.2 Specific Objectives

- Determine the understandable consensus algorithm of the proposed system.
- Implement Role-Based Access Control (RBAC) for blockchain-based IoT in smart farming.
- Design chaincode for crop and livestock monitoring.
- Evaluate the performance of the proposed system.

1.5 Scope of the Study

The implementation of Blockchain-based IoT in smart farming posed various challenges, including power consumption, data storage, latency, throughput, data security, and privacy. In this study, the scope focused on testing and analyzing the system's throughput and latency. we utilized a permissioned blockchain framework, such as HLF. The thesis specifically covered crop monitoring and livestock monitoring within the smart farming domain. The experimentation was conducted using a simulation based setup. The study solely relied on simulation observations for testing. As a result, the scope of our research covered the assurance of data security and privacy, as well as addressing latency and throughput concerns in the context of smart farming.

1.6 Significance of the Study

The significance of this study lies in the development of a blockchain-based IoT system for smart farming using Hyperledger Fabric, which effectively addresses key challenges in the agricultural sector. By implementing this system, stakeholders such as farmers, agronomists, veterinarians, and others can securely access and share data related to livestock and crop health, environmental conditions, and other relevant metrics. This

transparent platform enhances the accuracy and timeliness of decision-making, resulting in improved agricultural yields, reduced waste, and increased profitability.

Furthermore, the implementation of a permissioned blockchain like Hyperledger Fabric can ensure data privacy and security, preventing unauthorized access and manipulation of data. The use of role-based access control can also ensure that only authorized users can access and modify specific data, adding an extra layer of security. The proposed system can also contribute to the growing body of research on blockchain technology and its applications in the agricultural sector. The results of this study can provide insights into the challenges and opportunities in implementing blockchain-based IoT systems in smart farming, and can inform future research in this area.

1.7 Contribution of thesis work

The contributions of this study are outlined as follows:

- Designing chaincode for smart farming, specifically crop and livestock monitoring.
- Employing RBAC for all farming actors to ensure proper authorization and access control.
- Providing valuable insights for practitioners and researchers in related domains.
- Contributing to the advancement and improvement of real-world applications relying on this technology.

1.8 Thesis organization

The proposed system for blockchain-based IoT in smart farming is organized into five chapters. **Chapter 1,** includes the background, motivation, statement of the problem, objectives, scope, significance of the study, and thesis organization. **Chapter 2,** covers an overview of blockchain, Hyperledger Fabric, cloud computing, IoT in smart farming, integration of blockchain in smart farming, challenges of blockchain technology in smart farming, and related work. **Chapter 3,** focuses on the proposed architecture, including the blockchain network and performance metrics. **Chapter 4,** includes the Hyperledger implementation, experimental setup, experimental results, and discussion. **Chapter 5,** conclusions and recommendations based on the study's findings.

Chapter 2

Literature Review

2.1 Blockchain Overview

In 2008, "Satoshi Nakamoto" introduced blockchain technology as a public distributed ledger system to facilitate peer-to-peer transactions without the need for intermediaries like banks. Initially, the technology gained prominence in the context of decentralized cryptocurrencies, such as Bitcoin[5]. The idea of blockchain refers to a distributed ledger that is not centralized and stores timestamped transactions between multiple computers in a peer-to-peer network. The purpose is to prevent any tampering with previous records, allowing users to independently and transparently audit and verify transactions. The blockchain is essentially a collection of blocks that are linked using cryptographic methods, forming a growing stack of records. Each block requires a hash code of the previous block, a timestamp, and a set of confirmed transactions[9]. In simple terms, a blockchain is a method of organizing data in a decentralized way, similar to a book with an endless number of pages. Each page (i.e. block), contains new transactions added to the blockchain. The blockchain is managed autonomously through a peer-to-peer network and a publicly available time-stamping server. The decentralized and transparent structure of blockchain enables the tracing and securing of transaction workflow. This characteristic can effectively resolve the long-standing issue of double-spending. Each block in the blockchain is distinguished by a distinct hash value that serves as its identity. The initial block in the chain is referred to as the "genesis block," and it has no parent block[9]. Each block unit consists of a block header and a block body. Especially, the block header involves six components:

- **Block version** is a software version number indicates which con- sensus protocol to follow.
- **Markle Tree Root Hash:** is used to verify the hash code that identifies all block transactions.It recursively defined as a binary tree of hash codes.
- **Timestamp:** is given in seconds since 1/1/1970. It is used to im- mutably track the creation and update time of the block for block integrity guarantee.
- **N-Bits:** identifies the target threshold of hash code that specifies the valid block.

- **Nonce:** A nonce is a 4-byte field used in cryptographic communication that is chosen arbitrarily and can only be used once. It typically starts with '0s' and increases for each hash computation.
- **Parent Block Hash:** is a 256-bit hash code that refers to the previous block. Without this component, there would be no connection and chronology between blocks in the blockchain. Herewith, the the sequence of block hash described in Figure 3.2[10].

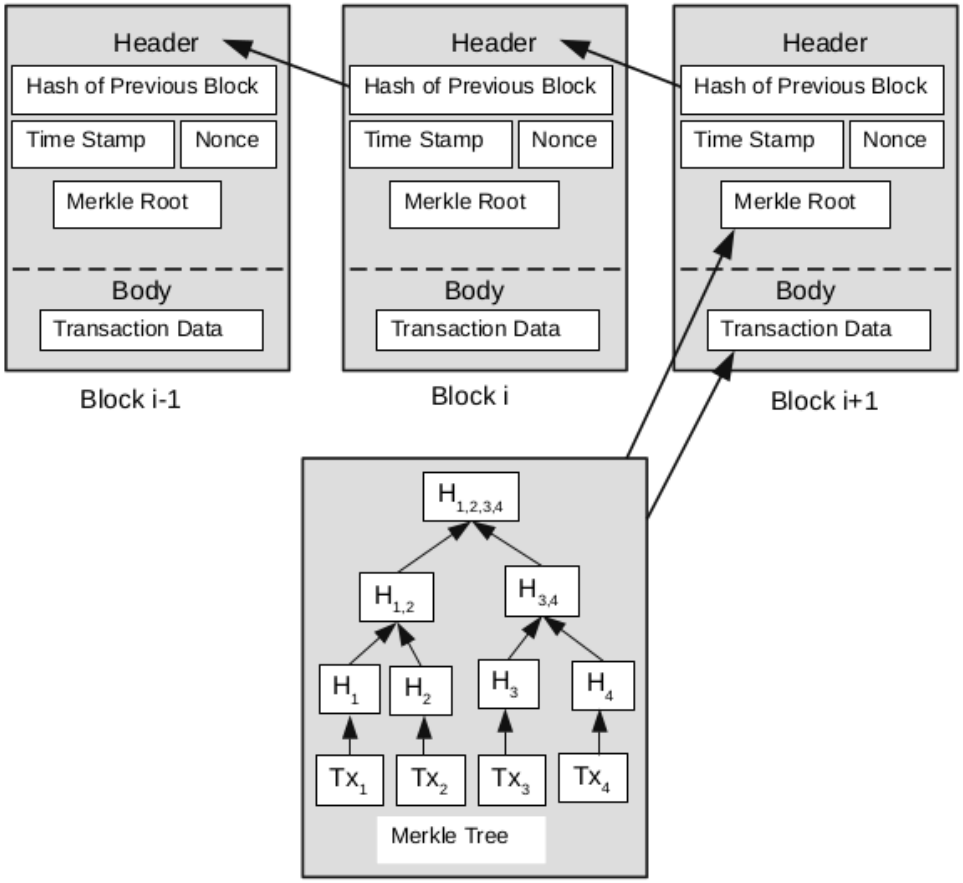


Figure 2.1: Sequence of hashed blocks in Blockchain[10]

Through the process, blockchain technology is adopted in different sectors (for instance, agriculture, health center, power grid, electronic voting, industry, and so on) because of its exciting features such as decentralization, trustworthiness, zero exchange transaction fee, and secure data storage system [5, 6]. Transactions are recorded in blocks, with the first block called the genesis block. Each block consists of a block header and block data. Miners utilize hashing techniques to validate blocks, while the block data stores the latest transactions. The key components of a blockchain include nodes, cryp-

tographic hash functions, ledgers, transactions, asymmetric-key cryptography, blocks, miners, and consensus algorithms[7].

- **Node:-** In blockchain technology, a node refers to a computer or device that is connected to the blockchain network and maintains a copy or part of the blockchain data. Nodes play a crucial role in ensuring the integrity of the blockchain by verifying and validating transactions. There are two types of nodes: full nodes and lightweight nodes. Full nodes store the complete blockchain history, participate in the consensus mechanism, and independently verify transactions. Lightweight nodes store a smaller portion of the blockchain and rely on full nodes for transaction verification and data updates.

- **Cryptographic hash functions:-** Cryptographic hash functions are algorithms used to digest a certain amount of data and produce a fixed-size hash output. The purpose of these hash values is to provide protection against attackers or hackers attempting to alter the original data.

- **Ledger:-** A ledger is a decentralized database used by blockchain to record all transactional data in a secure and immutable manner. It is a continuously growing list of records, called blocks, which contain transaction data and are linked using cryptographic hashes. Every node in the blockchain network has a copy of the ledger, and any change made to the ledger must be validated and approved by consensus of the network. This ensures that the ledger remains trustworthy and tamper-proof, as any attempt to alter the data in one copy of the ledger will be rejected by the consensus of the network.

- **Transactions:-**A transaction refers to the exchange of data between two parties in the blockchain network. It is a process that takes place between authorized parties, and can involve the transfer of digital assets, currency, or information. In a public blockchain, transactions are validated and added to the blockchain through a consensus mechanism. This occurs when a majority of nodes in the network agree on the validity of the transaction, ensuring that only authorized transactions are recorded on the blockchain.

- **Asymmetric-key cryptography:-** also known as public key cryptography, is a method of encryption that uses a pair of keys - a public key and a private key - to securely transfer data and protect the privacy of the parties involved in the

transaction. The public key can be openly shared and is used to encrypt data, while the private key is kept secret and is used to decrypt the data. This ensures that only the intended recipient with the private key can access the decrypted information.

- **Miners:-** Miners are participants in the blockchain network who validate transactions and add them to the block data before it is recorded on the ledger. They invest computational resources and hardware to perform the validation process, and in return, they receive incentives such as cryptocurrency rewards, like bitcoin or Ethereum.

- **Consensus:-** Consensus is the mechanism used in blockchain to ensure that all nodes on the network agree on the state of the ledger. It is the process of reaching a decision on the validity and order of transactions. Consensus algorithms make sure that no duplicate or conflicting transactions are added to the ledger and all nodes validate the blocks in the same way, thus ensuring the integrity and security of the blockchain. Consensus algorithms can vary depending on the type of blockchain, but they all aim to provide a trustworthy and decentralized way to validate transactions and maintain the network.

Trust is a fundamental aspect of blockchain technology that is established through the use of previous block's resulting hash to create the next block. To achieve consensus, miner nodes are responsible for validating the resulting hash and finding the hash for the next block. A Merkle tree is used to bundle a group of transactions together into blocks, with only the Merkle root hash being added to the block[11].

A blockchain transaction involves a sequence of operations performed on states and data stored on the blockchain. Before adding the block to the blockchain, the network nodes must validate the transaction. This process is known as the Consensus Mechanism. Proof of Work and Byzantine Fault Tolerance are the two commonly adopted methods for adding blocks to the network[12].

There are various types of blockchains, and their classification is based on factors such as parameters, managed data, availability, and access control. It is important to note that public or permissionless and private or permissioned blockchains are not synonymous. The distinction lies in the concepts of authentication, which determines who can access the blockchain (public vs private), and authorization, which indicates what the

participants can do (permissioned vs permissionless).

In the case of public blockchains, any individual or entity can participate in the network without the need for any approval or permission. Participants can act as simple nodes that participate in the network or as miners that assist in the validation process. Miners are incentivized with rewards for their contributions to the network in public blockchains like Bitcoin and Ethereum. The success of digital currencies like Bitcoin has brought significant attention to blockchain technology. However, public blockchains have their own set of limitations and drawbacks[13]. such as:

1) Low transaction throughput. Only about 7 transactions can be accepted per second.

2) Long transaction confirmation time. Each transaction takes about 1 hour to be finally confirmed.

3) Waste of resources: The Proof of Work (PoW) mechanism consumes a lot of computing resources and power.

4) Consistency issues.

5) Privacy issues: The transparency of the Bitcoin ledger means that transaction details are openly accessible to anyone on the network. This transparency leads to a lack of privacy in Bitcoin transactions.

To mitigate the challenges associated with public blockchains, private or permissioned blockchains have been implemented for specific organizations. Private blockchains limit participation to approved entities and require permission from the owner to access the network. Some private blockchains also have permissioned access, which controls the actions that users can perform, such as deploying smart contracts or acting as a miner node. For instance, Hyperledger Fabric and Ripple are examples of permissioned private blockchains.

2.1.1 Blockchain Characteristics

This section aims to consolidate and standardize the terminology used to describe key characteristics of blockchain technology[14]. These characteristics have been identified in various literature sources and are considered essential to understanding the nature

of blockchain. Following key characteristics have been identified for the Blockchain technology:

Decentralization

Decentralization is a fundamental characteristic of Blockchain, where the ledger exists on multiple nodes forming a network of computers working in a peer-to-peer (P2P) manner[15, 16]. The system uses a distributed structure to record, store, update, transmit, verify, and maintain information on the network. This eliminates the need for a central authority and transfers control to individual users, making the system fair and more secure. The Blockchain network uses consensus protocols, a set of rules and algorithms, to validate access to information and ensure its consistency and incorruptibility[16]. Consensus is achieved when enough devices agree on what should be recorded on the Blockchain. Each node in the network is independent, with equal rights and obligations, and does not affect the entire network if there is a node-level corruption, guaranteeing improved reliability and robustness of the system. The consensus mechanism also ensures that the information is genuine, as it requires complex calculations to edit or manipulate multiple copies of the information. The distributed nature of the Blockchain also prevents the risk of information being lost or destroyed due to dependency on a centralized location. Moreover, removing a centralized body collecting, recording, and maintaining information improves privacy and eliminates misuse of the data. Finally, lack of reliance on a centralized body for executing and validating transactions reduces intermediary costs and improves performance bottlenecks at central servers[14, 16].

Transparency

The Blockchain ledger allows complete transparency of transactions, allowing anyone to view the details and history of each transaction. This transparency is unique to Blockchain technology and ensures that information is not improperly changed, added, or removed. This high level of transparency is unprecedented, especially for large financial systems. This transparency is achieved through the use of several validating peer nodes on the Blockchain network without a centralized authority[15], as well as making the holdings and transactions of each public address accessible to anyone for traceable and transparent transaction records.

autonomy

Autonomy is a key feature of Blockchain technology. Unlike traditional transactions which rely on trust between parties or the involvement of a reliable third party to guarantee trust, Blockchain allows for a "trust-free" system[14]. This means that the transactions can occur directly between parties without a need for an intermediary. Blockchain achieves this through the use of cryptography which replaces the need for a third party as the governor of trust. "The privacy and unforgeability of asymmetric cryptography ensures that messages are protected and the sender identity is verified, allowing for reliable transactions on the Blockchain system"[17].

The distributed consensus algorithms used by nodes on the Blockchain network enable secure addition or updating of data on the distributed ledger, while also ensuring system integrity and ownership confirmation in the transaction process. These consensus protocols eliminate the need for third-party intervention, reducing transaction costs. This autonomy provided by Blockchain technology has numerous benefits, including increased transparency, accountability, and security in transactions[14].

Security

Blockchain technology is renowned for its robust security, primarily due to its utilization of asymmetric cryptography. This cryptographic approach employs public and private keys to guarantee the credibility and tamper-resistance of transactions, enhancing the overall security of the blockchain system.[17].The security of a Blockchain system encompasses three key aspects: integrity, confidentiality, and transaction authorization. The decentralized nature of Blockchain, along with its peer-to-peer consensus mechanism, mitigates the risk of a single point of failure, commonly found in centralized systems. This enhances the overall resilience of the Blockchain system, making it significantly more challenging to compromise or hack.

Immutability

Immutability refers to the characteristic of a Blockchain system where once data is added to the Blockchain, it cannot be changed or tampered with[15, 16]. This makes the entry of data permanent and unalterable, ensuring the integrity and authenticity of the information. Each block in the Blockchain is encrypted with a hash algorithm

and time-stamped, making it tamper-proof unless a consensus of the majority of the nodes of the whole system agrees to change it[16]. The Blockchain transactions can be viewed by anyone anytime, but once validated and added to the Blockchain, these transactions become irreversible and immutable. Even a small change will generate a different hash, which can be easily detected, ensuring that the shared ledger is immutable. The immutability characteristic of Blockchain provides numerous benefits, especially for financial transactions and financial audits, as it proves that the data has not been altered. However, it also presents challenges and issues, and some people have started to question its benefits.

