Optimized blockchain deployment and application for trusted industrial internet of things

Dun Li

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2024IPPAS016

Thèse de doctorat

TELECOM SudParis

IP PARIS

Optimized Blockchain Deployment and Application for Trusted Industrial Internet of Things

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Evry, le 24 septembre 2024, par

DUN LI

Composition du Jury :

Joaquin Garcia-Alfaro Professor, IP Paris, Telecom Paris - France	Président
Mirjana Ivanovic Professor, University of Novi Sad - Serbia	Rapporteur
Joanna Kolodziej Professor, NASK-PIB - Poland	Rapporteur
Abdelkader Lahmadi Associate Professor, ENSEM-Université de Lorraine - France	Examinateur
Joaquin Garcia-Alfaro Professor, IP Paris, Telecom Paris - France	Examinateur
Noel Crespi Professor, IP Paris, Telecom SudParis - France	Directeur de thèse
Roberto Minerva Professor, IP Paris, Telecom SudParis - France	Co-directeur de thèse

Doctor of Philosophy (PhD) Thesis
Institut-Mines Télécom, Télécom SudParis
& Institut Polytechnique de Paris (IP Paris)

Specialization

Computing, Data and Artificial Intelligence

presented by

Dun Li

Optimized Blockchain Deployment and Application for Trusted Industrial Internet of Things

Commitee:

Mirjana Ivanovic	Rapporteurs	Professor, University of Novi Sad - Serbia
Joanna Kolodziej	Rapporteurs	Professor, NASK-PIB - Poland
Joaquin Garcia-Alfaro	Examiner	Professor, IP Paris, Telecom Paris - France
Abdelkader Lahmadi	Examiner	Associate professor, ENSEM-Université de Lorraine - France
Noel Crespi	Advisor	Professor, IP Paris, Telecom SudParis - France
Roberto Minerva	Co-advisor	Professor, IP Paris, Telecom SudParis - France

Thèse de Doctorat (PhD) de
Institut-Mines Télécom, Télécom SudParis
et l'Institut Polytechnique de Paris (IP Paris)

Spécialité

Informatique, Données et Intelligence Artificielle

présentée par

Dun Li

Déploiement et Application Optimisés de la Blockchain pour un Internet Industriel des Objets de Confiance

Jury composé de :

Mirjana Ivanovic	Rapporteurs	Professor, University of Novi Sad - Serbia
Joanna Kolodziej	Rapporteurs	Professor, NASK-PIB - Poland
Joaquin Garcia-Alfaro	Examiner	Professor, IP Paris, Telecom Paris - France
Abdelkader Lahmadi	Examiner	Associate professor, ENSEM-Université de Lorraine - France
Noel Crespi	Directeur de thèse	Professeur, IP Paris, Telecom SudParis - France
Roberto Minerva	Co-encadrant	Professeur, IP Paris, Telecom SudParis - France

Dedication

To my most honorable mother, Xiaomei Ma.
I want to express my deepest gratitude to my mother, who has been my steadfast support and inspiration throughout my academic journey. Her immense love, endless patience, and constant encouragement have been the foundation of my strength and determination. I am forever grateful for everything you have done for me.

To my most respected father, WeiJi Li.
I want to express my heartfelt gratitude to my father, whose guidance, wisdom, and steadfast faith in me have been essential in my academic journey. His strong work ethic, integrity, and perseverance have inspired me and guided me in my pursuit of excellence. I owe a significant part of my success to your invaluable influence in my life.

To all my distinguished colleagues and friends.
I am sincerely grateful to all my classmates and colleagues who have been an important part of my academic and professional journey. Each of you has contributed in your unique way, providing me with valuable insights, support, and companionship. Thank you for being an extraordinary part of this journey and for the lasting friendships we have made.

Acknowledgements

I am deeply thankful for the kind support and guidance I've received. Completing this thesis is not just an academic achievement but also the result of many moments of discussion, encouragement, and crucial support from many people.

First and foremost, I extend my heartfelt gratitude to my mentors at Telecom Sudparis, Prof. Noel and Prof. Roberto. Their dedication, guidance, and confidence in my abilities have been the pillars of my academic journey. Prof. Noel's deep knowledge and careful mentorship have greatly improved my analytical skills, while Prof. Roberto's expertise and practical perspectives have been invaluable in understanding the complexities of my field.

I sincerely thank my colleagues Reza, Praboda, Amir, Mohsan, Wenhao, Leyuan, Myra, and Jing. Our teamwork and discussions have greatly enhanced my research. The unique insights and contributions they provided have been invaluable.

I extend my deepest gratitude to my family and friends, whose steadfast love, encouragement, and sacrifices have been my foundation. Your continuous support and belief in my goals have given me the strength to pursue my dreams.

To everyone who has been part of this journey (professors, peers, administrative staff, and the academic community), thank you for your contributions, big and small, to my doctoral experience. Your support has been indispensable and greatly appreciated throughout the completion of this work.

Dun Li
Every
15^{th} June 2024

Abstract

The continued advancement of the Industrial Internet of Things (IIoT) presents promising prospects and numerous opportunities for improving the operational frameworks of industrial systems. However, IIoT architectures face significant challenges, including centralized control, vulnerability to cyber attacks, privacy violations, and data accuracy issues. These challenges create significant obstacles in securing data, which is crucial for the growth of this technology. To address these issues, many researchers suggest integrating blockchain technology as a stable means to safeguard data within IIoT systems.

Blockchain's features of distributed storage, decentralization, and immutability offer distinct advantages in data secure storage, identity verification, and access control. Despite these benefits, as IIoT applications diversify and data scales expand, the high resource demand of blockchain systems clashes with the limited resources of IIoT devices, leading to unresolved contradictions and persistent issues within this solution. Existing blockchain architectures still lack anonymous and efficient IIoT identity authentication, with complex encryption and decryption processes inducing excessive system overhead. To address these issues, the thesis builds on prior research to optimize blockchain performance, aiming to resolve the shortcomings and bottlenecks in current blockchain-based IIoT architectures regarding data security protection.

Firstly, this thesis introduces a lightweight blockchain-enabled protocol designed for secure data storage in the dynamic IIoT environment. It incorporates bilinear mapping for system initialization, entity registration, and authentication technology to authenticate IIoT entities efficiently and securely, along with an off-chain data storage approach to ensure data integrity with reduced resource consumption.

Furthermore, the thesis addresses the limitations of Hyperledger fabric systems in high availability scenarios by proposing Trie-Fabric, which enhances transaction processing through a Directed Acyclic Graph (DAG) based transaction sorting algorithm. This approach significantly reduces terminated transactions, optimizes conflict handling, and increases efficiency by more than 60% in its best case, according to comparative experimental results.

To manage the increasingly sophisticated industrial processes and privacy-sensitive data generated by IIoT devices, the thesis proposes a smart contract-assisted access control scheme utilizing the Attribute-Based Access Control (ABAC) model. This scheme, supported by bloom filter components, demonstrates controlled contract execution times, stable system throughput, and a rapid consensus process in real-world simulations, making it highly capable of handling high-throughput and effective consensus even under large-scale request scenarios.

Lastly, the thesis introduces the Zero-Knowledge Proof (ZKP) algorithm, which integrates a non-interactive zero-knowledge proof protocol with Ciphertext-Policy Attribute-Based Encryption (CP-ABE) to enhance security and efficiency in IIoT content distribution. Combined with the Distributed Publish-Subscribe IIoT (DPS-IIoT) system using Hyper-

ledger fabric, it significantly improves bandwidth efficiency and overall throughput in IIoT environments.

Through comprehensive security performance evaluations and experimental results, this research confirms the protocols' effectiveness in minimizing system overhead, improving storage reliability, and enhancing overall IIoT data management and application security. This thesis provides an in-depth examination of advanced data management protocols and systems for the IIoT, which are crucial for advancing the manufacturing sector. Consequently, this work makes a significant contribution to the field of IIoT data security, offering scalable and robust solutions for current and future industrial systems.

Keywords

Blockchain, Industrial Internet of Things, Trie Tree, Hyperledger Fabric, Concurrency Optimization, Practical Byzantine Fault Tolerance Consensus, Directed Acyclic Graph, Transaction Sorting Algorithm, Information-centric Networking, Zero-Knowledge Proof.

Résumé

La progression continue de l'internet industriel des objets (IIoT) offre des perspectives prometteuses et de nombreuses possibilités d'améliorer les cadres opérationnels des systèmes industriels. Cependant, les architectures de l'IIoT sont confrontées à des défis importants, notamment le contrôle centralisé, la vulnérabilité aux cyberattaques, les violations de la vie privée et les problèmes d'exactitude des données. Ces défis créent des obstacles importants à la sécurisation des données, qui est cruciale pour la croissance de cette technologie. Pour résoudre ces problèmes, de nombreux chercheurs suggèrent d'intégrer la technologie blockchain comme moyen stable de protéger les données au sein des systèmes IIoT. Les caractéristiques de stockage distribué, de décentralisation et d'immutabilité de la blockchain offrent des avantages significatifs en matière de stockage sécurisé des données, de vérification de l'identité et de contrôle d'accès.

Malgré ces avantages, à mesure que les applications IIoT se diversifient et que les volumes de données croissent, la forte demande en ressources des systèmes blockchain se heurte aux ressources limitées des appareils IIoT, ce qui entraîne des contradictions non résolues et des problèmes persistants. Les architectures blockchain existantes manquent encore d'authentification d'identité IIoT anonyme et efficace, avec des processus de cryptage et de décryptage complexes induisant une surcharge excessive du système. Pour résoudre ces problèmes, cette thèse s'appuie sur des recherches antérieures pour optimiser les performances de la blockchain, visant à résoudre les lacunes et les goulets d'étranglement des architectures IIoT actuelles basées sur la blockchain en ce qui concerne la protection de la sécurité des données.

Tout d'abord, cette thèse présente un protocole léger basé sur la blockchain, conçu pour le stockage sécurisé des données dans l'environnement dynamique de l'IIoT. Il intègre la cartographie bilinéaire pour l'initialisation du système, l'enregistrement des entités et la technologie d'authentification pour authentifier les entités IIoT de manière efficace et sécurisée, ainsi qu'une approche de stockage des données hors chaîne pour garantir l'intégrité des données avec une consommation de ressources réduite.

En outre, la thèse aborde les limites des systèmes Hyperledger fabric dans les scénarios de haute disponibilité en proposant Trie-Fabric, qui améliore le traitement des transactions grâce à un algorithme de tri des transactions basé sur un graphe acyclique dirigé (DAG). Cette approche réduit considérablement les transactions interrompues, optimise la gestion des conflits et augmente l'efficacité de plus de 60% dans le meilleur des cas, selon des résultats expérimentaux comparatifs.

Pour gérer les processus industriels de plus en plus sophistiqués et les données sensibles à la vie privée générées par les appareils IIoT, cette thèse propose un schéma de contrôle d'accès assisté par smart contract utilisant le modèle de contrôle d'accès basé sur les attributs (ABAC). Ce concept, supporté par des composants de filtre de bloom, permet de bénéficier de temps d'exécution de contrat contrôlés, d'un débit de système stable et d'un processus de consensus rapide dans des simulations réelles, ce qui le rend capable de gérer

un débit élévé et un consensus efficace, même dans des scénarios de demande à grande échelle.

Enfin, cette thèse présente l'algorithme Zero-Knowledge Proof (ZKP), qui intègre un protocole de preuve à zéro connaissance non-interactif avec le chiffrement basé sur les attributs de la politique de chiffrement (CP-ABE) pour améliorer la sécurité et l'efficacité de la distribution de contenu IIoT. Combiné au système Distributed Publish-Subscribe IIoT (DPS-IIoT) utilisant Hyperledger fabric, il améliore considérablement l'efficacité de la bande passante et le débit global dans les environnements IIoT. Grâce à des évaluations complètes des performances de sécurité et à des résultats expérimentaux, ces recherches confirment l'efficacité des protocoles pour minimiser la surcharge du système, améliorer la fiabilité du stockage et renforcer la gestion globale des données de l'IIoT et la sécurité des applications.

Cette thèse fournit un examen approfondi des protocoles et systèmes de gestion de données avancés pour l'IIoT, qui sont cruciaux pour faire progresser le secteur de la fabrication. Par conséquent, ce travail apporte une contribution significative au domaine de la sécurité des données de l'IIoT, en offrant des solutions évolutives et robustes pour les systèmes industriels actuels et futurs.

Mots-clés

Blockchain, Internet Industriel des Objets, Arbre Trie, Hyperledger Fabric, Optimisation de la concurrence, Consensus de tolérance de faute byzantine pratique, Graphe acyclique dirigé, Algorithme de tri des transactions, Réseautage centré sur l'information, Preuve à divulgation nulle de connaissance.

Table of contents

Chapter 1

Introduction

Contents

1.1 Background and Motivation

As a typical application of the Internet of Things (IoT), the Industrial Internet of Things (IIoT) enhances resource allocation by connecting smart devices, enabling data collection and analysis, and significantly improving production efficiency and quality. However, with the increasing interconnection of industrial systems, data security and privacy protection issues are becoming critical concerns that cannot be ignored. The rapid expansion of IIoT devices has led to the generation of vast amounts of sensitive and missioncritical data. If compromised or leaked, such data could result in severe operational disruptions and substantial financial losses. Most IIoT applications currently rely on centralized servers for data storage and management, along with third-party communication protocols for information sharing and data transmission. The centralized architecture presents several challenges, including increased security risks due to single points of failure, high operational and maintenance costs, and response delays that undermine the real-time nature of IIoT systems. Among these, the security risks associated with unauthorized data access, tampering, and sharing have emerged as particularly critical issues, demanding immediate attention.

In the context of Industry 4.0, IIoT devices form a distributed and dynamic network that constantly interacts with the external environment, and the demand for powerful, scalable, and flexible security mechanisms is crucial. The vast amounts of heterogeneous data generated and exchanged between devices must be protected from unauthorized access while maintaining both integrity and confidentiality. However, traditional centralized approaches often fall short of meeting these stringent requirements, highlighting the need to explore decentralized solutions. Blockchain technology presents a promising approach to overcoming these security challenges. Its decentralized and immutable nature offers a secure foundation for managing and transmitting industrial data in IIoT environments, addressing many vulnerabilities inherent in centralized architectures. Specifically, blockchain enhances data storage security, strengthens communication integrity between IIoT devices, and facilitates distributed data processing. Despite these advantages, the high computational demands of blockchain, particularly in systems using the Proof of Work (PoW) consensus mechanism, lead to significant resource consumption and operational inefficiencies. These challenges create substantial obstacles to the large-scale adoption of blockchain in IIoT systems, where resource limitations are a key concern.

Thus, the primary research problem addressed in this thesis is the development of an optimized blockchain framework that overcomes the computational and scalability limitations of current blockchain-based IIoT solutions, while still providing a high level of security, privacy, and efficiency. This research focuses on proposing a lightweight, secure,

and scalable blockchain architecture tailored for IIoT environments, which can address the following key challenges:

- **Security and Privacy**: How can blockchain technology be leveraged to protect the confidentiality and integrity of IIoT data without overwhelming the system with computational overhead?
- **Scalability**: How can blockchain be optimized to handle the large-scale, high-frequency transactions typical of IIoT systems, without sacrificing performance or security?
- **Efficient Data Management**: How can novel data structures and cryptographic techniques be used to streamline secure storage, access control, and transaction processing in IIoT systems?
- **Advanced Cryptographic Techniques**: How can modern cryptographic techniques, such as zero-knowledge proof (ZKP), be integrated to further secure IIoT systems while maintaining performance efficiency?

This thesis addresses these questions by introducing a series of improvements to blockchain technology. By enhancing the robustness of blockchain while mitigating its computational demands, the proposed solution provides a trusted, low-overhead, and lightweight secure storage scheme. Incorporating cryptographic techniques such as asymmetric encryption and message authentication codes, the scheme ensures identity anonymity, traceability, and secure storage, all while significantly reducing computational overhead. In addition, innovative approaches like trie trees and directed acyclic graphs (DAGs) are explored to manage large-scale transactions and resolve concurrent conflicts, improving the overall scalability of blockchain in IIoT applications. The integration of smart contracts further strengthens security by enabling dynamic, fine-grained access control mechanisms tailored to real-time operational needs. These layered improvements ultimately culminate in the DPS-IIoT framework, offering a comprehensive solution to the multifaceted challenges of security, privacy, and scalability in IIoT systems. As a result, this thesis offers a strategic and systematic enhancement of blockchain technology, evolving from a fundamental secure data management system into a sophisticated, application-specific framework designed to meet the evolving demands of IIoT.

1.2 Publications List

Accept Papers

- Li D, Han D, Crespi N, et al. A blockchain-based secure storage and access control scheme for supply chain finance[J]. The Journal of Supercomputing, 2023, 79(1):

109-138.

- Li D, Li H, Crespi N, et al. Blockchain-Enabled Large Language Models for Prognostics and Health Management Framework in Industrial Internet of Things[C]//Blockchain and Trustworthy Systems: International Conference, BlockSys 2024, Hangzhou, China, July 12, 2024.

- Shao W, Wei Y, Rajapaksha P, Dun L, Noel C, et al. Low-latency Dimensional Expansion and Anomaly Detection empowered Secure IoT Network[J]. IEEE Transactions on Network and Service Management, 2023 (99): 1-1.

- Cai S, Han D, Li D, Crespi N, et al. An reinforcement learning-based speech censorship chatbot system[J]. The Journal of Supercomputing, 2022, 78(6): 8751-8773.

- Shao W, Rajapaksha P, Wei Y, D Li, et al. COVAD: Content-oriented video anomaly detection using a self attention-based deep learning model[J]. Virtual Reality & Intelligent Hardware, 2023, 5(1): 24-41.

Submitted Papers

- Li D, Xia B, Crespi N, et al. Trie-Fabric: A trie-alternative transaction serial sorting scheme in permissioned blockchain networks[J]. Advanced Engineering Informatics, 2024.

- Li D , Han D, Crespi N, et al. Blockchain in the Digital Twin Context: A Comprehensive Survey, [J]. ACM Computing Surveys, 2024.

- Li D, Crespi N, Roberto M, et al. MLAE: A Cosine Similarity-empowered Multi-loss Encoder Framework for Unsupervised Network Traffic Anomaly Detection[J]. IEEE Transactions on Instrumentation and Measurement, 2024.

- Li D, Liu H, Crespi N, et al. DPS-IIoT: Non-Interactive Zero-Knowledge Poof-inspired Access Control towards Information-Centric Industrial Internet of Things[J]. Computer Communications, 2024.

- Li D, Crespi N, Roberto M, et al. Blockchain-assisted Lightweight Data Storage Protocol for Large-scale Industrial Internet of Things[J]. IEEE Internet of Things, 2024.

- Li D, Li H, Crespi N, et al. Hyper-IIoT: A Smart Contract-inspired Access Control Scheme for Resource-constrained Industrial Internet of Things[J]. IEEE Transactions on Sustainable Computing, 2024.

1.3 Contributions of the Thesis

This thesis introduces a comprehensive framework designed to enhance the security, efficiency, and trustworthiness of IIoT environments. It addresses the complexities and emerging challenges within IIoT by leveraging blockchain technology and advanced cryptographic techniques. The core contributions of this thesis are presented through a series of publications, each focusing on specific aspects of the overall objective. Collectively, these publications contribute to developing a secure and efficient IIoT ecosystem. Below is a detailed explanation of each publication and its contribution to the thesis:

C.1 **Blockchain-assisted secure data storage protocol.** To address the **security and privacy** challenges in IIoT, this thesis proposes a lightweight and efficient data storage protocol leveraging blockchain technology to ensure decentralized storage and enhanced security. The proposed scheme combines blockchain with cryptographic principles and IPFS technology to guarantee secure data storage in industrial IIoT environments. It achieves conditional anonymity and robust security while minimizing the computational overhead associated with cryptographic operations. Key features include efficient message authentication, secure key management, and identity withdrawal mechanisms, thereby providing a resilient solution that balances IIoT's large-scale operational complexity with energy efficiency, data access speed, and long-term cost management.

C.2 **Trie-alternative transaction serial sorting scheme.** To solve the **scalability** problem in large-scale IIoT systems, this thesis presents a Trie-based serial sorting algorithm to handle the high volume of invalid transactions common in blockchain networks. By analyzing the inherent conflict dynamics in concurrent transactions within Hyperledger Fabric, this scheme constructs a conflict graph that eliminates circular dependencies and invalid transactions through topological sorting of a directed acyclic graph (DAG). This refined transaction sequence enhances processing efficiency and conflict resolution, significantly improving scalability and transaction throughput while ensuring IIoT device and stakeholder authentication remains secure and reliable.

C.3 **Smart contract-based access control scheme.** To address the challenge of **efficient data management** in IIoT, this thesis proposes a smart contract-based access control solution that enables fine-grained control over device access permissions. The use of attribute-based access control models

ensures a precise definition of equipment attributes, while smart contracts enforce custom access rules and log access activities for transparency and security. By optimizing smart contract management and querying mechanisms, this solution effectively reduces computing costs and response times, providing a secure and efficient way to manage IIoT data and device access in resource-constrained environments.

C.4 **Non-interactive zero-knowledge proof-inspired access control.** To enhance **advanced cryptographic techniques** for securing IIoT environments, this thesis introduces a non-interactive ZKP system combined with attribute-based ciphertext strategies. This approach ensures high-level privacy and reduces bandwidth consumption, while the proposed DPS-IIoT system, built on Hyperledger Fabric, enhances the security and reliability of IIoT networks. The ZKP-inspired system addresses the need for efficient bandwidth management and throughput in IIoT, offering a significant improvement over existing methods.

1.4 Relationship of Publications with Contributions

This section provides the relationships of publications with contributions.

- The publications "Blockchain-assisted Lightweight Data Storage Protocol for Large-scale Industrial Internet of Things" and "Blockchain-Enabled Large Language Models for Prognostics and Health Management Framework in Industrial Internet of Things" correspond to Contribution C.1.

- The publication "Trie-Fabric: A trie-alternative transaction serial sorting scheme in permissioned blockchain networks" corresponds to Contribution C.2.

- The publications "Hyper-IIoT: A Smart Contract-inspired Access Control Scheme for Resource-constrained Industrial Internet of Things" and "A blockchain-based secure storage and access control scheme for supply chain finance" correspond to Contribution C.3.

- The publication "DPS-IIoT: Non-Interactive Zero-Knowledge Poof-inspired Access Control towards Information-Centric Industrial Internet of Things" corresponds to Contribution C.4.

1.5 Outline of the Thesis

As Fig 1.1 shows, the thesis is structured into seven chapters.

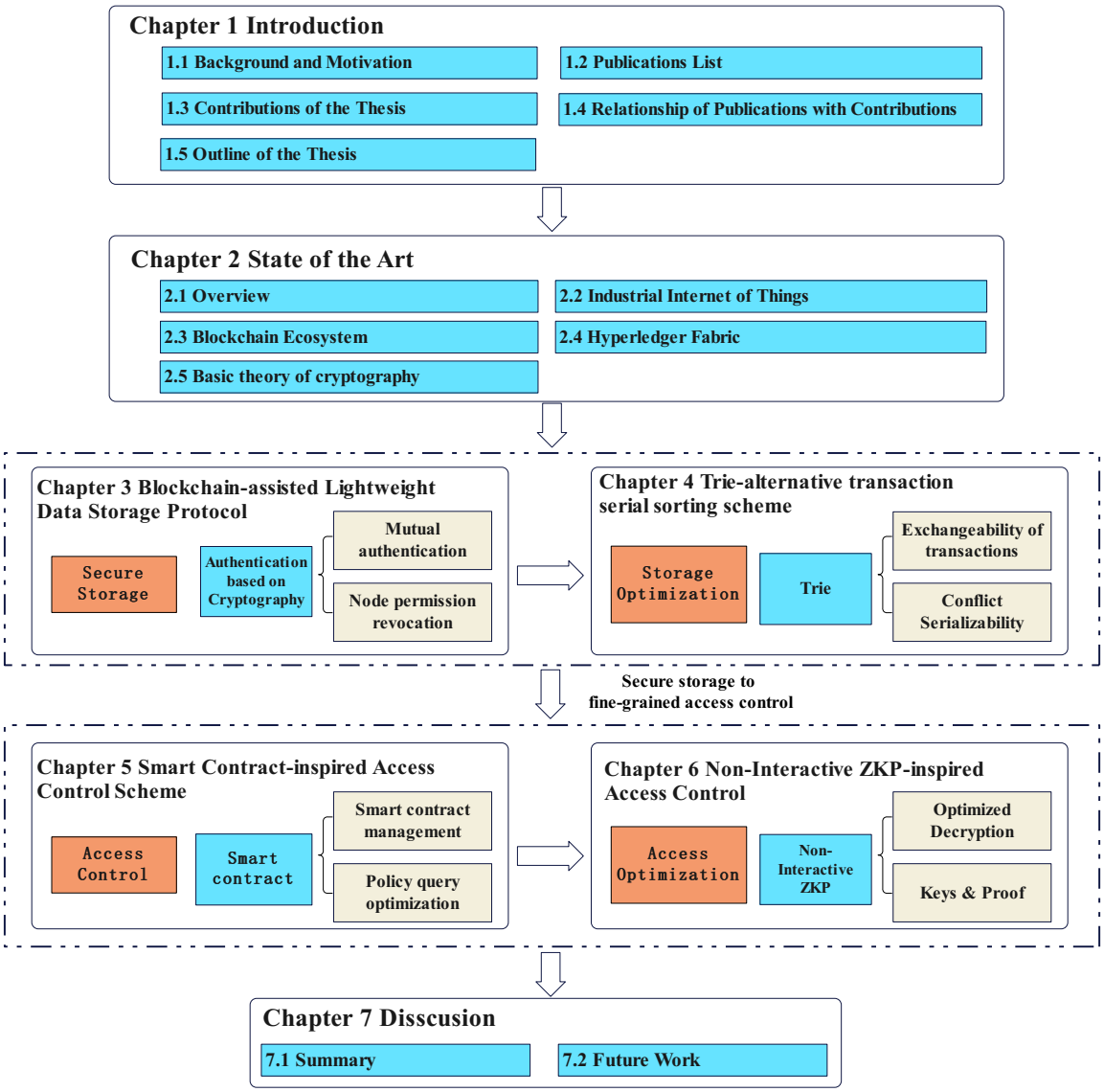


Figure 1.1: Outline of the Thesis

- **Chapter 1** introduces the research background, and the motivation for the study, and summarises the main contributions. It gives an overview of the project, describes the contents of each chapter, and outlines the overall structure of the thesis.
- **Chapter 2** provides an overview of the state of the art and related technologies

essential to the thesis, including the IIoT, blockchain ecosystem, Hyperledger fabric architecture, fundamental cryptography theories, and ZKP protocols.

- **Chapter 3** introduces the first innovation of this thesis: a Blockchain-assisted Lightweight Data Storage Protocol designed specifically for the expansive realm of Industrial IIoT. This protocol offers a robust solution that aligns the complex network structures of IIoT with the decentralized nature of blockchain, aiming to enhance both data security and system efficiency. The chapter details the framework, highlighting the use of smart contracts for authenticating entity components and enabling secure, anonymous identity verification. Additionally, it presents a tailored data storage model that integrates edge computing with blockchain technology to address the unique challenges of IIoT environments. The model ensures the integrity and safety of key data while enabling efficient and transparent data operations. Performance evaluations and simulations based on the Hyperledger fabric platform are presented to validate the protocol's effectiveness in reducing computational load, enhancing storage reliability, and strengthening overall security in IIoT data management and applications. This chapter offers scalable and robust data security solutions relevant to the evolving landscape of industrial systems, marking a significant advancement in the thesis.

- **Chapter 4** presents the second contribution of this thesis, focusing on "Trie-Fabric: A trie-alternative transaction serial sorting scheme in permissioned blockchain networks". This chapter delves into the complex transaction management challenges within permissioned blockchain environments like Hyperledger fabric, where high transaction throughput and efficient processing are paramount. The chapter introduces a novel transaction reordering algorithm, exploiting Trie structures and Directed Acyclic Graphs (DAGs) to effectively manage and resolve transaction conflicts. By replacing traditional sorting and validation methods with a Trie and DAG-based approach, this chapter demonstrates a significant improvement in handling large-scale transaction conflicts, thereby enhancing overall system performance and efficiency. Through careful design and implementation, including benchmark tests with SmallBank workload and comprehensive experiments, the chapter illustrates the superiority of Trie-Fabric in conflict resolution and transaction processing compared to the original Fabric system. The results show that Trie-Fabric can increase efficiency by more than 60% at the optimal point, marking a substantial improvement in transaction management for blockchain systems. This chapter contributes significantly to the thesis by providing a robust and efficient solution to the critical problem of transaction sorting in blockchain networks, especially for permissioned systems where

security, privacy, and efficiency are essential.

- **Chapter 5** presents the third contribution of this thesis, "Hyper-IIoT: A Smart Contract-inspired Access Control Scheme for Resource-constrained Industrial Internet of Things." It addresses the critical need for robust access control in the complex and dynamic IIoT environment. This chapter introduces a novel framework that uses blockchain technology and smart contracts to ensure secure storage and confidential data sharing within IIoT networks. By adopting an Attribute-Based Access Control (ABAC) model and integrating it with bloom filters, the chapter presents a scalable and efficient method for securing private data in IIoT environments. This approach significantly reduces system overhead and enhances security through smart contracts that manage complex data structures and enforce access controls. Additionally, the chapter includes comprehensive simulations and experiments to validate the effectiveness and efficiency of the proposed Hyper-IIoT framework. These validations demonstrate the framework's potential to transform IIoT ecosystems by providing a more private, reliable, and efficient data management and access control system. This chapter significantly advances the state of security and data management in IIoT, offering a scalable and robust solution for current and future industrial systems.

- **Chapter 6** presents the final contribution of this thesis, "DPS-IIoT: Non-Interactive Zero-Knowledge Proof-inspired Access Control for Information-Centric IIoT". This chapter addresses evolving security needs in the IIoT by integrating Information-Centric Networking (ICN) with advanced access control. The chapter introduces a novel framework that merges Ciphertext-Policy Attribute-Based Encryption (CP-ABE) with ICN in the IIoT, ensuring efficient data sharing and user privacy. An advanced ZKP protocol is crucial for effectively authenticating user attributes and minimizing bandwidth usage within CP-ABE systems. The DPS-IIoT model uses a Publish/Subscribe method to ensure scalability and efficient data distribution, reducing redundant network transmissions. Hyperledger fabric stores replicated access policies, and access control functions are supported by ZK-CP-ABE via smart contracts for high performance, verifiability, and replicability. Through detailed discussions and experimental evaluations, the chapter demonstrates the effectiveness of the DPS-IIoT framework in addressing IIoT security challenges. These contributions significantly enhance understanding and capabilities in IIoT security, laying a foundation for future research and development in secure, efficient, and user-centric IIoT environments.

- **Chapter 7** summarizes the contributions of the thesis and discusses potential future research directions to further enhance and expand upon the presented work.

Chapter 2

Preliminaries and State of the Art

Contents

2.1 Overview

This chapter provides an overview of the background knowledge and research foundations related to IIoT, the blockchain ecosystem, Hyperledger fabric, and cryptography.

2.2 Industrial Internet of Things

The IIoT integrates large-scale sensors, processors, and communication equipment, using various standards, services, and technologies. Mutual communication and data sharing between devices create ubiquitous connections between users, services, and equipment. As shown in Figure 2.1, each time an IIoT terminal interacts via the internet, it needs to send data to cloud service providers and upload it to third-party cloud data centres [1]. This section introduces the basic technologies of the IIoT from two perspectives: architecture and message protocols.

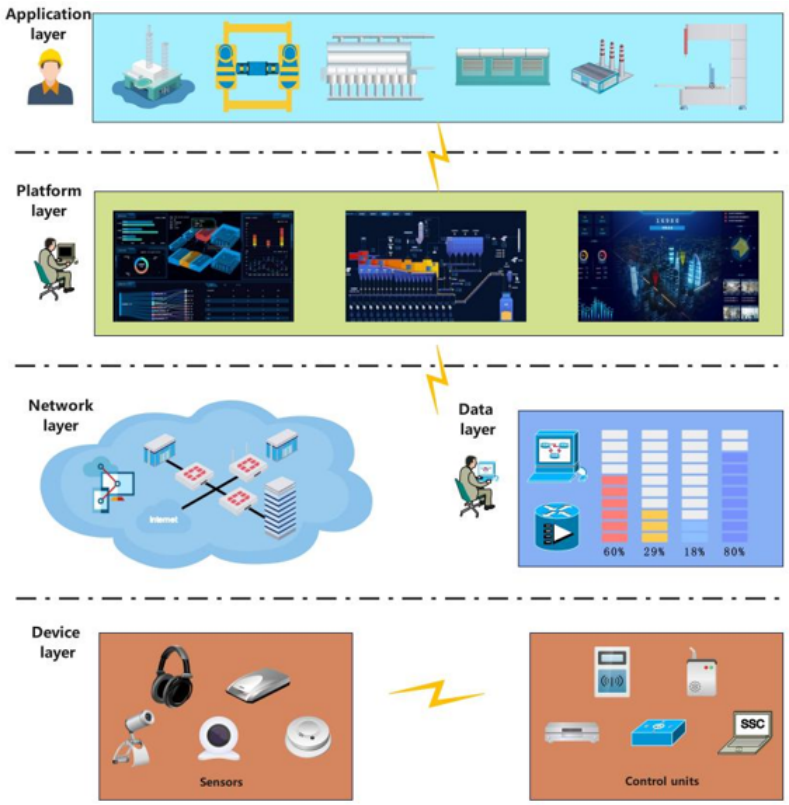


Figure 2.1: The network structure diagram of the Industrial Internet of Things

2.2.1 Architecture of the Internet of Things

IIoT consists of layers including data storage centers, cloud servers, and terminal devices. Compared to traditional internet devices, terminal devices are smaller, more mobile, and more resource-constrained. Using wireless technologies, IIoT incorporates concepts such as secure multiparty computation and cloud computing [2].

Basic architecture, as defined by early researchers and organizations, involved a simple three-layer IIoT structure, consisting of perception, network, and application layers, devised to meet the demands of industry growth and the increasing number of devices [3]. However, this structure was found inadequate for actual IIoT environments as it didn't cover all underlying technologies for data transmission to the IIoT platform. Additionally, the model was initially designed for specific communication media, such as wireless sensor networks, without considering operation on resource-constrained devices.

Service-oriented Architecture (SOA) has been highlighted as a key technology for integrating heterogeneous systems or devices in IIoT [4]. The SOA-based IIoT architecture is a five-layer structure (perception, object abstraction, middleware, application, and management layers), with middleware pairing services with requesters based on addresses and names, allowing IIoT applications to process heterogeneous objects without considering specific hardware platforms. The management layer oversees and manages the four underlying layers. From a technical standpoint, SOA is an adaptive architecture that meets the requirements of IIoT in terms of scalability, modularity, interoperability among heterogeneous devices, and efficient event-driven capabilities.

Cloud and fog-based edge computing have recently been integrated into the IIoT infrastructure [5]. In this structure, all sensors and actuators are connected to the cloud, where information generated by sensors is processed, and the computational results are returned to the actuators. From the information flow perspective, this represents an ideal model, allowing easy updates of control functions in the cloud and the use of big data technologies to analyze sensor-generated data sets and optimize production efficiency.

Information-Centric Networking (ICN) has become central with the proliferation of 4G and 5G. The core idea of ICN is to cache and route popular content copies through the network, reducing resource waste caused by repetitive transmission [6]. ICN decouples content from hosts, breaking the host-centric naming rules of TCP/IP protocols, and bases data exchange between nodes on content names rather than destination addresses [7,8]. In short, ICN has changed the focus of network transmission from "where is the content" to "what is the most important content."

2.2.2 Message Protocol

IIoT employs a diverse array of message transmission protocols designed to meet its varied needs, in contrast to the web's reliance on HTTP [9]. Over the past decades, organizations have developed protocols such as AMQP and JMS for swift transactional support [10], MQTT and CoAP for efficient data collection in constrained networks [11, 12], and XMPP and SIP for real-time message processing [13]. In contrast, protocols like RESTful, HTTP, and CoAP are suitable for internet traversal, making them ideal for web applications. Selecting the appropriate protocol requires a thorough understanding of its advantages and disadvantages, considering specific application requirements and security risks. This section introduces three prominent IIoT message transmission protocols: CoAP, MQTT, and AMQP, shown at the top of the IIoT network protocol stack (application layer) in Figure 2.2.

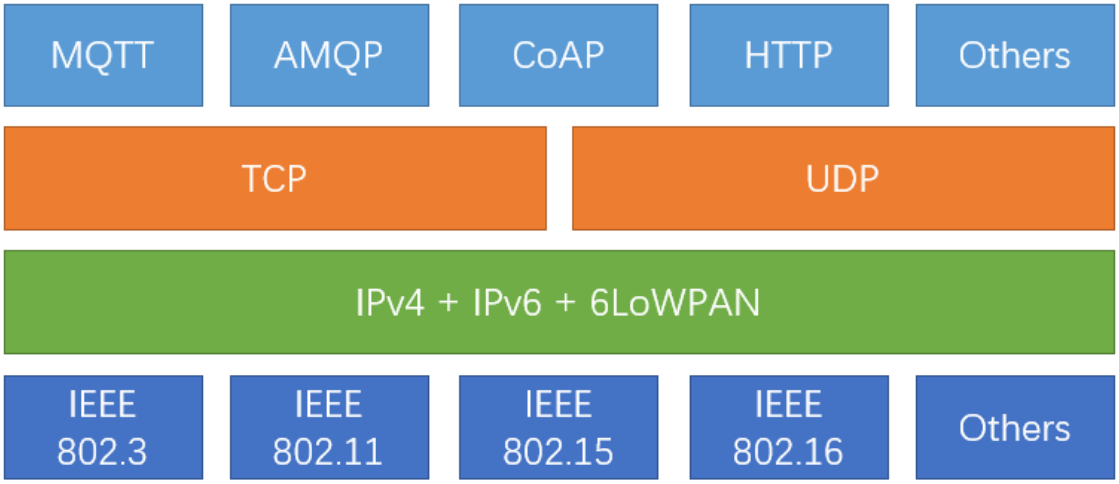


Figure 2.2: The Protocol Stack of IIoT Systems

Constrained Application Protocol (CoAP) is a typical application layer protocol for IIoT, defining a web transfer protocol that builds upon the representational state transfer (REST) functionalities of HTTP [14]. RESTful architecture simplifies data exchange between client and server via HTTP, ensuring stateless connections with cacheable responses. CoAP addresses the constraints of IIoT environments by facilitating CRUD operations (Create, Retrieve, Update, Delete) through HTTP methods such as GET, POST, PUT, and DELETE, thereby enabling the exposure and utilization of web services like SOAP. CoAP, unlike traditional REST, employs UDP instead of TCP by default, making it suitable for unstable IIoT networks. Additionally, CoAP adapts some HTTP features for low-power operations and functioning over unreliable links, aligning with IIoT needs.

Its compatibility with other REST-based applications facilitates seamless integration. The overall functionalities of the CoAP protocol are shown in Figure 2.3.

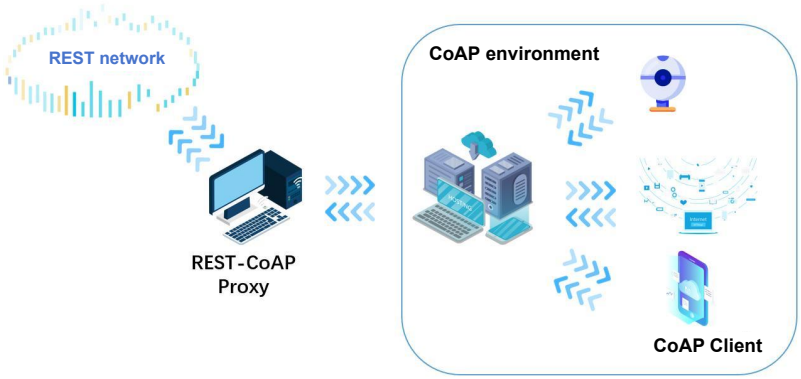


Figure 2.3: The Workflow of CoAP protocol

Message Queue Telemetry Transport (MQTT), developed by IBM researchers in 1999 and standardized by OASIS in 2013 [15], facilitates lightweight Machine to Machine (M2M) communication in constrained networks. Operating on a publish/subscribe model, MQTT allows clients to send messages to a broker, where other clients can either access them immediately or store them for later retrieval through subscriptions, as shown in Figure 2.4. Messages are published to topics, allowing clients to follow several topics at once and obtain corresponding messages. MQTT supports two specifications: MQTT v3.1, operating on TCP, and MQTT-SN, utilizing UDP and featuring indexed topic names for enhanced efficiency in constrained networks. With three QoS levels, MQTT caters to a variety of network environments, making it suitable for both M2M and S2M networks. Its widespread adoption in the industry has solidified MQTT as a preferred connection protocol for IIoT.

