FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

ADVANCING SCALABILITY, EFFICIENCY, AND STORAGE OPTIMIZATION IN BLOCKCHAIN FOR MOBILE INTERNET OF THINGS (MIOT) APPLICATIONS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Hussein Zangoti

2023

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Hussein Zangoti, and entitled Advancing Scalability, Efficiency, and Storage Optimization in Blockchain for Mobile Internet of Things (mIoT) Applications, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Sundaraja Sitharama Iyengar

Deng Pan

Leonardo Bobadilla

Jean Andrian

Wazir Zada Khan

Niki Pissinou, Major Professor

Date of Defense: June 15, 2023

The dissertation of Hussein Zangoti is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development and Dean of the University Graduate School

Florida International University, 2023

DEDICATION

I dedicate this dissertation to my beloved mother, father, siblings, family, and friends. Your constant encouragement, unwavering support, and strong belief in my potential have been essential in my academic voyage. The realization of this intellectual endeavor could not have happened without you.

ACKNOWLEDGMENTS

First, I sincerely appreciate my advisor Dr. Niki Pissinou for her invaluable knowledge, guidance, and support in my Ph.D. study. I'm incredibly thankful for her help in making me who I'm today.

Secondly, I would like to extend my sincere gratitude to my dissertation committee members, Dr. S.S. Iyengar, Dr. Deng Pan, Dr. Leonardo Bobadilla, Dr. Jean Andrian, and Dr. Wazir Zada Khan, for their invaluable assistance, suggestions, and insightful remarks on the research presented in this dissertation. I am particularly grateful to Dr. Wazir Zada Khan for his support throughout our research on designing a lightweight multidimensional blockchain for mobile IoT.

I also sincerely thank Dr. Abdur R. Shahid and Alex Makki-Pissinou for their assistance during my Ph.D. Their contributions to our efforts have been greatly appreciated, and their insightful comments and suggestions on my work have been an immense value to me.

I am also thankful for my late mates and friends, including Dr. Georges A. Kamhoua and Sheila Alemany. I was fortunate to work with some brilliant undergraduate students from the NSF REU and RET programs, including Omar J. Guerra, Joel Rodriguez, Manosij Roy Chowdhury, and Darren Ellsworth, for their contribution to the development of the multidimensional blockchain.

Lastly, I would like to express my gratitude to Jazan University, Florida International University, and the National Science Foundation's REU and RET programs for supporting my research.

ABSTRACT OF THE DISSERTATION

ADVANCING SCALABILITY, EFFICIENCY, AND STORAGE OPTIMIZATION IN BLOCKCHAIN FOR MOBILE INTERNET OF THINGS (MIOT) APPLICATIONS

by

Hussein Zangoti

Florida International University, 2023

Miami, Florida

Professor Niki Pissinou, Major Professor

The increasing adoption of blockchain technology in mobile Internet of Things (mIoT) networks requires the development of blockchain systems that are efficient, scalable, and optimized for resource utilization. While several studies have attempted to address these challenges, comprehensive solutions that adapt to the inherent mobility of mIoT systems are still lacking. This Ph.D. thesis investigates three innovative methods to advance the current blockchain model for mIoT systems.

First, a novel k-dimensional spatiotemporal, multidimensional, graph-based blockchain structure is introduced to address network partitioning issues caused by the mobility of IoT devices. This unique structure effectively manages blockchain nodes as they move between cell areas, resulting in smaller independent peer-to-peer subnetworks, each with its own blockchain copy. Experimental results demonstrate improved scalability and efficiency, with logarithmic growth as the blockchain size increases. Furthermore, the longest chain length is reduced by over 99.99% compared to traditional chain-based structures, making blockchain operations such as block appending or management more efficient.

Building upon the multidimensional blockchain foundation, the next stage of this research involves developing an efficient merging algorithm for graph-based

or multidimensional blockchains in mIoT networks. This algorithm addresses the challenge of merging partitioned blockchains that contain similar or identical blocks, which often require significant time and computational resources during the merging process. By leveraging depth-first search and Merkle tree techniques, the merging algorithm minimizes the time and computational resources spent on identical blocks, resulting in a 72% reduction in merging time compared to algorithms that do not handle block similarity.

Lastly, considering the limited storage capacity of mIoT systems, this thesis presents a novel Collective Signing-Based Blockchain Storage Optimization (CSBSO) model aimed at minimizing storage overhead in resource-constrained mIoT systems. The model utilizes the existing Collective Signing (CoSi) protocol to reduce storage requirements and leverages a multidimensional blockchain structure for efficient block management and retrieval. The storage optimization approach identifies and prunes the most irrelevant blocks based on the CoSi protocol. Evaluations using real-world datasets, such as the Ethereum Classic Blockchain and Facebook users datasets, demonstrate that the CSBSO model outperforms state-of-the-art storage optimization models, achieving approximately 92% storage space savings. These results underscore the potential of CoSi-based storage optimization in effectively reducing blockchain storage overhead in resource-limited applications.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

1.1 Background

The age of combining sensing, processing, and communication in moving, resource-constrained physical devices that connect and exchange data with other devices gives rise to a vast number of applications leading to endless possibilities and a realization of mobile Internet of Things (mIoT) applications. As IoT becomes more ubiquitous, safeguarding transactions becomes a critical piece of information and a vital factor for commercial success. Although it has been shown that blockchain can secure IoT, research and development in this area are still in progress. For example, current blockchain models for mobile IoT assume devices are resource-capable and there are fixed, powerful edge devices capable of providing global communication to all the nodes in the network. However, due to the mobile nature of IoT or network partitioning problems, nodes can move out of a cell area and split into smaller independent peer-to-peer subnetworks. Existing blockchain structures do not support the network partitioning problem or have limitations, such as poor efficiency, scalability, and insufficient storage optimization. This research stems from the recognition that the wide applicability of mobile IoT will remain elusive unless new techniques are introduced to efficiently represent moving objects by organizing and partitioning the data points based on specific conditions of moving through time and space of resource-constrained devices.

This work is one of the first steps that aims to address the challenges of blockchain-based applications in resource-constrained, self-organized, and self-configured mobile IoT devices facing the split and merge problems due to frequent node mobility and network partitioning, the inefficiency of blockchain in terms of scalability and stor-

age optimization. Our objective is to: (1) design and develop a multidimensional, graph-based blockchain structure that utilizes k-dimensional spatiotemporal space to scan only a few blocks for all blockchain operations and as a method for the blockchain to continue to provide its service when the network of mIoT miners is partitioned; (2) propose an efficient blockchain merging algorithm that minimizes the processing of identical blocks by utilizing techniques like Depth-First Search and Merkle Tree Hash, which help to identify identical blocks and save computational resources; (3) develop a Collective Signing-Based Blockchain Storage Optimization (CSBSO) model to minimize storage overhead of mIoT nodes. The class of application environments for which our research and approach are suitable and valuable includes time and space-dependent applications or systems that require some form of action, such as cyber-defense, defense logistics support, and the networking of the defense Internet of Military Things, tactical-level data management, event sharing or other lightweight dynamically data-driven applications.

1.2 Motivation

As the Internet of Things (IoT) becomes more ubiquitous and continues to grow and evolve, it enormously increases interconnected devices, producing a large amount of data. This influx of data calls for secure, efficient, and scalable data management solutions. Specifically, mIoT systems, a subset of IoT, face unique challenges such as dynamic networks, resource constraints, and the demand for effective data handling and storage solutions. Although blockchain technology holds the potential to tackle these challenges, existing implementations still exhibit limitations that need to be addressed to exploit blockchain's capabilities in mIoT environments fully.

Our motivation for this research arises from three primary concerns. These concerns are detailed below:

1. **Blockchain data structure**: Traditional blockchain structures usually rely on high network connectivity and robust edge devices. Unfortunately, these requirements are not well suited for mIoT systems, which frequently experience network partitioning and mobility challenges [KJW+18, WHH+19, ASHN21]. Blockchain structures must adapt to the dynamic nature of mIoT systems while ensuring data consistency and trust in an environment where trust is not inherently present. Additionally, the proposed blockchain structure should enable efficient data management among devices and adapt to constant changes in network topology while preserving shared data's security and privacy. Traditional blockchain systems are designed in a linear-based structure Nakamoto [Nak08]. This type of structure can result in undesirable performance in terms of scalability, throughput, and confirmation time [WYCX20]. In addition, chain-based data structures can result in computation, storage, and communication overheads, as observed by [XLZ+21]. Moreover, [LCP+20] pointed out that a single chain structure can experience centralization concerns because powerful nodes have a higher chance to generate the next block and high transaction fees because expensive nodes verify transactions. Several graph-based structures, such as Directed Acyclic Graph or (DAG)-based blockchain, were introduced to address these bottlenecks. Literature improvements include developing DAG-based blockchains that can process or confirm transactions in parallel, requiring fewer communications, computations, and storage overhead [SLZ16, PMIH18, BKLMC20, ZHZB20]. Despite all the benefits of the DAG structure, a large-scale DAG-based blockchain, such as IoT scale, consumes higher computations than traditional linear-based blockchains [WHH+19].

2. **Blockchain merging**: Graph-based and multidimensional blockchain structures can outperform chain-based structures in mIoT systems in multiple aspects, such as in terms of scalability, throughput, and confirmation time [WSNH19]. However, existing implementations of these graph-based blockchain structures tend to be inefficient when handling frequent split and merge events in mIoT applications. This leads to unnecessary processing of identical blocks between two blockchain splits and increased resource consumption [WYY+22, LWQ+19, ZMP+22]. A crucial requirement for mIoT systems is an efficient merging algorithm that minimizes the time and computational resources dedicated to processing identical blocks between different splits. Furthermore, the proposed algorithm should be adaptive to varying network conditions and resource availability, ensuring that mIoT devices can effectively collaborate under challenging circumstances.

3. **Blockchain storage**: One significant scalability challenge in blockchain systems is the storage overhead, especially in resource-constrained environments [WLC+22]. Existing storage optimization solutions, such as data compression or off-chain storage, may not be ideal for large-scale, resource-constrained mIoT devices due to their high complexity, compromised security by pruning essential blocks, and weakened decentralization by relying on few trusted nodes to store the entire blockchain copy [AMTS+22]. Innovative storage optimization techniques are needed, ones that can dynamically adjust storage requirements without forfeiting the advantages of blockchain technology. The proposed storage optimization model should balance data availability, flexibility to resource requirements, and resource consumption while adhering to blockchain decentralization and security principles.

In summary, this research is motivated by the need to adapt blockchain technology to better serve mIoT systems. We focus on enhancing blockchain structures for dynamic mIoT environments, improving blockchain merging efficiency, and innovating storage optimization solutions. With these motivations, we aim to advance the application of blockchain technology within the mIoT systems context.

1.3 Research Problems

Based on the motivation section, we have identified several research problems. These problems primarily focus on investigating blockchain solutions for mIoT, emphasizing dynamic network mobility, efficient and scalable blockchain structure, efficient merging algorithms, and storage optimization techniques. The research problems are as follows:

1. **Research Problem 1: How can we design an efficient and scalable blockchain structure that can support the dynamic network mobility and network partition problems in mIoT?**

   This research problem aims to understand the limitations of current blockchain structures (mainly chain-based and graph-based structures) considering the dynamic mobility nature of mIoT. Current blockchain models for mobile IoT assume there are fixed, powerful edge devices capable of providing global communication to all the nodes in the network. However, due to the mobile nature of IoT or network partitioning problems (NPP), nodes can move out of a cell area and split into smaller independent peer-to-peer subnetworks. Existing blockchain structures either do not support the network partitioning problem or have limitations [WYY+22]. We plan to explore innovative blockchain structures, such as graph-based structures, which have been shown to perform

better than chain-based structures, as shown in [WYCX20]. The blockchain structure should support network partitioning, mobility, and t. In addition, the proposed structure must also maintain data consistency, efficient block management, and provide trust in a trustless environment. Furthermore, we will analyze the trade-offs between various blockchain structures.

2. **Research Problem 2: How can we design an efficient blockchain merging algorithm that minimizes the time and computational resources?**

   Graph-based or multidimensional blockchains have been proposed to improve the scalability and efficiency of existing blockchain applications. However, when implemented in mobile Internet of Things (mIoT) networks, these blockchain systems can frequently split and merge and cause the merging algorithm to process many similar blocks (a block is similar or identical when it exists once in multiple blockchains). Similar blocks hinder the merging process, as they consume time and computational resources to scan, validate, and potentially merge similar blocks. This research problem aims to create an efficient merging algorithm that can reduce the computational overhead associated with processing and managing identical blocks. We plan to investigate existing merging algorithms and identify their shortcomings in mobile applications. The proposed algorithm should adapt to changing network conditions, avoid processing identical blocks, and resource availability. Finally, we will assess the proposed merging algorithm's performance and compare it with existing merging algorithms to show its effectiveness.

3. **Research Problem 3: Which storage optimization techniques can be employed to reduce the storage overhead for blockchain in mIoT?**

Due to the increase in data on the blockchain and the addition of blockchain characteristics, the traditional blockchain's full chain copy storage requirement causes a blockchain storage scalability problem. Blockchain systems' scalability issues include low throughput, excessive data load, and ineffective query engines. All of these difficulties are closely tied to data management [WLC+22]. This research problem aims to pinpoint and design a novel storage optimization technique for mIoT systems. We will investigate distinct data pruning approaches that decrease storage overhead. The proposed storage optimization model should balance data availability, query efficiency, and resource requirements. Furthermore, we will explore the limitations of current storage optimization solutions and compare them with our approach.

In summary, this section outlines three critical research problems concerning the mIoT blockchain: designing a scalable blockchain structure for dynamic networks, developing an efficient blockchain merging algorithm, and identifying storage optimization techniques. We aim to improve blockchain's performance in mIoT systems by addressing these challenges.

1.4 Research Objectives and Contributions

This section outlines the primary research objectives, addressing challenges concerning blockchain structures, blockchain merging efficiency, and storage optimization. This work aims to achieve these objectives through the development and experimental demonstration of the following:

1.4.1 Objective 1: Design a Scalable, Efficient, Graph-Based Blockchain Structure for Mobile IoT

The traditional blockchain structures for mIoT are usually built as a chain of blocks, where each block stores a list of transactions. In order to add or manage blocks, for instance, these models require traversing the entire blockchain chain to append and verify each block [Nak08]. This sequential processing of blocks can impact the blockchain model's efficiency and scalability, leading to higher computational resources [YDXJ20]. We aim to achieve the first objective by designing a blockchain that utilizes k-dimensional spatiotemporal space that can achieve efficiency by managing or scanning only a few blocks (fewer comparisons) for any blockchain operations. In addition, we aim to achieve scalability by achieving performance better than linear growth, which is a limitation of traditional blockchain structures.

- Examine the current chain and graph-based blockchain structures, their limitations in managing network partitioning, effectiveness in mobile environments, and other challenges in mobile IoT systems.
- Create a graph-based blockchain structure that is designed specifically for mobile IoT systems, capable of handling network partitioning or low connectivity.
- Assess the proposed blockchain structure performance in terms of scalability, efficiency, and practicality in mobile networks and comparing it with existing structures such as chain and graph-based blockchains.

Research Contributions 1: A Multidimensional Blockchain Framework For Mobile Internet of Things [ZMP+22]

Our contributions in this section include developing a graph-based blockchain structure that is immune to the dynamic merge and split nature of mobile IoT systems

while maintaining data consistency and trust in a trustless environment. The proposed model is called a multidimensional blockchain (MKDBC) [ZMP+22], which is a type of graph-based blockchain that structures blocks as k-dimensional trees using [Ben75]. The MKDBC can improve the scalability and efficiency over existing blockchain systems by utilizing an efficient algorithm using binary search [Ben75] for all blockchain operations. The MKDBC is composed of two main components. The first is an immutable k-dimensional tree structure for structuring and storing blocks, and the second is a mutable Merkle tree representing all the blocks within the blockchain. The MKDBC also employs multi-layer indexing to sort and organize blocks, which allows for faster block scanning, forging, and validation, resulting in increased efficiency and scalability of the overall blockchain system. The model is also a partition-tolerance blockchain that supports the splits and merges of blockchains in dynamic networks. Finally, we assess the proposed blockchain structure performance in terms of scalability, efficiency, and practicality in a mIoT, comparing it with existing structures such as chain and graph-based blockchains.

1.4.2 Objective 2: Enhance the Merging Efficiency in Graph-Based Blockchains for Mobile IoT Systems

Graph-based blockchain structures are designed to enhance the scalability and efficiency of various blockchain applications. However, when applying these models to mIoT, they tend to face some challenges that could potentially affect the merge efficiency. To illustrate, some blockchain models support networking partitioning, allowing the blockchain to split and merge. As these blockchain models continue to split and merge due to the low connectivity or moving out of cell areas, they tend to face the issue of identical blocks (a block is similar or identical when it exists once

in multiple blockchains) between two blockchains when they are trying to merge. This leads to the merging algorithm dealing with a large number of identical blocks, making the process more resource-intensive. We aim to develop an efficient merging algorithm that handles the issues of identical blocks. The investigation for this issue includes the following sub-objectives.

- Investigate existing blockchains that support blockchain merging and identify their limitations. The main focus is on blockchain models where network partitioning is likely.
- Propose an efficient merging algorithm that minimizes the processing of identical blocks during the merge process by utilizing techniques like Depth-First Search and Merkle Tree Hash, which help to identify identical blocks.
- Implement the suggested merging approach, evaluating its performance through multiple metrics such as the number of blocks scanned, skipped, merged, and the overall performance.
- Compare the proposed approach's efficiency with traditional merging methods and emphasize its improvements.

Research Contributions 2: An Efficient Approach for Merging Multidimensional Blockchains in Mobile IoT [ZP23]

The main contributions for blockchain merging include identifying factors that can affect the merging operations, such as the presence of identical blocks since nodes need to spend unnecessary time and resources on scanning or validating them. The second contribution is to design an efficient merging algorithm that combines Depth-First Search [Tar72] and Merkle tree or hash [Mer89]. Depth-first search is a technique we adopt to improve the efficiency of the merging process in the presence of

identical blocks, and Merkle tree or hash enables blockchain nodes to quickly and efficiently identify identical subtrees or groups of blocks by comparing their Merkle hashes. The merging algorithm combines the two technologies to help avoid scanning identical blocks or subtrees or going to higher depths in the blockchain and reduce the time and resources spent on processing similar information. Finally, we assess the model and compare it with the traditional merging algorithms.

The contributions are as follows:

- Identifying factors that can affect the merging process, such as all operations associated with managing identical blocks.
- We propose an efficient approach for merging multidimensional blockchains that reduces the amount of processing of identical blocks during the merging process.
- Employing Depth-First Search and Merkle Tree Hash, we design an efficient recursive algorithm that merges two multidimensional blockchains.
- Our simulation results show that the proposed approach outperforms the traditional merging approach in terms of scanned and skipped blocks and the reduction in merge time. Moreover, the proposed approach can reduce the merging time by more than 72% compared to traditional merging approaches.

1.4.3 Objective 3: Design a Collective Signing-Based Blockchain Storage Optimization Model for Mobile IoT

Blockchain storage emerges as a significant challenge, particularly in resource-constrained systems. In a standard blockchain model, all the nodes are generally expected to maintain a full copy of the blockchain, applying significant challenges to resource-

constrained devices. Many researchers have investigated this issue and proposed alternative storage optimization solutions, including block pruning, data compression, and on and off-chain data offloading to the cloud or specific trusted nodes. However, each type of solution has its limitations, such as pruning essential blocks, which could affect data integrity and weaken decentralization [WLC+22, LCZ+22, XFRZ20, NTG+22, ZLCD18, HGW+22]. We aim to implement a storage optimization solution that solves these problems using the Collective Signing protocol. The goal is to achieve sufficient blockchain storage optimization through the following sub-objectives.

- Investigate existing storage optimization methods and their limitations in solving the storage scalability problem in mIoT systems.
- Develop a Collective Signing-Based Blockchain Storage Optimization (CSBSO) model while leveraging a multidimensional blockchain structure and the CoSi protocol to minimize storage overhead.
- Create a storage optimization algorithm that identifies and removes irrelevant blocks of varying degrees.
- Conduct experiments with real-world datasets to evaluate the proposed CSBSO model's performance in storage space reduction and compare it with state-of-the-art techniques.

Research Contributions 3: CSBSO: Collective Signing-Based Blockchain Storage Optimization for Mobile Internet of Things

Our contributions to achieve blockchain storage optimization are that we have developed a dynamic Collective Signing-Based Blockchain Storage Optimization (CSBSO) model that seeks to dynamically reduce storage overhead for mobile IoT while

retaining the ability to adjust storage requirements based on the resources capability of mIoT devices. Another contribution is that we proposed a storage optimization technique targeted at identifying and eliminating irrelevant blocks of varying degrees of irrelevance ranging from the most to least relevant blocks using the CoSi protocol. Finally, extensive experiments demonstrate that our proposed technique can significantly reduce the storage cost of mIoT nodes when compared to existing storage optimization techniques by applying real-world datasets.

The main contributions of this section are as follows:

- We propose CSBSO, a Collective Signing-Based Blockchain Storage Optimization model for Mobile IoT, to reduce the storage costs of IoT nodes.
- Based on the CoSi protocol, we design a storage optimization algorithm that can identify the most relevant/irrelevant blocks and prune the irrelevant blocks.
- By employing two real-world datasets (i.e., Ethereum Classic Blockchain and the Facebook Users), our simulation results show that the proposed CSBSO can effectively optimize storage and outperform existing state-of-the-art techniques and save up to 92% of storage space in mobile IoT environments.

1.5 Dissertation Outline

The outline of this dissertation is as follows: Chapter 2 presents a review of the related work in the domain of blockchain data structures, partitionable blockchains, and storage optimization of blockchain systems. The details of the multidimensional blockchain model for mIoT are presented in Chapter 3. Chapter 4 proposes an efficient approach for merging graph-based blockchains. Chapter 5 proposes a novel Collective Signing-Based Blockchain Storage Optimization (CSBSO) model that aims to reduce the storage overhead. The last section of the dissertation, Chapter

6, provides an overview of the proposed works' limitations and suggests various directions for our future works.

CHAPTER 2

RELATED WORK

This chapter presents a comprehensive review of the existing work on blockchain data structures, blockchain merging, and blockchain storage optimization techniques, specifically addressing the challenges and solutions proposed for mobile IoT applications. The goal is to review existing literature, identify the different techniques and solutions, and give perspective on their limitations to help establish the foundation of this thesis's contribution. Finally, we discuss various approaches, including chain and graph-based blockchains data, merging approaches, and storage optimization solutions such as pruning, compression, and on and off-chain storage optimizations. We also highlight their contributions and limitations in the scope of mIoT.

2.1 Blockchain Data Structures

Several works were proposed to address the issues of scalability, throughput, and confirmation time when using chain-based blockchain structure. Examples of these works include sharding [WSNH19], sidechain [BCD+14], and cross-chain [ZABZ+19]. In sharding, the system divides pending transactions into smaller pieces called shards in order to process them in parallel [ZMR18]. Sidechain, an additional solution to improve traditional blockchain structure, allows digital assets to be transferred between the main chain and sidechains [WWC+19]. Cross-chain can help improve traditional blockchains by establishing communication between multiple blockchains and allowing digital assets to transfer between them. Although these approaches can enhance chain-based blockchain functionality, their backbone structures are still based on a chain-based blockchain structure [WYCX20] which is not suitable for wireless mobile networks. In addition, chain-based structures can suffer from linear

growth scalability and inefficiency. The end of this section will explain why graph-based blockchains are more suitable for mobile wireless networks than chain-based blockchains.

Shahid *et al.*[SPSK19] proposed a lightweight, scalable blockchain system for resource-constrained Internet of Things devices called "Sensor-Chain." Their proposed model allows nodes to split into multiple networks based on regions, and each network has its independent blockchain. The model also enables blockchains to merge by aggregating blocks into a single block, saving storage resources. However, the model periodically erases some historical blocks when aggregating blocks after the merge. This produces a fundamental issue because the missing historical blocks could contain critical data and negatively impact data availability and consistency. Furthermore, the model uses a chain-based blockchain structure, which is not favorable to recourse-constrained devices due to the poor overall performance metrics discussed earlier.

Due to resource limitations in IoT, many researchers proposed approaches to offload the blockchain data towards more centralized IoT resources such as edge, fog, or Roadside Units (RSU), where full nodes are located and can store the entire blockchain. A DAG-based blockchain system was proposed by Yang *et al.* [YDXJ20] to enable a lightweight and secure data storage structure for resource-constrained Vehicular Social Network devices (VSNs). Another work suggested aggregating the blockchain data through base stations, roadside infrastructure, or service providers. Danzi *et al.*[DKSP19] proposed a blockchain system for lightweight IoT devices with delay and communication tradeoffs. In their work, the blockchain data is aggregated in periodic updates to reduce the communication cost of the IoT clients. Each IoT client is connected to a set of blockchain networks through wireless base stations. Memon *et al.* [MLN+19] proposed a blockchain based DualFog-IoT system with

three configuration filters that can specify the type of incoming requests. These filters are Real-Time, Non-Real Time, and Delay Tolerant Blockchain applications. The DualFog architecture splits the fog layer into two parts. Fog Cloud Cluster where it communicates with the cloud, and Fog Mining Cluster which includes a group of trusted fog devices that are responsible for mining for blockchain-based applications. However, all the nodes must maintain communication with all other nodes or at least with one full node that's always connected through edge/fog devices.

