

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

BLOCKCHAIN FOR SECURED IoT AND D2D APPLICATIONS OVER 5G CELLULAR NETWORKS

A THESIS BY PUBLICATIONS PRESENTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER AND ELECTRONICS ENGINEERING

MASSEY UNIVERSITY, ALBANY, NEW ZEALAND

Houshyar Honar Pajooh

November 2021

Authors Declaration

This thesis was produced according to Massey University's "PhD thesis by publication" guidelines. This thesis is based on research that has either been published or is currently under review in MDPI(Sensors), Springer (Journal of big data), and IEEE IoT journal. In accordance with Sensors, SpringerOpen, and IEEE's copyright policy, this thesis contains the accepted and published version of each manuscript as the final version. Consequently, the content is identical to the published versions. Furthermore, the work contained within this thesis was published in several different journals. Therefore, some of the submitted chapters are relatively succinct, there is some repetition (particularly in the literature review of some chapters), and there are minor stylistic differences between the chapters themselves.

Houshyar Honar Pajoo

November, 2021

Copyright Statement

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Houshyar Honar Pajoo
November, 2021

Authenticity Statement

I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis.

Houshyar Honar Pajoo
November, 2021

Abstract

The Internet of things (IoT) is in continuous development with ever-growing popularity. It brings significant benefits through enabling humans and the physical world to interact using various technologies from small sensors to cloud computing. IoT devices and networks are appealing targets of various cyber attacks and can be hampered by malicious intervening attackers if the IoT is not appropriately protected. However, IoT security and privacy remain a major challenge due to characteristics of the IoT, such as heterogeneity, scalability, nature of the data, and operation in open environments. Moreover, many existing cloud-based solutions for IoT security rely on central remote servers over vulnerable Internet connections. The decentralized and distributed nature of blockchain technology has attracted significant attention as a suitable solution to tackle the security and privacy concerns of the IoT and device-to-device (D2D) communication. This thesis explores the possible adoption of blockchain technology to address the security and privacy challenges of the IoT under the 5G cellular system.

This thesis makes four novel contributions. First, a Multi-layer Blockchain Security (MBS) model is proposed to protect IoT networks while simplifying the implementation of blockchain technology. The concept of clustering is utilized to facilitate multi-layer architecture deployment and increase scalability. The K -unknown clusters are formed within the IoT network by applying a hybrid Evolutionary Computation Algorithm using Simulated Annealing (SA) and Genetic Algorithms (GA) to structure the overlay nodes. The open-source Hyperledger Fabric (HLF) Blockchain platform is deployed for the proposed model development. Base stations adopt a global blockchain approach to communicate with each other securely. The quantitative arguments demonstrate that the proposed clustering algorithm performs well when compared to the earlier reported methods. The proposed lightweight blockchain model is also better suited to balance network latency and throughput compared to a traditional global blockchain.

Next, a model is proposed to integrate IoT systems and blockchain by implementing the permissioned blockchain Hyperledger Fabric. The security of the edge computing devices is provided by employing a local authentication process. A lightweight mutual authentication and authorization solution is proposed to ensure the security of tiny IoT devices within the ecosystem. In addition, the proposed model provides traceability for the data generated by the IoT devices. The performance of the proposed model is validated with practical implementation by measuring performance metrics such as transaction throughput and latency, resource consumption, and network use. The results indicate that the proposed platform with the HLF implementation is promising for the security of resource-constrained IoT devices and is scalable for deployment in various IoT scenarios.

Despite the increasing development of blockchain platforms, there is still no comprehensive method for adopting blockchain technology on IoT systems due to the blockchain's limited

capability to process substantial transaction requests from a massive number of IoT devices. The Fabric comprises various components such as smart contracts, peers, endorsers, validators, committers, and Orderers. A comprehensive empirical model is proposed that measures HLF's performance and identifies potential performance bottlenecks to better meet blockchain-based IoT applications' requirements. The implementation of HLF on distributed large-scale IoT systems is proposed. The performance of the HLF is evaluated in terms of throughput, latency, network sizes, scalability, and the number of peers serviceable by the platform. The experimental results demonstrate that the proposed framework can provide a detailed and real-time performance evaluation of blockchain systems for large-scale IoT applications.

The diversity and the sheer increase in the number of connected IoT devices have brought significant concerns about storing and protecting the large IoT data volume. Dependencies of the centralized server solution impose significant trust issues and make it vulnerable to security risks. A layer-based distributed data storage design and implementation of a blockchain-enabled large-scale IoT system is proposed to mitigate these challenges by using the HLF platform for distributed ledger solutions. The need for a centralized server and third-party auditor is eliminated by leveraging HLF peers who perform transaction verification and records audits in a big data system with the help of blockchain technology. The HLF blockchain facilitates storing the lightweight verification tags on the blockchain ledger. In contrast, the actual metadata is stored in the off-chain big data system to reduce the communication overheads and enhance data integrity. Finally, experiments are conducted to evaluate the performance of the proposed scheme in terms of throughput, latency, communication, and computation costs. The results indicate the feasibility of the proposed solution to retrieve and store the provenance of large-scale IoT data within the big data ecosystem using the HLF blockchain.

Acknowledgement

Foremost, I would like to express my indebtedness and render my warmest thanks to my supervisor Dr. Mohammad Abdur Rashid for his continuous support to my Ph.D. study and research with his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me all the time during the research and writing of this thesis. Without his encouragement and advice, I could not have accomplished all that I have at Massey University.

I would also like to thank Associate Prof. Fakhrul Alam for extended discussions and valuable suggestions, which have contributed significantly to the improvement of the thesis. I am sincerely grateful to Prof. Serge Demidenko for providing valuable feedback during my study. I will be forever thankful for their time and insightful and helpful comments and guidance. Under their guidance, I became a better researcher, a better engineer, and a better person.

This thesis has been written during my stay at the Department of Mechanical and Electrical engineering within the School of Food and Advanced Technology, of the Massey University. I would like to thank the department for providing excellent working conditions and their support.

I am wholeheartedly happy to have got an opportunity to pursue Ph.D. at the Massey University, New Zealand. Studying at Massey University with a full scholarship was one of the most important phases of my life that helped me gain invaluable knowledge and experiences.

Last but not the least, this research would not have been possible without the support of my family. I am eternally indebted to my parents for their endless support and love.

Publications and Presentations

List of Publications

- M. A. Rashid and H. H. Pajoo, "A Security Framework for IoT Authentication and Authorization Based on Blockchain Technology," 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), 2019, pp. 264-271, doi: 10.1109/TrustCom/BigDataSE.2019.00043.
- H. Honar Pajoo, M. Rashid, F. Alam, and S. Demidenko, "Multi-Layer Blockchain-Based Security Architecture for Internet of Things," Sensors, vol. 21, no. 3, p. 772, Jan. 2021 [Online]. Available: <http://dx.doi.org/10.3390/s21030772>.
- H. Honar Pajoo, M. Rashid, F. Alam, and S. Demidenko, "Hyperledger Fabric Blockchain for Securing the Edge Internet of Things," Sensors, vol. 21, no. 2, p. 359, Jan. 2021 [Online]. Available: <http://dx.doi.org/10.3390/s21020359>.
- Honar Pajoo, H., Rashid, M.A., Alam, F. et al. IoT Big Data provenance scheme using blockchain on Hadoop ecosystem. J Big Data 8, 114 (2021). [Online]. Available: <https://doi.org/10.1186/s40537-021-00505-y>

Under Review:

- H. Honar Pajoo, M. Rashid, F. Alam, and S. Demidenko, "Experimental Performance Analysis of Scalable Distributed Hyperledger Fabric in a Large Scale IoT Testbed", IEEE IoT Journal, 2021

Oral presentations:

- Application of Blockchain in IoT Security and Privacy. Oral presentation: IEEE Wireless Workshop, Waikato University, Hamilton, New Zealand, 2019.
- Blockchain Application for Industry 4.0 Smart Factory and IIoT. Invited talk: 70Th LASRA Conference, Napier, New Zealand, 2019.