Advanced Message Queuing Protocol (AMQP) is a lightweight M2M protocol designed for enterprise-level messaging with a focus on reliability, security, resource allocation, and interoperability [16]. It offers an array of messaging functionalities, including reliable queues, topic-based publishing and subscribing flexible routing, and transaction support. AMQP utilizes TCP as its default transport protocol, ensuring security through TLS/SSL and SASL. Communication between clients and brokers is connection-oriented. AMQP's two main components are Exchanges, for routing messages to appropriate queues following preset rules, and Queues, for storing and distributing messages to recipients. In addition to point-to-point communication, AMQP further supports publish/subscribe models, offering two basic levels of Quality of Service (QoS): unreliable and reliable.

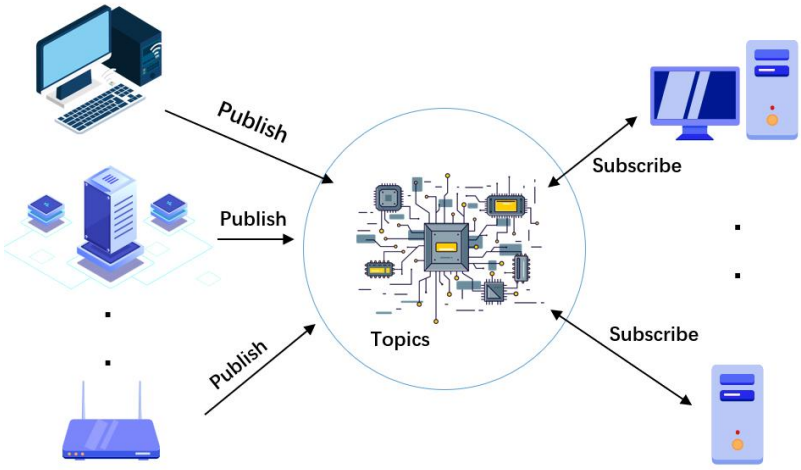


Figure 2.4: The Workflow of MQTT protocol

2.3 Blockchain Ecosystem

The concept of blockchain was first introduced by Satoshi Nakamoto in his white paper published in 2008 [17]. By employing an open distributed ledger, the network accurately records all transactions, significantly reducing the likelihood of altering block data without the majority consent of network participants. Operating independently of any centralized authority, blockchain integrates asymmetric encryption and distributed consensus, thereby establishing a transparent, secure, and scalable ecosystem.

2.3.1 Blockchain Structure

The blockchain functions as a decentralized ledger, known for its reliability due to the immutability of its data [18]. In a distributed environment, this tamper-resistant ledger ensures that data cannot be altered without following the predefined consensus protocol. Data within the ledger is organized in the form of transactions, which are grouped into blocks. The system continuously appends new blocks to the chain at the latest node. During this process, nodes organize transactions based on factors such as transaction fees or the order in which they are received and then perform hash calculations to determine the hash value for each transaction. If a transaction is tampered with, for instance by altering the address or amount, or if the transaction order is changed, the resulting Merkle tree root will be entirely different from the original. The first block in the chain, which has no parent block, is referred to as the genesis block [19]. Each block contains a timestamp, a nonce, and other identifying data. The basic process of blockchain formation can be described by Equation **??**.

$$Genesis \xrightarrow{ctgen} \{Block_1, Block_2, \cdots Block_n\} \tag{2.1}$$

The block body is where the majority of data is stored, including transaction data and smart contracts as shown in Figure 2.6. The verified transactions and transaction counter are both contained in the block body [20]. Consider a Bitcoin block as an example, a block may be up to 1MB in size, but a transaction is generally 250 bytes in size. Therefore, there is a limit of 4,000 transactions per block.

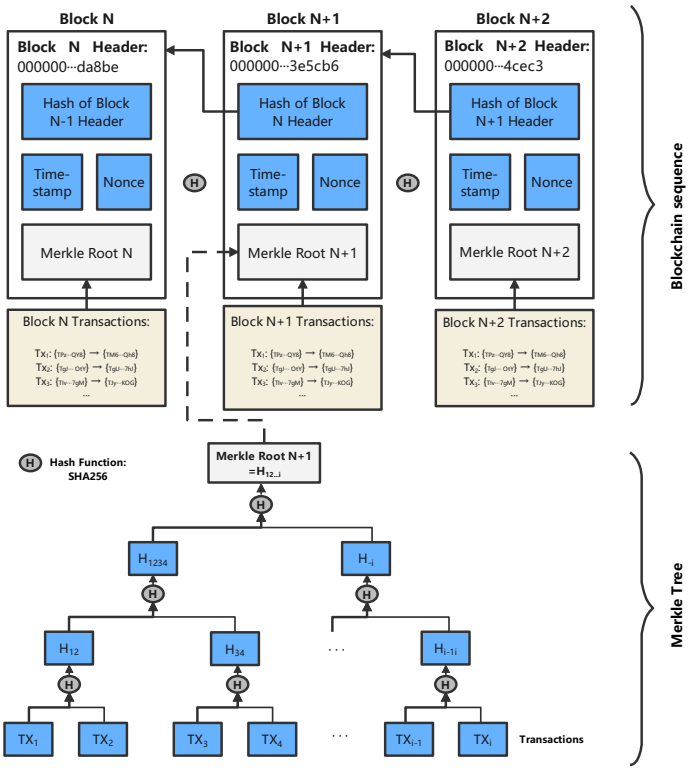


Figure 2.5: General structure of the blockchain

2.3.2 Blockchain Network

In blockchain networks, nodes serve as crucial elements embodying principles of equality, autonomy, and decentralization. Nodes can be added or modified within the system without causing disruptions. Each node continuously monitors the network for information broadcasts and, upon receiving new blocks or transactions from its neighbours, immediately validates the broadcasted message. This communication typically includes digital signatures, proof of work, hash values, blocks, and smart contracts [21]. To ensure the

integrity of data, messages are encrypted and added to the blockchain only after achieving consensus through a verification process.

2.3.3 Blockchain Encryption

Deploying blockchain requires setting up a peer-to-peer (P2P) network of devices or users, each ready to connect through the blockchain [22]. Each node generates a pair of cryptographic keys: a private key and a public key. Users exchange public keys to verify signed documents while keeping their private keys confidential to ensure security. To meet message security requirements, asymmetric cryptography is employed. Users keep their private keys secret and use the public keys of other users to verify document signatures. When a node initiates a transaction, it signs the transaction with its private key and broadcasts it for confirmation by peer nodes. The choice of consensus methods, which are verification methodologies, depends on the specific design objectives of the blockchain system.

2.3.4 InterPlanetary File System Storage

The InterPlanetary File System (IPFS) is an innovative hypermedia text transmission system, similar to a website that offers distributed storage [23]. The IPFS network uses a decentralized, encrypted storage system to store information. Files are divided into many pieces and stored on different network nodes. When a file is accessed or downloaded, the IPFS network instantly reconstructs it to its original state [24]. IPFS uses specific hashes to ensure data accuracy. Hashes are generated only for files submitted to the IPFS server, ensuring secure and high-integrity data storage. IPFS-based storage ensures the reliability, accessibility, and integrity of stored data. Additionally, IPFS storage is significantly less expensive than on-chain storage. Depending on the required storage amount, IPFS storage has a fixed monthly cost that is comparable to traditional storage solutions.

2.3.5 Consensus

In blockchain systems, there is no central authority responsible for maintaining ledger consistency across distributed nodes. Instead, consensus mechanisms are employed to ensure uniformity by verifying and encoding new transactions and blocks. These mechanisms must allow all participants to reach agreement in an untrusted environment while maintaining high fault tolerance [25]. Different systems utilize various consensus mechanisms, with Proof of Work (PoW), first implemented by the Bitcoin network, being both the earliest and most extensively used. In PoW, participants compete by using their computational resources to find the nonce value needed to generate a valid block. The difficulty of the mining process determines the effort required for consensus, with the difficulty value indi-

cating the approximate number of hash operations a node must perform to create a valid block. This serves as a critical benchmark for block production in the Bitcoin system. Bitcoin adjusts its difficulty approximately every two weeks to ensure that new blocks are generated at a consistent rate of about one every 10 minutes, regardless of changing network conditions [26]. The adjustment is made by recalculating the new target T, as shown in Equation 2.2.

$$D = D_{prev} \cdot \frac{t_{actual}}{2016 \cdot 10 \text{ mins}} \tag{2.2}$$

Where t_{actual} represents the time consumption of producing the previous 2,016 blocks, and D_{prev} is the former goal value. The generation of 2,016 blocks in less than two weeks implies an increase in overall computational capacity, necessitating an adjustment in the PoW difficulty. Additionally, various blockchain projects employ diverse consensus algorithms alongside PoW [27].

2.3.6 Transaction and Ledger

Transactions represent the fundamental form in which data is stored on the blockchain. When initiating a transaction, users typically allocate a transaction fee to the miner responsible for successfully creating a block. Miners are then tasked with bundling these transactions into the block they are generating. Each block can contain zero or more transactions. The unallocated transaction amount can be regarded as the transaction fee, as shown in Equation 2.3.

$$Inputs - outputs = Transaction_{fees} \tag{2.3}$$

As Figure 2.6 shows, the ledger consists of a state database, which keeps track of the most recent state, and a blockchain that stores immutable, ordered data in the form of blocks. Each transaction creates a set of key-value pairs representing assets, which are then added, modified, or removed from the ledger [28]. A transaction log with sorted transactions per block is constructed using a hash-linked block structure. In some blockchain implementations, new blocks (even empty ones) are continually added to maintain network security. The structure of the ledger and the cryptographic links between transactions are designed to ensure data integrity. Any modification to the ledger would compromise the entire hash chain, as the hash of each preceding transaction is embedded in the latest block,Äôs hash. This guarantees that all nodes maintain a consistent and stable state.

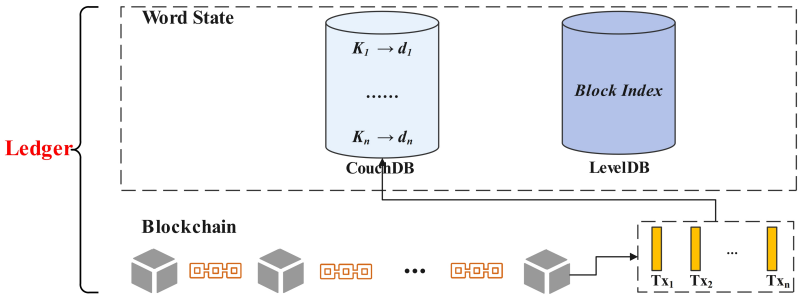


Figure 2.6: General structure of the blockchain

2.3.7 Smart Contract

Blockchain technology in its initial form, referred to as blockchain 1.0, emerged alongside the advent of cryptocurrencies such as Bitcoin [17]. The encrypted data on the blockchain has been used for various Industry 4.0 applications, including protecting sensor data. Blockchain technology has since advanced to the 2.0 era with the introduction of smart contracts [29, 30]. The Ethereum network supports the development of smart contracts, turning every computer into a part of a massively distributed, rentable machine. The emergence of smart contracts has further contributed to the creation of decentralized autonomous organizations (DAOs). Tools such as the web-based IDE Remix enable developers to create, exchange, and deploy Solidity-based smart contracts [31]. Typically, smart contracts are deployed on secure, permissioned networks, which eliminates the need for cryptocurrencies. This setup reduces the risk of malicious program infections and ensures more efficient and stable transaction processing. Blockchain-enabled smart contracts ensure the automatic execution of agreements between parties. For instance, Ethereum employs programming languages like Serpent and Solidity to construct these contracts. Once activated, the Ethereum Virtual Machine (EVM), running on a decentralized network, promptly executes the contract. Currently, contracts serve as the primary mechanism for deploying DApps and developing business process logic for various services [32].

2.4 Hyperledger Fabric

The Hyperledger Fabric is a componentised blockchain system, and as such is very flexible in terms of scaling and customisation, allowing for the construction of enterprise-scale applications. This section describes the basic components of the Hyperledger fabric system and the overall network architecture.

2.4.1 Node Configurations

In the Fabric system, key nodes comprise both physical and logical entities [33]. Physical nodes, such as Orderer, Peer, and CouchDB, are tangible components, while logical nodes like Committer, Endorser, and Leader represent various system functions. These physical nodes operate as distinct programs within Docker containers, each serving specific purposes. For instance, Orderer nodes manage consensus mechanisms, message broadcasting, and block distribution essential for ensuring data consistency. Peer nodes, segregated by organization, serve as data repositories, housing state, historical records, and block data. Additionally, the chaincode runtime node provides a Golang execution environment for smart contracts, functioning as a stateless, isolated environment. CouchDB, chosen for its data persistence, corresponds one-to-one with Peer nodes. Logically, Peer nodes are categorized into Committer, Endorser, and Leader functions, each serving unique roles in block validation, smart contract invocation, and block distribution, respectively.

2.4.2 Network Architecture

The Hyperledger fabric network is structured into two distinct layers, as shown in Figure 2.7. The first layer consists of Orderer nodes forming the Ordering Service Nodes (OSN) layer, where transactions achieve consensus for consistency and maintain data integrity throughout the blockchain system. The peer layer is organized according to organizational structures, with Peer nodes from the same organization forming a group within a P2P network, isolated from nodes in different peer layers. This layer also includes chaincode and CouchDB, which are invoked by Peer nodes to provide a richer set of functionalities.

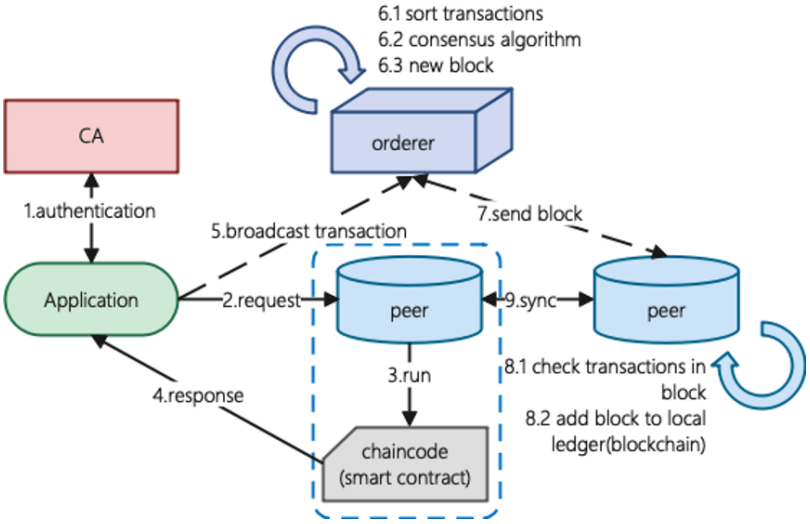


Figure 2.7: The network architecture of Hyperledger fabric

2.4.3 Transaction Life Cycle

Hyperledger fabric adopts a transaction processing model known as Execute, Order, Validate, and Commit, where clients interact with Peer nodes to submit contract invocation requests, Peers endorse the results, and the Orderer sequences transactions into blocks. In the implementation phase, clients invoke smart contract functions, Peers execute and endorse transactions, and the validation phase ensures data consistency. Additionally, the Orderer node manages message cache batches ($pendingBatch$) and message batches ($messageBatches$), distributing transactions accordingly before packaging them into blocks. Hyperledger fabric follows a transaction processing model consisting of four phases: Execute, Order, Validate, and Commit.

1. In the Execute phase, clients interact with endorsing Peers to submit contract invocation requests.
2. During the Order phase, the Orderer organizes endorsed transactions into blocks.
3. The Validate phase involves Peers verifying the correctness of transactions and ensuring consensus.
4. Finally, in the Commit phase, Peers write validated blocks to the ledger.

2.5 Cryptography Basics

Cryptography is a fundamental discipline in the field of information security, utilizing mathematical principles and computational algorithms to design and analyze secure communication protocols, ensuring confidentiality, integrity, authentication, and non-repudiation of data. It encompasses various encryption technologies, such as symmetric and asymmetric encryption, and hash functions, playing a crucial role in protecting digital information and online interactions.

2.5.1 Public Key Cryptosystem

The defining characteristic of public key cryptography lies in the use of two distinct keys, one for encryption and the other for decryption. Data is encrypted using the public key and decrypted with the private key. This method of asymmetric encryption ensures that, even with access to both the algorithm and the public key, decrypting the message without the private key remains computationally infeasible. Figure 2.8 describes the workflow of the public key cryptosystem. The process of encryption/decryption mainly consists of the following steps:

1. Recipient B generates a pair of keys $\langle PK_B, SK_B \rangle$ to encrypt and decrypt the message, where PK_B is the public key and SK_B is the private key.

2. Receiver B makes public the public key PK_B, while the private key SK_B is kept secret.

3. When sender A sends message m to B, the public key of B, PK_B, is used to encrypt m, expressed as $E_{PK_B}(m) = c$, where c is the ciphertext and E is the encryption algorithm.

4. After receiving ciphertext c, B decrypts it with its private key SK_B, which is expressed as $D_{SK_B}(c) = m$, where D is the decryption algorithm.

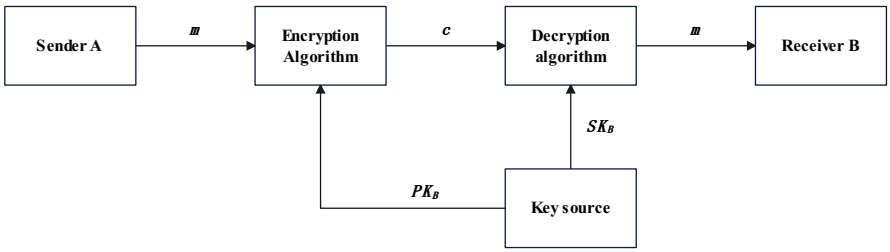


Figure 2.8: The encryption and decryption process of the public key system

Since only B keeps its private key, SK_B, no one else can decrypt the contents of ciphertext c. In general, the public key cryptosystem used in the above process should meet the following requirements:

1. The key pair $\langle PK_B, SK_B \rangle$ is computationally easy to obtain, and either key in the pair can be used for encryption, while the other is used for decryption.

2. The process by which sender A encrypts message m with receiver B's public key PK_B is computationally easy, expressed as $E_{PK_B}(m) = c$. At the same time, the decryption process $D_{SK_B}(c) = m$ by receiver B with its private key SK_B is computationally easy.

3. It is theoretically not feasible for an adversary to solve the corresponding private key SK_B based on B's public key PK_B, and it is also not feasible to solve the corresponding plaintext message m based on ciphertext c and public key PK_B.

2.5.2 Bilinear Mapping

Bilinear mapping is used to build specific types of cryptosystems and can be defined as functions that map elements of two vector spaces to elements of a third vector space. For

example, matrix multiplication, as a typical example of bilinear mapping, produces a new vector by a linear transformation of two vectors. In cryptography, bilinear mapping plays a crucial role in enhancing data security by establishing specific abstract mappings of data to safeguard information. It achieves this through two linear transformations: where the data represents the input, represents the linear transformation matrix, represents the offset vector, and is the output of the bilinear mapping.

2.5.3 Elliptic Curve Encryption

The Elliptic Curve Cryptosystem (ECC) offers enhanced security with shorter key lengths, leveraging the discrete logarithm problem of elliptic curves [34]. This section delves into fundamental concepts, starting with the definition and basic operations of elliptic curves. The elliptic curve λ defined over the finite field $GF(p)$ is expressed as:

$$y^2 = x^3 + ax + b \tag{2.4}$$

Here, p is a large prime number, and GF(p) represents a finite field with modulus p. The coefficients $a, b \in \mathrm{GF}(p)$ satisfy $4a^3 + 27b^2 \neq 0$. Explaining point addition on an elliptic curve involves a geometric interpretation: given points $P = (x_p, y_p)$ and $Q = (x_Q, y_Q)$, their sum $R = P + Q$ is obtained by intersecting a line passing through P and Q with the elliptic curve, yielding point R^*, and then finding the point of intersection with the X-axis, denoted as R. Scalar multiplication of points on an elliptic curve denoted as kP, is defined as repeated addition of P k times.

Elliptic curve discrete logarithm (ECDL) Problem. Given an elliptic curve E over a finite field GF(p), there is $Q = kP$, where $Q, P \in E$ and $k < p$. It is easy to compute Q for a given k and P, but difficult to compute $k \in \mathrm{GF}(p)$ for a given Q and p.

2.5.4 Hash Function and Message Authentication Code

In the cryptography research domain, the Hash function finds extensive application in integrity checks, digital signatures, message authentication codes, and other specific cryptographic tasks. In general, the hash function H is a public function that maps an arbitrary length of content M to a shorter, fixed-length value $H(M)$, called the hash value or summary of information for M. In addition, the hash function has the function of message validation, because any change in M will result in a change in the hash value $H(M)$. In general, hash functions should have the following properties:

1. The input of the hash function can be any length of content, and the length of the output value is fixed.

2. It is easy to calculate $h = H(M)$ given M, and computationally infeasible to find M based on h and $H(\cdot)$.

3. Given M, finding M^* such that $M \neq M^*$ and $H(M) = H(M^*)$ is computationally infeasible.

4. Finding any two distinct inputs, M and M^*, such that $H(M) = H(M^*)$ is computationally impractical.

To date, the widely used hash algorithms mainly include MD5 and secure hash algorithm (SHA-256). MD5 is often used to generate data fingerprints. This algorithm takes messages of any length as input and outputs a 128-bit message digest. On the other hand, the SHA algorithm boasts robust collision resistance, typically accepting input content of lengths up to 642 and generating a 256-bit message digest.

Message Authentication Code (MAC), also known as password checking or MAC, as a basic authentication technology, is widely used in communication. In this scheme, the communication key is used to generate a fixed-length check data block and attach the block to the message to realize data integrity verification. In this method, it is assumed that the communication key shared by A and B is K, and when A sends message M to B, A can calculate the verification code T of message M, as shown in Equation 2.5:

$$C(K, M) = T \tag{2.5}$$

In actual applications, message authentication codes are used to ensure the integrity and authenticity of messages. The receiver uses the shared key K to generate a new message verification code for the received messages and compares the code with the received MAC address. Assume that in the communication process, only the communication parties A and B are aware of the shared key K_1. The message and its corresponding MAC value are sent together to the receiver. Upon receiving them, the receiver recalculates the MAC for the received message using the same key K_1 and then compares this newly calculated MAC with the received one. If the two MAC values are equal, it indicates:

1. The message content has not been tampered with. The receiver needs to regenerate the MAC value based on the content of the message. If the message were maliciously altered during transmission, the received MAC value would not match the newly generated one.

2. The message is indeed from the legitimate sender. Since the shared key K_1 is only known to the communicating parties, if the two MAC values are equal, it confirms that the message originated from the true sender.

Thus, the receiver can verify the authenticity and integrity of the message by checking the message's authentication code. The MAC function, akin to the encryption algorithm, does not require the MAC function to be reversible.

2.5.5 Digital Signature

The message communication mode based on MAC can only ensure that the message comes from one of the communication parties. On the premise that the communication parties have not established a complete trust relationship, using only the message authentication code cannot exclude the disputes caused by mutual distrust. Digital signature technology can effectively solve this problem; digital signatures can be generated by encryption algorithms or specific signature algorithms. The digital signature algorithm should have the following properties:

1. It can verify the identity of the signer according to the content of the signature.
2. The digital signature can be verified by a third party to resolve disputes between the two parties.
3. The signature must be generated using the signature party's unique information to prevent signature forgery and denial.

A complete digital signature scheme typically encompasses key generation, digital signature generation, and signature verification algorithms. Currently, various digital signature algorithms, such as DSA, ElGamal, and Schnorr signature algorithms, play crucial roles in ensuring secure and reliable information transmission and storage schemes in the realm of information security.

2.5.6 Merkel Tree

Ralph Merkle named the binary hash tree the Merkle trees, which are data structures that use hash functions and are widely used to verify transactions in cryptocurrencies [35]. As shown in Figure2.9, the leaf nodes of the Merkle tree are metadata or Hash values of metadata, while the non-leaf nodes are hash values of child nodes in series. A Merkle root can be obtained through bottom-up pair-wise hashing. The Merkle tree has the following characteristics:

1. Any modification to the underlying data is propagated to the parent nodes through the hash layers, eventually affecting the Merkle root. This mechanism ensures that the integrity of multiple data pieces can be verified simultaneously in an efficient manner.

2. The binary tree structure allows for efficient querying, with the time complexity of verifying a leaf node being $\mathcal{O}(\log n)$. As a result, Merkle trees are particularly well-suited for managing large datasets.

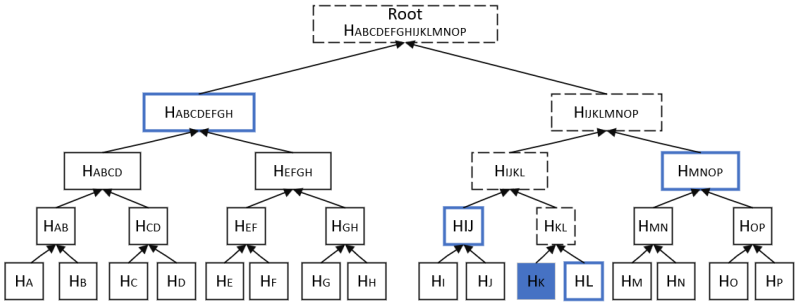


Figure 2.9: Merkle tree structure

In a Merkle tree comprising 16 leaves, the existence of a leaf node K can be demonstrated by constructing a path consisting of only $4 = \log_2 16$ hashes. The path dependencies of four hash values (blue labels) are respectively: H_L, H_{IJK}, H_{MNOP}, and $H_{ABCDEFGH}$. Along the four values in the path of the annotations (dotted line), is obtained by four Hash $H_{A\sim P}$, contrasting $H_{A\sim P}$ and $ROOT_K$ are consistent can check nodes.

$$H_{A\sim P} = \text{Hash}(\text{Hash}(\text{Hash}(\text{Hash}(H_K + H_L) + H_{IJ}) + H_{MNOP}) + H_{ABCDEFGH}) \quad (2.6)$$

2.5.7 Zero-knowledge Proof

The Zero-knowledge Proof (ZKP) system allows a knowledge holder to convince a questioner of the existence of certain knowledge without revealing any related information. In cryptographic terms, the knowledge holder, referred to as the prover (or Peggy), interacts with the questioner, known as the verifier (or Victor), in what is originally called an interactive ZKP. During this process, the verifier poses questions to the prover, who must respond correctly. If the prover gives any incorrect answers, the proof fails; however, if all answers are accurate, the interaction continues. As the exchange progresses through several rounds, the verifier becomes increasingly confident that the prover possesses the secret knowledge [36]. Typically, a Zero-knowledge Proof (ZKP) system with a set S involves an interactive process between a prover, executing a probabilistic polynomial-time strategy, and a verifier, using a computationally unbounded approach. The system adheres to the following principles.

1) Completeness: the prover with real knowledge can always be accepted by the verifier as Equation 2.7 shows.

$$\forall x \in L, \exists y \in \{0,1\}^* \text{ s.t. } \Pr\left[\text{Out}_V[P(x,y) \leftrightarrow V(x)] = 1\right] = 1 \tag{2.7}$$

2) Soundness: if the prover lacks the knowledge, then the verifier rejects the prover with a definitive advantage, as Equation 2.8 shows.

$$\forall P^*, \exists \text{ neg } \epsilon, \forall x \notin L, \forall y \in \{0,1\}^*, \Pr\left[\text{Out}_V[P(x,y) \leftrightarrow V(x)] = 1\right] \leq \epsilon(x) \tag{2.8}$$

3) Zero-knowledge: when the prover possesses the knowledge, the verifier can only conclude that the prover indeed has the knowledge without breaching the entire agreement, regardless of the methods employed by the verifier. However, the verifier cannot obtain any information regarding the knowledge itself, as depicted in Equation 2.9.

$$\begin{matrix}\{\text{View}_{V^*}\left[P(x,w) \leftrightarrow V^*(x)\right]\} \\ \{S(x)\}\end{matrix} \tag{2.9}$$

Non-interactive zero-knowledge proof: Interactive ZKP relies on repeated interactions between the prover and verifier to establish convincing proof with high probability. In contrast, non-interactive zero-knowledge proof (NIZKP) simplifies the process by reducing the interaction between the vendor and the verifier to one, thus eliminating the potential for collusion between the two parties, as shown in Figure 2.10. However, this approach may necessitate the involvement of a trusted third-party program to facilitate the verification process.

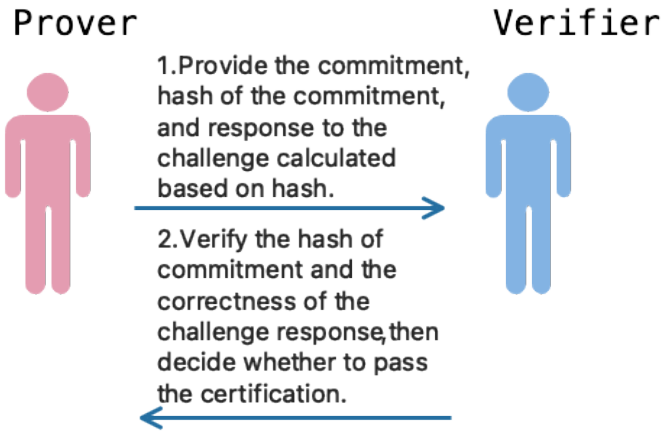


Figure 2.10: Non-interactive zero-knowledge proof

For example, in a grid consisting of nine squares, a third-party program determines which column or row to verify. Keeping the verification sequence confidential is crucial to prevent the verifier from passing it without possessing genuine knowledge.

2.6 Summary

In the discussion of Chapter 2, state-of-the-art practices and related technologies offer substantial benefits for data protection in the IIoT. However, these approaches also entail various challenges.

2.6.1 State of the Art

1. **Data Security and Privacy Protection:** Blockchain provides a decentralized ledger, ensuring data integrity and anonymity through cryptographic measures. This is essential for IIoT environments that require robust security and privacy, particularly in sectors like manufacturing and logistics where sensitive data is continuously exchanged. Advances in encryption techniques, such as those used in blockchain, have played a key role in enabling secure, distributed systems that safeguard industrial data.

2. **Efficiency and Transparency:** The use of smart contracts on blockchain platforms automates contract execution, ensuring transparency and immutability of transactions. By reducing the potential for fraud and human error, smart contracts streamline operational processes in IIoT systems, while maintaining trust between parties.

3. **Distributed Processing:** Blockchain, along with technologies like zero-knowledge proofs (ZKPs), enables nodes within IIoT networks to securely share data without compromising privacy. This improves data processing efficiency and enhances system-wide security, enabling more scalable and robust IIoT solutions.

4. **Consensus Mechanisms:** Consensus algorithms such as PoW and Proof of Stake (PoS) ensure trust and security in distributed IIoT networks by allowing all nodes to agree on a unified state. These mechanisms are crucial for maintaining data consistency and preventing malicious activities, forming the backbone of reliable blockchain systems.

2.6.2 Gaps and Issues

1. **Performance and Scalability:** While blockchain and smart contracts offer numerous benefits, they face significant performance and scalability challenges when dealing with large volumes of data. This is particularly critical for real-time industrial applications that require fast processing and minimal delays.

2. **Complexity and Cost:** The implementation and maintenance of blockchain technologies require specialized expertise and can involve high initial costs as well as

ongoing operational expenses. These factors can be a barrier to adoption, particularly for industries with tight budgets or limited technical resources.

3. **Standardization and Compatibility Issues:** The fragmented nature of IIoT devices and platforms creates challenges for interoperability. Without unified standards, integrating blockchain into diverse IIoT systems can be difficult, potentially limiting its broader application.

4. **Energy Efficiency and Latency:** Energyintensive consensus mechanisms, such as PoW, and the latency introduced by transaction validation processes can limit blockchain's use in real-time or resource-constrained IIoT environments. Addressing these inefficiencies is critical for making blockchain a viable solution for time-sensitive industrial applications.

In conclusion, although the current technologies provide robust tools for data protection in IIoT, addressing challenges including performance, cost, and standardization is necessary for widespread adoption.

Chapter 3

Blockchain-assisted Lightweight Data Storage

Contents

3.1 Introduction

IIoT connects physical objects (such as servers, gateways, and sensors) to virtual objects (such as data storage and processing services in the cloud) to form a highly complex network structure [37]. The devices connect and exchange data collected from the adjacent environment via the Internet to achieve real-time monitoring and intelligent scheduling of the industrial process [38]. Secure data storage is essential for subsequent data access, analysis, and sharing. However, as the scale and complexity of IIoT expand, the challenge of data storage security also increases [39, 40]. Reliance on external control centres for data management and device handling can lead to higher transaction and processing costs, potentially compromising system efficiency and stability [41, 42]. Additionally, IIoT systems are particularly vulnerable to cyber threats, with potential consequences ranging from data breaches and hardware damage to system-wide paralysis [43]. These security concerns pose a significant threat to the seamless operation of IIoT systems and can greatly impact the productivity and operational efficiency of enterprises [44]. Furthermore, IIoT devices and systems are prone to hacking threats, especially given their typically constrained storage capacity and computing power [45].

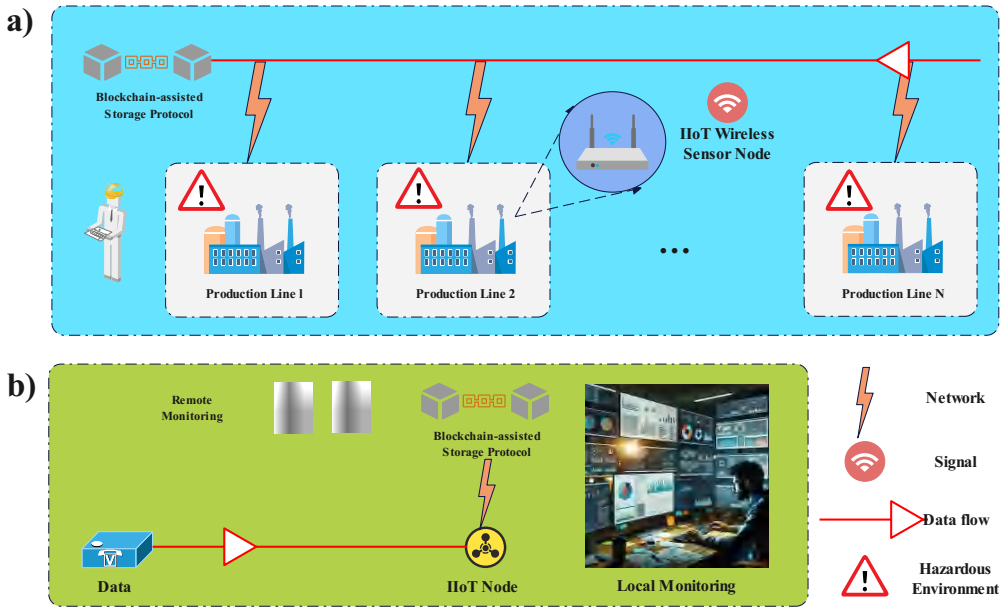


Figure 3.1: Structure of large-scale IIoT

Currently, there is a growing recognition of blockchain technology's effectiveness in enhancing data security within IIoT systems. The decentralized and immutable nature of blockchain provides a strong foundation for securely managing and transmitting industrial

data, making it an ideal solution for protecting sensitive information in IIoT environments. Firstly, the distributed ledger system of the blockchain provides a safe repository for the data generated by the IIoT device, which enhances security and transparency [46, 47]. Secondly, the blockchain-assisted communication protocol ensures the security communication between IIoT devices and the security transmission of critical data [48]. Moreover, blockchain's decentralized framework supports distributed data storage and processing, substantially reducing the risks associated with data breaches and system failures due to central point vulnerabilities [49, 50]. In addition to improving security, the decentralized structure greatly improves IIoT system decision-making and collaboration abilities.

On the other hand, the high computational consumption problem of blockchain is an important bottleneck in the current blockchain-based IIoT architecture for large-scale application and popularization due to inherent design flaws [51]. Many blockchain platforms, particularly public chain systems like Bitcoin and Ethereum, predominantly employ the PoW mechanism. However, the nature of the PoW mechanism depends on the hash operations of arithmetic competition, and a large number of nodes need to perform high-intensity calculations, resulting in a waste of resource waste and high energy consumption. This substantial computational requirement limits the processing capabilities of blockchain systems and raises operational costs for IIoT devices, creating major challenges in resource management and network efficiency [52].

To address the above challenges, this chapter proposes a trusted, low overhead, and lightweight secure storage scheme based on the blockchain mechanism. In this scheme, blockchain provides a real-time high level of confidentiality, privacy, and protection to the control system. In addition, this approach deploys asymmetric encryption and message authentication codes to achieve identity anonymity, traceability, and secure storage while reducing the computational overhead of the scheme.

3.2 Related Work

This section summarizes the related works. The blockchain-based IIoT network data security solutions and the further advancements are introduced in Section 3.2.1 and Section 3.2.2, respectively.

3.2.1 Blockchain-based Securing Data Solutions in IIoT Networks

By connecting industrial equipment to the Internet for data collection and analysis, IIoT faces significant security and privacy risks in cloud structures due to user or third-party managed encryption keys [53, 54]. To address these risks, Yu *et al.* [55] introduced a blockchain-driven Shamir threshold encryption framework STCChain, which can effectively

prevent data theft and secure keys, proving the effectiveness of cryptographic systems for IIoT data protection. Besides, to solve the node trust problem, Cha *et al.* [56] proposed a blockchain-based connectivity gateway to enhance node trust in IIoT, focusing on securely storing private user data within the blockchain. Qi *et al.* [57] proposed a tree-based data compression and hybrid access control for IIoT blockchains, enhancing efficient storage and selective data access. In addition, Huang *et al.* [58] explored a credit-based Proof of Work (PoW) mechanism, demonstrating its effectiveness in ensuring both security and transaction efficiency for IIoT data storage compared to general models. Liu *et al.* [59] proposed LightBlock, a lightweight structure that aims to simplify broadcast content and prevent the storage ledger in blockchain networks from growing indefinitely. Ma *et al.* [60] introduced BlockTDM, a customizable blockchain framework designed for trusted data handling in edge computing, incorporating mutual authentication protocols and leveraging smart contracts for data management. Meanwhile, *Kang et al.* [61] investigated the use of a consortium blockchain for secure data storage and sharing within vehicular edge networks, presenting a reputation-based data sharing model aimed at maintaining data quality

In summary, while blockchain technology is used to ensure the secure storage of IIoT data, current research has improved and refined the basic models by enhancing network consensus, constructing hierarchical blockchain structures, and introducing cryptographic verification mechanisms.

3.2.2 Advancements in Blockchain-Based IIoT Storage Solutions

Although the above research has made great progress, there is still room for improvement in addressing the high system overhead in blockchain-based storage schemes for IIoT. Wu *et al.* [62] developed MapChain-D, a dual-layered blockchain architecture, comprising a data chain linked to an index chain for optimized data storage and retrieval, tailored for IIoT contexts with storage and communication limitations. Addressing scalability and privacy challenges in blockchain-based access control for IIoT, a novel strategy incorporating transaction sharding was proposed in [63]. Furthermore, Zhang *et al.* [64] crafted a blockchain-enabled protocol for privacy-centric, multifactor device authentication in cross-domain IIoT, utilizing an on-chain accumulator for efficient, unlinkable identity verification of devices in various domains. Rathee *et al.* [65] developed an effective cybersecurity communication strategy that leverages a trust evaluation system and blockchain technology, ensuring accurate decision-making during message exchanges. Cui *et al.* [66] introduced an innovative and anonymous cross-domain authentication model leveraging blockchain technology. This model enhances security, scalability, and decentralized management. However, it faces challenges such as potential latency in cross-domain communication and the complexity of managing authentication credentials across multiple domains, which can increase

operational costs and management overhead. Xu *et al.* [67] proposed a blockchain-based big data sharing framework tailored for resource-limited edge devices. This framework introduces a Proof-of-Collaboration (PoC) consensus mechanism, significantly reducing computation complexity and storage overhead, thus making it particularly suitable for edge devices with low computational capacity. Moreover, the framework enhances communication efficiency through transaction filtering and offloading schemes, and by introducing new types of blockchain transactions and blocks. Integrating blockchain, encryption algorithms, and cloud storage technology to achieve the storage and sharing of industrial data was proposed in [68, 69].