Kim *et al.* [KJ17] proposed a graph-based blockchain system, called Binary Blockchain, that can split and merge blockchains to handle the mining congestion problem. Mining congestion is a major problem in blockchains which can cause higher transaction confirmation time. The Binary Blockchain system splits chains when the load goes above a threshold and reduces the number of chains when the load goes below a threshold. However, the proposed model must maintain communication between other multiple subchains using sync blocks. These sync blocks were introduced to ensure balance mining between multiple chains. In our proposed model, we assume that a split of forgers can work together without maintaining any type of communication with all network participants.

Geng *et al.* [GNH21] proposed a solution for tasks accomplishment assurance in blockchain systems for IoT. The system uses a DAG-based blockchain to ensure nodes can participate in one-to-many and many-to-one tasks accomplishment without acting maliciously. The authors addressed the incapability of single-chain blockchain to support one-to-many and many-to-one dependencies. Their solution involves branching/merging of a DAG blockchain to support one-to-many and many-to-one dependencies that satisfies recognizability, compatibility, and authenticability.

Laube *et al.* [LMAA19] proposed a graph or DAG-based blockchain model where a block can have one or multiple parent/child blocks. Their work is the first to solve the split and merge problem (network partitioning problem) caused by nodes' mobility in mobile ad hoc networks (MANETs) using DAG. However, their model does not detect topology changes such as network split and only relies on flooding to disseminate data, maintain communication, and passively detect changes in topology. Not being able to actively detect topology changes can cause issues with adjusting consensus for each split [CLP+20]. Our model can actively detect topology changes and adjust consensus accordingly, even when the network is dealing with the network partitioning problem using the work proposed by [MVL+21].

In summary, most related works, such as [WSNH19, BCD+14, ZABZ+19, ZMR18, SPSK19], use a chain-based blockchain structure that stores all the data on a single chain, limiting the processing/mining of transactions/blocks due to the mining competitiveness [WSNH19]. All miners or forgers can work on a single chain, which could result in them doing the exact operations and cause wasted valuable resources such as computational power, communication, and storage. Furthermore, only working on a single chain does not allow nodes to process/add blocks to the public ledger simultaneously, affecting the system's overall throughput.Considering mobile wireless networks, they usually tend to have resource-constrained devices, making chain-based blockchain structures an unfavorable fit for them. Instead, our model uses a graph-based blockchain which has shown to have an overall better performance, as seen in [WSNH19]. Many related works, such as [XZZ+22, YDXJ20, DKSP19, MLN+19, YN21], assume there are fixed, powerful devices (or high connectivity) capable of continuously providing global communication to all the nodes at all times. However, due to the mobility nature of IoT and MANETs, nodes can split into smaller and independent peer-to-peer subnetworks due to the absence of

edge devices. Our model is partition-tolerant and does not require high network connectivity or fixed, powerful edge devices to provide connectivity to all nodes at all times. Current graph-based blockchains, such as DAG-based blockchains [YDXJ20, LMAA19, CZC21, LCP+20, GYZ+19], have a better partition tolerance because the blockchain structure can adapt to the dynamic changes in network typologies. However, they can suffer from the following limitations. First, some rely on full connectivity to all nodes using edge devices. Second, although they can perform better than chain-based blockchains, the DAG structure does not intrinsically order blocks, but partial ordering is possible as in [KJW+18, LWQ+19]. For ledgers to achieve consensus, they may need to traverse to ancestors' blocks [GNH21]. Third, a large-scale DAG-based blockchain consumes higher computational resources than traditional linear-based blockchains [WHH+19]. The proposed model in Chapter 3.4 can always order blocks which can help facilitate the split and merge of blockchains by efficiently scanning and adding blocks between multiple blockchains.

2.2 Blockchain Merging

Laube *et al.* [LMAA19] proposed a graph or DAG-based blockchain model where a block can have one or multiple parent/child blocks. Their work is the first to solve the split and merge problem (network partitioning problem) caused by nodes' mobility in mobile ad hoc networks (MANETs) using DAG. Cordova *et al.* [CLP+20] also proposed another DAG-based blockchain that structures blocks based on network partitioning. However, their model does not detect topology changes such as network split and only relies on flooding to disseminate data, maintain communication, and passively detect changes in topology. Not being able to actively detect topology changes can cause issues with adjusting consensus for each split [CLP+20].

Our model can actively detect topology changes and adjust consensus accordingly, even when the network is dealing with the network partitioning problem using the work proposed by [MVL+21].

Geng *et al.* [GNH21] proposed a solution for tasks accomplishment assurance in blockchain systems for IoT. The system uses a DAG-based blockchain to ensure nodes can participate in one-to-many and many-to-one tasks accomplishment without acting maliciously. The authors addressed the incapability of single-chain blockchain to support one-to-many and many-to-one dependencies. Their solution involves branching/merging of a DAG blockchain to support one-to-many and many-to-one dependencies. However, The unique constraints and characteristics of mIoT systems, such as low connectivity or moving out of cell areas, are not specifically addressed in their work.

Fang *et al.* [FHB+22] implemented a solution to improve blockchain resilience when dealing with network partition. The blockchain design allows a dynamic group of consensus and active miners to collectively sign each block using the ByzCoin protocol developed by [KJG+16]. The model supports network partition by creating multiple shards or forks based on the number of network partitions, replacing the notion of the longest chain rule with all chains. A blockchain split happens when fewer active users sign the new block (or a partition of the consensus group), and a merge happens when more active users sign the new block. The merging process links different forks into the main fork. However, this model does not actively detect the changes in typologies and only relies on passive detection by checking the number of validators of the new block. Meanwhile, the blockchain backbone structure is based chain-based structure that can suffer from linear scaling [WYCX20]. Finally, it's also unclear how the model can react to frequent merges because frequent merges can cause multiple forks. This makes it challenging to maintain and organize the

blocks/forks in order to facilitate scanning and merging blocks, and the study only examined up to 3 partitions.

Hood *et al.* [HONS21] explored the partitionable blockchain consensus problem and implemented a solution that uses a group of partitioning detectors that provide crucial information to the algorithm, such as whether a block was delivered to all peers or whether a block was mined before a blockchain split. The authors also proposed two merging scenarios, competitive and cooperative merges, without providing details about the algorithms. Meanwhile, these merging algorithms can only retain some but not all of the blocks by running a block catch-up procedure on linear chains, which requires traversing all the blocks resulting in an efficient sequential processing [YDXJ20]. In addition, the effect of frequent splits and merges and block similarity on the merging algorithms is not discussed.

To summarize, one major issue with current blockchain merging approaches is that they often rely on inefficient and unscalable data structures. For example, the backbone blockchain structure is based on a chain-based structure in [FHB+22] approach, which is not well-suited to handling devices with limited resources and large-scale applications [WYCX20]. As a result, these approaches can quickly become slow, limiting their usefulness in real-world applications. Another issue is that current graph-based approaches often require nodes to process an entire copy of the blockchain, including identical blocks, which can be impractical for devices with limited storage or computational resources. This can lead to potential slow performance and scalability issues, making it difficult for blockchain systems to handle large-scale or resource-constrained applications.

2.3 Blockchain Storage Optimization Techniques

Traditional blockchain models require all nodes to store a full copy of the data at each node, posing some substantial challenges concerning scalability, storage, and performance for IoT devices. Addressing the issues is paramount, prompting many researchers to propose various innovative approaches for storage optimization in blockchain for IoT

One way to achieve storage optimization in blockchain applications is to maintain a full copy among designated nodes. Zihuan Xu et al. [XHC18] proposed a group-based storage scheme for a blockchain system called CUB. The Consensus Unit (CU) based concept is introduced in which different groups of nodes form a unit, and each unit assigns at least one copy of Blockchain blocks for storage. Blocks Assignment Optimization (BAO) determines the allocation of blocks to each unit, which ensures the full utilization of storage space and minimizes query cost for the optimal assignment. They have also presented solutions to address the dynamic scenarios when new blocks arrive and nodes join or leave the CU. Finally, the authors used BLOCKBENCH to perform a benchmark evaluation and verify the effectiveness of their proposed CU-based Blockchain system. Similarly, Yanqing Fan et al. [FQZ+22] proposed a group storage mechanism based on Double-Layer Blockchain Network called DLBN, which improves the blockchain's internal data composition structure. The DLBN contains two types of blockchain nodes, which serve as the system's storage and consensus layers. The storage layer nodes are divided into multiple storage units (SUs), and all nodes in the SU jointly maintain a copy of the entire blockchain, reducing the storage burden on the nodes. The consensus layer handles tasks like transaction sequencing, validation, and block wrapping, which increases system transaction throughput. Based on the DLBN model, they

proposed a reputation-based consensus mechanism, a block storage allocation algorithm, and a transaction query optimization algorithm. The storage model based on the DLBN can effectively improve system transaction throughput and reduce node storage capacity while ensuring system security through experimental verification and analysis. However, deploying these storage optimization approaches in mobile IoT applications may encounter challenges, especially when nodes within the network unexpectedly vanish or malfunction, potentially leading to the permanent loss of crucial data. The dynamic solutions presented in [XHC18, FQZ+22] may not be sufficient in ensuring data integrity in the face of node malfunctions or disappearances because their approaches rely on the formation of Consensus Units (CUs) to store at least one copy of blockchain or storage units (SUs) responsible for maintaining a copy of the entire blockchain.

Wenhui Yang et al. [YDXJ20] proposed LDV, a lightweight Blockchain for vehicular social networks (VSNs) that utilizes a Directed Acyclic Graph (DAG)-based blockchain structure. Based on an in-depth analysis of VSNs, the authors proposed the social-based data reduction approach. Each node stores only the relevant data within the subject groups of interest and discards the irrelevant data. They also present the historical data pruning method within a group, which meets the storage requirement by reducing the number of duplicates stored in each node, thereby avoiding the massive storage costs associated with large-scale groups containing large amounts of data. According to experimental results, LDV saves 97.13% of storage space and has good scalability. However, the study lacks details about the community interests and does not implement their approach using real-world datasets that contain real social network information.

Zhuofan Liao et al. [LCZ+22]. have proposed a DAG-Blockchain Storage Strategy Based on Graph Partitions in Edge-cloud industrial Internet-of-things (IIoT)

called GpDB. The GpDB includes a graph partition algorithm based on transaction freshness that can divide a DAG-blockchain topology in edge servers into two parts that will be retained and removed. The activity of the two nodes that initiated the transaction determines the transaction's freshness. The node activity indicates whether the node actively engages in edge computing behaviors in the IIoT environment. In terms of storage cost, GpDB outperforms LDV and Layerchain by 62% and 74%, respectively, and with an increasing number of transactions, GpDB has good scalability in reducing storage cost and better transaction throughput than IOTA. However, the model relies on edge infrastructure which could affect blockchain decentralization.

To avoid requiring all nodes to store a full copy of the blockchain, which can pose challenges in terms of scalability, storage, and performance, particularly for resource-constrained devices. Several blockchain systems propose storing the full copy of the blockchain ledger in off-chain storage solutions such as cloud, edge or distributed data storage [LWLX19, SJ20, DKJG19, MRR+20, XZYS20, SS19, LYW+21, LCZ+22, XFRZ20, NTG+22, ZLCD18, PB19, HGW+22]. In addition to the off-chain storage solution, other works suggest a group of nodes to maintain the full ledger instead of having all nodes store a full copy. For example, [LNZ+16, TLJ+20, XHC18, DDK+21], suggest assigning a group of nodes, usually trusted, to jointly maintain a full blockchain data rather than having all the nodes store a full copy individually. Although the previous approaches can help reduce the storage burden on the nodes and improve the scalability and performance, they rely on fewer trusted nodes to maintain a full copy which can weaken decentralization [WLC+22].

Various methods have been explored to address storage issues in blockchain systems, such as pruning older blocks. Nakamoto introduced two techniques, Reclaiming Disk Space (RDS) and Simplified Payment Verification (SPV), in [Nak08].

However, SPV has notable drawbacks, including dependence on centralized third parties for storing the complete ledger, which can result in security risks and network centralization [Bitb, XHC18]. Several researchers have proposed alternative strategies for pruning older blocks, as seen in [FHBS19, Bita, LCL+20]. However, these approaches might not be ideal, as they can prune relevant blocks or compromise the blockchain's security. Another pruning method involves downloading blockchain snapshots, as suggested by [MKP+21]. In a recent study, [WWZC21] proposed a more targeted pruning approach centered around transaction weight. This technique assigns weights to unspent transactions within each block, which assists in identifying the blocks to be pruned. Nevertheless, this approach is better suited for crypto-asset applications and may not be optimal for IoT applications [AMTS+22].

In summary, many approaches have been explored to tackle storage challenges in blockchain systems. However, these methods often exhibit limitations, such as a reliance on off-chain solutions that depend on full nodes setting at the edge or fog, which can weaken decentralization. Additionally, they are more suitable for specific applications, such as crypto-asset-based applications, rather than general IoT use cases. Furthermore, they lack targeted and flexible storage solutions capable of identifying the most or least relevant blocks at any given time and dynamically adjusting storage requirements.

CHAPTER 3

A MULTIDIMENSIONAL BLOCKCHAIN FRAMEWORK FOR MOBILE INTERNET OF THINGS

The adoption of blockchain in the Internet of Things (IoT) has been increasing due to the various benefits that blockchain brings, such as security and privacy. Current blockchain models for mobile IoT assume there are fixed, powerful edge devices capable of providing global communication to all the nodes in the network. However, due to the mobile nature of IoT or network partitioning problems (NPP), nodes can move out of a cell area and split into smaller independent peer-to-peer subnetworks. Existing blockchain structures either do not support the network partitioning problem or have limitations. This chapter introduces a multidimensional, graph-based blockchain structure, that utilizes k-dimensional spatiotemporal space, to address the challenges of applying blockchain in mobile networks with limited resources. Experimental results show that a multidimensional blockchain structure can improve scalability and efficiency as the blockchain grows in size, similar to logarithmic growth, and reduce the longest chain length by more than 99.99% compared to the traditional chain-based blockchain structure.

Our contributions in this chapter include:

- developing a graph-based blockchain structure that is immune to the dynamic merge and split nature of mobile IoT systems while maintaining data consistency and trust in a trustless environment.
- designing a blockchain system that can improve scalability and efficiency over existing blockchain systems by utilizing an efficient algorithm using binary search [Ben75] for blockchain operations.

The rest of this chapter is organized as follows: Section 4.1 presents an introduction. Section 3.2 is an overview and background knowledge related to this chapter. Section 3.3 describes the preliminaries and system assumptions. Section 3.4 shows the proposed model. In Section 3.5, we discuss the simulation setup and experimental results. And lastly, Section 3.6 presents the conclusion of this chapter.

3.1 Introduction

Blockchain presented by Nakamoto [Nak08] is a time-stamped append-only log technology that is usually decentralized, immutable and has led to innovative applications in many areas such as finance, healthcare, distributed systems, voting, industry, and real estate. When integrating blockchain into other domains with limited resources, such as mobile IoT, which is the focus of this chapter, there are multiple challenges to address. Traditional IoT blockchain systems rely on high network connectivity, which implies they depend on fixed, powerful edge devices capable of continuously providing global communication to all the nodes [KJW+18]. However, considering the mobility nature of IoT, nodes can split into smaller and independent peer-to-peer subnetworks due to the absence of edge devices. Existing blockchain structures either do not support the network partitioning problem or have limitations such as poor efficiency or resource consumption [ASHN21, WHH+19].

A chain-based blockchain usually aims to grow on a single chain (i.e., the longest chain) in which blocks are ordered by their creation time. When a network split occurs, two or more subnetworks can continue adding blocks to their blockchains, creating distinct blockchain copies. This scenario is possible in mobile networks, and the data from all subchains or blockchain copies could be equally important. When these subnetworks attempt to merge again, chain-based consensus algorithms

usually favor selecting the blockchain copy with the longest chain and appending any future blocks on the longest chain. This approach also ignores other blocks from sub-chains or forks, which can be crucial for future use.

Traditional blockchain systems are designed in a single chain or a linear-based structure Nakamoto [Nak08] (we will use linear-based and chain-based interchangeably). This type of structure can result in undesirable performance in terms of scalability, throughput, and confirmation time [WYCX20]. In addition, chain-based data structures can result in computation, storage and communication overheads as observed by [XLZ+21]. Moreover, [LCP+20] pointed out that a single chain structure can experience centralization concerns because powerful nodes have a higher chance of generating the next block. Another concern is high transaction fees because transactions are processed by powerful miners that require high resource consumption. Several graph-based structures, such as Directed Acyclic Graph or DAG-based blockchain, were introduced to address these bottlenecks. Literature improvements include developing DAG-based blockchains that can process or confirm transactions in parallel, requiring fewer communications, computations, and storage overhead [SLZ16, PMIH18, BKLMC20, ZHZB20]. Despite all the benefits of the DAG structure, a large-scale DAG-based blockchain, such as IoT scale, consumes higher computations than traditional linear-based blockchains [WHH+19].

3.2 Background

3.2.1 Blockchain

The oldest form of blockchain dates back to 1990 by Haber *et al.* [HS90] which describes how to cryptographically seal and time-stamp a digital document. The oldest

running form of blockchain, from 1995 - present, is by the New York Times [TC20] using the work presented by [HS90]. The term blockchain became more popular with the introduction of bitcoin in 2008 by an anonymous entity known as Satoshi Nakamoto [Nak08]. A blockchain system consists of a peer-to-peer (P2P) network where each node stores a copy of a distributed ledger (or distributed database) [HS21]. A blockchain consists of blocks that are chained together using hash values [WWC+19]. Each block can store committed digital interactions that can happen in the blockchain network [YMRS19]. Committed digital interactions can include the amount of digital assets sent from one account to another, temperature readings, logs, or any kind of data to be stored. The blockchain grows over time by appending valid blocks to the blockchain by special nodes called miners or forgers [ZXD+18, ASHN21]. Blockchains are also designed to achieve some of the following goals: First, eliminating the need to have a trusted third party. For example, blockchain does not require any intermediary or central authority to perform valid transactions, and nodes in blockchain systems can agree on the trustfulness of any block using consensus algorithms [DZZ19]. Additional goals include achieving user privacy by concealing users' private information while keeping records publicly available, and ensuring data integrity and immutability. There are three types of blockchains public (such as Nakamoto [Nak08] and Ethereum [W+14]), private, and consortium blockchains [XWS+17].

To achieve trust in a blockchain, the system uses consensus algorithms that allow nodes to agree on any block's validity and trustfulness before appending it into the blockchain [DZZ19]. Some of the most widely used consensus algorithms are proof of work (PoW) [B+02, Nak08], proof of stake (PoS) [BPS16], delegated proof of stake (DPoS) [Lar14], and practical Byzantine fault tolerance (PBFT) [CL+99]. For more details about consensus protocols, we refer to this paper [XZLH20]. In

(PoW), miners compete to solve a computationally-expensive mathematical puzzle. The more hashing power a node has, the more work it can do and the higher the chance to solve the puzzle, mine the next block, and receive the reward [WWC+19]. With blockchain systems that use PoS, the consensus algorithm requires miners or forgers to lock assets before being selected to be validators and forging any block. In general, forgers with higher locked assets (stake) are assumed to be more trustful and have higher chances to forge the next block [DKJG19]. This chapter applies PoS as a consensus algorithm since it does not require extensive computations, which is more suitable for lightweight IoT devices.

3.2.2 Merkle Tree

A Merkle tree, proposed by Merkle *et al.* [Mer89], is a balanced hash tree that stores hash values, see Fig. 3.1. Many research works utilize the Merkle tree architecture in distributed computing, such as bitcoin. Some of the Merkle tree's major applications include comparison and verification of data [BSHC18]. Every leaf node in the Merkle tree is labeled with a hash that is generated from its data content. Every non-leaf (parent) node is labeled with a hash that is generated by concatenating the hashes of its children. Hashing the tree works as a bottom-up approach starting from the leaf nodes up to the root node. In the end, a unique Merkle hash is assigned to the root node. This Merkle hash is used in blockchain systems and is stored in the block header. The final Merkle hash represents proof of the validity of all transactions within the block. One thing to note is that any modification to any transaction will result in a different Merkle root hash value [WWC+19, WHH+19, LYHK07].

Fig. 3.1 depicts a simple example of the Merkle tree. The structure is divided into two parts. The first part contains the Merkle tree, and the second part is the data

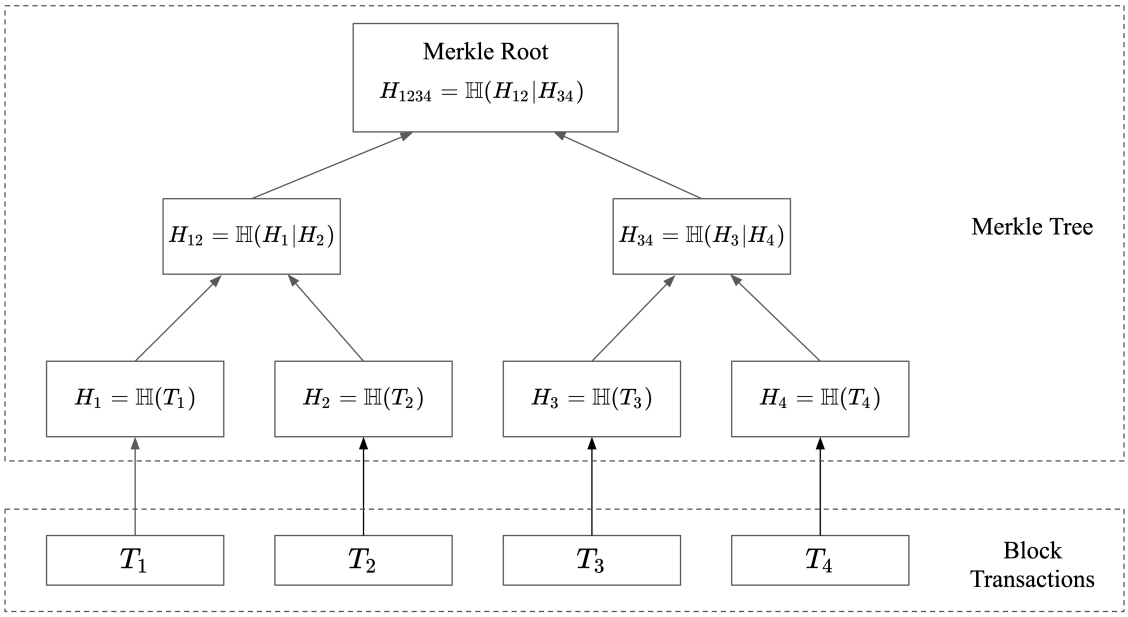

Figure 3.1: Merkle Tree Structure [BSHC18].

to be hashed, in our case, block transactions. For instance, the leaf node H_1 stores the hash value of transaction T_1, and the hash value is calculated using the hashing function $\mathbb{H}(T_1)$ using [Mer89], the same applies to other transactions. Moving up in the tree, every non-leaf node concatenates its children's hashes, hashes them, and stores a new hash value. In the example, the non-leaf node H_{12} concatenates H_1 and H_2 and assigns a new hash to H_{12} as follows, $H_{12} = \mathbb{H}(H_1|H_2)$. This process continues until the root node is reached, and the root node H_{1234} assigns the root hash to be $H_{1234} = \mathbb{H}(H_{12}|H_{34})$.

3.2.3 k-d Tree

A k-d tree, introduced in the 1970s by Bentley *et al.* [Ben75], is a multidimensional binary search tree. The k symbol indicates the number of dimensions, formally known as discriminators. The k-d tree takes sets of inputs or points and sorts them

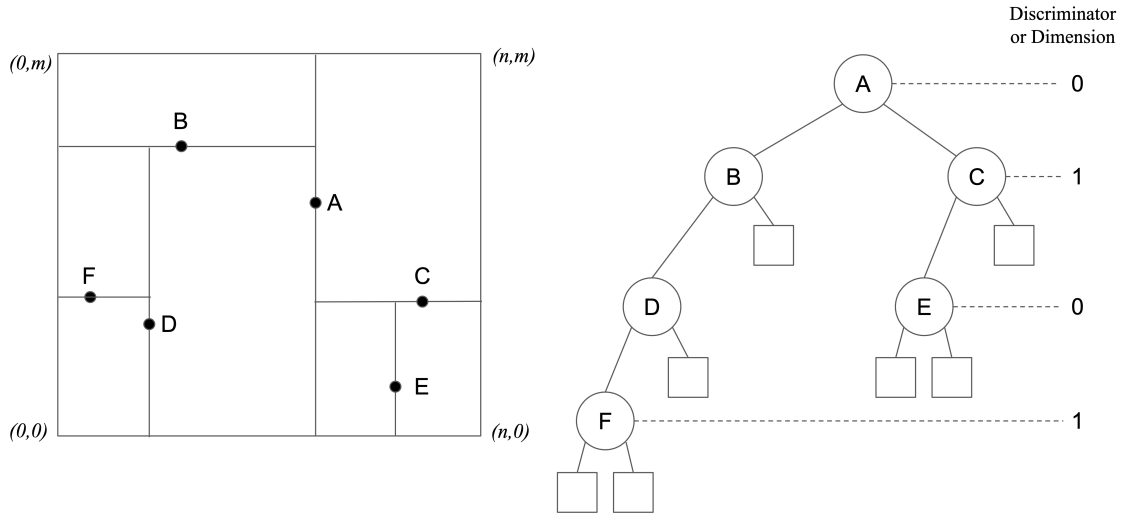

Figure 3.2: A k-d tree structure with 2-dimensions [Ben75].

in a k-dimensional space; see Fig. 3.2. The k-d tree is a useful and efficient data structure for cases like range search and nearest neighbor search [Moo90].