Anonymity

The anonymity feature of Blockchain technology enables transaction authentication without disclosing personal information of the involved parties, thereby supporting privacy[16]. This is achieved through the use of a defined algorithm for exchanging data between nodes, which establishes trust without the need for node information to be revealed or verified, enabling anonymous information transfer.

Integrity

Integrity is a core attribute of Blockchain systems, ensuring the accuracy and consistency of stored data over time[14]. This feature is achieved through the decentralized and immutable nature of the shared ledger across the Blockchain network. Once a data block is added to the Blockchain, it cannot be edited or modified, and the transaction record of that block is permanently preserved in the system. This ensures that the data is reliable and trustworthy, as multiple copies of it exist on various nodes across the network.

Fault Tolerance

The fault tolerance of a blockchain system refers to its ability to continue operating correctly even when some of its nodes fail or behave incorrectly. This is an important characteristic of blockchain because it ensures that the system can maintain its integrity and reliability despite the possibility of errors or attacks[14]. Blockchain networks are designed to be inefficient and redundant by using a peer-to-peer architecture, where each node is considered equal and can act as both a client and a server. This means that

even if some nodes go offline or experience network issues, the system can continue to function and reach consensus. Blockchains are designed to be Byzantine Fault Tolerant, which means that they can recover from both fail-stop faults and Byzantine faults. Fail-stop faults occur when nodes fail to participate in the consensus protocol due to software or hardware issues, while Byzantine faults occur when nodes start to misbehave due to bugs or malicious attacks. A consensus protocol is considered fault tolerant if it can recover from these types of failures and continue to operate correctly.

Automatic

Smart contracts are automated programs on the blockchain that handle tasks, transactions, and data storage[14]. The consensus protocols used in Blockchain allow for automatic verification of data and transactions by all nodes in the system. The Blockchain itself is also maintained and validated automatically through the protocol without requiring manual intervention. However, it's important to note that just like any other computer program, smart contracts can have bugs and errors that may not be easy to fix or update.

2.1.2 Consensus Algorithm

A consensus algorithm serves as a crucial mechanism utilized by blockchain networks to establish consensus among all participating nodes regarding the current state of the blockchain. It plays a fundamental role in maintaining the integrity and security of the network. Consensus algorithms govern the verification and inclusion of new transactions into the blockchain, as well as ensure that all nodes possess an identical copy of the blockchain. Each existing blockchain system has implemented its own specific consensus algorithm to achieve agreement among network nodes[18, 19]. These the most common algorithms are Proof of Work(PoW), Proof of Stake(PoS), Practical Byzantine Fault Tolerance(PBFT), and Proof of Authority(PoA)

Proof of Work(PoW)

Proof of Work (PoW) is a widely used consensus algorithm introduced by Bitcoin. Nodes in the network compete to find a nonce value by using their computational power in a process called mining[20]. The higher the difficulty level, the fewer blocks are produced. To prevent a single user from gaining control over the system, no user should

use more than 50 percent of the total processing power. In this model, the user's power is directly proportional to the system's overall computational power. The primary objective of consensus models is to eliminate fraudulent nodes, including those that are honest. In a decentralized network, one node must be selected to record all transactions. Random selection is one method, but it is susceptible to attacks. Therefore, if a user or node wants to record transactions, they must demonstrate that their network is not vulnerable to attacks[21].

In this consensus model, users must solve a computationally difficult problem to receive an incentive when a node is added to the blockchain. The network node uses a block header and a nonce to calculate a cryptographic hash function, such as SHA-256. Miners change the nonce to obtain different hash values. They calculate a value that is equal to or less than the consensus value. When a miner achieves the target value, the miner's block is broadcasted to all other nodes in the network. The other nodes confirm the validity of the hash value to each other. If the new block is approved, it is connected to the blockchain by the other miners. Valid blocks can be generated in parallel when the target value is found almost simultaneously by multiple miners. In these cases, branches of blocks, called competing forks, are formed. In this PoW protocol, the longer chain is considered authentic. However, the PoW algorithm requires miners to use a lot of computational power, which leads to a waste of resources.

Proof of Stake (PoS)

Proof of Stake (PoS) was developed as an alternative to the energy-consuming PoW technique[21, 22]. In the PoS algorithm, there is no competition among nodes. Instead, the network chooses a validator node (known as a transaction validator node). The node is selected in advance to be a part of the Proof of Stake and goes through a process of difficulty adjustment similar to PoW. If the initial validator does not validate the transaction, the network selects the subsequent node as a validator, and the process repeats until a validator is found. PoS is considered the best alternative to PoW without wasting resources. It is well-known that in the PoS protocol network, miners with more blocks are less likely to attack. Additionally, miners with more coins are given the ability to produce the next block, but this selection process is unfair, as the wealthiest miner in the network would begin to dominate the others. To address this issue, several solutions have been suggested that consider a miner's number of blocks compared to

the network number. One such solution is Peercoin, which favors selection based on the age of the coin. In Peercoin, the probability of mining the next block is higher for a larger or older set of coins. Another example is Blackcoin, which uses randomization to predict the next block generator. It selects the equation in conjunction with the stake size that gives the lowest hash value. Many blockchains are planning to move gradually from PoW to PoS.

Practical Byzantine Fault Tolerant (PBFT)

The Practical Byzantine Fault Tolerant (PBFT) consensus algorithm is a version of the consensus algorithm that requires the approval of the majority of authority nodes (i.e., at least $N/2 + 1$) for a chain to become a part of the permanent records[22]. This process is more efficient in creating permissioned chains and requires fewer message exchanges. In case a validator does not validate a transaction, the network assigns the next node as a validator.

Hyperledger is the most successful and popular permissioned blockchain used in the industrial and IoT sectors. The RAFT Consensus Protocol is used by permissioned blockchains deployed in enterprise environments, which is a better fit because it is more straightforward and less resource-consuming[18]. The RAFT protocol uses a Crash Fault Tolerant (CFT) ordering service implementation that can tolerate up to $N/2$ failures in the system. The RAFT protocol follows a "leader and follower" approach, where a leader node is dynamically chosen among the ordering nodes in the channel, and the followers replicate its decisions. Deploying RAFT's ordering service is simpler and easier to manage compared to Kafka-based ordering services. Additionally, the RAFT configuration can be created directly from the orderer, unlike the Kafka case, which requires the creation of a Zookeeper cluster to allow the state machine replication process and cannot be configured directly from orderer services. Figure 2.2 [18] illustrates the block creation process using the RAFT consensus protocol.

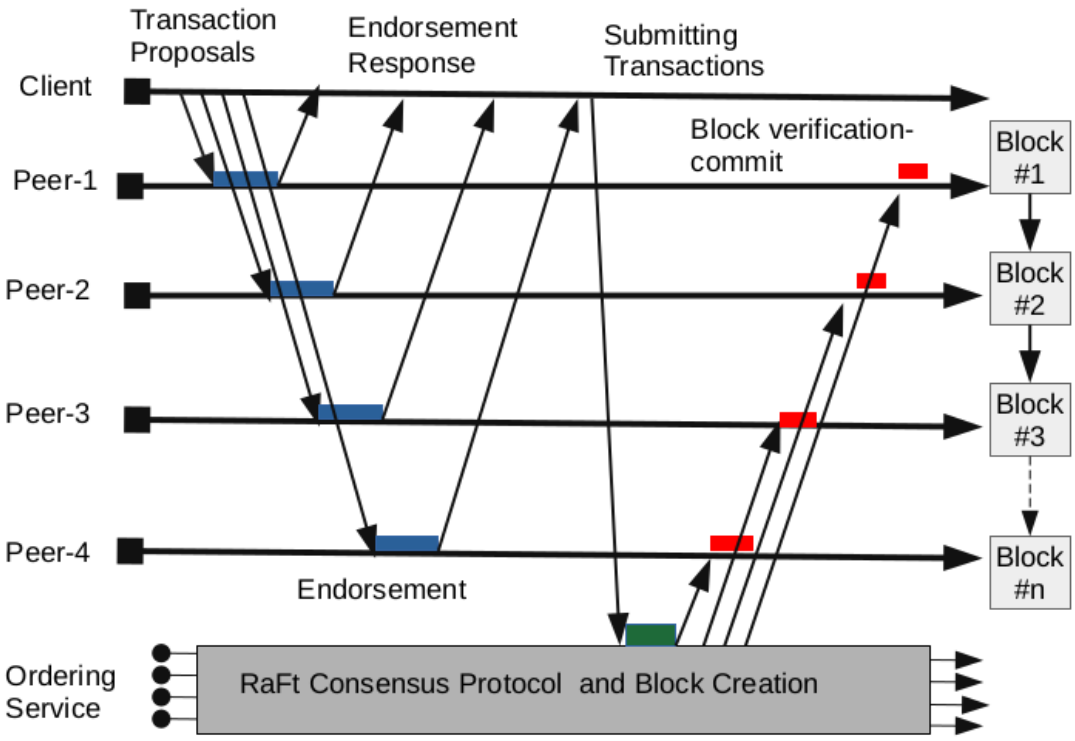


Figure 2.2: RAFT consensus protocol and block creation overview [18]

2.2 Hyperledger Fabric

Hyperledger project, which created an enterprise blockchain development platform, was founded by the Linux Foundation in 2015[12, 13]. Hyperledger Fabric is different from Bitcoin and Ethereum in that it does not contain any cryptocurrencies. This means that only authorized network members are able to access the network, and not just anyone can join the network. Hyperledger Fabric is a permissioned blockchain platform that offers a modular architecture to enable plug-in features in various components, such as membership services and consensuses. The platform is unique in that it does not have its own cryptocurrency. Only network members can access the network, and not anyone can join the network without permission.

The deployment of the Hyperledger Fabric platform is specifically designed for the Linux-Ubuntu operating system, where each component of the platform is initiated as a Docker container[13]. The Docker container creates a secure environment by isolating application programs from physical resources and separating containers from one another. This sandboxed environment ensures the security of the application. The platform uses PBFT as the mechanism to validate transactions and create blocks.Each member

group manages a peer that is responsible for carrying out transactions and monitoring its ledger. The Ordering Service (OS) provides services such as broadcasting messages and ensuring message delivery, while the Certificate Authority (CA) offers certificate services to the blockchain participants network. Overall, Hyperledger Fabric offers a secure and flexible platform for enterprise blockchain development[23].

Hyperledger Fabric is a modular framework that provides scalable components such as encryption, authentication, consensus algorithms, smart contracts, data storage, and other services. All programs run within Docker containers, which provide a secure and isolated environment for each application. Authorization is required for all nodes to join the blockchain network. The channel ensures that the transaction and data are available only to the nodes that are members in the channel. Hyperledger Fabric is an alliance chain that uses a Kafka message queue-based consensus mechanism to reach consensus in scenarios involving numerous large-scale applications. Compared to public chains, Hyperledger Fabric overcomes their limitations and provides a more secure and efficient solution for enterprise blockchain applications.

Hyperledger Fabric employs chaincode, also known as smart contracts, to manage transactions. These smart contracts are computer program codes that facilitate the development and execution of network-based applications. The privacy of transactions between network participants is maintained by isolating them within a channel[12]. Access to the transaction and its associated data is restricted to only those nodes that are members of the channel.

2.2.1 Hyperledger Fabric Components

The main components of the Hyperledger Fabric architecture are peer, ordering service, system chaincodes, Endorsement Policies,channel and client applications[13].

Membership Service

The Membership Service Provider (MSP) is a critical component of the Hyperledger Fabric architecture that manages the identities of all nodes in the system, including clients, peers, and Ordering Service Nodes (OSNs). The MSP is responsible for issuing node credentials that are used for authentication and authorization purposes, ensuring that only authorized nodes can access the network, and validating all transactions by

authorized nodes. It plays a significant role in maintaining the security and integrity of the Hyperledger Fabric blockchain network[24].

Fabric is a permissioned blockchain, meaning that all interactions among nodes occur through authenticated messages, typically using digital signatures. Each node contains an MSP component that authenticates transactions, verifies their integrity, signs and validates endorsements, and authenticates other blockchain operations. The MSP also includes tools for key management and node registration. The MSP can be instantiated in different ways, with the default implementation using standard PKI methods for authentication based on digital signatures and accommodating commercial certification authorities (CAs). Fabric-CA is a stand-alone CA provided by Fabric, and alternative MSP implementations are also possible, such as one relying on anonymous credentials for authorizing a client to invoke a transaction without linking it to an identity.

Fabric supports two modes for setting up a blockchain network. In offline mode, credentials are generated by a CA and distributed out-of-band to all nodes, with only peers and orderers being registered. For enrolling clients, Fabric-CA provides an online mode that issues cryptographic credentials. The MSP configuration must ensure that all nodes, especially all peers, recognize the same identities and authentications as valid. The MSP also allows identity federation, where multiple organizations operate a blockchain network. Each organization issues identities to its members, and every peer recognizes members of all organizations. This can be achieved with multiple MSP instantiations, such as by creating a mapping between each organization and an MSP.

Ledger

In the HLF architecture, the ledger component plays a crucial role in maintaining the ledger and state on persistent storage. It is responsible for facilitating various phases of the transaction lifecycle at each peer node, including simulation, validation, and ledger update[24]. It consists of two main components: the block store and the peer transaction manager (PTM).

The block store is an append-only structure that persists transaction blocks, and includes a few indices for random access to a block or transaction in a block. The block store guarantees high performance since blocks are immutable and arrive in a definite order.

The PTM is responsible for maintaining the latest state in a versioned key-value store.

It stores a tuple for each unique entry key stored by any chaincode, containing its most recently stored value, the latest version, and the block sequence number and sequence number of the transaction that stores the entry within the block. The PTM uses a local key-value store, such as LevelDB or Apache CouchDB, to realize its versioned variant.

During transaction execution, the PTM maintains two sets: readset and writeset. The readset captures tuples (key, ver) for each entry accessed through the GetState function, while the writeset stores tuples (key, val) for each entry updated using the PutState function. Additionally, the PTM supports range queries, where it computes a cryptographic hash of the query results (a set of tuples (key, ver)). To facilitate this, the PTM adds the query string, the hash, and the associated tuples to the readset. This comprehensive approach ensures the PTM's ability to isolate and protect application data while enabling efficient querying capabilities.

The PTM ensures the reliability of transactions by validating them in a specific order. It checks if a transaction conflicts with any preceding transactions within the same block or earlier. For each key in the readset, it compares the recorded version with the latest version in the system. If there is a difference (assuming all previous valid transactions have been committed), the transaction is considered invalid. In the case of range queries, the PTM re-executes the query and verifies that the computed hash matches the one stored in the readset. This approach prevents unexpected data reads and ensures that transactions are processed in a consistent manner.

In case of a peer crash during the ledger update, the PTM has already performed validation and marked transactions as valid or invalid within the block. The ledger writes the block to the ledger block store, flushes it to disk, and subsequently updates the block store indices. The PTM then applies the state changes from writeset of all valid transactions to the local versioned store, computes and persists a value savepoint denoting the largest successfully applied block number, and uses it to recover the indices and latest state from the persisted blocks when recovering from a crash.

Peer

A peer node plays two primary roles in a blockchain network[25].The peer node in Hyperledger Fabric performs two main functions. Firstly, it executes the chaincode, which represents the business logic of a smart contract, on behalf of the user. This execution

involves processing and validating transactions according to the defined rules and logic. Secondly, the peer node stores and maintains the shared state data of the blockchain on a file system. It provides the chaincode with access to this shared state through specific ledger APIs. There are two types of peer nodes: endorsing peers and committing peers. Endorsing peers possess the chaincode logic and validate transactions by endorsing them. On the other hand, committing peers do not possess the chaincode logic but are responsible for storing and maintaining the ledger and current state, known as StateDB, in a key-value storage format. This storage format enables the chaincode to interact with and modify the state using a database query language.

Endorsement Policies

An endorsement policy in a blockchain network is a set of rules that defines which peer nodes must simulate a transaction and endorse its execution results by digitally signing them. Since chaincodes in a blockchain network can run on peer nodes that are not fully trusted, non-deterministic execution and trusting the results from any given peer can be problematic[26]. The endorsement policy addresses these issues by specifying a group of peer nodes that must participate in the endorsement process. This policy is typically based on boolean expressions that involve the identities of network principals, who are members of specific organizations within the network. By enforcing endorsement policies, the blockchain network ensures that transactions are executed consistently and trustworthily across all participating nodes.

System Chaincodes

System chaincodes in Hyperledger Fabric are similar to user chaincodes in terms of programming model, but they are integrated within the peer executable instead of being a separate entity. They serve important functions in the fabric network and include different types of system chaincodes[26].