Contents

Abstract	iii
Acknowledgement	v
Publications and Presentations	vi
Contents	vii
List of Figures	xii
List of Tables	xvi
1. Introduction	1
1.1. Motivation.....	5
1.2. Thesis Overview.....	7
1.3. Thesis Outline.....	9
2. The Blockchain Technology for the IoT	11
2.1. What is Blockchain?.....	11
2.1.1. How Does It Work?.....	13
2.1.2. Hashing.....	14
2.1.3. The Block and Mining.....	14
2.1.4. Securing Data.....	17
2.1.5. Blockchain Implementation for the Internet of Things.....	20
2.2. Ethereum and DAPPS.....	21
2.2.1. Distributed Applications (DAPPS).....	21
2.2.2. Transactions.....	22
2.2.3. Smart Contracts.....	22
2.3. Blockchain Consensus Algorithms.....	23
2.4. Hyperledger Fabric.....	27
2.4.1. Distributed Ledger Technology for Business Applications.....	27
2.4.2. Assets, ChainCode and Ledger.....	29

2.4.3.	Membership Services and Permissioned Network.....	30
2.4.4.	Nodes and Channels.....	31
2.4.4.1.	Ledger Implementation.....	33
2.4.5.	Peers Nodes, Anchors, and Endorsers.....	36
2.4.6.	Orderer Nodes.....	38
2.4.7.	Membership Servers and Certification Authority.....	40
2.4.8.	ChainCode Development.....	41
2.5.	Hyperledger Fabric for Cellular-based IoT.....	42
2.5.1.	Hyperledger Fabric Components.....	42
2.5.2.	Membership Service Provider (MSP).....	43
2.5.3.	Fabric Certification Authority.....	43
2.5.4.	Ordering Service.....	43
2.5.5.	Peer Nodes.....	44
2.5.6.	Permissioned Network.....	44
2.5.7.	Confidential Transactions.....	44
2.5.8.	Hyperledger Fabric Policies.....	45
3.	Multi-Layer Blockchain-Based Security Architecture for Internet of Things	46
	Abstract.....	47
3.1.	Introduction.....	47
3.2.	Related Works.....	49
3.2.1.	Authentication and Authorization in IoT.....	49
3.2.2.	Blockchain-Based Frameworks for IoT Security and Privacy.....	49
3.2.3.	Permissioned Blockchain in IoT.....	50
3.2.4.	Layer-Based IoT Blockchain.....	50
3.3.	Multi-Layer Security Framework.....	52
3.3.1.	LAYER-1.....	54
3.3.2.	LAYER-2.....	54
3.3.3.	LAYER-3.....	56
3.4.	Framework Implementation.....	57
3.4.1.	Network Self-Clustering.....	57

3.4.1.1.	GA Phase: CH Selection with Genetic Algorithm.....	58
3.4.1.2.	Optimization of Distance and Coverage by GA.....	60
3.4.1.3.	Network Changes Optimization Using SA.....	60
3.4.1.4.	Clustering Results.....	61
3.4.2.	Blockchain Implementation.....	62
3.4.2.1.	Development Environment.....	62
3.4.2.2.	Smart Contract for Modeling Transactions.....	63
3.4.3.	Performance Evaluation.....	64
3.5.	Security Analysis of the Framework.....	66
3.5.1.	Framework Privacy.....	67
3.5.2.	Heterogeneity and Flexibility.....	67
3.5.3.	Authentication.....	67
3.5.4.	Scalability.....	67
3.6.	Conclusions.....	67
3.7.	References.....	69
4.	Hyperledger Fabric Blockchain for Securing the Edge Internet of Things	73
	Abstract.....	74
4.1.	Introduction.....	74
4.2.	Related Work.....	76
4.2.1.	Overview.....	76
4.2.2.	IoT Blockchain.....	77
4.2.3.	Blockchain for Mobile Edge Computing.....	78
4.2.4.	Blockchain for Data Sharing and Traceability.....	78
4.3.	Blockchain Overview.....	79
4.3.1.	Consensus Algorithm.....	79
4.3.2.	Smart Contracts.....	81
4.4.	Hyperledger Fabric IoT System Model.....	81
4.4.1.	Overall Design.....	81
4.4.2.	Multi-Layer IoT Blockchain Network.....	81
4.4.2.1.	Layer-1.....	81

4.4.2.2.	Layer-2.....	81
4.4.2.3.	Layer-3.....	82
4.4.2.4.	Layer-4.....	83
4.4.3.	Local Authentication and Authorization of IoT Devices in Layer-1	83
4.4.4.	Secured IoT Blockchain for Edge Computing Nodes in Layer-2.....	85
4.4.4.1.	Nodes in IoT Edge Hyperledger.....	85
4.4.4.2.	ChainCode in IoT Edge.....	86
4.4.4.3.	Certificate Authority.....	87
4.4.4.4.	Ledger Implementation.....	87
4.4.5.	Base Station Nodes with High Computational Power in Layer-3.....	88
4.4.6.	Layer-4 Off-Chain Storage.....	88
4.5.	Performance Evaluation.....	89
4.5.1.	Experimental Setup and Implementation.....	89
4.5.2.	System Configurations.....	90
4.5.3.	Transaction Throughput.....	90
4.5.3.1.	Desktop Measurements.....	91
4.5.3.2.	Raspberry Pi Measurements.....	91
4.5.4.	Transactions Latency.....	92
4.5.5.	Resource Consumption.....	93
4.5.6.	CPU and Memory Use Measurements.....	93
4.5.6.1.	Desktop Setup.....	94
4.5.6.2.	Raspberry Pi Setup.....	95
4.5.7.	Network Use Measurements.....	96
4.6.	Conclusions.....	98
4.7.	References.....	99
5.	Experimental Performance Analysis of Scalable Distributed HLF in a Large	
	Scale IoT Testbed	103
	Abstract.....	104
5.1.	Introduction.....	104

5.2. Related Works.....	106
5.3. Hyperledger Fabric.....	107
5.3.1. Transaction Flow in Hyperledger Fabric.....	107
5.3.1.1. Phase 1: Endorsement Phase.....	108
5.3.1.2. Phase 2: Ordering Phase.....	108
5.3.1.3. Phase 3: Validation Phase.....	108
5.4. Configuration Parameters and Key Metrics.....	108
5.4.1. Key Parameters Definition.....	109
5.4.2. Performance Metrics.....	109
5.4.2.1. Transaction Throughput.....	109
5.4.2.2. Transaction Latency.....	109
5.4.2.3. Network Size and Scalability.....	109
5.4.2.4. Block Size.....	109
5.4.3. Test Environment.....	109
5.5. Results and Discussion.....	110
5.5.1. Impact of Transaction Sending Rate - Single Host.....	110
5.5.2. Impact of Transaction Sending Rate - Multiple Host.....	111
5.5.3. Impact of Endorsement Policy.....	112
5.5.4. Impact of Block Size - Multiple Host.....	112
5.5.5. Impact of Network Size.....	113
5.5.6. Resource Consumption.....	113
5.5.7. Summary of Performance Analysis.....	114
5.6. Conclusion.....	115
5.7. References.....	115
6. IoT Big Data provenance scheme using blockchain on Hadoop ecosystem	117
Abstract.....	118
6.1. Introduction.....	118
6.2. Background.....	122
6.2.1. Blockchain.....	122

6.2.2. Big Data systems.....	123
6.2.2.1. Hadoop ecosystem.....	124
6.3. Related works.....	125
6.3.1. Blockchain-based data provenance in IoT.....	125
6.3.2. Blockchain-based data verification.....	126
6.4. System model and architecture.....	127
6.4.1. The blockchain-based high-level scheme.....	128
6.4.2. Hyperledger framework.....	128
6.4.3. Hadoop data storage.....	130
6.4.4. ChainCode.....	130
6.4.5. Client library.....	130
6.4.6. Edge computing.....	131
6.5. System implementation.....	132
6.5.1. Experiment setup.....	133
6.6. Results and discussions.....	134
6.6.1. Throughput and response time measurements.....	134
6.6.2. Large scale IoT environment evaluations.....	137
6.6.3. Data provenance and tracking resource consumption.....	137
6.7. Conclusion.....	141
6.8. References.....	142
7. Conclusion and Future Research Directions	144
7.1. Conclusions.....	144
7.2. Future Research Directions.....	150
A Appendix: A	152
Additional References	157

List of Figures

1.1	The total size of the blockchain minus database indexes.....	6
2.1	Simplified Block Structure.....	13
2.2	A block structure in Blockchain.....	15
2.3	Structure of a Block and Block Header.....	16
2.4	Merkle Tree Structure.....	17
2.5	Simple blockchain forking.....	20
2.6	HLF components and elements.....	32
2.7	Ledger implementation.....	34
2.8	Peers Nodes, Anchors, and Endorsers.....	36
2.9	Peers Nodes, Anchors, and Endorsers.....	36
2.10	The high-level overview of the proposed architecture.....	45
4.1	Multi-Layer model for Internet of Things (IoT) network.....	53
4.2	The network model based on a multi-layer structure.....	53
4.3	Local authorization service.....	54
4.4	Overall network architecture the IoT-enabled cellular system.....	55
4.5	Network clustering scheme for cellular IoT network.....	57
4.6	Genetic Algorithm-GA-Simulated Annealing.....	58
4.7	GA algorithm pseudocode.....	59
4.8	SA Algorithm pseudocode.....	60
4.9	Performance of the proposed clustering algorithm.....	63
4.10	The implementation structure of the blockchain IoT framework.....	64

4.11	Latency vs. transaction sending rate.....	66
4.12	Throughput vs. transaction sending rate.....	66
5.1	Overview of the RAFT consensus protocol and block creation.....	80
5.2	Conceptual framework of the integrated IoT blockchain platform.....	82
5.3	Local authentication flow.....	84
5.4	Blockchain-based edge services.....	85
5.5	Proposed HLF network transaction flow.....	86
5.6	Ledger implementation flow.....	88
5.7	Experimental setup and system under test.....	90
5.8	Effects of transaction sizes - Desktop setup.....	91
5.9	Effects of transaction sizes - Raspberry Pi setup.....	92
5.10	Latency for all ChainCode operation.....	93
5.11	Latency vs. transaction sending rate.....	94
5.12	CPU and memory usage - Peer - Desktop setup.....	95
5.13	CPU and memory usage - client - Desktop setup.....	95
5.14	CPU and memory usage - Peer - RPi setup.....	96
5.15	CPU and memory usage - client - RPi setup.....	96
5.16	Network use for peer process with no transactions.....	97
5.17	Network use vs. load sizes with/without external storage.....	97
6.1	HLF network system architecture with major components.....	107
6.2	Fabric transaction flow.....	108
6.3	HLF-based distributed system model.....	108
6.4	Experimental setup and components for performance evaluation.....	110
6.5	Transactions sending rate vs. throughput – single Host.....	111
6.6	Transactions sending rate vs latency- single Host.....	111
6.7	Transactions sending rate vs throughput for multiple host arrangement.....	111