Fig. 3.2 shows a 2-d tree which represents a set of points $\{A, ..., F\}$ in a plane. In the tree, the circles represent the points; and the squares represent *null* leaf spaces that can accept the next input or node. At first, root node A splits the domain or points with a vertical line into two sub-domains (B and C). Both subdomains in this example have an approximately equal size number of points. The splitting process continues until no splitting is required. Nodes at discriminator 0 split a set of points with a vertical line while nodes at discriminator 1 split a set of points with a horizontal line [Ben75].

3.2.4 Merkle k-d Tree

A Merkle k-d tree consists of a k-d tree with a Merkle tree representation of the k-d tree as described in Sections 4.2.3 and 4.2.2. These are the main building blocks of the proposed model. The multidimensional blockchain is an immutable k-d tree, and

the Merkle tree is a modifiable representation of the multidimensional blockchain, more details in Section 3.4.

3.3 Preliminaries

This section presents an overview of the blockchain model, notations, assumptions, and proposed model.

The blockchain model uses Proof-of-Stake (PoS) for the consensus mechanism. Proof-of-Work (PoW) is considered a computationally expensive consensus algorithm; therefore, PoS seems more logical for mobile, wireless, and lightweight devices because it does not require a lot of computation to forge the next block. Instead of competing to forge the next block, a node (forger) is selected to forge the next block at fixed time interval fT (check Table 4.1 for notations). The selection is based on uniform random mining, and every node at the initialization has the same chance to mine the next block. The forger gathers all transactions, prepares an $mkdBlock$, adds and broadcasts the block to all peers in the network, in which every peer will verify the new block and add the block to the blockchain $\mathbb{T}$.

3.3.1 System Model and Assumptions

The proposed model consists of a set of IoT devices and a plane that is divided into smaller regions called cells. Each cell can contain a set of IoT devices and a local blockchain. A blockchain split happens when a group of nodes tries to split into two or more groups, each with its dedicated cell. A merge can happen when certain conditions occur: 1) when two or more local networks meet in a single cell. 2) when a local network gets access to a full node that stores the entire blockchain and is always connected to the Internet using, for example, RSU or edge devices [CLP+20].

Table 3.1: Notations

Symbol	**Meaning**
$\mathbb{T}$	Merkle k-d blockchain (MKDBC)
$block$	An arbitrary block
$mkdBlock$	A Merkle k-d block in a $\mathbb{T}$
$hash$	The hash of $block$ or $mkdBlock$
$mhash$	The Merkle hash of $mkdBlock$
tx	Transaction abbreviation
cT	Current time
gT	Genesis time
fT	Forge time interval
C	A cell or region
$\mathbb{C}$	Set of all cells
sC	spatial constraint
s	A sensor
S	A set of sensors
$\mathbb{S}$	Set of all sensors
G	local network
G_i^t	A local network of nodes in cell i at time t
$\mathbb{G}$	Set of all local networks
n	Total number of sensors
V_i^t	Set of vertices of local network G_i^t
$\mathbb{T}_i^t$	A local Merkle k-d blockchain in network G_i^t
$\mathbb{T}^{gT}$	MKD blockchain at genesis time gT
$cDim$	Current dimension
$tDim$	Total number of dimensions of $\mathbb{T}$, ranging from 0 to k

To ensure the authenticity and integrity of transactions in a graph-based blockchain, the block creator signs all transactions within the block as in [KJW+18]. Transactions contain sensor readings such as temperature readings. Another assumption is a single or multiple nodes can move from one cell to another. Our model can actively detect topology changes and adjust consensus accordingly even when the network is dealing with the network partitioning problem using [MVL+21]. Each local blockchain network can work independently without relying on other blockchain networks. Nodes are assumed to be mobile within a cell using Random Way-point

Mobility found in [HV07] and Reference Point Group Mobility (RPGM) for group trajectory based on [HGPC99]. Each node is capable of performing simple data aggregation tasks such as finding max, min, mean, etc. [PPJP12]. The proposed model does not require any additional powerful devices since nodes participating in the blockchain can perform all the necessary operations. In addition, the model utilizes a permissioned version of blockchain where a centralized authority controls who participates in the blockchain and assigns a public and private key for each node. Each node has the same genesis block. We also assume that nodes reveal their identities to each other using a privacy-preserving method such as [LLC+18]. The nodes achieve consensus using proof of stake (PoS) consensus algorithm as in [BPS16]. The consensus algorithm is set to have finality, which means the consensus protocol does not allow the presence of equally valid blocks or subchains. This is achievable using many approaches, such as applying three consensus phases before committing any request. The three phases are: pre-prepare, prepare, and commit as in [XFL+21, CL+99]. Another approach is the NEAR protocol [Ski20] which can achieve finality in one round of communication. And lastly, the work by Ethereum [Eth] to implement single-slot finality for Ethereum in around 16 seconds.

3.4 The Multidimensional or Merkle k-d Blockchain Framework

The Merkle k-d blockchain model (MKDBC) consists of two major components, as shown in Fig. 3.3. The first one is a multidimensional blockchain. Unlike traditional blockchain models where blocks are structured as a linear-chain, the MKDBC structures and sorts blocks in a way similar to a k-d tree [Ben75], see Section 4.2.2. The (genesis block, previous block) in MKDBC have the same analogy

as the (root, parent) in k-d trees, respectively. The block components of MKDBC are similar to any other traditional block except in the block header; we have an extra field that stores the block dimensions as a list of k values where k is the number of dimensions of the blockchain or $[Dim_0, Dim_1, ..., Dim_k]$.

The second component of MKDBC is a Merkle tree representation of the entire blockchain using the method designed by [Mer89], see Section 4.2.3. The primary goals of using the Merkle tree are to provide an extra layer of security and as a tool for fast comparison and verification of blocks; more details in Section 3.5. Each time a block is added to the multidimensional blockchain, the Merkle tree gets updated. Generating and updating the Merkle hash tree is an efficient process since it only

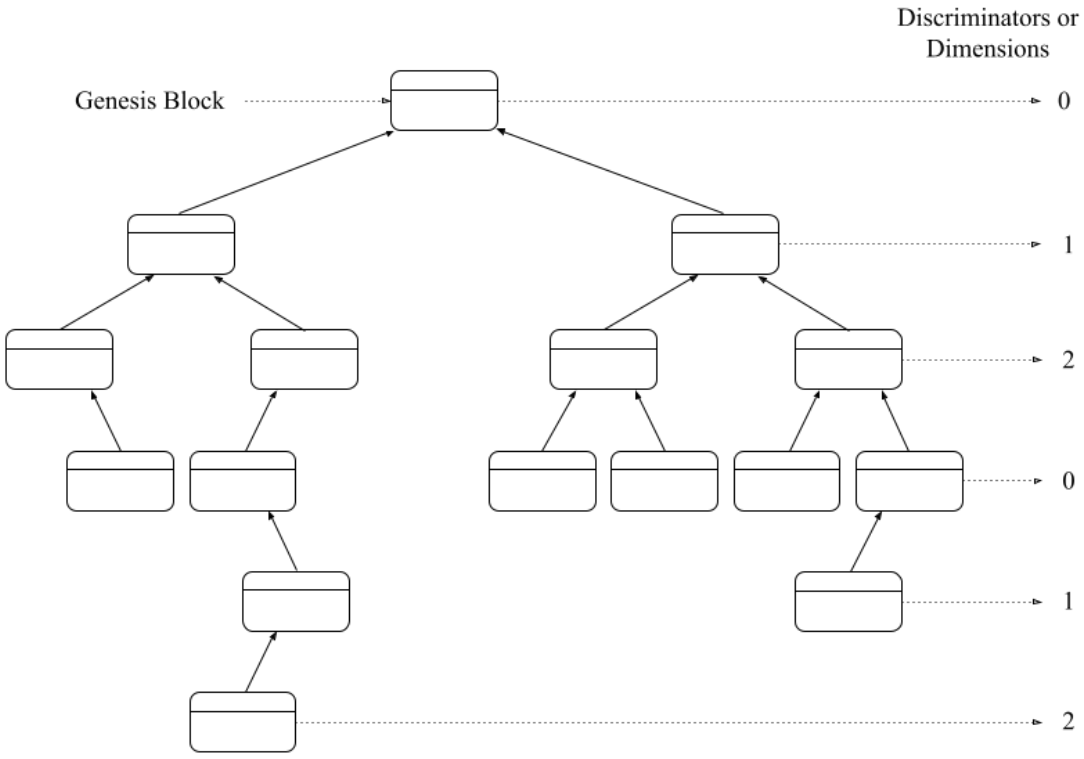

Figure 3.3: The multidimensional blockchain structure with 3 dimensions or $k = 3$, the dimensions are $0, 1, 2$.

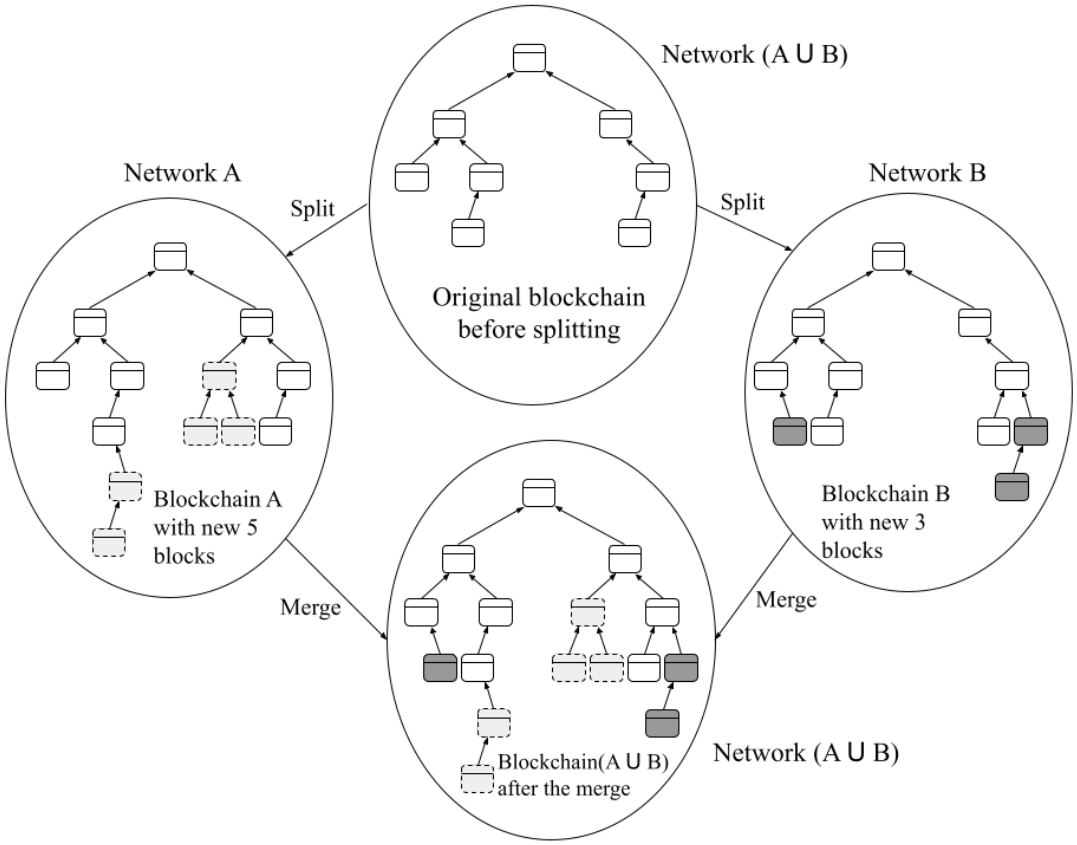

Figure 3.4: The process of split and merge in multidimensional blockchains.

needs a few modifications to the tree rather than searching and updating the entire tree [Mer89]. The Merkle tree is stored separately from the blockchain since the Merkle tree is not immutable and requires frequent updates each time a block is added to the blockchain.

Because nodes are highly mobile in mobile wireless networks, our model utilizes spatiotemporal data, such as coordinates and time, as discriminators or dimensions for the multidimensional blockchain. Although it's possible to adjust the blockchain dimensions based on the application, we decided to use coordinates and time dimensions for simplicity and the randomness of nodes' movements. The randomness generated from node movements turned out to be helpful in balancing the multidi-

mensional tree; more explanations about this are in Section. 3.5. The process of blockchain split and merge is shown in Fig. 3.4. The rest of this section will explain the algorithms of our model.

Algorithm 1, lines 1 - 5, start with initializing an empty Merkle k-d blockchain or $\mathbb{T}$. Later, the blockchain gets the first mkdBlock as a genesis block using $insertMkdBlock$ function, and the Merkle hash is updated using $updateMerkleHash$. Once the genesis block is added to the blockchain and the Merkle hash is updated, every node will download a copy of the blockchain at the genesis time or $\mathbb{T}^{gT}$. As mentioned in Subsection 3.3.1, for any node to join the network, the node is required to have a copy of the blockchain at the genesis time or $\mathbb{T}^{gT}$. This is possible because we are using a permissioned type of blockchain where there is a centralized entity controlling who joins the network. The reason why we require a node to have at least a copy of $\mathbb{T}^{gT}$ is to apply some balance to the blockchain tree as it grows in size, which will be explained further in Section 4.5.1. Finally, lines 7 - 11 show whenever it is time to forge a new block, a forger will be selected in each local network G_i^t to forge the next mkdBlock.

Algorithm 2 handles nodes' mobility management or when nodes are moving from one region to another. Lines 1 - 3 check if a set of nodes $S.C_{cur}$ is joining a another cell C_{new_cell}. First, the algorithm decides which blockchain to keep $\mathbb{T}_k$ and which blockchain to merge $\mathbb{T}_m$. There are many factors to choose from, but for simplicity, we chose to merge blockchains based on their sizes or to merge the smaller blockchain with the larger one. Secondly, after deciding which blockchain to keep and which one to merge, lines 4 - 8 select an aggregator node to handle the blockchain merge. Each block in $\mathbb{T}_m$ will be scanned and merged, if needed, into $\mathbb{T}_k$. The merge also utilizes the same method of adding blocks as in Algorithm 3. Lines 6 - 9 delete the blockchain from either the current or new cell, depending on which

Algorithm 1 MKDBC Management

Input: Current time: cT. Genesis time: gT. Forge time interval: fT. Set of all local networks: $\mathbb{G}$.

Output: Updated MKDBC Tree: $\mathbb{T}$

1: $\mathbb{T} \leftarrow \varnothing$
2: $insertMkdBlock(\mathbb{T}, mkdGenesisBlock)$ See Algorithm 3
3: $updateMerkelHash(\mathbb{T}, mkdGenesisBlock)$
4: **for** each $s \in \mathbb{S}$ **do**
5: $\quad download(\mathbb{T}^{gT})$
6: **end for**
7: **if** $(cT - gT) \mod fT == 0$ **then**
8: $\quad$ **for** each $G_i^t \in \mathbb{G}^t$ **do**
9: $\quad\quad Forger \leftarrow selectForger(V_i^t)$
10: $\quad\quad newMkdBlock \leftarrow creatMkdBlock(Forger)$
11: $\quad\quad insertMkdBlock(\mathbb{T}_i^t, newMkdBlock)$, See Algorithm 3
12: $\quad$ **end for**
13: **end if**

blockchain to merge and keep, update their cell/network information, and download the blockchain $\mathbb{T}_k$ after the merge is complete.

Algorithm 3 describes a recursive function *insertBlock* which inserts or forges *block* into the Merkle k-d blockchain $\mathbb{T}$ as an *mkdBlock*. The insertion method is a modified version from the k-d tree insertion found in Section 4.2.2 or [Ben75]. Algorithm 3 starts by taking multiple inputs which include the MKDBC tree: $\mathbb{T}$, the block to be forged: *block*, a parent block to compare with: *mkdBlock*, current dimension: *cDim*, and the total number of dimensions of $\mathbb{T}$: *tDim*. At the first iteration, lines 1 - 3 check if the first parent *mkdBlock* or *T.root* is *null*, however, the model assumes all nodes have the same genesis mkdBlock and hence the algorithm will skip lines 1 - 3. Next, at line 4, the algorithm checks whether *block* is a duplicate at the current dimension in $\mathbb{T}$ or not. Later, at line 6 - 9, the algorithm continues to recursively scan the tree cycling between each dimension until it hits a *null* or a space that accepts the next block. The algorithm hits a *null* leaf at line 1 where *block* will be forged and placed in $\mathbb{T}$ at the proper location.

Algorithm 2 MKDBC Management During Mobility

Input: Two MKD Blockchain Trees: $\mathbb{T}$. Set of local networks: G. Set of cells: C. Set of sensors: S

Output: Updated MKD Blockchain Tree: $\mathbb{T}$

if $S.C_{cur} \neq C_{new\_cell}$ **then**
  $\mathbb{T}_m \leftarrow blockchainToMerge(\mathbb{T}_{cur}, \mathbb{T}_{new\_cell})$
  $\mathbb{T}_k \leftarrow blockchainToKeep(\mathbb{T}_{cur}, \mathbb{T}_{new\_cell})$
  $Aggregator \leftarrow selectAggregator(V_m^t, V_k^t)$
  for each $mkdBlock_m \in \mathbb{T}_m^t$ **do**
    if $mkdBlock_m \notin \mathbb{T}_k^t$ **then**
      $newMkdBlock_m \leftarrow$
      $creatMkdBlock(Aggregator)$
      $insertMkdBlock(\mathbb{T}_k^t, newMkdBlock_m)$, See Algorithm 3
    end if
  end for
  $delete(\mathbb{T}_m^t)$ from either $S.C_{cur}$ or C_{new_cell}
  $S.C_{cur} \leftarrow C_{new\_cell}$
  $G_{new\_new}^t \leftarrow G_{new\_new}^t \cup G_{curr}^t$
  $V_m^t.download(\mathbb{T}_k^t)$ from peers V_k^t
end if

Algorithm 3 insertMkdBlock

Input: MKDBC Tree: $\mathbb{T}$. The block to be forged: $block$. A parent block to compare with: $mkdBlock$. Current dimension: $cDim$. Total number of dimension of $\mathbb{T}$: $tDim$

Output: MKDBC Tree with the new forged block: $\mathbb{T}$

if $mkdBlock == null$ **then**
  $mkdBlock \leftarrow forgeBlock(block)$
  $updateMerkleHash(\mathbb{T}, mkdBlock)$, See algorithm 4
else if $block == mkdBlock.data$ **then**
  return 'duplicate block'
else if $block[cDim] < mkdBlock[cDim]$ **then**
  $mkdBlock.left \leftarrow$
  $insertMkdBlock(\mathbb{T}, block, mkdBlock.left, (cDim + 1) \bmod tDim,\ tDim)$
else
  $mkdBlock.right \leftarrow$
  $insertMkdBlock(\mathbb{T}, block, mkdBlock.right, (cDim + 1) \bmod tDim,\ tDim)$
end if

Algorithm 4 presents a recursive function to update the Merkle hash of the entire MKDBC tree. The update function is called immediately after forging any

Algorithm 4 updateMerkleHash

Input: MKDBC Tree: $\mathbb{T}$, An arbitrary block: *block*
Output: Updated MKDBC Tree: $\mathbb{T}$

1: $parent \leftarrow getParentBlock(T, block)$, See Section 4.2.2
2: **if** $block == \mathbb{T}.root$ **then**
3: **return**
4: **else if** $block == \mathbb{T}.parent.left$ **then**
5: $\mathbb{T}.parent.mhash \leftarrow$
6: $hash(block.mhash|\mathbb{T}.parent.right.mhash)$, See Section 4.2.3
7: **return** $updateMerkleHash(\mathbb{T}, parent)$
8: **else**
9: $\mathbb{T}.parent.mhash \leftarrow$
10: $hash(\mathbb{T}.parent.left.mhash|block.mhash)$
11: **return** $updateMerkleHash(\mathbb{T}, parent)$
12: **end if**

new block. Calculating or updating the Merkle hash is an efficient process and does not require scanning or updating the entire tree but rather one branch of the tree [Mer89]. The update function takes the MKDBC tree $\mathbb{T}$ and an arbitrary *block* or *mkdBlock* in $\mathbb{T}$. At line 1, the algorithm starts with finding the *parent* block of *block* in $\mathbb{T}$. The method used to find the *parent* block is similar to the k-d search as in Section 4.2.2. Finding the *parent* block also gives access to the *parent*'s child blocks. Lines 2 - 11 check if *block* is the root of $\mathbb{T}$; if the condition is met, the function terminates since there is no need to update the Merkle hash of $\mathbb{T}$. Next, if *block* is not the root of $\mathbb{T}$, *block* will be compared to check if it is the right or left child of *parent* and update $T.parent.hash$ accordingly using the method in Section 4.2.3. In summary, the recursive function $updateMerkleHash(T, block)$ starts from the most recently forged block, and applies all the necessary Merkle hash updates to the root or genesis block.

3.5 Evaluation

3.5.1 Simulation Setup

For the simulation, we used 80 nodes on a grid with size $m \times n$. All nodes get a copy of the initial blockchain, which only includes the genesis block. Blocks are similar to any other traditional blockchain except for an extra field that stores the block dimensions; the dimensions used are $[x\_coordinate, y\_coordinate, time]$. The order of dimensions is critical, as we will see later. The main goal of ordering is to allow the multidimensional blockchain to grow as balanced as possible on both sides of the genesis block. To achieve some balance in the blockchain tree, we set the genesis block dimensions, in this case, the coordinates, as the mid-point of the grid or 50. Since nodes are highly mobile and travel randomly within the grid, future blocks can be added on both sides of the genesis block. In addition, to allow additional balance to the blockchain tree, we set *time* as the last in the dimension list $[x\_coordinate, y\_coordinate, time]$. This is crucial because time is an incremental value, and if we set time to be the first dimension, the blockchain will only grow on one side of the blockchain tree, in this case, the right side. If *time* is the first dimension, the genesis block will have a *time* dimension set to 0, and any future block will have *time* dimension > 0 and hence will be added to the right side of the blockchain tree.

For node mobility, we implemented two mobility models, Reference Point Group Mobility (RPGM) based on this work [HGPC99] and Random Waypoint Mobility [BHPC04]. In RPGM, the nodes within a group are uniformly distributed inside a circle, and the circle center represents the group center [Sic09]. In addition, the group center travels on a random trajectory, and all nodes move at a random velocity

ranging from 0 to 1. The code to generate these types of movements is publicly available on GitHub [Pan].