One such type is the Lifecycle System Chaincode (LSCC), which is responsible for managing chaincode installation, instantiation, and upgrade. It helps to ensure that only valid chaincodes are deployed in the network and allows for version control.Another type is the Endorsement System Chaincode (ESCC), which plays a crucial role in the transaction endorsement process. It executes the transaction logic and generates a proposal response, which is then digitally signed by the endorsing peers.

The Validation System Chaincode (VSCC) is responsible for validating the endorsement signatures against the endorsement policy defined for the chaincode. It ensures that only the required set of endorsing peers have signed the transaction and that the policy has been satisfied before the transaction can be committed. The Configuration System Chaincode (CSCC) manages channel configuration, such as adding or removing peers from a channel, updating the channel's endorsement policy, and other similar actions. It allows for the dynamic management of the network's configuration without having to bring down the entire network.

Channel

In Hyperledger Fabric, a channel represents a private and isolated communication subnet that connects multiple peers. Transactions conducted within a channel are only visible to the participating peers and members of that channel. The ledger and chaincodes associated with the channel are unique and cannot be modified from outside the channel. Furthermore, the consensus mechanism is applied separately for each channel, which means that transactions on different channels are not ordered or linked to one another.

Ordering Service

In the Hyperledger Fabric architecture, the Ordering Service is a crucial component that handles the ordering and sequencing of transactions. Its primary role is to create blocks of transactions and ensure their proper order and integrity before distributing them to peer nodes. The Ordering Service comprises nodes known as Ordering Service Nodes (OSNs), which play an active role in the consensus protocol to achieve agreement on the transaction order.

The Ordering Service supports a pluggable consensus mechanism, which allows for different consensus algorithms to be used based on the needs of the network. By default, a serial ordering is achieved using an underlying Kafka/Zookeeper cluster. The OSNs publish transactions to Kafka topics and leverage the ordered and immutable nature of records in Kafka topic to generate a unique ordered sequence of transactions in a block.The Ordering Service manages multiple channels in the network and provides channel-specific services, including ordering and sequencing of transactions, and block distribution to peer nodes[24]. It also enforces channel-level policies, such as the max-

imum block size and maximum time between block cuttings. The Ordering Service ensures that the blocks are delivered in a tamper-proof and reliable manner to all peer nodes in the network. On every channel, it provides the following services:

(1) **Atomic broadcast:** This service facilitates the establishment of transaction order by implementing the broadcast and deliver calls. These calls are responsible for broadcasting the transactions to the participating nodes and delivering them in the agreed-upon order.

(2) **Channel reconfiguration:** When members of a channel modify it by broadcasting a configuration update transaction, the ordering service facilitates the reconfiguration of the channel.

(3) **Access control (optional):** In certain configurations, the ordering service acts as a trusted entity that restricts the broadcasting of transactions and receiving of blocks to specified clients and peers. This service is responsible for ensuring that only authorized participants are allowed to access the channel.

The ordering service is initialized with a genesis block on the system channel that carries a configuration transaction defining the ordering service properties. In production, the OSNs implement the operations and communicate through the system channel, with Apache Kafka providing the atomic broadcast function. The OSNs act as intermediaries between peers and Kafka.

When an OSN receives a new transaction, it injects it directly into the atomic broadcast. The OSNs batch transactions from the atomic broadcast to form blocks, which are cut based on one of three conditions: (1) the block has reached the specified maximum number of transactions; (2) the block has reached the maximum size in bytes; or (3) a specified amount of time has elapsed since the first transaction of a new block was received.

The batching process is deterministic, ensuring that the same blocks are produced at all nodes. To ensure deterministic block production in the third case, an OSN starts a timer when it reads the first transaction in a block from the atomic broadcast. If the block is not yet cut when the timer expires, the OSN broadcasts a special time-to-cut transaction on the channel, indicating the sequence number of the block that it intends to cut. Upon receiving the first time-to-cut transaction for a given block number, every

OSN immediately cuts a new block. The OSNs persist a range of the most recently delivered blocks directly to their file system so that they can answer peers retrieving blocks through deliver.

The ordering service also provides reconfiguration of a channel when its members modify the channel by broadcasting a configuration update transaction. Optionally, the ordering service may provide access control to restrict broadcasting of transactions and receiving of blocks to specified clients and peers.

Client

The responsibility of creating a transaction proposal lies with the client application. The client submits the proposal to one or more peers simultaneously, and these peers respond with endorsements that comply with the endorsement policy. Once the endorsements are collected, the client broadcasts the transaction to the orderer, where it is included in a block and sent to all peers for validation and commitment.

2.3 Cloud Computing

In the current age of the Internet, there exist millions of websites online. However, maintaining such sites requires a stack of servers, which can be quite costly. Additionally, these servers need to be monitored and maintained consistently to ensure constant traffic rates. This maintenance requires hiring individuals to organize and manage the servers, and store data in data centers. Unfortunately, these efforts can distract from achieving business goals. To avoid this issue, businesses are turning to "Cloud Computing". This refers to the practice of using a network of remote servers to store, manage, and process data from anywhere in the world, rather than relying on local servers or personal computers. Through cloud computing, data and applications can be delivered to an organization's devices via the internet[21]. Cloud computing offers a range of benefits by integrating data centers, resources, and servers via the internet. These services operate on a pay-per-use model and can be accessed globally at a significantly reduced cost, thus enhancing employee collaboration. Furthermore, cloud software is automatically updated, making it easy to manage. Service users also have control over their documents in the cloud. However, cloud computing has limitations. The flexible nature of cloud data makes it vulnerable to security and privacy challenges, as well as attacks.

Moreover, during periods of heavy user traffic, the cloud may experience downtimes.

Cloud computing offers a variety of services, which can be broadly categorized into three delivery models. The first model is Software as a Service (SaaS), where the Cloud Service Provider hosts the application and makes it available to customers over the internet. The provider delivers the complete application as a single platform of software running in the cloud, offering multiple services for many users. However, customers do not have control over the cloud infrastructure. Examples of SaaS include Amazon Web Services, SalesForce.com, and Google Mail.

The second model is Platform as a Service (PaaS), where the cloud service provider allows users to deploy their applications and programming languages within the platform. The key difference between SaaS and PaaS is that while SaaS hosts the entire application in the cloud, PaaS provides a platform for the application. An example of PaaS is the Google search engine.

The third model is Infrastructure as a Service (IaaS), which enables users to directly access storage, processing, and other resources over the network. Virtualization is utilized in IaaS to distribute physical resources and meet the resources demand from cloud customers. The best virtualization method involves setting up independent virtual machines that are separate from the underlying hardware and other virtual machines. To ensure security, servers are provided with a unique IP address. Examples of IaaS include Amazon EC2 and GoGrid.

2.4 Internet of Things in Smart Farming

2.4.1 Overview

The Internet of Things (IoT) is making a significant impact on diverse sectors such as manufacturing, healthcare, communication, energy, and agriculture, with the aim of reducing inefficiencies and enhancing performance across all industries[8]. The concept of IoT refers to a network of physical objects, devices, and machines embedded with sensors, software, and other technologies that enable them to communicate with each other and exchange data over the internet[29]. These interconnected devices encompass a wide range of items, including household appliances, electronics, furniture, industrial and agricultural machinery, vehicles, and even individuals. By generating and

collecting data about their operating environment, these devices facilitate monitoring, control, and optimization of various processes, leading to improved efficiency and the emergence of new services and applications. The IoT ecosystem encompasses various components, such as sensors, gateways, cloud services, analytics, and applications, all working harmoniously to enable seamless communication and interaction between devices and humans[27].

The idea of IoT is not new, but its popularity has grown significantly in recent years, primarily due to advancements in supporting technologies. These advancements of improved hardware, characterized by smaller size and lower power consumption, as well as developments in internet connectivity, wireless connections between devices, cloud computing, artificial intelligence, and big data. These technological elements collaborate to create a network of interconnected devices that can exchange data and information while actively responding to inputs from the network.

In smart farming, sensors play a critical role in IoT technology. They enable the measurement of various environmental variables, including temperature, humidity, soil moisture, light intensity, and others. These sensors can be strategically placed across the farm to gather real-time data that can be analyzed to make informed decisions. For instance, IoT sensors can be used to monitor soil moisture levels, providing farmers with feedback on when to irrigate crops, the ideal time to plant, and the best fertilizers to use based on soil conditions. This information can be transmitted to farmers through an IoT-enabled platform, enabling them to make quick and informed decisions.

Furthermore, in the realm of smart farming, the Internet of Things (IoT) can also be utilized to monitor livestock health, track their movements, and maintain records of their feeding patterns. By deploying IoT-enabled cameras and other sensors, farmers can track the behavior of their animals, identify symptoms of illnesses at an early stage, and take necessary steps to prevent disease outbreaks.

To achieve precision agriculture, four key factors need to be considered: physical structure, data acquisition, data processing, and data analytics. The physical structure is crucial to ensure that the system is designed in a way that can control the sensors, actuators, and other devices effectively, minimizing the risk of any unwanted incidents[28]. In precision agriculture, the role of sensors is critical in gathering data concerning soil quality, temperature, weather conditions, light intensity, and moisture levels. In addi-

tion, devices perform various control functions, including node discovery, device identification, and naming services. These tasks are executed by sensors and devices that are governed by a microcontroller. Typically, a remote device or computer connected to the Internet controls the microcontroller, enabling remote monitoring and control of the precision agriculture system. The physical configuration of the system is also crucial in ensuring proper functionality and preventing potential issues.

Data Acquisition is a crucial step in smart farming, which involves the collection of data from various sensors and devices to provide meaningful insights into the farm's conditions. There are two main components of data acquisition: IoT Data Acquisition and Standard Data Acquisition.

IoT Data Acquisition

IoT data acquisition involves the utilization of multiple protocols to enable the seamless transfer of data from sensors and devices to a central server or cloud platform for further analysis and processing. Several commonly employed protocols in this domain include MQTT, Websocket, AMQP, CoAP, DDS, and HTTP. Each protocol serves a specific purpose and offers distinct features to accommodate the diverse requirements and conditions of farm environments.

MQTT, known as Message Queuing Telemetry Transport, is a lightweight publish-subscribe messaging protocol. It facilitates efficient communication between IoT devices and a central broker, allowing devices to publish data while subscribers receive relevant information. MQTT is particularly well-suited for resource-constrained devices and networks, making it an ideal choice in scenarios where efficiency is crucial.

Websocket, on the other hand, is a communication protocol that enables real-time, bidirectional communication between a client and a server. It establishes a persistent connection, allowing continuous data transmission and immediate updates. Websocket is valuable in IoT applications that require real-time communication and two-way data exchange.

AMQP, or Advanced Message Queuing Protocol, ensures reliable and secure message exchange between devices and systems. It can handle high message volumes and offers features such as message queuing, routing, and transactions. AMQP is designed to facilitate interoperability between different systems and devices.

CoAP, known as the Constrained Application Protocol, is a lightweight communication protocol designed specifically for resource-constrained devices and low-power networks. It adopts a request-response model similar to HTTP but with minimal overhead. CoAP is highly energy-efficient and particularly suitable for IoT implementations in agricultural settings where simplicity and energy conservation are essential considerations.

DDS, or Data Distribution Service, is a publish-subscribe messaging standard designed for real-time and mission-critical systems. It enables high-performance data distribution among devices in complex IoT architectures. DDS focuses on reliable and efficient data exchange, supporting different communication models and offering fine-grained data filtering and management.

HTTP, or Hypertext Transfer Protocol, is a widely adopted protocol used for web communication. In IoT data acquisition, HTTP is often employed to transmit data from IoT devices to servers or cloud platforms. It follows a request-response model and enables integration with web-based systems.

Standard Data Acquisition

Traditional data acquisition systems in agriculture commonly utilize protocols such as ZigBee, Wi-Fi, LoRaWAN, Sigfox, and ISOBUS to establish wireless connectivity between sensors and devices. These widely accepted protocols have been widely adopted due to their reliability and established communication capabilities. They enable seamless data transfer and efficient data acquisition in various agricultural applications, ensuring smooth and effective operations.

ZigBee is specifically designed for short-range data transmission in agriculture, connecting sensors, devices, and control systems within a limited area. With its low power consumption, ZigBee is ideal for energy-efficient applications where reliable communication is essential, even at relatively low data rates.

In contrast, Wi-Fi is a popular wireless communication protocol that offers high-speed data transmission over short to medium ranges. It is commonly used in agriculture to establish connectivity between sensors, devices, and farm equipment. Wi-Fi's higher bandwidth capabilities make it suitable for real-time data monitoring, remote control, and integration with web-based systems.

LoRaWAN is a specialized protocol designed for long-range communication with low power consumption, catering specifically to agricultural applications. It excels in remote monitoring and control scenarios, enabling data transmission across vast agricultural areas. Its low power consumption ensures extended battery life for sensors and devices, making it well-suited for long-term deployments. LoRaWAN also provides reliable connectivity, even in challenging rural environments.

ISOBUS is an agricultural communication protocol that serves the purpose of enabling interoperability among diverse agricultural machinery and equipment. It guarantees smooth data exchange and seamless transmission of control commands between different equipment brands and types. By utilizing ISOBUS, tractors, implements, and precision farming equipment can operate in an integrated and efficient manner, operating seamlessly as a unified entity.

These protocols, ZigBee, Wi-Fi, LoRaWAN, Sigfox, and ISOBUS, serve as reliable and established means of wireless connectivity in standard data acquisition systems for agriculture. Each protocol has its own unique features and applications, catering to specific requirements and enabling effective communication and data transfer in agricultural settings.

Data Processing

Data processing is a crucial component of precision agriculture and involves various features. Image or video processing is used to analyze the data collected from cameras or drones, which can help detect plant diseases, pests, and other anomalies. Data loading involves the transfer of data from sensors to the cloud or a central database for further analysis. Decision support systems use data analytics to provide farmers with real-time insights and recommendations for improving crop yield and efficiency. Data mining involves the extraction of patterns and trends from large datasets, which can help identify areas for improvement and optimize farming practices. Additional features can be added to the data processing component depending on the specific needs of the system.

Data Analytics

Data analytics plays a vital role in smart farming, encompassing two key components: monitoring and controlling[28]. The monitoring aspect involves three basic applications in smart agriculture, namely Livestock Monitoring, Field Monitoring, and Greenhouse Monitoring.Livestock Monitoring utilizes multiple sensors to track various animal health parameters, including temperature, heart rate, and digestion. This enables farmers to proactively prevent disease outbreaks and ensure the well-being of their livestock. Field Monitoring focuses on reporting environmental conditions in the field, such as soil fertility, temperature, humidity, gas levels, air pressure, water pressure, and crop disease monitoring. Intelligent IoT devices and sensors collect this data, empowering farmers to make informed decisions on irrigation, fertilization, and crop harvesting. Smart Greenhouse design eliminates the need for manual intervention by employing intelligent IoT devices and sensors to measure climate parameters. This technology enables farmers to control the greenhouse environment, optimizing plant growth and yield.

The field of ubiquitous computing has been significantly transformed by the introduction of IoT, which has found diverse industrial applications utilizing a wide range of sensors. Nevertheless, there are certain constraints inherent in IoT that must be tackled to enhance its overall efficiency as a system.

- **Privacy:** The information gathered from IoT devices is sent to a centralized cloud storage for analysis and processing, often involving the involvement of a third party. This sharing of data without the explicit consent of the user can potentially result in data breaches that could jeopardize the privacy of individuals.

- **Standards:** Lack of standards and regulations can result in the interoperability issues between different IoT devices, which can cause undesirable consequences while dealing with the configured devices. This can lead to difficulties in integrating different devices from different manufacturers into a single IoT system. The lack of standards can also create security vulnerabilities and increase the complexity of maintaining and updating IoT systems. Therefore, it is crucial to have established standards and regulations to ensure the seamless integration of IoT devices and enhance their overall security and efficiency.

- **Latency:** Latency refers to the delay that occurs when information is transmitted between devices. In the case of IoT, latency can occur when there are delays in the communication between connected devices, which can affect the responsiveness of the system. This can be particularly problematic for applications that require real-time processing, such as autonomous vehicles or industrial control systems. Addressing latency issues in IoT systems is important to ensure that devices can communicate quickly and efficiently with one another.

2.5 Integration of Blockchain and IoT in Smart Farming

Integrating blockchain technology with IoT devices can improve privacy and security by decentralizing data storage and eliminating the need for a centralized cloud server farm. With blockchain, data is stored on a distributed ledger that is secured by cryptography, making it virtually impossible for unauthorized access or tampering. Additionally, blockchain provides a transparent and immutable record of all transactions, ensuring data integrity and accountability. The use of smart contracts also allows for automated execution of agreements between IoT devices without the need for intermediaries.

The integration of the IoT can greatly benefit farmers in monitoring and ensuring the quality of their crops, plants, and animals. With the help of IoT software installed on smartphones, computers, or tablets, farmers can remotely monitor and control soil quality, irrigation activities, pest and disease infestations, and other farm-related activities. By integrating IoT with blockchain technology, the agricultural food supply chain can become more efficient and predictable as the entire process can be monitored and verified.