6.8	Transactions sending rate vs latency for multiple host arrangement.....	111
6.9	Transactions sending rate vs throughput for various endorsement policies.....	112
6.10	Transactions sending rate vs latency for various endorsement policies.....	112
6.11	Impact of block sizes on system throughput.....	113
6.12	Impact of block sizes on transactions latency.....	113
6.13	Impact of network size on the system throughput.....	113
6.14	Impact of network size on transactions latency.....	113
6.15	Peers average CPU usage.....	114
6.16	Peers average disk write usage.....	114
6.17	Peers average memory consumption.....	114
6.18	Peers average network traffic In.....	114
6.19	Peers average network traffic Out.....	114
7.1	The HDFS system architecture.....	124
7.2	Detail operations and system structure of MapReduce.....	125
7.3	Proposed blockchain-enabled secure big data provenance scheme.....	129
7.4	Local mutual authentication procedure at IoT edge computing.....	131
7.5	Throughput and response time for various batch sizes.....	135
7.6	Latency vs. transaction sending rate (5 Peers 10 Blocks).....	136
7.7	Latency vs. transaction sending rate (5 Peers 50 Blocks).....	136
7.8	Large scale IoT system throughput of edge gateways.....	138
7.9	IoT gateway and peer node CPU utilization.....	138
7.10	Peer process CPU and Memory utilization.....	139
7.11	Client application CPU and Memory usage.....	139
7.12	Client application profiling.....	140
7.13	Network utilization for peer process with 100 models of 100 MBs.....	140

List of Tables

2.1	Existing consensus algorithm in current blockchain technologies.....	26
2.2	Comparison of well-known blockchain implementation and their key features.....	28
3.1	GAs Parameter Settings.....	61
3.2	SA Parameter Settings.....	61
3.3	Hyperledger and Ethereum performance metric summary.....	65
3.4	Security challenge comparison of blockchain applications in IoT systems.....	66
4.1	Statistics analysis of SET ChainCode latency.....	93
5.1	The system under test parameters and metrics.....	110
5.2	Network nodes and load sizes.....	110
6.1	Raspberry hardware specifications.....	133
6.2	Hadoop cluster experimental setup and specifications.....	134

Abbreviations

ASLPR	Application-Specific Low Power Routing
BCS	Blockchain Structures
BFT	Byzantine Fault Tolerant
BS	Base Station
CA	Certification Authority
CC	ChainCode
CH	Cluster Head
CID	Client Identity
CLI	Command Line Interface
CoAP	Constrained Application Protocol
CPSs	Cyber-Physical Systems
CPU	Central Processing Unit
CRL	Certificate Revocation List
D2D	Device-to-Device
dApps	distributed Applications
DDoS	Distributed Denial-of-Service
DHE	Diffie-Hellman Ephemeral
DLTS	Distributed Ledger Technologies
DPoS	Delegated Proof of Stake
EC	Evolutionary Computation
ECC	Elliptic Curve Cryptography
ERA	Energy-aware Routing Algorithm
FSFLA	Fuzzy Shuffled Frog Leaping Algorithm

GA	Genetic Algorithm
GAPSO	Genetic Algorithm and Particle Swarm Optimization
gPRC	google Remote Procedures Calls
HLF	Hyperledger Fabric
IDE	Integrated Development Environment
IIoT	Industrial IoT
IoT	Internet of Things
LHB	Lightweight Hyperledger Blockchain
LSB	Lightweight Scalable Blockchain
LTS	Long Term Support
MAC	Message Authentication Code
MCNs	Multihop Cellular Networks
MSP	Membership Service Providers
NFS	Network File Systems
OPM	Open Provenance Model
OS	Ordering Service
PBFT	Practical Byzantine Fault Tolerance
PKI	Public Key Infrastructure
PoA	Proof of Authority
PoB	Proof of Bandwidth
PoBT	Proof of Block and Trade
PoC	Proof of Concept
PoET	Proof of Elapsed Time
PoS	Proof of Stake
PoW	Proof of Work
RoA	Rating of Allocation
RPi	Raspberry Pi
RSA	Rivest–Shamir–Adleman
SA	Simulated Annealing
SDK	Software Development Kit

SDN	Software Defined Networking
SFLA	Shuffled Frog Leaping Algorithm
SI	Swarm Intelligence
SSL	Secure Sockets Layer
SUT	System Under the Test
TLS	Transport Layer Security
TSL	Transport Layer Security
TSP	Transactions Per Second
WSNs	Wireless Sensor Networks

Chapter 1

Introduction

The rapid evolution of the Internet changed its paradigm from a traditional interconnected network of computing devices to an environment that virtually connects every “thing.” The Internet of Things (IoT) is a ubiquitous network of devices and computers that facilitate communication and data exchange with users or interconnected devices. Smart IoT devices require connectivity to external services to perform basic functionalities. This growing connectivity offers real-time and customized services for IoT users, including many smart appliances. The Service Providers (SPs) collect the data sensed by IoT devices in centralized cloud servers for further data processing and provides users with personalized services. Since the IoT interacts with humans, machines, and environments, failures or data corruptions in the IoT systems can lead to severe consequences.

The IoT faces various challenges in enabling secure, safe, and scalable ecosystems. Most IoT platforms rely on external cloud service providers. Cloud dependency poses numerous security and privacy risks. The collected data from IoT devices ranges from privacy-sensitive data (such as healthcare data) to personal details (habits and social life). Many IoT devices lack fundamental security protections. Besides, various security challenges are due to the intrinsic features of IoT networks: devices and vendors heterogeneity, network scale, lack of central control, multiple attack surfaces, situational and context-aware risks. Numerous security exposures are identified in connected smart devices ranging from smart locks [1] to smart vehicles [2]. A massive distributed denial of service (DDoS) attack left much of the websites inaccessible due to poorly secured IoT devices in October 2016 [3]. The remarkable point was that many of the

compromised devices launching the attack were relatively small computing devices consisting of webcams, residential gateways, printers, home appliances, i.e., the IoTs. These highlights mandate the need for robust security solutions for IoT ecosystems. Integrating smart and connected IoT devices into our everyday life requires providing security safeguards for IoT devices and networks.

The sheer amount of generated data as well as increasing pervasive and dissemination of data collection raise serious concern about security and privacy. The lack of basic security protections exacerbates the privacy vulnerabilities of many first-generation IoT products. Also, this data can be implemented in many customized and sophisticated services to provide useful utilities for end users. However, it can be deployed to construct algorithms that can reveal user activities pattern, private information, and personalized lifecycle data. For example, energy companies can track their consumers' energy consumption patterns and build a virtual profile of the users that may compromise their privacy. The traditional Internet deploys a well-developed security protocol such as The SSL/TLS (Secure Socket Layer/Transport Layer Security) [4]. However, the new wireless and cellular networks have unique characteristics that distinguish them from the conventional Internet and need new robust solutions to address the security challenges. The IoT settings need a single integrated solution to support the heterogeneous requirements ranging from safety-critical systems to sensor nodes. Most of the existing security measures, including Kerberos, SSL/TLS, and widely-used solutions for Wireless Sensor Network (WSN) and Mobile Ad hoc Network (MANET), are mainly developed for homogeneous networks. They may not be directly applicable in heterogeneous IoT ecosystems.

The aforementioned challenges have been explored in literature with some suggested solutions. An access control mechanism is proposed in [5] built on distributed capability-based access control to protect sensitive information. The research in [6] implemented IP-sec and TLS to provide authentication and privacy. The authors in [7] proposed a privacy management mechanism to investigate the risk of disclosing data. The proposed method allowed IoT users to decide on whether to share data with SPs or not. Provide a biometric-based security solution for IoT is proposed in work [8]. The authentication and authorization solution is presented in [9] based on distributed gateways. The proposed framework provides remote authentication and authorization for IoT devices.

It is found that the following challenges are not fully addressed by the existing security and

privacy solutions in the literature.

1. The IoT systems are often heterogeneous with the diversity in resource-constraint devices and intermittent connectivity. This heterogeneous ecosystem is diverse in terms of security requirements and resource availability within the IoT systems.
2. The complex security solutions can not be implemented on IoT devices with restricted resources, including computation resources, memory, and bandwidth. The robust solution should consider these limitations [10].
3. The existing IoT networks are mainly based on a centralized authentication server to provide IoT devices identification, authentication, and authorization. The IoT ecosystems are connected through cloud servers for communication. This model fails to address the scalability of the massive number of connected IoT devices. Besides, the single point of failure vulnerability is a significant drawback of such systems that can impact the entire network [10].
4. The performance of the lightweight scalable blockchain approach for IoT over 5G cellular system can be enhanced through implementing a multi-layer framework. The multi-layer structure needs to be further explored through implementing an improved clustering algorithm suitable for IoT ecosystem.
5. The existing solutions fail to address the challenges associated with the data provenance of large data in IoT systems with the concerns about the low resource consumption. The robust solution needs to explore blockchain technology's capability for provenance tracking in the big data system environment.
6. Due to the open environment nature of the IoT operation, adversaries have physical access to IoT devices and platforms and/or through wireless, increasing the potential risks.