3.5.2 Experimental Results

Fig. 3.5 shows a summary of 835,000 blockchain snapshots captured throughout the simulation. The goal is to find the maximum chain length or height in the multidimensional blockchain and compare it with: 1) the traditional chain-based blockchain height such as Nakamoto [Nak08] and 2) a balanced k-d tree height [Ben75] (best case scenario). It's essential to consider the blockchain height because having longer branches results in more comparisons (i.e., more resource consumption) to do any operation, such as appending/merging/scanning blocks, in the multidimensional blockchain. First, we grouped the blockchain snapshots based on their blockchain sizes in an increment of 50 where group one includes all the blockchains with sizes $\geqslant 0$ and $\leqslant 49$ and so on. Then for each group, we calculated the maximum chain length or height per snapshot and found the total average of the maximum chain length of the group. Based on Fig.3.5, the average maximum chain height, of all MKDBC blockchains with a size around 8000 blocks, is roughly 55. And the maximum chain height, of a traditional blockchain with 8000 blocks, is 8000. Comparing the maximum height of MKDBC with the traditional chain-based blockchain, we can see that MKDBC can reduce the maximum chain length by more than 99.99%. This means, on average, it only scans less 0.01% of the total blocks to find the place to forge or merge the next block in MKDBC. This reduction allows the multidimensional blockchain to perform efficiently as the blockchain grows; the next Fig. 3.6 will demonstrate this. Some could argue that a smaller sub-chain length (leaf block to genesis block) can allow malicious nodes to redo the work of that particular

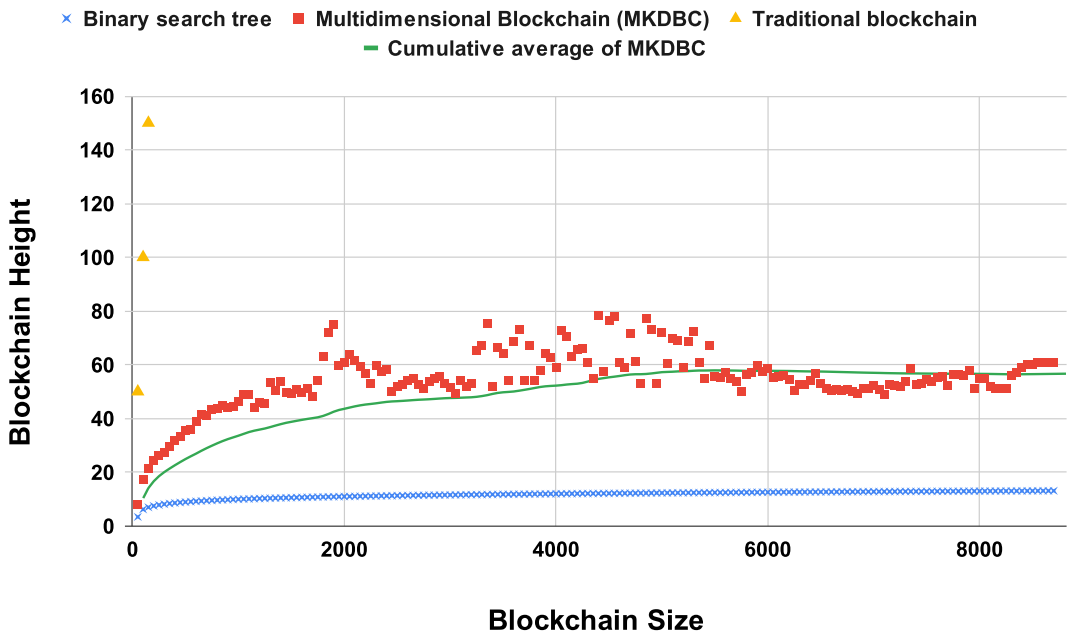

Figure 3.5: The longest chain height or length, averaged per blockchain size for the multidimensional blockchain (MKDBC), compared with a traditional chain-based blockchain [Nak08], and a balanced binary tree [Ben75]. Lastly, the cumulative average of MKDBC longest height is represented by the line

sub-chain. This is one of the reasons we use a Merkle tree representation of the multidimensional blockchain, which is to verify whether a sub-chain is valid relative to the entire blockchain or has been compromised. Any modification to any sub-chain will result in a different Merkle hash for the entire multidimensional blockchain. An important observation is that traditional blockchain systems' height or chain length grows linearly, and the chart can only cover a height up to 150 for a blockchain with 150 blocks. However, the MKDBC height stays relatively similar to a logarithmic growth, showing signs of efficiency and scalability as the model grows.

The following evaluation chart or Fig. 3.6 shows the time the MKDBC takes to forge or merge a block in a unit of time (normalized), where 1 is the maximum recorded time. The dots represent the forging time of more than 20,000 random

blocks from random blockchains of different sizes. Unlike traditional blockchain, where blocks are appended to the end of the chain with almost no time, our model requires searching the multidimensional blockchain for the right location to append the next block. Luckily, the MKDBC utilizes a binary search operation to find the right place to forge the next block. The details on how to search and forge the next block are explained in the recursive Algorithm 3. As shown in Fig. 3.6, the time to search and forge blocks shows an efficient and scalable growth.

To demonstrate another practical use of multidimensional blockchain for mobile IoT with network partitioning problems, Fig. 3.6 shows a cluster of blocks, between blockchains with sizes ranging between 1 - 1000, with higher than normal forging time. The reason is that those blocks were forged in blockchains that stayed relatively stationary in particular areas (we will call them stationary blockchains). Since we're using coordinates as the first and the second dimensions, the blockchain continued to grow largely on one side of the multidimensional blockchain, resulting in longer than normal sub-chains or branches (More details in Section 4.5.1). The higher the branches, the more time it takes to forge or merge blocks since it involves more traversing and comparisons. Interestingly, the forging time declines as those stationary blockchains start to grow, move, and merge with other blockchains. The transition from being stationary blockchains to becoming more active or mobile can help in adding additional balancing to the blockchain tree and produce blockchains with shorter branches or heights, resulting in improving the forging time.

Next, we evaluate the relationship between the maximum depth and the forge time in a 1-d blockchain using two figures. Firstly, in Fig. 3.7, we present the maximum blockchain depth based on four different dimensions. In this experiment, we first select the block ID as the block dimension, an incremental number starting from 0. Since this ID is an incremental value, any block added to the blockchain will be

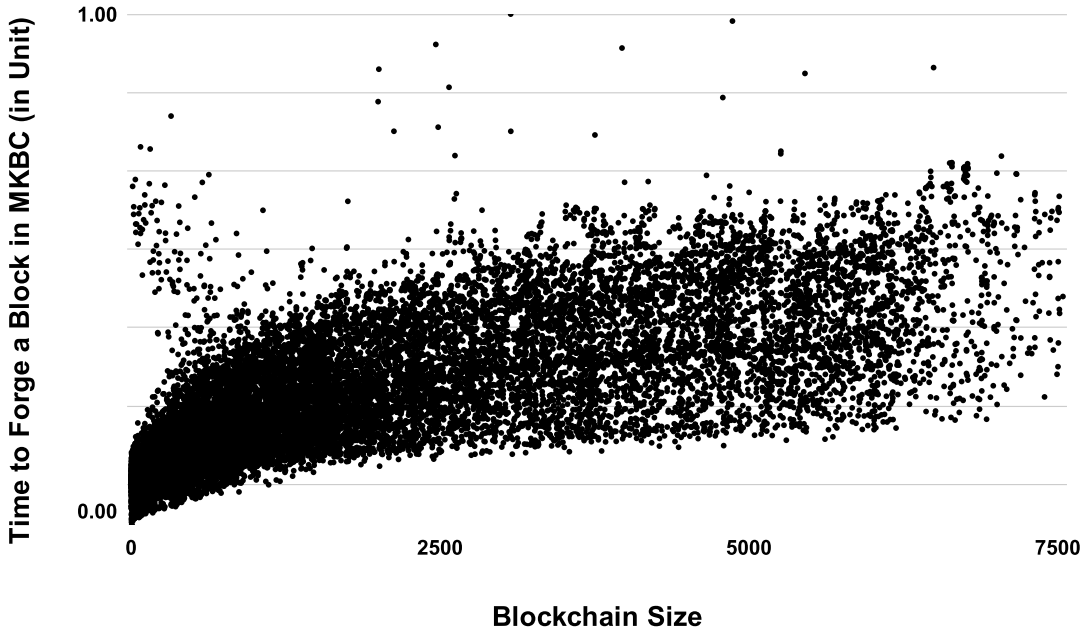

Figure 3.6: Time to forge a block in the multidimensional blockchain (MKDBC), normalized to a unit of time where 1 is the maximum recorded time.

added on the right side of the blockchain tree, resulting in one single branch similar to a single-chain blockchain. Next, we select the miner ID as the block dimension, which can be any unique ID, such as a public key. Since a miner can forge multiple blocks, many blocks can have the same block dimension, resulting in the blockchain growing with higher depth. We can see in the figure that selecting miner ID as a block dimension can result in a blockchain depth of 300 when the blockchain size is around 160k. We can introduce randomness to help reduce the blockchain depth, as it can help provide more unique values for block dimensions. In our third selection, we concatenate the miner ID and region ID to be the next type of block dimension. We can see that we can minimize the blockchain depth to around 40 instead of 300 when the blockchain size is around 160k by only concatenating a random value to the miner ID, resulting in around 86% reduction. In the last experiment, we set

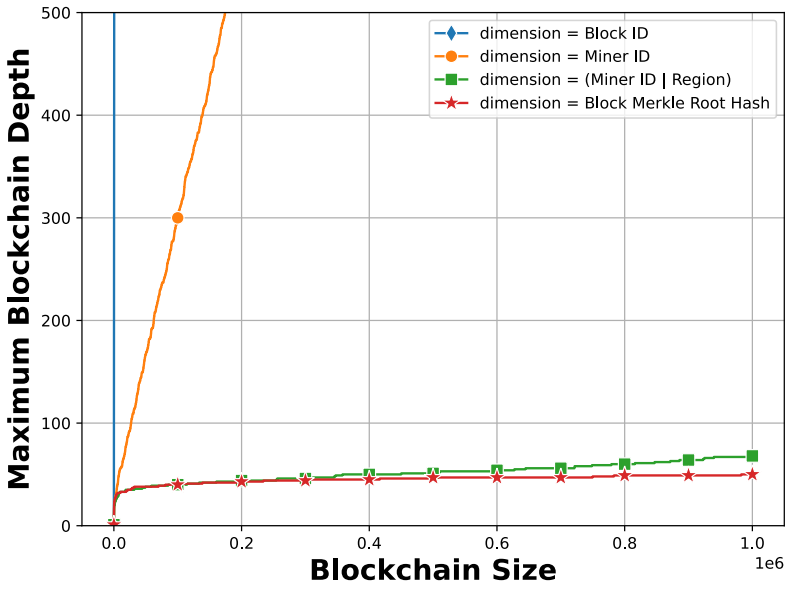

Figure 3.7: The maximum blockchain depth of a multidimensional blockchain based on four types of block dimensions.

the block dimension to be a fixed and unique value that is unlikely to repeat, such as the block Merkle root hash, which is the Merkle hash of all transactions within the block. Since this value is unlikely not repeat, blocks are forged and positioned randomly in the blockchain tree, allowing the multidimensional blockchain tree to grow with the least maximum depth or around 48 maximum depth at a blockchain size of one million.

Secondly, in Fig. 3.8, we investigate the effect of the maximum blockchain depth on the performance of the multidimensional blockchain in terms of block forging time. We can see that blockchain that contains block IDs as the block dimensions can perform the worst. This is because to append any block, we would need to traverse all blockchain blocks because all blocks are forged in a single branch or chain. The result of high forging time is that the forging process requires too many block comparisons to find the next available position to append the next block. For

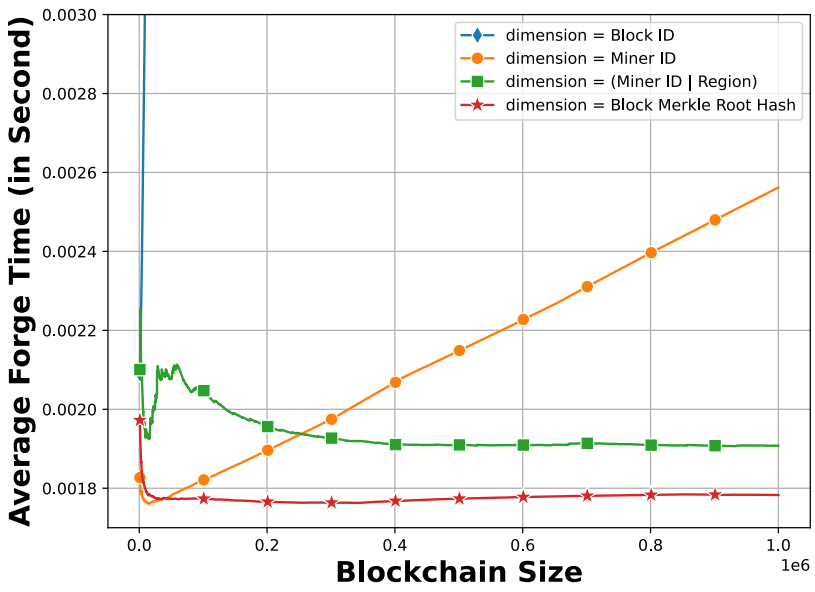

Figure 3.8: The average forging time of a multidimensional blockchain based on four types of block dimensions.

miner ID as a blockchain dimension, it shows linear growth because miners continue to add more blocks with the same dimension over time, which can result in linear growth of both the maximum blockchain depth and average forging time. As shown in the figure, we can significantly reduce these linear growths by concatenating the region ID to the miner ID. Lastly, the last dimension, block Merkle root hash, can outperform all other dimensions achieving the lowest blockchain depth and the best average forging time due to its ability to make the blockchain grow with smaller branches which requires few comparisons. The takeaway from this experiment is that selecting a random and unique value, such as block Merkle root hash, as a blockchain dimension can significantly reduce the maximum blockchain depth and improve blockchain performance.

3.6 Conclusion

This chapter presents a multidimensional graph-based blockchain model that eliminates the need for having fixed and powerful peripherals for mobile IoT. The model can allow a blockchain system to function even when dealing with the network partitioning problem while maintaining the blockchain's security and privacy. Experimental results show the multidimensional blockchain can achieve: efficiency by having to scan only a few blocks (fewer comparisons) for any blockchain operations, and scalability by achieving performance similar to logarithmic growth. In future work, we plan to study the effect of dimensionalities on the performance of the multidimensional blockchain. We will investigate the impact by experimenting with different numbers and kinds of dimensions.

CHAPTER 4

AN EFFICIENT APPROACH FOR MERGING MULTIDIMENSIONAL BLOCKCHAINS IN MOBILE IOT

Graph-based or multidimensional blockchains have been proposed to improve the scalability and efficiency of existing blockchain applications. However, when implemented in mobile Internet of Things (mIoT) networks, these blockchain systems can frequently split and merge and cause the merging algorithm to process a large number of similar blocks (a block is similar or identical when it exists once in multiple blockchains). The presence of similar blocks hinders the merging process, as it consumes time and computational resources to scan, validate, and potentially merge similar blocks. This chapter presents an efficient approach for merging graph-based or multidimensional blockchains in mIoT networks by avoiding similar blocks and effectively detecting and merging new blocks that were created after the split. Our proposed merging algorithm employs depth-first search and Merkle tree techniques to minimize the time and computational resources spent on identical blocks. Finally, we evaluate the performance of our method in highly mobile networks and demonstrate that it can execute the merge with a more than 72% reduction in time in comparison to merging algorithms without block similarity handling.

The following are the main contributions of this chapter:

- We propose an efficient approach for merging multidimensional blockchains that reduces the amount of processing of identical blocks during the merging process.
- Employing Depth-First Search and Merkle Tree Hash, we design an efficient recursive algorithm that merges two multidimensional blockchains.

- Our simulation results show that the proposed approach outperforms the traditional merging approach in terms of scanned and skipped blocks and the reduction in merge time. Moreover, the proposed approach can reduce the merging time by more than 72% compared to traditional merging approaches.

The chapter is organized as follows: Section 4.1 presents an introduction. Section 4.2 presents an overview and background knowledge related to this chapter. Section 4.3 describes the preliminaries and system assumptions. Section 4.4 shows the proposed model. In Section 4.5, we discuss the simulation setup and experimental results. Finally, in Section 4.6, we presents the conclusion of this chapter.

4.1 Introduction

In recent years, blockchain technology has generated much interest due to its ability to provide secure and private groundbreaking solutions for various problems in many domains, such as finance, healthcare, distributed systems, voting, industry, and real estate. Another essential field that benefits from blockchain and is related to this chapter is the mobile Internet of Things (mIoT). These mIoT systems can adopt blockchain technology and benefit, for example, from decentralization, security, privacy, and quality of service (QoS) [Nov18, YZX+22, NPDS21]. But the benefits of blockchain in mIoT can also come at a cost. For instance, traditional blockchain systems are designed in a chain-based structure similar to the blockchain structure in Nakamoto [Nak08]. This type of structure can result in undesirable performance in terms of scalability, throughput, and confirmation time which make the design unsuitable for mIoT systems [WYCX20].

Fortunately, many blockchain structures were proposed to address the bottlenecks of chain-based blockchain, with a majority of them as graph-based blockchain

structures such as Directed Acyclic Graph or DAG-based blockchains [YDXJ20, LMAA19, CZC21, LCP+20, GYZ+19], and multidimensional blockchain [ZMP+22]. We will use graph-based blockchains and multidimensional blockchains interchangeably. Besides the performance improvements of graph-based blockchains over chain-based blockchains in mIoT, graph-based blockchains also have better support for network mobility and partition tolerance when dealing with the split and merge problem [WYY+22, LWQ+19]. However, when graph-based blockchains in mIoT encounter frequent split and merge, multidimensional blockchains can experience block similarity (i.e., two blockchains have a large number of similar blocks). The amount of block similarity between two blockchains can affect the merge efficiency and cause the merging algorithm to spend extra time and resources on scanning, validating, or potentially merging identical blocks, branches, or subtrees.

The current graph-based blockchains, including multidimensional blockchains, may not fit within the resource capacity of individual devices, leading to potential slower performance and scalability issues. Therefore, a practical merging approach must be developed to address this issue and ensure that the resource capacity of mIoT devices is not exceeded.

An example scenario where the proposed approach can be applied is in smart cities, where numerous IoT devices are used to manage various aspects such as traffic, pollution, or energy consumption. Since these devices generate vast amounts of data that require secure and efficient blockchain systems, a multidimensional blockchain structure can provide the necessary decentralization and security. However, the frequent split and merging of blockchains can negatively impact the merging efficiency and result in the unnecessary processing of identical blocks [ZMP+22, WYY+22, LWQ+19]. To the best of our knowledge, this is the first work that addresses improving the merge efficiency of graph-based blockchains by minimizing the time

and computational resources wasted on processing identical blocks between multiple blockchains.

4.2 Overview

4.2.1 Blockchain

The earliest form of blockchain was proposed by [HS90] in 1990 and describes how to secure and timestamp digital documents. The oldest running blockchain, which has been in operation since 1995, was implemented by the New York Times based on Harber's work [TC20]. The term "blockchain" became more widely known with the introduction of Bitcoin in 2008 by Nakamoto [Nak08]. A blockchain system consists of a network of nodes that each store a copy of a distributed ledger. The ledger is organized into blocks that are linked together using hash values, and each block can store committed digital interactions that take place on the network, such as the transfer of digital assets, temperature readings, or any other data [HS21, WWC$^+$19, YMRS19]. The blockchain grows over time by appending valid blocks to the blockchain by special nodes called miners or forgers. Blockchains are also designed to achieve some of the following goals: eliminating the need to have a trusted third party, achieving user privacy by concealing users' private information while keeping records publicly available, and ensuring data integrity and immutability [ZXD$^+$18, ASHN21].

4.2.2 k-d Tree

A k-d tree is an efficient data structure, or a multidimensional binary search tree [Ben75]. The k symbol indicates the number of dimensions known as discriminators.

The k-d tree takes sets of inputs or points from a k-d space and sorts them in the tree in an order based on the current dimension. Some applications of the k-d tree are range search and nearest neighbor search [Moo90].

4.2.3 Merkle Tree

The Merkle tree was proposed by [Mer89], which is a balanced hash tree that stores hash values representing some data. Many research works utilize the Merkle tree architecture in distributed computing, such as bitcoin and Ethereum. The general use of the Merkle tree is for data comparison and verification [BSHC18]. The hashing process of the tree is a bottom-up approach starting from the leaf nodes up to the root node. A unique Merkle hash (known as Merkle root) is assigned to the root node or block. In blockchain applications, the final Merkle hash represents proof of the validity of all transactions within the block. Finally, any modification to any transaction will result in a different Merkle root hash value[WWC$^+$19, WHH$^+$19, LYHK07]

4.2.4 Multidimensional Blockchains

A multidimensional blockchain (MKDBC) [ZMP$^+$22] is a type of graph-based blockchain that structures blocks as k-dimensional trees using [Ben75]. The MKDBC is composed of two main components. The first is an immutable k-dimensional tree structure for structuring and storing blocks, and the second is a mutable Merkle tree representing all the blocks within the blockchain. The MKDBC also employs multi-layer indexing to sort and organize blocks, which allows for faster scanning, forging, and validation of blocks, resulting in increased efficiency and scalability of the overall blockchain system. The MKDBC is also a partition-tolerance blockchain that

supports the splits and merges of blockchains in dynamic networks. The block components of MKDBC resemble those of traditional blocks but with an additional field in the block header that stores the block dimensions as a list of k values, represented as $[Dim_0, Dim_1, ..., Dim_k]$. In mobile wireless networks, the MKDBC model can utilize spatiotemporal data, such as coordinates and time, as discriminators or dimensions for the multidimensional blockchain. However, the blockchain dimensions can be adjusted based on any application need. It's recommended to be cautious when selecting block dimensions, as it appears to be in [ZMP+22], and selecting some block dimensions that can introduce randomness (such as coordinates) to avoid linear growth of the tree. Choosing only incremental block dimensions (such as time or block number) can result in the blockchain tree to grow linearly and can result in linear growth for any operations.

4.3 Preliminaries

4.3.1 System Model and Assumptions

The proposed model assumes a blockchain can split into two or multiple blockchains, and multiple blockchains can merge again to form a single blockchain network. A blockchain split happens when a group of nodes tries to split into two or more groups, each with its dedicated cell. A merge can happen when certain conditions occur: 1) when two or more local networks meet in a single cell. 2) when a local network gets access to a full node that stores the entire blockchain and is always connected to the Internet using, for example, RSU or edge devices [CLP+20]. For simplicity, we merge the smaller size blockchain with the larger one. The split and merge can have significant implications for network participants and cause confusion

and uncertainty within the blockchain network, such as conflicting transactions and the authenticity and integrity of transactions. However, there are many ways to mitigate these implications. To help resolve conflicting transactions when merging parts of the blockchains, we can apply methods, such as [LSZ15, SLZ16, SZ18], more specifically, we have adopted a way that allow multiple nodes to contribute simultaneously when creating a block to resolve conflicting transactions. Moreover, to ensure the authenticity and integrity of transactions, we have adopted methods similar to [KJW+18, CLP+20] by encapsulating additional data into each block that refer to multiple blocks. Nodes are assumed to be mobile within a cell using Random Way-point Mobility found in [HV07], and Reference Point Group Mobility (RPGM) for group trajectory based on [HGPC99]. Each node is capable of performing simple data aggregation tasks such as finding max, min, mean, etc. [PPJP12]. Transactions contain sensor readings such as temperature readings. The proposed model does not require any additional powerful devices since nodes participating in the blockchain can perform all the necessary operations. The model also uses a permissioned version of the blockchain where a centralized authority controls who participates in the blockchain and assigns a public and private key for each node, and each node has the same genesis block. We also assume that nodes reveal their identities to each other using a privacy-preserving method such as [LLC+18]. The consensus algorithm is set to have finality, which means the consensus protocol does not allow the presence of equally valid blocks or subchains. This is achievable using many approaches, such as applying three consensus phases before committing any request. The three phases are: pre-prepare, prepare, and commit as in [XFL+21, CL+99]. Another approach is the NEAR protocol [Ski20], which can achieve finality in one round of communication. And lastly, the work by Ethereum [Eth] to implement single-slot finality for Ethereum in around 16 seconds.

Table 4.1: Table of Symbols

Symbol	Meaning
$\mathbb{T}$	Multidimensional blockchain
$block$	An arbitrary block
v	A neighboring block
$mkdBlock$	A Merkle k-d block in a $\mathbb{T}$
$hash$	The hash of $block$ or $mkdBlock$
$mhash$	The Merkle hash of $mkdBlock$
C	A cell or region
$\mathbb{C}$	Set of all cells
s	A sensor
S	A set of sensors
$\mathbb{S}$	Set of all sensors
G	local network
$\mathbb{G}$	Set of all local networks
n	Total number of sensors
V_i	Set of sensors of local network G_i
$\mathbb{T}_i$	A Multidimensional blockchain in network G_i

4.4 An Efficient Approach for Merging Multidimensional Blockchains

In this section, we detail our proposed merging algorithm for multidimensional blockchains. Before we start defining the merging algorithm, we would like to point out to the reader that the merging algorithm presented in this chapter is an extension of this paper [ZMP+22]. Still, this approach can be deployed to other graph-based blockchains since they can support DFS traversal or building Merkle Trees. The previous work requires all nodes to process a full copy of the blockchain during the merging. However, the proposed merging algorithm can reduce the amount of processing of blocks when merging two blockchains. Later, in this section, we present a general description of our model and two algorithms for the efficient merge with details to explain them.

As stated earlier, the efficiency of the merging process is affected by the presence of identical blocks since nodes need to spend time and resources on scanning or validating similar blocks. To improve the efficiency of the merging process of [ZMP+22] or any merging algorithm that process a full copy, we present a new merging algorithm with two components. Our merging algorithm extends, not replaces, two existing technologies, which are Depth-First Search by Tarjan[Tar72] and Merkle tree or hash by [Mer89]. Depth-first search is a technique we adopt to improve the efficiency of the merging process in the presence of identical blocks. The merging algorithm process a blockchain copy to be merged starting at the genesis or root block and exploring as far as possible along each branch before backtracking and exploring other branches. The second technology or component is the Merkle tree or hash, which enables nodes to quickly and efficiently identify identical blocks by comparing the hashes of different blocks. Finally, our algorithm combines the two technologies to help avoid scanning identical blocks or subtrees or going to higher depths in the blockchain and reduce the time and resources spent on processing similar information.