However, despite these advancements, challenges such as system overhead, latency, and complexity in credential management still need to be addressed to achieve more efficient and scalable IIoT solutions.

3.3 System Design

This section introduces the data structure in the model definition, the private data resource storage model, and the relationship between the models.

3.3.1 System model

Figure 3.2 shows the data storage model proposed in this manuscript mainly includes Trusted Registration Authority (RA), Data Provider (DP), End Users (EUs), Blockchain (BC), and Data Storage Server (DS).

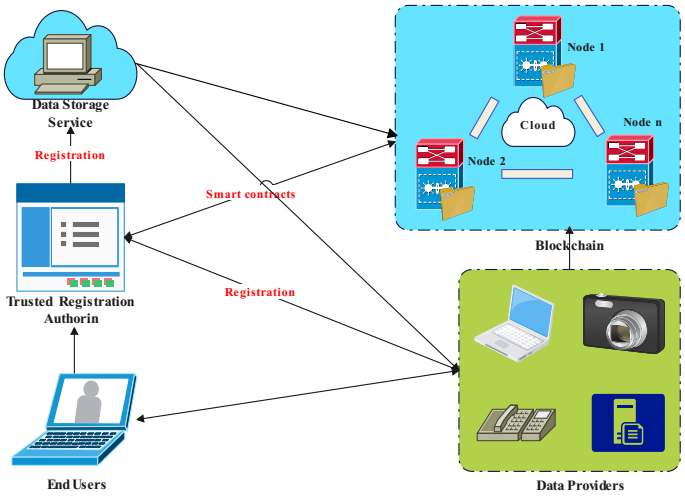


Figure 3.2: The structural relationship of the proposed system model

3.3.1.1 Registration Authority (RA)

Initially granted high-level trust privileges, the RA handles registration requests from system components, generates key pairs for authorized users, and invokes smart contracts to record registration information. Since the RA generates the component's public key based on the identity information of the EUs, only the RA can reveal the real identity of the entity. However, the RA is restricted from altering data within the block, as this would require bypassing the established consensus mechanism.

3.3.1.2 Data Provider (DP)

This component acts as a provider of IIoT data, usually an edge server, tasked with accumulating and synthesizing industrial data from a diversity of intelligent devices. As an autonomous node, DP joins the blockchain network as an independent node and can maintain communication with remote storage services.

3.3.1.3 End User (EU)

This component is intelligent terminal equipment, such as personal computers, smartphones, industrial control devices, and authorized user terminals that can download and audit stored industrial data.

3.3.1.4 Blockchain (BC)

Blockchain consists of data provider DPs interconnected through a network, BC provides a decentralized storage environment, and industrial data stored in BC cannot be modified.

3.3.1.5 Data Storage Service (DS)

This component is responsible for storing complete industrial data to maintain the limited storage resources of the blockchain, and IPFS, as a kind of distributed storage system for data storage, has a larger storage level than the blockchain.

3.3.2 Model Assumptions

This model is based on the following pre-conditions:

- DS functions as an external storage system, accountable for preserving comprehensive industrial data. Correspondingly, BC stores the hash values of this data, ensuring alignment between DS records and their respective hashes in BC.

- Data access within BC and DS is restricted solely to authorized entities, and RA holds the authority to revoke access privileges of any entity deemed malicious through smart contract invocation.

- All the interactions involving RA, EU, and DP are safeguarded via secure channels, such as Transport Layer Security (TLS), thereby precluding the possibility of malicious entities intercepting communication content.

3.3.3 Design Goals

Through the model design, this manuscript aims to achieve the following design objectives.

- Identity anonymity: this scheme is designed to safeguard the identity confidentiality of authorized nodes, ensuring that unauthorized or malevolent users are unable to discern the true identity of any entity involved.

- Secure storage: ensuring the integrity and confidentiality of stored IIoT data from malicious tampering.

- Traceability: tracking and punishing the behaviour of malicious nodes is an important goal of this scheme, and it should be ensured that authorized entities can detect and track the malicious behaviours of nodes and their true identity.

- Efficiency: due to the need for timely analysis and storage of massive IIoT data, the system overhead for calculation and storage of the solution should be kept at a low level.

3.4 Model Deployment

This section provides a comprehensive overview of the suggested framework. A detailed list of key notations employed throughout this manuscript is presented in Table 3.1.

3.4.1 System Initialization

The RA is responsible for the initialization setup of the system, and the initialization process includes the basic initialization of the system and the setup of the blockchain system.

3.4.1.1 Initialization of System Parameters

RA completes the initialization of system parameters as follows.

Table 3.1: Key Notations

Notation	Description
P_{pub}	Public Key of the System
$Pubk_i$	Public Key of Entity i
$HMAC\,(.)$	Hash function
$Encrypt\,(.)$	Encryption function
$Decrypt\,(.)$	Decryption function
EU_i, DP_j	End User i, Data Provider j
PID	Anonymous certificates
RA	Registration Authority
SK	Session Key
σ_i	Anonymous credentials for entity i
$RC_{(x)}$	Storage credentials for content x
$CT_{(x)}$	Encrypted content of x
$HV_{(r_{i,j})}$	Hash values of IIoT data $r_{i,j}$
$ST_{(r_{i,j})}$	Encrypted Content of IIoT Data $r_{i,j}$

- RA identifies a sufficiently large prime number q to establish a cyclic group G of order p. Additionally, RA selects a hash function H, mapping from the set $\{0,1\}^*$ to the group Z_q^*.
- RA selects an asymmetric encryption function $Encrypt(\cdot)$ and a message authentication code generator $HMAC(\cdot)$.
- RA picks a random number s from the set Z_q^* and calculates the public key P_{pub} as the product of s and P.
- RA announces the system parameters as $P = \{H, Encrypt(\cdot), HMAC(\cdot), P_{pub}\}$.

3.4.1.2 Initialization of Blockchain

Blockchain initialization is primarily executed by the trusted entity RA, involving the following steps.

- RA creates configuration files, such as genesis.json, necessary for the blockchain setup.
- RA securely transmits these configuration parameters in encrypted form to the authorized node DP.

Considering the absence of complex digital currency management mechanisms in Hyperledger fabric and its alignment with the Internet of Things application scenarios, Hyperledger fabric is selected as the blockchain platform.

3.4.2 Entity Registration

In the proposed architecture, all entities must undergo a registration process with the RA, affirming the corresponding authentic identities. Focusing on the entity EU_i as a representative example, the registration procedure unfolds as follows.

- EU_i sends its identity information $Info_i$ to RA through secure channels, RA checks if EU_i has been registered.
- RA encrypts the received message $Info_i$ as Encrypt $(P_{pub},\ \text{Info}_{\ i} \| Ts_i) \rightarrow CT_{(\text{Info}_{\ i})}$ and generates a anonymous identity as $\sigma_i = CT_{(Info_i)} * P$.
- RA determines a random value r_i within the group Z_q^* and computes x_i as $r_i \oplus (s^* H(CT_{(\text{Info}_i)} || P_{pub}))$. Subsequently, it calculates the public key $Pubk_i$ as $x_i \cdot P$.
- For EU_i, RA selects x_i as its private key and dispatches the data trio $(\sigma_i, x_i, Pubk_i)$. Upon receiving $(\sigma_i, x_i, Pubk_i)$, EU_i verifies the equation $x_i^* P = Pubk_i$ to confirm the authenticity of the received keys.
- RA calls the Algorithm 1 to upload defined data entries $(NULL, \sigma_i, CT_{\text{Info}_{\ i}}, Pubk_i)$.

Algorithm 1 registSC(·)

% *Invoked by the RA to update KMIS.*
if (owner!= msg.sender) return 0;
Struct KIM {
byte σ, String CT, String Pubk, DateTime TS
}
Public KIM $KIMs[MaxNum]$;
H $(\delta_{\text{i}} \| \text{CT} \| \text{Pubk}_{\text{i}}) \rightarrow$ Key ;
If exists(Key, KIMs) **then**
  return Error;
End If
$KIM\_Obj =$ **new**$KIM(\delta_i,\ \text{CT},\ \text{Pubk}_{\ i},\ \text{TS})$;
$KIMs.add(Key \Rightarrow KIM\_Obj)$;
return OK;

3.4.3 Entity Authentication

To ensure robust system security, mutual authentication plays a critical role. This concept is illustrated by examining the mutual authentication process between DP_j and EU_i as follows.

1. EU_i forms the symmetric key as $SK_{ij} = x_i \cdot Pubk_j$ and calculates the hash $sig_i = H(\sigma_i \parallel Pubk_i \parallel t_i)$. The message is then encrypted to $E(SK_{ij}, \{\sigma_i, Pubk_i, t_i, sig_i\}) \rightarrow CT_{(Msg_i)}$. EU_i also encrypts SK_{ij} using DP_j's public key: $Encrypt(Pubk_j, SK_{ij}) \rightarrow CT_{(SK_{ij})}$. Finally, EU_i transmits to DP_j: $\{CT_{(SK_{ij})}, CT_{(Msg_i)}\}$.

2. Upon receipt of $\{CT_{(SK_{ij})}, CT_{(Msg_i)}\}$, DP_j retrieves the decryption key as $Decrypt(sk_j, CT_{(SK_{ij})})$ and accesses the message $\{\sigma_i, Pubk_i, t_i, sig_i\}$.

3. DP_j verifies the time validity with $|t_{now} - t_i| \leq \Delta t$, where Δt is the time threshold. It then calculates the hash $H(\sigma_i \parallel Pubk_i \parallel t_i) \rightarrow sig_i^*$ and confirms the authenticity by checking $sig_i^* \stackrel{?}{=} sig_i$, validating the received data.

4. DP_j constructs the symmetric key $SK_{ji} = x_j \cdot Pubk_i$ and hashes the message content $sig_j = H(\sigma_j \parallel Pubk_j \parallel t_j)$. Then, DP_j encrypts the message as $E(SK_{ji}, \{\sigma_j, Pubk_j, t_j, sig_j\}) \rightarrow CT_{(Msg_j)}$. DP_j encrypts SK_{ji} with the public key of EU_i: $Encrypt(Pubk_i, SK_{ji}) \rightarrow CT_{(SK_{ji})}$. Eventually, DP_j sends $CT_{(Msg_j)}$ and $CT_{(SK_{ji})}$ to EU_i as: $DP_j \rightarrow EU_i$: $\{CT_{(SK_{ji})}, CT_{(Msg_j)}\}$.

5. EU_i will verify the validity of the received $CT_{(Msg_j)}$ according to *Step 2* and *Step 3*. A secure session can only be created when both EU_i and DP_j authentication are successful. Furthermore, based on the above interaction steps, it can be seen that $SK_{ij} = x_i \cdot Pubk_j = x_i \cdot x_j \cdot P = x_j \cdot x_i \cdot P = x_j \cdot Pubk_i$. Therefore, the session key can be further defined as Equation 3.1.

$$SK = x_i \cdot x_j \cdot P \tag{3.1}$$

3.4.4 Data Storage

Considering an edge device DP_j that acquires sensory data $r_{i,j}$, the associated data storage procedure is presented as follows.

1. DP_j runs the ECDH protocol to establish a secure session with the storage service DS and follows Equation 3.1 to establish the session key SK with DS.

2. DP_j encrypts data $r_{i,j}$ according to Equation 3.2 and generates ciphertext $CT_{(r_{i,j})}$.

$$CT_{r_{i,j}} = Encrypt(Pubk_j, \{\sigma_j, r_{i,j}\}) \tag{3.2}$$

Where $Pubk_j$ is the public key of DP_j, and σ_j is an anonymous credential.

3. DP_j computes the hash $HV_{(r_{i,j})}$ of $\{\sigma_j, r_{i,j}, t_{now}\}$ as Equation 3.3.

$$HV_{r_{i,j}} = HMAC(SK||x_j, H(\sigma_j||r_{i,j}||t_{now})) \tag{3.3}$$

Subsequently, DP_j uses the session key SK to encrypt the data according to Equation 3.4 and submit it to DS through a secure channel, that is $DP_j \rightarrow DS : CT_{(session)}$, obtain the returned storage credentials $RC_{(r_{i,j})}$.

$$CT_{(session)} = Encrypt(SK, (CT_{(r_{i,j})}||HV_{(r_{i,j})})) \tag{3.4}$$

4. DP_j calls the smart contract **storeSC**$(\cdot)$ (i.e., Algorithm 2) to store the hash value $HV_{r_{i,j}}$ as the data fingerprint. It should be noted that only authorized individuals can call the smart contracts.

5. DP_j obtains the storage key $Key_{(r_{i,j})}$ of the data hash $HV_{r_{i,j}}$ in the smart contract, and then constructs a data item $< Key_{(r_{i,j})} \Rightarrow RC_{(r_{i,j})} >$ to describe the mapping relationship between blockchain transactions and offline data, achieving collaborative consistency of data between blockchain and cloud.

Algorithm 2 storeSC$(\cdot)$

```
Struct KIM {
    byte σ, String Pubk, String HV, String RC, DateTime TS
}
Address owner = AutoGet(account);
If (owner!= msg.sender) return 0;
mapping(string ⇒ IIoTData ) mapData;
Obj = new IIoTData (Pubk,σ,HV,RC);
H(HV||Pubk_i) → Key;
If (exists(mapData[Key]) == true) then
    return True;
End if
mapData.add(Key ⇒ Obj);
return True;
```

The integrity verification of $r_{i,j}$ can be achieved through a given tuple $Q =< RC_{(r_{i,j})}, Pubk_j >$. First, use the stored credentials $RC_{(r_{i,j})}$ of the data in DS to retrieve the data mapping table $MTable_{(bc \rightarrow db)}$, obtaining the storage key $Key_{(r_{i,j})}$ in $mapData$. Subsequently, get the stored hash as $mapData[Key_{(r_{i,j})}].HV \rightarrow SHash$. Finally, calculate the raw hash of the stored $r_{i,j}$ as $H(Decrypt(SK, CT_{(r_{i,j})})) \rightarrow Ohash$, and determine whether $r_{i,j}$ has been tampered by verifying whether $Ohash == Shash$.

3.4.5 Permission Revocation

The data provider DP_j generates and provides various sensing data to the system, and EU_i provides feedback on data quality to RA as $bh_{i\to j} \in (0,1)$. Assume that RA gathers the feedback information about DP_j is $RatingSet = \{bh_{1\to j}, bh_{2\to j}, \cdots, bh_{n\to j}\}$, and RA calculates the comprehensive evaluation about DP_j during $[t_l, t_h]$ as $Rs_j = \sum_{i=1}^{n} bh_{i\to j}$. If Rs_j is below the selected threshold $\eta \in Z_q^*$, it indicates that the data quality provided by DP_j is low. RA should revoke the data storage permission of DP_j, and the permission revocation process (as shown in Algorithm 3) mainly includes the following steps.

1. RA constructs a permission revocation transaction as $\mathrm{Tx}_{(\mathrm{revoke})} = \{\delta_{\mathrm{j}}, \mathrm{Rs}, [\mathrm{t_1}, \mathrm{t_h}], \mathrm{Sig_{RA}}, \mathrm{ts_{now}}\}$, and broadcasts $Tx_{(revoke)}$ to the blockchain member nodes $\{EU_1, EU_2, \cdots, EU_n\}$.

2. EU_i sends the feedback message $Feedback_{(Tx)}$ to RA, RA will revoke the identity of DP_j based on the received feedback message $Feedback_{(Tx)}$. To be specific, RA constructs an identity revocation blockchain ledger $revokeLedger$, and further set $\mathrm{revokeLedger}[\mathrm{H}(\delta_{\mathrm{j}})] = True$.

3. RA revokes the identity ciphertext $CT_{(Info_j)}$ by using the identity δ_j, and performs decryption operation as $\mathrm{Decrypt}\left(\mathrm{s}, \mathrm{CT}_{(\mathrm{Info_j})}\right) \to \mathrm{Info_j}$. Subsequently, RA broadcasts the real identity $Info_j$ to various participants in the system.

Algorithm 3 revokeSC($\cdot$)

Input: Rs, δ_j, $revokeLedger$
Output: True or False
If $(Rs \geq \eta)$ **then**
  return False;
End if
$\mathrm{Tx}_{(\mathrm{revoke})} = \{\mathrm{Rs}, \delta_{\mathrm{j}}, (\mathrm{t_l}, \mathrm{t_h}), \mathrm{Sig_{RA}}, \mathrm{TS_{now}}\}$;
$\mathrm{RA} \to \{\mathrm{EU_1}, \mathrm{EU_2}, \ldots, \mathrm{EU_n}\} : \mathrm{Tx}_{(\mathrm{revoke})}$;
$\mathrm{EU_i} \to \mathrm{RA} : \mathrm{Feedback} = \{\delta, \mathrm{status}, \mathrm{Sig_i}, \mathrm{TS_{now}}\}$;
If $(Feedback_{(1,2,\cdots i)} \geq \omega)$ **then**
  $\mathrm{revokeLedger}[\mathrm{H}(\delta_j)] = \mathrm{True}$;
  $\mathrm{ObtainInfo}(\delta_{\mathrm{j}}, \mathrm{Sig_{RA}}) \to \mathrm{CT}_{(\mathrm{Info_j})}$;
  $\mathrm{Decrypt}\left(\mathrm{s}, \mathrm{CT}_{(\mathrm{Info_j})}\right) \to \mathrm{Info_j}$;
  $\mathrm{RA} \to \{\mathrm{EU_i}, \mathrm{DS}, \mathrm{DP_i}\} : \{\mathrm{Info_i}, , \mathrm{Sig_{RA}}, \mathrm{ts_{now}}\}$;
End if
return True;

3.5 Security Analysis

This section gives an analysis of the security requisites and potential network attacks of the solution based on the previously established security criteria.

3.5.1 Identity Anonymous

In this proposed framework, each EU_i utilizes an anonymized identifier σ_i, substituting their real identity for system interactions. This protocol facilitates anonymous communications through the formula $\sigma_i = CT_{(Info_i)} \cdot P$, thereby masking the actual identity. This can be represented mathematically as: $\sigma_i = Anonymize(CT_{(Info_i)}, P)$. The anonymity assurance hinges on the fact that only the trusted authority RA can decode the true identities, denoted as $RA_{decode}(\sigma_i) = TrueIdentity$. Consequently, entities within the system maintain their anonymity effectively.

3.5.2 Data Integrity

Data integrity is crucial for maintaining trust and reliability within the system. The proposed framework leverages cryptographic methods, including digital signatures and hash functions, to ensure the integrity of transactions and prevent unauthorized alterations. Let Tx_i denote a transaction signed by entity EU_i. The digital signature of Tx_i is represented as $Sig_{EU_i}(Tx_i)$. The inclusion of digital signatures in transactions ensures that any tampering with the transaction content can be detected, as the signature verification process will fail if the data has been altered. Moreover, hash functions are employed to generate unique identifiers for data blocks. Let $H(data)$ denote the hash value of a data block. By storing and comparing hash values of data blocks, the system can quickly identify any changes to the data. If the hash value of a data block differs from the expected value, it indicates potential tampering or unauthorized modifications.

3.5.3 Condition Traceability

This framework effectively guarantees the anonymity and unlinkability of its components, ensuring that external attackers are unable to track the activities of any entity based on behavioural data, thereby safeguarding privacy and security. Concurrently, the trusted authority RA possesses the capability to ascertain the actual identity of any entity, denoted as EU_i, by utilizing anonymous credentials σ_i. In mathematical terms, RA retrieves the encrypted identity information $CT_{(Info_i)}$ from σ_i and the public parameter P, and then decrypts it using its private key s. This process expressed as $RA_{decrypt}(CT_{(Info_i)}, s)$, enables RA to monitor and trace the activities of any system entity and unmask the identities of malicious actors.

3.5.4 Analysis of Network Attack Resistance

1) Impersonation Attacks. If malicious attacker A attempts to impersonate verified entity u_i, they need to obtain the corresponding private key $x_i \in Z_q^*$. The private key x_i, which resides within the set Z_q^*, is meticulously crafted by RA using a securely generated random number r from the same set, intertwined with robust hash algorithms. This intricate construction ensures that an external entity, like attacker A, is incapable of replicating or simulating the creation of the random number r within Z_q^*. Consequently, it becomes infeasible for A to autonomously fabricate a legitimate private key, thereby safeguarding the integrity of the private key generation process. If A uses anonymous identity σ_i to impersonate the entity u_i, then in subsequent message communication, it is also necessary to provide information signed by the private key x_i, but A cannot generate the legitimate private key.

2) Replay attacks. Replay attack refers to the attacker attacking the receiving host by intercepting legitimate messages and resending messages that have been confirmed by the receiving clients. To resist replay attacks, this scheme embeds the current timestamp ts_{now} in message communication, and any entity u_i can detect whether the message has expired. Specifically, assuming $ts_{receive}$ as the message reception time, π represents the estimated message propagation delay, it can be determined whether it is a replay message by checking $|ts_{(receive)} - ts_{(now)}| \leq \pi$.

3) Sybil attacks. Sybil attacks, denoted as A_{Sybil}, target the data redundancy backup mechanisms, $B_{\text{redundancy}}$, of distributed systems, S_{dist}. These attacks exploit virtualization technology to fabricate multiple virtual nodes, V_i, from a single hardware device, H_{single}, thereby manipulating the data distribution mechanism, D_{dist}, of S_{dist}. This phenomenon can be represented as $A_{\text{Sybil}} : H_{\text{single}} \rightarrow \{V_1, V_2, \ldots, V_n\}$. In the context of a blockchain system, B_{chain}, which operates on a peer-to-peer (P2P) distributed network N_{P2P}, there exists a notable vulnerability to Sybil attacks impacting the system's integrity. To mitigate this, each device node, D_i, in the network must provide unique authentication credentials, $C_{\text{auth}}(D_i)$, serving as an identifier for participating in N_{P2P}. The role of a Resource Authority (RA), denoted as R_{A}, is crucial in this regard, as it is responsible for recording (R_{record}) and verifying (R_{verify}) these credentials. This mechanism can be represented as $R_{\text{A}}(R_{\text{record}}, R_{\text{verify}}) \rightarrow N_{\text{P2P}}$, ensuring that a single device cannot undermine the decentralized nature of B_{chain} through the creation of virtual member nodes, thereby preserving the system's integrity against A_{Sybil}.

4) Session key attack. To acquire the appropriate session key, K_{session}, attackers A must concurrently procure the private keys, $K_{\text{priv}}^{(i)}$ and $K_{\text{priv}}^{(j)}$, of both communicating entities, E_i and E_j. These private keys are generated as random numbers, R_i and R_j, by the Resource Authority (RA), denoted as R_{A}. The generation process can be expressed as $K_{\text{priv}}^{(i)} = H(R_i)$

and $K_{\text{priv}}^{(j)} = H(R_j)$, where H is a secure hash algorithm. These keys are transmitted over a secure channel, represented as $T_{\text{secure}}(K_{\text{priv}}^{(i)}, K_{\text{priv}}^{(j)})$. For A to obtain K_{session}, they must overcome the cryptographic challenge associated with the Elliptic Curve Digital Signature Algorithm (ECDSA), symbolized as C_{ECDSA}. The probability P_A of A successfully solving C_{ECDSA} and thereby acquiring a valid K_{session} is extremely low, mathematically represented as $P_A(K_{\text{session}}|C_{\text{ECDSA}}) \approx 0$. Therefore, unless A can resolve C_{ECDSA}, the likelihood of obtaining a valid K_{session} is negligibly small.

3.6 Performance Evaluation and Analysis

This section first describes the experimental configuration details in Section 3.6.1. Subsequently, it formulates a comparative analysis with analogous research efforts, followed by a thorough examination of the performance metrics of the proposed scheme, as shown in Section 5.2.

3.6.1 Basic Configurations of the Experiments

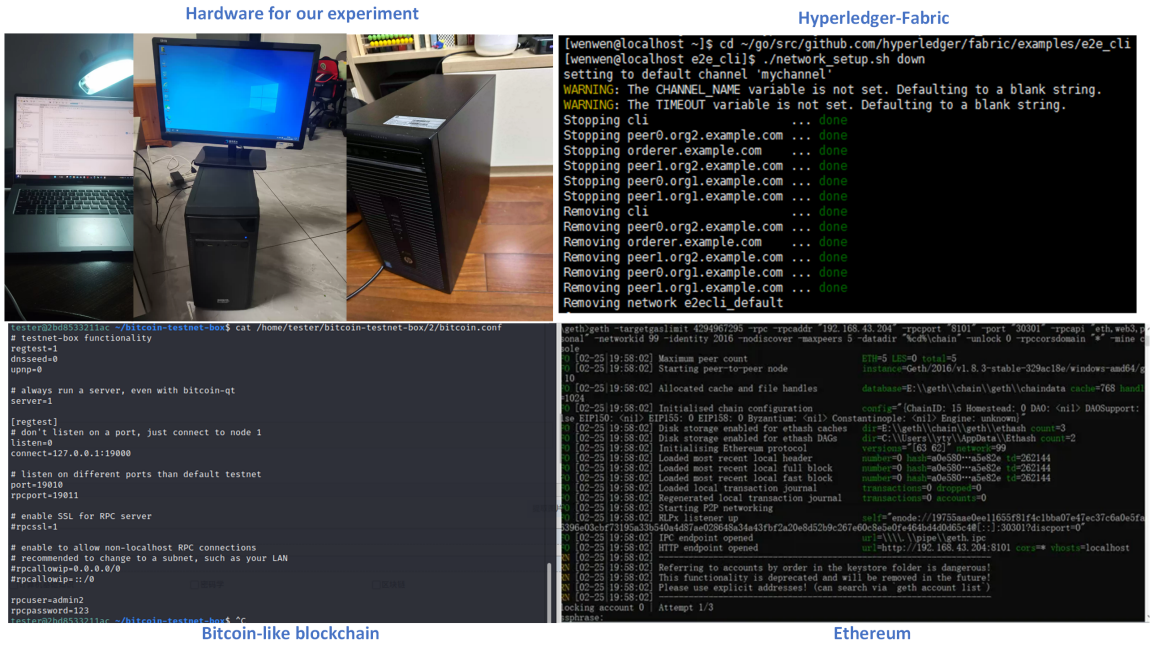


Figure 3.3: The underlying blockchain platforms

In this manuscript, a prototype system of the design solution is implemented using Hyperledger-fabric 1.4. The platform leverages Golang to develop smart contracts and utilizes Hyperledger Caliper, a professional client tool, for secure management, monitoring, and ensuring

programmability and scalability. The experimental solution deploys a blockchain consisting of 15 authorized nodes using a desktop computer as a hardware platform running on the Deepin/Linux operating system using Core(TM) i5-11300H as a processor with 16.0 GB RAM. Following the deployment of smart contracts, a Node. js-based client is used to evaluate their performance. Additionally, blockchain platforms are utilized as the foundational systems, with their operating status shown in Figure 3.3.

3.6.2 Simulation Results and Analysis

The deployed smart contract is tested for performance using a Node.js client (Algorithms 1, 2, and 3 are the subject of this testing). Table 3.2 presents a systematic tabulation of the evaluation's results. It is important to emphasize that the stages of transaction generation, validation, and synchronization are included in the measured runtime. Each test iteration is carried out 100 times to guarantee correctness, allowing an average elapsed time to be calculated for a thorough evaluation.

Table 3.2: Smart contract execution time (ms)

Smart contract	**Maximum duration**	**Minimum duration**	**Average duration**
$registSC(\cdot)$	4628	2491	3559
$storeSC(\cdot)$	3487	2245	2866
$revokeSC(\cdot)$	3876	2147	3012

Moreover, building on the experimental framework outlined in this manuscript, further tests were conducted on the time consumed by key cryptographic operations within the scheme. A 10KB file was used as the benchmark data for these tests. The outcomes are concisely displayed in Table 3.3, providing a tangible insight into the computational capabilities of the current experimental setup. To further assess the performance of this scheme, including the computational and communication costs, this section lists the execution time of the main operations or the bit length of data units as shown in Table 3.4.

Table 3.3: The time cost of encryption operations (ms)

Arithmetic	**Maximum duration**	**Minimum duration**	**Average duration**
SHA-256	246	121	183
RSA encryption	458	107	282
AES-128 encryption	346	205	275
ECC signature	475	193	334

Considering that critical operations such as cryptography and smart contracts will cause computational overhead for the system, this section compares the design scheme with the

Table 3.4: Opration execution duration

Symbol	**Operation Description**	**Duration/Bits**
T_{bp}	Bilinear Pairing	25.3 ms
T_{Gm}	Scalar Multiplication on G	12.2 ms
T_{mbp}	Multiplication in G_T	21.5 ms
T_e	Modular Exponentiation on Z_q	13.6 ms
T_h	General Hash Function	34.8 ms
$\|Info_i\|$	Bit Length of Registration Information	9032 bits
$\|CT_{(x)}\|$	Size of Encrypted Data x in Bits	4096 bits
$\|\sigma_i\|$	Bit Length of Anonymous Identifier	128 bits
$\|x_i\|, \|Pubk_i\|$	Length of Private and Public Key Bits	512 bits
$\|ACK\|$	Bit Length of *ack* Message	256 bits

schemes in references [60, 61, 67]. According to the experiment results in Table 3.5, the computational overhead of this scheme is only higher than that of the scheme in Ref [67], due to the additional invocation overhead caused by the application of smart contracts for data storage tasks.

Table 3.5: The comparisons of time consumption (ms)

Programmatic	Computational overhead
Xu et al. [67]	$T_{bp} + 2T_{Gm} + 2T_h \approx 119.3$
Ma et al. [60]	$T_{bp} + 2T_{Gm} + 2T_e + 3T_h \approx 181.1$
Kang et al. [61]	$2T_{bp} + T_{Gm} + 2T_e + T_{mbp} + 4T_h \approx 250.7$
This manuscript	$T_h + 2T_{bp} + T_{G_m} + T_{\text{storesc}} \approx 127.6$

The experiment focuses on evaluating the entity registration and data storage stages, informed by the cryptographic operations involved in the discussed schemes. Figure 3.4 shows the average elapsed time of each scheme for simulating 100 to 1,000 registration requests under the condition of setting the registered data to be 100 KB. It highlights a consistent trend where the average latency in processing registration requests escalates as the volume of requests increases. Specifically, within the scope of our proposed scheme, the processing delay gradually increases, starting at 245.3 milliseconds for a batch of 100 requests and rising to 431.6 milliseconds as the request count reaches 500. Meanwhile, the corresponding request processing delay of Ma et al. [60] increases from 368.9 ms to 547.3 ms. Overall, the time consumption of our scheme in the identity registration phase is at a relatively low level, and the time cost of our scheme is just from 245.3 ms to 643.7 ms. The main reason is the use of more concise cryptographic operation steps and smart contracts (i.e. registSC($\cdot$)) to store the registration information of entities, which reduces the overall

computational consumption of the scheme.

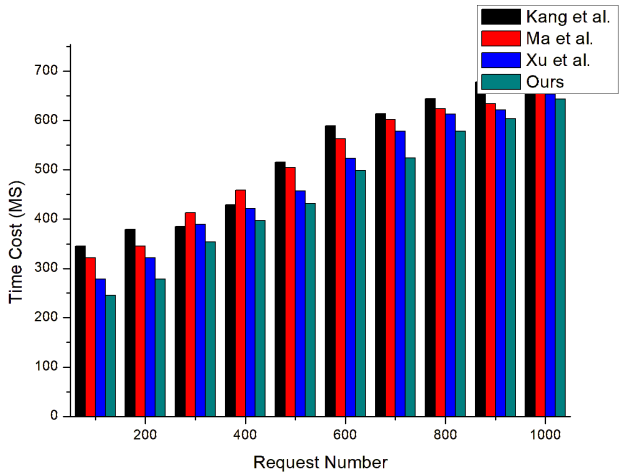


Figure 3.4: Comparison of computational cost for registration

Storage requests for 100KB-1000KB of unstructured data (JSON) were simulated using Apache JMeter 5.2 in the experiment to assess data storage performance. Notably, without considering storage costs in off-chain databases, the time consumption of storage operations primarily consists of data encryption operations and blockchain synchronization time. Specifically, data storage requests were simulated, and the average processing latency was determined using multi-threaded technology. The design outperforms other comparative schemes in terms of efficiency, as shown in Figure 3.5. The average time cost of a request is 453.2 ms for a data volume of 300 KB, and approximately 689.4 ms for a data volume of 800 KB. Meanwhile, the corresponding time cost of other comparisons is 784.5 ms (Ref [67]), 732.6 ms (Ref [60]), and 832.3 ms (Ref [61]). Since this scheme is based on the blockchain to provide a trustworthy storage environment, utilizes smart contracts and combines a small number of cryptographic operations to ensure the security of the stored data, while improving the overall efficiency of the system. Furthermore, our scheme has a relatively small fluctuation in time cost compared to other comparative schemes.

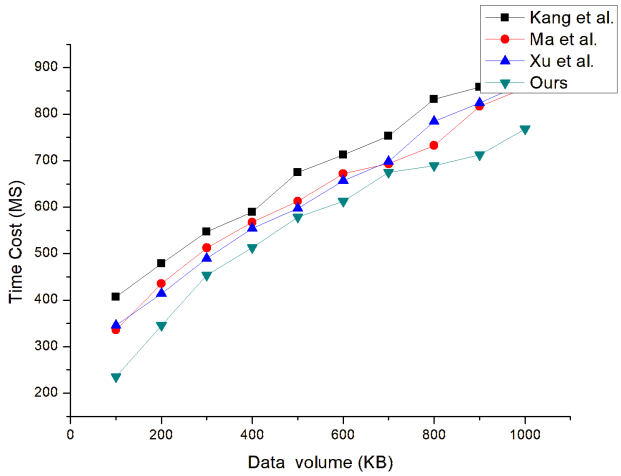


Figure 3.5: The performance comparison of this scheme with other state of art schemes

To estimate the communication overhead, the scheme primarily considers two phases: entity registration and IoT data storage. The communication overhead of these two phases is analyzed separately. Figure 3.6 shows the simplified flowchart of inter-entity communication. The communication overhead can be represented as the sum of various components: $\{|Info_i| + |\sigma_i| + |x_i| + |Pubk_i| + |ACK|\} = 10440$ (bits). Consequently, the aggregate communication cost for this entire process is quantifiable as $\{|CT_{(SK_{ij})}| + |CT_{(auMsg_i)}| + |CT_{(SK_{ji})}| + |CT_{(auMsg_j)}| + |SK| + |CT_{(r_{i,j})}| + |RC_{(r_{i,j})}| + |ACK|\} = (4096 * 5 + 256 + 256 + 160 + |CT_{(r_{i,j})}|) = 21152$ (bits). Therefore, the total communication overhead of this scheme can be computed as $TC_{total} = TC_{(registration)} + TC_{(storage)} = 31592(bits) \approx 3949$ (bytes).

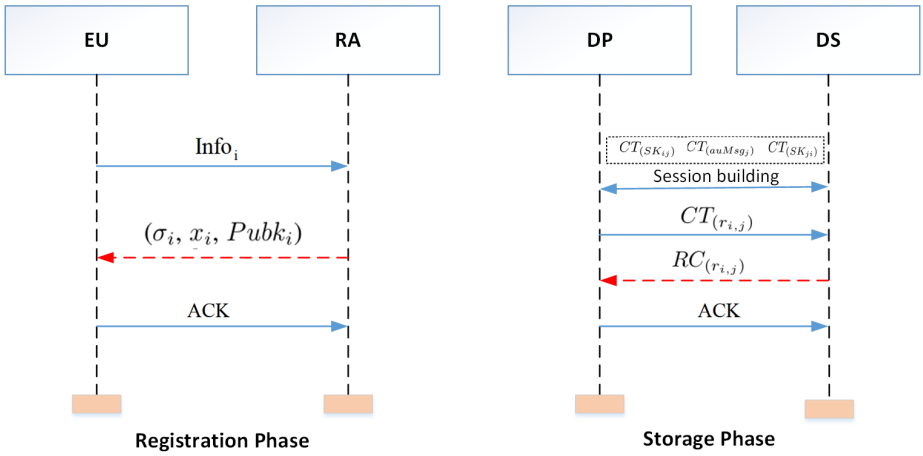


Figure 3.6: The simplified flowchart of inter-entity communication

Furthermore, a comparative analysis of the exchange overhead of this scheme and other state-of-the-art solutions was conducted, as presented in Table 3.6. It's worth noting that the calculation of communication overhead does not encompass the consumption of network bandwidth by various types of descriptive data during session construction. Nevertheless, the overall communication overhead remains relatively low and suitable for deployment in IIoT environments.

Table 3.6: Comparision of Communication Cost.

Scheme	Message Structure	Size (byte)
Xu et al. [67]	$\{M_i', ts, \sigma_i\}$	$184 + \|M_i'\|$
Ma et al. [60]	$\{HM_{v_i}, Sig_{k_{v_i}^l}(HM_{v_i}), K_{v_i}^l, Sig_{sk}\}$	$216 + \|msg\|$
Kang et al. [61]	$\{< ID_i, U_i, V_i, W_i >, \ldots, msg, m, t\}$	$200 + \|msg\|$
Our scheme	$\{Cert_{V_r}, msg, ts_g, \delta_{V_r}\}$	$168 + \|msg\|$

3.7 Conclusions

This chapter introduces a blockchain-based protocol for secure data storage in the IIoT, which combines robust entity authentication with bilinear mapping for rapid system initialization. By employing blockchain-ensured encryption, the proposed protocol achieves significant reductions in resource consumption while ensuring data security. The comprehensive evaluation of our solution reveals its effectiveness in reducing system overhead and enhancing storage reliability, particularly in identity registration and handling large-scale data.

Chapter 4

Trie-alternative transaction serial sorting

Contents

4.1 Introduction

Blockchain has evolved from the initial Bitcoin [70] to an extensive technical field, involving many fields including industry [71], finance [72, 73], supply chain management [74], education [46] and health care [75, 76]. In this context, the open-source blockchain system Hyperledger fabric has garnered significant recognition within practical application domains, thanks to its innovative plug-in architecture and emphasis on privacy and security [77–80]. Compared to classical distributed database systems, Hyperledger fabric provides membership services, divides the node scope according to the identity of the organization, and successfully realizes the non-trust cooperation of multiple organizations [81, 82]. The Fabric system ensures that sensitive information is shared among organizations whilst public data is passed across all organizations, maintaining data consistency across the blockchain system with a consensus algorithm.

Hyperledger fabric employs an optimistic concurrency control mechanism that allows multiple transactions to proceed simultaneously at various stages, enhancing overall system throughput [83]. Utilized by Hyperledger, the Execute-Order-Validate (EOV) model aims to create a high-performance blockchain system by enabling parallel transaction execution [84]. Concurrent transaction conflicts can significantly increase suspensions or invalidations, reducing the system's overall transactional throughput [85]. The processing of transactions consists of several stages, among which transaction categorization is a key step. To date, the transaction sorting algorithm adopted by Fabric is mainly based on timestamps and other parameters. However, several issues in this method include (i) the timestamp may be affected by the time between nodes not being synchronized, and (ii) when a large number of transactions are issued, relying only on the timestamp may cause a transaction processing bottleneck [86]. Additionally, traditional sorting and verification methods frequently fall short in addressing the issue of concurrent read-write conflicts inherent in large-scale transactions, necessitating more sophisticated and efficient approaches to enhance transaction processing [87].