Blockchain technology has attracted significant attention in recent years to improve IoT's security, privacy, anonymity, auditability, and reliability. Blockchain is an auditable, immutable, and timestamp ledger of blocks used to store and share data in a distributed structure [11]. The stored data includes an extensive range such as payment history (Bitcoin [12]), a contract [13], personal data [14], or commodity trading data [15,16]. All participating nodes have a replica of the distributed digital shared ledger of transactions. The blockchain ensures high auditability

by storing all transactions in the shared ledger. All participants need to verify and confirm the addition of new transactions into the blockchain ledger.

Miners are the particular nodes in the network that perform adding newly generated transactions to a pending transactions pool. The size of each block and the number of transactions within each block are predefined. Therefore the miner nodes embed all pending transactions into a block when the specific predefined size is reached. The process of appending a newly generated block into the blockchain ledger is called mining. The mining is based on a consensus algorithm that guarantees blockchain security against malicious miners and introduces randomness. The Proof of Work (PoW) [17] and Proof of Stake (POS) [13] are the most popular consensus algorithms among existing blockchain technologies. The PoW consensus process requires significant computational resources. Miner nodes with high capability and more computational resources most likely solve the consensus algorithm and mine the next block. On the other hand, the miners in the PoS lock the asset or stack to mine the new block in the blockchain. Miners that have the higher stack mine the next block. Each user needs to be identified by a Public Key (PK) to create transactions to ensure a high anonymity level.

Hyperledger Fabric (HLF) or simply Fabric [18] was first introduced in 2017 to create a private and permissioned blockchain platform as one of the projects of Hyperledger under the auspices of the Linux Foundation [19]. Preceding permissioned blockchain platforms suffer from many restrictions linked to their permissionless nature or inherited from deploying the order-execute architecture. In particular: hard-coded consensus protocol, trust model, a smart contract written in fixed and non-standard languages, limited performance due to sequential execution of all transactions by all peers, confidentiality, and issues with programmatically deterministic transactions. HLF is an open-source blockchain platform that overcomes the aforementioned limitations. The Fabric has been used widely across various industries and use-cases and has been deployed in more than 400 prototypes, proofs-of-concept, and production distributed-ledger systems. The work in [20] proposed a new framework using smart contracts for access management of IoT devices that enhance the overall IoT ecosystem security. Blockchain technology is used as an underlying network to provide security and privacy for IoT-enabled services [21]. However, there are issues with applying the existing blockchain solutions directly in the IoT ecosystem, which are discussed in the following section.

1.1 Motivation

As outlined earlier, blockchain technology has attracted many researchers to adopt this emerging technology to enhance security, privacy, and anonymity of the IoT and D2D ecosystems. However, integrating the existing blockchain platforms into the IoT and D2D context is challenging and encounter the following problems:

- *Throughput:* Throughput is determined as the total number of stored transactions per second (TPS). The throughput in the conventional blockchain platform is limited. This number for Bitcoin is about 7 transactions per second, while the number of transactions per second generating from massive IoT devices is extensive and exceeds such limitations.
- *Latency:* There is a delay linked to confirmation of a transaction by all nodes in the network. The time for transaction confirmation in Bitcoin is up to 30 minutes, and in Ethereum, it takes up to 30 seconds. There are specific limits for the delay requirements due real-time nature of most IoT applications, and the service providers have restricted it.
- *Overheads and Network Scalability:* Since all nodes hold a copy of the blockchain ledger, all nodes maintain the block verification process for broadcasted blocks in the typical blockchain implementation. This arises scalability issues since the processing overhead of a large amount of broadcasted traffic would surge dramatically with the number of network participants. However, many IoT devices are limited in terms of their bandwidth and processing capabilities. Furthermore, to ensure the integrity of the blockchain, a newly added node needs to download a copy of the entire ledger and checks for the validity of the whole chain.
- *Complexity of consensus algorithms :* Most blockchain consensus algorithms require a considerable amount of resources from participating nodes that are not feasible to be applied directly on resource constraint IoT devices.
- *Blockchain memory requirements:* The massive number of IoT devices generates a vast number of transactions requiring a considerable memory footprint at the blockchain nodes to store the blockchain ledger. The current Bitcoin blockchain includes more than 620 million transactions, requires more than 330GB of storage space [22]. With the rapid

growth of the IoT ecosystem, the number of transactions exceeds the Bitcoin blockchain. Figure 1.1 shows the total size of the blockchain minus database indexes in gigabytes.

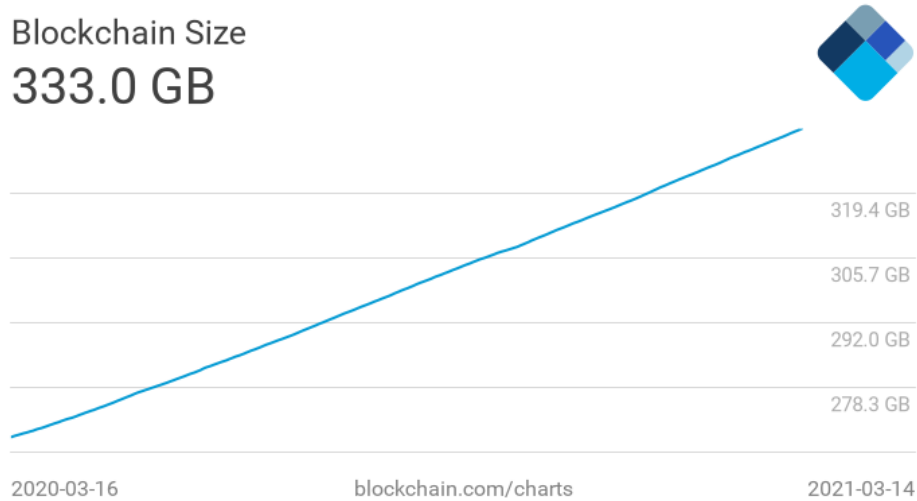


Figure 1.1: The total size of the blockchain minus database indexes.

- *Storage requirement of IoT applications:* Most IoT applications require diverse storage options to store their collected data. Besides, they need to keep the data for a specific period. Smart IoT devices collect data and send it to service provider (SP) for further processing and storage. Therefore, the related transactions only valid for that particular period. In some cases, progressive actions can be summarized into a specific transaction representing the entire process. Hence, the current blockchain solutions don't offer this flexibility, and they store transactions in an immutable and permanent fashion that can not be altered.
- *Limited privacy:* All data in the blockchain network is publicly visible to all participants in the current blockchain platforms. User privacy disclosure risk is high in the blockchain as all user devices permanently store the transaction in the blockchain. A linking attack happens when multiple linked transactions are generated by the same user that causes identity deanonymization. The interaction between the IoT device and SPs can be compromised by monitoring the frequency that has been used to record a transaction by a user [23]. Permissioned or private blockchain solutions limit the disclosure range. In order to cryptographically protect the data and achieve privacy, additional layers, such as zero-knowledge proofs or a commitment scheme, are required .
- *Key Management:* Most IoT applications need to authorize the user before generating

transactions. The key management process facilitates issuing the corresponded public and private Keys (PK-) for authentication and authorization. Users can update the public keys(PK+) to enhance security and privacy. Therefore, managing and storing the keys for multiple IoT devices on a large scale is a complicated and time-consuming process.

1.2 Thesis Overview

Blockchain can provide solutions to address the security challenges faced by the centralized cloud-based IoT and the issues related to the reliability, privacy, and management of the IoT and D2D communication devices. This research work considers the limitations related to the constrained resources of the IoT and mobile D2D devices because the conventional blockchain solution uses a consensus algorithm demanding a significant amount of computing power and leading to considerable network traffic. This research evaluates models that facilitate access to IoT resources using trusted and secure communication through implementing smart contracts. The advantages of using the proposed blockchain approach are immutability, decentralized trust, high availability, and transparency. The main contribution of this thesis is adoption of the blockchain technology for the IoT ecosystem over 5G cellular networks by addressing the challenges mentioned in the previous section.

Furthermore, it investigates the deployment of the P2P network blockchain for D2D technology, which allows direct communication of mobile devices within the 5G networks. This thesis considers a private blockchain to provide high security and lightweight authentication for lightweight IoT devices to restrict and manage participants continuously.

This thesis has made the following contributions:

Chapter 1 demonstrates the research motivation and shows the challenges that are not fully addressed by existing security and privacy solution. It explores the existing blockchain solutions and the feasibility of implementing blockchain technology for IoT, focusing on Hyperledger Fabric (HLF) blockchain in Chapter 2.