We demonstrate our merging algorithm with an example as depicted in Fig. 4.1. Consider a graph-based or multidimensional blockchain named $BC1$ that splits and creates another copy $BC2$ due to mobility or the loss of connectivity between miners. Both $BC1$ and $BC2$ can continue to forge new blocks as shown in the picture after the split. It is also possible that both blockchains can have identical sub-trees of blocks (such as block B, C, D, E, F). In addition, new blocks (such as H, I, J) can be forged into other leaves or branches. An advantage of multidimensional blockchain is that the blockchain can grow with minimal height and numerous leaves [ZMP+22]. For instance, a fully balanced multidimensional blockchain with height h can have up to 2^h leaves which can accepts new blocks [CLRS09]. Considering the numerous

amount of leaves and the frequent spit and merge, new blocks can be forged under leaf blocks which can promote the existence of many identical sub-trees. Later, when both $BC1$ and $BC2$ attempt to merge, we run our merging algorithm that utilizes both Depth-First Search (DFS) and Merkle Tree. For simplicity, $BC2$ was merged with $BC1$ because $BC2$ was smaller in size before the merge. The merging algorithm starts scanning $BC2$ from the root or genesis block using DFS and checks the Merkle Hash of the block. Later, the algorithm checks if the same block exists in $BC1$. Since the block exists, we can see that both $BC1$ and $BC2$ have new distinct blocks which can cause the root or genesis blocks in both blockchains to have different Merkle hashes. This indicate no need to forge the root block of BC but also the root of the sub-tree under the root is not an identical sub-tree. Later, the merging algorithm check the next neighboring block or block B. We can see that block B exists in both $BC1$ and $BC2$ but also the sub-trees under block B are identical. This indicate there is no need to check any other block under block B and can be safely skipped and move to the next neighboring block. We continue to run the merging algorithm and forge new blocks (such as block J) and skip blocks under the root of identical sub-trees (such as block C, D, E, F). Further details about the merging algorithm are available in the subsequent sub-section.

Next, we present two algorithms 5 and 6, which demonstrate two multidimensional blockchains merging with each other. Algorithm 5 gets initiated when one cell merges with another cell. The algorithm starts by taking two multidimensional blockchains and deciding what blockchain to keep and merge. As mentioned in the System Model and Assumptions or subsection 4.3.1, we merge the smaller blockchain with the larger one for simplicity. In line 4, we initialize a root variable to be the blockchain's genesis block of $\mathbb{T}_m$. Later, we run a recursive depth-first merge starting from the genesis block of $\mathbb{T}_m$ by calling algorithm 6. Once the merge is complete, we

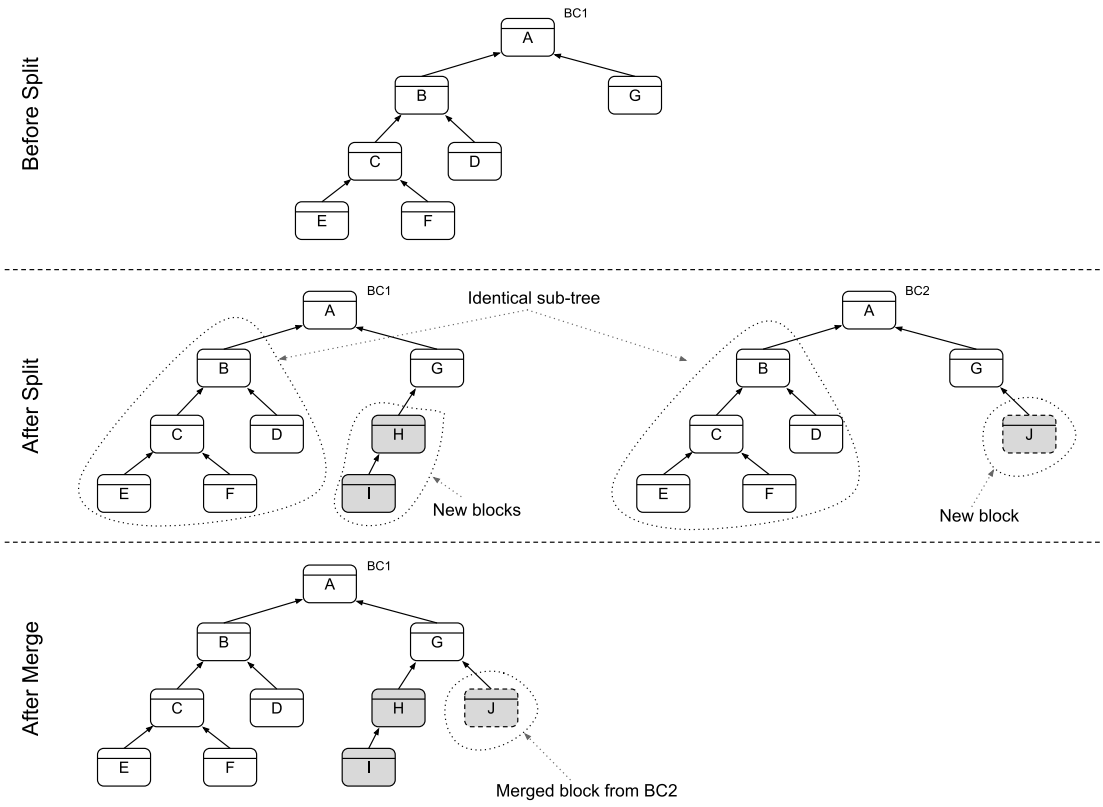

Figure 4.1: An example illustrating the split and merge process of multidimensional blockchains and the identification of identical sub-trees.

remove the blockchain from the nodes that hold a copy of $\mathbb{T}_m$, update their network topology or status, and download $\mathbb{T}_k$.

The algorithm 6 is a recursive depth-first merge that utilizes our blockchain model's second component (Merkle tree). The algorithm takes three inputs, including two multidimensional blockchains and an arbitrary block. We start scanning $\mathbb{T}_m$ from the genesis block and check if the block status is visited. Since it's the genesis at the beginning, we skip to line 4 and mark the block as visited. Next in line 5 - 7, we check if the block is not in $\mathbb{T}_k$; if the condition is valid, we merge the block to $\mathbb{T}_k$ using the insertion method found in [ZMP+22]; otherwise, we skip to the next part. Later in line 8 - 14, we explore every neighboring block v. For each v in $\mathbb{T}_m$,

Algorithm 5 Blockchains Merge Management

Input: Two Blockchain Trees: $\mathbb{T}_{cur}, \mathbb{T}_{new\_cell}$. Set of local networks: G. Set of cells: C. Set of sensors: S

Output: Merged Blockchain Tree: $\mathbb{T}$

1: **if** $S.C_{cur} \neq C_{new\_cell}$ **then**
2: $\mathbb{T}_m \leftarrow blockchainToMerge(\mathbb{T}_{cur}, \mathbb{T}_{new\_cell})$
3: $\mathbb{T}_k \leftarrow blockchainToKeep(\mathbb{T}_{cur}, \mathbb{T}_{new\_cell})$
4: $root \leftarrow \mathbb{T}_m.root$
5: $DFSMerge(\mathbb{T}_m, \mathbb{T}_k, root)$, see algorithm 6
6: $delete(\mathbb{T}_m)$ from either $S.C_{cur}$ or C_{new_cell}
7: $S.C_{cur} \leftarrow C_{new\_cell}$
8: $G_{new\_new} \leftarrow G_{new\_new} \cup G_{curr}$
9: $V_m.download(\mathbb{T}_k)$ from peers V_k
10: **end if**

we check if the block is in $\mathbb{T}_k$ and if v in both $\mathbb{T}_m$ and $\mathbb{T}_k$ and have identical Merkle hashes. Equal Merkle hashes indicate that the merging algorithm hit similar blocks or subtrees. Next, the algorithm avoids or minimizes scanning identical blocks with higher depth by marking similar blocks as *visited*. We mark them as *visited* so the algorithm stops exploring any block with higher depth in the future, which is possible as in line 1 - 3. After marking identical blocks as *visited*, the algorithm moves to the next neighboring block and recursively runs the DFSMerge function, line 12, on that neighboring block.

4.5 Performance Analysis

4.5.1 Simulation Setup

For the simulation, we used 80 nodes on a grid with size $m \times n$. Blocks are similar to any other traditional blockchain except for an extra field that stores the block dimensions; the dimensions used are $[x\_coordinate, y\_coordinate, time]$.

Algorithm 6 DFSMerge (Depth-First Search Merge)

Input: Two Blockchain Trees: $\mathbb{T}_m, \mathbb{T}_k$, An arbitrary block: $\mathbb{T}_m.block$
Output: Updated $\mathbb{T}_k$

if visited $[\mathbb{T}_m.block] == True$ **then**
  return
end if
$visited[\mathbb{T}_m.block] \leftarrow True$
if $\mathbb{T}_m.block \notin \mathbb{T}_k$ **then**
  $insertMkdBlock(\mathbb{T}_k, \mathbb{T}_m.block)$, insertion algorithm is available in [ZMP+22]
end if
for each $v \in \mathbb{T}_m.block.neighbors$: **do**
  if $v \in \mathbb{T}_k$ and $\mathbb{T}_k.v.merkleHash == \mathbb{T}_m.v.merkleHash$: **then**
    visited[v] ← True
  else
    DFSMerge$(\mathbb{T}_m, \mathbb{T}_k, v)$
  end if
end for

For node mobility, we implemented two mobility models, Reference Point Group Mobility (RPGM) based on this work [HGPC99] and Random Waypoint Mobility [BHPC04]. In RPGM, the nodes within a group are uniformly distributed inside a circle, and the circle center represents the group center [Sic09]. In addition, the group center travels on a random trajectory, and all nodes move at a random velocity ranging from 0 to 1. The code to generate these types of movements is publicly available on GitHub [Pan].

4.5.2 Experimental Results

In this section, we examine our merging algorithm that deals with similar blocks (improved merge) with the merging algorithms of graph-based or multidimensional blockchains (normal merge), such as [ZMP+22], which require processing a full blockchain copy and do not deal with block similarity.

The experimental results for the normal merge model (Fig. 4.2) show that, as the blockchain size increases, the cumulative average percentage of blocks remains constant for both scanned and skipped blocks at 100% and 0%, respectively. This indicates that all blocks are scanned and considered for a potential merge in normal merge algorithms.

In contrast, Fig. 4.3 shows that the improved merge has a much lower percentage of scanned blocks at only 9% instead of 100% in the normal merge, indicating that it is more efficient in selecting the blocks to be scanned. Similarly, the improved merge has a higher percentage of skipped blocks at 91% instead of 0% in the normal merge. Scanning fewer blocks and skipping a large portion could indicate that the improved merge can effectively identify and skip irrelevant or identical blocks. This could lead to significant performance improvements in blockchain merging time, as we will see in the subsequent figures.

Final notes on Fig. 4.2 and 4.3 are: A) for both models, the percentage of merged blocks goes up to approximately 30% and decreases to around 2%. B) In Fig. 4.3, it is worth mentioning that there is a difference between the number of scanned and merged blocks. There are two reasons for this case. First, if both the blockchains to be merged have similar leaf blocks, these blocks will be scanned but not merged because they are the first identical blocks to be found. The second reason if both blockchains have identical subtrees. Our model scans blocks until it founds identical subtrees; the root blocks of the subtrees will be scanned since they are the first identical blocs; however, any identical blocks with higher depth than the subtree root will be skipped.

Next, the merging time for blockchains using normal and improved merge is presented in Fig. 4.4. In this study, we propose an improved merge that incorporates additional computations over normal merge, such as marking blocks as visited

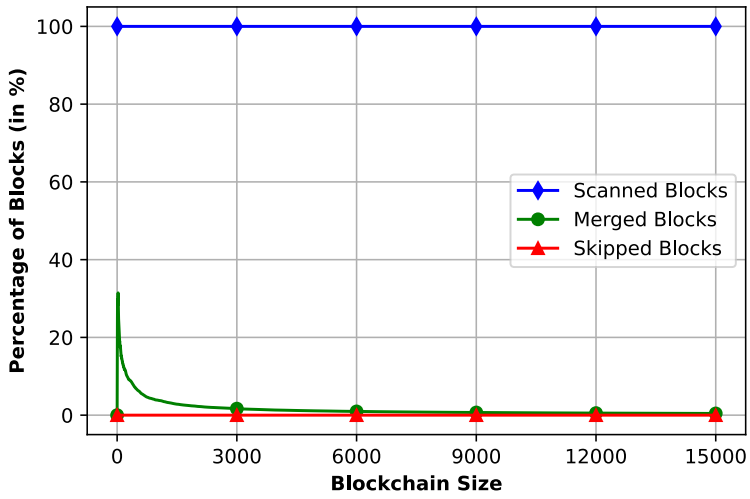

Figure 4.2: The percentage of scanned, merged and skipped blocks for (Normal Merge) using [ZMP+22].

and performing Merkle hash comparisons. Despite these added computations, our proposed algorithm can still achieve better merging performance and demonstrate logarithmic growth, indicating its efficiency and scalability as the blockchain grows in size. However, to fully understand the efficiency of our proposed model, the following figure will further explain the efficiency of our proposed model compared to the normal merge algorithm.

In order to evaluate the efficiency of our proposed model, we compare its performance to the normal merge method described in [ZMP+22]. We measure the reduction in merge time or the amount of time saved between the two approaches to quantify the improvement. Specifically, we use the two lines in Fig. 4.4 to measure the reduction by calculating the change or difference between the two lines. As illustrated in Fig. 4.5, the time reduction is initially noisy when the blockchains have a small number of blocks. This is expected as there are relatively few similar blocks between the two blockchains at this stage. However, as the blockchain grows, the

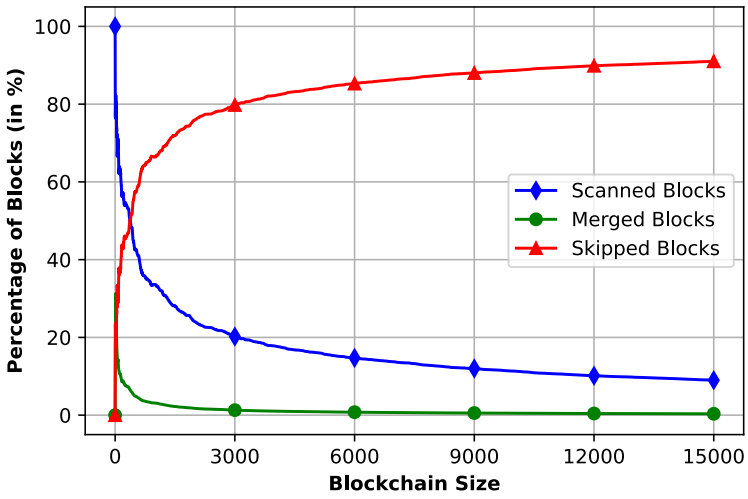

Figure 4.3: The percentage of scanned, merged and skipped blocks for our model (Improved Merge) or algorithm 5 and 6.

number of similar blocks increases, leading to a corresponding reduction in merge time for our proposed model. Finally, at a blockchain size of 15000 blocks, our model completes the merging process on average with more than 72% reduction in time compared to the normal merge method. This improvement is achieved by detecting and avoiding the need to scan and validate similar blocks, thereby minimizing wasting time and computational resources on those identical blocks.

4.6 Conclusion

In conclusion, identical blocks can affect the merging efficiency of blockchains by requiring the merging algorithm to scan and validate identical blocks between multiple blockchains. By adopting depth-first search and Merkle tree, blockchain systems can improve the efficiency of their merging and reduce the impact of identical blocks by allowing nodes to minimize the time and computational resources spent on processing identical blocks.

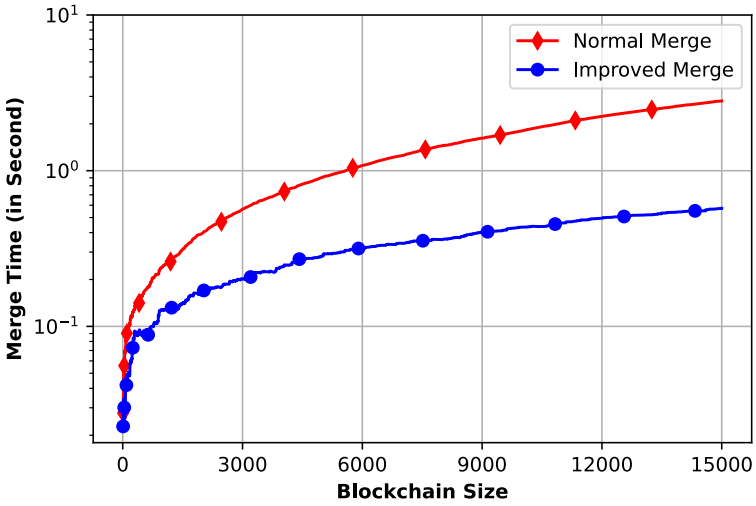

Figure 4.4: The blockchain merge time of Normal Merge and Improved Merge.

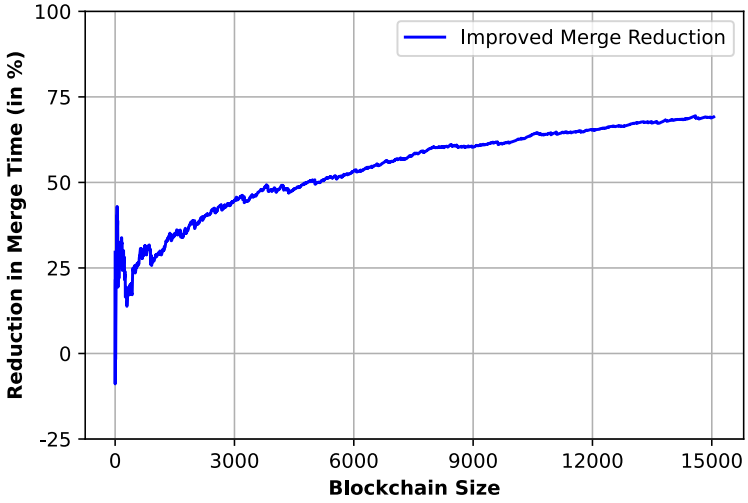

Figure 4.5: The reduction in merge time of our model compared to [ZMP+22], measured by taking the difference between Normal and Improved merge lines in Fig. 4.4.

CHAPTER 5

CSBSO: COLLECTIVE SIGNING-BASED BLOCKCHAIN STORAGE OPTIMIZATION FOR MOBILE INTERNET OF THINGS

In the context of mobile Internet of Things (IoT), the efficiency and scalability of blockchain storage emerge as a significant challenge, particularly in resource-constrained systems. This chapter proposes a Collective Signing-Based Blockchain Storage Optimization (CSBSO) model that aims to reduce the storage overhead using an efficient blockchain structure. The proposed CSBSO extends the existing Collective Signing (CoSi) protocol and utilizes a multidimensional blockchain structure to reduce the storage overhead and manage/retrieve blocks efficiently. Our approach for storage optimization encompasses identifying and pruning the most irrelevant blocks based on CoSi protocol. We evaluate CSBSO using two real-world datasets, the Ethereum Classic Blockchain dataset and the Facebook users dataset. We also evaluate CSBSO under various conditions and compare it with the state-of-the-art storage optimization model. Our results demonstrate that the proposed model CSBSO can significantly outperform the state-of-the-art storage optimization model, achieving approximately 92% storage optimization. Finally, the results showcase that CoSi-based storage optimization can effectively reduce the blockchain storage overhead in resource-constrained applications.

The main contributions of this chapter are as follows:

- We propose CSBSO, a Collective Signing-Based Blockchain Storage Optimization model for Mobile IoT, to reduce the storage costs of IoT nodes.
- Based on the CoSi protocol, we design a storage optimization algorithm that can identify the most relevant/irrelevant blocks and prune the irrelevant blocks.

- By employing two real-world datasets (i.e., Ethereum Classic Blockchain and the Facebook Users), our simulation results show that the proposed CSBSO can effectively optimize storage and outperform existing state-of-the-art techniques and save up to 92% of storage space in mobile IoT environments.

The rest of the chapter is organized as follows. Section 5.1 presents an introduction to the work. Section 5.2 outlines the background. Section 5.3 presents the proposed CSBSO model. Section 5.4 evaluates the proposed blockchain storage method. Finally, Section 5.5 concludes the chapter.

5.1 Introduction

Blockchain is a distinct distributed ledger technology that has found widespread application in a wide range of fields. Due to the increase in data on the blockchain and the addition of blockchain characteristics, the traditional blockchain's full chain copy storage requirement causes a blockchain storage scalability problem. Blockchain systems' scalability issues include low throughput, excessive data load, and ineffective query engines. All of these difficulties are closely tied to data management [WLC+22]. Existing approaches prioritize reducing storage pressure on blockchain nodes over data availability, resulting in a long average response time for users attempting to access the blockchain. The massive storage demand limits the involvement of devices with limited resources, such as sensors and other low-power IoT devices [DZZ19]. The existing solutions to the issue of blockchain storage scalability are classified into two categories off-chain and on-chain schemes. The Off-chain schemes store the blockchain ledger data in existing storage systems, such as blockchain storage solutions based on Distributed Hash Table (DHT), Inter Planetary File System (IPFS), and cloud-based blockchain storage solutions. The On-

chain techniques, such as collaborative storage solutions and light node solutions, primarily leverage compression or sharding technologies to limit the amount of data stored on the blockchain node.

More recent studies, such as [SPR21, CLY19, MZFZ19, YLZ21, Wan22], proposed data compression methods to reduce the storage overhead of blockchain models. These methods involve designing compression algorithms that can reduce either the size of transactions or blocks. However, these compression-based schemes offer storage optimization benefits but also present limitations. For example, these compression algorithms have a high complexity due to the need for a unified data-sharing service between the nodes, which cannot be suitable for large-scale and resource-constrained devices such as IoT. Additionally, data compression approaches can lead to increased latency, as highlighted in [AMTS+22]. The traditional blockchain's topological scale will significantly expand as a result of the Mobile IoT's fast data generation, which will continue to increase the cost of nodes' storage.

The existing storage optimization methods, like group-based storage [XHC18], off-chain storage [LWLX19, SJ20, DKJG19, MRR+20, XZYS20, SS19, LYW+21, LCZ+22, XFRZ20, NTG+22, ZLCD18, PB19, HGW+22], and pruning [FHBS19, Bita, LCL+20], have demonstrated their potential in lessening the storage demands for blockchain systems. However, these solutions come with some drawbacks, such as depending on a limited number of trusted nodes to maintain a full copy, assuming high connectivity and availability to cloud, edge or fog, weakened decentralization, and weakened security as a result of pruning important blocks [WLC+22, LCZ+22, XFRZ20, NTG+22, ZLCD18, HGW+22]. As a consequence, it is crucial to investigate new storage optimization strategies that can tackle these issues while accommodating the needs of IoT and mobile applications.

To address the abovementioned issues, we have developed a dynamic Collective Signing-Based Blockchain Storage Optimization (CSBSO) model that seeks to dynamically reduce storage overhead for mobile IoT while retaining the ability to adjust storage requirements. Our proposed model takes advantage of a multidimensional blockchain structure to cut storage expenses and effectively manage and retrieve blocks using an efficient data structure. Our model also utilizes Collective Signing (CoSi) protocol [STV+16] to minimize the storage overhead. Our approach for storage optimization is targeted at identifying and eliminating irrelevant blocks of varying degrees of irrelevance ranging from the most to least relevant blocks using the CoSi protocol. Extensive experiments demonstrate that our proposed technique can significantly reduce the storage cost of IoT nodes compared to existing storage optimization techniques.

5.2 Background

5.2.1 Internet of Things (IoT)

The Internet of Things encompasses a vast network of interconnected physical and heterogeneous devices embedded with sensors, software, and connectivity features that facilitate data collection and exchange across various sectors, including smart homes, healthcare, transportation, agriculture, and manufacturing [KS18]. By integrating blockchain technology into IoT systems, we can address issues such as security, privacy, and scalability through blockchain in a decentralized, secure, and transparent nature. However, traditional blockchain models may be unsuitable for IoT environments due to their substantial storage and computational demands, which can be challenging for resource-limited IoT devices [DKJG17]. As a result, nu-

merous lightweight blockchain approaches have been proposed to minimize storage overhead, enhance scalability, and optimize resource utilization while maintaining security, privacy, and decentralization in IoT applications [YDXJ20, XHC18, LCZ+22, WWZC21].

5.2.2 Blockchain

Blockchain technology has emerged as an innovative approach to store, share, and verify digital transactions [HS21, WWC+19, YMRS19]. It functions like a digital ledger that employs sophisticated cryptographic techniques to ensure that transactions are secure and reliable. Unlike traditional systems, blockchain does not require intermediaries to validate transactions, leading to increased efficiency, cost-effectiveness, and security [DZZ19]. The earliest form of blockchain was proposed by Haber *et al.* [HS90] in 1990 and describes how to secure and timestamp digital documents. The oldest running blockchain, which has been in operation since 1995, was implemented by the New York Times based on Haber's work [TC20]. The term "blockchain" became more widely known with the introduction of Bitcoin in 2008 by Nakamoto [Nak08]. A blockchain system comprises a network of decentralized machines that work collaboratively to ensure that transactions are stored accurately and securely [ASHN21, ZXD+18]. To achieve trust in a blockchain, the system uses consensus algorithms that allow nodes to agree on any block's validity and trustfulness before appending it to the blockchain [DZZ19]. Some of the most widely used consensus algorithms are proof of work (PoW) [B+02, Nak08], proof of stake (PoS) [BPS16, DKJG19], delegated proof of stake (DPoS) [Lar14], and practical Byzantine fault tolerance (PBFT) [CL+99]. There are three types of blockchains: public (such as Nakamoto [Nak08] and Ethereum [W+14]), private, and consortium blockchains

[XWS+17]. Blockchain technology has numerous applications, from financial management to medical record sharing and supply chain tracking [TC20].