To validate the presented issue, 10 data keys are initiated, each assigned a random value. Subsequently, a pair of keys is randomly selected, one key is used for a read operation while the other is used for a write operation. A series of transactions are then generated using Caliper [88] for the first benchmarking phase. Two organizations are established within this setup, each equipped with a Peer node. CouchDB [89] serves as the state database, with Solo Consensus [90] employed as the consensus mechanism. Based on this framework, two key performance metrics are defined: the effective rate, given by $EffectiveRate = \frac{\mu_{valid}}{\mu_{\max}} = \varphi_{valid}$, and the termination rate, articulated as $TerminationRate = \frac{\mu_{invalid}}{\mu_{\max}} = \varphi_{invalid}$, where μ means the number of variables. As Figure 4.1 shows, among the seven trials

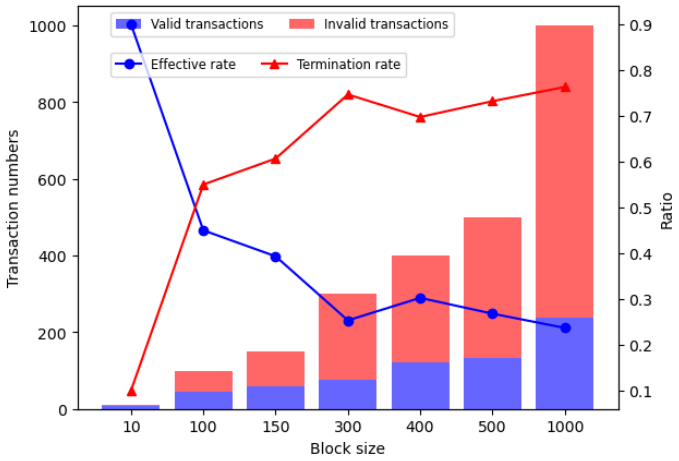


Figure 4.1: An illustration of large-scale terminated transactions in Hyperledger fabric.

conducted, six groups exhibited an effective transaction volume that did not surpass 50%. Concurrently, as $\mu_{\max}$ increases, both the suspension rate surges and system efficiency diminishes. This is predominantly attributed to the simultaneous execution of transactions.

Consider two transactions: T_1 and T_2. Transaction T_1 initially executes a write operation on data key K_1, subsequently progressing to the sorting phase. During this interval, T_2 conducts a read operation on K_1, accessing the value before any modifications. Consequently, upon submitting these two transactions, T_2 is rendered invalid owing to discrepancies in version numbering. An augmentation in the value of $\mu_{\max}$ invariably leads to an intensified prevalence of conflicts stemming from simultaneous transactions within a singular block. These conflicts present considerable obstacles, thereby constraining the transaction throughput of Hyperledger fabric [91].

Trie trees and directed acyclic graphs (DAGs) offer unique advantages for blockchain and transaction management. The Trie is a specialized data structure that efficiently organizes dynamic sets of strings, allowing for rapid retrieval of elements sharing common prefixes [92]. The hierarchical nature facilitates swift searches, insertions, and deletions, which is ideal for systems requiring auto-complete, IP routing, and textual analysis [93]. Conversely, DAGs are graphs characterized by directed edges without cycles, enabling them to represent dependency relationships and perform topological sorting, an arrangement of vertices that aligns with the directionality of the edges [94]. Tries can manage large datasets with shared prefixes, such as account addresses, enabling quick and effective access to transaction records. DAGs, meanwhile, address transaction dependencies by eschewing circular references, a vital feature for preserving transactional integrity and chronological order [95]. These capabilities of trie trees and DAGs strengthen the efficiency of transaction

processing and bolster the capacity of blockchains to expand while retaining credibility.

Based on the above-mentioned analysis, this chapter integrates Trie structures with DAGs, creating a synergy that significantly improves the resolution of transaction conflicts and management of concurrency within the Hyperledger fabric. The Trie's hierarchical nature facilitates efficient, orderly data retrieval, while the acyclic nature of DAGs aids in detecting transactional cycles, enabling a more streamlined process for transaction processing and validation. This Trie and DAG-based approach, conceptualized to substitute traditional sorting and validation methods, showcases the potential for advanced algorithms to resolve the intricate challenges of large-scale transaction management in blockchain systems.

4.2 Related Work

This section surveys related work for this study, with the improvements in consensus algorithms presented in section 4.2.1 and the optimization in transaction conflict relations described in section 4.2.2.

4.2.1 Consensus Algorithm Optimization

Consensus algorithms are critical for blockchain network reliability and efficiency. Practical Byzantine Fault Tolerance (PBFT) [96, 97] stands out for tolerating f faulty nodes within $3f + 1$ nodes and is a focus for enhancing Hyperledger fabric's consensus mechanism. Despite PBFT's robustness, its high communication costs and lengthy consensus process have prompted research into improvements [98]. As such, Chen *et al.* [99] have advanced PBFT by introducing Credit Reinforce Byzantine Fault Tolerance (CRBFT), where nodes are dynamically rated for reliability, significantly improving system performance and security. Building on similar aspirations to optimize communication, Li *et al.* [100] proposed a layered approach to PBFT, reducing node communication overhead by limiting interactions within node layers. Furthermore, Liu *et al.* [101] developed a pipelined byzantine fault tolerance algorithm, optimizing consensus through threshold signatures and streamlined processes. Wang *et al.* [102] introduced an enhanced PBFT consensus mechanism that operates on a bidirectional fading credit value system. This mechanism selects nodes to form a committee based on votes and credit scores. Within the committee, consensus is achieved using the PBFT algorithm, and the credit score of each node is adjusted based on their performance during the consensus phase. As the blockchain height increases, credit scores converge toward 50, thereby improving the quality of nodes participating in the consensus process.

Table 4.1: Symbols and meanings of this manuscript

Symbols	Implication
T_i^s	Transactions in sequence s with index subscript i
$<_s$	$T_i <_s T_j$ indicates that T_i precedes T_j in the transaction sequence s
$B_s^x(n)$	Block B, where the transaction sequence is s and the block number is x, $\mu_max = n$
$r(k_1)$	Read the data key k_1
$w(k_1) = v$	The data key k_1 is written to the value v
$T_{xs}(B)$	All transactions in block B
$\varpi(k_1)^i$	Transaction T_i reads the version number of the key k_1
$\varpi(k_1)^X$	Data key k_1 Version number in the status database
$RS(T_i)$	The read set data key for $T - i$
$WS(T_i)$	The write set data key for T_i

4.2.2 Transaction Conflict Relations Optimization

Hyperledger fabric is notably susceptible to concurrency dilemmas, particularly in environments requiring swift response times. Addressing this, Sharma *et al.* [83] utilized directed graphs to map the complex web of transaction conflicts within a block, subsequently proposing a transaction reordering algorithm designed to terminate invalid transactions proactively. Built on this foundation, Nasirifard *et al.* [103] have enhanced the core system by integrating Conflict-free Replicated Datatypes (CRDTs), transforming conflicting transactions into a format that inherently reduces rates, thereby streamlining transaction processing. The scope of research extends beyond conflict resolution, with efforts by Narayanam *et al.* [104] introducing a DAG structure, which enables Fabric nodes to validate multiple blocks concurrently, thereby enhancing network resource efficiency. To address node communication delays, Berendea *et al.* [105] improved upon Fabric protocol to expedite block distribution, while Pi *et al.* [106] proposed xFabLedger, offloading ledger storage to remote databases to ease node burden and improve scalability. Complementing these advancements, Li *et al.* [107] have proposed a hardware acceleration scheme, poised to enhance the Fabric system's execution and validation phases. This innovation is particularly noteworthy as it holds the potential to significantly expedite the transaction lifecycle, a critical factor in high-stakes transaction environments.

4.3 Problem Definition

In this section, we describe the definition and analysis of transaction conflict. Table 4.1 summarizes the symbols used in the rest of this article.

4.3.1 Multi-version Concurrency Control

Hyperledger fabric employs Multi-Version Concurrency Control (MVCC) to navigate read-write conflicts within transactions, but it is limited to a coarse-grained approach for halting conflicting transactions [108]. Consider the case when the transaction sequence s represented by Equation 4.1, when committed according to the order of s, T_1 updates key k_1, resulting in T_2 and T_3 reading an old version of k_1, and finally only transaction T_1 is valid.

$$s = T_1\{r(k_1), w(k_1)\} \rightarrow T_2\{r(k_1), w(k_2)\} \rightarrow T_3\{r(k_1), w(k_3)\} \tag{4.1}$$

However, if the transaction submission order is changed, a new transaction sequence $s^{'}$ is formed as shown in Equation 4.2.

$$s^{'} = T_3\{r(k_1), w(k_3)\} \rightarrow T_2\{r(k_1), w(k_3)\} \rightarrow T_1\{r(k_1), w(k_1)\} \tag{4.2}$$

When committed according to the order of s, T_2 and T_3 first read the latest version of k_1, and then T_1 updates k_1, making all three transactions valid. By reordering the transactions, it is possible to make certain invalid transactions valid, and there is no need to resubmit transactions for execution and processing. The practices of transaction batching and transaction reordering are widely implemented in traditional databases [109].

Due to this, we analyze in this section the conflicted relationship between Hyperledger fabric transactions and transaction serializability from the perspective of a database.

4.3.2 Fabric Transactions Conflict Relationship

In traditional concurrent databases, there are three types of data access conflicts: write-write conflicts, read-write conflicts, and write-read conflicts. This sub-section discusses the impact of these three types of conflicts on Hyperledger fabric transactions and identifies the causes of conflicts between transactions T_i^s and T_j^s. WS and RS are read set data keys and write set data keys, respectively.

4.3.2.1 $WS(T_i^s) \cap WS(T_j^s) \neq \emptyset$

The given transactions T_i^s and T_j^s, satisfy the condition as given in Equation 4.3.

$$\begin{aligned} T_i^s, T_j^s &\in B_s^x(n) \\ T_i^s <_s T_j^s, i, j &\in [0, n) \subset N_+, i < j \\ WS(T_i^s) \cap WS(T_j^s) &= \{k_1\} \end{aligned} \tag{4.3}$$

When the Peer node commits according to the sequence s, transactions T_i^s and T_j^s perform write operations on data key k_1. If T_i^s is committed first, then the version number of key k_1 is updated as shown in Equation 4.4.

$$\varpi(k_1)^X = \langle x, i \rangle \tag{4.4}$$

Then submit T_j^s and change the version number of key k_1 as shown in Equation 4.5.

$$\varpi(k_1)^X = \langle x, i \rangle \rightarrow \langle x, j \rangle \tag{4.5}$$

After the final block B_s^x is submitted, the latest version number of key k_1 is $\varpi(k_1)^X = \langle x, j \rangle$. Therefore, there are no write-write conflicts ($conflict - ww$) in the Hyperledger fabric system, which are denoted as $conflict - ww$ in Figure 4.2.

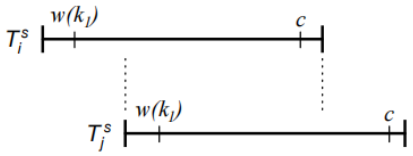


Figure 4.2: Conflict-ww

4.3.2.2 $RS(T_i^s) \cap WS(T_j^s) \neq \emptyset$

The given transactions T_i^s and T_j^s, satisfy the conditions as indicated in Equation 4.6.

$$\begin{aligned} T_i^s, T_j^s \in B_s^x(n) \\ T_i^s <_s T_j^s, i, j \in [0, n) \subset N_+, i < j \\ RS(T_i^s) \cap WS(T_j^s) = \{k_1\} \end{aligned} \tag{4.6}$$

As a Peer submits data key k_1 according to S, transaction T_i^s reads data key k_1, and transaction T_i^s writes data key k_1, Procedure First submit T_i^s, read the key k_1 version is the latest version of the state database at this time, that is, Equation 4.7.

$$\varpi(k_1)^i = \varpi(k_i)^X \tag{4.7}$$

The next step is to submit T_j^s and update the key k_1 version as shown in Equation 4.8.

$$\varpi(k_1)^X = \langle x, j \rangle \tag{4.8}$$

Once the block B_s^x is submitted, the system deems both T_i^s and T_j^s as valid. Consequently, this indicates the absence of read-write conflicts in Hyperledger fabric, which are represented as $conflict - rw$ in Figure 4.3.

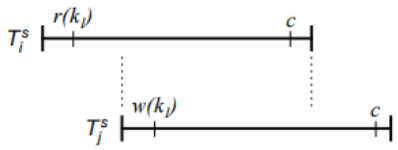


Figure 4.3: Conflict-rw

4.3.2.3 $WS(T_i^s) \cap RS(T_j^s) \neq \emptyset$

The given transactions T_i^s and T_j^s, satisfy the condition as shown in Equation 4.9.

$$\begin{gathered} T_i^s, T_j^s \in B_s^x(n) \\ T_i^s <_s T_j^s, i, j \in [0, n) \subset N_+, i < j \\ WS(T_i^s) \cap RS(T_j^s) = \{k_1\} \end{gathered} \tag{4.9}$$

The Peer node submits the data according to the s sequence. Transaction T_i^s writes data to data key k_1, and transaction T_j^s reads data to k_1. When T_i^s is submitted, the version number of key k_1 is updated as indicated in Equation 4.10.

$$\varpi(k_1)^X = \langle x, i \rangle \tag{4.10}$$

Next, the transaction T_j^s is subsequently submitted. However, at this point, the version number of the read key k_1 is outdated, as shown in Equation 4.11.

$$\varpi(k_1)^j = \varpi(k_1)_{old}^X \neq \langle x, i \rangle \tag{4.11}$$

Therefore, upon submission of block B_s^x, transaction T_j^s is flagged as invalid and requires re-submission by the client. In the Hyperledger fabric system, these disputes are characterized by write-read conflicts, represented as $conflict - wr$ in Figure 4.4.

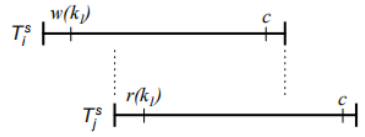


Figure 4.4: Conflict-wr

4.3.3 Exchangeability of Transactions

The exchangeability of transactions implies that the transition of $\Rightarrow s'$ is achievable merely by swapping the arrangement of transactions. Therefore, a $conflict - wr$ relationship can

be transformed into a $con-rw$ relationship through this exchange. T_i and T_j in s are $conflict-wr$ relations, and T_i and T_j in s' are $conflict-rw$ relations. Assuming that the conflicted relationship is not considered, the version of key k_1 will be updated to $\langle x, i\rangle$ after T_i submits, and the current version $\varpi(k_1)^X_{old}$ of k_1 will be read after T_j submits, which means that s and s' are equivalent. For two transaction sequences s and s' in block B^x can be shown as Equation 4.12.

$$\begin{aligned} T_i <_s T_j \\ T_j <_{s'} T_i \\ WS\left(T_i\right) \cap RS\left(T_j\right) = \{k_1\} \end{aligned} \tag{4.12}$$

4.3.4 Conflict Serializability

This sub-section describes the analysis of transaction conflicts within the Hyperledger structure, drawing on the principles of database concurrency control. Specifically, it examines transaction serialization, the construction of conflict graphs, and the detection of cycles. Based on the exchangeability of transactions, the definition of the serializability of conflicts in Hyperledger fabric transactions can be obtained.

Definition 1 Conflict Serializability. For all transactions in block B^x_s, there are at least two $conflict-wr$ between transactions. If there is a transaction sequence s' such that Hyperledger fabric submits blocks according to that sequence without terminating any transactions, s is conflict-serializable.

By identifying at least one sequence s' as outlined in Definition 1, the system can effectively reduce the proportion of invalid transactions. Although the relationship between two transactions is easy to determine, when multiple transactions are grouped, it is difficult to find the true conflict between each transaction. In the database domain, conflict graphs among transactions are usually established to analyze transaction conflict relationships. Based on this tendency, this sub-section defines the conflict graph of Hyperledger fabric.

Definition 2 Conflict graph. Let s represent a transaction sequence of block $B^X_S(n) = T_1, T_2, \ldots, T_n$, then the conflict graph$G_x = (V, E)$ of block B^x_s can be represented as Equation 4.13.

$$\begin{aligned} V &= B^x_s \\ (T_i, T_j) \in E &\Leftrightarrow RS(T_i) \cap WS(T_j) \neq \emptyset \\ T_i, T_j &\in B^x_s \\ i, j &\in [0, n) \subset N_+, i \neq j \end{aligned} \tag{4.13}$$

Definition 2 illustrates that the edges of the conflict graph G portray the read-write relationship between transactions, specifically $conflict-rw$ and $conflict-wr$. Importantly,

the structure of the conflict graph within the same block remains unchanged, regardless of alterations in transaction order. In Fabric, certain transactions need to be terminated, identified as cycles in the conflict graph G.

Definition 3 Cycle. If $T_i <^s T_{i+1} <^s \cdots <^s T_{i+k} <^s T_j$ is a sub-sequence in the transaction sequence s, and there is a $conflict - rw$ relationship between pairs, and a $conflict - wr$ relationship between T_i and T_j, then there is a cycle in s.

Based on Definition 3, cycles in Fabric cannot be eliminated merely by rearranging transactions; instead, terminating one of the transactions is necessary. This necessity arises because swapping T_i and T_j may convert a $conflict - wr$ into a $conflict - rw$, but it also introduces a new $conflict - wr$ between T_j and T_{i+k}. Essentially, any reshuffling of transactions within a cycle inevitably leads to a $conflict - wr$. Therefore, terminating a transaction becomes crucial for resolving the cycle structure.

Combining the above definitions, the following theorems are derived:

Theorem 1. Let s represent a sequence of all transactions in block B_s^x, then s is serializable if and only if G_x is acyclic.

Certificate of Necessity. Let s be conflict-serializable, then there is a serial sequence s', indicating that all $conflict - wr$ in s, such as $T_i <_s< T_j$, become $conflict - rw$, or $T_i <_{s'}< Tj$, in s'. It can be seen from the definition of the conflict graph that the conflict graph of block B_s^x does not change with the transaction sequence. Therefore, if G_x has a cycle, that is, there is a subsequence $T_i \to T_{i+1} \to \ldots T_{i+k} \to T_j \to T_i$ in G_x. According to the definition of the cycle, no matter how the order of this subsequence changes, there will inevitably be $conflict - wr$ conflict, which is in contradiction with the serialization of s'.

Certificate of Sufficiency. In contrast, if G_x is acyclic, it undergoes topological sorting. This means that nodes in each output round have only successor nodes and lack predecessor nodes. This implies that nodes from the previous output round have only conflict-rw relationships or no conflict at all with the current node. Consequently, the final output sequence s' is serial, confirming that s is serializable.

4.4 Algorithms Design

To construct a conflict graph for a block, according to the above description, it is necessary to establish the connection between data keys and transactions, achieving a one-to-one correspondence between keys and transactions. This section details the design of the Trie-Fabric. The data structure based on the Trie tree is introduced in section 4.4.1. Then section 4.4.2 describes the specific design of the proposed serial transaction classification algorithm, and finally, an example analysis is provided in section 4.4.3.

4.4.1 Trie Tree

Trie, commonly referred to as a prefix tree or dictionary tree, is an efficient data structure designed for rapid string retrieval. It is characterized by the following attributes:

1. The root node does not contain characters, except the root node, other nodes contain one character;
2. A path from the root to the leaf represents a string;
3. Subtrees of all nodes, except leaf nodes, represent strings with the same prefix.

The Trie tree offers highly efficient operations for string insertion, search, and deletion, typically boasting a time complexity of $O(m)$, where m denotes the string length in question. Therefore, this manuscript adopts the Trie tree structure to establish a link between keys and transactions. Using the data key as Trie's index, each leaf node contains two lists, cataloguing transactions that read or write to the key. As Figure 4.5 shows, the set $T1, T2, T3, T4$ forms a block, interacting with two data keys $a60$ and $be7$. This leads to the creation of two subtrees rooted in the Trie. The leaf node from the path $a \rightarrow 6 \rightarrow 0$ lists transactions related to key $a60$, with the read list including $T2, T3, T4$, and the write list $T2, T4$. Similarly, the path $b \rightarrow e \rightarrow 7$ follows this pattern. After the block is converted into a Trie tree, the conflict relationship between transactions can be clearly expressed, because in the same leaf node, the transaction in the read list must have $conflict - rw$ or $conflict - wr$ relationship with the transaction in the write list.

4.4.2 Transaction Serial Sorting Algorithm

The Trie-based transaction sequential sorting algorithm proposed in this chapter has three steps.

Step 1: Build the Trie tree. To begin, a Trie tree $\mathcal{T}$ is defined, functioning as a data structure to encapsulate the associations between data keys k and their corresponding transactions τ. The construction of $\mathcal{T}$ is governed by Algorithm 4, which begins with the iteration over the set of transactions $\mathbb{T}$ contained within a block. Each transaction $tx_i \in \mathbb{T}$ is processed in sequence, where the subscript i denotes the ordinal position of tx_i in $\mathbb{T}$, serving as an implicit timestamp. For every transaction tx_i, the algorithm traverses its read set $\mathbb{R}_{tx_i}$, where each element is a tuple (k, v), with k being the data key and v the value read. The transaction tx_i is then adjoined to the read list $\mathbb{L}_k^{read}$, which is situated at the leaf node of $\mathcal{T}$ corresponding to key k. A similar procedure is employed for the write set $\mathbb{W}_{tx_i}$ of tx_i. The algorithm iterates through each (k, v') pair, appending tx_i to the write list $\mathbb{L}_k^{write}$ at the leaf node of $\mathcal{T}$ associated with key k. This algorithmic approach

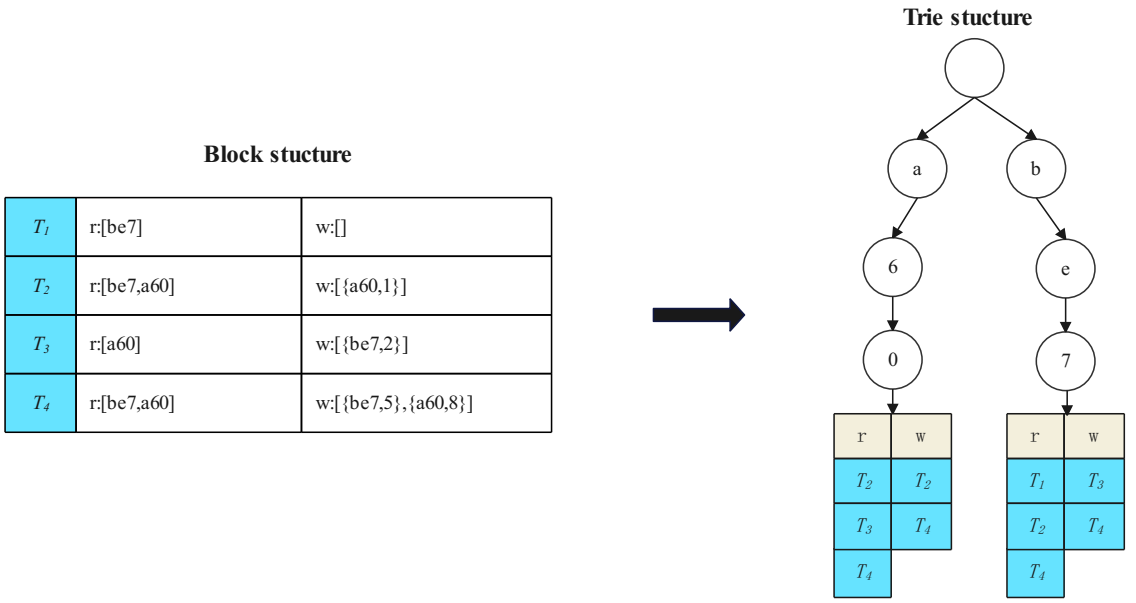


Figure 4.5: The process of turning the block to the trie tree structure

constructs a Trie tree $\mathcal{T}$ that simplifies the process of identifying transactions linked to specific operations, whether read or write, on a given key k. Consequently, $\mathcal{T}$ enables expedited retrieval of transactions τ relevant to any data key k within the block structure.

Step 2: Cycle detection. The protocol for cycle detection is detailed in Algorithm 5. The procedure initiates with an exhaustive traversal over each terminal node, designated as $item_i$, within the Trie framework. During this traversal, the read-list $RList_i$ and the write-list $WList_i$ pertinent to each $item_i$ are methodically analyzed to construct the conflict graph $G(V, E)$, by Definition 2. In this context, V and E represent the vertices (transactions) and edges (conflicts) of G, respectively. It is asserted that G is optimized, lacking any redundant edges or nodes and that no vertex in V serves as a progenitor or progeny of itself. For cycle detection, the algorithm systematically examines each transaction within the block, employing a depth-first search (DFS) denoted as $DFS(G, v)$, targeting the vertex v in G. This search is tailored to identify cyclical paths wherein the originating and terminating vertices are identical, constrained by the condition of equivalent timestamps, formalized by the equality $\tau(v) = \tau(v')$ for any vertex pair v and v'. If a cycle is discerned, the transaction at the genesis of the cycle is nullified. Subsequently, G is modified to eliminate the implicated vertex v, along with the associated edges $E(v)$, thereby transforming G into a DAG, effectively purging any cyclical transactions.

Step 3: Output serial sequence. For G, Algorithms 6 outputs a serial sequence s by topological sorting, and Hyperledger fabric submits transactions according to that

Algorithm 4 Build the Trie tree

Require: *Block* $B_s^x(n)$
Ensure: *Trie tree*
1: $trie = NewTrie()$;
2: **for do**$txIndex, tx$ in $B_x^s(n)$
3: //Specify a timestamp for tx txIndex;
4: **for** *key* in $tx.ReadSet$ **do**
5: //trie inserts tx into the read list of the key according to the key index;
6: **end for**
7: **for** *key* in $tx.WriteSet$ **do**
8: // trie inserts tx into the write list of the key according to the key index;
9: **end for**
10: **end for**
11: **return** *trie*;

Algorithm 5 Cycle detection

Require: $B_s^x(n)$, *trie*
Ensure: *DAG G, data key set Keys*
1: Create a conflict graph G;
2: **for** each leaf item in trie **do**
3: **for** tx_i in $item.RList$ **do**
4: **for** tx_j in $item.WList$ **do**
5: **if** $tx_i = tx_j$ **then**
6: continue;
7: **end if**
8: **if** $(tx_i, tx_j) \in G, E$ **then**
9: continue;
10: **end if**
11: $G, E = \{(tx_i, tx_j)\} \bigcup G, E$;
12: **end for**
13: **end for**
14: $Keys = \{item.Key\} \bigcup Keys$;
15: **end for**
16: **for** $txIndex, tx$ in $B_s^x(n)$ **do**
17: **if** G to exist in the tx cycle as the starting point and end point **then**
18: terminates tx;
19: Remove the nodes of tx and all its constituent edges from G;
20: **end if**
21: **end for**
22: **return** $G, Keys$;

Table 4.2: The read-write set of all transactions in a block

timestamp	Trading	Readset	Writeset
0	T_0	K_0,K_1	K_2
1	T_1	K_3,K_4,K_5	K_0
2	T_2	K_6,K_7	K_3,K_9
3	T_3	K_2,K_8	K_1,K_4
4	T_4	K_9	K_5,K_6,K_8
5	T_5	K_10	K_7
6	T_6	K_3	K_10

sequence without any conflicts.

Algorithm 6 Output serial sequence

Require: G
Ensure: *Serial sequence* s
1: **for** $G.V \neq \emptyset$ **do**
2: **for** $node \in G.V$ and $node.precursor = \emptyset$ **do**
3: $s = \{node\} \bigcup s$
4: Remove node from $G.V$;
5: Delete $(node, node.next)$ from G.E ;
6: **end for**
7: **end for**
8: **return** s;

4.4.3 Analysis

This section uses an example to demonstrate the correctness of the sequential transaction sorting algorithm proposed. As shown in Table 4.2, if a block contains seven transactions $T_1, T_2, T_3, T_4, T_5, T_6$ and reads and writes to 11 data keys $K_0 - K_10$, the transaction T_3, T_4, T_6 will be terminated if the original Hyperledger fabric processing logic is followed.

Step 1: Initialization. First, the Trie tree is created by Algorithm 4. Figure 4.6 shows the established Trie tree structure. Beginning at the root node, which holds the character 'K', all subsequent child nodes inherit the prefix 'K'. The terminal nodes are leaves, and traversing from the root to any leaf node allows for querying a key $K_i, i \in [1, 10] \neq N_+$, where $i \in [1, 10]$ and i is a positive integer, as well as performing read-write operations on that key.

Step 2: Cyclomatic. Figure 4.7 is the conflict graph G created according to Definition 2. Next, the cycle judgment operation is performed according to Algorithm 5. The first one is transaction T_0, and there is a cycle c_1 as shown in Equation 4.14.

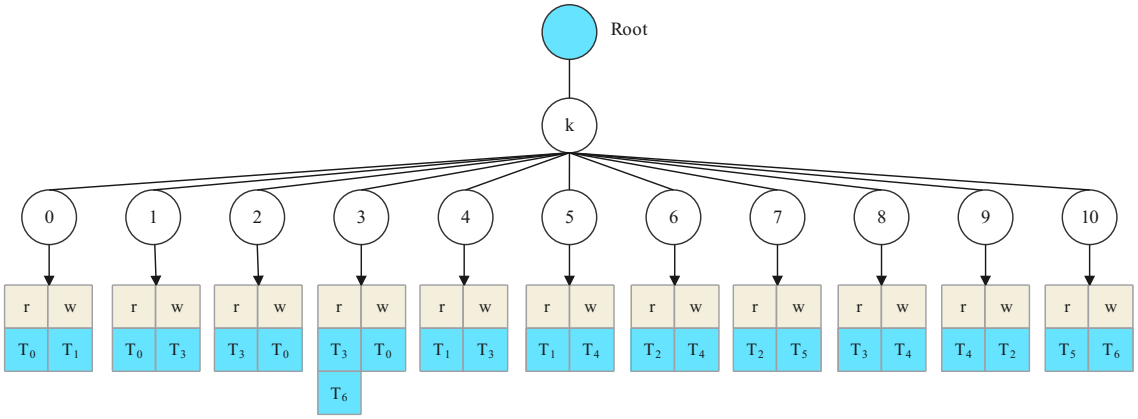


Figure 4.6: Trie tree structure

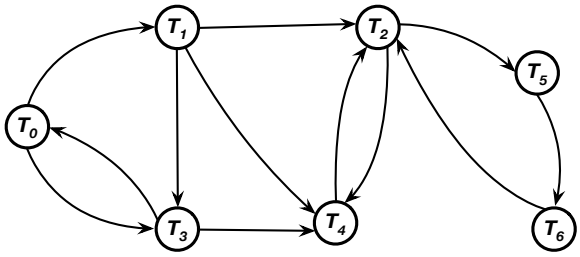


Figure 4.7: Conflict graph

$$c_1 = T_0 \rightarrow T_3 \rightarrow T_0 \tag{4.14}$$

Therefore, it is necessary to terminate the transaction T_0 and delete the relevant nodes and edges in G. For transaction T_1, there is no cycle. For transactionT_2, there are cycles c_2 as Equation 4.15, and c_3 as shown in Equation 4.16.

$$c_2 = T_2 \rightarrow T_4 \rightarrow T_2 \tag{4.15}$$

$$c_3 = T_2 \rightarrow T_5 \rightarrow T_6 \rightarrow T_2 \tag{4.16}$$

Thus, T_2 is terminated and removed from G. The residual transactions T_3, T_4, T_5, and T_6 no longer form loops in the conflict graph G, resulting in the DAG as shown in Figure 4.8.

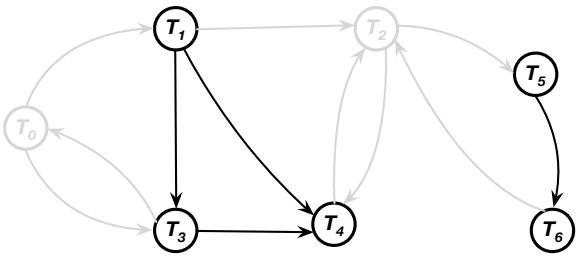
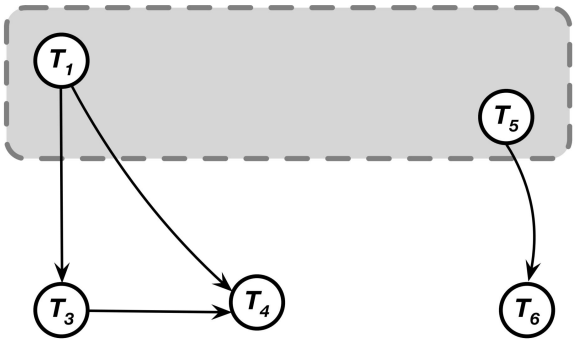


Figure 4.8: Directed acyclic graph

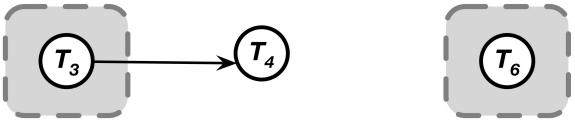
Step 3: Serialization. The DAG G is outputted through step 2, and the final serial transaction sequence s is determined following Algorithm 6. As illustrated in Figure 4.9, the topological sorting occurs in three rounds. The first round, shown in Figure 4.9(a), outputs transactions T_1 and T_5. The second round, depicted in Figure 4.9(b), outputs transactions T_3 and T_6. Finally, outputs transaction T_4 in the third round, as shown in Figure 4.9(c). Consequently, the serial sequence produced is $s = \{T_1, T_5, T_3, T_6, T_4\}$, allowing transactions to be submitted in this order without any conflicts.

4.5 Experimental Results and Analysis

This section presents the simulation tests conducted to evaluate the performance of the proposed transaction serialization sorting algorithm. Section 4.5.1 introduces the benchmark test set, while Section 4.5.2 details the experimental results and comparative analysis.



(a) First round



(b) Second round

(c) Third round

Figure 4.9: Topological sorting process

Table 4.3: Type of transaction in SmallBank

Number	Transaction type	Readset	Writeset
1	DepositChecking	1xR	1xW
2	WriteCheck	1xR	1xW
3	TransactSaving	1xR	1xW
4	SendPayment	2xR	2xW
5	Amalgamate	2xR	2xW
6	Query	1xR	

4.5.1 Benchmark Set

To make the experimental results more accurately reflect the real production environment, the Benchmark test set needs to contain controllable conflict relations, showing the rule that the optimization effect of the algorithm proposed in this manuscript on the original problem changes with the intensity of the conflict. This section introduces three independent variables during the generation of the benchmark test set: the maximum block size μ_{max}, the read-write transaction ratio p, and the Zipf's law frequency constant α. The maximum block size μ_{max} specifies the total number of concurrent transactions, the read-write transaction ratio p represents the proportion of transactions with read-write set structures within a block, and the Zipf's law frequency constant α indicates the frequency of transactions accessing the same data key within a block.

4.5.1.1 Workload

This work selects the SmallBank benchmark [110] to simulate a real-world financial industry scenario in terms of workload. Initially, 10,000 accounts are defined and random values are set for each account as the term balance and current balance in the account, and each account generates a data key through a unique data key index as shown in Equation 4.17.

$$key := SHA512(*smalbank*)[0:6] + SHA512(id)[0:64] \tag{4.17}$$

The data key is a 70-bit fixed-length string. As indicated in Table 4.3, the SmallBank benchmark specifies six transaction types. DepositChecking, WriteCheck, and TransactSaving feature read-write sets of size one. SendPayment and Amalgamate have read-write sets of size two, while the Query transaction comprises a single read set of size one. DepositChecking, WriteCheck, TransactSaving, SendPayment, and Amalgamate are categorized as read-write transaction types, while Query is categorized as a read-only transaction type. Since transactions possess atomicity similar to that of database transactions, Fabric processes data operations in batches within the read-write sets of transactions without the ability to partition execution. Therefore, the level of conflict varies when selecting different

Table 4.4: Probability distribution for each type of transaction

Transaction type	DepositChecking	WriteCheck	TransactSaving
Chance	$\frac{p}{5}$	$\frac{p}{5}$	$\frac{p}{5}$
Transaction type	SendPayment	Amalgamate	Query
Chance	$\frac{p}{5}$	$\frac{p}{5}$	$1-p$

transaction types for concurrent execution in the Hyperledger fabric system. For instance, a block with a higher proportion of read-only transactions is more likely to undergo successful submission, given the reduced likelihood of encountering a $conflict-wr$.

4.5.1.2 Ratio of Read-write Transactions

Assuming that the ratio of read-write transactions is p, the probabilities of the five types of read-writer transactions appearing in the block are $\frac{p}{5}$ respectively. As shown in Table 4.4, the proportion of read-only transaction types in the block is $1-p$. By adjusting the value of p, the proportion of $conflict-wr$ in the generated block files varies, and the impact of different read-write scenarios on the transaction terminate rate is detected in trie-Fabric.

4.5.1.3 Zipf's Law Access Frequency Constant

Simply altering the ratio of read-write transactions is not enough to manage the occurrence of conflicting transactions. It's also crucial to adjust the frequency of accessing accounts for read-write operations, which in turn controls the number of conflicts and cycles within a block. This manuscript operates under the hypothesis that access frequency inversely correlates with account ID size, where smaller account IDs experience higher access rates. To manage access frequency for each account adeptly, Zipf's law is utilized.

$$P(X=r)=\frac{C}{r^{\alpha}}, C=1.0 \tag{4.18}$$

The probability of accessing account r is denoted by $P(X=r)$ as indicated in Equation 4.18, and Equation 4.19 is obtained by taking the logarithm of both sides.

$$\begin{aligned} lgP(X=r) &= lg\frac{C}{r^{\alpha}} \\ &= lgC-\alpha lgr \\ &= -\alpha lgr \end{aligned} \tag{4.19}$$

It's clear that a higher value of the constant α correlates with an increased likelihood of accessing account r. If the block size $\mu_{\max}=1000$, the number of transactions accessing the first 100 accounts presents α different distributions with the value of α. When $\alpha=0$,

$P(X = R)$ represents the uniform distribution, and the probability of accessing 10,000 accounts equals. Therefore, Figure 4.10 shows that there is little difference in the number of accounts with ID 1-100 being read and written. When $\alpha = 1.0$, Figure 4.11 shows that the account with ID 1 is read and written twice as often as the account with ID 2, three times as often as the account with ID 3, etc. When transactions access the account with this frequency, there will be a large number of $conflict - wr$ in the block, but the possibility of a cycle is very small. When $\alpha = 2.0$, as shown in Figure 4.12, the number of transactions read and written to the account with ID 1 far exceeds that of other accounts. In this case, there must be a large number of cycle structures in the block, because there will be many transactions read and written to the account with ID=1 in the same block. $conflict - wr$ and $conflict - rw$ must be formed, and if the conflicting transaction also reads and writes other accounts, there will be a cycle structure. By specifying different α values when generating the benchmark set, the probability of different transactions reading and writing to the same account can be effectively adjusted.