Chapter 3 proposed and developed a Multi-Layer Blockchain-Based security architecture for IoT. The new network model based on multi-layer distributed blockchain can be regarded as an organic combination of blockchain technology and clustering techniques that effectively utilize

network clustering performance and capabilities and significantly improve the overall security and reliability of the IoT ecosystem. Clustering is one of the critical steps of implementing multi-layer architecture. Therefore, a new network clustering method is applied. It is based on the evolutionary computation that deploys multi-objective fitness functions relevant to heterogeneous IoT networks. The decentralized, fast, and self-clustering mechanism divides the IoT network into clusters while considering node mobility.

Chapter 4 shows how Hyperledger Fabric Blockchain can be implemented for securing the edge IoT. A blockchain-enabled edge computing approach is considered and implemented for the IoT network with an open-source HLF blockchain platform. A layer-wise security architecture is designed according to different nodes and functionality capabilities to fit the scalable IoT applications. The infrastructure includes Base Stations (BS), Cluster Heads (CH), and IoT devices facilitating access control policies and management. Mutual authentication and authorization schemes for IoT devices are proposed and implemented to ensure the security of the interconnected devices in the scalable IoT platform. An HLF blockchain middle-ware module embedded in the IoT gateways ensures secure data transactions for the IoT distributed applications. Off-chain data storage and blockchain distributed data storage are employed to support data traceability.

Experimental performance analysis of scalable distributed HLF in a large scale IoT testbed is presented in Chapter 5. Performance computation and evaluation represent significant challenges for current blockchain systems, particularly during complex smart contract execution. The Fabric network is orchestrated by various components, including endorsers, ordering services, and committers. It constitutes different transaction processing phases: the endorsement phase, ordering phase, validation, and commit phase. Therefore, the Fabric encompasses configurable parameters, such as block size, channels, endorsement policy, and state databases. Finding the right set of values for this range of parameters is the main challenge in adapting an efficient blockchain system that has been explored in this chapter.

Chapter 6 Proposed and measured the performance of IoT big data provenance scheme using Blockchain on the Hadoop ecosystem. The IoT systems face challenges in performing various identity management, maintaining the trustworthiness of data, providing access control to numerous data within the network, and detecting abnormal behaviors. Data provenance is a solution to tackle the challenges mentioned above in IoT by recording information about data

operations, data origins and analyzing the data history from its source to the current state. Embedding the data provenance enriched by blockchain technology into big data applications enhances system security and privacy while ensures data availability. The blockchain-enabled data provenance mechanism for big data applications in IoT systems guarantees data verifiability and integrity since the data operations are recorded in the form of the transaction by every block in the blockchain network. A provenance mechanism is applicable to record the origin of multiple sensor data to meet these concerns. The blockchain-based provenance system's scalability is enhanced by implementing the high capacity of big data systems such as the Hadoop Distributed File System (HDFS). A model is designed to integrate the blockchain technology and IoT big data system incorporating edge computing in a large-scale IoT environment. The goal is to provide the data provenance, integrity, traceability, and accountability for large-volume of generated data by IoT devices and store it in a secure and verifiable big data ecosystem.

Chapters 3,4,5,and 6 are peer-reviewed research publications and have self-contained literature review that establish the state-of-the-art research works and identify the research gaps in the literature. Therefore, there is no traditional literature review chapter in this thesis.

1.3 Thesis Outline

The outline of the thesis is as follows:

- **Chapter 1** provides the motivation of the research, research goals, research contributions and the outline of the thesis.
- **Chapter 2** presents an overview of the blockchain technology for IoT as the background knowledge for the thesis. An overview of the Ethererum is provided, along with the blockchain technology implementation for the IoT ecosystem. Furthermore, this chapter contains a comprehensive description of the Hyperleger Fabric blockchain as a suitable solution for the IoT environment.
- **Chapter 3** proposes and develops a multi-layer blockchain-based security architecture for IoT. In this chapter, the framework architecture and the multi-layer system model are detailed. It also provides the proposed IoT blockchain framework implementation and associated results. The clustering approach is formulated to find the optimal number of

clusters within the IoT system over 5G networks. The proposed clustering algorithm is based on the Genetic Algorithm (GA) and Simulated Annealing (SA). The performance analysis of the proposed technique is presented. Compared to other methods, the proposed model achieves significant improvements in terms of load, coverage, distances, and the optimal number of cluster heads (CHs). This chapter illustrates the challenges addressed by implementing the proposed system model. Finally, this chapter concludes and provides further research directions.

- **Chapter 4** proposes, implements, and studies the performance of a HLF blockchain for securing the edge IoT. In the beginning, a system model is introduced, and the details of the system design are elaborated. The chapter presents a novel method of adopting blockchain technology for edge IoT devices. The obtained results are presented, including results from real-life IoT applications. Finally, this chapter contains the conclusion and directions for further works.
- **Chapter 5** presents the experimental performance analysis of scalable distributed HLF in a large-scale IoT test-bed. This chapter provides an overview of the HLF blockchain technology and the target platforms. The study also presents the methodology for evaluating Fabric implementations, the key configuration metrics, and the experimental setup. Then, a discussion of the results and their implications are covered. Finally, the chapter illustrates simulation outcomes and proves the effectiveness of the proposed model.
- **Chapter 6** proposes and measures the performance of IoT big data provenance schemes using blockchain on the Hadoop ecosystem. The chapter introduces blockchain technology, security settings, big data systems, and the primary settings of the model. Subsequently, the proposed model is extended to protect large-scale IoT data storage. The system implementation is presented. Finally, detailed model analysis and performance evaluations are presented.
- **Chapter 7** delivers the concluding remarks with a summary of the performed research, achievements and contributions of this research work. Finally, areas for further research are identified and recommended as future research directions.

Chapter 2

The Blockchain Technology for the Internet of Things

In this chapter, the fundamentals of blockchain and the widespread implementation of the technology is discussed to provide background knowledge for the rest of the thesis. The blockchain basic concepts are discussed in Section 2.1. Section 2.2 provides an overview of the Ethereum and smart contract concept. A comprehensive discussion on the implementation of various consensus algorithms is presented in Section 2.3. To address the blockchain technology deployment within the IoT ecosystem and improve blockchain performance, the use of Hyperledger Fabric (HLF) blockchain is proposed. A comprehensive introduction to HLF is presented in Section 2.4.

2.1 What is Blockchain?

At its simplest core, blockchain is no more than a distributed database. Think of it as a sizeable worldwide computer where everyone can securely access data and execute transactional code. All transactions are stored in blocks of data. These blocks are made to make them very hard to manipulate or fake once stored on the blockchain. Due to the nature of blockchain, it can be seen as a trustworthy way to store data in scenarios where there is no trust. This could be

monetary transactions between anonymous strangers on the internet or the ability to securely store medical information in a way that can only be accessed by those who are allowed. It is also worth mentioning that blockchain is generally not a place to store large amounts of data for each transaction. For example, generally, images will not be stored on the blockchain, but it might store information to validate if an image is being tampered with or not. Most data stored on the blockchain is focused on transactions and states of objects rather than the actual objects themselves.

The notion of a cryptographically secured chain of blocks was described by Stuart Haber and Scott Stornetta in 1991 [24]. Nick Szabo [25] did the first recognized work on the decentralized digital currency using similar technology in 1998. But it took almost ten years until the blockchain concept was getting mature and published by Satoshi Nakamoto in 2008 [12]. Satoshi is by most considered to be the founder of bitcoin in 2009. There is massive investment being done by traditional legacy software companies like IBM and Microsoft, but there is also a large and rapidly growing mass of startup companies getting their blockchain implementations out to the market.

Blockchain is a distributed network of participants that is formed in a peer-to-peer (P2P) manner. A group of nodes (data centers, devices, computers, etc.) shares information or files in a P2P fashion placed on top of the network layer. Blockchain technology has attracted considerable attention from industries and academia in different fields (including economy, finance, law, and computer science). Nodes keep a copy of the entire blockchain ledger and run the consensus algorithm on the blockchain state. Therefore, blockchain is a transparent and secure decentralized transactional ledger. Blockchain is a decentralized network technology that provides the participant's anonymity, robust security, and transaction immutability [26]. Figure 2.1 shows the chained blocks of the blockchain ledger. The blockchain is natively object-oriented, where code and data reside together. However, objects are securely separated from each other. There is no one in control over a blockchain. It cannot be stopped, and it cannot have a central failure. There is no power cord to pull nor a single point of failure. Blockchain, by its nature, is accessible and verifiable. Everyone that has access to the blockchain can verify every single transaction from the beginning of time and enables everyone to audit everything.