A blockchain-based smart farming system allows farmers to receive agricultural data quickly on a single integrated platform [1]. Blockchain technology ensures that data is stored securely and transparently, making it resistant to tampering and providing a permanent record of transactions. This is achieved through multiple cryptographic techniques and consensus mechanisms that verify the authenticity of data before it is added to the blockchain. The immutability of the blockchain data provides transparency, anonymity, and traceability, as well as assurance that the data will be reliable

and available when needed in the future. The integration of blockchain in agriculture can improve predictability and minimize output uncertainty, leading to increased profits and reduced resource waste.

IoT combined with blockchain technology might play a crucial role in smart agriculture and the food supply chain without the use of a trusted third party, and all stakeholders could gain greatly [30]. The Internet of Things (IoT) component focuses on the data collected by sensors deployed on farms. IoT devices generate data, which is stored in the system. For example, during the manufacturing stage, critical information and production log data such as product name and origin are collected. Later, data on the growth of the product will be recorded several times, and all stakeholders will have access to this data. The blockchain will provide data storage, consensus, encryption, decryption, and verification functions. Smart contracts will be utilized to execute the required logic at certain times, enhancing scalability, streamlining the process, and saving costs[30].

2.6 Challenges of Blockchain Technology in Smart Farming

The integration of blockchain and IoT is intended to provide security for peer-to-peer transactions and safeguard IoT devices against cyber threats. Nevertheless, there are four significant challenges that must be addressed. These include denial-of-service (DoS) attacks, Sybil attacks, Eclipse attacks, and Routing attacks[30]. A denial-of-service (DoS) attack is a type of cyber-attack in which an attacker attempts to disrupt the availability of an IoT device or network by overwhelming it with traffic or requests, rendering it unable to respond to legitimate traffic. In a DoS attack, the attacker floods the target device with packets or requests, using up its resources and bandwidth, which can result in a temporary or permanent disruption of the device's normal operation. DoS attacks can be launched from a single source or from multiple sources, making them difficult to defend against.

The Sybil attack is a type of attack that targets Blockchain-based IoT networks in smart farming. The attackers create fake connections in the network by using fake IoT nodes. This attack aims to deceive the network by creating multiple identities that are controlled by a single attacker. By doing this, the attacker can control the network and manipulate data. It can be difficult to distinguish between genuine and fake connections, which makes it challenging to detect and prevent the Sybil attack.

An eclipse attack is a type of attack on peer-to-peer networks where an attacker tries to isolate a targeted node by surrounding it with malicious nodes that are controlled by the attacker. The attacker aims to control all incoming and outgoing communications of the victim node, and manipulate or tamper with the data exchanged between the victim node and other legitimate nodes. This type of attack can cause significant damage to IoT devices in smart farming systems as it can prevent the devices from communicating with the rest of the network and compromise their security and functionality.

A router attack is a type of attack where an attacker tries to intercept and modify the messages being transmitted between IoT devices before they reach their intended destination. The attacker may be able to gain access to the router that the devices are connected to and modify the routing tables or other settings to redirect the traffic to their own device. This allows the attacker to manipulate the data being transmitted and potentially steal sensitive information or cause damage to the system. It is a significant threat to the security and integrity of blockchain-based IoT networks[3].

The size of the blockchain is an important factor to consider in IoT networks because as more transactions occur, the blockchain size will increase significantly. This can lead to a need for large memory size to store the blockchain, which in turn leads to waste of computational power and resources. To mitigate this challenge, lightweight blockchain solutions such as mini-blockchain have been developed. These solutions are designed to reduce energy consumption and optimize the use of resources in blockchain-based peer-to-peer transactions.

Another important consideration for blockchain based IoT in smart farming applications is throughput, which refers to the number of transactions that can be processed within a given period of time. A blockchain algorithm that can process transactions quickly is needed to ensure efficient and timely completion of agricultural transactions. Additionally, latency, which is the time it takes to create a new block in the blockchain, is also an important factor to consider in ensuring the reliability and efficiency of blockchain-based IoT networks in smart farming[30].

2.7 Related Work

In this section, we review and highlight related work found in the literature on blockchain based IoT in smart farming. Torky, M., and Hassanein, A. E [30] pointed out the op-

portunities, challenges, and issues of blockchain integrated with IoT in precision agriculture and they proposed a novel blockchain model to mitigate the drawbacks of IoT sensor devices implementation in smart farming. The agricultural data are transacted in a secure and efficient manner by using a blockchain distributed ledger. Furthermore, they figured out the challenges blockchain to deploy in smart farming with regarding to security, power consumption, storage devices, latency and throughput.

Hang et al. [31] suggested that blockchain-based secure fish farms for integrity of agricultural data. The proposed architecture has 4 components (such as fish farm, blockchain network, data storage and end-user). The fish farming environment has been designed with real-time monitoring and management system by using various sensors (temperature sensor, water level sensor, oxygen sensor, and PH sensor) and the actuators (water pump, pond heater, fish feeder, and light LED) responsible for regulating the environment based on the sensor data. Peer-to-peer data transaction between the end-users and devices is held by using a blockchain network. Blockchain network interconnected to fish farm environment and the sensors and the actuators exchange information in a fish farm in order to build complete fish farm legacy. They proposed a permissioned blockchain network to overcome the privacy data exposure and to ensure authorization and authentication. However, they didn't consider the performance metrics of latency and throughput of the blockchain network.

Hu et al. [32] proposed a blockchain-based edge computing solution for the organic agricultural supply chain (OASC) to mitigate the challenges of centralization, monopoly, and asymmetry that have led to a lack of information transparency within the OASC. In case of lack of transparency, the Organic Agricultural Supply Chain leads to trust disclosure of customers. The authors presented an Ethereum based blockchain integrated with Edge Computing in order to manage the trust of diverse architecture and computing models in a distributed manner. The stakeholders (such as users and providers, IoT data sources, software components, fog nodes and cloud storage) maintain independently by their own blockchain services. They introduced a consensus algorithm to improve the performance and the costs have been declined when compared to the traditional consensus algorithm.

Salah et al. [6] conducted a literature review and proposed a solution that utilizes blockchain technology to establish a traceability system for soybeans. The aim is to en-

hance the safety and integrity of agricultural products within the food supply chain. The integration of Ethereum blockchain and smart contracts enables effective traceability and tracking of soybeans. This system significantly improves the efficiency, safety, and reliability of soybean traceability, utilizing a decentralized file system where all transactions are securely recorded and stored in the blockchain's distributed ledger. However, the Ethereum network's reliance on public consensus may raise privacy concerns for sensitive data in the agricultural industry. It is crucial to address these drawbacks and explore potential solutions.

Vangala et al. [1] suggested A comprehensive literature focuses on the integration of blockchain and IoT in agriculture to ensure secure sensing. The paper explores the enhancement of information security for sensing devices in smart farming by leveraging blockchain technology. It delves into the identification and mitigation of security threats specific to smart farming, discusses how blockchain technology can enhance data security, and addresses the drawbacks associated with implementing blockchain-based IoT in the context of smart farming.

Patil et al. [33] suggested that a framework for lightweight blockchain based secure smart greenhouse farming. The architecture consists of four components (such as smart greenhouse, overlay network, cloud storage and end user). They proposed security framework in physical layer, communication layer, database layer and interface layer to mitigate the security and privacy issues of IoT nodes. While, they didn't point out the throughput and latency challenges in blockchain network.

Kafhali et al. [34] suggested that an architecture to manage the IoT data using blockchain and fog computing to mitigate internet of things (IOT) challenges with regards to multiple accessibility of devices. The fog computing layer sensitive data can be accessed locally instead of sending to the cloud layer. The edge node keep track and control the IoT devices that collect, analysis and store the data. The Software Define Network (SDN) and Network Function Virtualizations (NFV) are integrated to the system for resource optimal management.

Friha et al. [35] proposed that robust security framework based on blockchain and SDN for fog computing enabled agriculture internet of things. The proposed security architecture is composed of three main parts: such as, an agricultural IoT data management system, a blockchain-based integrity monitoring scheme, and a virtual switch software

to supports software-defined networking technologies to improve network management.

The contribution and challenges of the article related to the proposed framework are presented in Table 2.1. The table provides a summary of the key findings and obstacles discussed in the article.

Table 2.1: Related Work Summary

No.	Author	Method Used	Contribution	Challenges
1.	Hu et al. [32]	Ethereum based blockchain integrated with Edge Computing	Trust management and Ensure correctness and reliability of data	Data privacy exposure
2.	Hang et al. [31]	Permissioned based blockchain for secure fish farming	Overcome the privacy data exposure and ensure authorization and authentication	Computational overhead
3.	Mohamed Torkey and A.E Hassanein [30]	Survey on the challenges and issues Blockchain-based IoT	Proposed novel blockchain model	power consumption,throughput and latency, privacy and security
4.	Vangala et al. [1]	Comprehensive literature on smart secure sensing for blockchain based IoT in agriculture.	dentify security requirement and security threats of IoT nodes Designed blockchain based security protocol in smart agriculture	Data security and privacy
5.	Patil et al. [33]	Lightweight blockchain based secure smart greenhouse farming	Mitigate the security and privacy issues of IoT nodes	Throughput and Latency

2.8 Summary

The proposed system has been designed by using private or permissioned blockchain in order to mitigate the latency and throughput challenges of public or permission less blockchain. The private blockchain is designed for private organization and the users have to be authenticated and authorized by the organization admin to access the blockchain. So, in smart farming there are different wireless sensors to automate the farming land and the sensors data transaction takes place by using blockchain technology.

Chapter 3

Research Methodology

3.1 Overview

In this section, we presented an overview of the proposed architecture, including its key components and design principles. We highlighted the blockchain network and discussed the transaction workflow and performance metrics. The aim was to provide a comprehensive understanding of the system's structure and functionality, demonstrating the effectiveness of the proposed approach in smart farming.

3.2 The Proposed Framework Overview

The proposed framework for smart farming consists of four components as shown in Figure 3.1 below: farming sensor nodes, fog computing layer, cloud computing layer, and end users. The farming sensor nodes collect real-time data on farming parameters. The fog computing layer, deployed at the network edge, hosts the blockchain network for secure data storage and transaction management. The cloud computing layer provides scalable resources for data storage and advanced analytics. End users, such as farmers and agricultural experts, can access the system's data and analytics to make informed decisions. This integrated framework enables efficient data collection, local processing, secure storage, and advanced analytics for improved smart farming practices.

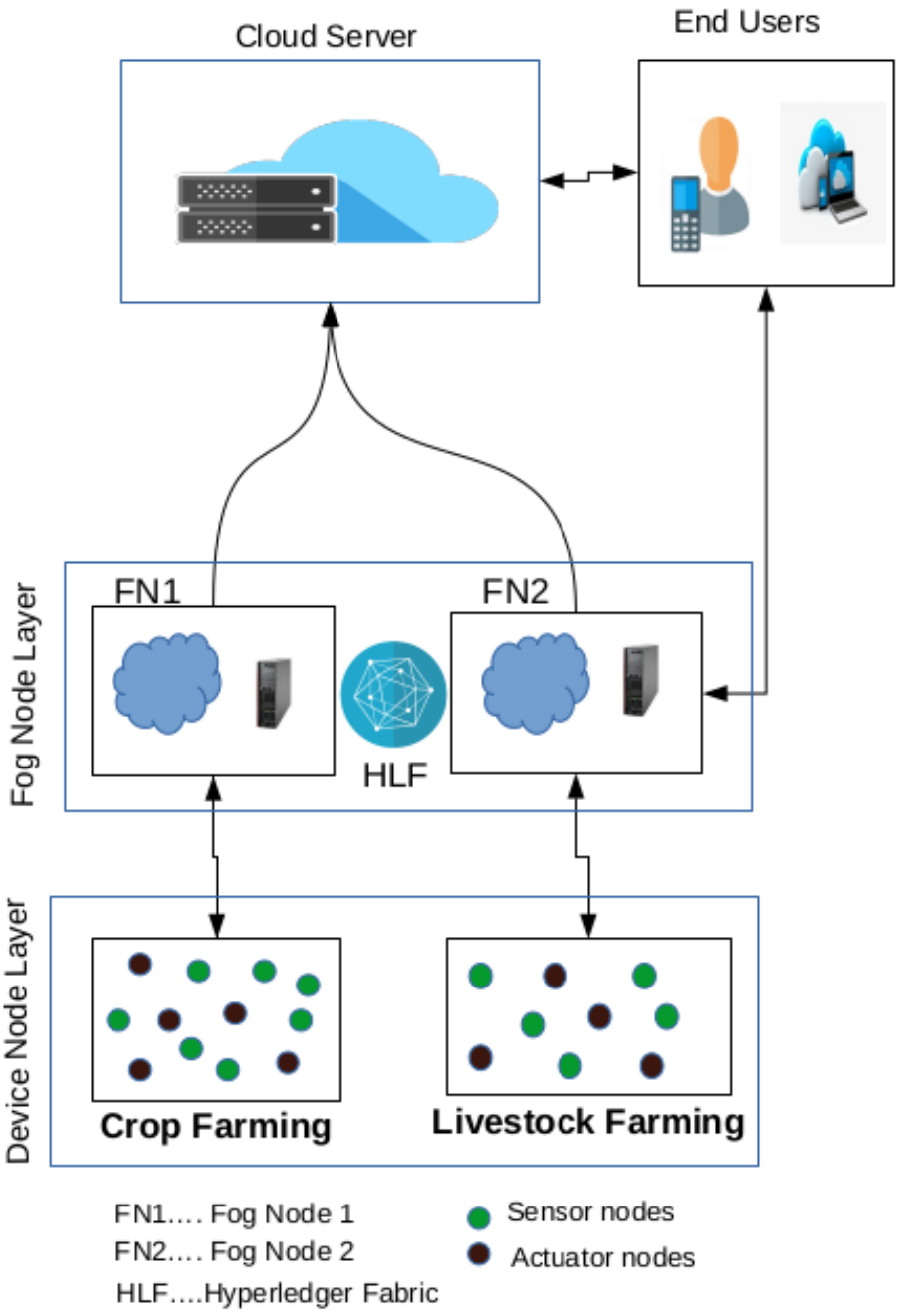


Figure 3.1: A proposed framework

3.2.1 Farming Device nodes

Farming device nodes are an important component of the proposed framework for blockchain-based IoT in smart farming. These nodes are designed to automate farming operations and optimize crop and livestock yields. There are two types of device nodes - one for crop farming and the other for livestock farming. The crop farming device nodes can be equipped with a variety of sensors and actuators that can help optimize irrigation, fertilization, ventilation, and heating systems. For example, sensors for measuring soil moisture, temperature, and nutrient levels can be used to automate irrigation and nutrient delivery, while actuators such as water pumps and valves can be used to control irrigation systems. Similarly, sensors for measuring air temperature and humidity can help optimize ventilation and heating systems, while actuators such as fans and heaters can be used to control these systems.

The livestock farming device nodes can also be equipped with sensors and actuators that can help optimize ventilation, feeding, and watering systems. For example, sensors for monitoring water and feed consumption, as well as animal behavior, can provide insights into the health and well-being of the livestock. Actuators such as feeders and water dispensers can be used to automate feeding and watering systems.

In the proposed system for blockchain-based IoT in smart farming, MQTT is used as the communication protocol between the sensor nodes and the fog nodes. This protocol is designed to be lightweight and efficient, making it an ideal choice for IoT devices with limited resources. MQTT (Message Queuing Telemetry Transport) is a lightweight publish/subscribe messaging protocol that is designed for IoT devices with low bandwidth and power consumption requirements. The use of MQTT enables efficient and reliable communication between the sensor nodes and the fog nodes, with low overhead and minimal latency. Secure communication is enabled in the MQTT protocol by leveraging Transport Layer Security (TLS). and user authentication, ensuring that only registered and authorized devices can communicate with the fog nodes.

For security purposes, each sensor node is registered in the HLF blockchain network. This ensures that only registered sensor nodes can send sensor data to the fog nodes, and helps prevent unauthorized access to the network. The RBAC (Role-Based Access Control) is also used to manage access to the data stored on the HLF blockchain network, ensuring that only authorized users can access and modify the data.

By integrating these sensors and actuators with the fog nodes and HLF blockchain network, you can create a real-time monitoring and control system for your crops and livestock. The fog nodes can analyze the data generated by the device nodes and store it on the HLF blockchain network, ensuring that the data is secure, transparent, and immutable. This can help with compliance and auditing requirements, and also enable end users to access the ledger data to get updated sensor data.