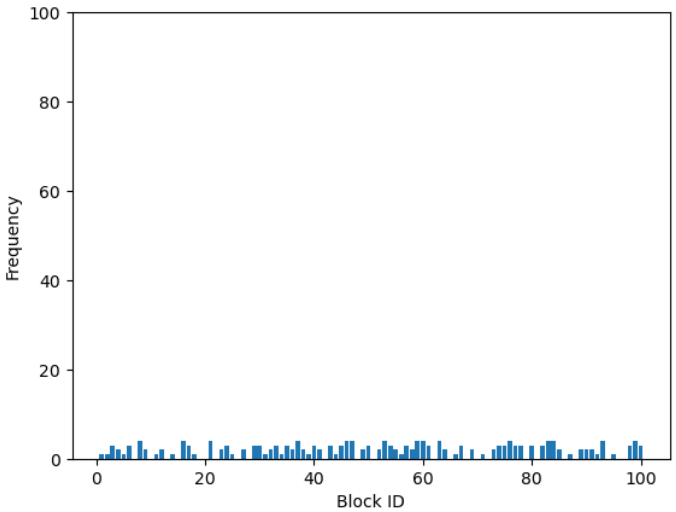


Figure 4.10: The frequency of transactions accessing when $\alpha = 0$

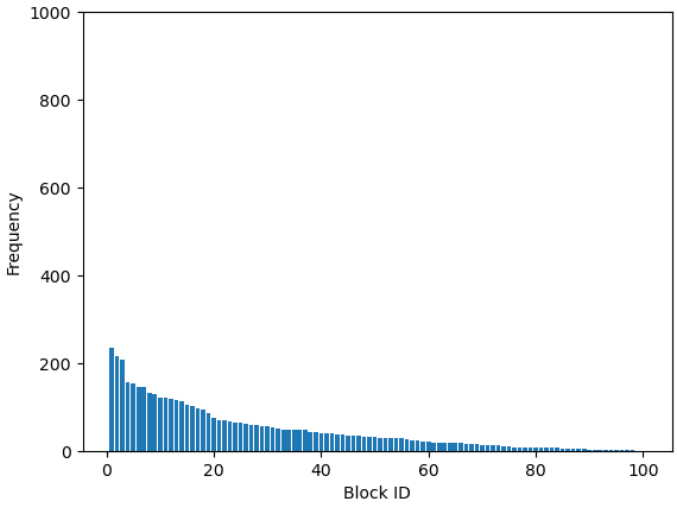


Figure 4.11: The frequency of transactions accessing when $\alpha = 1$

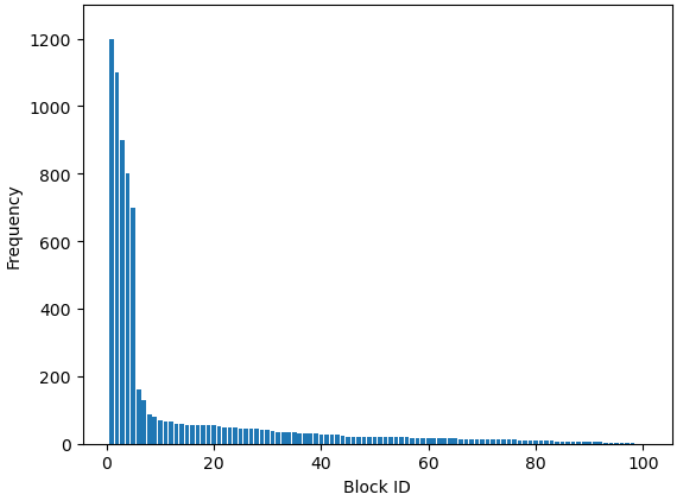


Figure 4.12: The frequency of transactions accessing when $\alpha = 2$

Finally, all experimental parameter values in this manuscript are listed. As detailed in Table 4.5, seven different values of $\mu_{\max}$ are set for the experiment. It's evident that a higher $\mu_{\max}$ value leads to a greater number of concurrent transactions, thereby increasing the possibility of conflicts. In this section, only three P-values are set to test the system conflict handling capability when the ratio of read-write transactions ranges from low to high. The Zipf's law access frequency constant α is incremented in steps of 0.2, ranging from 0 to 2.0. A benchmark set was created using these parameter combinations and tested on both Hyperledger fabric and Trie-Fabric, yielding several sets of experimental results.

Table 4.5: Probability distribution for each type of transaction

Experimental parameter	Parameter value
Block size μ_{max}	50, 100, 200, 300, 400, 500, 1000
Ratio of read-write transactions	0.05, 0.50, 0.95
Zipf's law access frequency constant	0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0

4.5.2 Experimental Results

This chapter validates the effectiveness of the proposed Trie-Fabric through multiple sets of experimental results. First, it compares the throughput of all transactions within a block before and after executing the transaction serialization sorting algorithm.

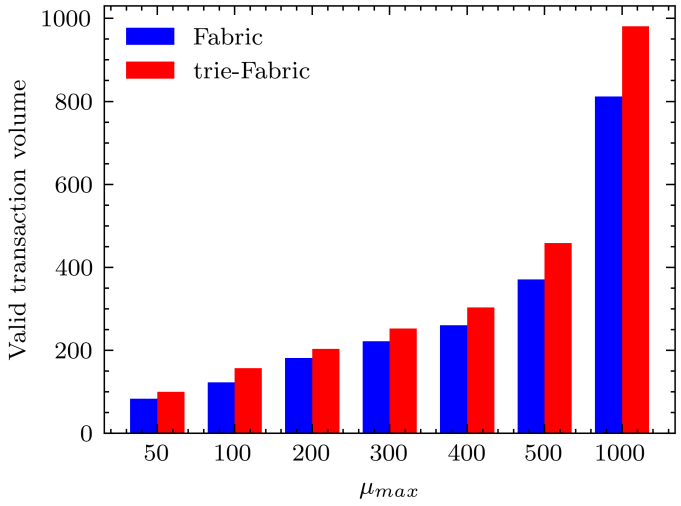


Figure 4.13: Valid transaction volume comparison when P = 0.05

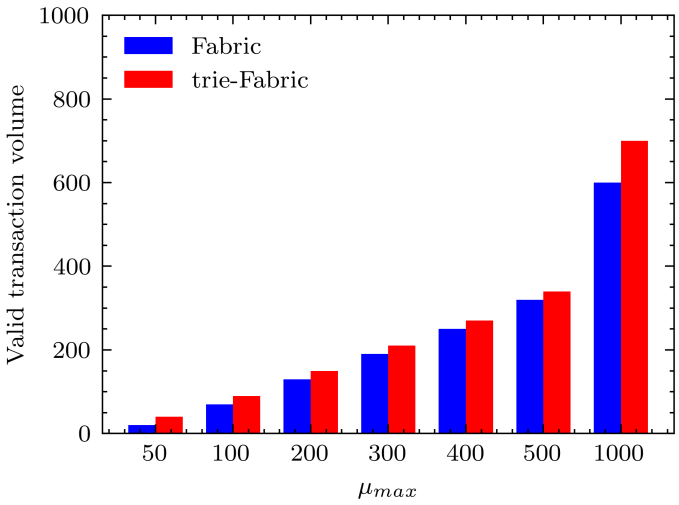


Figure 4.14: Valid transaction volume comparison when P = 0.50

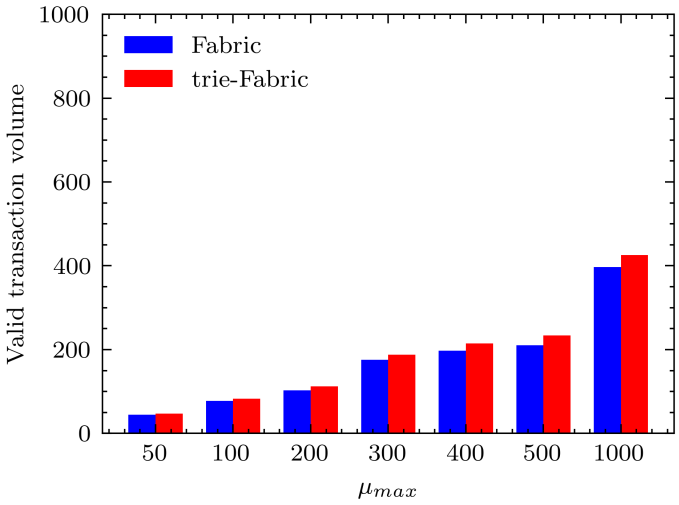


Figure 4.15: Valid transaction volume comparison when P = 0.95

The comparison of the number of successful transactions with varying maximum sizes at $p = 0.05$ is shown in Figure 4.13. It can be seen that when $\mu_{\max} = 50$, the count of effective transactions for both Fabric and trie-Fabric is nearly the same. This similarity arises because the smaller $\mu_{\max}$ value limits the number of concurrent transactions, allowing those with potential conflicts to be spread over different blocks, thus avoiding termination. However, processing smaller blocks in Hyperledger fabric is time-consuming and can negatively impact system performance. As shown in Figure 4.14, increasing $\mu_{\max}$ results in trie-Fabric recording more valid transactions than Fabric, while keeping the num-

ber of terminated transactions low. Particularly, at $\mu_{\max} = 1000$, trie-Fabric boosts the count of valid transactions by 18.8%. With $p = 0.05$, the block mainly comprises read-only transactions. The $conflict - wr$ conflicts between read-only and read-write transactions usually don't create cycles, leading to more effective transactions. In contrast, at $p = 0.95$, where read-write transactions make up a large portion, many cycles occur within the block, causing numerous transactions to be suspended. Figure 4.15 indicates that at $\mu_{\max} = 300$, trie-Fabric increases the maximum number of valid transactions by only 9.0%. Analyzing all three experimental datasets, the most significant improvement with trie-Fabric is observed at $p = 0.50$ and $\mu_{\max} = 1000$, where the optimization elevates the number of effective transactions by 44.5%.

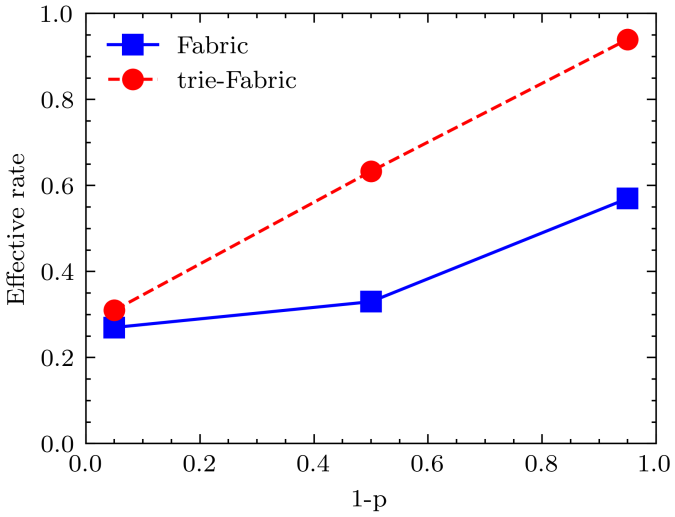


Figure 4.16: The trend of Effective rate with p

The above series of experiments show that the effective transaction in the conflict environment is related to the p value, so we compare and test the change rule of the efficient rate with the p. As shown in Figure 4.16, the horizontal axis represents the value of $1 - p$, which is the proportion of read-only transactions.The horizontal axis represents the effective rate, $\mu_{\max} = 1000$, $\alpha = 1.2$ for the experiment. For the same system, when the p value is smaller, the ratio of read-write transactions is smaller, and the efficiency is higher because there is no conflict between read-only transactions. When the experimental results of the two systems are compared, it is shown that the efficiency of trie-Fabric is always higher than that of the original Fabric through transaction serial sorting, and the maximum increase is 67.3% when $p = 0.05$.

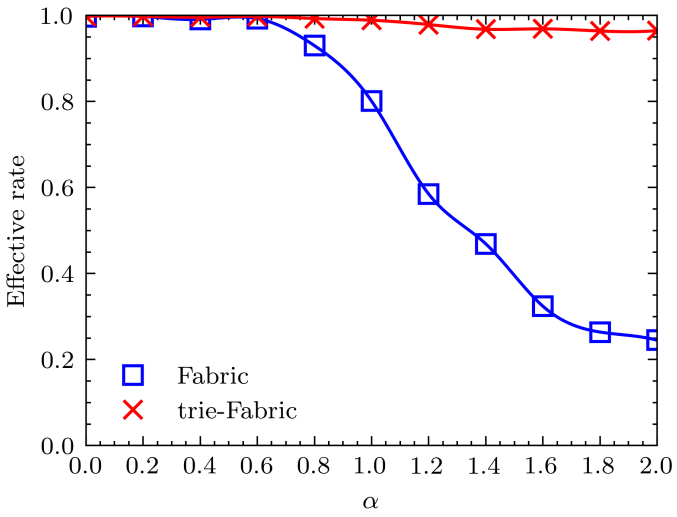


Figure 4.17: When $p = 0.05$, the trend of Effective rate with α

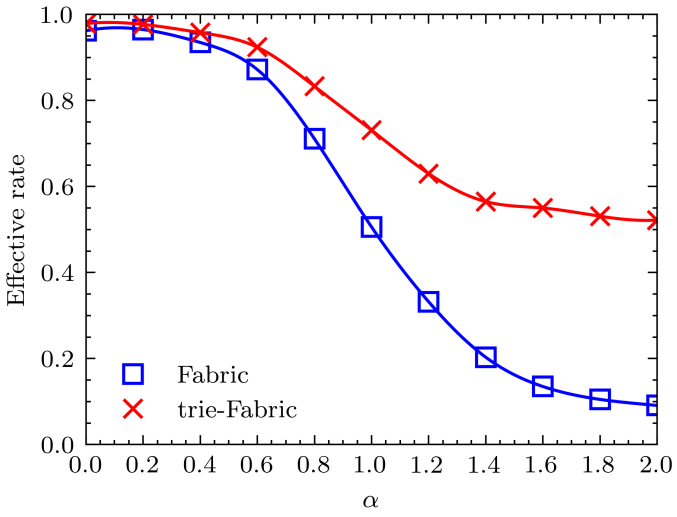


Figure 4.18: When $p = 0.50$, the trend of Effective rate with α

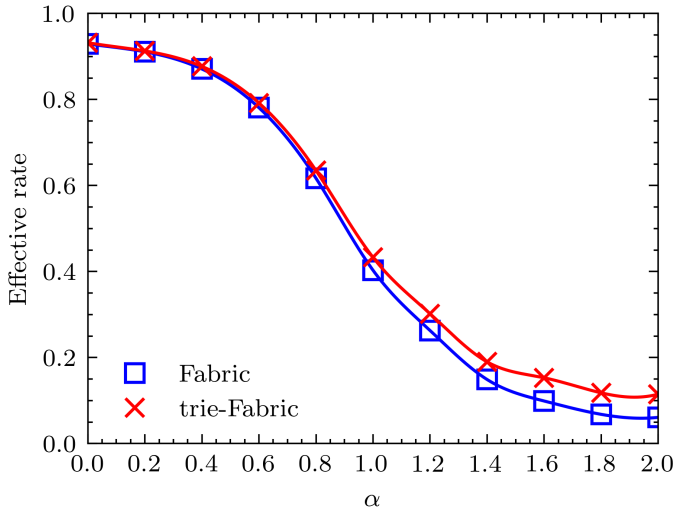


Figure 4.19: When $p = 0.95$, the trend of Effective rate with α

Finally, the effect of visit frequency α on efficiency was studied through multiple sets of experiments. To maximize the collision scope, a block size $\mu_{\max} = 1000$ was selected. As shown in Figure 4.17, although the frequency of transactions reading and writing to the same account increases with higher values of α, due to the small value of p, most of the blocks are $conflict - wr$ formed by read-only transactions and read-write transactions. Trie-Fabric converts most invalid transactions into valid ones by reordering them. Therefore, while the efficiency of Fabric decreases as α increases, Trie-Fabric's efficiency remains relatively stable.

Figure 4.18 demonstrates the efficiency trend when $p = 0.50$. Under this condition, the block contains an equal proportion of read-only and read-write transactions, which implies that both transaction types contribute to the formation of $conflict - wr$. As α increases, the efficiency of Trie-Fabric shows a decreasing trend. The number of transactions terminated by Trie-Fabric also rises, attributable to conflict cycles formed by read-write transactions. However, despite these challenges, trie-Fabric maintains an efficiency level exceeding 50%, demonstrating superior conflict handling capabilities compared to the original Fabric system.

The changing trend of efficiency when $p = 0.95$ is shown in Figure 4.19. In this scenario, since most blocks are read-write transactions, $conflict - wr$ conflicts mainly occur among these transactions. With the increase in α, the frequency of each transaction accessing a small number of accounts rises, significantly boosting the likelihood of conflicting transactions forming cycles. In such situations, trie-Fabric is compelled to terminate these transactions. Experimental results reveal that the efficiency trends of both Trie-Fabric

and the Fabric system follow a similar pattern, though the improvement in Trie-Fabric is not markedly pronounced. In high read-write concurrency scenarios like this, it is advisable to promptly terminate transactions and notify clients to resubmit, thereby avoiding unproductive transaction cycles.

4.6 Conclusion

This chapter introduces Trie-Fabric, an experimental blockchain system developed with a DAG-based Trie tree-structured serial transaction sorting algorithm to address the high incidence of suspended transactions in Hyperledger fabric. The chapter analyzes conflict issues arising from concurrent transactions within the standard Fabric system and proposes a reordering method using Trie-Fabric to resolve these issues. A benchmark set generated from SmallBank,Äôs workload is introduced, along with three variables that control the conflicting interactions within the test set. Comparative studies suggest that Trie-Fabric excels in managing conflicts, potentially improving efficiency by over 60% compared to the original Fabric system.

Chapter 5

Smart Contract-inspired Access Control

Contents

5.1 Introduction

IIoT, as a typical Cyber-Physical System (CPS), intricately combines wireless sensor networks, communication protocols, and internet infrastructure to oversee and optimize intelligent industrial operations [111]. These IIoT devices, connected through both wired and wireless channels, continuously interact with their environment, generating a wide variety of data types [112, 113]. This distributed network not only enhances system scalability and personalization but also facilitates effective monitoring and improved product quality, marking a shift from traditional centralized manufacturing systems to more decentralized and dynamic approaches [114]. Consequently, in this distributed IIoT ecosystem, a complex network of collaborative requirements arises among various devices and management entities, underscoring the imperative need for efficient coordination and communication [115].

However, with the expansion of industrial operations and the increasing interconnectivity of devices, traditional IIoT architectures are under strain to support such extensive networks [116]. This situation is exacerbated by the growing complexity of data types and communication protocols within these vast networks. Additionally, the incorporation of cutting-edge technologies such as AI and machine learning in IIoT systems further intensifies the demand for more robust architectures. Interactions over the Internet within the IIoT system involve data transfer to third-party servers, raising concerns about data security, privacy, and control [117]. Most current research on trust mechanisms in IIoT tends to neglect the environment's unique demands, particularly the limited and diverse computational and storage capabilities of IIoT devices. This oversight leads to practical challenges in applying these mechanisms, necessitating more customized and practical solutions [118, 119].

Furthermore, existing IIoT trust research often relies on trusted third parties or inter-domain trust assumptions, which are challenging to fully realize in practical industrial settings [120]. As the number of industrial devices continues to grow, centralized IIoT architectures encounter numerous hurdles. Maintaining data confidentiality is especially critical within the vast systems of industrial manufacturing, where any breach could have severe repercussions on system security [121–123]. Furthermore, with the diverse array of device types present in contemporary IIoT systems, the efficient utilization of generated data becomes increasingly intricate [124]. Consequently, centralized storage systems struggle to effectively manage data, leading to concerns regarding data authenticity, integrity, and validity [125].

In addressing these challenges, smart contracts facilitated by blockchain technology have emerged as a powerful solution, significantly enhancing security, trust, and data integrity in the IIoT. By embedding operation rules into the blockchain, smart contracts

make IIoT network interactions transparent, autonomous, and immutable [126, 127]. This transparency ensures that all parties involved in IIoT transactions can verify the integrity and authenticity of the data, fostering trust among stakeholders. Moreover, smart contracts align with the distributed architecture of IIoT, managing data transactions based on established rules and delivering immediate, automated responses to security threats [128]. This proactive approach to security enhances the resilience of IIoT systems against various cyber threats. Additionally, smart contracts adapt to the varied technical needs of IIoT devices, allowing for precise access control and secure data sharing [129, 130]. This adaptability is especially crucial in complex IIoT environments with unsynchronized device operations, reducing the dependence on central systems and minimizing data breach risks [131].

Overall, smart contracts play a crucial role in enhancing the security, efficiency, and reliability of IIoT systems, positioning them as a foundational technology in the future industrial landscape. Based on the above-mentioned analysis, this chapter introduces a data access control framework for IIoT utilizing the Hyperledger fabric platform, denoted as Hyper-IIoT. This framework attains secure storage and confidential data sharing within IIoT by harnessing the capabilities of distributed networks and consensus-based authentication. In terms of functionality, Hyper-IIoT adopts an Attribute-Based Access Control (ABAC) model to provide a flexible, dynamic, and fine-grained approach to data access control. Access control policies are formulated through smart contracts in conjunction with bloom filters, thus mitigating the computational overhead of the system by incorporating the bloom filter model.

5.2 Related Work

Smart contracts offer a crucial security mechanism and a reliable operating environment for managing access to IIoT data [132,133]. To ensure precise access control for asynchronously operating devices by different stakeholders, attribute-based access control models, often integrated with smart contracts, are employed to safeguard resources and information on IIoT devices [134]. For instance, Yang *et al.* [135] developed a secure data-sharing protocol using blockchain technology, enhancing single key exposure protection, enabling efficient keyword searches, and supporting dynamic user and key management. Conversely, Liu *et al.* [136] proposed an IoT access control system using Hyperledger fabric, yet it falls short in providing thorough auditing for access records.

To address the limitations of existing models, especially the latency issues in smart contract operations, researchers are exploring new additions to blockchain-based IIoT system access control modules. Wu *et al.* [62] presented MapChain-D, a framework that integrates distributed hash tables with Ethereum storage, specifically catering to environments with

significant latency and bandwidth constraints and thus serving resource-limited IIoT devices effectively. Similarly, the combination of multiple blockchain distributed keys and fog computing, as proposed in [137], shows potential in reducing operation times across various links, thereby boosting user privacy and security. Despite these advancements, a major hurdle remains in the form of high computational complexity resulting from direct interactions between IIoT devices through smart contracts, leading to unresolved challenges in managing the resultant high computational complexity.

5.3 Problem Definition

This section introduces the problem definition, including the interplay among models outlined in Subsection 5.3.1, the data structure described in Subsection 5.3.2, and interconnectivity in Subsection 5.3.3.

5.3.1 Definition of the Structural Relationship

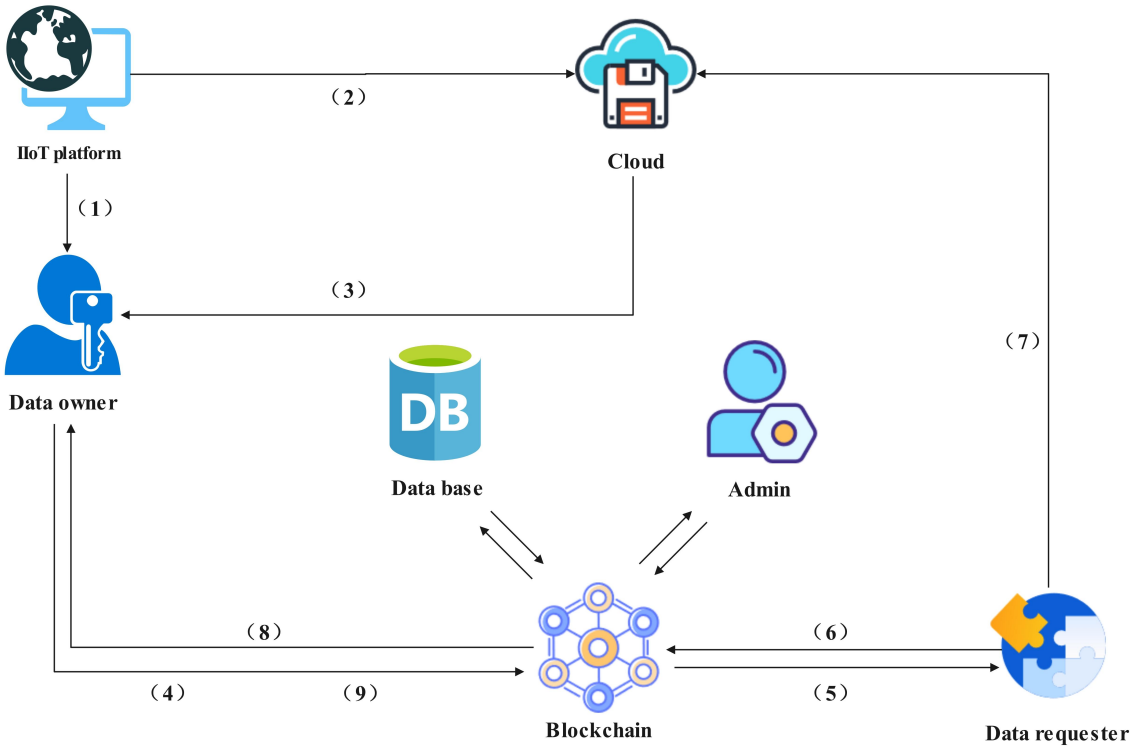


Figure 5.1: Structural relationship of Hyper-IIoT

The storage protocol introduced in this work mitigates the blockchain network's load by designating inconsequential template data as public. This data is hosted on a cloud server, enabling users to retrieve it via the server's address independently. In contrast, private data contains sensitive information that needs to be kept safe in a blockchain network. Data owners should implement access control policies for private data and manage private data by auditing access records. Figure 5.1 illustrates how users interact with private data.

5.3.2 Model Data Structure Definition

In this subsection, we present a dependable mechanism for safeguarding private data, that primarily comprises the enduring format of private data, access control policies, and records.

5.3.2.1 PersistentData

This model defines private data in a uniform format, as shown in Equation 5.1:

$$PD = \{persistentId, persistentData, persistentUrl\} \tag{5.1}$$

where PD represents the persistent data set of private data in the IIoT application scenario, $persistentId$ represents the unique primary key identification of persistent data resources, and $persistentData$ represents the persistent format of private data. $persistentURL$ Indicates the URL where public data can be accessed on the cloud server.

5.3.2.2 Access Policy

The access control strategy of the Hyper-IIoT model is the core to achieving fine-grained access control. The data structure is mainly based on the ABAC model, and its structure is shown in Equation 5.2 - 5.5:

$$PL = \{AUser, APD, AAuth, AEnu\} \tag{5.2}$$

$$AUser = \{userId, role, PKUser\} \tag{5.3}$$

$$APD = \{dataId, signer, signData, dataKey, dataUrl\} \tag{5.4}$$

$$AEnu = \{createTime, endTime, IP, signPKuser\} \tag{5.5}$$

where PL signifies the access policy related to the attributes such as user, data, environment, and permission. $AAuth$ represents the permission attributes, tasked with delineating

access privileges for private and public data endpoints within the IIoT domain. It comprises attributes such as $authUrl$ and $authData$, which correspond to the access control attributes for public data URLs and private data, respectively, with a general focus on validating the access permissions for private data. On the other hand, $AEnu$ signifies the set of environmental properties, delineating the contexts under which a data requester may access private environments. Environment properties encompass attributes like $createTime$ (policy creation time), $endTime$ (policy expiration time), ip (requesting node's IP address), and $signPKuser$ (data owner's signature on the policy to prevent tampering or forgery).

5.3.2.3 Record

To ensure the traceability of the Hyper-IIoT model, the system defines the access record as four data structures: $RECORD$, $REQUEST$, $RESPONSE$, and HIS, as shown in Equation 5.6 - 5.9.

$$RECORD = \{REQUEST, RESPONSE, HISTORY\} \tag{5.6}$$

$$REQUEST = \{AUser, APD\} \tag{5.7}$$

$$\begin{aligned} RESPONSE = \{&policyID, owner, requestID, \\ &status, endTime, timestamp\} \end{aligned} \tag{5.8}$$

$$HIS = \{resourceId, requestor, version\} \tag{5.9}$$

$RECORD$ encompasses privacy data access records for auditing, comprising $REQUEST$ (data request record), $RESPONSE$ (data owner's response record), and $HISTORY$ (data set access history). $REQUEST$ contains attributes about the requester and data set, while $RESPONSE$ includes details like $policyId$ (index for accessing blockchain's control policy), $owner$ (data owner), $requestID$ (index for request record), and $timestamp$ (data owner's response time). $HISTORY$ serves as a ledger for data set interactions, autonomously documenting pertinent details of requesters within the blockchain network. This registration occurs in conjunction with the access instance, precisely when a data collection, identified by $resourceId$, is successfully acquired by a requester.

5.3.3 Model Interrelation

The Hyper-IIoT model encapsulates a triad of interactive processes: Policy-to-Data, Data-to-Log, and Log-to-Policy. In this stage, the access control model ensures auditability

by recording data requester IDs, creating request and response records, and validating legitimacy based on access, request, and response records stored in the blockchain network. A direct correspondence is established between access logs and confidential data, simplifying the auditing process.

5.4 System Model and Design

The access control system based on smart contracts and bloom filter proposed in this paper is a reliable private data protection scheme. This section shows the architectural design and corresponding details of Hyper-IIoT.

5.4.1 System Architecture

Hyper-IIoT is deployed in the consortium chain, and the main design components include Users, Certification Authority (CA), Administrator (Admin), and Blockchain Network (BN), which provides user access modules that form the underlying structure. The system, integrating a smart contract-based access control model enhanced with a Bloom filter, is illustrated in Figure 5.2.

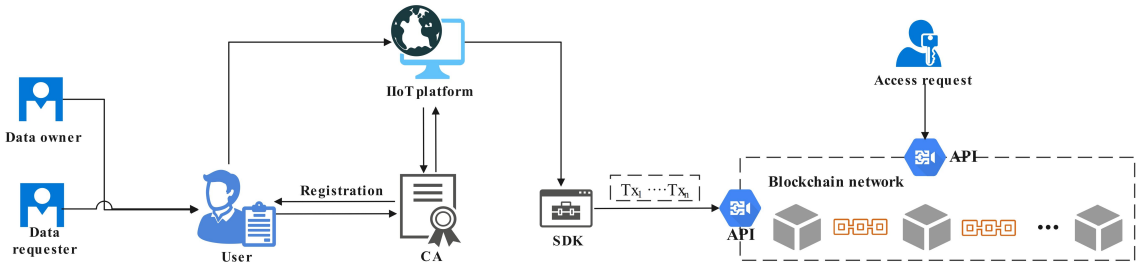


Figure 5.2: Structural Design of Hyper-IIoT

5.4.1.1 IIoT

The IIoT device management platform is tasked with handling diverse datasets, including video, audio, and textual content, produced by IoT devices. During this operation, data is classified into private and public categories based on the system's predefined segregation criteria, with the latter being transmitted to the cloud server for preservation.

5.4.1.2 User

The system encompasses data owners who upload, manage access, and respond to data requests and data requesters requiring permission for access. Data owners can audit their data resources via access control policies, enhancing data security and traceability.

5.4.1.3 CA

This component issues certificates to uniquely identify new nodes, requiring each user to possess a CA certificate for various procedures like connecting to and authenticating with the blockchain network.

5.4.1.4 Admin

Responsible for managing the basic capabilities, including network startup and smart contract installation rights, while overseeing access control through CA authentication, access policy formulation, user data access rights assignment, and maintaining policies, enabling Hyper-IIoT to authorize users for data-related activities.

5.4.1.5 BN

The key platform of the system, which consists of data storage and identity permission authentication, manages the storage of private data, access control, user authentication, and auditing, requiring authorization from the CA for users and Admin for data access, whereas data circulation and retention on the blockchain ledger are governed by smart contracts, culminating in a comprehensive data exchange ecosystem.

5.4.2 Smart contract design

This paper introduces four types of smart contracts for user management, data management, and access control, which are crucial for enabling Hyper-IIoT business logic and ensuring system efficiency, data privacy, and integrity.

5.4.2.1 User Management Contracts (UMC)

This contract is responsible for the management of user names and serial numbers, including the $CreateUser()$, $QueryUser()$, and $CheckUser()$.

$CreateUser()$ method: The initial step for user participation involves user registration, where $CreateUser()$ generates a unique user identification, $UserNumber$, based on the $UserName$ field.

$QueryUser()$ method: Facilitates user information retrieval using $UserName$ and $UserNumber$, including a check for existing users during registration.

$CheckUser()$ method: Before seeking authorization via the access policy, users must undergo authentication, which verifies the user's signature $SignID$, public key Pub_k, and private key Pri_k.

5.4.2.2 Access Control Contracts (ACC)

This function checks if the request to access the data is aligned with the preset access control policy. It includes key functions $Auth()$, $GetAttrs()$, $CheckPolicy()$, and $CheckAccess()$.

$Auth()$ method: Uses Admin-distributed public keys for request encryption, facilitating user request authentication and identity verification.

$GetAttrs()$ method: This module processes the attributes in received requests, focusing on subject and object characteristics, denoted as $\{S, O\}$. Additionally, it configures environmental attributes E, forming an integrated set of attributes $\{S.O.E\}$.

$CheckAccess()$ method: Algorithm 7 shows that this method validates access requests by checking if they align with established access policies, involving retrieval of attributes using $GetAttrs()$, querying the corresponding policy with $QueryPolicy()$, and verifying attribute satisfaction, with an invalid request result if no supporting policy is found or if attributes E and A fail to satisfy the policy. The deployment interface of ACC is shown in Figure 5.3.

```
root@jtli-ubuntu:/home/gopath/src/github.com/Hyper-IIoT/client/nodejs# node ./invoke.js acc CheckAccess 'ABACR{Su, Ou, Eu}'
Wallet path: /home/gopath/src/github.com/Hyper-IIoT/client/nodejs/wallet
acc CheckAccess ABACR{Su, Ou, Eu}
Transaction has been submit, result is: OK
```

Figure 5.3: The interface of $CheckAccess()$ method installation

Algorithm 7 The $CheckAccess()$ method checks the access request

Require: $ABAC_{Request}$
Ensure: $TrueorError$
1: $\{Su, Ou, Eu\} \leftarrow GetAtrrs(ABAC_{Request})$;
2: //Retrieve attribute info from the access control request based on attributes
3: $PolicySet = \{Policy_1, \ldots, Policy_n\} \leftarrow QueryPolicy(Su, Ou)$;
4: //Get attribute information based on attribute-based access control request
5: **if** PolicySet is Empty **then**
6: **return** $Denial()$
7: **end if**
8: **for** each $Policy$ in $PolicySet$ **do**
9: //Check each policy for compliance with visitor criteria
10: $\{Ap, \ldots, Pp, Ep\} \leftarrow Policy$
11: **if** $Value(Pp) == 1$ and $Eu \cap Ep$ is Not Empty **then**
12: **return** $Approval$
13: **end if**
14: **end for**
15: **return** $Denial()$

5.4.2.3 Private Data Control Contract (PCC)

This contract is used to manage both private and public data. It enables the data proprietor to amalgamate private and public data into singular resources, facilitating uploads and downloads. The contract encompasses methods as follows:

AddData() method: Enables the data owner to combine private and public data into a single data resource. Once combined, the consolidated data can subsequently be transmitted to the permissioned blockchain network.

GetData() method: This function permits data owners to access the data resources via a controlled policy mechanism, which coordinates with the blockchain's indexing system for retrieval.

5.4.2.4 Access Policy Management Contract (PMC)

This work defines the following four ways to operate $ABACPolicy$.

AddPolicy() method: It involves the CA executing the $CheckPolicy()$ module to ensure policy validity before addition. Only policies that pass this check are recorded in the SDB and blockchain, as detailed in Algorithm 8.

DeletePolicy() method: To remove expired policies, the administrator can explicitly invoke this function, or it can be executed automatically when the $CheckAccess()$ module runs.

UpdatePolicy() method: Invoked by the administrator to modify $ABACPolicy$, and it can be done by either deleting the policy and writing the modified record to SDB and the blockchain or by adding the modified policy after updating.

QueryPolicy() method: Allows administrators to retrieve $ABACPolicy$ details by attributes $ABACPolicy\_S$ or $ABACPolicy\_O$, as all policies are stored in CouchDB.

5.4.3 Policy Query Pptimization based on Bloom Filter

The bloom filter, denoted as BF, is a spatially efficient, probabilistic data structure tailored for approximate set membership verification, as explored by Waikhom *et al.* [138]. Comprised of an array of t bits, $BF = (b_1, \ldots, b_t)$, all initialized to zero. For a set of x elements $A = \{\alpha_1, \alpha_2, \ldots, \alpha_x\}$ and p unique hash functions $\{hash_1, hash_2, \ldots, hash_p\}$ that map uniformly to the range $[1, n]$, the insertion process of an element $\alpha_i \in A$ involves calculating p hash values $\{hash_1(\alpha_i), \ldots, hash_p(\alpha_i)\}$ and setting the corresponding indices in BF to one ($BF[hash_j(\alpha_i)] = 1$ for $1 \leq j \leq p$, $1 \leq i \leq x$). The BF can produce two types of indications: certainty of the data's presence or a probability of its presence, with the latter known as a false positive, which erroneously indicates the presence of data not

Algorithm 8 The AddPolicy() method customizes access policies

Require: $ABACP$
Ensure: $TrueorError$
1: @implement SmartContract Interface;
2: //Implementation of smart contract interface
3: $APIstubChaincodeStub \leftarrow Invoke()$;
4: **if** $err! = Null$ **then**
5: **return** $Error(BadPolicy)$;
6: **end if**
7: //Generate a unique index of the access policy based on the host and guest attributes
8: //Store the strategy in the blockchain
9: **if** $err! = Null$ **then**
10: **return** OK
11: **end if**
12: **return** OK;

inserted. A response of $BF[hash_j(\alpha_i)] = 0$ conclusively signifies that the element α_i does not exist in BF.

BF is utilized to enhance query efficiency within smart contracts, specifically to test the membership of variables within queries initiated by an Admin, thereby reducing the computational demands and the consequent system response delays. Assuming that the attribute sets of $ABACPolicy\_S$ and $ABACPolicy\_O$ with x elements are $\{k_0^s : value_0^s, \ldots, k_i^s : value_i^s, \ldots, k_{x-1}^s : value_{x-1}^s\} \in ABACPolicy\_S$ and $\{k_0^o : value_0^o, \ldots, k_i^O : value_i^O, \ldots, k_{x-1}^O : value_{x-1}^O\} \in ABACPolicy\_O$ respectively, the query procedure can be defined as follows:

Step 1: Build a bloom filter structure BF_A consisting of a t-bit array (all default values are 0) and p independent hash functions.

Step 2: Enter the defined set of smart contract attributes $\{ABACPolicy\_S, ABACPolicy\_O\}$, update the array $BF_A = (b_1, \ldots, b_n)$ with a value of 1, and its corresponding mapping process is $BF[hash(ABACPolicy\_S, ABACPolicy\_O)] = Index^{\{ABACPolicy\_S, ABACPolicy\_O\}}$.

Step 3: Input the attribute set $\{ABACPolicy\_S^{'}, ABACPolicy\_O^{'}\}$ of the query contract, $Hash_A$ it, and output the result set $Index^{S^{'}, O^{'}}$.

Step 4: The result set $Index^{\{ABACPolicy\_S^{'}, ABACPolicy\_O^{'}\}} = \{I_0^{'}, I_1^{'}, \ldots, I_p^{'}\}$ is iterated successively to determine the $b_j = BF(I_j)$ of the element $I_j^{'} \in Index^{\{ABACPolicy\_S^{'}, ABACPolicy\_O^{'}\}}$ in the BF_A index position obtained previously. If $b_j = 0$, it is determined that the attribute set $\{ABACPolicy\_S^{'}, ABACPolicy\_O^{'}\}$ of the contract in this query does not exist in the contract attribute set $\{ABACPolicy\_S, ABACPolicy\_O\}$ defined by the system, and the query process ends. If the value of $Index^{\{ABACPolicy\_S^{'}, ABACPolicy\_O^{'}}$ in the corresponding index position of BF_A is 1, then $\{k_j^{s^{'}} : value_j^{s^{'}}, k_j^{O^{'}} : value_j^{O^{'}}\} \in$

$\{ABACPolicy\_S, ABACPolicy\_0\}$ is indicated.

Step 5: Loop through Step 3 and Step 4. If all elements$\{ABACPolicy_S^{'}, ABACPolicy_O^{'}\}$ and $\{k_0^{s^{'}} : value_0^{s^{'}}, \ldots, k_i^{s^{'}} : value_i^{s^{'}}, \ldots, k_{x-1}^{s^{'}} : value_{x-1}^{s^{'}}\} \in ABACPolicy\_S^{'}$ in $\{k_0^o : value_0^o, \ldots, k_i^O : value_i^O, \ldots, k_{x-1}^O : value_{x-1}^O\} \in ABACPolicy\_O^{'}$ exist in $\{ABACPolicy\_S\}$ and $\{ABACPolicy\_O\}$, the query policy matches, and the query ends.

The above steps simplify the computational overhead of the query, quickly exclude elements that are not part of the attribute set, and the computational complexity $O(p)$ is only related to the computational efficiency. The computational complexity for querying the target contract is reduced from $O(t^2)$ to $O(p * t)$.

5.4.4 System Workflow

As shown in Figure 5.4, the blockchain-based access control policy model mainly consists of five parts, and this part will explain the workflow of each part in detail.