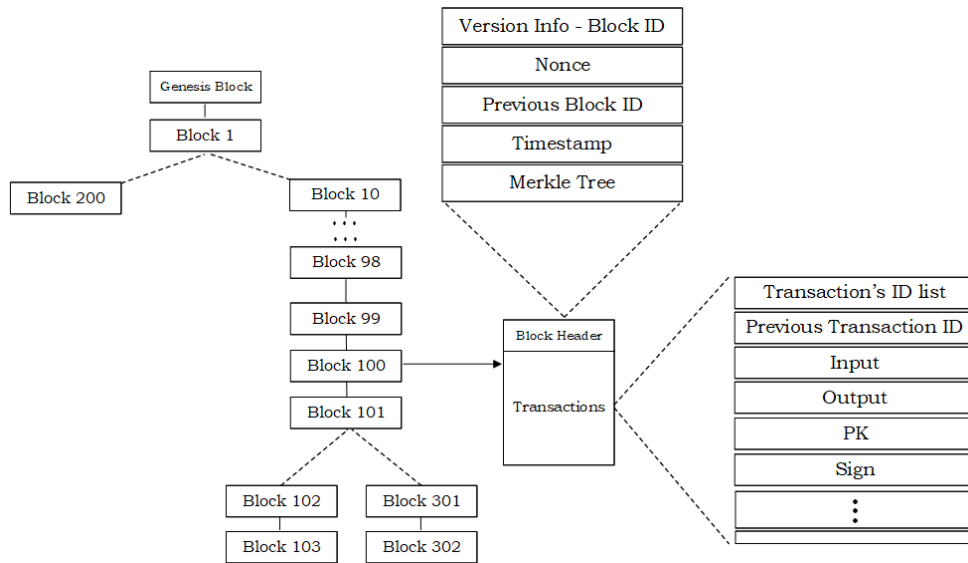


Figure 2.1: Simplified Block Structure.

2.1.1 How Does It Work?

It begins with someone doing a single or a group of transactions. A transaction is typically sending data in the form of a contract. Depending on the blockchain implementation, it can also involve cryptocurrency being sent from one account to another. The transactions are sent to an extensive P2P network of computers. These are generally distributed all over the ecosystem. Each computer is called a node, and they all have a copy of the existing data. Then the transaction is executed and validated based on preshared contracts and scripts. This ensures that all nodes execute using the same set of rules. When the transaction has been executed, the result is embedded in the blockchain. Since this is done at each Node, an attacker needs to compromise every Node in the chain in order to compromise the transaction. All transactions are atomic, where the entire operation run or nothing at all. Transactions run independently of each other. So no two operations can interact or interfere with each other, and it has to be inspectable. This gives a unique possibility for securing and auditing solutions on a wide scale. Blockchain objects are immortal means that all data from an object are permanent. The code for an object can never be changed, and you can never delete an object externally. The only way to remove an object from the blockchain is that if it is programmed to remove itself.

2.1.2 Hashing

The core of any blockchain resides in the concept of hashing. Hashing is basically to execute a mathematical algorithm that creates a result with a given length regardless of the input given. The result of a hashing function is called a hash, and it can be seen as digital fingerprints. Hashing is a one-way function, meaning the process will always return the same result given the same input but never regenerate the input based on the result of the hashing algorithm. The more advanced a user wants algorithms to be, the more power it takes to execute. A widespread hashing algorithm used with blockchain is the SHA 256. It is one of several checksum hashing algorithms, and it will produce a long text string as a result. The American National Security Agency designs it, and it is made available to the public. The SHA is a family of hashing algorithms. The number following the names lets one know the complexity of the implementation. A very common use for hashing is when storing passwords in a database. The longer a password is, the less likely it is that the hacker can find the input that created a particular hash. Because hash algorithms always give the same result given the same input and always give a fixed length of the result, they are also ideal for verifying the consistency of larger amounts of data. Even a single comma would result in a different hash result. So when comparing two hashes, it is very quick to determine if they match or not. Hashing algorithms are also used in numerous areas of modern security where the need for consistency is high, for example, insecure communication where timestamps are exchanged.

A Merkle tree is a hash of hashes, making it quick and relatively easy to confirm large amounts of data and transactions. In the Merkle tree, groups of hashes are hashed together to create a hash of hashes. These hashes are then hashed together as well, creating what we call a root hash. In order to figure out if something has changed anywhere in the Merkle tree, we only need to see if the root hash has changed, and one can then follow the tree down to see where the change is done.

2.1.3 The Block and Mining

A block consists of data and its resulting hash. If any change occurred in data within the block, the hash would change, and the block will be invalid. It also includes the nonce, which is input to the hashing algorithm that would result in the first part of the hash to be something predefined

like a set of zeros. It is impossible to predict the nonce to be considered in proof of work by the machine creating the hash. Consider that we require the hash to have a leading number of four zeros. Whenever a change happens, there is a need to rerun the hashing algorithm until it figures out which nonce to set. This is called mining the block. A structure of a block is presented in Figure 2.2.

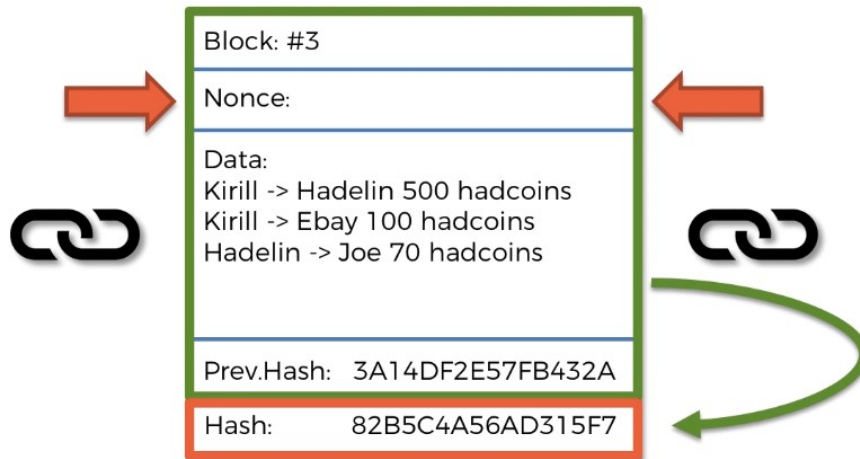


Figure 2.2: A block structure in Blockchain.

The block also contains a block number, meaning which order it has in the blockchain. A block also must contain a timestamp, but most importantly, the block in a blockchain will include the hash of the previous block. If any of the values in one of the blocks is altered, all the blocks following in the chain will be broken. This is also where the Merkle tree (Hash Tree) is used. The only way to fix the blockchain will then be to mine all the blocks after the changed block and make the nonced hashes all over. In the blockchain, a chain of blocks is distributed to a vast number of computers. The chain exists in multiple locations. Depending on the implementation of blockchain, it could be millions of replications of the chain. It could easily be determined if something has changed, even if one chain has been re-mined. The resulting hashes would be different from one chain, and blockchain works in a way where the chain with the most work put into it wins. The altered chain would then be rejected by the distributed blockchain and be removed.

Each block contains the main data (Transactions), previous block hash, current block hash, timestamp, and other information (Block header). Figure 2.3 illustrates a simplified block

structure in the blockchain and block linkage in the network. The transaction structure, block header contents, active and stable blocks are shown in this figure. Marked blocks in black represent functional blocks, and blue ones are called stale blocks. The first block in the blockchain is called the genesis block. Multiple children are connected to a parent block simultaneously. The main data is application dependent which will be described based on the application for which the blockchain is implemented, such as IoT data transaction.

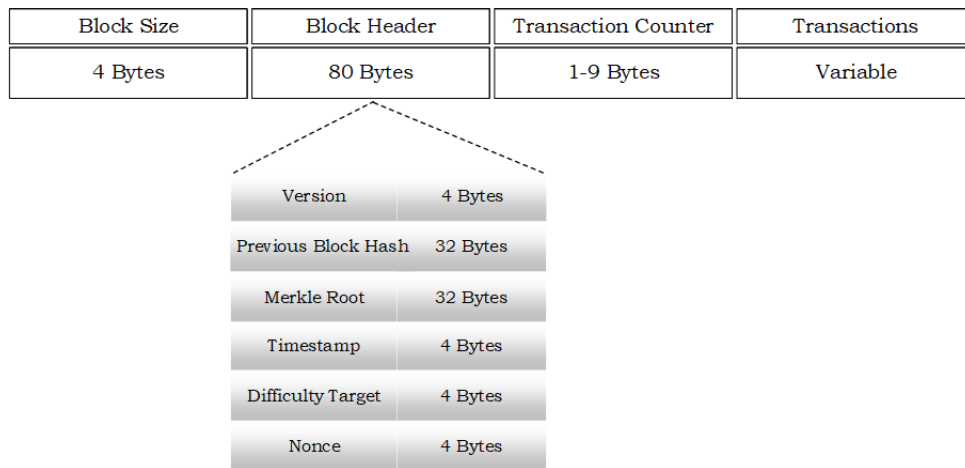


Figure 2.3: Structure of a Block and Block Header.

The block header structure is highly dependent on the blockchain type. The figure 2.3 shows a high-level representation of a block and the associated block header. Each block header includes a unique identifier of the block generated from the hash of all content of the block called the BID. Previous BID refers to the previous block used to link blocks (chain of blocks) and provides immutability within the blockchain ecosystem. If the stored transaction has been compromised, the hash of the associated block, which is linked and recorded to the next block, is no longer consistent; thus, the malicious activity or attack will be exposed. As mentioned before, the miner constructs a Merkle tree based on transaction history and block information. As depicted in Figure ref 2.4, the Merkle tree recursively hashes the established transactions of the block stored as the various leaves of a tree. The block header keeps the Merkle tree root to maintain the transaction membership verification in a block with a better speed and better performance.