5.2.3 Collective Signing

Collective Signing (CoSi) [STV+16] is a protocol that ensures an authority or leader's statement (*e.g.* transaction or block) is valid and signed publicly by a decentralized group of witness cosigners. This collective signing occurs in four distinct phases and gets initiated by the leader; the phases are announcement, commitment, challenge, and response. The main goal of implementing this protocol is to generate a statement signature that can be verified by anyone who participated in the collective signing process. CoSi can be implemented with blockchain applications to provide a more robust commitment of blocks. Several blockchain applications have already adopted the CoSi protocol, including [KKJG+16, FHB+22], other built Practical Byzantine Fault Tolerance (PBFT) consensus atop CoSi protocol such as [STV+16]. The Collective Signing prototype is open-source and can be found on GitHub [Ded]. For more details about the CoSi protocol, we refer the reader to this paper [STV+16].

5.2.4 Storage Optimization

The adoption of blockchain technology in IoT applications to securely store and manage data has become very popular in the past few decades. Nonetheless, traditional blockchain models mandate all nodes to store a full copy of the data at each node, posing some substantial challenges concerning scalability, storage, and performance for IoT devices. Addressing the issues is paramount, prompting many researchers to propose various innovative approaches for storage optimization in blockchain for IoT. One of the storage optimization solutions is on and off-chain

storage, where the complete blockchain data is stored in cloud, edge, or distributed data storage systems with the goal of removing the storage burden on IoT nodes. Another solution is proposing data compression algorithms that can help reduce the block size, precisely the data payload within the block. Additionally, strategies like pruning older or the least interested blocks have been considered to solve the storage optimization problem in IoT [AMTS+22].

5.3 Collective Signing-Based Blockchain Storage Optimization for Mobile IoT

The multidimensional blockchain was designed to enable a fast and efficient blockchain data structure that can retain, manage and retrieve blocks, making it ideal for large and complex blockchain applications. However, the efficiency and scalability of blockchain storage can be a significant challenge, particularly in resource-constrained systems such as mobile IoT.

This section presents our model, which aims to design an efficient and scalable storage optimization for blockchain systems. Our model utilizes existing technologies, such as a binary or k$-d$ tree, to provide a fast and efficient blockchain structure for managing and retrieving blocks. Using an efficient blockchain structure, our model can reduce the blockchain storage overhead for resource-constrained applications, such as mobile IoT. Our approach for storage optimization includes identifying and pruning the most irrelevant blocks. The proposed model assumes nodes are operating in highly mobile environments. We expect some nodes to move out of communication areas and be disconnected from the internet for some time. Therefore, our model is designed to deal with these scenarios and manage the blockchain

storage overhead for one or multiple groups of miners when they split or merge in highly mobile environments.

For consensus, the model adopts Practical Byzantine Fault Tolerance (PBFT) consensus protocol [CL+99] with the ability to collectively sing a block by all group members or witnesses as in [KKJG+16, STV+16]. In addition to utilizing multidimensional blockchains as a fundamental blockchain structure, our blockchain storage optimization solution extends, rather than replaces, Collective Signing protocol [STV+16] to effectively manage and optimize the blockchain storage.

The multidimensional blockchain organizes blocks based on block dimensions. Our blockchain model uses only one dimension, which contains information derived from the CoSi protocol and spatial data. The reason for selecting a dimension based on CoSi protocol is we can mark and identify every blockchain split and merge, allowing for seamless retrieval and management of the blockchain data [FHB+22].

Our block dimension structure consists of two primary components: The first is a list of all miners' public keys per CoSi group since each signature verifier knows the public keys of the leader and witness cosigners [KKJG+16]. The public keys list is then sorted to prevent generating different hashes (from different permutations) for the same list or set. Once the public keys list is sorted, a fixed-size SHA256 hash value (CoSi group hash) is generated for the list of miners' public keys using [Dan15]. The second component is a value that has some randomness characteristics, such as location information related to the CoSi group, which can be location coordinates or region IDs. Including randomness within the block dimension is primarily to minimize the maximum blockchain depth, which can positively impact the blockchain performance (more details presented in the evaluation section or 5.4). Finally, our model combines the (CoSi group hash) and (location information) into one string and utilizes it as the block dimension. The CoSi group hash allows each group to

append all blocks under one subtree in the blockchain, making it easier to manage and retrieve blocks, while location information helps in minimizing the blockchain's maximum depth and prevent the blockchain depth from growing linearly, which could compromise the performance. Although employing two dimensions (or $2-d$ blockchain) for our block structure is possible, it would not enable marking each split or merge and cause each group to append their blocks under multiple subtrees, making it more challenging to manage and retrieve blocks.

5.3.1 The Proposed Solution

In this subsection, we explain the dynamic behavior of CoSi groups in a mobile blockchain environment using a specific example and Figure 5.1. We consider a scenario involving two CoSi groups, each having its own blockchain copy. Initially, $CoSi\_groupA$ and $CoSi\_groupB$ have six and seven miners unique to their group and had not been part of any other groups. These miners can collectively sign blocks within their groups. Suppose a miner from $CoSi\_groupB$ joins $CoSi\_groupA$ or Blockchain Network A. Due to the miner's movement, this leads to the formation of $CoSi\_groupA.1$ and $CoSi\_groupB.1$. Using the miner's blockchain copy, new and unique blocks from Blockchain Network B will be merged into Blockchain Network A. Later, the miner then can participate and sign new blocks along with $CoSi\_groupA.1$.

Suppose the same miner that moved Blockchain Network B to A moves back to Blockchain Network B. This will lead to forming $CoSi\_groupA$ and $CoSi\_groupB$ again.

On the $CoSi\_groupA$ side, if the CoSi group decides to reduce the storage, they can run a pruning algorithm (Algorithm 8) that can traverse all the blocks and

find both relevant and irrelevant blocks and prune the irrelevant ones based on the similarity score (using Eq. 5.1) between the current $CoSi\_groupA$ and CoSi group that signed each block. Besides the original blocks or white blocks, we have three grey blocks, both the solid and dotted grey blocks will return a similarity score of 1 (most relevant block) because all members of $CoSi\_groupA$ participated in signing these blocks. However, the dashed grey block was signed by $CoSi\_groupB$ which will return 0 similarity score (irrelevant block) because no member within $CoSi\_groupA$ has participated in signing that block. Hence the block will be pruned, and only the block header and block hash will be kept.

Similarly, on the $CoSi\_groupB$ side, the miner that moved back to Blockchain Network B will have a copy of Blockchain Network A.1. All new and unique block will be merged into Blockchain Network B, which include the solid and dashed grey blocks. If the CoSi group decides to reduce the storage, they can run the same pruning algorithm. We can see that the solid grey block will be pruned because it has a similarity score of 0 or no member in $CoSi\_groupB$ has participated in signing that block. However, the dotted grey block has one member from $CoSi\_groupB$ that signed that block and will return a similarity score of $\frac{1}{7}$ using Eq. 5.1 for both $CoSi\_groupA$.1 and $CoSi\_groupB$. We can keep this block because it could contain information about a member of $CoSi\_groupB$ that was part of $CoSi\_groupA$.1 and mark the block as somehow relevant. In general, blocks that have a similarity score of 1, between 0 and 1, and 0 can be marked as most relevant, somehow relevant, and irrelevant blocks, receptively. In this example, we only prune blocks with a similarity score of 0, but in the evaluation, we use a pruning threshold ranging from 0 and 1 and present its effect in reducing the storage. Finally, The details regarding pruning blocks are available later in this section or Algorithm 8.

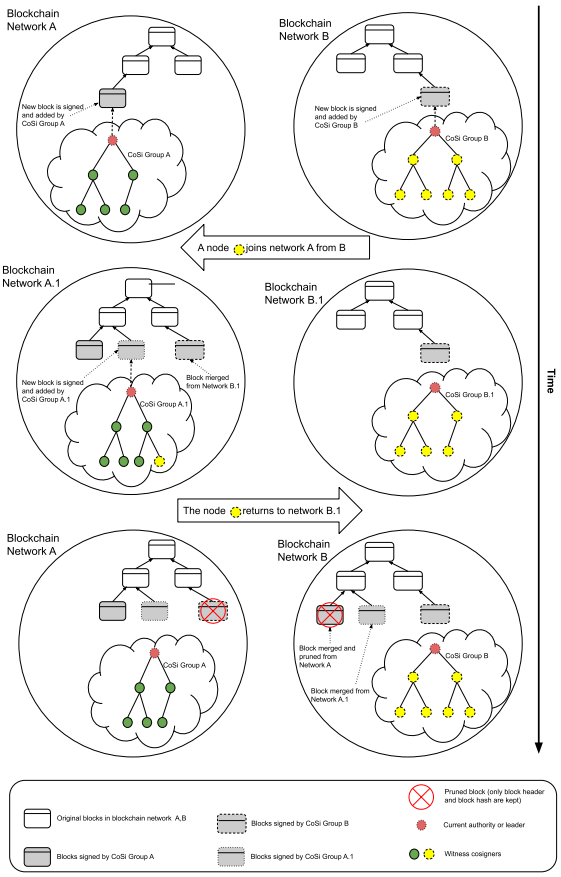

Figure 5.1: CoSi architecture and example when a new miner joins a new CoSi group.

$$CoSi\_similarity = \frac{|CoSi\_group1 \cap CoSi\_group2|}{\min(|CoSi\_group1|, |CoSi\_group2|)} \tag{5.1}$$

The $CoSi\_similarity$ equation (Eq. 5.1) derived from Jaccard similarity coefficient/index [RV96] calculates the similarity score between two CoSi groups, $CoSi\_group1$ and $CoSi\_group2$. It does so by dividing the number of common public keys between the groups (denoted by $|CoSi\_group1 \cap CoSi\_group2|$) by the minimum size of the two groups, i.e., the smaller number of public keys in either $CoSi\_group1$ or $CoSi\_group2$ ($\min(|CoSi\_group1|, |CoSi\_group2|)$). The resulting $CoSi\_similarity$

score reflects the degree of similarity between the two CoSi groups, ranging from 0 (no similarity) to 1 (identical CoSi groups).

Algorithm 7 outlines the process of adding a new block to the blockchain using a binary search approach. The algorithm takes two inputs, a new block denoted as $newBlock$, and the current blockchain denoted as $\mathbb{T}_{cur}$. If the current blockchain has no block, the new block is set as the blockchain's genesis block at line 3. Otherwise, the algorithm iterates through the blocks or nodes to find the correct position to insert the new block. To achieve this, the algorithm utilizes a binary search approach that compares the new and current blocks' dimensions. Specifically, the algorithm determines the axis on which the current node and new block differ and compares the corresponding dimension values. For example, in line 9, if the dimension value of the new block is smaller, the algorithm proceeds to the left child block of the current node, and if the value is equal or larger (line 16), the algorithm proceeds to the right child or block. This process of dimensions comparison continues until the algorithm reaches a node with no child block. Once the algorithm finds the right place to insert the new block, the algorithm proceeds to perform a collective signing by all miners within the current CoSi group. In the end, the new block will be appended to the blockchain.

The following algorithm 8, denoted as Prune Blockchain, is proposed to optimize the storage efficiency of a blockchain network. The main purpose of this algorithm is to identify blocks that do not significantly contribute to the current group of nodes and prune blocks to reduce the amount of data stored in the blockchain within the nodes. The algorithm takes as input the Genesis block, a list of CoSi groups, and a similarity threshold.

The algorithm starts by calling the "travers" function located at line 1 which takes a block as input. The algorithm recursively traverses the blockchain starting

Algorithm 7 Add Block

Input: New Block: $newBlock$. Current Blockchain: $\mathbb{T}_{cur}$. CoSi group: $CoSi\_group$.

Output: Updated blockchain

if $\mathbb{T}_{cur}$ is empty **then**
  $Collectively\_sign(newBlock, leader, witnesess)$
  $\mathbb{T}_{cur}.root \leftarrow newBlock$
  return
end if
$current\_node \leftarrow \mathbb{T}_{cur}.root$
while $current\_node$ is not None **do**
  $axis \leftarrow current\_node.depth$ mod $len(current\_node.dim)$
  if $dim[axis] < current\_node.dim[axis]$ **then**
    if $current\_node.left$ is None **then**
      $Collectively\_sign(newBlock, leader, witnesess)$
      $current\_node.left \leftarrow$ newBlock
      return
    end if
    $current\_node \leftarrow current\_node.left$
  else
    if $current\_node.right$ is None **then**
      $Collectively\_sign(newBlock, leader, witnesess)$
      $current\_node.right \leftarrow$ newBlock
      return
    end if
    $current\_node \leftarrow current\_node.right$
  end if
end while

from the genesis block. Later, the algorithm checks whether the currently traversed block is empty. If empty, the function terminates; otherwise, it calculates the similarity score between the current CoSi group and the block's CoSi group using the CoSi similarity function in algorithm 9. The block is pruned if the similarity score is less than or equal to the threshold. The algorithm then traverses the blockchain recursively, traversing the left and right children of the current block. Once it has traversed all the children of the current block, the algorithm moves to the next block until it has traversed the entire blockchain and terminates.

Algorithm 8 Prune Blockchain

Input: Genesis block: $\mathbb{T}.genesis$. List of CoSi group: $current\_CoSi\_group$. Similarity threshold: $threshold$.

Output: Pruned blockchain

```
1: function traverse(block):
2: if block is null then
3:    return
4: end if
5: CoSi_similarity_score ← CoSi_similarity (current_CoSi_group, block.CoSi_group)
6: if CoSi_similarity_score < threshold then
7:    prun(block)
8: end if
9: traverse (block.left)
10: traverse (block.right)
11: end function
12: traverse (T.genesis)
```

Algorithm 9, called CoSi Similarity, aims to calculate a similarity score using Jaccard similarity coefficient or index [RV96] between two sets or two CoSi groups. The input to the algorithm consists of two sets: the set of public keys for the current CoSi group ($CoSi\_group1$) and the set of public keys that signed a specific block ($CoSi\_group2$). The algorithm first finds the common public keys between the two sets to calculate the similarity score. It then computes the ratio of the length of the common public keys to the minimum length of the two input sets. This ratio determines the similarity score, which determines the degree of relevance of the current block to the current CoSi group. Overall, the algorithm takes the input of these two sets and outputs a similarity score between 0 and 1 that is used in algorithm 8.

Algorithm 9 CoSi Similarity

Input: Two sets: $CoSi\_group1, CoSi\_group2$

Output: Similarity score: $CoSi\_similarity$

1: common_public_keys $\leftarrow$ CoSi_group1 $\cap$ CoSi_group2

2: CoSi_similarity $\leftarrow \frac{\text{length(common\_public\_keys)}}{\text{min(length(CoSi\_group1),length(CoSi\_group2))}}$ [RV96]

3: **return** CoSi_similarity

5.4 Evaluation

In this section, we present our system assumptions, an evaluation of our model (collective signing-based storage optimization) under multiple conditions and compare it with the state-of-art model (community interest-based storage optimization) such as [YDXJ20] using two real-world datasets [Big] to present blockchain storage using the Ethereum Classic, and [LM12] to present community interests based on real Facebook users. Through the utilization of these datasets, we aim to show the effectiveness of our proposed model in comparison to existing approaches. We aim to show the effectiveness of our proposed model compared to existing approaches. We used 400 nodes on a grid with size $m \times n$ for the simulation. All nodes get a copy of the initial blockchain, which only includes the genesis block. Blocks are similar to any other traditional blockchain except for an extra field that stores the block dimension; the dimension used is the concatenation of $[CoSi\_group_hash|group\_center\_location]$.

Before conducting the evaluation, it is essential to adhere to the best practices to let the multidimensional blockchain operate as efficiently as possible based on the multidimensional blockchain model presented in [ZMP+22]. This entails the selection of dimensions that can minimize the blockchain depth to improve blockchain operations, such as forging, scanning, and validation. With this in mind, the collective signing-based storage optimization model is evaluated under these conditions.

As seen in Fig. 5.2, we present an assessment of the maximum blockchain depth for four different multidimensional blockchains with 1, 2, 3, and 4 dimensions, as well

as a DAG blockchain. The dimensions include x and y coordinates, node ID, and time. Results demonstrate that using a 1-d blockchain can result in a comparable maximum depth to that of 2, 3, and 4-d blockchains, but not for the DAG blockchain. It is also notable that the number of dimensions does not affect the blockchain depth as long as all of them have dimensions that exhibit randomness or uniqueness, such as coordinates or the block or Merkle hash. Notably, a multidimensional blockchain with one dimension (preferably a dimension with a broader spectrum of values less likely to repeat) exhibits the lowest time to forge a block compared to the others, as demonstrated in the following figure or Fig. 5.3. This is because using a 1-d multidimensional blockchain can reduce the time spent on block comparisons and switching between different dimensions.

In light of the importance of reducing the blockchain depth or height to enhance blockchain operations, selecting appropriate dimensions that can effectively achieve this objective is also critical. As shown previously, a 1-d blockchain can achieve the most desirable performance. Another advantage of using only one dimension for the blockchain is that we can mark every blockchain merge and split by using a unique identifier for each CoSi group, which can facilitate an efficient tracking of block production and management of the blockchain [FHB+22]. This unique identifier is referred to as the CoSi group hash, which is generated by collecting the public keys of all forgers within the CoSi group, sorting them to prevent other permutations, and generating a hash for all the sorted public keys, which is then used as for the block dimension. However, it is paramount to note that the CoSi group hash has a limited spectrum. Specifically, all blocks generated by the same CoSi group will have the same block dimension, leading to linear growth in the maximum blockchain depth and potentially impeding the overall performance of

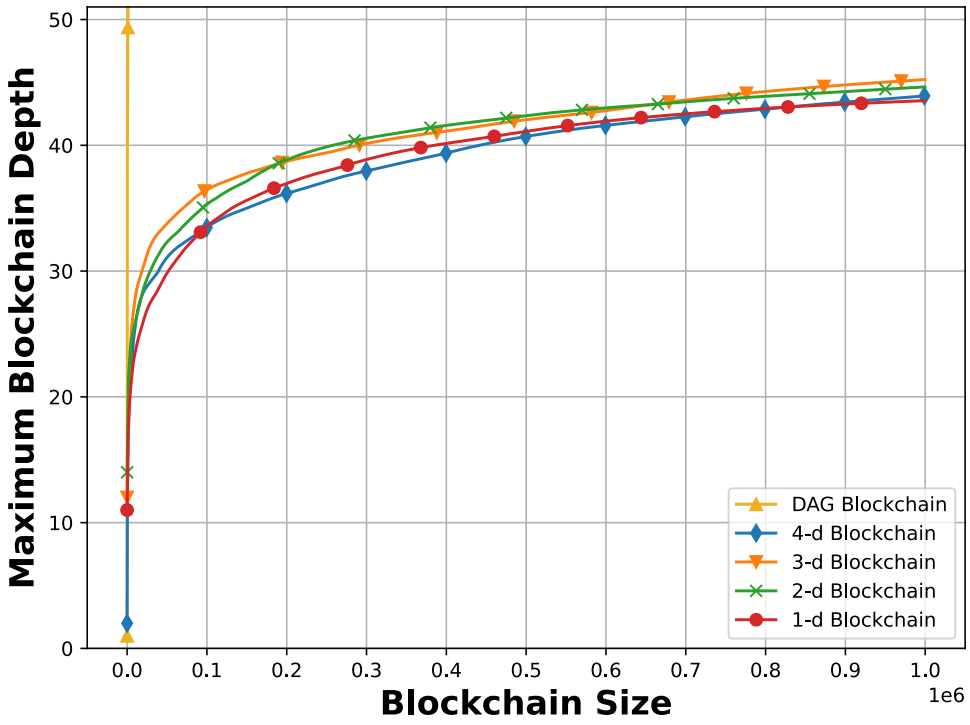

Figure 5.2: Maximum blockchain depth comparison for multidimensional and DAG blockchains.

the blockchain, and resulting in longer branches because future blocks will be added to the right side of the blockchain tree.

To mitigate this issue, introducing randomness to block dimensions is a potential strategy to reduce the maximum blockchain depth, as stated in [ZMP+22]. Fortunately, our simulation model incorporates a stochastic process that employs node movement using random walk [HV07, Pan]. This randomness feature can be leveraged and combined with the CoSi group hash to create a more robust block dimension. Specifically, our final block dimension comprises a string that contains (a fixed-size CoSi hash and the CoSi group location). Fig. 5.4 represents a 1-d blockchain with a block dimension with a limited spectrum (fixed size CoSi group

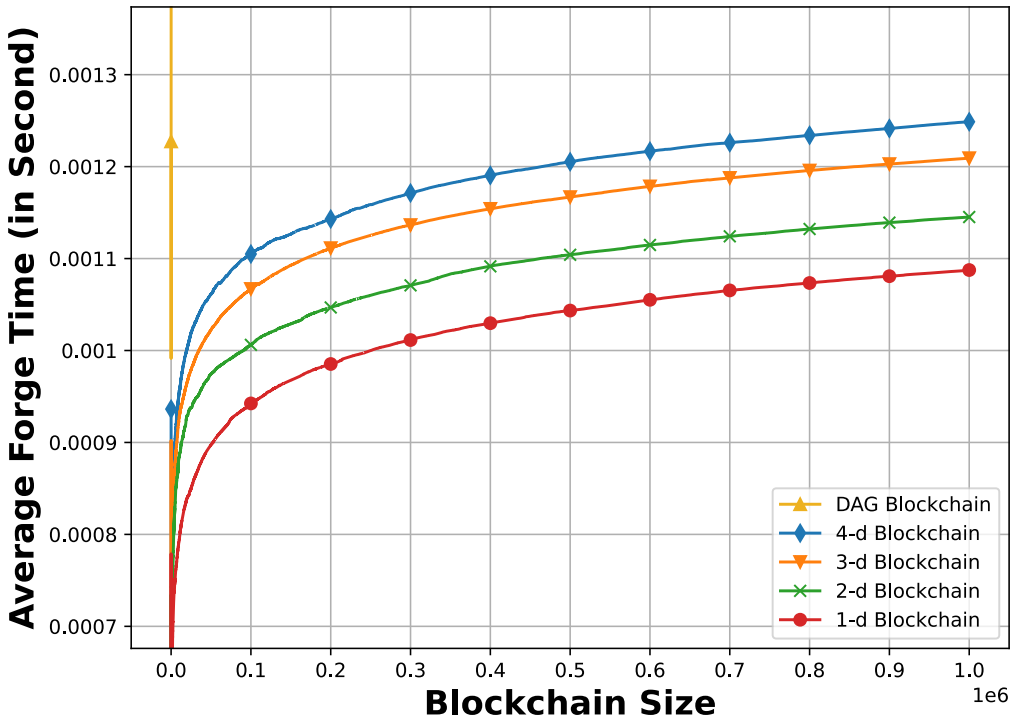

Figure 5.3: Time to forge a block in multidimensional blockchains with different dimensions and DAG blockchain.

hash) and a block dimension with a wide spectrum (fixed size CoSi group hash and CoSi group location). The results of our evaluation reveal that using one dimension with a limited spectrum can result in a longer maximum depth with linear growth. In contrast, a broader spectrum can minimize the maximum depth resulting in growth similar to logarithmic growth, which can significantly reduce the maximum blockchain depth.