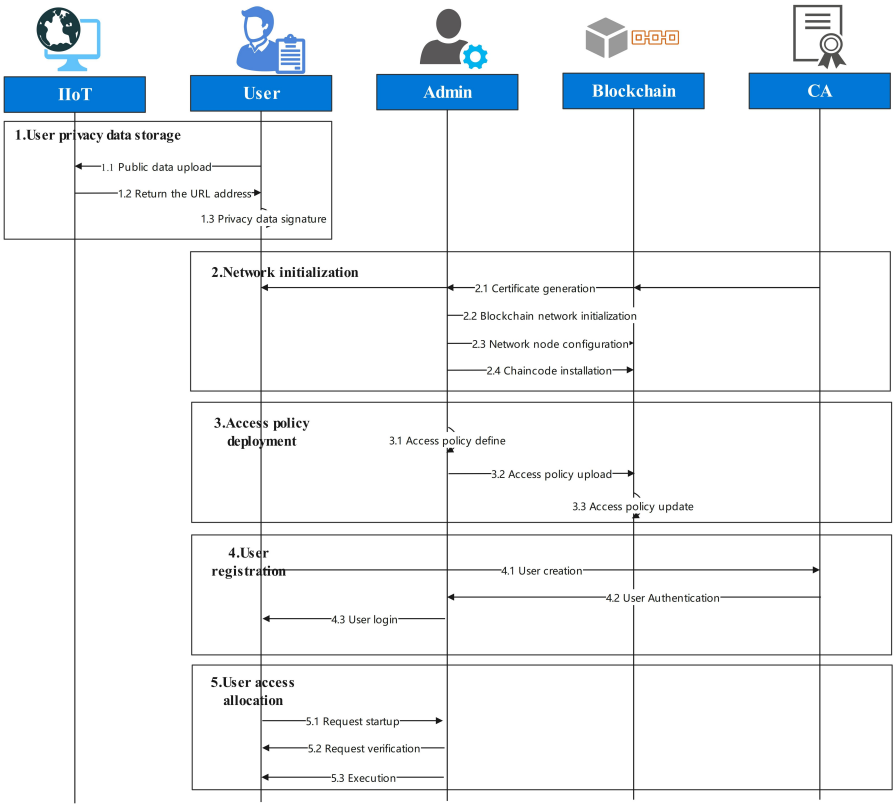


Figure 5.4: Workflow for Hyper-IIoT solutions

5.4.4.1 Data Storage

Data owners first need to upload their private data resources and establish corresponding access control policies to finalize the storage procedure of private data within the blockchain network. Suppose that there is a user $user1$ with $priData$, $pubData$, and public key $PKowner$, so the specific process of storing private data is as follows:

Step 1: The user transfers public data to the cloud server.

Step 2: The cloud server provides URL data in response.

Step 3: The user must hash the private data and obtain the result sign_data.

5.4.4.2 Blockchain Network Initialization

After configuring the experimental environment, the system must be started first. The initial setup of the blockchain network mainly includes the generation of certificates for each part of the blockchain, the creation and connection of nodes between the network, and the installation of chain codes and policies on each node. The specific implementation process is executed in the form of a script, and the steps are as follows:

Step 1: The certificate is generated. The certificates, public and private keys of User, Orderer, Peer, and Admin are uniformly generated by the CA, as shown in Equation 5.10:

$$\begin{cases} User \rightarrow CA \rightarrow \{Ct_{User}, pubk_{User}, prik_{User}\} \\ Orderer \rightarrow CA \rightarrow \{Ct_{Orderer}, pubk_{Orderer}, prik_{Orderer}\} \\ Peer \rightarrow CA \rightarrow \{Ct_{Peer}, pubk_{Peer}, prik_{Peer}\} \\ Admin \rightarrow CA \rightarrow \{Ct_{Admin}, pubk_{Admin}, prik_{Admin}\} \end{cases} \tag{5.10}$$

Step 2: Creation and connection of blockchain network nodes. Each user in the network constitutes a node. In a stand-alone environment, the creation of each node is equivalent to the creation of a docker container, ensuring that each node has the same system environment. In a multi-machine environment, each computer is equivalent to a node and does not need to have the same environment, it only needs to have the same client to achieve communication between nodes. For constructing a blockchain network, employing the Fabric-provided configtxgen tool *ctgen* is essential, with the process detailed in Equation 5.11.

$$Fabric \xrightarrow{ctgen} Block_1 \xrightarrow{kafka} \{Block_1, Block_2, \ldots, Block_n\} \tag{5.11}$$

Step 3: Configure network nodes. The configuration information for nodes and channels is written to the transaction as shown in Equation 5.12.

$$\{Ct_{Orderer}, Cf_{Orderer}, Ct_{Peer}, Cf_{Peer}\} \xrightarrow{SDK} \{T_{x_1}, \ldots, T_{x_n}\} \tag{5.12}$$

Step 4: Chaincode installation. chaincode installation is performed by the administrator according to Equation 5.13. Installing a chaincode on a blockchain network node can facilitate the authentication and endorsement of transactions on that node.

$$Chaincode \rightarrow Admin \overset{SDK}{\rightarrow} Peer \tag{5.13}$$

5.4.4.3 Access Policy Deployment

The system customizes access policies to ensure users' access rights to relevant data. The deployment procedure is as follows.

Step 1: Access policy definition. Admin provides management functions for attribute-based access control policies developed based on business rules, as shown in Equation 5.14:

$$\{S, O, P, E\} \rightarrow Admin \rightarrow \{ABACPolicy\} \tag{5.14}$$

Step 2: Access policy upload. Once the access policy is defined, it will be broadcast to the blockchain network in the steps of Equation 5.15.

$$\{ABACPolicy\} \rightarrow Admin \rightarrow \{Block_1, Block_2, \ldots, Block_n\} \tag{5.15}$$

Step 3: Access policy updates. The blockchain network does this by executing chaincode, updating the Ledger state, and saving the policy to the database, as shown in Equation 5.16:

$$\{ABACPolicy\} \rightarrow \{Ledger, LevelDB\} \tag{5.16}$$

5.4.4.4 User Registration

Registering users involves recording the identity details of data owners and requesters on the blockchain. The steps for registration are outlined as follows.

Step 1: User creation. This step first needs to check whether $UserName$ and $UserNumber$ already exist in the CA. If the member is new, the CA proceeds with registering the $User$, leading to the creation of the fields $\{SignID, Pubk, Prik, Ct\}$, as shown in Equation 5.17:

$$User \rightarrow CA \rightarrow \{SignID, Pubk, Prik, Ct\} \tag{5.17}$$

$UserName$ and $UserNumber$ are then stored in the database, as shown in Equation 5.18:

$$\{UserName, UserNumber\} \rightarrow \{Ledger, LevelDB\} \tag{5.18}$$

Step 2: User authentication. After registration, if the user wants access to create, query, and update the corresponding data, they need to obtain authentication by calling the $checkUser()$ method and then invoke the pre-set access policy to obtain authorization as shown in Equation 5.19.

$$User \rightarrow CA \overset{checkUser()}{\longrightarrow} Admin \rightarrow Policy_{User} \tag{5.19}$$

Step 3: User login. Upon user authentication, the server initially verifies the existence of the member. Once confirmed, the member's public and private keys, Pub_k and Pri_k, respectively, facilitate access to the Fabric blockchain network. Subsequently, the member's data retrieval is executed through the $queryUser()$ function. A successful data query triggers the server to dispatch the retrieved information and deliver the login outcome to the member.

5.4.4.5 User Access Allocation

User access authorization is the core of access control. Authorization must ensure sharing and dynamics while ensuring the integrity and synchronization of financial project data. User access authorization requires strict policy definition based on actual service logic. The assignment steps are as follows.

Step 1: Request activation. The user initiates an access request to the blockchain, which is verified upon arrival at Admin as shown in Equation 5.20.

$$User \overset{Request}{\longrightarrow} Admin \tag{5.20}$$

Step 2: Request validation. After receiving an access request from the user, Admin calls the $GetAtrrs()$ method to get the user properties and then the $CheckAccess()$ method for validation as shown in Equation 5.21.

$$\begin{aligned} &Admin \overset{GetAttrs()}{\longrightarrow} User\{S,O\} \longrightarrow User\{S,O,E\} \\ &Admin \overset{Checkaccess()}{\longrightarrow} User\{S,O,E\} \longrightarrow User\{S,O,E,P\} \end{aligned} \tag{5.21}$$

Step 3: Execute. After the administrator verifies the user's access policy, the system processes the user's request in line with the established access policy. If the return result is 1, the access is passed. If the return result is 0, the access is denied, as shown in Equation 5.22.

$$Admin \overset{ABACPolicy}{\rightarrow} User\{S,O,E,P\} \begin{cases} 1, & Pass \\ 0, & No \end{cases} \tag{5.22}$$

5.5 Performance Evaluation and Analysis

This section describes the experimental process for evaluating functionality and presents the performance and comparative results of the proposed Hyper-IIoT. Section 5.5.1 details the experimental setup and the parameters configured for the evaluation, while Section 5.5.2 delves into the analysis of the experimental data and the consequent performance implications.

5.5.1 System Environment and Configuration

The experimental evaluation in this study is executed in a stand-alone environment, configured as detailed subsequently.

5.5.1.1 Network Structure

The proposed network topology contains nine types of nodes. This setup comprises four database nodes ($Fabric - couchdb$), two certificate authority nodes ($Fabric - ca$), four peer nodes ($Fabric - peer$), one order node ($fabric - orderer$), and one client node ($fabric - tools$). Additionally, there are four nodes each for user-managed contracts (UMC and AMC), private data control contracts (PCC), and access control policies (PMC), as shown in Table 5.1. We emulate a distributed multi-machine environment utilizing three Raspberry Pi microcomputers, which are depicted in Figure 5.5.

Table 5.1: Symbols used.

Node	Implication	Number
UMC	The user manages the contract nodes	4
AMC	The user manages the contract nodes	4
PCC	Private data control contract nodes	4
PMC	Access control policy nodes	4

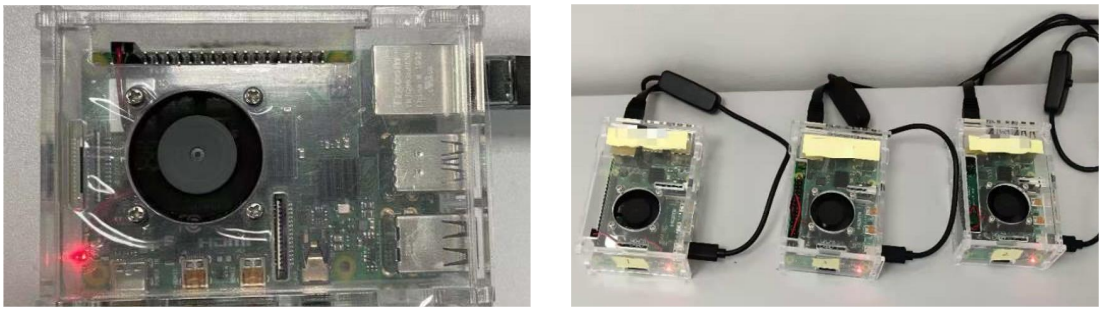


Figure 5.5: Raspberry Pi microcontroller devices

5.5.1.2 Chaincode Deployment

Within the Hyper-IIoT framework, chaincode is responsible for defining properties, managing access control, and orchestrating system operations. This encompasses the processes of installation, instantiation, and subsequent upgrades.

Step 1: Installation. Chaincode installation is initiated after completing blockchain network initialization, performed through Hyperledger client nodes, and installed into each peer node.

Step 2: Instantiation. The instantiated chain code is specified after it has been installed on any peer node

Step 3: Upgrade. Before updating the chain code, it is imperative to install the new version; the update will take effect exclusively on peer nodes that have the revised chain code in place.

5.5.2 Performance Test and Experimental Results

To evaluate Hyper-IIoT's performance, this study conducts three sets of simulation experiments to assess system concurrency, transaction time, and throughput under real-world conditions, demonstrating its ability to maintain high throughput and effective consensus in a distributed system even under large-scale request scenarios.

5.5.2.1 Contract Execution Time

Rooted in the operational logic, this research initially assesses the execution time for user management, policy management, and access control contracts, as shown in Figure 5.6, Figure 5.7, Figure 5.8, and Figure 5.9. These results highlight a consistent and gradual rise in contract execution time corresponding to increased block size, indicative of a well-regulated and systematic growth pattern. Moreover, it is evident that the *add*() and *update*() methods require more time to execute compared to the *query*() and *delete*() methods, primarily due to the greater number of data writes they involve. However, this difference in execution time is sufficiently minor to be effectively managed, underscoring the robustness of the system's performance under varying workloads.

Furthermore, Hyper-IIoT integrates a bloom filter-enhanced API into the smart contract in Hyperledger fabric. Figure 5.10 presents the efficiency of bloom filters in processing queries across varying file sizes. The data indicate a systematic increase in processing time correlating with heightened concurrency, underscoring the bloom filter's compatibility with scalable data storage and query scenarios. For query precision, the bloom filter's byte size and hash function count are finely tuned by the anticipated data dimensions. Subsequently, the MD5 hash is stored as a key-value pair within the state database, prompting an update

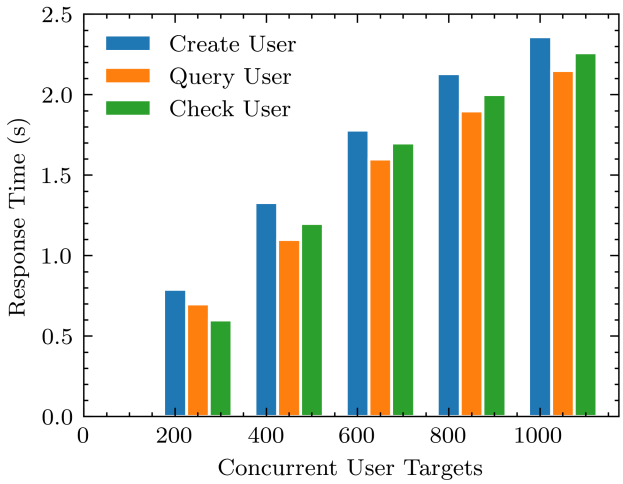


Figure 5.6: The execution time of UMC

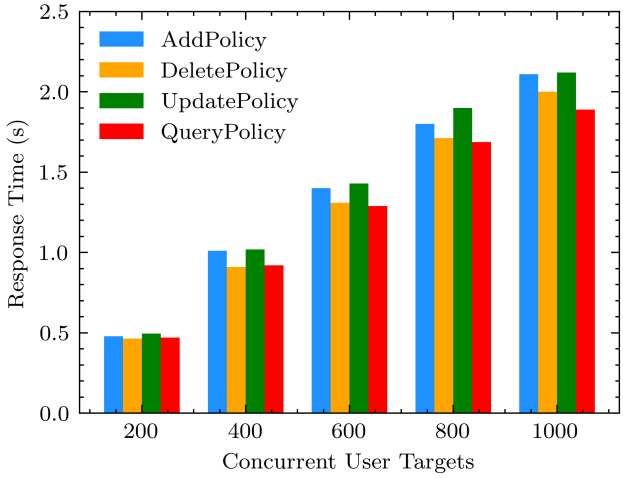


Figure 5.7: The execution time of PMC

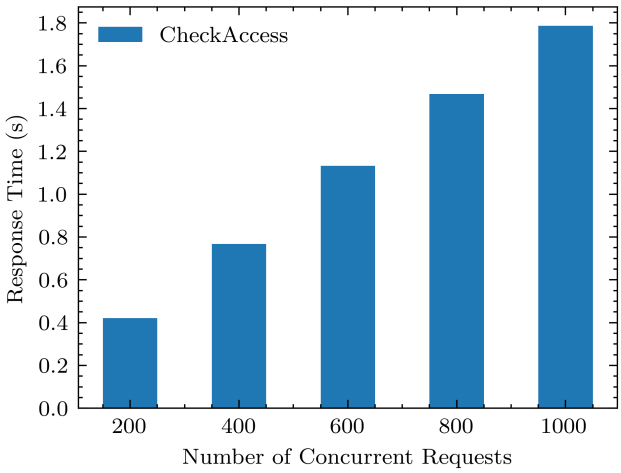


Figure 5.8: The execution time of ACC

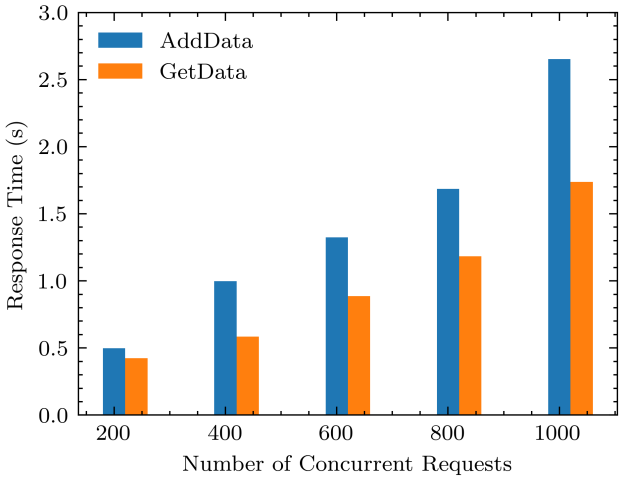


Figure 5.9: The execution time of PCC

to the ledger. A comparative analysis between the bloom filter-enabled and traditional search methodologies underpin the augmented efficacy of the bloom filter approach as data volume escalates, as illustrated in Figure 5.11.

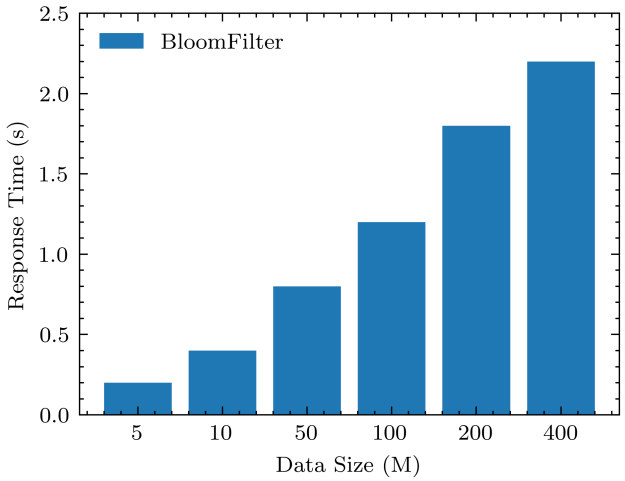


Figure 5.10: Query time consumption of bloom filters

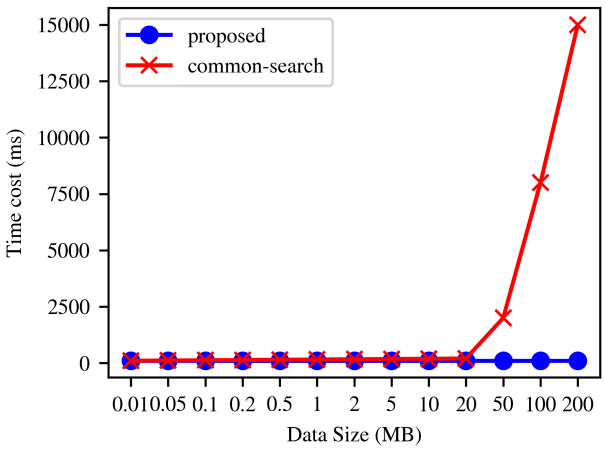


Figure 5.11: Comparing the query time between the bloom filter-assisted scheme and the regular scheme

5.5.2.2 TPS of Smart Contracts

This section provides a detailed evaluation of the Transaction Per Second (TPS) of Hyper-IIoT across simulated scenarios with varying concurrency levels. The system's throughput, particularly under the Kafka consensus protocol, demonstrates unparalleled reliability and

stability, as evidenced by the comprehensive analysis depicted in Figure 5.12, Figure 5.13, Figure 5.14, and Figure 5.15. It is worth noting that while the administrative contract, responsible for addition and modification functions, exhibits a slightly reduced throughput compared to the contract focused on querying and validation, this discrepancy arises due to the inherent read and write demands associated with administrative tasks. Nevertheless, the Hyper-IIoT system consistently maintains robust TPS and stable performance, even under stringent operational demands, reaffirming its suitability for industrial applications.

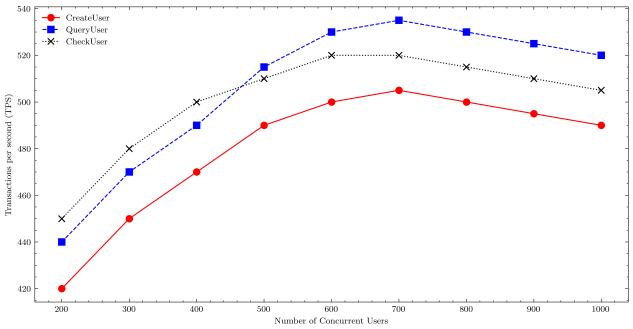


Figure 5.12: TPS of UMC

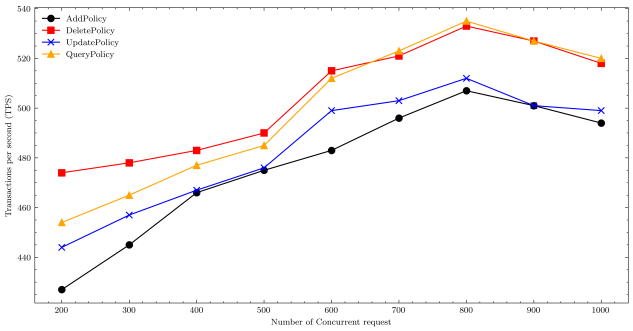


Figure 5.13: The execution time of PMC

5.5.2.3 Consensus Time Comparison

A comprehensive comparison of the consensus duration and throughput between the Kafka and PoW consensus mechanisms across a network spanning from 10 to 100 nodes was conducted in this section. Figure 5.16 provides detailed insights into the time consumption required to achieve consensus using the proposed system, shedding light on the efficiency of each consensus mechanism. Besides, Figure 5.17 shows the significant advantage of the Kafka consensus algorithm over PoW, particularly in meeting the high-throughput demands essential for this IIoT application. These findings offer valuable guidance in selecting the

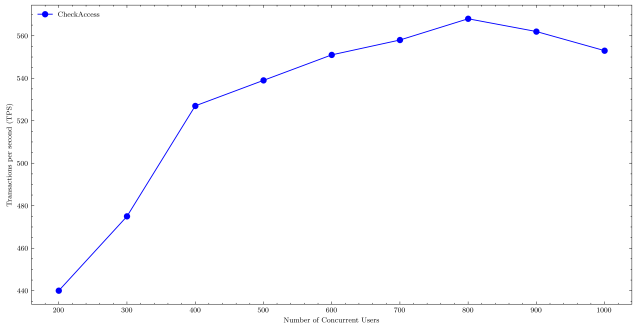


Figure 5.14: The execution time of ACC

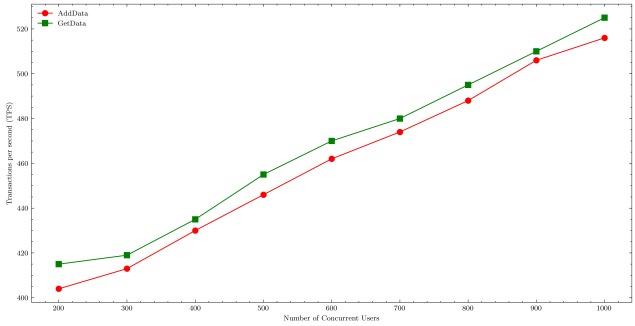


Figure 5.15: The execution time of PCC

most suitable consensus mechanism for IIoT systems, ensuring optimal performance and scalability.

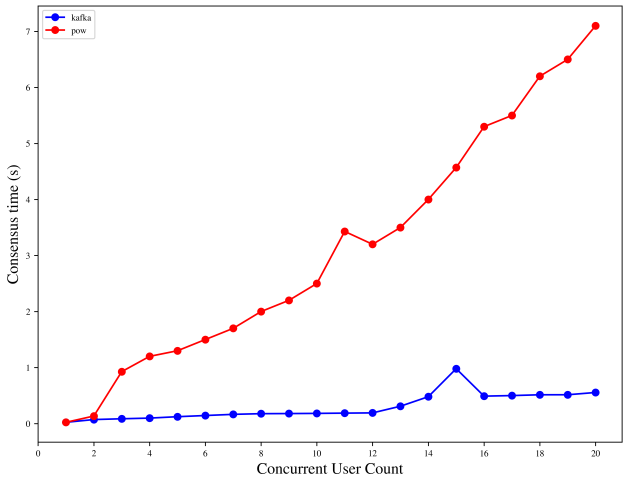


Figure 5.16: The Consensus time comparison between Kafka and PoW

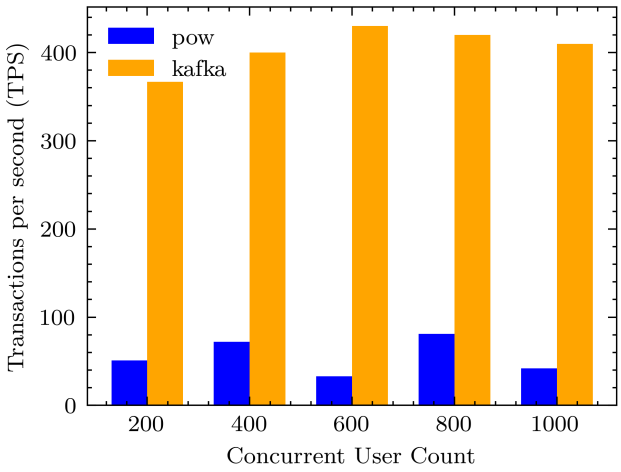


Figure 5.17: Consensus time comparison

5.6 Conclusion

This chapter proposes an IIoT data access control framework based on smart contracts and bloom filters to achieve secure data storage and access control, named Hyper-IIoT. Hyper-IIoT expands the support for accurate data governance and access control, preserving the concepts of safe data storage, effective data access, and the traceability of sensitive data

access to handle evolving data formats and rising privacy needs. The access policy in the system is programmable, offering adaptability to diverse access requisites. Through the implementation of ABAC, Hyper-IIoT has accomplished fine-grained access control, and the bloom filter deployment improves system performance. The experimental results show that Hyper-IIoT exhibits well-controlled contract execution times, stable throughput, and efficient consensus in a distributed environment and highlight the advantages of the Kafka consensus mechanism in achieving rapid consensus and high throughput, meeting the requirements in real-world applications.

Chapter 6

Non-Interactive ZKP-inspired Access Control

Contents

6.1 Introduction

The development of the IIoT represents a significant change in the digital age, fundamentally reshaping the way industries operate and interact with technology. Information-centric networking (ICN) is a strategic advance toward improved data localization in the context of the IIoT [139]. This change emphasizes the need for sophisticated security measures, particularly access management. Controlling access is essential in IIoT environments to protect critical industrial data's integrity and confidentiality [140]. Given the dynamic and multifaceted nature of IIoT networks, traditional host-centric access control approaches are proving insufficient [141]. Therefore, to adapt to the decentralized and content-centric characteristics, a more flexible, scalable, and security-focused strategy is urgently needed for the IIoT environment based on ICN.

Nevertheless, with the increasing adoption of real-time data services in IIoT, traditional methods for P2P packet transfer have faced significant challenges. These include: 1) the heightened risk of a single point of failure, which can precipitate informational redundancy and squander resources within clustered networks [142]; 2) reduced system throughput attributable to the limitations imposed by bandwidth and computational capacity, which can severely hinder operational efficiency [143, 144]; 3) compromised data security, stemming from encryption protocols that fail to meet the demanding standards of contemporary IIoT frameworks; and 4) a suboptimal tolerance for network disruptions, reflecting the inherent volatility of IIoT communication infrastructures [145]. Adding to these technical challenges is the fundamental nature of most IIoT applications, which prioritize content-centricity. Therefore, a thorough reassessment of traditional communication protocols is necessary in light of this shift towards an ICN architecture to ensure they align with the constantly evolving needs of the IIoT ecosystem.

In this context, ICN has been adopted in IIoT implementations for its efficient data distribution capabilities for high-demand content [146]. As shown in Figure 6.1, ICN introduces the innovative concept of Named Data Objects (NDOs), leverages router caching, employs multicast communication, and decouples senders from receivers to optimize data dissemination efficiency. Despite these advantages, ICN implementation involves managing complex data structures and codebases. Additionally, a significant concern is the inherent lack of encryption in NDOs, potentially exposing sensitive information to unauthorized entities and posing a risk to data owner privacy [147]. Another critical consideration is the need for robust authentication and access control mechanisms within ICN-based IIoT implementations to mitigate the risk of unauthorized access and data breaches.

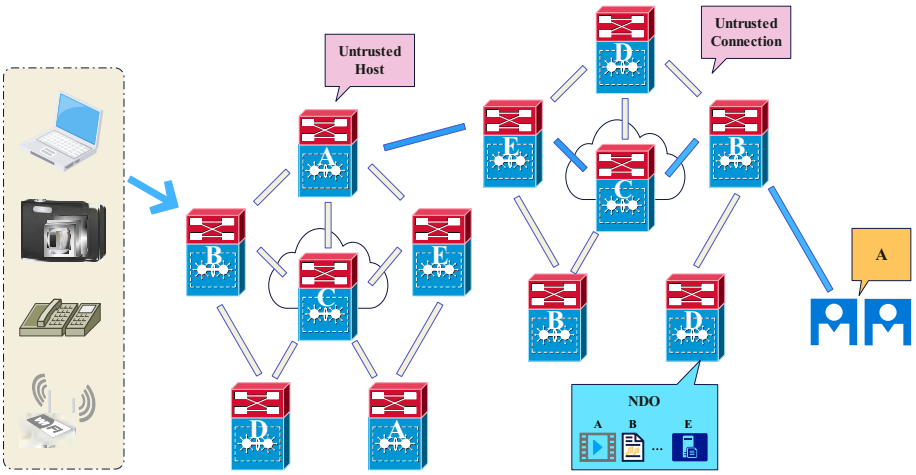


Figure 6.1: ICN structure

However, traditional Zero-Knowledge Proof (ZKP) systems typically require communication between the prover and verifier, which might cause constraints in large-scale, distributed IIoT networks [148]. In contrast, non-interactive ZKPs allow proof verification without such interaction, making them highly suitable for decentralized environments like IIoT [149, 150]. Compared with traditional asymmetric encryption techniques (like RSA), which require data to be encrypted multiple times for each recipient, ZKP encrypts data only once as opposed to N times for each recipient, significantly reducing resource consumption and streamlining the encryption process while preserving data security [151]. Simultaneously, non-interactive ZKP provides IIoT with a more private and trustworthy verification method without needing a trusted third party [152].

Considering the advantages and limitations of ICN, Ciphertext-Policy Attribute-Based Encryption (CP-ABE), and blockchain technology, this manuscript aims to effectively address the security challenges and comply with the performance standards within the IIoT framework by strategically integrating CP-ABE with ICN in the IIoT context, thus efficient and distributed data sharing while maintaining robust user privacy through meticulous access control mechanisms. Moreover, the manuscript pioneers an innovative approach by embedding a non-interactive ZKP protocol within the CP-ABE framework, thereby introducing the concept of Distributed Publish-Subscribe Industrial Internet of Things (DPS-IIoT), which assures the authenticity of private keys, significantly minimizing the bandwidth consumption typically associated with ineffective access control.

6.2 Related Work

This section reviews the integration of CP-ABE and blockchain technologies in IIoT, focusing on three main aspects: CP-ABE application in IoT in section 6.2.1, the combination with blockchain in section 6.2.2, and the role of ICN under IIoT scenarios in in section 6.2.3.

6.2.1 Attribute-Based Encryption with IoT

The integration of CP-ABE in IIoT mainly targets enhancing efficiency in key management, access structuring, and algorithmic complexities. For example, Li *et al.* [153] introduced a robust data-sharing system that operates on a distributed, publisher-driven model, aiming at reducing the communication overhead in CP-ABE systems. In another study, Li *et al.* [154] combines CP-ABE with identity-based signature (IBS) to enable distributed authentication in big-data sharing environments. Zhao *et al.* [155] focused on attribute revocation, proposing an outsourced CP-ABE scheme that maintains fixed ciphertext and key sizes. Lyes *et al.* [156] addressed vital management challenges in dynamic IoT environments with a Batch-Based CP-ABE system. Finally, Hao *et al.* [157] developed a CP-ABE system supporting entirely hidden attribute-based access policies using a garbled Bloom filter.

6.2.2 Access Control with Blockchain

Integrating blockchain technology with access control mechanisms has brought about innovative solutions. For instance, Wang *et al.* [158] combined Ethereum and ABE for secret critical supervision and fine-grained data access control. Fan *et al.* [159] utilized blockchain for recording access policies and implemented a user self-authentication system. Then, a unique integration of blockchain and CP-ABE was proposed in [160], addressing challenges in blockchain regulation and privacy protection. Zhang *et al.* [161] combined ABE with blockchain in IoT and cloud computing environments, using byzantine fault-tolerant mechanisms for faster consensus. Han *et al.* [162] developed an innovative ABAC model, focusing on auditable access control to enhance privacy within IoT environments.

6.2.3 ICN with IoT

The application of ICN in IoT is an evolving research field. ICN enhances the availability of IoT data through distributed data caching while introducing new challenges in the authentication field. Xue *et al.* [163] explored the impacts of EDOS attacks in CP-ABE encrypted cloud systems. Meanwhile, Salvador *et al.* [164] introduced an innovative architecture merging CP-ABE's adaptability with symmetric key encryption's efficiency, aiming for secure data transfer and privacy. While current research demonstrates significant advancements

in integrating CP-ABE, blockchain, and ICN within IoT environments to enhance security, efficiency, and scalability, there remains a critical need for further exploration into optimized, cohesive solutions that address the rapidly evolving and specific challenges of IIoT systems.

6.3 System Model

This section describes the proposed asynchronous distributed network, DPS-IIoT, designed by combining the core principles of ICN with the advanced framework of IIoT for content dissemination. The entity definition and workflow are introduced in section 6.3.1 and section 6.3.2, respectively.

6.3.1 Entity Definition

The architecture of DPS-IIoT is constituted by six crucial elements: the Device, generating data; the Publisher, responsible for data issuance; the Rendezvous Node, acting as the system's directory; the Forwarding Node, facilitating data transit; the Subscriber, the end consumer of the information; and the Blockchain, a foundational component that fortifies the network's security and integrity. These components collectively form the backbone of the DPS-IIoT, as shown in Figure 6.2.

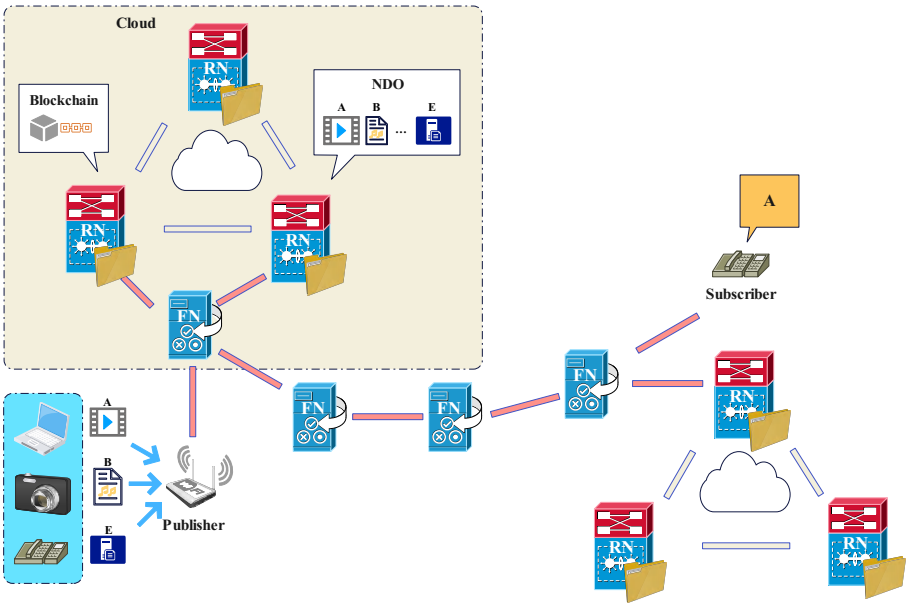


Figure 6.2: Entity definition of DPS-IIoT architecture

Device (D): In the IIoT ecosystem, a device, represented as D, is the source of data

generation, producing Named Data Objects (NDO_i). Each NDO_is is distinctively tagged within a specified namespace $\mathcal{N}$, playing a critical role in the network's data dynamics. These NDO_is can be strategically duplicated across various nodes within the system, a measure that significantly boosts the network's redundancy and resilience.

Publisher (P): The Publisher denoted as P, is central in structuring the IIoT landscape. It exerts control over the namespace $\mathcal{N}$ and meticulously formulates access control policies, represented as Π. When these policies are coupled with the NDO_i, they facilitate controlled dissemination of information towards Rendezvous Nodes, thereby effectively regulating access to the data.

Rendezvous Node (RN_k): The Rendezvous Node, labelled RN_k, is a crucial intermediary within the architecture. It serves a dual role: resolving NDO_i requests and temporarily storing these data objects. By implementing encryption based on predefined policies Π, the Rendezvous Node ensures secure and effective policy enforcement. As an integral part of the blockchain network, the Rendezvous Node oversees secure policy execution and facilitates efficient content navigation.

Forwarding Node (FN_l): The Forwarding Node (FN_l) is pivotal in orchestrating the network's transmission architecture, overseeing the efficient conveyance of NDO_i towards the designated Rendezvous Nodes. Acting as a crucial intermediary, FN_l optimizes data dissemination by strategically routing and forwarding information packets through the network. Additionally, it adjusts transmission settings to match changing network conditions, ensuring timely and reliable data delivery. By actively managing and optimizing the network, FN_l improves overall performance and resilience, enabling smooth communication within the IIoT ecosystem.

Blockchain: The blockchain, crucial to the system's security, functions as a distributed ledger $\mathcal{L}$. This ledger $\mathcal{L}$ supports the network's structural integrity and security framework. It securely manages the network's access control processes via smart contracts $\mathcal{SC}$, ensuring that policy management Π is governed by immutable rules ρ_Π and transparent verification protocols $\mathcal{V}_\Pi$. Mathematically, the blockchain's role can be expressed as $\mathcal{L}(\mathcal{SC}, \Pi) \rightarrow \{\rho_\Pi, \mathcal{V}_\Pi\}$, highlighting its foundational role in establishing a robust layer of trust and reliability across the network's operations.

Subscriber (Σ): Subscribers, denoted as Σ, are dynamically engaged with the network, actively requesting specific Named Data Objects (NDO_i) from Rendezvous Nodes via requests tagged as msg_{sub}. These Σ entities are tuned to receive updates and content that match their interests in the namespace $\mathcal{N}$, demonstrating their proactive involvement in the network's data exchange ecosystem.

6.3.2 Workflow

The architecture of DPS-IIoT is designed to orchestrate secure and efficient data management intricately. As shown in Figure 6.3, the system's operational sequence is carried out in four phases.

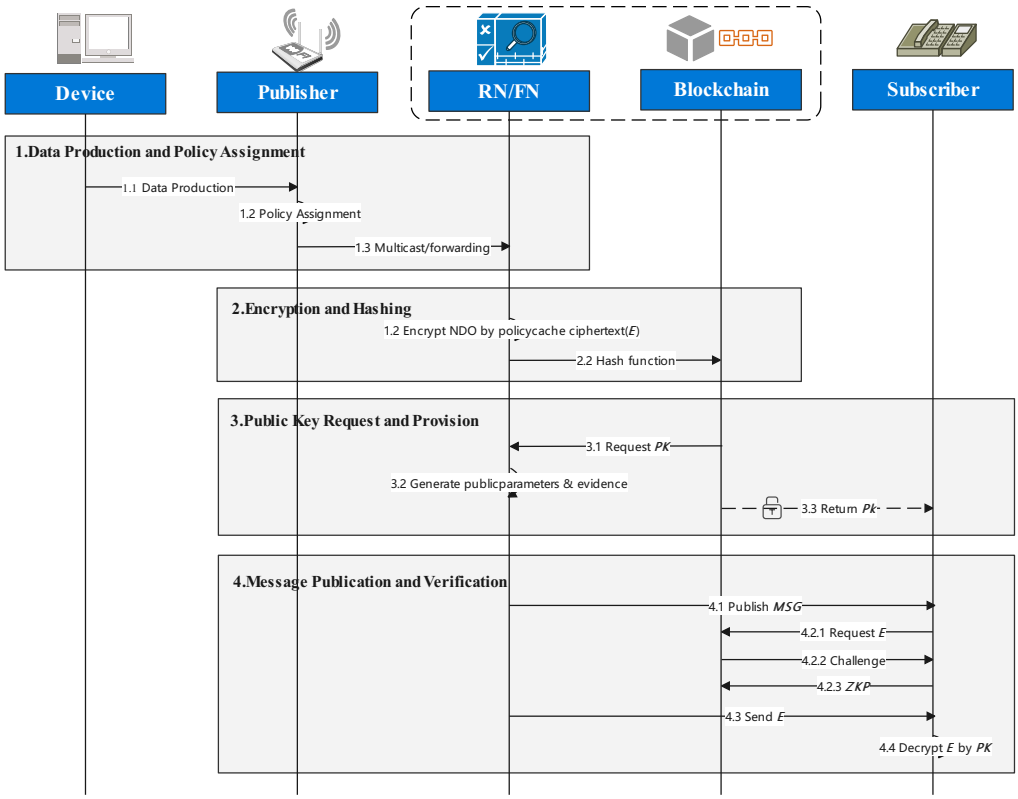


Figure 6.3: Workflow of DPS-IIoT

Step 1: Data Production and Policy Assignment.