Various types of sensors are used in precision agriculture to measure different parameters. Temperature sensors can measure soil, plant, and air temperature, which are important for controlling soil processes, plant growth, and root growth, as well as for predicting crop yield. Humidity sensors are used to measure humidity and air temperature, which can affect water regulation and gas exchange during plant photosynthesis. Soil moisture sensors are responsible for measuring the current moisture level present in the soil, playing a crucial role in determining the optimal timing for irrigation. Soil pH sensors gauge soil acidity levels, which impact nutrient availability, microbial presence, and ultimately, plant growth. Nano-biosensors have immense potential in the agricultural field and can accurately detect various fertilizers, pesticides, and soil quality. Wind speed sensors are crucial in smart agriculture for assessing wind speed and direction. Specialized monitoring devices, such as raindrop sensors, are needed to count raindrops due to unpredictable heavy rainfall caused by rapid climate changes. Droplet sensors can be used for the precise application of plant protection chemicals to avoid over-usage of pesticides, which can damage the crop and ecosystem.

3.2.2 Fog Computing layer

In 2006, Cisco introduced Fog computing to address the challenges of availability, latency, cost, and energy consumption that come with IoT design [36]. Fog computing is composed of numerous fog nodes, small, lightweight devices that offer computing, networking, and storage resources to complete tasks that end devices are unable to accomplish. Real-time analysis and latency-sensitive applications are run at this layer, and not all of the data gathered is transferred to the cloud for processing [35].

Fog nodes are responsible for managing devices, doing analytics, visualizing data, and communicating with edge devices and cloud layers for long-term storage or more intensive processing. The HLF blockchain network and smart contract are deployed in

this layer, making it feasible to access and store various sensor transactions. The fog nodes use Wi-Fi, Bluetooth, or Zigbee technology for connectivity, and the communication protocol between the fog and cloud layers is MQTT (Message Queuing Telemetry Transport). In scenarios where a nearby fog infrastructure can execute tasks more efficiently and with reduced power consumption compared to sending them to the cloud layer, fog-to-fog communication becomes feasible.

In our proposed architecture, we introduce the concept of dedicated fog servers acting as fog nodes to optimize the accessibility and storage of sensor transactions within the blockchain network. The Blockchain Client, a software component running on the fog node, takes charge of creating transactions, organizing them into batches, and delivering them to the validator for grouping into blocks and eventual commitment to the blockchain. By incorporating the fog node layer, we leverage the benefits of fog computing, which offers edge-based computing and storage resources. This approach reduces latency, minimizes energy consumption, and enhances the overall efficiency of the system.

3.2.3 Cloud Computing Layer

The cloud computing layer in our proposed framework is responsible for long-term storage, intensive processing, and data analysis. This layer is composed of cloud nodes, which offer a scalable and flexible infrastructure to store and process large amounts of data generated by the IoT devices. In this layer, we have deployed a global HLF blockchain network, which stores the complete transaction history in a secure and immutable way. The cloud nodes are distributed across different geographical locations, which provides redundancy and mitigates the risk of single point failure. The cloud layer communicates with the fog node layer and edge devices for data processing and storage. The cloud nodes are connected through high-speed internet and can be easily scaled up or down based on the demand of the system. The cloud computing layer plays a crucial role in providing reliable, scalable, and secure storage and processing capabilities for the proposed framework.

3.2.4 End Users

The end users layer in the proposed framework consists of four main actors: farm admin, farmers, agronomist, and veterinarian. These actors have different roles in the

smart farming ecosystem and are responsible for different aspects of the farming process. To access the data stored in the fog nodes and cloud server, the end users must first register to the HLF blockchain network. The registration process involves generating a public key and a private key using elliptic curve cryptography (ECC). The Fabric client provides the registered users with a signed certificate and stores their identity on their wallet.

Once enrolled, the end users can retrieve data from the fog nodes and cloud server. When they want real-time information about their farming field, they can directly access the fog node. In the HLF blockchain network, the end users can view and verify transactions related to their activities. The data in the blockchain network is immutable, ensuring that the end users can trust the data they retrieve. In addition, the end users can use smart contracts to automate their activities and reduce the need for manual intervention. For example, farmers may use smart contracts to manage their crop cycles, while veterinarians may use them to monitor animal health.

Overall, the end users layer plays a crucial role in the proposed smart farming framework by enabling stakeholders to access and verify data in a secure and efficient manner.

3.3 Blockchain Network

The proposed system utilizes HLF technology to establish a private blockchain network. Unlike public blockchains that involve resource-intensive mining, this private blockchain is optimized for less computationally demanding operations. As a result, it is well-suited for deployment in fog and cloud nodes with limited computing resources. The fog nodes in the network perform resource-intensive functions, such as producing and propagating blocks and monitoring transactions, as full blockchain nodes.

To ensure the security and integrity of the blockchain network, registration of peers and nodes is done using a Certificate Authority (CA). This ensures that only authorized peers and nodes can participate in the network and that all communications between them are encrypted and secure. In terms of consensus algorithm, the proposed system uses the RAFT algorithm, which is a lightweight consensus algorithm that enables no proof-of-work and resource limits mining. This makes it easier to deploy the blockchain network while still maintaining a high level of security and efficiency.

3.3.1 Fabric Transaction Workflow

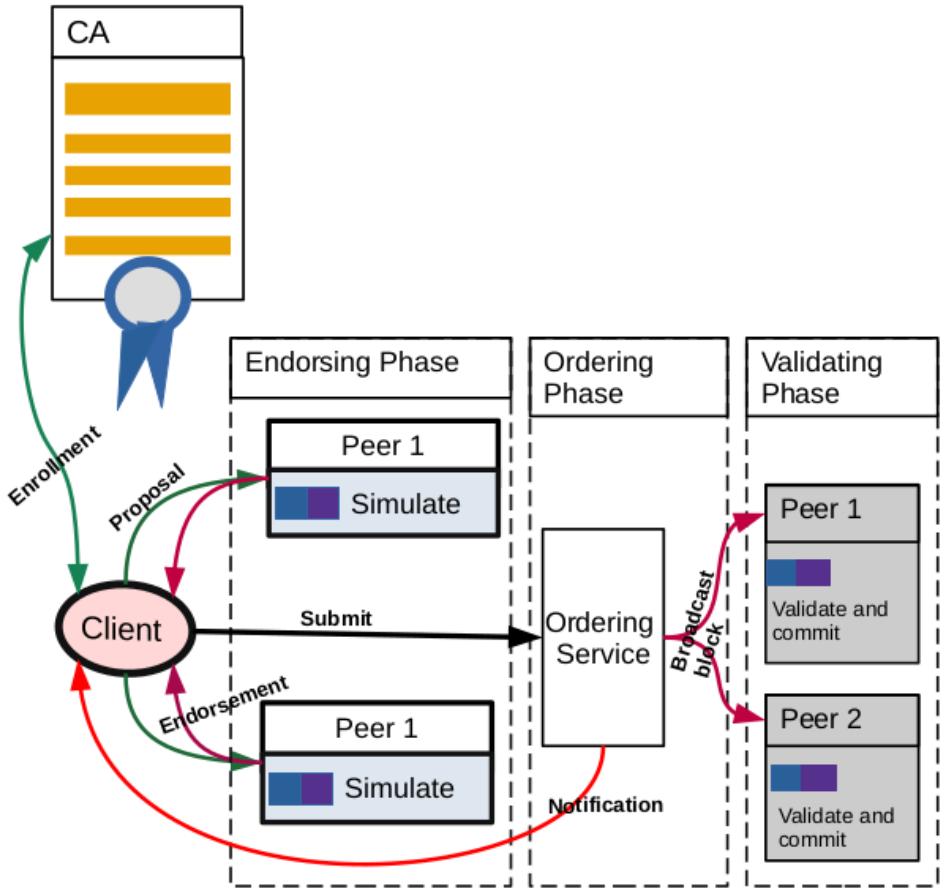


Figure 3.2: Transaction workflow of hyperledger Fabric

Certification Phase

In this phase, new clients such as sensor nodes, actuators, fog nodes, cloud nodes and end users registered and request digital Membership Service provider(MSP). The certificate authority check it out their device id, then issue signed digital certificate to the new coming nodes or devices.

Endorsing Phase

In endorsing phase, the client propose transaction to the endorser fog nodes, the endorser fog nodes check it out the transaction is new or the transaction is proposed previously. After that, the endorser nodes send signed messages to the client. When the client receive transaction signed message, the signed transaction is broadcast to the orderer.

The transaction proposal (or, simply, proposal) is signed by clients and sent to one or more endorsers for execution during the execution phase. Remember that each chaincode's endorsement policy implicitly specifies a set of endorsers. A proposal includes the identity of the submitting client, the transaction payload in the form of an operation to perform, parameters, and the identifier of the chaincode, a nonce that can only be used once by each client (like a counter or random value), and a transaction identifier derived from the client identifier and the nonce[24]. An endorser simulates a proposal by performing a operation on the specified chaincode installed on the block chain. The chaincode runs in a Docker container isolated from the main endorser process.

The endorsement process entails conducting a simulation of a proposal using the local blockchain state of the endorsing peer, independent of synchronization with other peers. It is important to note that the simulation results are not permanently stored in the ledger by end-users. The creation of a chaincode's state is limited to that specific chaincode and is not directly accessible by other chaincodes. The primary responsibility of the chaincode is to manage and update the state of the blockchain, utilizing operations such as GetState, PutState, and DelState. The chaincode does not store the local state within the program code itself. If necessary, a chaincode can invoke another chaincode with appropriate permissions to access its state within the same channel. During the proposal simulation, each endorsing peer generates a readset that records the version dependencies, including the keys and their corresponding version numbers. Additionally, a writeset is created to capture the state updates resulting from the simulation, including the updated keys and their new values. Following the simulation, the endorsing peer provides the client with a proposal response that includes an endorsement message. This message contains the readset, writeset, along with relevant metadata such as the transaction ID, endorser ID, and a cryptographic signature from the endorsing peer.

Ordering Phase

In this phase, when a client has gathered sufficient endorsements on a proposal, it constructs a transaction and submits it to the ordering service. The transaction includes a set of endorsements, transaction information, and transaction payload, which consists of the chaincode operation and parameters. During the ordering phase, a total order is established for all submitted transactions within each channel. In other words, ordering service broadcasts endorsements and achieves consensus on transactions, even in the

presence of faulty orderers. Furthermore, the ordering service generates a hash-chained sequence of blocks containing transactions, achieved by batching multiple transactions into groups or blocks. This batching technique enhances the throughput of the broadcast protocol, which is a well-known approach employed in fault-tolerant broadcasts[24].

The ordering service's interface, at its most basic level, only supports the following two peer-invoked operations that are implicitly parameterized by channel identifiers:

- `broadcast(tx)`: A client calls this operation to propagate a transaction of their choice. `tx`, which usually contains the transaction payload and a signature of the client, for dissemination.
- `B ← deliver(s)`: A client calls this to retrieve block `B` with non-negative sequence number `s`. The block contains a list of transactions `[tx1, ..., txk]` and a hash-chain value `h` representing the block with sequence number `s−1`, i.e., `B = ([tx1, ..., tx2], h)`. As the client may call this multiple times and always returns the same block once it is available, we say the peer delivers block `B` with sequence number `s` when it receives `B` for the first time upon invoking `deliver(s)`.

Although the blockchain network may have a large number of peers, only a small subset of nodes are expected to implement the ordering function. In our proposed system, we have utilized the Raft consensus algorithm, which provides a built-in gossip service for disseminating delivered blocks from the ordering service to all peers. The implementation of gossip is scalable and independent of the specific ordering service, ensuring compatibility with the Raft consensus algorithm. This approach enhances the modularity of our system. Additionally, the ordering service performs access control checks to determine if a client is authorized to broadcast messages or receive blocks on a specific channel.

Validation Phase

Blocks are delivered to peers either directly by the ordering service or through gossip. A new block then enters the validation phase which consists of three sequential steps:

1. The endorsement policy evaluation occurs in parallel for all transactions within the block. The task of evaluating the endorsement is carried out by the so-called validation system chaincode (VSCC), a static library that is a component of the

blockchain's configuration and is in responsible of assessing the endorsement with respect to the endorsement policy configured for the chaincode.If the endorsement is not satisfied, the transaction is designated as invalid, and its results are not taken into consideration.

2. A read-write conflict check is done for all transactions in the block sequentially. It checks to make that the versions of the keys in the readset field are still the same for each transaction by comparing them to the current state of the ledger, which is held locally by the peer. The trans-action is marked as invalid and its effects are ignored if the versions do not match.

3. The ledger update phase runs last, wherein the block is added to the locally stored ledger and the blockchain state is updated. In particular, while adding the block to the ledger, the outcomes of the validity checks in the first two steps are also maintained in the form of a bit mask identifying the transactions that are valid within the block. The state can now be more easily rebuilt in the future.Furthermore, all state updates are applied by writing all key-value pairs in writeset to the local state.

The default VSCC in Fabric allows monotone logical expressions over the set of endorsers configured for a chaincode to be expressed. The VSCC evaluation verifies that the set of peers, as expressed through valid signatures on endorsements of the transaction, satisfy the expression.

3.3.2 Blockchain consensus mechanism

Our proposed architecture for a blockchain-based IoT system in smart farming utilizes the Raft algorithm as the consensus mechanism. It is well-suited for use in permissioned blockchains where the participants are known and trusted. The Raft algorithm ensures that all transactions recorded on the blockchain are validated and agreed upon by all nodes in the network. In comparison to public permissionless blockchains, our proposed system utilizes a permissioned blockchain with the Raft consensus algorithm to enhance verification and security. Deploying the 'fabfarm' chaincode provides the advantage of restricting bad actors from participating in transactions and queries. They are unable to write, read, or commit transaction information, thus improving the system's integrity and security.

The utilization of the RAFT consensus algorithm ensures that transaction verification and validation are conducted solely by orderers who possess physical membership within the consortium. This approach significantly enhances the efficiency, speed, and scalability of the proposed blockchain platform, enabling it to handle a substantial number of confirmed transactions per second across multiple ledgers. Furthermore, it results in reduced transaction costs and energy consumption, improved scalability, and the achievement of fault-tolerant, attack-resistant, and collusion-resistant transactions.

The Raft algorithm elects a leader among the nodes in the system who manages the consensus process. The process of electing a leader occurs through a series of steps where each node sends a RequestVote message to all other nodes. Once a candidate receives a majority of votes from the nodes in the network, it becomes the leader responsible for managing the replication of the log containing all transactions added to the blockchain. The leader communicates with other nodes to ensure they all have a copy of the log and are in agreement about the state of the blockchain.

The Raft algorithm, known for its ease of understanding and implementation, high fault-tolerance, and scalability, proves to be an ideal choice for our proposed system. By utilizing the Raft consensus mechanism, we establish a transparent, secure, and trusted platform that facilitates the seamless exchange of services specifically designed for monitoring crops and livestock.

3.3.3 Data Security and Privacy

In the proposed system for smart farming, a data-centric approach to information security was implemented to address the issue of data security and privacy[37]. This was achieved by combining public and private key cryptography with RBAC (Role-Based Access Control) and ECDSA-SHA-256 (Elliptic Curve Digital Signature Algorithm - Secure Hash Algorithm 256) to provide a robust foundation for data security.

Traditionally, data security has been controlled through application front-end and middle tier security logic, which can create inefficiencies, redundancy, and potential security vulnerabilities. Replicating data into various data lakes can further increase the risk of data exposure, as it creates another potential attack surface for hackers. To defend the data itself at the data layer, the permissions for access and modification of data were stored as codified data elements. This data-centric security approach ensured that the

inherent permissions were carried along with the root data, even when the data was accessed or moved to another location. This approach was particularly useful for shared datasets that were openly accessible by stakeholders, each with varying degrees of access permission.

RBAC was used to control data access based on roles and responsibilities. This ensured that only authorized users could access data, and they could only access the data that was relevant to their role. ECDSA-SHA-256 was used to secure the data and prevent tampering. With this cryptographic technique, the data was hashed and digitally signed using elliptic curve cryptography. This ensured that the data was secure and tamper-proof, as any changes made to the data would be detected through verification of the digital signature.

Finally, the data was stored on a permissioned private blockchain, which further enhanced data security and privacy. The blockchain provided a tamper-proof, immutable, and decentralized database that could be accessed by authorized stakeholders with varying degrees of access permissions. The use of a private blockchain ensured that the data was only accessible to authorized stakeholders and was not publicly available.

In summary, the proposed system for smart farming used a data-centric approach to information security, combining RBAC and ECDSA-SHA-256 to provide a robust foundation for data security. By storing data on a permissioned private blockchain, data security and privacy were further enhanced. This system ensured that only authorized stakeholders could access and modify data, and that the data was tamper-proof and immutable.