The timestamp, the last field in the block header refers to the time when the block was generated. Miner appends (mines) the block into the blockchain ledger by implementing a consensus algorithm such as Proof of Work (PoW) [12] and Proof of Stake (PoS) [13]. The consensus algorithm performs the blockchain consistency among blockchain participants and ensures the randomness

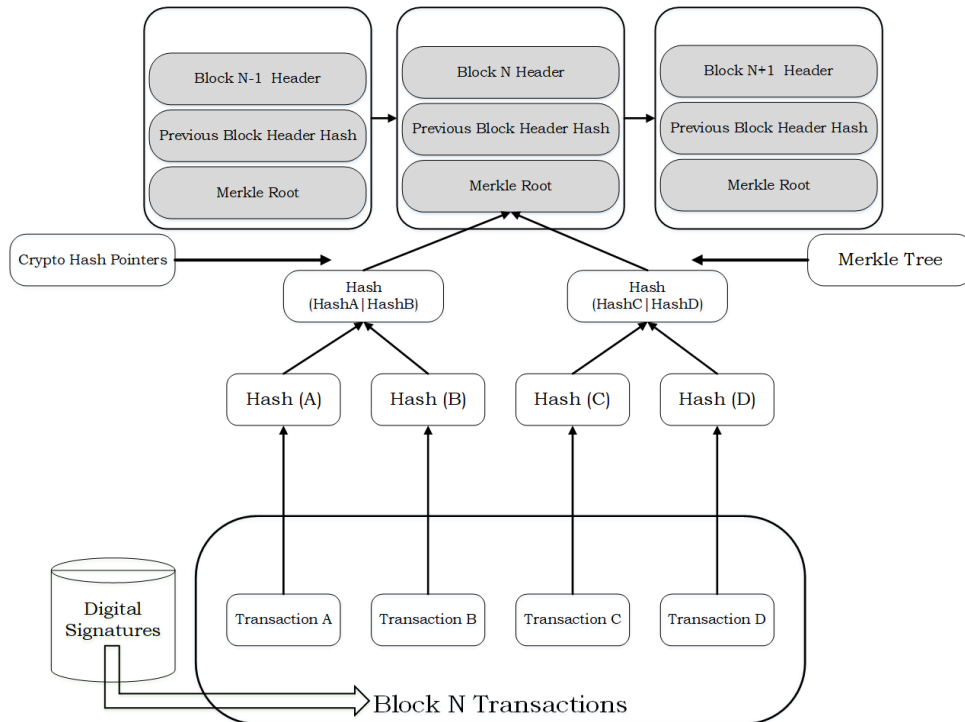


Figure 2.4: Merkle Tree Structure.

between various miners. The randomness is necessary to stop malicious miners from mining blocks continuously and enhance blockchain ecosystems' security. Section 2.3 further explores the existing consensus algorithms in various blockchain platforms. The blockchain broadcasts the mined block to all network participants and nodes. Each node adds the newly generated block into its local replica of the blockchain ledger after validating the integrated transactions. Following this process, the newly mined block will be removed from the pending transactions pool by the miners.

2.1.4 Securing Data

Data stored in the blockchain is generally available to everyone who has access to the chain. This gives some challenges, but it will also lead to thinking about security first. It is not a big challenge in some applications that everyone has access to everything, but it is necessary to assure some degrees of control on data in most cases. There are two ways to handle this. The first is obfuscation. To make the data relevant only to those who know its value. One example of this is Bitcoins. With the address of any account, it is a long string of letters and numbers. No one in the chain knows who the address is physically connected to, and they don't need

to know, but everyone knows every transaction going between the different account addresses. It is optional to be anonymous or not. Another example of obfuscation is to have the data contain IDs and status codes. The problem with relying only on obfuscation is that it always risks someone making the connections due to a breach in the system controlling the keys and connecting the data. To deal with this, blockchain use encryption. When a message or the transaction payload is encrypted, it applies a two-way algorithm to the message in a way that can only be decoded if you know a password or a passphrase. The data can be available to everyone, but only those who have the keys to decrypt the message can make sense of it. There are many different encryption algorithms, and as with hashing, they are continuously evolving. One of the most secure in use is now called Advanced Encryption Standard or AES for short. It uses long keys to produce heavy-duty encryption. The harder it is to crack. The original message passes through an AES 128 encryption with a generated key. The structure of the block is kept the same, but the message itself is encrypted. The block can still be evaluated and verified by the blockchain participants, but the content will only be available to those with the key.

A transaction will be hashed before execution and will be broadcasted to each participating Node. A transaction in the blockchain maintains the basic communication primitive that forms the information exchanges between two nodes. In a most general sense, transactions are an encrypted data structure stored in files known as blocks in the blockchain. Transaction ID (TID) represents the hash of all other transaction fields to create the transaction's unique identifier. Genesis transaction denotes the first transaction in the ledger. Previous transaction ID (PTID) indicates the previous transaction that facilitates the link between successive transactions made by the same entity (or node). A list of timestamped blocks builds a blockchain ledger that records all transaction history in the network in an ordered manner. There may exist dependencies between different transactions whereby specific fields generated in one transaction (outputs) represent the inputs in another transaction. Each transaction consists of separate fields for Input and Output. The blockchain system guarantees anonymity by implementing a Public Key (PK+). A blockchain node changes the PK+ used as the transaction identifier to enhance anonymity. The PK field in a block includes the hash of this PK+ that reduces the size of transactions and protects the transactions against malicious attacks on the Private Key (PK-). A public key and generated hash are used to sign the previous transaction digitally by the transaction owner, presented as a sign filed. Following this process, the previous transaction

owner uses their private Key to sign the hash.

Blockchain broadcasts all transactions to the network participants. The blockchain guarantees security through the mining process to verify, create, publish, and broadcast blockchain blocks. Transaction verification and validation are the primary objects of the mining process. Miners are particular nodes that perform transaction verification by validating the embedded signature using the linked private key. An incentive-driven model is being used in several blockchain platforms to generate new coins through the mining process. The miners are the nodes who can solve the next block in the blockchain receive these new coins as the reward, and the blockchain system checks for the previous transaction ID's existence. The final step consists of checking for particular blockchain framework rules and verifying other fields in the transactions.

The verified transactions are placed into a pool of pending transactions to be added to the blockchain. Following this process, the Merkle tree [27] will be constructed based on block information and transaction history downloaded by miners to make Merkle root. As shown in Figure 2.3, Each miner collects and combines pending transactions into a block based on the predefined size known as the Block size. The maximum block size limits the block added to the blockchain, but a miner can select desired transaction numbers. In Bitcoin, this maximum size is 1MB to accommodate transactions within this space.

It may be recalled that the blockchain is a distributed technology in which multiple miners can simultaneously broadcast the same block, creating a fork. As shown in Figure 2.5, the fork occurs where two blocks have the same previous block ID reference. Most blockchain platforms address this issue by implementing "the longest ledger" concept where the ledger consists of more chained blocks is elected as the valid ledger. The transactions in the forked blocks are considered invalid transactions. The confirmation time is the indicator that ensures that the end-user transactions are stored in the longest ledger by waiting for a number of blocks to be embedded into the chain of the block that contains their transactions. This confirmation time varies for different blockchain platforms and applications; for example, in Bitcoin, it takes 30 minutes to store three blocks in the blockchain [12].

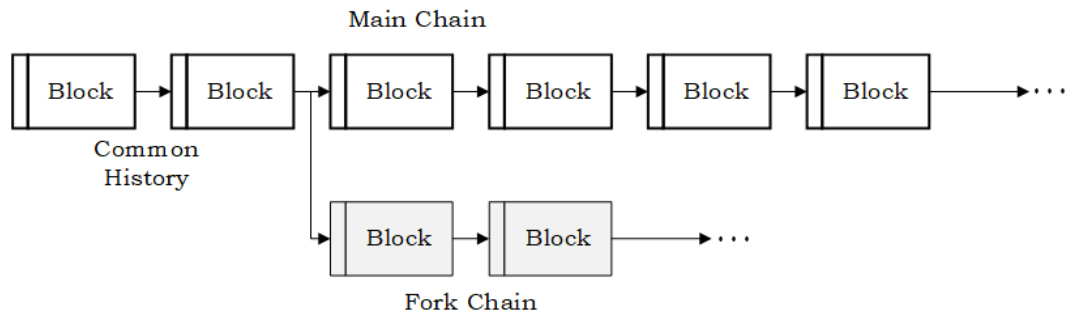


Figure 2.5: Simple blockchain forking.

2.1.5 Blockchain Implementation for the Internet of Things

In order to integrate blockchain technology into the IoT system, the most appropriate blockchain framework needs to be designed and deployed. The selected blockchain framework needs to implement the consensus algorithm, which best suits the resource-constrained IoT environment. This section presents some of the existing blockchain platforms and their practical applicability to be deployed within IoT applications. Various aspects of these implementations are reviewed, including associated consensus approaches, applications, operation mode (permissioned or permissionless), accessibility (private or public), and other related information. The characteristics of the blockchain system are affected by the features above and its performance, scalability, and availability. The scalability indicates the number of supported participants by the blockchain network and the size of the network. The performance refers to throughput and network latency which are the critical parameters of an IoT system. The accessibility of the network participants to a replica of the distributed ledger is defined as availability [28].