- Produce data: Let D be the data produced by a device, such that $D \in \mathcal{D}$, where $\mathcal{D}$ is the set of all possible data.
- Assign name scope, make policy: A policy π is defined as a function $\pi : \mathcal{D} \to \mathcal{P}$, where $\mathcal{P}$ is the set of all possible policies. The publisher assigns the data a naming scope N and policy $\pi(D)$.
- Multicast/Forwarding: The data D is multicast or forwarded to nodes in the network $\mathcal{N}$, where $\mathcal{N}$ is the set of all nodes.

Step 2: Encryption and Hashing.

- Encrypt NDO by policy, cache ciphertext(E): The encrypted data E is given by $E = \text{Enc}_\pi(D)$, where Enc is the encryption function parameterized by policy $\pi(D)$.
- Put HASH: A hash function $h : \mathcal{D} \to \mathcal{H}$ maps data D to a hash value $h(D)$, where $\mathcal{H}$ is the set of all possible hashes. The hash of the encrypted data is $h(E)$.

Step 3: Public Key Request and Provision.

- Request PK: A subscriber requests the public key PK, where $PK \in \mathcal{K}$ and $\mathcal{K}$ is the set of all possible public keys.
- Generate public key parameters + validation proof: Parameters θ and validation proof ϵ are generated, where $\theta, \epsilon \in \mathcal{K}$.
- Return PK: The public key PK is returned, $PK = f(\theta, \epsilon)$, where f is a function generating PK from parameters θ and validation proof ϵ.

Step 4: Message Publication and Verification.

- Publish MSG: Let M be a message published, $M \in \mathcal{M}$, where $\mathcal{M}$ is the set of all possible messages.
- Request E: The encrypted data E is requested by the subscriber.
- Challenge: A challenge C is issued, $C \in \mathcal{C}$, where $\mathcal{C}$ is the set of all possible challenges.
- ZKP: A zero-knowledge proof function $\mathcal{Z}$ confirms that the subscriber can access E without revealing D or PK.
- Send E: The encrypted data E is sent to the subscriber after successful verification.
- Decrypt E by PK: The subscriber decrypts the encrypted data E using the public key PK, to retrieve the original data D. This is represented as $D = \text{Dec}_{PK}(E)$, where Dec is the decryption function.

This architecture ensures stringent access control, maintaining data integrity and privacy throughout the data's lifecycle in the DPS-IIoT environment.

6.4 Proposed ZK-CP-ABE Scheme

This section introduces the detailed algorithm design. The protocol for initial identity authentication using non-interactive ZKPs, $Auth(ID_{entity}) \xrightarrow{ZKP} Verified$, is at the basis of the suggested approach. This process obviates the need for traditional public key exchange mechanisms and mitigates the risks associated with private key exposure.

6.4.1 Optimized Decryption in CP-ABE Systems for IIoT Environments

In standard CP-ABE frameworks, the decryption model denoted as $\mathcal{D}(C, PK, SK)$, mandates the complete download of ciphertext C aligned with the access policy $\mathcal{P}$. This protocol leads to high computational and bandwidth utilization, particularly when $\mathcal{D}(C, PK, SK)$ fails due to key errors, resulting in wasted data transmission (Δ_{trans}) and decryption efforts ($\Delta_{decrypt}$). The algorithm adopts a selective approach by computing the cryptographic hash function $H(\cdot)$ over the original data D, yielding a condensed representation denoted as $H(D)$. Subsequently, only this hashed value is encrypted using the encryption function $\mathcal{E}(\cdot)$, resulting in ciphertext $\mathcal{C} = \mathcal{E}(H(D), \mathcal{A})$. By encrypting the hash instead of the entire dataset, the algorithm achieves a substantial reduction in the volume of encrypted data, optimizing bandwidth usage and enhancing overall system efficiency. Before decryption, the algorithm embeds an essential authentication phase, $\mathcal{A}(C, PK, SK)$, employing a ZKP mechanism $\mathcal{ZKP}$ to validate the legitimacy of the decryption request. This pre-validation step aims to verify the authenticity of the decryption request before proceeding, ensuring authorized access to the encrypted data.

6.4.2 System Setup and Key Generation

The setup of DPS-IIoT begins with the selection of a sizable prime p and the creation of a bilinear group G_0. Within this group, a generator g is chosen, and a hash function $H : \{0,1\}^* \rightarrow G_0$ is designated as a random oracle to map binary attributes to group elements. Two random numbers α and β are then selected from the group of integers under multiplication modulo p, and system parameters $h = g^\beta$ and $u = e(g,g)^\alpha$ are computed. Finally, the system parameters are organized into public key $PK = \{G_0, g, h, u\}$ and master key $MK = \{\alpha, \beta\}$.

6.4.3 Generation of User-Specific Keys and validation proof

The design of this setup fulfils two key objectives: firstly, it computes the private key (also called secret key) SK by utilizing initial parameters and user-specific attributes; secondly, it generates public proof for SK to enable ZKP validation. Let Y represent a collection of essential attributes with assigned weights crucial for decryption. The algorithm selects a random value r from the finite field $\mathbb{Z}_p$. Each attribute y_j in Y independently assigns a unique random value r_j from $\mathbb{Z}_p$.

Algorithm 9 is designed to generate validation proof supporting the possession of the private key SK. Let n be a large composite number, specifically the product of two distinct large primes, p and q, such that $n = p \cdot q$. The length of the validation proof VP in bytes is represented by t, which is determined by the system's desired security level. This

Algorithm 9 Generate Key and validation proof

Require: $Y = \{y_1, y_2, \ldots, y_m\}$ (weighted attributes), $MK = \{\alpha, \beta\}$ (master key)
Ensure: SK (private key), VP (validation proof for SK)
1: Select a random value r from the finite field $\mathbb{Z}_p$
2: Initialize $SK = \{\}$
3: **for** each attribute y_j in Y **do**
4: Select a unique random value r_j from $\mathbb{Z}_p$
5: Compute $D_j = g^r \cdot H(y_j)^{r_j}$
6: Compute $D'_j = g^{r_j}$
7: Add D_j and D'_j to SK
8: **end for**
9: Compute $D = g^{(\alpha+r)/\beta}$ and add to SK
10: Choose large primes p and q, set $n = p \cdot q$
11: Select length t for validation proof VP
12: Compute $X = |Hash256(SK)[0, ..., t]|^2$
13: Formulate $VP = \{X, n = p * q\}$
14: **return** SK, VP

relationship is formalized in Equation 6.1.

$$\begin{aligned} X &= \left(\sum_{i=0}^{t} \text{Hash256}(SK)_i \right)^2 \\ &= \left(\sum_{i=1}^{t} y_i \right)^2 \\ &= \sum_{i=1}^{t} y_i^2 \\ &= \sum_{i=1}^{t} x_i \\ VP &= \{X, n = pq\} \end{aligned} \tag{6.1}$$

6.4.4 Encryption of Data under Access Control Policies

Initially, a hash function $\mathcal{H}$ is applied to the dataset M to generate a data summary $M_{meta} = \mathcal{H}(M)$. This technique effectively enhances storage efficiency and reduces bandwidth during data transmission.

As Algorithm 10 shows, the encryption process initiates at the root node R of $\mathcal{T}$ and proceeds recursively to each node x, where it constructs a unique polynomial q_x. Utilizing Lagrange interpolation, defined as $f(x) = \sum_{i=1}^{d} f(x_i) \left(\prod_{j=1, j\neq i}^{d} \frac{x - x_j}{x_i - x_j} \right)$, the polynomial q_x

Algorithm 10 Encrypt Data

Require: M (plaintext), A (access structure), $PK = \{G_0, g, h, u\}$ (public key)
Ensure: CT (ciphertext)

Compute $M_{\text{meta}} = \mathcal{H}(M)$ using hash function $\mathcal{H}$
Initialize access tree $\mathcal{T}$ based on A
Select a random secret s from $\mathbb{Z}_p$
Define polynomial q_R at root R with $q_R(0) = s$
for each node x within the tree $\mathcal{T}$ **do**
    if x equals the root node R **then**
        Proceed to next node
    else
        Set $q_x(0)$ equal to the value of $q_{\text{parent of } x}$ evaluated at the index of x
        Randomly select d_x points to fully establish the polynomial q_x
    end if
end for
Compute $\bar{C} = M \cdot e(g, g)^{\alpha \cdot s}$ and $C = h^s$
for each leaf node l in $\mathcal{T}$ **do**
    Compute $C_l = g^{q_l(0)}$
    Compute $C'_l = H(\text{att}(l))^{q_l(0)}$
end for
Assemble ciphertext $CT = \{\mathcal{T}, \bar{C}, C, \{C_l, C'_l\}_{\forall l \in \text{Leaves}(\mathcal{T})}\}$
return CT

at each node x is defined with a maximum degree d_x equal to the threshold of the node, $k_x - 1$. The encryption process initiates at the root node R, where a random secret s is chosen from the set $\mathbb{Z}_p$. This secret establishes the initial condition of the polynomial q_R at R by setting $q_R(0)$ equal to s. For each subsequent node x within the tree $\mathcal{T}$, the zeroth value of their respective polynomial $q_x(0)$ is determined based on the value of the polynomial at their parent node, specifically as $q_{\text{parent of } x}(\text{index of } x)$. To completely define the polynomial q_x at each node x, d_x distinct points are randomly selected, where d_x denotes the degree of the polynomial at node x. During decryption, the user's private key attributes, denoted as inputs to these polynomials, allow for the reconstruction of plaintext M through calculated interpolation. With L representing the set of leaf nodes in $\mathcal{T}$, the ciphertext CT is calculated as Equation 6.2.

$$\begin{aligned} CT =& \mathcal{T}, \bar{C} = M \times e(g,g)^{\alpha \times s}, C = h^s \\ & \forall l \in L : C_l = g^{q_l(0)}, C_l' = H(\text{att}(l))^{q_l(0)} \end{aligned} \tag{6.2}$$

6.4.5 Verification of Private Key Authenticity

The *Proof* algorithm, incorporating the Feige-Fiat-Shamir (FFS) protocol, uses a mathematical approach to confirm the existence of a private key SK without revealing its details. Specifically, it enables the prover P (the data requester) to authenticate their identity using SK before requesting encrypted data from the storage server, thereby optimizing bandwidth usage, as shown in Algorithm 11.

In the verification process, the prover P initiates by selecting a random number r within the range $(0, n)$, where n is a publicly known parameter about the validation proof VP. Next, using a function f based on the private key SK and the random integer r, P computes a specified value a. This computation ensures that data access requests can only be processed after verification. The formalization of this procedure aims to minimize unnecessary data transmission by granting access exclusively to requests that successfully pass this authentication phase, as shown in Equation 6.3.

$$Initialize(VP) = \{r, a | r \in (0,n), a \equiv r^2 \mod n\} \tag{6.3}$$

Further, the verifier, labelled as V (who may represent either the data owner or the entity overseeing storage services), constructs a sequence of variables p, derived from the maintained proof VP associated with the private key SK. V subsequently initiates a series of challenges, as shown in Equation 6.4.

$$\begin{aligned} Challenge(VP) &= \text{Generate(seed\_value, sequence\_length)} \\ &= \{p_i\}_{i=1}^{\text{sequence\_length}} \end{aligned} \tag{6.4}$$

Algorithm 11 Zero-Knowledge Proof Verification

Require: SK (private key), VP (validation proof)
Ensure: Boolean value indicating proof validity
1: Prover P generates a random variable $r \in (0, n)$
2: Prover P computes $a = f(r, SK)$
3: Initialize verification sequence e by the Verifier V
4: V issues a challenge based on VP
5: **for** $i = 1$ to t **do**
6: Generate $e_i \in \{0, 1\}$ randomly
7: **end for**
8: Prover P responds with $ANS = r \cdot \prod_{i=1}^{t} y_i^{e_i} \mod n$
9: Verifier V checks the response ANS
10: Compute $\lambda = \prod_{i=1}^{t} VP.x_i^{e_i} \mod n$
11: **if** $ANS^2 \equiv a \cdot \lambda \mod n$ **then**
12: **return** True
13: **else**
14: **return** False
15: **end if**

Upon receipt of the challenge, the prover P computes a response, denoted by ANS, using the private key. This response is then sent back within a single round of communication, as shown in Equation 6.5.

$$ANS = r \prod_{i=1}^{t} y_i^{e_i} \bmod n \tag{6.5}$$

The verifier, denoted by V, assesses the response ANS using the validation proof VP and the established public parameters, as shown in Equation 6.6.

$$\begin{aligned} Check(ANS, VP) &= \begin{cases} 1, ANS^2 \equiv a\lambda \bmod n \\ 0, ANS^2 \neq a\lambda \bmod n \end{cases} \\ \lambda &= \prod_{i=1}^{t} VP.x_i^{e_i} \bmod n \end{aligned} \tag{6.6}$$

The verifier V is able to issue *Challenge* multiple times to meet the set-up security threshold. If any single challenge is not met within a round, the entire *Proof* procedure will be promptly terminated.

6.4.6 Decryption of Ciphertext to Obtain Metadata

The decryption algorithm, denoted as *Dec*, is formulated as a recursive procedure. In the context of the encryption phase, the access policy is articulated through a tree structure.

Correspondingly, we instantiate a recursive algorithm, $DecNode(CT, SK, x)$, which commences its decryption process from the root and proceeds through successive layers of the tree, as shown in Algorithm 12.

Algorithm 12 Decryption Process

Require: CT (Ciphertext), SK (private Key), PK (Public Key)
Ensure: M_{meta} (Decrypted Metadata)

function DecNode(CT, SK, x)
  if x matches a leaf node **then**
    $i \leftarrow attribute(x)$
    return $\frac{e(D_i, C_x)}{e(D'_i, C'_x)}$ ▷ $D_i = g^r \cdot H(i)^{r_i}, C_x = h^{q_x(0)}, D'_i = g^{r_i}, C'_x = H(i)^{q_x(0)}$
  else
    Initialize $F_x \leftarrow 1$
    for each child node z of x **do**
      $F_z \leftarrow$ DecNode(CT, SK, z)
      Update F_x using F_z and Lagrange interpolation
    end for
    return F_x
  end if
end function
$\mathcal{T} \leftarrow$ Extract tree structure from CT
$\bar{C}, C \leftarrow$ Extract ciphertext components from CT
$R \leftarrow$ Root of tree $\mathcal{T}$
$M_{meta} \leftarrow$ DecNode(CT, SK, R)
return M_{meta}

The inputs to the algorithm are the ciphertext, the private key SK attached to a specific collection of attributes, and the node x in the tree $\mathcal{T}$. If x corresponds to a leaf node, where $i = \text{att}(x)$, the decryption is defined by a subsequent computation, as shown in equation 6.7.

$$\begin{aligned} DecNode(CT, SK, x) &= \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ &= \frac{e(g^r \times H(i)^{r_i}, h^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} \\ &= e(g, g)^{r \times q_x(0)} \end{aligned} \tag{6.7}$$

Consider a non-leaf node x. For each child node z of x, the function $DecNode$ is executed independently to determine F_z. Define S_x as the set containing all such children nodes z, where the cardinality of S_x is denoted by k_x. The condition $k_x > 0$ necessitates that $F_z \neq 1$. Conversely, if $k_x = 0$, it follows that $F_z = 1$. The formulation of F_x is represented as Equation 6.8.

$$F_x = \prod_{z \in S_x} F_z^{\Lambda(\mathrm{ind}(x), S'_x)}(0) = \tilde{e}(g,g)^{r\gamma q_x(0)}, \tag{6.8}$$

6.5 Security Analysis

This section delves into a comprehensive security analysis of the ZK-CP-ABE algorithm and the DPS-IIoT system, using mathematical proofs and formal modelling to assert their robustness against various cyber threats.

6.5.1 Data Confidentiality

Theorem 1: The ZK-CP-ABE algorithm ensures strict data confidentiality against polynomial-time adversaries.

Proof: Given a ciphertext C encrypted under a set of attributes A, let $\mathcal{A}$ be a polynomial-time adversary. The probability $\epsilon_{\mathcal{A}}$ that $\mathcal{A}$ decrypts C without possessing the correct set of attributes is negligible, as shown in Equation 6.9.

$$\epsilon_{\mathcal{A}} = \Pr[\mathcal{A} \text{ decrypts } C] \leq \frac{1}{2^{\mathrm{k}}} \tag{6.9}$$

6.5.2 Data Immutability

Theorem 2: DPS-IIoT system guarantees data immutability, preventing unauthorized data alteration.

Proof: Define D as a data block within DPS-IIoT, and $T(D)$ as a tampering operation. The probability P_{tamper} of successful tampering is shown as Equation 6.10.

$$P_{\text{tamper}}(D,T) = \sum_{i=1}^{n} \frac{e^{-\lambda_i \cdot t_i}}{1 + e^{\theta_i \cdot d(H(D,i), H(T(D,i)))}} \tag{6.10}$$

6.5.3 Robustness Against External Attacks

Theorem 3: The DPS-IIoT system demonstrates resilience against external cyber threats, including DDoS attacks and unauthorized data interception.

Proof: Let $\mathcal{N}$ represent the DPS-IIoT network. The resilience R_{external} against external attacks is modeled as Equation 6.11.

$$R_{\text{external}}(\mathcal{N}) = 1 - \exp(-\gamma \cdot \Phi(\mathcal{N})) \tag{6.11}$$

where γ represents the network defense parameter, and $\Phi(\mathcal{N})$ is the probability distribution of attack vectors.

6.5.4 Resistance to Internal Threats

Theorem 4: The ZK-CP-ABE algorithm is resistant to internal threats, including insider attacks and access policy manipulation.

Proof: Let $\mathcal{I}$ denote an internal adversary within the ZK-CP-ABE framework. The security against internal threats S_{internal} is quantified by Equation 6.12.

$$S_{\text{internal}}(\mathcal{I}) = 1 - \frac{\sum_{i=1}^{m} \rho_i \cdot \delta(H(\mathcal{I}, i), \Pi)}{m} \tag{6.12}$$

where ρ_i are the risk factors associated with internal roles, δ is the function measuring deviation between the adversary's access and the actual policy Π, and m is the total number of internal roles.

6.6 Experimental Results

This section presents a comprehensive comparative analysis of the introduced methods. Details on experimental settings are delineated in Section 6.6.1. The analysis evaluates the proposed algorithm and system across three key dimensions: the performance of the algorithm as assessed in Section 6.6.2, the scalability performance evaluation is presented in Section 6.6.3, and the system throughput is examined in Section 6.6.4.

6.6.1 Experimental Settings

In this manuscript, a comprehensive set of experimental configurations is carefully planned to demonstrate the system's ability to accommodate large volumes of traffic while minimizing bandwidth utilization. For experimental evaluation, the hardware configuration comprises two personal computers (PCs): the first equipped with a 3.60GHz i7-7700 CPU and 8GB of RAM, and the second furnished with an i7-7500U CPU at 2.90GHz coupled with 16GB of RAM. Additional apparatus includes a 1.5GHz Raspberry Pi with 4GB RAM, powered by an ARM A72 CPU.

6.6.2 Algorithm Performance Evaluation

The analysis undertakes a comparative assessment of the ZK-CP-ABE scheme against advanced CP-ABE structures, optimized for industrial applications. Key research efforts in this domain focus on improvements like minimizing private key sizes, symbolized as $\delta(SK)$ [165], enhancing policy and access control frameworks $\mathcal{F}_{AC}$ [166], and reducing the computational complexities Θ_{enc} and Θ_{dec} of encryption and decryption processes [167]. Moreover, the strategy incorporates a distributed key distribution mechanism $\mathcal{D}_{CA}$ by multiple Certificate Authorities (CAs) to enhance system robustness ρ.

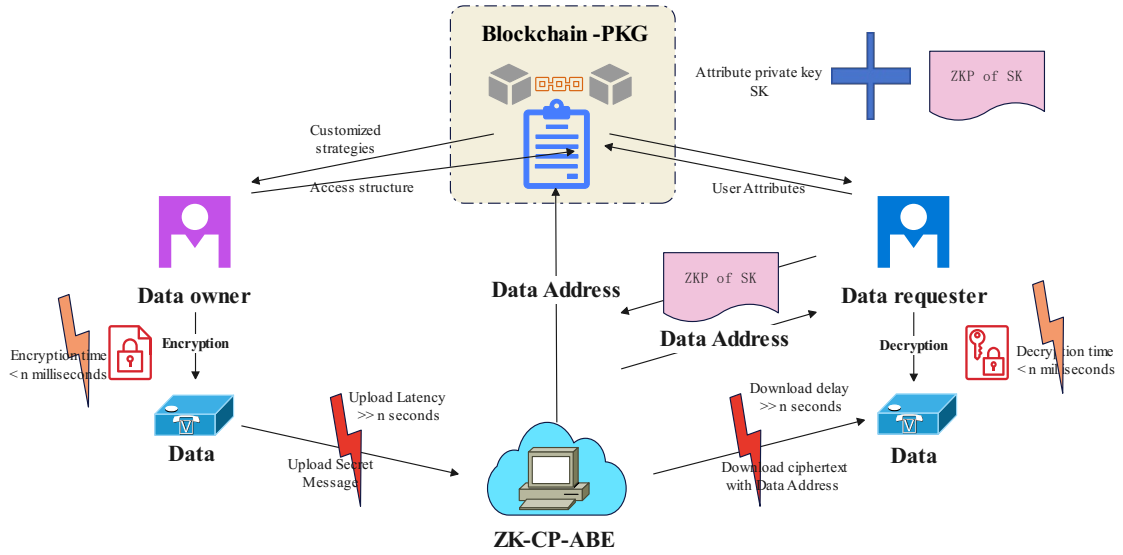


Figure 6.4: Design Principle of ZK-CP-ABE

With the IIoT's shift towards greater cloud integration and the growth of streaming media services, bandwidth demands on cloud infrastructures β_{cloud} have surged significantly. This increase in β_{cloud} is disproportionate to the gains in computational γ_{comp} and storage σ_{store} capacities. Addressing this imbalance requires more than just algorithmic refinement $\mathcal{R}_{alg}$. As shown in Figure 6.4, the employed methodology extends traditional CP-ABE enhancements by integrating a ZKP mechanism, denoted as $\mathcal{ZKP}_{enc-dec}$, within the encryption-decryption process. This approach discreetly verifies the existence of a user's private key SK, reducing bandwidth consumption $\Delta\beta_{unprod}$ from ineffective access control queries. Additionally, $\mathcal{ZKP}_{enc-dec}$ enables a shift towards a bandwidth-efficient symmetric encryption model in the CP-ABE framework, representing a significant leap in data security for IIoT systems. To quantify the bandwidth efficiency $\eta_{bandwidth}$ of the proposed algorithm, a comparative analysis was performed against the benchmarks in [165], [166], and [167], focusing on bandwidth utilization improvements $\Delta\eta_{bandwidth}$ and performance enhancement $\Delta\rho_{performance}$. The experimental framework is outlined as follows.

6.6.2.1 Computational Analysis Relative to Attribute Policies

For a set of attributes $\mathcal{A}$, with the cardinality $|\mathcal{A}| \in [10, 20, ..., 90, 100]$, we assess the computational overhead $C_{\text{overhead}}(f, \mathcal{A})$ for functions $f \in \{GenKey, Enc, Dec\}$. Our findings highlight a clear correlation, indicating that the computational overhead $C_{\text{overhead}}(f, \mathcal{A})$ is directly proportional to the cardinality $|\mathcal{A}|$. This underscores the significant impact of the number of attributes within the policy on the computational demands of the system.

6.6.2.2 Bandwidth Consumption Analysis for Varying Data Sizes

Consider M as a parameter denoting the size of plaintext data, which varies within a predetermined range. We examine the bandwidth requirement $B_{\text{alg}}(M)$ associated with each data magnitude M to gauge the adaptability of the ABE algorithm to fluctuations in data size. By scrutinizing the function $B_{\text{alg}}(M)$, we aim to gain quantitative insights into the scalability and efficiency of the algorithm, thereby informing its practical applicability across varying data sizes.

6.6.2.3 Analysis of Operational Duration to Transmission Latency

With the attribute count set at 50, we examine the operational duration $T_{\text{op}}(M)$ and transmission latency $T_{\text{trans}}(M)$ across a range of data sizes $M \in [2, 4, ..., 512, 1024]$ MB. This analysis allows us to compute the ratio $R_{\text{efficiency}}(M) = T_{\text{op}}(M)/T_{\text{trans}}(M)$, providing a quantitative measure of the equilibrium between processing and transmission efficiency. Such a metric serves as a crucial indicator for evaluating algorithm performance across various data sizes, facilitating informed decision-making regarding algorithm selection and optimization strategies.

6.6.3 Scalability performance evaluation

The advantages of our approach are evident from the following illustrations.

6.6.3.1 Algorithmic Complexity Analysis

Evaluating the functions $GenKey()$, $Enc()$, and $Dec()$ within our framework reveals a minimal operational time variance Δt_{op} compared to alternative algorithms. This variance is quantified as $\Delta t_{\text{op}}(f) \approx$ a few milliseconds (ms), for each function f in the set $\{GenKey, Enc, Dec\}$. For benchmarking, equivalent tests based on Ref [146,152,168] were deployed on a similarly configured PC, and parallelism tests were performed using identical clients and logic. This marginal discrepancy underscores the comparable efficiency of our framework's cryptographic operations as shown in Figure 6.5, Figure 6.6, and Figure 6.7.

6.6.3.2 Bandwidth Utilization Efficiency

Defined as the bandwidth utilization for processing plaintext of size M, $B_{\text{alg}}(M)$ represents a crucial metric in our evaluation. The proposed algorithm significantly optimizes bandwidth consumption, with the formula $B_{\text{alg}}(M) = B_{\text{ZKP}} + \mathcal{O}(1)$ demonstrating marked efficiency gains for larger M values. Here, B_{ZKP} denotes the consistent bandwidth expenditure associated with ZKP challenges, a characteristic that remains constant regardless of

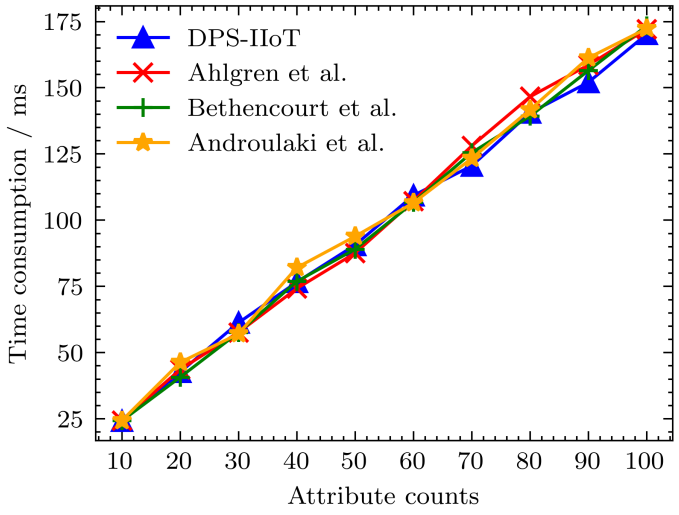


Figure 6.5: The performance of $GenKey()$

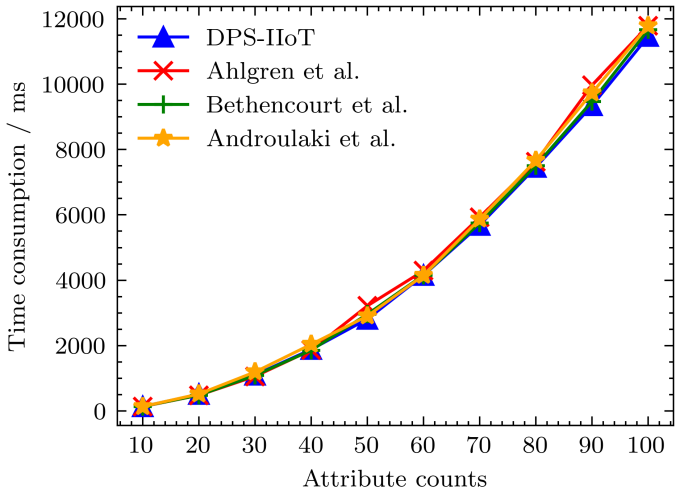


Figure 6.6: The performance of $Enc()$

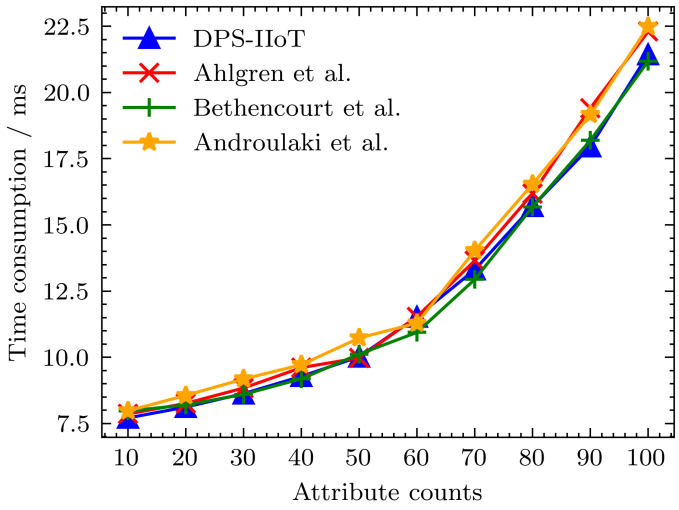


Figure 6.7: The performance of $Dec()$

plaintext size, as depicted in Figure 6.8. This observation highlights the algorithm's capacity to efficiently manage bandwidth resources, particularly evident in scenarios involving larger data sizes, thus contributing to enhanced system performance and scalability.

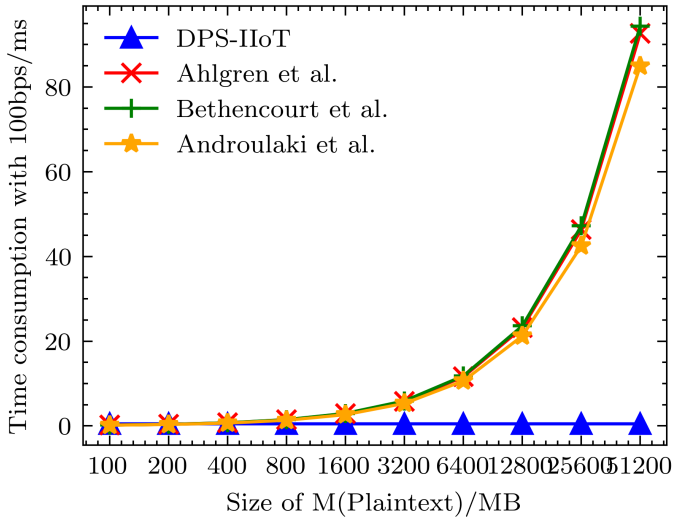


Figure 6.8: Bandwidth consumption of algorithm execution

6.6.3.3 Proportional Analysis of Time Consumption

As the size of the plaintext data M expands, there's a notable rise in the proportion of CP-ABE operational time relative to the transmission time, symbolized as $R_{CP-ABE}(M)$.

Particularly for larger data sizes, $R_{CP-ABE}(M)$ tends to reach close to 100%. Despite this increase, the time efficiency of our algorithm consistently maintains a stable and moderate pace, as shown in Figure 6.9. Thus, the ZK-CP-ABE algorithm presented in this study demonstrates significant optimization in bandwidth utilization. Its design and implementation align effectively with the requirements of IIoT environments, particularly in the era of burgeoning data volumes. This algorithm stands out as a robust solution for addressing the challenges of data management and security in the ever-expanding IIoT landscape.

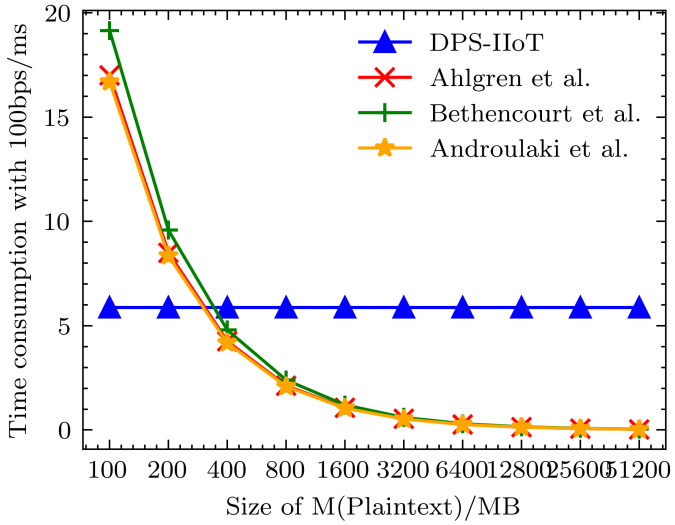


Figure 6.9: Proportional comparison of CP-ABE time cost to data transmission time

6.6.4 System Performance Evaluation

$PC1$ is deployed with several virtual machines, each equipped with a customized chain code (smart contract). To emulate a realistic IC-IoT environment, MQTT is adopted as the communication protocol, and the broker, EMQ, is deployed on $PC1$ within a Docker container. On $PC2$, subscriber and publisher clients are implemented to interact via the MQTT protocol, with each client possessing the capability to invoke the chain code within the blockchain. Additionally, a client is deployed on a Raspberry Pi to emulate a device, programmed to periodically upload data and maintain transmission in adherence to the MQTT protocol. To assess the system's throughput capabilities in a distributed context, experiments were conducted simulating various counts (denoted as N) of concurrent subscriber/publisher clients accessing the blockchain system over a predefined duration, aiming to evaluate the average response time (RT). Specifically, N is varied within the set $\{50, 100, \ldots, 500\}$. In the context of CP-ABE, the attribute count for testing is consistently

set at $K = 50$, and the data block size is fixed at $S = 512KB$. The resulting statistical data is shown in Figure 6.10.

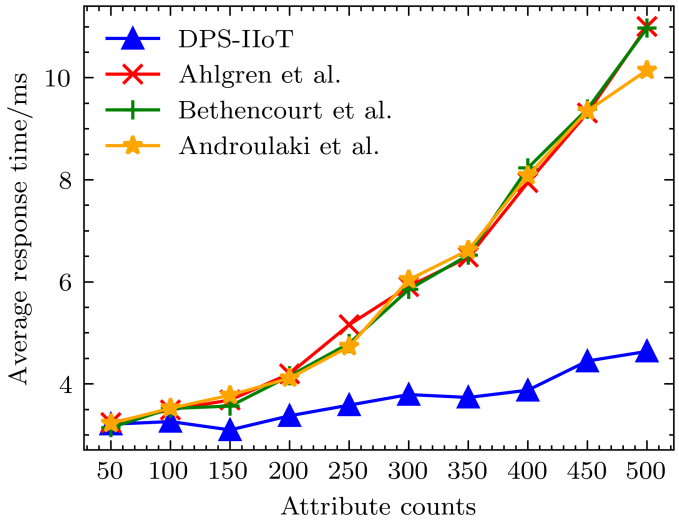


Figure 6.10: The mean response time under different levels of concurrent client loads.

6.7 Conclusion

IIoT is increasingly relying on external parties for data processing, storage, and other computing tasks, reflecting a growing trend towards integrating IIoT frameworks with cloud-based services. Meanwhile, the ICN model aims to improve network architecture for more efficient content access and distribution, particularly in managing disruptions and high-traffic scenarios. This chapter introduces the ZK-CP-ABE algorithm, specifically designed for the IIoT domain, alongside the DPS-IIoT system, which establishes a scalable and robust access control framework. The advanced ZK-CP-ABE algorithm incorporates the ZKP protocol, assigning secret keys based on user-specific attributes. Additionally, the system utilizes an efficient non-interactive ZKP, allowing user attributes to be verified seamlessly without direct communication. Experimental results and comparative analyses demonstrate that these methods significantly reduce network bandwidth usage while maintaining high throughput levels.

Chapter 7

Disscusion

Contents

7.1 Summary

Blockchain is widely used in finance, healthcare, logistics, and other sectors due to its decentralized architecture, record traceability, and data immutability. In the context of Industry 4.0, the rapid adoption of IIoT has significantly enhanced the intelligence, automation, and efficiency of the manufacturing and processing industries. However, as data volumes in the industrial sector increase dramatically, the related security threats are also becoming more severe. The application of blockchain technology in IIoT settings presents challenges, particularly in balancing operational performance with security, maintaining transparency without compromising privacy, and ensuring compatibility with a wide range of IIoT devices. This study addresses these challenges by improving existing blockchain-based IIoT frameworks with complementary technologies. Specifically, we propose secure storage mechanisms, optimized sorting algorithms, fine-grained access control, and access control systems based on non-interactive ZKPs. The contributions of this paper are summarised as follows:

- **Lightweight Data Storage Protocol**: This thesis proposes a secure data storage protocol based on blockchain for fast system initialisation through strong entity authentication and bilinear mapping. The protocol significantly reduces resource consumption while ensuring data security by employing off-chain data storage and blockchain-ensured encryption. A comprehensive evaluation showed the solution to be highly effective in reducing system overhead and enhancing storage reliability.

- **Trie-alternative transaction serial sorting scheme**: This thesis proposes Trie-Fabric, a sequential transaction ordering algorithm based on the Trie tree structure of a directed acyclic graph (DAG), for solving the high incidence of pending transactions in Hyperledger fabric. The study shows through a comparative study that Trie-Fabric performs well in managing conflicts, improving efficiency by more than 60% compared to the original Fabric system.

- **Smart Contract-inspired Access Control Scheme**: This thesis proposes an IIoT data access control framework based on smart contracts and Bloom filters called Hyper-IIoT. The framework extends the support for precise data governance and access control and achieves fine-grained access control through ABAC. Experimental results show that Hyper-IIoT exhibits good contract execution time, stable throughput and efficient consensus in a distributed environment.

- **Non-Interactive Zero-Knowledge Poof-inspired Access Control**: As IIoT increasingly tends to delegate data processing, storage, and other computational tasks

to external parties, this thesis proposes the DPS-IIoT system, which aims to build a scalable and robust access control framework. The system employs the ZK-CP-ABE algorithm and the NIZKP protocol, which effectively reduces the network bandwidth usage while maintaining a robust throughput level.

7.2 Future Work

This section outlines the perspective of some future works to expand the research content of this thesis. This paper has focused on exploring the efficiency and security of data processing in IIoT environments and has achieved some results in addressing system security and data protection in IIoT. However, given the evolving technological landscape and the rapid complexity inherent in the IIoT ecosystem, there exists a compelling impetus to delve into many emerging areas to chart a course for future research enquiries and explorations. The following areas will be the focus of future research as relevant technologies continue to evolve:

- **Enhancing the scalability of the blockchain**: Develop more flexible data storage solutions to adapt to changes in IIoT data structure and format, ensuring optimal data processing efficiency regardless of the dynamics of IIoT data, thus maintaining a high level of system performance.

- **Improving conflict detection capabilities**: Explore using advanced machine learning algorithms for smarter transaction processing and proactive conflict resolution.

- **Building adaptive sorting frameworks**: Construct a transaction reordering framework that can dynamically adapt to changing network states, keeping system throughput optimised by responding to changes in network traffic and transaction volume.