Next, we explore the impact of high and low mobility for our model by manipulating the interaction between CoSi groups. To achieve this, we utilized a parameter called aggregation value (agg) as detailed in [Pan]. We selected two values, 0.1 and 0.9, where a smaller aggregation value denotes the nodes are randomly distributed

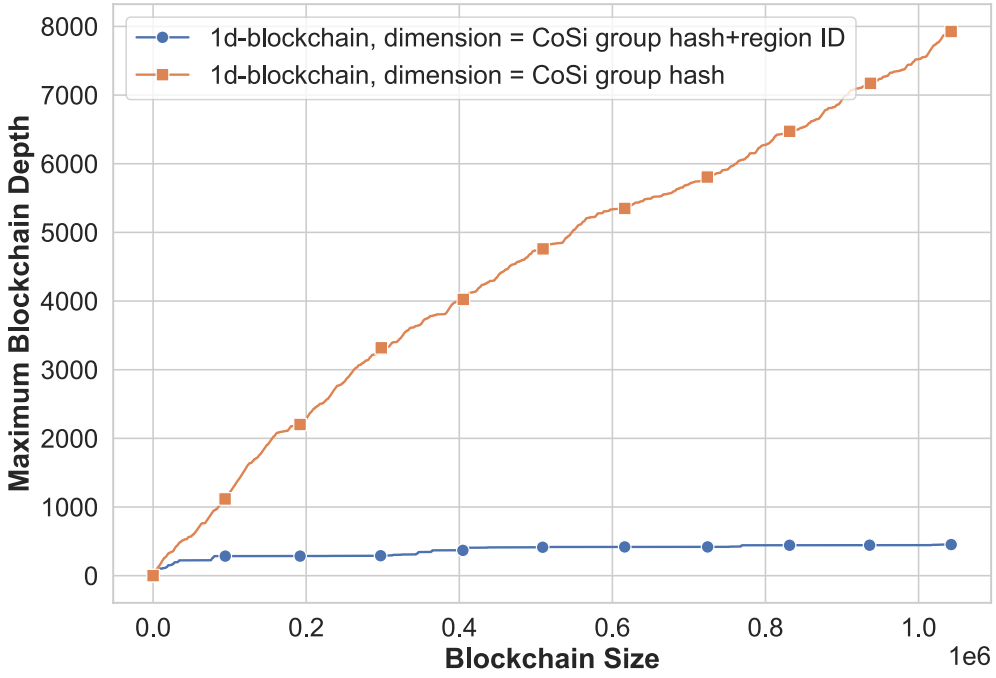

Figure 5.4: Comparison of maximum blockchain depth with and without region information in block dimension.

away from the group center, resulting in high interactions between CoSi groups, and a larger aggregation value indicates the nodes are closer to the group center, leading to low interactions between CoSi groups. Results presented in Fig. 5.5 show that a model with higher mobility ($agg = 0.1$) tends to have a lower maximum blockchain depth because more interactions between different CoSi groups create more CoSi group hashes as the blockchain grows. Since blocks are placed based on their dimension values, these CoSi group hashes help append blocks with wider spectrum dimensions. In contrast, lower interaction between groups ($agg = 0.9$) can result in appending more blocks with the same dimensions, which can increase the maximum blockchain depth.

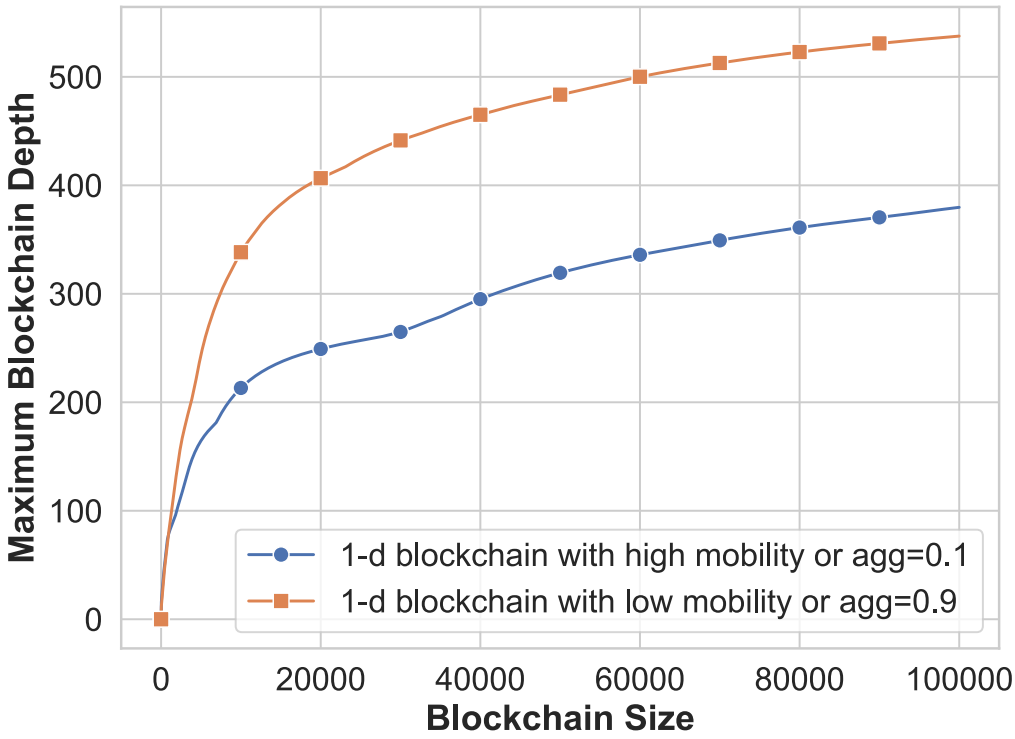

Figure 5.5: Effect of mobility on maximum blockchain depth.

In this study, we also explore the time required to forge a block in our model, using the same aggregation values for low and high mobility conditions, as depicted in Figure 5.6. Two observations can be discussed in this context. First, as we saw in previous figures (Fig.5.2 and 5.3), minimizing the maximum blockchain depth can result in improving the blockchain operations. However, the time to forge blocks remains relatively stable, as demonstrated in Fig. 5.6. This outcome can be attributed to the fact that collective signing is an expensive process that requires all miners within each CoSi group to sign their respective blocks. As such, collectively signing a block generally takes longer than forging a block by a single miner. Secondly, we observe that CoSi groups tend to have larger sizes in high-mobility scenarios due to

increased interactions among other groups. This can result in longer block forging times than smaller CoSi groups in low-mobility scenarios.

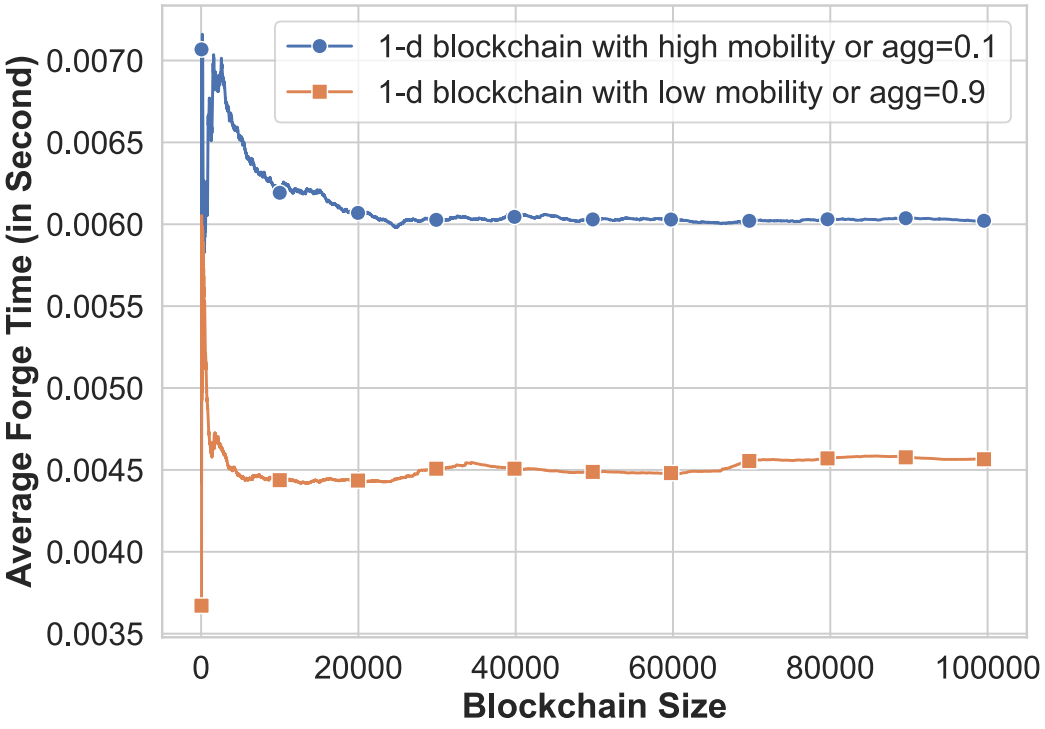

Figure 5.6: Block forge time comparison for low and high mobility conditions.

5.4.1 Evaluating The Model

The next part of this section is to evaluate our model (CoSi-based storage optimization) with another state-of-the-art model (community interest-based storage optimization [YDXJ20]) using two real-world datasets. The first dataset is [Big], which is used to generate actual block sizes based on the Ethereum Classic dataset. The second dataset is the social circles in Facebook [LM12], which introduces community interest and injects it to the miners since [YDXJ20] lacks the details regarding com-

munity topics or interests. It is also important to note that we adopted a parameter or threshold to tune storage optimization for both models to prune or keep the least and most relevant blocks.

Fig. 5.7 shows our CoSi-based storage optimization under five distinct thresholds and their overall mean. When the threshold is set to zero, the model traverses through all blocks and prunes those with a similarity or Jaccard score less than or equal to zero, based on Algorithm 8 and [RV96]. We believe blocks with similarity scores equal to zero should be the first to be pruned, given their irrelevance to their current CoSi group. Our results demonstrate that the CoSi group-based storage optimization model can effectively prune the most irrelevant blocks per each CoSi group, leading to storage space savings of approximately 93%. However, it should be noted that other thresholds can yield even greater storage space savings but may prune some relevant blocks to their corresponding CoSi group. Ultimately, the overall mean of our CoSi group-based storage optimization model is around 92%, indicating the average storage saved per each CoSi group.

Since Fig. 5.7 measures the average storage saved per each CoSi group using our model, it is pertinent to explore the potential implications of an increase in CoSi group size, which may require fewer blocks to be pruned. In order to address this concern, we present Fig. 5.8 to answer this question which provides insights into the relationship between CoSi group size and the percentage of storage saved. In this figure, we consider CoSi group sizes ranging from 4 to 36 miners per group. We can conclude from Fig. 5.8 that as the CoSi group size increases, more blocks need to be retained, reducing the percentage of storage saved.

Next, we compare our CoSi-based storage optimization with community interest-based storage optimization found in [YDXJ20] using a real-world community interest dataset of Facebook users [LM12]. The work done by [YDXJ20] lacks the details

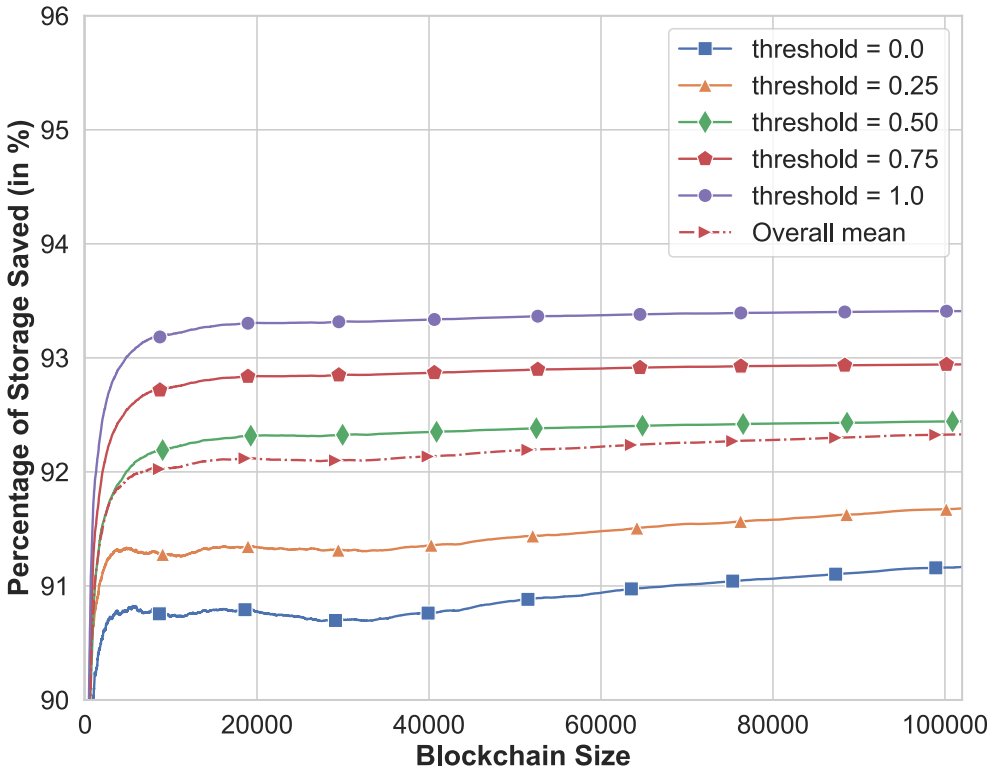

Figure 5.7: CoSi-Based storage optimization results under five distinct thresholds and their overall mean.

about how to define community topics or interests, so we implemented community interests based on Facebook users [LM12] and injected them into the miners in the simulations.

As demonstrated in Fig. 5.9, the evaluation of storage optimization based on community interests was conducted under five distinct thresholds. When the threshold is set to 0, we aim to prune blocks with a similarity or Jaccard score equal to 0 using the same Jaccard or similarity formula found in Algorithm 9. Instead of using public keys, we use community interests to calculate the score.

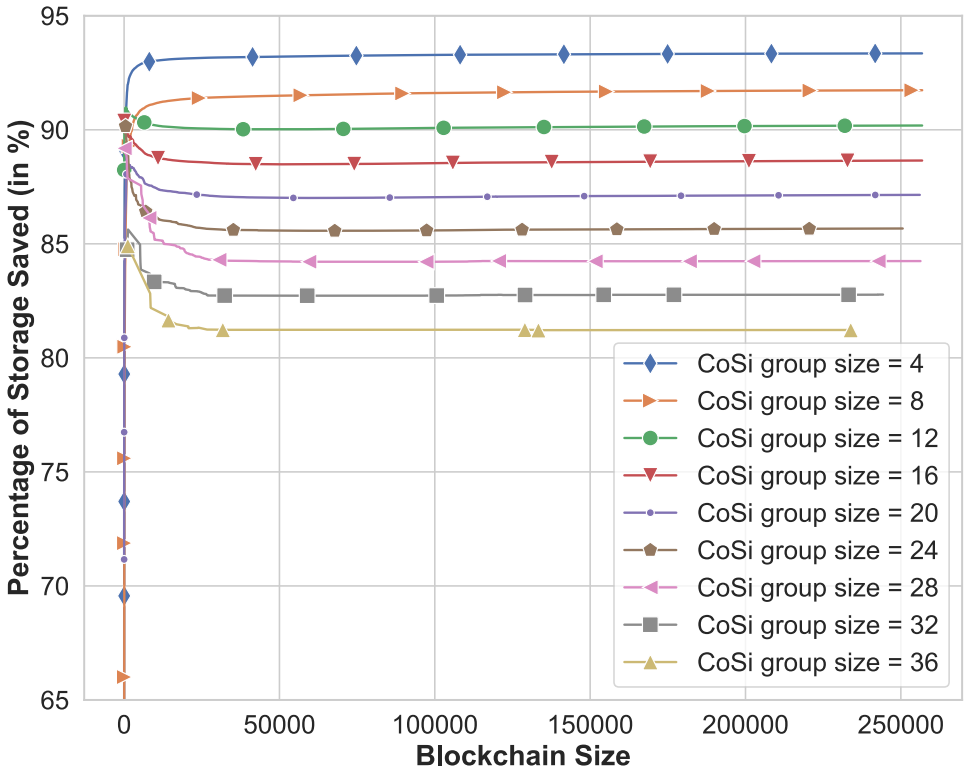

Figure 5.8: Relationship between CoSi group size and percentage of storage saved.

Based on the community interests derived from the Facebook dataset, results indicated a negligible overlap in interests among communities, resulting in almost no blocks being pruned, and the percentage of storage saved remained close to zero. Furthermore, when the threshold is less or equal to one, the resulting data was identical to the data obtained at a threshold of 0.75 because no two communities share 100% of their interests based on the dataset. Overall, community interest-based storage optimization resulted in storage savings of approximately 11%.

Lastly, Fig. 5.10 presents a comprehensive comparison between our proposed CoSi-based storage optimization model, the community interest-based storage opti-

mization model, as outlined in [YDXJ20], and the blockchain storage optimization based on unspent transactions (ESS) [WWZC21]. By utilizing multiple real-world datasets that represent blockchain size, community interests, and unspent transactions, we can summarize that our CoSi-based storage optimization model can achieve around 92% storage optimization, which is significantly higher compared to the 11% achieved by the community interest-based model. This performance discrepancy can be attributed to the fact that our proposed model can find and prune the most irrelevant blocks for each group, which is a challenging task to achieve in the community interest-based model due to the overlap in shared interests between communities, making it more challenging to identify and remove the most irrelevant blocks for each group.

Furthermore, we compare our model with the ESS model found in [WWZC21]. As shown in Fig. 5.10, we can see that the ESS model starts a higher storage optimization, but later as the blockchain grows, the percentage of storage saved drops to around 81% when the blockchain size reaches $600,000$. ESS can achieve higher optimization at the beginning because 99.9% of all blocks between block height 0 to $100,000$ are pruned because they are considered in the ESS model as old blocks with lower query probabilities. However, as the blockchain grows, newer blocks have higher probabilities of being queried, and hence many of the new blocks are not pruned, and only 74.9% of the blocks are pruned at blockchain height between $500,000$ to $600,000$. Moreover, the ESS evaluation ends at $600,000$ because their dataset [Psy] is capped at a blockchain with a height of $600,000$ blocks. Another observation of the ESS model is that its storage optimization tends to degrade over time. Since the dataset caps at $600,000$ blocks, we draw a trend for the ESS model to estimate the future values using a polynomial regression model using [Ost12]. We can see the ESS model is estimated to continue to degrade in the future. However,

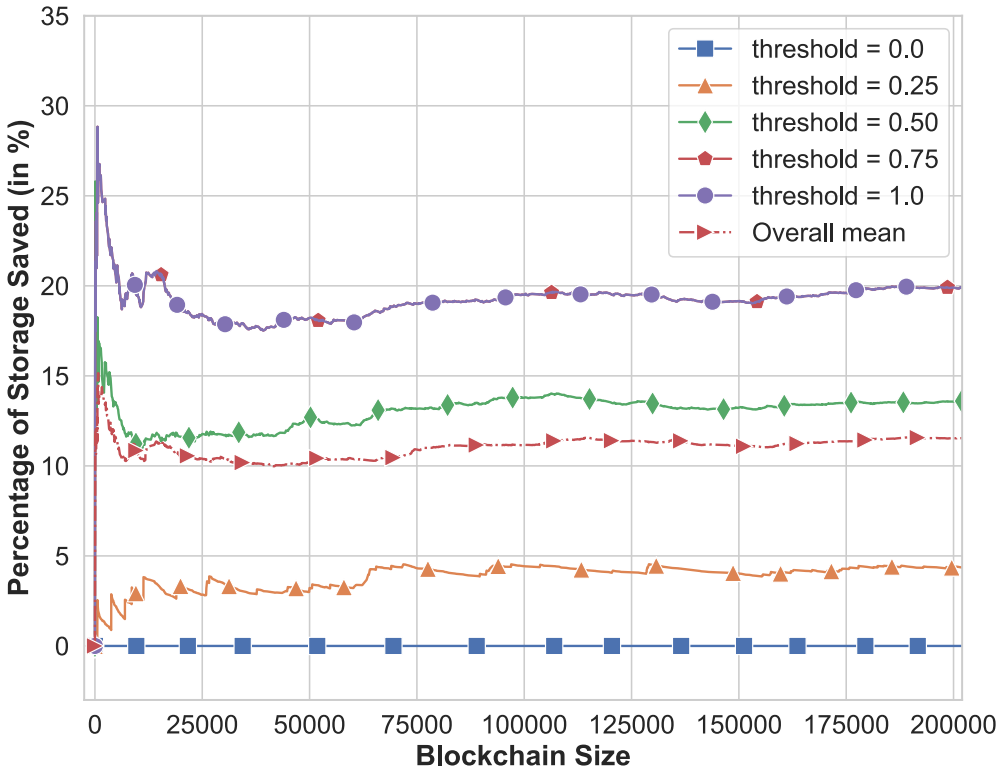

Figure 5.9: Community interest-Based storage optimization [YDXJ20] results under five distinct thresholds and their overall mean using Facebook dataset [LM12].

our and the community interest-based storage optimization models continue to run stable over time.

Overall, our evaluation results demonstrate that the CoSi-based storage optimization model outperforms the community interest-based and ESS models and can be a useful approach to optimize storage space in blockchain systems.

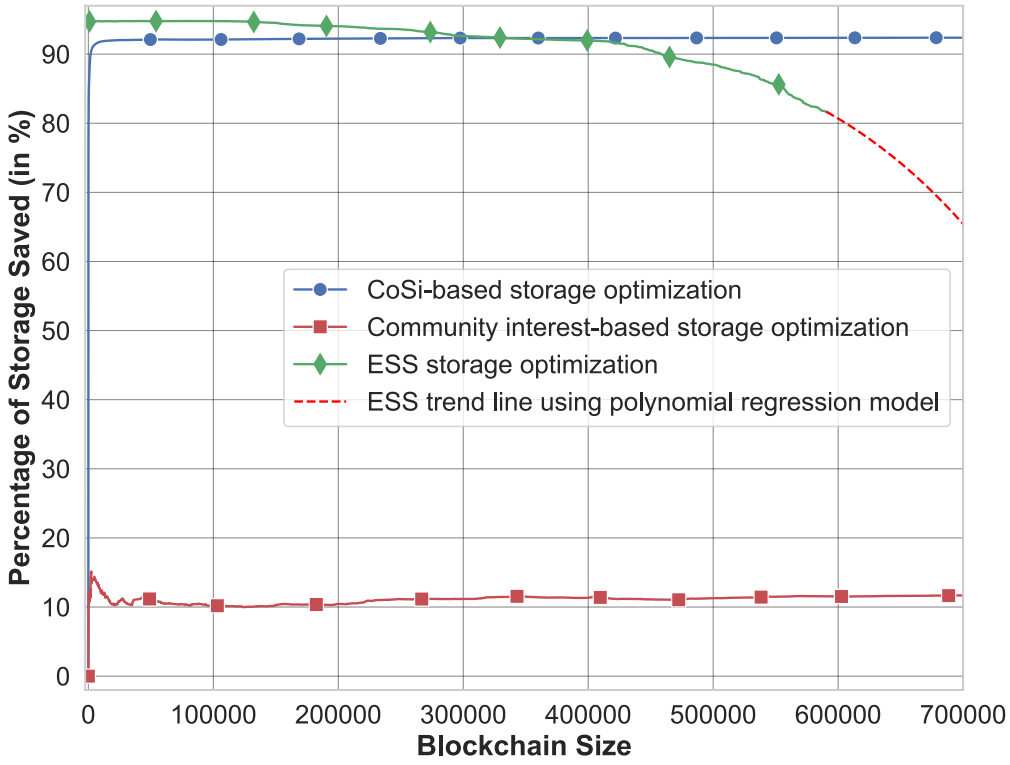

Figure 5.10: Comparison between CoSi-based (our model), community interest-based storage optimization model based on [YDXJ20], and the ESS model [WWZC21].

5.5 Conclusion

In conclusion, this chapter proposes a novel Collective Signing-Based Blockchain Storage Optimization (CSBSO) model that uses an efficient blockchain structure to address the scalability issue of blockchain storage in mobile IoT. The proposed model utilizes a multidimensional blockchain structure and extends the existing CoSi protocol to reduce the storage overhead and manage and retrieve blocks more efficiently. The approach for storage optimization includes identifying all kinds

of blocks (relevant, somehow relevant, and irrelevant blocks) and pruning all the irrelevant blocks per each group based on the Collective Signing protocol. Our experiments with two real-world datasets show that our collective signing-based storage optimization can effectively optimize storage and outperform existing state-of-the-art techniques and save up to 92% of storage space.

CHAPTER 6

LIMITATIONS, FUTURE WORK, AND CONCLUSION

In this dissertation, we discuss how to advance the current blockchain model to achieve a more efficient, scalable, and resource-optimized blockchain for mIoT systems. In Chapter 3, we propose a multidimensional, graph-based blockchain for mIoT. We then design an efficient approach for merging multidimensional in Chapter 4. Afterward, in Chapter 5, we propose CSBSO, a collective signing-based blockchain storage optimization. In this chapter, we discuss the limitations of each work, possible future directions, and concluding remarks.

6.1 Limitations and Future Work

6.1.1 An Optimized Collective Signing Protocol for Blockchain in Mobile IoT

In Chapter 5, we introduce a blockchain storage optimization model that utilizes a collective signing protocol to reduce the blockchain storage overhead in mIoT. We also mentioned techniques that can help reduce the maximum blockchain depth to maintain the intrinsic feature that allows multidimensional blockchains to run efficiently. An example of such a technique is introducing stochastic randomness based on nodes' movements to the block dimension. Adding stochastic randomness in the block dimension reduces the blockchain depth, which can help improve blockchain efficiency because it allows us to traverse the blockchain with fewer comparisons [ZMP+22]. Despite efforts to minimize the maximum blockchain depth, the time to forge blocks can fluctuate and then stabilize for 400 nodes due to the inherent complexities of the CoSi process in Figure 5.6. We expect the average forge time

to increase if the number of nodes increases because collective signing requires all miners within each CoSi group to sign their respective blocks. So if the group becomes larger, what would the impact of the CoSi protocol be on the performance and efficiency of blockchain systems? In addition, what strategies or alternative methods could be developed to mitigate these limitations? Future work or direction can explore ways to optimize the CoSi process to deal with larger groups in dynamic environments while maintaining a sufficient number of nodes to sign new blocks.