ECDSA-SHA-256

ECDSA-SHA-256 is a digital signature algorithm that combines the Elliptic Curve Digital Signature Algorithm (ECDSA) and the Secure Hash Algorithm 256 (SHA-256) to offer secure and efficient data signing. ECDSA-SHA-256 generates a pair of keys, namely a private key and a public key, which can be utilized for message signing and verification purposes.

The ECDSA-SHA-256 algorithm comprises multiple stages, including key generation, signature generation, and signature verification. In the key generation stage, a random integer is chosen, and a pair of public and private keys is calculated. In the signature

generation stage, another random integer is selected, and a signature for the message is generated using the sender's private key. In the signature verification stage, the recipient computes a signature for the message using the sender's public key and verifies if the computed signature matches the received signature. These steps ensure the integrity and authenticity of the message, providing a secure method for verifying the sender's identity and the integrity of the transmitted data.

The EDSA-SHA-256 algorithm involves key generation, message signing, and signature verification to ensure message integrity and authenticity[38].

ECDSA key generation: To generate key users follow this steps.

1. Select a random integer d from the interval $[2, n-2]$
2. Compute $Q = d \cdot P$.
3. he public and private keys of user A are (E, P, n, Q) and d_t respectively.

ECDSA Signature Generation : User A signs the message m using the following steps.

1. User A selects a random integer k from the interval $[2, n-2]$.
2. Compute $k \cdot P = (x_1, y_1)$, and let $r = x_1 \bmod n$.
   - If x_1 is an element of the binary field $\mathbb{F}_{2^k}$, it is assumed that x_1 is represented as a binary number.
   - If $r = 0$, go back to Step 1 and select a new random integer k.
3. Compute $k^{-1} \pmod n$.
4. Compute $s \equiv k^{-1} \cdot (H(m) + d \cdot r) \pmod n$.
   - Here, H is the secure hash algorithm SHA -256.
   - If $s = 0$, go back to Step 1 and select a new random integer k.
5. The ECDSA signature for the message m is represented by the pair of integers (r, s).

ECDSA Signature Verification: User B verifies User A's ECDSA signature (r, s) on the message m by applying the following steps:

1. Compute $c = s^{-1} \bmod n$ and $H(m)$.
2. Compute $u_1 = H(m) \cdot c \bmod n$ and $u_2 = r \cdot c \bmod n$.
3. Compute $u_1 \cdot P + u_2 \cdot Q = (x_0, y_0)$ and let $v = x_0 \bmod n$.
4. Accept the signature if $v = r$.

3.4 Performance Evaluation Metrics

The primary objective of any deployed blockchain applications is to maintain submitted transactions by network participants, transaction verification and ordering processes, block generation, and store the transaction outcome in a distributed ledger. Therefore, the blockchain system performance can be evaluated with the following performance metrics:

- **Throughput:** The maximum number of transactions that the blockchain system can handle, and record the ledger's transaction outcomes in a given time.
- **Latency:** The time between the transaction invoking by a client and writing the transaction to the ledger.
- **Computational Resources:** Hardware and network infrastructure required for the blockchain operation.

3.4.1 Transaction Throughput

Deployment, execution, and invoking of smart contracts in different blockchain systems occur at different speeds. It is needed therefore to monitor the transaction throughput[39]. It is measured as the rate of committing valid transactions by the HLF network in a defined period. The formal mathematical description of the transaction throughput can be obtained as:

$$TPS_i = \frac{Count(T_x in(T_s, T_e))}{T_e - T_s} \tag{3.1}$$

where, $\mathbf{T_x}$ is the total number of submitted transactions, $\mathbf{T_e}$ is the last block commit time, and $\mathbf{T_s}$ is the initial transaction submission time. The transaction throughput of N peers is calculated by taking the median:

$$\overline{TPS} = \frac{\sum_i TPS_i}{N} \tag{3.2}$$

3.4.2 Transaction Latency

When the transaction is sent to the network, it takes some time to be confirmed by the system[39]. Transaction latency is the amount of time taken from the point the transaction is submitted to the point when the transaction is confirmed and committed with

the result being available across the network. This metric is measured per transaction. However, in most cases, the experiment provides various statistics on overall transactions such as high, average, low, and standard deviations. During a period started at $\mathbf{T_s}$ and ended at $\mathbf{T_e}$, the transaction sent to the peer is shown by $\mathbf{Tx_input}$, and $\mathbf{Tx_confirmed}$. The average latency of the peer i can be computed using the following equation:

$$AL_i = \frac{\sum_{T\,x}(t_{Txconfirmed} - t_{Txinput})}{Count(T_xin(T_s, T_e))} \tag{3.3}$$

The latency of all smart contracts is calculated by taking the median:

$$\overline{AL} = \frac{\sum_i AL_i}{N} \tag{3.4}$$

Chapter 4

Results and Discussion

4.1 Overview

This chapter presents a detailed account of the implementation of HLF, the experiments conducted to evaluate the performance of the proposed architecture, and a brief discussion of each result. The chapter also describes the range of tests performed and the hardware, software, and tools used in the experiments.

4.2 Hyperledger Fabric Implementation

In this section, we covered the HLF implementation process in a step-by-step manner. This included setting up the development environment, developing the chaincode, starting up the Fabric network with CA and CouchDB, creating and joining a channel, deploying the chaincode, enrolling user identities, invoking and querying the chaincode, and testing and monitoring the network.

4.2.1 Development Environment Setting Up

To set up the development environment, we installed and configured the necessary software tools, including Go programming language version go1.19.4 linux/amd64, Docker version 20.10.21, docker-compose version 1.25.0, git version 2.25.1, node.js version 18.15.0, and curl version 7.68.0.

Go is a statically typed, compiled programming language that is used to build efficient and reliable software. Docker is a containerization platform that allows us to package our applications into containers, making them easy to deploy and run across different environments. Docker-compose is a tool for defining and running multi-container Docker applications. Git is a version control system that allows us to keep track of changes to our code and collaborate with others. Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It is used for building scalable network applications. Curl is a command-line tool used for transferring data from or to a server.

Using Docker and Docker-compose allows us to create a containerized environment for our Hyperledger Fabric network, which makes it easy to deploy and manage the network components. We also used Git to version control our code and collaborate with

others. Node.js was used to run JavaScript-based scripts, and Curl was used to interact with the network through command-line requests. After installing these tools, we then installed Hyperledger Fabric version 2.4.7, which provides the necessary libraries and tools to develop and run blockchain applications.

4.2.2 Chaincode Development

In the chaincode development phase, we used the Go programming language to design a chaincode with six different functions for write and read transactions to the ledger. The functions included creating crop data, creating livestock data, updating crop sensor data, updating livestock sensor data, querying crop data, and querying livestock data. The chaincode was designed to handle the business logic of the application and was meant to be deployed on the Hyperledger Fabric network. Once the chaincode was designed, it was ready for deployment on the network.

4.2.3 Fabric Network Start Up

During the Fabric network start-up phase, we initiated the Fabric network using the network.sh up command, which also started the CouchDB and CA applications. The purpose of using CouchDB is to store the state of the blockchain network, and it offers a rich query language and easy data synchronization, which is beneficial for blockchain-based applications. Additionally, CouchDB provides a scalable and fault-tolerant data store that can handle large amounts of data.

During the network initialization phase, we also launched the Certificate Authority (CA) application. The CA application serves as a key component for managing the identities of network participants, ensuring that only authorized individuals can actively participate in the network. It facilitates the issuance and revocation of security certificates, while verifying and authenticating the identities of network participants. Leveraging the CA application provides a secure and scalable approach to effectively manage the identities of network participants, bolstering the overall security and trustworthiness of the network.

During the network initialization process, we completed the creation and joining of a channel. This channel plays a vital role in partitioning the blockchain network into separate sub-networks, enabling the establishment of private channels that are exclusively

visible to authorized participants. This essential feature fosters a secure and confidential communication environment among participants, ensuring that transactions are processed and validated solely by those who possess the necessary authorization.

4.2.4 Chaincode Deployment

n the deployment phase of the chaincode, we successfully implemented the "fabfarm" chaincode, which was specifically designed to meet the requirements discussed during its development. The deployment process involved packaging the "fabfarm" chaincode into a deployable package and installing it on the peer nodes. We utilized the "peer" command-line tool to install and instantiate the chaincode, specifying the endorsing peer organizations, channel name, as well as the chaincode name and version.

Deploying the "fabfarm" chaincode offers significant advantages. It allows authorized participants to interact with the blockchain network, invoking the chaincode's functions related to crop and livestock monitoring. This capability enables participants to securely write new data to the blockchain and retrieve existing data, ensuring the integrity, transparency, and immutability of the recorded information.

4.2.5 Users Enrollment and Registration

Establishing a Hyperledger Fabric network involves an important stage called user enrollment and registration. During this process, digital identities are created for network participants, granting them the necessary access and permissions. The initial step is enrolling the network administrator through the Hyperledger Fabric CA client. This entails generating a private key and a certificate, both signed by the Fabric CA, to authenticate the administrator within the network. These credentials serve as the basis for securely managing user identities and their interactions within the network.

Once the administrator is enrolled, they have the ability to register additional participants in the network, including farmers, agronomists, and veterinarians. This process involves creating a distinct identity for each participant, generating a private key and certificate for their authentication, and associating their identity with the relevant organization within the network.

During the registration process, the administrator can also assign specific roles and permissions to each participant, such as read or write access to the ledger or access to

specific chaincode functions. This ensures that only authorized participants can interact with the network and access sensitive data.The identities of the participants are stored in local wallets, which can be accessed by the Hyperledger Fabric SDK to interact with the network. This allows participants to invoke chaincode functions, query the ledger, and read or write data to the network based on their assigned roles and permissions.

4.2.6 Invoking and Querying

In our chaincode, we have implemented several functions for creating and updating crop and livestock data, as well as updating sensor data for both types of farming. These functions are considered as write or invoke operations on the ledger. We have also implemented read or query functions for retrieving sensor data for crops and livestock.

Farmers have the authorization to access and retrieve information about their own crops and livestock using our system. This enables them to efficiently monitor and manage their agricultural assets. Meanwhile, veterinarians are limited to accessing livestock data solely for the purpose of monitoring their health. This restriction is in place to safeguard the privacy and security of livestock owners. Agronomists, on the other hand, have the responsibility of overseeing and regulating the health and soil conditions of crop farming areas. They have the necessary permissions to retrieve crop farming data, allowing them to analyze the information and offer recommendations for enhancing crop health and productivity.

By carefully managing permissions and access to data, our system ensures that each user is only able to access the data that they need for their specific role within the smart farming ecosystem. This helps to ensure the privacy, security, and integrity of the data while also providing farmers, veterinarians, and agronomists with the tools they need to optimize their farming practices.

4.2.7 Testing and Monitoring the Network

Testing and monitoring the network is crucial during the implementation of Hyperledger Fabric. After setting up the network and deploying the chaincode, it is important to perform comprehensive testing to verify its proper functioning. This includes assessing the functionality of the chaincode, such as invoking and querying operations, as well as evaluating the performance and scalability of the network. Thorough testing

ensures that the network operates as expected, identifies any potential issues or bugs, and allows for timely troubleshooting and optimization. Additionally, ongoing monitoring helps to ensure the network's continued performance, adherence to standards, and reliability. To keep track of the network's performance, Prometheus and Grafana can be set up to provide real-time metrics and insights. These metrics include information like block height, transaction throughput, and peer status, which help identify any performance issues and troubleshoot them. It's important to continuously monitor the network for stability and reliability. By conducting thorough testing and monitoring, stakeholders can ensure that the Hyperledger Fabric implementation is working as intended and achieving its goals.

4.3 Experimental Setup

The experimental setup focused on testing a proposed system for designing a blockchain-based IoT solution in smart farming using Hyperledger Fabric permissioned blockchain. The setup included the following components: a host machine running Windows 11 with an Intel Core i9 processor clocked at 3.6 GHz, 32 GB of RAM, and a 500 GB hard disk. The virtualized environment was created using VirtualBox 7.0.8, within which a single virtual machine was set up with Ubuntu 20.04 as the guest operating system. The virtual machine was allocated 20 GB of RAM and 30 GB of hard disk space.

Within the virtual machine, the Hyperledger Fabric images and binaries were installed to establish a complete Hyperledger Fabric network. The experiment aimed to evaluate the performance of chaincode functions using Caliper benchmarks, an open-source blockchain benchmarking framework.

The performance evaluation comprised three main tests. The first test focused on measuring the system's performance when invoking chaincode functions, assessing throughput (transactions per second) and latency (response time). The second test evaluated the system's performance during data updates through chaincode functions, measuring throughput and latency. Lastly, the third test examined the system's querying capabilities for crop and livestock data, measuring throughput and latency. In this performance evaluation, we focused on the main smart contract functions that are shown in Table 4.1.

The performance evaluation in the experimental setup conducted different configurations of workload. In terms of invoking performance, the system's performance was assessed by adjusting the transaction load and size, measuring the throughput and latency during the execution of chaincode functions. For updating performance, the evaluation focused on varying the transaction load while maintaining a constant transaction size, providing valuable insights into the system's performance during data update operations. Similarly, for querying performance, the evaluation involved adjusting the transaction load while keeping the transaction duration consistent, enabling the measurement of the system's throughput and latency for querying operations. These various workload configurations facilitated a comprehensive evaluation of the system's performance across different transaction loads and sizes, yielding valuable insights into its overall performance. Tables 4.2, 4.3, and 4.4 summarize the settings used for performance evaluation in the write and read operations, respectively.

Table 4.1: Chaincode functions of the system

Main Function	**Operation Type**
createCropData	Write
readCropData	Read
createLivestockData	Write
readLivestockData	Read
updateCropSensorData	Write
updateLivestockSensorData	Write

Table 4.2: The Invoking Performance Evaluation Settings.

Test Number	1 2 3 4
Functions under test	Write operations
work number	5 worker
Transaction number	500 5000 20000 50000
Type of control rate	Fixed-load
transactionLoad(TPS)	tab:writesetting 50 100 150 200

Table 4.3: The Updating Performance Evaluation Settings.

Test Number	1 2 3 4 5 6
Functions under test	Write operations
work number	5 worker
Transaction number	5000
Type of control rate	Fixed-load
transactionLoad(TPS)	50 100 150 200 250 300

Table 4.4: The performance evaluation settings used for the Read operations.

Test Number	1 2 3 4
Functions under test	Read operations
work number	5 workers
Transaction duration	30 seconds
Type of control rate	Fixed-load
Transaction Load (TPS)	1000 2000 3000 4000

4.4 Experimental Result

In the experimental results, we conducted three experiments by varying the transaction load while using rate control with a fixed load. The first experiment focused on the write operation of two functions: invoking crop and livestock data. The experimental settings for this experiment can be found in Table 4.2.

The second experiment aimed to update the crop sensor data and livestock sensor data, with the setup details provided in Table 4.3. The objective was to evaluate the system's performance during data update operations. Lastly, the third experiment involved read livestock and crop information, following the specifications outlined in Table 4.4. These experiments, utilizing rate control with a fixed load, provided valuable insights into the system's performance under different transaction loads.

Experiment 1: In experiment 1, we conducted two tests with different variations. In the first test, we varied the transaction numbers while keeping the transaction load constant at 100 TPS. The transaction numbers were varied within a range specified for the experiment. In the second test, we focused on varying the transaction load while keeping the transaction number fixed at 5000. The transaction load was varied within a speci-

fied range for this particular test. These variations allowed us to evaluate the system's performance under different transaction load and number transactions as described in table 4.1 ,and the experiment setup as mentioned in Table 4.2. The experiment involved collecting the Caliper benchmark performance reports for each test, including throughput and average latency. In Test 1,we conducted experiments by varying the transaction numbers from 500 to 50,000, while keeping a constant transaction load of 100. As demonstrated from the bar chart in Figure 4.1 provides valuable insights into the performance of the system.

For the create Crop operation, the throughput values exhibited an increasing trend, ranging from 86.0 to 156.2 transactions per second (TPS), as the transaction numbers increased. This indicates that the system's ability to process create Crop transactions improved with a larger workload. On the other hand, the latency values remained relatively consistent, ranging from 390 to 500 milliseconds, suggesting stable response times for create Crop transactions across different transaction numbers. Regarding the create Livestock operation, the throughput values displayed slight fluctuations, ranging from 101.8 to 169.5 TPS. This suggests that the system's performance in processing create Livestock transactions may vary slightly with different transaction numbers. Likewise, the latency measurements for generating Livestock transactions remained relatively stable, with values consistently ranging from 390 to 410 milliseconds.