- **Improving the consensus mechanism**: Subsequent research can consider starting from improving the underlying architecture of the blockchain to fundamentally solve the efficiency, compatibility and practicality problems that still exist in this paper and existing research. Accelerate the consensus process and expand the application scope of Hyperledger fabric to reduce latency and bandwidth consumption to meet the real-time processing needs of large-scale IIoT applications.

- **Developing a stable and open source large-scale blockchain-based IIoT data management platform**: Currently, numerous blockchain projects operate

across various platforms, algorithms, and consensus mechanisms, leading to inefficiencies and compatibility challenges that hinder the creation of cohesive market synergy. Future researchers can consider developing a more stable and unified large-scale platform to support the landing of a larger range of applications.

- **Producing large-scale datasets based on real application scenarios of IIoT**: Currently, the lack of large-scale real application endorsement for blockchain research is a common problem faced by this paper and other related research. Therefore, with the maturity of technology, policy, market and other conditions, subsequent researchers need to gradually apply blockchain technology to real large-scale or even super-large-scale application scenarios, collect the results, and continuously improve and innovate to complete the real application landing.

- **Establishing a comprehensive blockchain security verification standard and framework**: The security research of blockchain technology has not yet formed a complete system, and the subsequent research can consider establishing a unified standard for the security verification of blockchain, to systematically carry out systematic testing of blockchain-based system proposals and prototypes.

References

[1] P. Singh, M. Masud, M. S. Hossain, and A. Kaur, "Cross-domain secure data sharing using blockchain for industrial iot," *Journal of Parallel and Distributed Computing*, vol. 156, pp. 176–184, 2021.

[2] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "A comprehensive survey on interoperability for iiot: Taxonomy, standards, and future directions," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–35, 2021.

[3] W. Kassab and K. A. Darabkh, "A–z survey of internet of things: Architectures, protocols, applications, recent advances, future directions and recommendations," *Journal of Network and Computer Applications*, vol. 163, p. 102663, 2020.

[4] R. Chen, J. Guo, and F. Bao, "Trust management for soa-based iot and its application to service composition," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 482–495, 2014.

[5] M. Laroui, B. Nour, H. Moungla, M. A. Cherif, H. Afifi, and M. Guizani, "Edge and fog computing for iot: A survey on current research activities & future directions," *Computer Communications*, vol. 180, pp. 210–231, 2021.

[6] A. Djama, B. Djamaa, and M. R. Senouci, "Information-centric networking solutions for the internet of things: A systematic mapping review," *Computer Communications*, vol. 159, pp. 37–59, 2020.

[7] S. Mishra, V. K. Jain, K. Gyoda, and S. Jain, "An efficient content replacement policy to retain essential content in information-centric networking based internet of things network," *Ad Hoc Networks*, vol. 155, p. 103389, 2024.

[8] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Recent advances in information-centric networking-based internet of things (icn-iot)," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2128–2158, 2018.

[9] N. Naik and P. Jenkins, "Web protocols and challenges of web latency in the web of things," in *2016 Eighth International Conference on ubiquitous and future networks (ICUFN)*. IEEE, 2016, pp. 845–850.

[10] N. Q. Uy and V. H. Nam, "A comparison of amqp and mqtt protocols for internet of things," in *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*. IEEE, 2019, pp. 292–297.

[11] S. Bandyopadhyay and A. Bhattacharyya, "Lightweight internet protocols for web enablement of sensors using constrained gateway devices," in *2013 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2013, pp. 334–340.

[12] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali, "Comparison of two lightweight protocols for smartphone-based sensing," in *2013 IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*. IEEE, 2013, pp. 1–6.

[13] N. Bahri, S. Saadaoui, M. Tabaa, M. Sadik, and H. Medromi, "Wireless technologies and applications for industrial internet of things: A review," *Advances on Smart and Soft Computing: Proceedings of ICACIn 2020*, pp. 505–516, 2021.

[14] C. Bormann, A. P. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[15] N. Naik, "Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http," in *2017 IEEE international systems engineering symposium (ISSE)*. IEEE, 2017, pp. 1–7.

[16] S. Vinoski, "Advanced message queuing protocol," *IEEE Internet Computing*, vol. 10, no. 6, pp. 87–89, 2006.

[17] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[18] M. Swan, *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", 2015.

[19] S. Singh and N. Singh, "Blockchain: Future of financial and cyber security," in *2016 2nd international conference on contemporary computing and informatics (IC3I)*. IEEE, 2016, pp. 463–467.

[20] N. O. Nawari and S. Ravindran, "Blockchain technology and bim process: review and potential applications." *J. Inf. Technol. Constr.*, vol. 24, no. 12, pp. 209–238, 2019.

[21] I. Homoliak, S. Venugopalan, D. Reijsbergen, Q. Hum, R. Schumi, and P. Szalachowski, "The security reference architecture for blockchains: Toward a standardized model for studying vulnerabilities, threats, and defenses," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 341–390, 2020.

[22] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 2019.

[23] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[24] M. Steichen, B. Fiz, R. Norvill, W. Shbair, and R. State, "Blockchain-based, decentralized access control for ipfs," in *2018 Ieee international conference on internet of things (iThings) and ieee green computing and communications (GreenCom) and ieee cyber, physical and social computing (CPSCom) and ieee smart data (SmartData)*. IEEE, 2018, pp. 1499–1506.

[25] U. Bodkhe, D. Mehta, S. Tanwar, P. Bhattacharya, P. K. Singh, and W.-C. Hong, "A survey on decentralized consensus mechanisms for cyber physical systems," *IEEE Access*, vol. 8, pp. 54 371–54 401, 2020.

[26] L. Wang, X. Shen, J. Li, J. Shao, and Y. Yang, "Cryptographic primitives in blockchains," *Journal of Network and Computer Applications*, vol. 127, pp. 43–58, 2019.

[27] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Ieee, 2018, pp. 1545–1550.

[28] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE transactions on knowledge and data engineering*, vol. 30, no. 7, pp. 1366–1385, 2018.

[29] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, pp. 2–1, 2014.

[30] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 839–858.

[31] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.

[32] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2084–2106, 2019.

[33] M. Belotti, N. Božić, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3796–3838, 2019.

[34] X. Li, J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah, and S. Kumari, "A robust ecc-based provable secure authentication protocol with privacy preserving for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3599–3609, 2017.

[35] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the theory and application of cryptographic techniques*. Springer, 1987, pp. 369–378.

[36] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE network*, vol. 35, no. 4, pp. 198–205, 2021.

[37] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial internet of things: A cyber-physical systems perspective," *Ieee access*, vol. 6, pp. 78 238–78 259, 2018.

[38] K. K. Patel, S. M. Patel, and P. Scholar, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 2016.

[39] W. Liang, Y. Yang, C. Yang, Y. Hu, S. Xie, K.-C. Li, and J. Cao, "Pdpchain: A consortium blockchain-based privacy protection scheme for personal data," *IEEE Transactions on Reliability*, 2022.

[40] A. E. Omolara, A. Alabdulatif, O. I. Abiodun, M. Alawida, A. Alabdulatif, H. Arshad *et al.*, "The internet of things security: A survey encompassing unexplored areas and new insights," *Computers & Security*, vol. 112, p. 102494, 2022.

[41] W. Shao, Y. Wei, P. Rajapaksha, D. Li, Z. Luo, and N. Crespi, "Low-latency dimensional expansion and anomaly detection empowered secure iot network," *IEEE Transactions on Network and Service Management*, no. 99, pp. 1–1, 2023.

[42] C. Diao, D. Zhang, W. Liang, K.-C. Li, Y. Hong, and J.-L. Gaudiot, "A novel spatial-temporal multi-scale alignment graph neural network security model for vehicles prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 904–914, 2022.

[43] J. Cai, W. Liang, X. Li, K. Li, Z. Gui, and M. K. Khan, "Gtxchain: A secure iot smart blockchain architecture based on graph neural network," *IEEE Internet of Things Journal*, 2023.

[44] J. Long, W. Liang, K.-C. Li, Y. Wei, and M. D. Marino, "A regularized cross-layer ladder network for intrusion detection in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1747–1755, 2022.

[45] S. Munirathinam, "Industry 4.0: Industrial internet of things (iiot)," in *Advances in computers*. Elsevier, 2020, vol. 117, no. 1, pp. 129–164.

[46] D. Li, D. Han, Z. Zheng, T.-H. Weng, H. Li, H. Liu, A. Castiglione, and K.-C. Li, "Moocschain: A blockchain-based secure storage and sharing scheme for moocs learning," *Computer Standards & Interfaces*, vol. 81, p. 103597, 2022.

[47] H. Li, D. Han, and C.-C. Chang, "Dac4sh: A novel data access control scheme for smart home using smart contracts," *IEEE Sensors Journal*, vol. 23, no. 6, pp. 6178–6191, 2023.

[48] N. Hu, D. Zhang, K. Xie, W. Liang, K. Li, and A. Zomaya, "Multi-graph fusion based graph convolutional networks for traffic prediction," *Computer Communications*, vol. 210, pp. 194–204, 2023.

[49] S. Zhang, B. Hu, W. Liang, K.-C. Li, and A.-S. K. Pathan, "A trajectory privacy-preserving scheme based on transition matrix and caching for iiot," *IEEE Internet of Things Journal*, 2023.

[50] H. Li, D. Han, and M. Tang, "A privacy-preserving storage scheme for logistics data with assistance of blockchain," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4704–4720, 2021.

[51] D. Li, D. Han, B. Xia, T.-H. Weng, A. Castiglione, and K.-C. Li, "Fabric-gc: A blockchain-based gantt chart system for cross-organizational project management," *Computer Science and Information Systems*, vol. 19, no. 3, pp. 1213–1240.

[52] Y. Wu, H.-N. Dai, and H. Wang, "Convergence of blockchain and edge computing for secure and scalable iiot critical infrastructures in industry 4.0," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2300–2317, 2020.

[53] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale internet of things data storage and protection," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 762–771, 2018.

[54] G. Wang, Z. Shi, M. Nixon, and S. Han, "Chainsplitter: Towards blockchain-based industrial iot architecture for supporting hierarchical storage," in *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019, pp. 166–175.

[55] L. Tan, K. Yu, C. Yang, and A. K. Bashir, "A blockchain-based shamir's threshold cryptography for data protection in industrial internet of things of smart city," in *Proceedings of the 1st Workshop on Artificial Intelligence and Blockchain Technologies for Smart Cities with 6G*, 2021, pp. 13–18.

[56] S.-C. Cha, J.-F. Chen, C. Su, and K.-H. Yeh, "A blockchain connected gateway for ble-based devices in the internet of things," *ieee access*, vol. 6, pp. 24 639–24 649, 2018.

[57] S. Qi, Y. Lu, Y. Zheng, Y. Li, and X. Chen, "Cpds: Enabling compressed and private data sharing for industrial internet of things over blockchain," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2376–2387, 2020.

[58] J. Huang, L. Kong, G. Chen, M.-Y. Wu, X. Liu, and P. Zeng, "Towards secure industrial iot: Blockchain system with credit-based consensus mechanism," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3680–3689, 2019.

[59] Y. Liu, K. Wang, Y. Lin, and W. Xu, "Lightchain: a lightweight blockchain system for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3571–3581, 2019.

[60] M. Zhaofeng, W. Xiaochang, D. K. Jain, H. Khan, G. Hongmin, and W. Zhen, "A blockchain-based trusted data management scheme in edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2013–2021, 2019.

[61] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE internet of things journal*, vol. 6, no. 3, pp. 4660–4670, 2018.

[62] T. Wu, G. Jourjon, K. Thilakarathna, and P. L. Yeoh, "Mapchain-d: A distributed blockchain for iiot data storage and communications," *IEEE Transactions on Industrial Informatics*, 2023.

[63] Y. Liu, J. Liu, J. Yin, G. Li, H. Yu, and Q. Wu, "Cross-shard transaction processing in sharding blockchains," in *Algorithms and Architectures for Parallel Processing: 20th International Conference, ICA3PP 2020, New York City, NY, USA, October 2–4, 2020, Proceedings, Part III 20*. Springer, 2020, pp. 324–339.

[64] Y. Zhang, B. Li, J. Wu, B. Liu, R. Chen, and J. Chang, "Efficient and privacy-preserving blockchain-based multifactor device authentication protocol for cross-domain iiot," *IEEE Internet of Things Journal*, vol. 9, no. 22, pp. 22 501–22 515, 2022.

[65] G. Rathee, C. A. Kerrache, and M. Lahby, "Trustblksys: A trusted and blockchained cybersecure system for iiot," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1592–1599, 2022.

[66] J. Cui, N. Liu, Q. Zhang, D. He, C. Gu, and H. Zhong, "Efficient and anonymous cross-domain authentication for iiot based on blockchain," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 2, pp. 899–910, 2022.

[67] C. Xu, K. Wang, P. Li, S. Guo, J. Luo, B. Ye, and M. Guo, "Making big data open in edges: A resource-efficient blockchain-based approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 4, pp. 870–882, 2018.

[68] J. Wei, Q. Zhu, Q. Li, L. Nie, Z. Shen, K.-K. R. Choo, and K. Yu, "A redactable blockchain framework for secure federated learning in industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17 901–17 911, 2022.

[69] P. Kumar, R. Kumar, A. Kumar, A. A. Franklin, S. Garg, and S. Singh, "Blockchain and deep learning for secure communication in digital twin empowered industrial iot network," *IEEE Transactions on Network Science and Engineering*, 2022.

[70] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:236214795

[71] M. K. Hasan, M. Akhtaruzzaman, S. R. Kabir, T. R. Gadekallu, S. Islam, P. Magalingam, R. Hassan, M. Alazab, and M. A. Alazab, "Evolution of industry and blockchain era: monitoring price hike and corruption using biot for smart government and industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 9153–9161, 2022.

[72] D. Li, D. Han, and H. Liu, "Fabric-chain & chain: a blockchain-based electronic document system for supply chain finance," in *Blockchain and Trustworthy Systems: Second International Conference, BlockSys 2020, Dali, China, August 6–7, 2020, Revised Selected Papers 2*. Springer, 2020, pp. 601–608.

[73] D. Li, D. Han, N. Crespi, R. Minerva, and K.-C. Li, "A blockchain-based secure storage and access control scheme for supply chain finance," *The Journal of Supercomputing*, vol. 79, no. 1, pp. 109–138, 2023.

[74] A. Raja Santhi and P. Muthuswamy, "Influence of blockchain technology in manufacturing supply chain and logistics," *Logistics*, vol. 6, no. 1, p. 15, 2022.

[75] Z. Sun, D. Han, D. Li, X. Wang, C.-C. Chang, and Z. Wu, "A blockchain-based secure storage scheme for medical information," *EURASIP Journal on Wireless Communications and Networking*, vol. 2022, no. 1, p. 40, 2022.

[76] Z. Sun, D. Han, D. Li, T.-H. Weng, K. Li, and X. Mei, "Medrss: A blockchain-based scheme for secure storage and sharing of medical records," *Comput. Ind. Eng.*, vol. 183, p. 109521, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:261042226

[77] R. Malik, H. Raza, M. Saleem *et al.*, "Towards a blockchain enabled integrated library managment system using hyperledger fabric: Using hyperledger fabric," *International Journal of Computational and Innovative Sciences*, vol. 1, no. 3, pp. 17–24, 2022.

[78] J. Cai, W. Liang, X. Li, Z. Gui *et al.*, "Gtxchain: A secure iot smart blockchain architecture based on gnn," *IEEE Internet of Things Journal*.

[79] C. Diao, D. Zhang, W. Liang *et al.*, "Spatial-temporal multi-scale alignment graph neural network security model for vehicles prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 904–914, 2023.

[80] F. Wang, J. Cui, Q. Zhang, D. He, and H. Zhong, "Blockchain-based secure cross-domain data sharing for edge-assisted industrial internet of things," *IEEE Transactions on Information Forensics and Security*, 2024.

[81] M. Uddin, M. Memon, I. Memon, I. Ali, J. Memon, M. Abdelhaq, and R. Alsaqour, "Hyperledger fabric blockchain: Secure and efficient solution for electronic health records," *Comput. Mater. Contin*, vol. 68, no. 2, pp. 2377–2397, 2021.

[82] D. Li, D. Han, T.-H. Weng, Z. Zheng, H. Li, H. Liu, A. Castiglione, and K.-C. Li, "Blockchain for federated learning toward secure distributed machine learning systems: a systemic survey," *Soft Computing*, vol. 26, no. 9, pp. 4423–4440, 2022.

[83] A. Sharma, F. M. Schuhknecht, D. Agrawal, and J. Dittrich, "Blurring the lines between blockchains and database systems: the case of hyperledger fabric," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 105–122.

[84] P. Ruan, D. Loghin, Q.-T. Ta, M. Zhang, G. Chen, and B. C. Ooi, "A transactional perspective on execute-order-validate blockchains," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 543–557.

[85] J. Ni, J. Xiao, S. Zhang, B. Li, B. Li, and H. Jin, "Fluid: Towards efficient continuous transaction processing in dag-based blockchains," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[86] R. Xu and Y. Chen, "μdfl: A secure microchained decentralized federated learning fabric atop iot networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2677–2688, 2022.

[87] J. Xiao, S. Zhang, Z. Zhang, B. Li, X. Dai, and H. Jin, "Nezha: Exploiting concurrency for transaction processing in dag-based blockchains," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 269–279.

[88] (2022) Hyperledger. caliper. [Online]. Available: https://github.com/hyperledger/caliper-benchmarks

[89] J. C. Anderson, J. Lehnardt, and N. Slater, *CouchDB: the definitive guide: time to relax.* " O'Reilly Media, Inc.", 2010.

[90] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "Solo: Segmenting objects by locations," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer, 2020, pp. 649–665.

[91] J. A. Chacko, R. Mayer, and H.-A. Jacobsen, "Why do my blockchain transactions fail? a study of hyperledger fabric," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 221–234.

[92] K. Tsuruta, D. Köppl, S. Kanda, Y. Nakashima, S. Inenaga, H. Bannai, and M. Takeda, "c-trie++: A dynamic trie tailored for fast prefix searches," *Information and Computation*, vol. 285, p. 104794, 2022.

[93] B. Finkbeiner, C. Hahn, M. Stenger, and L. Tentrup, "Efficient monitoring of hyperproperties using prefix trees," *International Journal on Software Tools for Technology Transfer*, vol. 22, no. 6, pp. 729–740, 2020.

[94] B. Wang, M. Dabbaghjamanesh, A. Kavousi-Fard, and S. Mehraeen, "Cybersecurity enhancement of power trading within the networked microgrids based on blockchain and directed acyclic graph approach," *IEEE Transactions on Industry Applications*, vol. 55, no. 6, pp. 7300–7309, 2019.

[95] M. Bhandary, M. Parmar, and D. Ambawade, "A blockchain solution based on directed acyclic graph for iot data security using iota tangle," in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*. IEEE, 2020, pp. 827–832.

[96] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[97] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[98] W. Liang, Y. Yang, C. Yang, Y. Hu *et al.*, "Pdpchain: A consortium blockchain-based privacy protection scheme for personal data," *IEEE Transactions on Reliability*, vol. 72, no. 2, pp. 586–598, 2023.

[99] P. Chen, D. Han, T.-H. Weng *et al.*, "A novel byzantine fault tolerance consensus for green iot with intelligence based on reinforcement," *Journal of Information Security and Applications*, vol. 59, p. 102821, 2021.

[100] W. Li, C. Feng, L. Zhang, H. Xu, B. Cao, and M. A. Imran, "A scalable multi-layer pbft consensus for blockchain," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1146–1160, 2020.

[101] Y. Liu, J. Liu, Q. Wu, H. Yu, Y. Hei, and Z. Zhou, "Sshc: A secure and scalable hybrid consensus protocol for sharding blockchains with a formal security framework," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 2070–2088, 2020.

[102] Z.-F. Wang, S.-Q. Liu, P. Wang, and L.-Y. Zhang, "Bw-pbft: Practical byzantine fault tolerance consensus algorithm based on credit bidirectionally waning," 2023.

[103] P. Nasirifard, R. Mayer, and H.-A. Jacobsen, "Fabriccrdt: A conflict-free replicated datatypes approach to permissioned blockchains," in *Proceedings of the 20th International Middleware Conference*, 2019, pp. 110–122.

[104] K. Narayanam, A. Kaul, K. Kumar, and P. Dayama, "Partial order transactions on permissioned blockchains for enhanced scalability," in *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE, 2021, pp. 149–156.

[105] N. Berendea, H. Mercier, E. Onica, and E. Riviere, "Fair and efficient gossip in hyperledger fabric," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 190–200.

[106] B. Pi, Y. Pan, E. Zhou, J. Sun, T. Miyamae, and M. Morinaga, "xfabledger: Extensible ledger storage for hyperledger fabric," in *2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC) 2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC)*. IEEE, 2021, pp. 5–11.

[107] Y. Li, H. Liu, Y. Chen, J. Gao, Z. Wu, Z. Guan, and Z. Chen, "Fastblock: Accelerating blockchains via hardware transactional memory," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 250–260.

[108] 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)*. IEEE, 2018, pp. 264–276.

[109] B. Ding, L. Kot, and J. Gehrke, "Improving optimistic concurrency control through transaction batching and operation reordering," *Proceedings of the VLDB Endowment*, vol. 12, no. 2, pp. 169–182, 2018.

[110] (2019) Hyperledger. caliper-benchmarks. [Online]. Available: https://github.com/hyperledger/caliper-benchmarks/blob/main/src/fabric/scenario/smallbank/go/smallbank.go

[111] Y. Long, C. Peng, W. Tan, and Y. Chen, "Blockchain-based anonymous authentication and key management for internet of things with chebyshev chaotic maps," *IEEE Transactions on Industrial Informatics*, 2024.

[112] A. Mahmood, L. Beltramelli, S. F. Abedin, S. Zeb, N. I. Mowla, S. A. Hassan, E. Sisinni, and M. Gidlund, "Industrial iot in 5g-and-beyond networks: Vision, architecture, and design trends," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4122–4137, 2021.

[113] S. Zhang, B. Hu, W. Liang, K.-C. Li, and B. B. Gupta, "A caching-based dual k-anonymous location privacy-preserving scheme for edge computing," *IEEE Internet of Things Journal*, 2023.

[114] S. Zhang, Z. Yan, W. Liang, K.-C. Li, and C. Dobre, "Baka: Biometric authentication and key agreement scheme based on fuzzy extractor for wireless body area networks," *IEEE Internet of Things Journal*, 2023.

[115] L. Da Xu, Y. Lu, and L. Li, "Embedding blockchain technology into iot for security: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 452–10 473, 2021.

[116] R. Verma, N. Dhanda, and V. Nagar, "Towards a secured iot communication: A blockchain implementation through apis," in *Proceedings of Third International Conference on Computing, Communications, and Cyber-Security: IC4S 2021*. Springer, 2022, pp. 681–692.

[117] H. Lee, "Effective dynamic control strategy of a key supplier with multiple downstream manufacturers using industrial internet of things and cloud system," *Processes*, vol. 7, no. 3, p. 172, 2019.

[118] B. Pourghebleh, K. Wakil, and N. J. Navimipour, "A comprehensive study on the trust management techniques in the internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9326–9337, 2019.

[119] W. Liang, Y. Li, K. Xie, D. Zhang, K.-C. Li, A. Souri, and K. Li, "Spatial-temporal aware inductive graph neural network for c-its data recovery," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[120] B. Chen, S. Qiao, J. Zhao, D. Liu, X. Shi, M. Lyu, H. Chen, H. Lu, and Y. Zhai, "A security awareness and protection system for 5g smart healthcare based on zero-trust architecture," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 248–10 263, 2020.

[121] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, and M. Song, "Performance optimization for blockchain-enabled industrial internet of things (iiot) systems: A deep reinforcement learning approach," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3559–3570, 2019.

[122] N. Hu, D. Zhang, K. Xie, W. Liang, C. Diao, and K.-C. Li, "Multi-range bidirectional mask graph convolution based gru networks for traffic prediction," *Journal of Systems Architecture*, vol. 133, p. 102775, 2022.

[123] J. Li, D. Han, Z. Wu, J. Wang, K.-C. Li, and A. Castiglione, "A novel system for medical equipment supply chain traceability based on alliance chain and attribute and role access control," *Future Generation Computer Systems*, vol. 142, pp. 195–211, May 2023.

[124] J. Wan, J. Li, M. Imran, D. Li *et al.*, "A blockchain-based solution for enhancing security and privacy in smart factory," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3652–3660, 2019.

[125] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2018.

[126] B. K. Mohanta, S. S. Panda, and D. Jena, "An overview of smart contract and use cases in blockchain technology," in *2018 9th international conference on computing, communication and networking technologies (ICCCNT)*. IEEE, 2018, pp. 1–4.

[127] M. M. Merlec and H. P. In, "Sc-caac: A smart contract-based context-aware access control scheme for blockchain-enabled iot systems," *IEEE Internet of Things Journal*, 2024.

[128] S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart contract: Attacks and protections," *IEEE Access*, vol. 8, pp. 24 416–24 427, 2020.

[129] R. Saha, G. Kumar, M. Conti, T. Devgun, T.-h. Kim, M. Alazab, and R. Thomas, "Dhacs: Smart contract-based decentralized hybrid access control for industrial internet-of-things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3452–3461, 2021.

[130] Y. Wei, K. Gai, J. Yu, L. Zhu, and K.-K. R. Choo, "Trustworthy access control for multiaccess edge computing in blockchain-assisted 6g systems," *IEEE Transactions on Industrial Informatics*, 2024.

[131] W. Wang, H. Huang, Z. Yin, T. R. Gadekallu, M. Alazab, and C. Su, "Smart contract token-based privacy-preserving access control system for industrial internet of things," *Digital Communications and Networks*, vol. 9, no. 2, pp. 337–346, 2023.

[132] H. Li and D. Han, "A novel time-aware hybrid recommendation scheme combining user feedback and collaborative filtering," *Mobile Information Systems*, vol. 2020, pp. 1–16, 2020.

[133] W. Liang, M. Tang, J. Long, X. Peng, J. Xu, and K.-C. Li, "A secure fabric blockchain-based data transmission technique for industrial internet-of-things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3582–3592, 2019.

[134] S. Zhou, K. Li, L. Xiao, J. Cai, W. Liang, and A. Castiglione, "A systematic review of consensus mechanisms in blockchain," *Mathematics*, vol. 11, no. 10, p. 2248, 2023.

[135] N. Yang and C. Tang, "Blockchain-assisted secure data sharing protocol with a dynamic multi-user keyword search in iiot," *IEEE Internet of Things Journal*, 2023.

[136] 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.

[137] M. Ma, G. Shi, and F. Li, "Privacy-oriented blockchain-based distributed key management architecture for hierarchical access control in the iot scenario," *IEEE access*, vol. 7, pp. 34 045–34 059, 2019.

[138] L. Waikhom, S. Nayak, and R. Patgiri, "A survey on bloom filter for multiple sets," in *Modeling, Simulation and Optimization: Proceedings of CoMSO 2020*. Springer, 2021, pp. 775–789.

[139] B. Nour, K. Sharif, F. Li, S. Biswas, H. Moungla, M. Guizani, and Y. Wang, "A survey of internet of things communication using icn: A use case perspective," *Computer Communications*, vol. 142-143, pp. 95–123, 2019.

[140] C. Sandeepa, B. Siniarski, N. Kourtellis, S. Wang, and M. Liyanage, "A survey on privacy for b5g/6g: New privacy challenges, and research directions," *Journal of Industrial Information Integration*, vol. 30, p. 100405, 2022.

[141] Y. Lu, D. Wang, M. S. Obaidat, and P. Vijayakumar, "Edge-assisted intelligent device authentication in cyber–physical systems," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3057–3070, 2022.

[142] G. Tian, Y. Hu, J. Wei, Z. Liu, X. Huang, X. Chen, and W. Susilo, "Blockchain-based secure deduplication and shared auditing in decentralized storage," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 3941–3954, 2021.

[143] H. Li, D. Han, and M. Tang, "A privacy-preserving storage scheme for logistics data with assistance of blockchain," *IEEE Internet of Things Journal*, 2021.

[144] H. Liu, D. Han, and D. Li, "Behavior analysis and blockchain based trust management in vanets," *J. Parallel Distributed Comput.*, vol. 151, pp. 61–69, 2021.

[145] L. Feng, J. Lin, F. Qiu, B. Yu, Z. Jin, J. Wang, J. Cheng, and S. Yao, "Sdac-bbpp: A secure dynamic access control scheme with blockchain-based privacy protection privacy for iiot," *IEEE Transactions on Network and Service Management*, 2024.

[146] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, 2012.

[147] A. Sahai and B. Waters, "Fuzzy identity-based encryption," *IACR Cryptol. ePrint Arch.*, vol. 2004, p. 86, 2005.

[148] S. Khezr, A. Yassine, R. Benlamri, and M. S. Hossain, "An edge intelligent blockchain-based reputation system for iiot data ecosystem," *IEEE transactions on industrial informatics*, vol. 18, no. 11, pp. 8346–8355, 2022.

[149] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *STOC '85*, 1985.

[150] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *STOC '88*, 1988.

[151] C. P. Sah, "Robustness of zero-knowledge proofs using rsa problem," in *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2022, pp. 40–44.

[152] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. Cocco, and J. Yellick, "Hyperledger fabric: a distributed operating system for permissioned blockchains," *Proceedings of the Thirteenth EuroSys Conference*, 2018.

[153] R. Li, H. Asaeda, and J. Li, "A distributed publisher-driven secure data sharing scheme for information-centric iot," *IEEE Internet of Things Journal*, vol. 4, pp. 791–803, 2017.

[154] R. Li, H. Asaeda, J. Li, and X. Fu, "A verifiable and flexible data sharing mechanism for information-centric iot," *2017 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2017.

[155] Y. Zhao, M. Ren, S. Jiang, G. Zhu, and H. Xiong, "An efficient and revocable storage cp-abe scheme in the cloud computing," *Computing*, vol. 101, pp. 1041–1065, 2018.

[156] L. Touati and Y. Challal, "Batch-based cp-abe with attribute revocation mechanism for the internet of things," *2015 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1044–1049, 2015.

[157] J. Hao, C. Huang, J. Ni, H. Rong, M. Xian, and X. Shen, "Fine-grained data access control with attribute-hiding policy for cloud-based iot," *Comput. Networks*, vol. 153, pp. 1–10, 2019.

[158] S.-R. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38 437–38 450, 2018.

[159] K. Fan, J. Wang, X. Wang, H. Li, and Y. Yang, "A secure and verifiable outsourced access control scheme in fog-cloud computing," *Sensors (Basel, Switzerland)*, vol. 17, 2017.

[160] C. Yuan, M. xue Xu, X. Si, and B. Li, "Blockchain with accountable cp-abe: How to effectively protect the electronic documents," *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 800–803, 2017.

[161] Y. Zhang, D. He, and K.-K. R. Choo, "Bads: Blockchain-based architecture for data sharing with abs and cp-abe in iot," *Wirel. Commun. Mob. Comput.*, vol. 2018, pp. 2 783 658:1–2 783 658:9, 2018.

[162] D. Han, Y. Zhu, D. Li, W. Liang, A. Souri, and K.-C. Li, "A blockchain-based auditable access control system for private data in service-centric iot environments," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3530–3540, 2021.

[163] K. Xue, W. Chen, W. Li, J. Hong, and P. Hong, "Combining data owner-side and cloud-side access control for encrypted cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 13, pp. 2062–2074, 2018.

[164] S. Pérez, D. Rotondi, D. Pedone, L. Straniero, M. Nuñez, and F. Gigante, "Towards the cp-abe application for privacy-preserving secure data sharing in iot contexts," in *IMIS*, 2017.

[165] F. Guo, Y. Mu, W. Susilo, D. Wong, and V. Varadharajan, "Cp-abe with constant-size keys for lightweight devices," *IEEE Transactions on Information Forensics and Security*, vol. 9, pp. 763–771, 2014.

[166] S. Wang, H. Wang, J. Li, H. Wang, J. Chaudhry, M. Alazab, and H. Song, "A fast cp-abe system for cyber-physical security and privacy in mobile healthcare network," *IEEE Transactions on Industry Applications*, vol. 56, pp. 4467–4477, 2020.

[167] S. Ding, C. Li, and H. Li, "A novel efficient pairing-free cp-abe based on elliptic curve cryptography for iot," *IEEE Access*, vol. 6, pp. 27 336–27 345, 2018.

[168] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," *2007 IEEE Symposium on Security and Privacy (SP '07)*, pp. 321–334, 2007.

List of figures

List of tables

DOCTORAL SCHOOL

Titre : Déploiement et Application Optimisés de la Blockchain pour un Internet Industriel des Objets de Confiance

Mots clés : Blockchain, Internet Industriel des Objets, Arbre Trie, Hyperledger Fabric, Graphe acyclique dirigé, Preuve à divulgation nulle de connaissance

Résumé : La progression continue de l'internet industriel des objets (IIoT) offre des perspectives prometteuses et de nombreuses possibilités d'améliorer les cadres opérationnels des systèmes industriels. Cependant, les architectures de l'IIoT sont confrontées à des défis importants, notamment le contrôle centralisé, la vulnérabilité aux cyberattaques, les violations de la vie privée et les problèmes d'exactitude des données. Ces défis créent des obstacles importants à la sécurisation des données, qui est cruciale pour la croissance de cette technologie. Pour résoudre ces problèmes, de nombreux chercheurs suggèrent d'intégrer la technologie blockchain comme moyen stable de protéger les données au sein des systèmes IIoT. Les caractéristiques de stockage distribué, de décentralisation et d'immutabilité de la blockchain offrent des avantages significatifs en matière de stockage sécurisé des données, de vérification de l'identité et de contrôle d'accès.

Malgré ces avantages, à mesure que les applications IIoT se diversifient et que les volumes de données croissent, la forte demande en ressources des systèmes blockchain se heurte aux ressources limitées des appareils IIoT, ce qui entraîne des contradictions non résolues et des problèmes persistants. Les architectures blockchain existantes manquent encore d'authentification d'identité IIoT anonyme et efficace, avec des processus de cryptage et de décryptage complexes induisant une surcharge excessive du système. Pour résoudre ces problèmes, cette thèse s'appuie sur des recherches antérieures pour optimiser les performances de la blockchain, visant à résoudre les lacunes et les goulets d'étranglement des architectures IIoT actuelles basées sur la blockchain en ce qui concerne la protection de la sécurité des données.

Tout d'abord, cette thèse présente un protocole léger basé sur la blockchain, conçu pour le stockage sécurisé des données dans l'environnement dynamique de l'IIoT. Il intègre la cartographie bilinéaire pour l'initialisation du système, l'enregistrement des entités et la technologie d'authentification pour authentifier les entités IIoT de manière efficace et sécurisée, ainsi qu'une approche de stockage des données hors chaîne pour garantir l'intégrité des données avec une consommation de ressources réduite.

En outre, la thèse aborde les limites des systèmes Hyperledger fabric dans les scénarios de haute disponibilité en proposant Trie-Fabric, qui améliore le traitement des transactions grâce à un algorithme de tri des transactions basé sur un graphe acyclique dirigé (DAG). Cette approche réduit considérablement les transactions interrompues, optimise la gestion des conflits et augmente l'efficacité de plus de 60% dans le meilleur des cas, selon des résultats expérimentaux comparatifs.

Pour gérer les processus industriels de plus en plus sophistiqués et les données sensibles à la vie privée générées par les appareils IIoT, cette thèse propose un schéma de contrôle d'accès assisté par smart contract utilisant le modèle de contrôle d'accès basé sur les attributs (ABAC). Ce concept, supporté par des composants de filtre de bloom, permet de bénéficier de temps d'exécution de contrat contrôlés, d'un débit de système stable et d'un processus de consensus rapide dans des simulations réelles, ce qui le rend capable de gérer un débit élévé et un consensus efficace, même dans des scénarios de demande à grande échelle.

Enfin, cette thèse présente l'algorithme Zero-Knowledge Proof (ZKP), qui intègre un protocole de preuve à zéro connaissance non-interactif avec le chiffrement basé sur les attributs de la politique de chiffrement (CP-ABE) pour améliorer la sécurité et l'efficacité de la distribution de contenu IIoT. Combiné au système Distributed Publish-Subscribe IIoT (DPS-IIoT) utilisant Hyperledger fabric, il améliore considérablement l'efficacité de la bande passante et le débit global dans les environnements IIoT. Grâce à des évaluations complètes des performances de sécurité et à des résultats expérimentaux, ces recherches confirment l'efficacité des protocoles pour minimiser la surcharge du système, améliorer la fiabilité du stockage et renforcer la gestion globale des données de l'IIoT et la sécurité des applications.

Cette thèse fournit un examen approfondi des protocoles et systèmes de gestion de données avancés pour l'IIoT, qui sont cruciaux pour faire progresser le secteur de la fabrication. Par conséquent, ce travail apporte une contribution significative au domaine de la sécurité des données de l'IIoT, en offrant des solutions évolutives et robustes pour les systèmes industriels actuels et futurs.

Title : Optimized Blockchain Deployment and Application for Trusted Industrial Internet of Things

Keywords : Blockchain, Industrial Internet of Things, Trie Tree, Hyperledger Fabric, Directed Acyclic Graph, Zero-Knowledge Proof

Abstract : The continued advancement of the Industrial Internet of Things (IIoT) presents promising prospects and numerous opportunities for improving the operational frameworks of industrial systems. However, IIoT architectures face significant challenges, including centralized control, vulnerability to cyber attacks, privacy violations, and data accuracy issues. These challenges create significant obstacles in securing data, which is crucial for the growth of this technology. To address these issues, many researchers suggest integrating blockchain technology as a stable means to safeguard data within IIoT systems.

Blockchain's features of distributed storage, decentralization, and immutability offer distinct advantages in data secure storage, identity verification, and access control. Despite these benefits, as IIoT applications diversify and data scales expand, the high resource demand of blockchain systems clashes with the limited resources of IIoT devices, leading to unresolved contradictions and persistent issues within this solution. Existing blockchain architectures still lack anonymous and efficient IIoT identity authentication, with complex encryption and decryption processes inducing excessive system overhead. To address these issues, the thesis builds on prior research to optimize blockchain performance, aiming to resolve the shortcomings and bottlenecks in current blockchain-based IIoT architectures regarding data security protection.

Firstly, this thesis introduces a lightweight blockchain-enabled protocol designed for secure data storage in the dynamic IIoT environment. It incorporates bilinear mapping for system initialization, entity registration, and authentication technology to authenticate IIoT entities efficiently and securely, along with an off-chain data storage approach to ensure data integrity with reduced resource consumption.

Furthermore, the thesis addresses the limitations of Hyperledger fabric systems in high availability scenarios by proposing Trie-Fabric, which enhances transaction processing through a Directed Acyclic Graph (DAG) based transaction sorting algorithm. This approach significantly reduces terminated transactions, optimizes conflict handling, and increases efficiency by more than 60% in its best case, according to comparative experimental results.

To manage the increasingly sophisticated industrial processes and privacy-sensitive data generated by IIoT devices, the thesis proposes a smart contract-assisted access control scheme utilizing the Attribute-Based Access Control (ABAC) model. This scheme, supported by bloom filter components, demonstrates controlled contract execution times, stable system throughput, and a rapid consensus process in real-world simulations, making it highly capable of handling high-throughput and effective consensus even under large-scale request scenarios.

Lastly, the thesis introduces the Zero-Knowledge Proof (ZKP) algorithm, which integrates a non-interactive zero-knowledge proof protocol with Ciphertext-Policy Attribute-Based Encryption (CP-ABE) to enhance security and efficiency in IIoT content distribution. Combined with the Distributed Publish-Subscribe IIoT (DPS-IIoT) system using Hyperledger fabric, it significantly improves bandwidth efficiency and overall throughput in IIoT environments. Through comprehensive security performance evaluations and experimental results, this research confirms the protocols' effectiveness in minimizing system overhead, improving storage reliability, and enhancing overall IIoT data management and application security. This thesis provides an in-depth examination of advanced data management protocols and systems for the IIoT, which are crucial for advancing the manufacturing sector. Consequently, this work makes a significant contribution to the field of IIoT data security, offering scalable and robust solutions for current and future industrial systems.

Institut Polytechnique de Paris
91120 Palaiseau, France