6.1.2 Investigating the Energy and Memory Cost of Merging Blockchains in mIoT

The process of merging multiple blockchains may result in significant energy consumption, especially when dealing with sizable blockchains that frequently split and merge. Considering those scenarios, the merging process can present challenges for mobile IoT devices due to their restricted energy capacities. Although Chapter 4 presents a proof-of-concept, further investigation is required to study the actual cost of the memory and energy requirements. Future work could investigate the implication of merging multiple blockchains by measuring the energy cost and assessing the memory requirements. Another future work could also identify potential challenges and limitations in implementing the proposed merging process in real-world mobile IoT environments.

6.1.3 Mitigating Security Risks During Blockchain Merging

When merging multiple blockchains, an attacker can take advantage of launching multiple attacks. For example, an attacker can simultaneously post the same trans-

action on multiple blockchains. If the merging process does not carefully manage this type of malicious behavior, an attacker can submit one transaction multiple times, which can cause double-spending. Future work to address this issue could include ensuring proper synchronization during the merge process by designing a protocol to confirm the order of transactions or implementing measures to verify the state of multiple chains. Another possible attack that a malicious user can launch is the Sybil attack. For example, an attacker can create multiple identities and control the network; more specifically, the attacker can control the flow of information within the network, deciding which transactions get processed or blocks get added to the blockchain. To mitigate the risk of a Sybil attack, a possible future direction could be implementing a distributed reputation system to help verify nodes' identities to prevent a malicious node from creating multiple identities.

6.1.4 Protecting Resource-Optimized Blockchain Applications from Storage Manipulation Attacks

In Chapter 5, pruning blocks based on CoSi protocol is the primary method to achieve storage optimization. Although the CoSi protocol can identify and prune the most irrelevant blocks and keep the relevant blocks, a malicious entity can still initiate attacks to manipulate the available storage for some nodes. For example, an attacker can launch block flooding, which can overwhelm the system by sending multiple blocks to manipulate nodes' storage. This can affect the nodes by not being able to optimize the storage to store the most relevant block, which can compromise the overall blockchain integrity. Another way for an attacker to apply a storage manipulation attack is by creating multiple identities to generate relevant blocks, making the system store fake but perceived as relevant blocks. Possible future work

to mitigate those risks is to design systems that can detect those attacks and enforce some limit on block or transaction generation. Another potential future work is to build more robust pruning techniques, which can determine if the perceived relevant blocks are part of a malicious attack by a specific node or group aiming to manipulate network participants' storage.

6.2 Conclusion

In this dissertation, we present different approaches to advance the scalability, efficiency, and storage optimization of blockchains in mIoT. The scope of this research covers blockchain-based applications in resource-constrained, self-organized, and self-configured mobile IoT devices. In our first study, we propose a multidimensional, graph-based blockchain structure that utilizes k-dimensional spatiotemporal space to address the challenges of applying blockchain in mobile networks with limited resources and the performance limitation of traditional chain or graph-based blockchains. We then observed that these multidimensional blockchains could frequently split and merge in dynamic mobile networks, leading to wasted computational power in processing identical blocks. To address this issue, we developed an efficient merging approach for multidimensional or graph-based blockchains, which minimizes the processing of identical blocks during merging in Chapter 4. Afterward, in Chapter 5, we propose CSBSO, a collective signing-based blockchain storage optimization model, to reduce the storage costs of mobile IoT nodes. Finally, 6, we highlight some limitations of each work and provide a road map for future research directions.

BIBLIOGRAPHY

[AMTS+22] Nana Kwadwo Akrasi-Mensah, Eric Tutu Tchao, Axel Sikora, Andrew Selasi Agbemenu, Henry Nunoo-Mensah, Abdul-Rahman Ahmed, Dominik Welte, and Eliel Keelson. An overview of technologies for improving storage efficiency in blockchain-based iiot applications. *Electronics*, 11(16):2513, 2022.

[ASHN21] Alia Al Sadawi, Mohamed S Hassan, and Malick Ndiaye. A survey on the integration of blockchain with iot to enhance performance and eliminate challenges. *IEEE Access*, 9:54478–54497, 2021.

[B+02] Adam Back et al. Hashcash-a denial of service counter-measure. 2002.

[BCD+14] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 72, 2014.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[BHPC04] Christian Bettstetter, Hannes Hartenstein, and Xavier Pérez-Costa. Stochastic properties of the random waypoint mobility model. *Wireless networks*, 10(5):555–567, 2004.

[Big] Google BigQuery. Ethereum classic blockchain. Accessed on 10 January, 2023.

[Bita] Bitcoin.org. Bitcoin core version 0.11.0 released. Accessed on 9 March, 2023.

[Bitb] Bitcoin.org. Potential spv weaknesses. Accessed on 9 March, 2023.

[BKLMC20] Georgios Birmpas, Elias Koutsoupias, Philip Lazos, and Francisco J Marmolejo-Cossío. Fairness and efficiency in dag-based cryptocurrencies. In *International Conference on Financial Cryptography and Data Security*, pages 79–96. Springer, 2020.

[BPS16] Iddo Bentov, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. *IACR Cryptol. ePrint Arch.*, 2016:919, 2016.

[BSHC18] Zijian Bao, Wenbo Shi, Debiao He, and Kim-Kwang Raymond Chood. Iotchain: A three-tier blockchain-based iot security architecture. *arXiv preprint arXiv:1806.02008*, 2018.

[CL+99] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[CLP+20] David Cordova, Alexandre Laube, Guy Pujolle, et al. Blockgraph: A blockchain for mobile ad hoc networks. In *2020 4th Cyber Security in Networking Conference (CSNet)*, pages 1–8. IEEE, 2020.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, USA, 3rd edition, 2009.

[CLY19] Xiaojiao Chen, Sianjheng Lin, and Nenghai Yu. Bitcoin blockchain compression algorithm for blank node synchronization. In *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6. IEEE, 2019.

[CZC21] Mingrui Cao, Long Zhang, and Bin Cao. Toward on-device federated learning: a direct acyclic graph-based blockchain approach. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[Dan15] Quynh H Dang. Secure hash standard. 2015.

[DDK+21] Panagiotis Drakatos, Erodotos Demetriou, Stavroulla Koumou, Andreas Konstantinidis, and Demetrios Zeinalipour-Yazti. Towards a blockchain database for massive iot workloads. In *2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW)*, pages 76–79. IEEE, 2021.

[Ded] Dedis. Dedis/cothority: Scalable collective authority. Accessed on 12 January, 2023.

[DKJG17] Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. Blockchain for iot security and privacy: The case study of a smart home. In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pages 618–623. IEEE, 2017.

[DKJG19] Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. Lsb: A lightweight scalable blockchain for iot security and anonymity. *Journal of Parallel and Distributed Computing*, 134:180–197, 2019.

[DKSP19] Pietro Danzi, Anders E Kalør, Čedomir Stefanović, and Petar Popovski. Delay and communication tradeoffs for blockchain systems with lightweight iot clients. *IEEE Internet of Things Journal*, 6(2):2354–2365, 2019.

[DZZ19] Hong-Ning Dai, Zibin Zheng, and Yan Zhang. Blockchain for internet of things: A survey. *IEEE Internet of Things Journal*, 6(5):8076–8094, 2019.

[Eth] Ethereum.org. Paths toward single-slot finality.

[FHB+22] Juncheng Fang, Farzad Habibi, Kevin Bruhwiler, Fayzah Alshammari, Abhishek Singh, Yinan Zhou, and Faisal Nawab. Pelopartition: Improving blockchain resilience to network partitioning. In *2022 IEEE International Conference on Blockchain (Blockchain)*, pages 274–281. IEEE, 2022.

[FHBS19] Martin Florian, Sebastian Henningsen, Sophie Beaucamp, and Björn Scheuermann. Erasing data from blockchain nodes. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 367–376. IEEE, 2019.

[FQZ+22] Yanqing Fan, Tie Qiu, Lidi Zhang, Tianyi Xu, Wenyuan Liu, Xiaobo Zhou, and Zhiguo Wan. Dlbn: Group storage mechanism based on double-layer blockchain network. *IEEE Internet of Things Journal*, 9(20):19649–19659, 2022.

[GNH21] Tieming Geng, Laurent Njilla, and Chin-Tser Huang. Smart markers in smart contracts: Enabling multiway branching and merging in blockchain for decentralized runtime verification. In *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8. IEEE, 2021.

[GYZ+19] Fengxian Guo, F Richard Yu, Heli Zhang, Hong Ji, Mengting Liu, and Victor CM Leung. Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing. *IEEE Transactions on Wireless Communications*, 19(3):1689–1703, 2019.

[HGPC99] Xiaoyan Hong, Mario Gerla, Guangyu Pei, and Ching-Chuan Chiang. A group mobility model for ad hoc wireless networks. In *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 53–60, 1999.

[HGW+22] Zhuoer Hu, Hui Gao, Taotao Wang, Daoqi Han, and Yueming Lu. Joint optimization for mobile edge computing-enabled blockchain systems: A deep reinforcement learning approach. *Sensors*, 22(9):3217, 2022.

[HONS21] Kendric Hood, Joseph Oglio, Mikhail Nesterenko, and Gokarna Sharma. Partitionable asynchronous cryptocurrency blockchain. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2021.

[HS90] Stuart Haber and W Scott Stornetta. How to time-stamp a digital document. In *Conference on the Theory and Application of Cryptography*, pages 437–455. Springer, 1990.

[HS21] Sung-Jung Hsiao and Wen-Tsai Sung. Employing blockchain technology to strengthen security of wireless sensor networks. *IEEE Access*, 9:72326–72341, 2021.

[HV07] Esa Hyytiä and Jorma Virtamo. Random waypoint mobility model in cellular networks. *Wireless Networks*, 13(2):177–188, 2007.

[KJ17] Yoohwan Kim and Juyeon Jo. Binary blockchain: Solving the mining congestion problem by dynamically adjusting the mining capacity. In *International Conference on Applied Computing and Information Technology*, pages 29–49. Springer, 2017.

[KJG+16] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th usenix security symposium (usenix security 16)*, pages 279–296, 2016.

[KJW+18] Kolbeinn Karlsson, Weitao Jiang, Stephen Wicker, Danny Adams, Edwin Ma, Robbert van Renesse, and Hakim Weatherspoon. Vegvisir: A partition-tolerant blockchain for the internet-of-things. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1150–1158. IEEE, 2018.

[KKJG+16] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. USENIX Association, 2016.

[KS18] Minhaj Ahmad Khan and Khaled Salah. Iot security: Review, blockchain solutions, and open challenges. *Future generation computer systems*, 82:395–411, 2018.

[Lar14] Daniel Larimer. Delegated proof-of-stake (dpos). *Bitshare whitepaper*, 2014.

[LCL+20] Yuzheng Li, Chuan Chen, Nan Liu, Huawei Huang, Zibin Zheng, and Qiang Yan. A blockchain-based decentralized federated learning framework with committee consensus. *IEEE Network*, 35(1):234–241, 2020.

[LCP+20] Yixin Li, Bin Cao, Mugen Peng, Long Zhang, Lei Zhang, Daquan Feng, and Jihong Yu. Direct acyclic graph-based ledger for internet of things: Performance and security analysis. *IEEE/ACM Transactions on Networking*, 28(4):1643–1656, 2020.

[LCZ+22] Zhuofan Liao, Siwei Cheng, Jingyu Zhang, Wenbing Wu, Jin Wang, and Pradip Kumar Sharma. Gpdb: A graph-partition based storage strategy for dag-blockchain in edge-cloud iiot. *IEEE Transactions on Industrial Informatics*, 2022.

[LLC+18] Lun Li, Jiqiang Liu, Lichen Cheng, Shuo Qiu, Wei Wang, Xiangliang Zhang, and Zonghua Zhang. Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 19(7):2204–2220, 2018.

[LM12] Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. *Advances in neural information processing systems*, 25, 2012.

[LMAA19] Alexandre Laube, Steven Martin, and Khaldoun Al Agha. A solution to the split & merge problem for blockchain-based applications in ad hoc networks. In *2019 8th International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN)*, pages 1–6. IEEE, 2019.

[LNZ+16] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 17–30, 2016.

[LSZ15] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 528–547. Springer, 2015.

[LWLX19] Yinqiu Liu, Kun Wang, Yun Lin, and Wenyao Xu. Lightchain: a lightweight blockchain system for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 15(6):3571–3581, 2019.

[LWQ+19] Yinqiu Liu, Kun Wang, Kai Qian, Miao Du, and Song Guo. Tornado: Enabling blockchain in heterogeneous internet of things through a space-structured approach. *IEEE Internet of Things Journal*, 7(2):1273–1286, 2019.

[LYHK07] Feifei Li, Ke Yi, Marios Hadjieleftheriou, and George Kollios. Proof-infused streams: Enabling authentication of sliding window queries on streams. In *Proceedings of the 33rd international conference on Very large data bases*, pages 147–158, 2007.

[LYW+21] Kunchang Li, Yifan Yang, Shuhao Wang, Runhua Shi, and Jianbin Li. A lightweight privacy-preserving and sharing scheme with dual-blockchain for intelligent pricing system of smart grid. *Computers & Security*, 103:102189, 2021.

[Mer89] Ralph C Merkle. A certified digital signature. In *Conference on the Theory and Application of Cryptology*, pages 218–238. Springer, 1989.

[MKP+21] Roman Matzutt, Benedikt Kalde, Jan Pennekamp, Arthur Drichel, Martin Henze, and Klaus Wehrle. Coinprune: Shrinking bitcoin's blockchain retrospectively. *IEEE Transactions on Network and Service Management*, 18(3):3064–3078, 2021.

[MLN+19] Raheel Ahmed Memon, Jian Ping Li, Muhammad Irshad Nazeer, Ahmad Neyaz Khan, and Junaid Ahmed. Dualfog-iot: Additional fog layer for solving blockchain integration problem in internet of things. *IEEE Access*, 7:169073–169093, 2019.

[Moo90] Andrew William. Moore. *Efficient memory-based learning for robot control*. PhD thesis, 1990.

[MRR+20] Sachi Nandan Mohanty, KC Ramya, S Sheeba Rani, Deepak Gupta, K Shankar, SK Lakshmanaprabu, and Ashish Khanna. An efficient lightweight integrated blockchain (elib) model for iot security and privacy. *Future Generation Computer Systems*, 102:1027–1037, 2020.

[MVL+21] David Cordova Morales, Pedro B Velloso, Alexandre Laube, Guy Pujolle, et al. C4m: A partition-robust consensus algorithm for blockgraph in mesh network. In *2021 5th Cyber Security in Networking Conference (CSNet)*, pages 82–89. IEEE, 2021.

[MZFZ19] Alexander Marsalek, Thomas Zefferer, Edona Fasllija, and Dominik Ziegler. Tackling data inefficiency: Compressing the bitcoin blockchain. In *2019 18th IEEE international conference on trust, security and privacy in computing and communications/13th IEEE international conference on big data science and engineering (trustcom/bigdatase)*, pages 626–633. IEEE, 2019.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, page 21260, 2008.

[Nov18] Oscar Novo. Blockchain meets iot: An architecture for scalable access management in iot. *IEEE internet of things journal*, 5(2):1184–1195, 2018.

[NPDS21] Dinh C Nguyen, Pubudu N Pathirana, Ming Ding, and Aruna Seneviratne. Secure computation offloading in blockchain based iot networks with deep reinforcement learning. *IEEE Transactions on Network Science and Engineering*, 8(4):3192–3208, 2021.

[NTG+22] Clement Nartey, Eric Tutu Tchao, James Dzisi Gadze, Bright Yeboah-Akowuah, Henry Nunoo-Mensah, Dominik Welte, and Axel Sikora. Blockchain-iot peer device storage optimization using an advanced time-variant multi-objective particle swarm optimization algorithm. *EURASIP Journal on Wireless Communications and Networking*, 2022(1):1–27, 2022.

[Ost12] Eva Ostertagová. Modelling using polynomial regression. *Procedia Engineering*, 48:500–506, 2012.

[Pan] André Panisson. Istituto per l'interscambio scientifico i.s.i. Accessed on 12 January, 2023.

[PB19] Chan Kyu Pyoung and Seung Jun Baek. Blockchain of finite-lifetime blocks with applications to edge-based iot. *IEEE Internet of Things Journal*, 7(3):2102–2116, 2019.

[PMIH18] Huma Pervez, Muhammad Muneeb, Muhammad Usama Irfan, and Irfan Ul Haq. A comparative analysis of dag-based blockchain architectures. In *2018 12th International conference on open source systems and technologies (ICOSST)*, pages 27–34. IEEE, 2018.

[PPJP12] Sitthapon Pumpichet, Niki Pissinou, Xinyu Jin, and Deng Pan. Belief-based cleaning in trajectory sensor streams. In *2012 IEEE International Conference on Communications (ICC)*, pages 208–212. IEEE, 2012.

[Psy] Psychowo. Bitcoinv0.20.0-utxo-parse. Accessed on 15 January, 2023.

[RV96] Raimundo Real and Juan M Vargas. The probabilistic basis of jaccard's index of similarity. *Systematic biology*, 45(3):380–385, 1996.

[Sic09] Mihail L Sichitiu. Mobility models for ad hoc networks. In *Guide to Wireless Ad Hoc Networks*, pages 237–254. Springer, 2009.

[SJ20] DM Sheeba and S Jayalakshmi. 'lightweight blockchain to improve security and privacy in smarthome. *Int. J. Recent Technol. Eng.(IJRTE)*, 8(6):5021–5027, 2020.

[Ski20] Alexander Skidanov. Doomslug vs pbft, tendermint, and hotstuff, Feb 2020.

[SLZ16] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. *Cryptology ePrint Archive*, 2016.

[SPR21] Alexe Luca Spataru, Ciprian-Petrisor Pungila, and Marco Radovancovici. A high-performance native approach to adaptive blockchain smart-contract transmission and execution. *Information Processing & Management*, 58(4):102561, 2021.

[SPSK19] Abdur R Shahid, Niki Pissinou, Corey Staier, and Rain Kwan. Sensorchain: a lightweight scalable blockchain framework for internet of things. In *2019 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)*, pages 1154–1161. IEEE, 2019.

[SS19] Muhammad Yanuar Ary Saputro and Riri Fitri Sari. Securing iot network using lightweight multi-fog (lmf) blockchain model. In *2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, pages 183–188. IEEE, 2019.

[STV+16] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping authorities" honest or bust" with decentralized witness cosigning. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 526–545. Ieee, 2016.

[SZ18] Yonatan Sompolinsky and Aviv Zohar. Phantom. *IACR Cryptology ePrint Archive, Report 2018/104*, 2018.

[Tar72] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[TC20] Horst Treiblmaier and Trevor Clohessy. *Blockchain and Distributed Ledger Technology Use Cases*. Springer, 2020.

[TLJ+20] Yuechen Tao, Bo Li, Jingjie Jiang, Hok Chu Ng, Cong Wang, and Baochun Li. On sharding open blockchains with smart contracts. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1357–1368. IEEE, 2020.

[W+14] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[Wan22] Jiaping Wang. Txilm: Lossy block compression with salted short hashing, June 21 2022. US Patent 11,368,286.

[WHH+19] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. A survey on consensus mechanisms and mining strategy management in blockchain networks. *Ieee Access*, 7:22328–22370, 2019.

[WLC+22] Qian Wei, Bingzhe Li, Wanli Chang, Zhiping Jia, Zhaoyan Shen, and Zili Shao. A survey of blockchain data management systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 21(3):1–28, 2022.

[WSNH19] Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 41–61, 2019.

[WWC+19] Mingli Wu, Kun Wang, Xiaoqin Cai, Song Guo, Minyi Guo, and Chunming Rong. A comprehensive survey of blockchain: From theory to iot applications and beyond. *IEEE Internet of Things Journal*, 6(5):8114–8154, 2019.

[WWZC21] Xiaoqing Wang, Chunping Wang, Kun Zhou, and Hongbing Cheng. Ess: An efficient storage scheme for improving the scalability of bitcoin network. *IEEE Transactions on Network and Service Management*, 19(2):1191–1202, 2021.

[WYCX20] Qin Wang, Jiangshan Yu, Shiping Chen, and Yang Xiang. Sok: Diving into dag-based blockchain systems. *arXiv preprint arXiv:2012.06128*, 2020.

[WYY+22] Huan Yu Wu, Xin Yang, Chentao Yue, Hye-Young Paik, and Salil S Kanhere. Chain or dag? underlying data structures, architectures, topologies and consensus in distributed ledger technology: A review, taxonomy and research issues. *Journal of Systems Architecture*, page 102720, 2022.

[XFL+21] Jiali Xing, David Fischer, Nitya Labh, Ryan Piersma, Benjamin C Lee, Yu Amy Xia, Tuhin Sahai, and Vahid Tarokh. Talaria: A framework for simulation of permissioned blockchains for logistics and beyond. *arXiv preprint arXiv:2103.02260*, 2021.

[XFRZ20] Mengtian Xu, Guorui Feng, Yanli Ren, and Xinpeng Zhang. On cloud storage optimization of blockchain with a clustering-based genetic algorithm. *IEEE Internet of Things Journal*, 7(9):8547–8558, 2020.

[XHC18] Zihuan Xu, Siyuan Han, and Lei Chen. Cub, a consensus unit-based storage scheme for blockchain system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 173–184. IEEE, 2018.

[XLZ+21] Minghui Xu, Chunchi Liu, Yifei Zou, Feng Zhao, Jiguo Yu, and Xiuzhen Cheng. wchain: a fast fault-tolerant blockchain protocol for multihop wireless networks. *IEEE Transactions on Wireless Communications*, 20(10):6915–6926, 2021.

[XWS+17] Xiwei Xu, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso, and Paul Rimba. A taxonomy of blockchain-based systems for architecture design. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 243–252. IEEE, 2017.

[XZLH20] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, 22(2):1432–1465, 2020.

[XZYS20] Xuesong Xu, Zhi Zeng, Shengjie Yang, and Hongyan Shao. A novel blockchain framework for industrial iot edge computing. *Sensors*, 20(7):2061, 2020.

[XZZ+22] Minghui Xu, Feng Zhao, Yifei Zou, Chunchi Liu, Xiuzhen Cheng, and Falko Dressler. Blown: a blockchain protocol for single-hop wireless networks under adversarial sinr. *IEEE Transactions on Mobile Computing*, 2022.

[YDXJ20] Wenhui Yang, Xiaohai Dai, Jiang Xiao, and Hai Jin. Ldv: A lightweight dag-based blockchain for vehicular social networks. *IEEE Transactions on Vehicular Technology*, 69(6):5749–5759, 2020.

[YLZ21] Bin Yu, Xiaofeng Li, and He Zhao. Pow-bc: A pow consensus protocol based on block compression. *KSII Transactions on Internet & Information Systems*, 15(4), 2021.

[YMRS19] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. *arXiv preprint arXiv:1906.11078*, 2019.

[YN21] Jiawei Yuan and Laurent Njilla. Lightweight and reliable decentralized reward system using blockchain. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2021.

[YZX+22] Li Yang, Yifei Zou, Minghui Xu, Yicheng Xu, Dongxiao Yu, and Xiuzhen Cheng. Distributed consensus for blockchains in internet-of-

things networks. *Tsinghua Science and Technology*, 27(5):817–831, 2022.

[ZABZ+19] Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J Knottenbelt. Sok: communication across distributed ledgers. 2019.

[ZHZB20] Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. Solutions to scalability of blockchain: A survey. *Ieee Access*, 8:16440–16455, 2020.

[ZLCD18] Qiuhong Zheng, Yi Li, Ping Chen, and Xinghua Dong. An innovative ipfs-based storage model for blockchain. In *2018 IEEE/WIC/ACM international conference on web intelligence (WI)*, pages 704–708. IEEE, 2018.

[ZMP+22] Hussein Zangoti, Alex Pissinou Makki, Niki Pissinou, Abdur R Shahid, Omar J Guerra, and Joel Rodriguez. A multidimensional blockchain framework for mobile internet of things. In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 935–944. IEEE, 2022.

[ZMR18] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 931–948, 2018.

[ZP23] Hussein Zangoti and Niki Pissinou. An efficient approach for merging multidimensional blockchains in mobile iot. In *The 19th Annual International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. IEEE, 2023. (In press).

[ZXD+18] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.

VITA

HUSSEIN ZANGOTI

	Born, Jazan, Saudi Arabia
2010	B.Sc., Computer Science Jazan University Jazan, Saudi Arabia
2015	M.S., Computer Science Monmouth University Long Branch, NJ, USA
2022	Doctoral Candidate, Computer Science Florida International University Miami, FL, USA

PUBLICATIONS AND PRESENTATIONS

1. Zangoti, H., Makki-Pissinou, A., Pissinou, N., Shahid, A.R., Guerra, O., and Rodriguez, J., (2022, December). A Multidimensional Blockchain Framework For Mobile Internet of Things. In *The 21st IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 935-944). IEEE.
2. Zangoti, H., Pissinou, N., (2023, June). An Efficient Approach for Merging Multidimensional Blockchains in Mobile IoT. In *The 19th Annual International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)* (In press). IEEE.
3. Zangoti, H., Khan, W. Z., Pissinou, N., (2023, April). CSBSO: Collective Signing-Based Blockchain Storage Optimization for Mobile Internet of Things. In *Distributed Ledger Technologies: Research and Practice* (Submitted). ACM.