In Test 2, as described in Figure 4.2 the system's performance was evaluated by varying the transaction load while keeping the transaction numbers constant. Four transaction load levels were tested: 50, 100, 150, and 200. The measurements recorded for the create crop data and create livestock data operations include throughput (transactions per second) and latency (time per transaction). For the create Crop operation, the throughput values ranged from 139.1 to 148.1 transactions per second, while the latency values varied from 240 to 840 seconds. As for the create Livestock operation, the throughput values ranged from 150.4 to 165.8 transactions per second, and the latency values ranged from 210 to 780 second

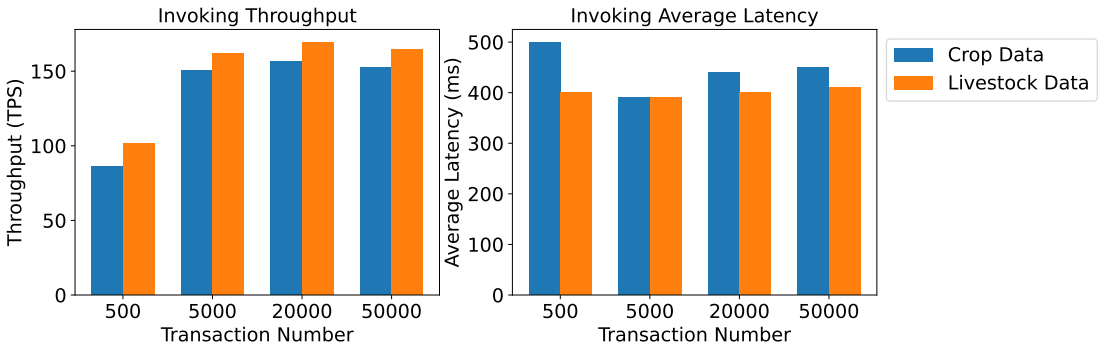


Figure 4.1: Throughput and Average Latency for Invoking by Varying Transaction Number

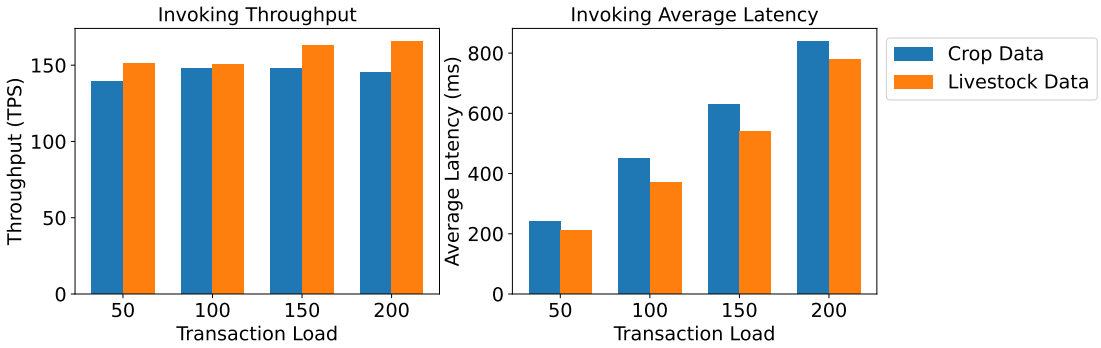


Figure 4.2: Throughput and Average Latency for Invoking by Varying Transaction Load

Experiment 2: In this experiment, the performance of two functions, update Crop Sensor and update Livestock Sensor, was tested by varying the transaction load from 50 to 300 TPS while keeping a fixed-load rate control and a constant transaction duration of 30 seconds. The benchmark was configured based on Table 4.2. In the Experiment 2 for update Crop Sensor, as demostrated from figure 4.3 the system's performance was evaluated by varying the transaction load while keeping the transaction numbers constant. The transaction load values tested were 50, 100, 150, 200, 250, and 300. The measurements recorded for the update crop sensor and update livestock sensor operations include throughput and latency.

For the update crop sensor operation, the throughput values exhibited an increasing trend, ranging from 205.2 to 229.8 TPS, as the transaction load increased. This indicates that the system's ability to process update Crop sensor transactions improved with a larger workload. On the other hand, the latency values showed a slight increase, ranging from 160 to 840 milliseconds. This suggests that the time taken to execute an update crop sensor transaction may slightly increase with higher transaction loads. Similarly, for the updateLivestock operation, the throughput values demonstrated an increasing

trend, ranging from 224.6 to 239.7 TPS, with the increasing transaction load. The latency values ranged from 140 to 750 milliseconds, showing a slight increase as well.

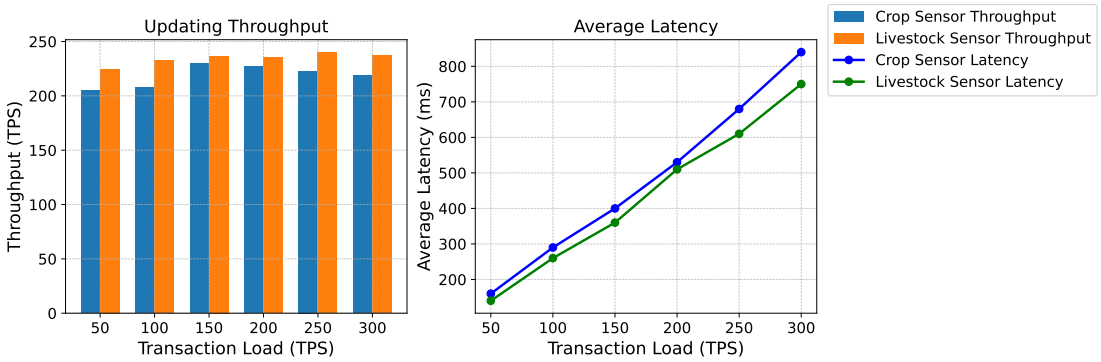


Figure 4.3: Throughput and Average Latency for updating sensor data

Experiment 3: In Experiment 3, the performance of read operations, specifically querying crop data and querying livestock data, was evaluated. The benchmarks were set up according to the configuration described in Table 4.4. As depicted in Figure 4.4, the querying crop operation exhibited high throughput values, ranging from 2479.8 to 2538.5 TPS, across various transaction loads. This indicates efficient processing of a significant number of querying crop data transactions within a fixed duration. The latency values consistently remained low, ranging from 60 to 90 milliseconds, indicating quick response times for read crop transactions. Similarly, for the querying livestock operation, the system maintained high throughput values, ranging from 2395.0 to 2440.1 TPS, for different transaction loads.

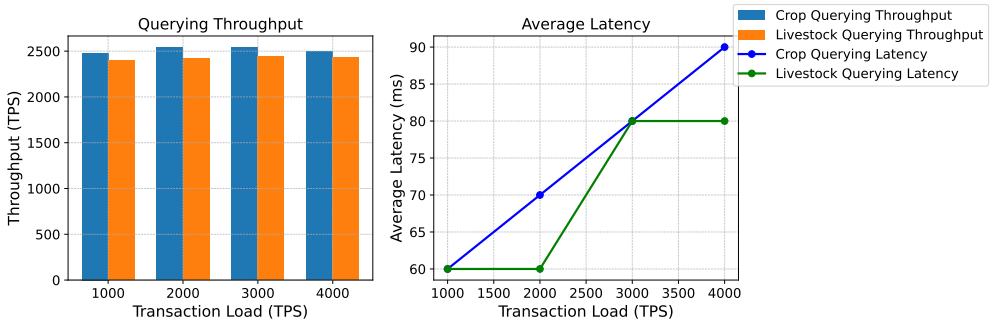


Figure 4.4: Throughput and Average Latency for read crop data

4.5 Discussion of the Results

In a series of experiments using the Caliper benchmarks, we evaluated the performance of the proposed blockchain-based IoT in smart farming. Experiment 1 examined the system's performance by varying transaction numbers while keeping the transaction load constant. It demonstrated increasing throughput values for crop and livestock operations, indicating effective handling of higher transaction volumes. The system maintained consistent response times with relatively stable latency values. Experiment 2 focused on update operations and revealed high throughput values across different transaction loads, shows the efficiency of the system. Although there was a minor increase in latency, the response times remained reasonable. Experiment 3 evaluated read operations, demonstrating high throughput and low latency, which indicated the system's effectiveness in different transaction scenarios and its ability to provide quick information retrieval.

Additionally, Experiment 1 revealed that increasing transaction load resulted in higher average latency and transaction throughput, suggesting increased contention during the endorsement and ordering phases. Similarly, Experiment 2 demonstrated a peak in transaction throughput, indicating improved resource utilization during these phases. In contrast, Experiment 3 showed no significant changes in performance metrics for read-only operations with increasing transaction load. These operations, being less resource-intensive and less prone to contention, enabled the system to handle higher transaction loads without compromising latency or throughput. This highlights the system's ability to process a greater number of read-only transactions without impacting network performance. Response times ranged from 60 to 80 milliseconds.

Regarding security, our system leveraged the security features provided by Hyperledger Fabric, such as permissioned access and channel-based communication. These features ensured that only authorized participants with valid digital certificates could access and contribute to the network, enhancing the overall security of the system. Additionally, we implemented a RBAC mechanism, assigning specific roles and permissions to farming actors. This restricted access to data and functionalities based on predefined roles, minimizing the risk of unauthorized access or data breaches.

Moreover, Hyperledger Fabric's support for private channels and private data collec-

tions ensures the confidentiality and privacy of sensitive data exchanged within the system. By selectively sharing data among specific participants, the system minimizes the risk of unauthorized access or data breaches. The combination of the security features provided by Hyperledger Fabric and the incorporation of RBAC establishes a robust security framework for our blockchain-based IoT system. This framework guarantees the integrity and confidentiality of the data exchanged between the device node layer, fog node layer, cloud server, and end users, minimizing the risk of unauthorized access or data breaches.

In summary, the performance outcomes indicate the effectiveness and scalability of our suggested IoT system, while the security considerations emphasize the built-in security capabilities of Hyperledger Fabric and the added layer of RBAC. Collectively, these factors enhance the overall efficiency and security of our blockchain-based IoT system, facilitating dependable and secure data exchange across distributed layers.

Chapter 5

Conclusion and Recommendation

In this chapter, we present the conclusion and recommendations based on our study on designing a blockchain-based IoT system for smart farming using Hyperledger Fabric technology.

5.1 Conclusion

This study presents a novel blockchain-based IoT system designed for smart farming, leveraging the capabilities of Hyperledger Fabric technology. The system architecture comprises four layers: device node layer, fog node layer, cloud server, and end users. Through rigorous experimentation using Caliper benchmarks, the system's performance was evaluated. The results showcased impressive performance with satisfactory to high levels of throughput and consistently low latency values. The system demonstrated its ability to handle a significant number of transactions efficiently while maintaining quick response times. This blockchain-based IoT system offers a promising solution for enhancing the operational efficiency of smart farming.

The integration of Hyperledger Fabric technology within the system brings forth a robust and secure framework that ensures authorized access and protects sensitive data and transactions. Leveraging channel-based communication, the system ensures the confidentiality of transactions, which is crucial in safeguarding valuable agricultural information. Furthermore, the implementation of Role-Based Access Control (RBAC) adds an extra layer of security by assigning specific roles and permissions to farming actors, preventing unauthorized actions and ensuring a trustworthy ecosystem.

To conclude, the suggested blockchain-driven IoT system for smart agriculture has demonstrated satisfactory performance in terms of data throughput and response time. By incorporating Hyperledger Fabric technology and deploying Role-Based Access Control (RBAC), the system has established a secure and efficient environment for farming operations. These findings underscore the capacity of blockchain technology to augment the efficiency and security of IoT systems in the context of smart farming, effectively addressing the distinct requirements and obstacles encountered in the smart farming.

5.2 Recommendation

In terms of future work, there are several areas where researchers can contribute to the design and security of blockchain-based IoT systems using Hyperledger Fabric. One important aspect is the development of intrusion detection and malicious activity detection mechanisms. By leveraging deep learning and machine learning algorithms, researchers can create sophisticated systems that can identify unauthorized access attempts and detect malicious activities within the system. This proactive approach enables early detection and response to potential security breaches. Another area of focus is the mitigation and detection of distributed denial-of-service (DDoS) and Sybil attacks. Researchers can explore innovative techniques to identify and counter these types of attacks in real-time. By developing robust methods for detecting and responding to such attacks, the overall resilience and security of the system can be improved.

Additionally, the integration of distributed file systems, such as the InterPlanetary File System (IPFS), with Hyperledger Fabric can offer significant benefits. This integration allows for decentralized and efficient storage of large data files, ensuring data availability, resilience, and integrity within the blockchain network. Exploring the advantages and challenges of this integration can lead to more scalable and reliable blockchain-based IoT systems.

By addressing these areas of future work, researchers can make valuable contributions to the advancement and security enhancement of blockchain-based IoT systems. Their efforts in developing intrusion detection mechanisms, mitigating attacks, and integrating distributed file systems will lead to more secure and efficient systems, benefiting various industries including smart farming.

Reference

[1] A. Vangala, A. K. Das, N. Kumar, and M. Alazab, "Smart secure sensing for iot-based agriculture: Blockchain perspective," *IEEE Sensors Journal*, vol. 21, no. 16, pp. 17 591–17 607, 2020.

[2] A. D. Boursianis, M. S. Papadopoulou, P. Diamantoulakis, A. Liopa-Tsakalidi, P. Barouchas, G. Salahas, G. Karagiannidis, S. Wan, and S. K. Goudos, "Internet of things (iot) and agricultural unmanned aerial vehicles (uavs) in smart farming: A comprehensive review," *Internet of Things*, vol. 18, p. 100187, 2022.

[3] S. V. Akram, P. K. Malik, R. Singh, G. Anita, and S. Tanwar, "Adoption of blockchain technology in various realms: Opportunities and challenges," *Security and Privacy*, vol. 3, no. 5, p. e109, 2020.

[4] A. Maroli, V. S. Narwane, and B. B. Gardas, "Applications of iot for achieving sustainability in agricultural sector: A comprehensive review," *Journal of Environmental Management*, vol. 298, p. 113488, 2021.

[5] U. Bodkhe, S. Tanwar, P. Bhattacharya, and N. Kumar, "Blockchain for precision irrigation: Opportunities and challenges," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 10, p. e4059, 2022.

[6] K. Salah, N. Nizamuddin, R. Jayaraman, and M. Omar, "Blockchain-based soybean traceability in agricultural supply chain," *Ieee Access*, vol. 7, pp. 73 295–73 305, 2019.

[7] P. Kaur and A. Parashar, "A systematic literature review of blockchain technology for smart villages," *Archives of Computational Methods in Engineering*, pp. 1–52, 2021.

[8] M. Ayaz, M. Ammad-Uddin, Z. Sharif, A. Mansour, and E.-H. M. Aggoune, "Internet-of-things (iot)-based smart agriculture: Toward making the fields talk," *IEEE access*, vol. 7, pp. 129 551–129 583, 2019.

[9] E. Said Mohamed, A. Belal, S. Kotb Abd-Elmabod, M. A. El-Shirbeny, A. Gad, and M. B. Zahran.

[10] Y.-C. Liang and Y.-C. Liang, "Blockchain for dynamic spectrum management," *Dynamic Spectrum Management: From Cognitive Radio to Blockchain and Artificial Intelligence*, pp. 121–146, 2020.

[11] I. Mistry, S. Tanwar, S. Tyagi, and N. Kumar, "Blockchain for 5g-enabled iot for industrial automation: A systematic review, solutions, and challenges," *Mechanical systems and signal processing*, vol. 135, p. 106382, 2020.

[12] Q. Nasir, I. A. Qasse, M. Abu Talib, and A. B. Nassif, "Performance analysis of hyperledger fabric platforms," *Security and Commu[?]nication Networks*, vol. 2018, 2018.

[13] H. Liu, D. Han, and D. Li, "Fabric-iot: A blockchain-based access control system in iot," *IEEE Access*, vol. 8, pp. 18 207–18 218, 2020.

[14] M. N. M. Bhutta, A. A. Khwaja, A. Nadeem, H. F. Ahmad, M. K. Khan, M. A. Hanif, H. Song, M. Alshamari, and Y. Cao, "A survey on blockchain technology: Evolution, architecture and security," *IEEE Access*, vol. 9, pp. 61 048–61 073, 2021.

[15] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1508–1532, 2019.

[16] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE international congress on big data (BigData congress)*. Ieee, 2017, pp. 557–564.

[17] Y. Xinyi, Z. Yi, and Y. He, "Technical characteristics and model of blockchain," in *2018 10th international Conference on communication Software and networks (ICCSN)*. IEEE, 2018, pp. 562–566.

[18] H. Honar Pajooh, M. Rashid, F. Alam, and S. Demidenko, "Hyperledger fabric blockchain for securing the edge internet of things," *Sensors*, vol. 21, no. 2, p. 359, 2021.

[19] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future generation computer systems*, vol. 107, pp. 841–853, 2020.

[20] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[21] C. V. B. Murthy, M. L. Shri, S. Kadry, and S. Lim, "Blockchain based cloud computing: Architecture and research challenges," *IEEE Access*, vol. 8, pp. 205 190–205 205, 2020.

[22] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International journal of web and grid services*, vol. 14, no. 4@inproceedingsthakkar2018performance, title=Performance benchmarking and optimizing hyperledger fabric blockchain platform, author=Thakkar, Parth and Nathan, Senthil and Viswanathan, Balaji, booktitle=2018 IEEE 26th international symposium on modeling, analysis, and simulation of computer and telecommunication systems (MASCOTS), pages=264–276, year=2018, organization=IEEE , pp. 352–375, 2018.

[23] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: throughput, latency and scalability," in *2019 IEEE international conference on blockchain (Blockchain)*. IEEE, 2019, pp. 536–540.

[24] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[25] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2018, pp. 264–276.

[26] ——, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *2018 IEEE 26th international symposium on modeling, analysis, and simulation of computer and telecommunication systems (MASCOTS)*. IEEE, 2018, pp. 264–276.

[27] E. Navarro, N. Costa, and A. Pereira, "A systematic review of iot solutions for smart farming," *Sensors*, vol. 20, no. 15, p. 4231, 2020.

[28] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem, "A survey on the role

of iot in agriculture for the implementation of smart farming," *Ieee Access*, vol. 7, pp. 156 237–156 271, 2019.

[29] I. Mistry, S. Tanwar, S. Tyagi, and N. Kumar, "Blockchain for 5g-enabled iot for industrial automation: A systematic review, solutions, and challenges," *Mechanical systems and signal processing*, vol. 135, p. 106382, 2020.

[30] M. Torky and A. E. Hassanein, "Integrating blockchain and the internet of things in precision agriculture: Analysis, opportunities, and challenges," *Computers and Electronics in Agriculture*, vol. 178, p. 105476, 2020.

[31] L. Hang, I. Ullah, and D.-H. Kim, "A secure fish farm platform based on blockchain for agriculture data integrity," *Computers and Electronics in Agriculture*, vol. 170, p. 105251, 2020.

[32] S. Hu, S. Huang, J. Huang, and J. Su, "Blockchain and edge computing technology enabling organic agricultural supply chain: A framework solution to trust crisis," *Computers & Industrial Engineering*, vol. 153, p. 107079, 2021.

[33] A. S. Patil, B. A. Tama, Y. Park, and K.-H. Rhee, "A framework for blockchain based secure smart green house farming," in *International Conference on Ubiquitous Information Technologies and Applications, International Conference on Computer Science and its Applications*. Springer, 2018, pp. 1162–1167.

[34] S. El Kafhali, C. Chahir, M. Hanini, and K. Salah, "Architecture to manage internet of things data using blockchain and fog computing," in *Proceedings of the 4th international conference on big data and internet of things*, 2019, pp. 1–8.

[35] O. Friha, M. A. Ferrag, L. Shu, and M. Nafa, "A robust security framework based on blockchain and sdn for fog computing enabled agricultural internet of things," in *2020 International Conference on Internet of Things and Intelligent Applications (ITIA)*. IEEE, 2020, pp. 1–5.

[36] I. Ahmad, S. Abdullah, M. Bukhsh, A. Ahmed, H. Arshad, and T. F. Khan, "Message scheduling in blockchain based iot environment with additional fog broker layer," *IEEE Access*, vol. 10, pp. 97 165–97 182, 2022.

[37] K. Dey and U. Shekhawat, "Blockchain for sustainable e-agriculture: Literature

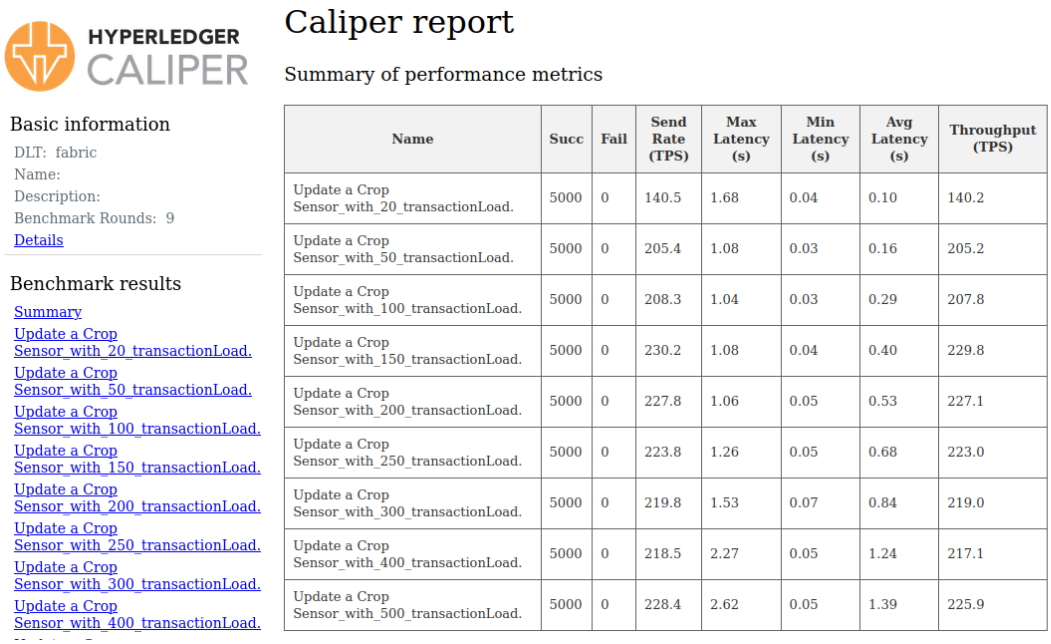
review, architecture for data management, and implications," *Journal of Cleaner Production*, vol. 316, p. 128254, 2021.

[38] S. Bojie, J. Loughlin, and A. Chettih, "Elliptic curve cryptography a public key system," 2013.

[39] H. Honar Pajooh, M. A. Rashid, F. a. Alam, and S. Demidenko, "Experimental performance analysis of a scalable distributed hyperledger fabric for a large-scale iot testbed," *Sensors*, vol. 22, no. 13, p. 4868, 2022.

Appendix

In Appendix 1: We present a snapshot of the Caliper benchmark report for the crop updating test. The image provides a visual representation of the test results and highlights the performance evaluation of the crop updating functionality.



HYPERLEDGER CALIPER

Basic information

DLT: fabric
Name:
Description:
Benchmark Rounds: 9
Details

Benchmark results

Summary
Update a Crop Sensor_with_20_transactionLoad.
Update a Crop Sensor_with_50_transactionLoad.
Update a Crop Sensor_with_100_transactionLoad.
Update a Crop Sensor_with_150_transactionLoad.
Update a Crop Sensor_with_200_transactionLoad.
Update a Crop Sensor_with_250_transactionLoad.
Update a Crop Sensor_with_300_transactionLoad.
Update a Crop Sensor_with_400_transactionLoad.

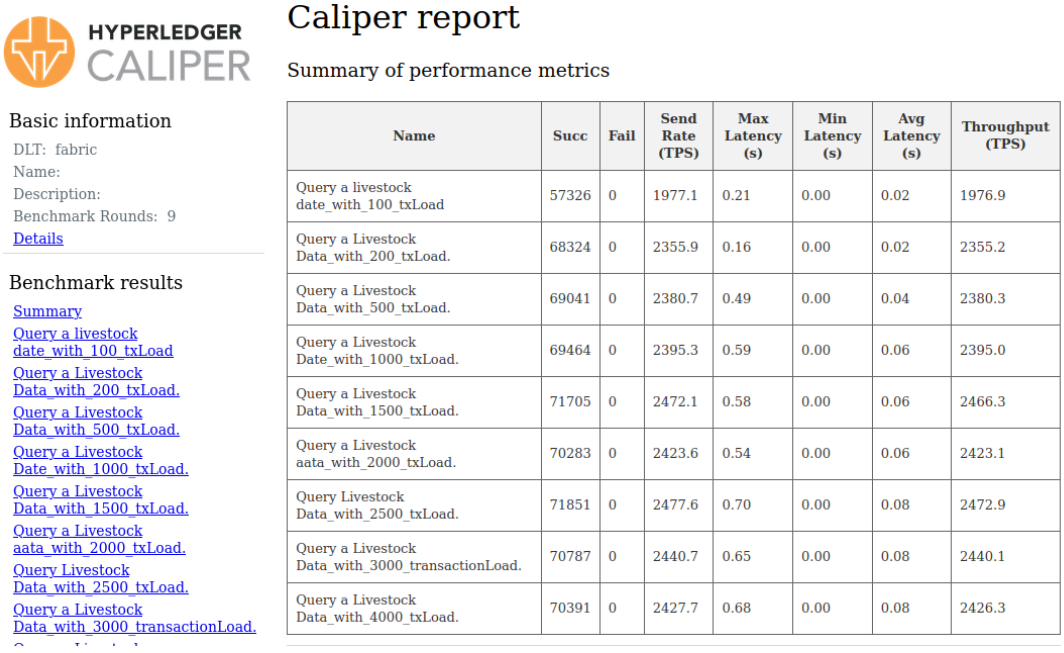
Caliper report

Summary of performance metrics

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
Update a Crop Sensor_with_20_transactionLoad.	5000	0	140.5	1.68	0.04	0.10	140.2
Update a Crop Sensor_with_50_transactionLoad.	5000	0	205.4	1.08	0.03	0.16	205.2
Update a Crop Sensor_with_100_transactionLoad.	5000	0	208.3	1.04	0.03	0.29	207.8
Update a Crop Sensor_with_150_transactionLoad.	5000	0	230.2	1.08	0.04	0.40	229.8
Update a Crop Sensor_with_200_transactionLoad.	5000	0	227.8	1.06	0.05	0.53	227.1
Update a Crop Sensor_with_250_transactionLoad.	5000	0	223.8	1.26	0.05	0.68	223.0
Update a Crop Sensor_with_300_transactionLoad.	5000	0	219.8	1.53	0.07	0.84	219.0
Update a Crop Sensor_with_400_transactionLoad.	5000	0	218.5	2.27	0.05	1.24	217.1
Update a Crop Sensor_with_500_transactionLoad.	5000	0	228.4	2.62	0.05	1.39	225.9

Caliper Benchmark Report for Updating Crop Sensor

In Appendix 2: we provide a snapshot of the Caliper benchmark report showcasing the querying of livestock data. The image displays the test results, specifically focusing on the performance evaluation of the livestock data querying functionality.



HYPERLEDGER CALIPER

Basic information

DLT: fabric
Name:
Description:
Benchmark Rounds: 9
Details

Benchmark results

Summary
Query a livestock date_with_100_txLoad
Query a Livestock Data_with_200_txLoad.
Query a Livestock Data_with_500_txLoad.
Query a Livestock Date_with_1000_txLoad.
Query a Livestock Data_with_1500_txLoad.
Query a Livestock aata_with_2000_txLoad.
Query Livestock Data_with_2500_txLoad.
Query a Livestock Data_with_3000_transactionLoad.

Caliper report

Summary of performance metrics

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
Query a livestock date_with_100_txLoad	57326	0	1977.1	0.21	0.00	0.02	1976.9
Query a Livestock Data_with_200_txLoad.	68324	0	2355.9	0.16	0.00	0.02	2355.2
Query a Livestock Data_with_500_txLoad.	69041	0	2380.7	0.49	0.00	0.04	2380.3
Query a Livestock Date_with_1000_txLoad.	69464	0	2395.3	0.59	0.00	0.06	2395.0
Query a Livestock Data_with_1500_txLoad.	71705	0	2472.1	0.58	0.00	0.06	2466.3
Query a Livestock aata_with_2000_txLoad.	70283	0	2423.6	0.54	0.00	0.06	2423.1
Query Livestock Data_with_2500_txLoad.	71851	0	2477.6	0.70	0.00	0.08	2472.9
Query a Livestock Data_with_3000_transactionLoad.	70787	0	2440.7	0.65	0.00	0.08	2440.1
Query a Livestock Data_with_4000_txLoad.	70391	0	2427.7	0.68	0.00	0.08	2426.3

Caliper Benchmark Report for Querying Livestock Data

In Appendix 3: we present a snippet of the chaincode function implemented in Go language for creating crop data within the smart contract. The code snippet showcases the CreateCropDataWithSensorData function, which is responsible for creating crop data and associated sensor data within the Hyperledger Fabric blockchain network.

```
func (s *SmartContract) CreateCropDataWithSensorData(ctx
contractapi.TransactionContextInterface, cropID string,
cropName string, cropOwner string, location string,
germinationDate string, harvestingDate string,
sensorData []SensorData) error {
existing, err := ctx.GetStub().GetState(cropID)
if err != nil {
return fmt.Errorf("failed to read from world state: %v",
    ↪ err)
}
if existing != nil {
return fmt.Errorf("the crop %s already exists", cropID)
}
crop := CropData{
CropName: cropName,
CropOwner: cropOwner,
Location: location,
GerminationDate: germinationDate,
HarvestingDate: harvestingDate,
}
cropJSON, err := json.Marshal(crop)
if err != nil {
return err}
err = ctx.GetStub().PutState(cropID, cropJSON)
if err != nil {
return fmt.Errorf("failed to write to world state: %v",
    ↪ err)
}
for _, data := range sensorData {
```

```
sensorKey := fmt.Sprintf("sensor_%s_%s", cropID,
data.SensorType)
sensor := SensorData{
SensorType: data.SensorType,
Value: data.Value,
Unit: data.Unit,
Timestamp: data.Timestamp,
}
sensorJSON, err := json.Marshal(sensor)
if err != nil {
return err
}
err = ctx.GetStub().PutState(sensorKey, sensorJSON)
if err != nil {
return err}}
return nil}
```

Appendix 4: shows the code snippet of the Query Livestock function from the smart contract. This function plays a crucial role in retrieving crop data along with its associated sensor data from the Hyperledger Fabric blockchain network.

QueryLivestockDataWithSensorData

```
func (s *SmartContract) QueryLivestockDataWithSensorData
   ↪ (ctx
 contractapi.TransactionContextInterface,
 livestockID string) (*LivestockData, error) {
livestockDataJSON, err := ctx.GetStub().GetState(
   ↪ livestockID)
if err != nil {
return nil, fmt.Errorf("
failed to read from world state: %v", err)
}
if livestockDataJSON == nil {
return nil, fmt.Errorf("the livestock with ID
```

```
%s does not exist", livestockID)
}
var livestockData LivestockData
err = json.Unmarshal(livestockDataJSON, &livestockData)
if err != nil {
return nil, err
}
// Filter sensor data based on the required criteria
filteredSensorData := make([]SensorData, 0)
for _, sensorData := range livestockData.SensorData {
if sensorData.Value > 100 {
filteredSensorData = append(filteredSensorData,
   ↪ sensorData)
}}
livestockData.SensorData = filteredSensorData
return &livestockData, nil}
```

Appendix 5: presents the code snippets for updating crop sensor data and livestock sensor data.

```
func (s *SmartContract) UpdateCropDataWithSensorData(ctx
   ↪ contractapi.TransactionContextInterface, cropID
   ↪ string, sensorData []SensorData) error {
   ↪ cropDataJSON, err := ctx.GetStub().GetState(cropID)
            if err != nil {
                  return fmt.Errorf("failed to read from
                     ↪ world state: %v", err)
            }

            if cropDataJSON == nil {
                  return fmt.Errorf("the crop %s does not
                     ↪  exist", cropID) }
                  var cropData CropData
            err = json.Unmarshal(cropDataJSON, &cropData)
```

```
            if err != nil {
                  return err}
                  cropData.SensorData = append(cropData.
                     ↪ SensorData, sensorData...)

            updatedCropDataJSON, err := json.Marshal(
               ↪ cropData)
            if err != nil {
                  return err
            }

            err = ctx.GetStub().PutState(cropID,
               ↪ updatedCropDataJSON)
            if err != nil {
                  return err }
                  return nil }
func (s *SmartContract) UpdateLivestockDataWithSensorData
   ↪ (ctx contractapi.TransactionContextInterface,
   ↪ livestockID string, sensorData []SensorData) error
   ↪ {
            livestockDataJSON, err := ctx.GetStub().
               ↪ GetState(livestockID)
            if err != nil {
                  return fmt.Errorf("failed to read from
                     ↪ world state: %v", err) }
                  if livestockDataJSON == nil {
                  return fmt.Errorf("the livestock with
                     ↪ ID %s does not exist",
                     ↪ livestockID)}
                  var livestockData LivestockData
            err = json.Unmarshal(livestockDataJSON, &
               ↪ livestockData)
            if err != nil {
```

```
            return err }
        livestockData.SensorData = append(
            ↪ livestockData.SensorData, sensorData
            ↪ ...)

        updatedLivestockDataJSON, err := json.Marshal
            ↪ (livestockData)
        if err != nil {
            return err }
            err = ctx.GetStub().PutState(
                ↪ livestockID,
                ↪ updatedLivestockDataJSON)
        if err != nil {
            return err }
            return nil }
```