Blockchains can be either public or private. In a public blockchain, everyone with an internet connection can connect to the chain. The way the public chains are funded is that one generally needs to pay for storage, transaction, and execution costs to the entities that have joined the chain. Since the public blockchains are generally widely distributed, there is also no given point for attack for a hacker. They cannot target a single data center or company to bring down the chain. But this really depends on how well the community supports the chain. One of the most significant benefits and challenges of a public blockchain in a business context is that it is democratic. The community has to decide how forks are being handled. Many consider these not to be real blockchains when looking at private blockchain as they are not truly distributed and democratic. A private blockchain is closer to a traditional database.

2.2 Ethereum and DAPPS

Ethereum is one of the largest and most well-established blockchains, and it is based on the proposal from Vitalic Buterin in 2013 [13]. It was publicly available in 2015. Since it is open-source, it is straightforward to use it for private blockchains. Ethereum is a platform and a programming language running on a blockchain to build and publish distributed applications called DAPPS. Ethereum has its cryptocurrency called Ether that in many ways is similar to bitcoins. One benefit with Ethereum is that it has fast transaction times where the block time is set to a few seconds compared to minutes for Bitcoin. This can make it more suitable for applications where a fast response is needed. Another big thing about Ethereum is that it has a method for funding transactions depending on their computational complexity, bandwidth use, and storage needs. This is a difference from the bitcoin blockchain, where transactions compete equally with each other. The proof of work that other blockchains are using has had some negative impact where large computing compasses are solving complex problems just to prove that they put work into it. This is a big waste of resources, and Ethereum is moving more towards proof of stake, where they use direct economic stake instead of proof of work.

2.2.1 Distributed Applications (DAPPS)

A traditional application architecture could have a service setup that would provide the application in storage, data logic, and user credentials. A client would run an interface of some kind, and these would have a set communication channel. With distributed applications on the blockchain, this can be constructed a bit differently. Shared data is available on the distributed blockchain and exists in many locations, so it doesn't matter if one of the servers goes down. The same applies to the data logic, where the logic is shared on the blockchain. As soon as a contract is uploaded to the chain, it will spread and execute regardless of where it runs from. The client in a distributed setup also looks a bit different since it will be responsible for storing its own user credentials and typically store more of the application data. Building applications this way will make the solution very resistant to system failure, and it means that they can fully operate without a centralized system.

When deploying code to Ethereum, it will create smart contracts built in a language called solidity. Smart contracts are pieces of code that live on the blockchain and execute commands

exactly how they were told to by a shared logic. The contracts make up what will be the data logic for our distributed applications. They can read other contracts, make decisions, send ether, and execute other contracts. Contracts will exist and run as long as the network. They will only stop running if they run out of transactional funding or if they were programmed to self-destruct.

2.2.2 Transactions

The term transaction, when used in Ethereum, needs to refer to the data package that stores a message to be sent from an externally owned account to another account on the blockchain. A transaction contains the receipt of the message, a signature identifying the sender, a value field which is the amount of WEI to transfer from the sender to the recipient. A gas price value representing the fee the sender is willing to pay for gas. The higher price one is willing to pay, the higher miners will rank one's work. There is also a start gas value representing the maximum number of computational steps the transaction executed is allowed to take. This would be the maximum cost, and it will make sure that one is not able to start infinite loops in one's code. It is also important to the miner to get an estimate over how much they can earn by doing the transaction.

2.2.3 Smart Contracts

Making a smart contract is started by writing the code in a supported language. That will be different depending on the blockchain implementation. When a compile is done successfully, it will be uploaded and wait for it to be mined. After the contract is successfully mined, a client can start interacting with it. One would create a user interface towards the contract in most cases, but one can actually interact directly with it through http post operations.

As with most other programming languages, there are access modifiers. Access modifiers are keywords used to ensure that one's code only can be executed from where one expects. Public means that it can be accessed from everywhere. Private limits access only to this contract. Internal means that this contract and those contracts deriving from it can access the method or property. External will disallow internal access and make it only accessible externally. Making

something private only prevents the other contracts from accessing and modifying the information, but it still will be visible to the whole world outside of the blockchain.

2.3 Blockchain Consensus Algorithms

This section summarizes the existing consensus algorithms implemented in available blockchain platforms.

The most broadly used consensus algorithm in various blockchain networks is Proof of Work [12]. PoW is a mathematical puzzle that needs to be solved by a miner for new block validation while adding a new block. The miners devote effort to find a nonce value, so the block contains hash and the nonce contains a particular number of leading zeros. PoW applies a difficulty value to define the number of leading zeros. Therefore, the number of the generated block is limited to one block in the network per 10 minutes. PoW is a resource-consuming protocol as it involves solving a hard-to-solve cryptographic puzzle.

Proof of Stake is the primary consensus algorithm used in Ethereum [13]. PoS is proposed to address computational resources and energy usage challenges in blockchain platforms. In PoS, the miner node selects a certain amount of assets to mine blocks, and the power of mining relies on the value of chosen assets. PoS depends on the nodes that invested more assets on it and less probability of attacking the blockchain. PoS deploys CASPER protocol to perform the consensus process.

Proof of Capacity (PoC) [29], also called Proof of Space (PoSpace), implements the idle space of the disk of the local computer to maintain the mining process. PoC is similar to the PoW consensus algorithm; however, it utilizes disk space instead of applying computation resources. All nodes in PoC store the solutions for series of computation puzzles that are hard to find but easy to verify. The node with the fastest solution for the puzzle in the latest block will mine the new block and get the mining rewards.

Proof of Authority (PoA) is a consensus algorithm that depends on a chosen set of trusted nodes (Authorities). PoA is similar to PoS; however, miners' mining power is predefined based on their rule in the network instead of the number of locked assets [30]. All network participants identify the pre-approved miners and their associated identities. The chain becomes a part of

the permanent records when the majority of authorize nodes, at least $N/2 + 1$, sign off the chain. The miner with a higher reputation has a higher chance of mining new blocks.

Intel released a new consensus algorithm for blockchain platforms known as Proof of Elapsed Time (PoET) [31]. PoET uses Trusted Execution Environment (TEE) in Intel CPUs for running the leader-election-based consensus algorithm. PoET is integrated with trusted blockchain environments such as Hyperledger. The algorithm needs a waiting time selected randomly from a trusted network before appending a block to the blockchain. The TimeChecker function does the random time selection verification.

RepuCoin [32] is a consensus algorithm based on the reputation of the validators. The mining power of miners increases with the node reputation and likely has more chance of appending a new block to the blockchain. The algorithm initiates by identifying a group of miners with a high reputation. All participants know this group of nodes to measure the importance and their reputation. A randomly selected leader is responsible for the mining of the next block. The leader selection process is undertaken through voting by members of the group based on the weighted practice using the node reputation. The packet overhead increases during the voting process in a large distributed environment such as IoT with many validators.

The authors in [33] proposed a Byzantine agreement-based algorithm where the miners can achieve consensus in one round called AlgoRand. The mining process does not involve rewards for miners, i.e., validators. Randomly selected validators validate the next block. The newly generated block is broadcasted to the network and each validator votes to one block. The consensus can be achieved when all validators vote to the block, and the block with the highest number of votes will be chosen as the next block. Such algorithms' bottleneck in large IoT networks appears with an increasingly significant bandwidth overhead from the participants.

In Practical Byzantine Fault Tolerance (PBFT), all the nodes contribute to the voting process to add the next block, and the agreement is achieved once more than two-thirds of all nodes agree upon that block. PBFT needs a minimum of $3f + 1$ replicas to perform normally, where f indicates the maximum number of faulty replicas. This minimum number ensures enough non-faulty replicas to find the faulty ones byzantine and crashes. Byzantine Fault Tolerance (BFT) protocols are a powerful mechanism to attain highly reliable and available systems. PBFT can reach the consensus quicker and more with fewer resources compared to PoW. Also, it does not require owning assets similar to PoS to participate in the consensus process [29].

Federated Byzantine Agreement (FBA) [34] is a distributed version of BFT where any node can be a validator, and the validator forms a group of validators randomly to build a quorum. Nodes reach agreement once the quorums have intersections and the selected leader mines the next block. However, the packet overhead is a bottleneck due to the cost of checking for intersections and forming a quorum.

Raft is based on a voting process proposed to enhance the Paxos algorithm's deployment in practical systems and make it more understandable. Raft and Paxos are categorized as non-Byzantine fault tolerance algorithms, and they can tolerate crash faults up to 50% of the nodes. The consensus includes two stages: the election of the leader and log replication. The elected leader maintains the transaction ordering. Randomized timeout is utilized for the leader selection stage. The next step starts with elected leaders accepting the log entries from clients and broadcasting transactions and making the transaction log version [52,69]. This consensus algorithm has high throughput and low latency. However, its performance relies on the leader node, which is dominant in the system [35,36].

Some of these consensus mechanisms, such as PoW and PoC, are limited by the system's total computing power amount. Therefore, the systems without sufficient computation power and enough storage are vulnerable to 51% attacks. Limited throughput is another main limitation of the consensus algorithms mentioned above. The existing approaches limit the blockchain platform throughput because the number of blocks appended to the chain is limited. The key features of the existing consensus algorithms in current blockchain technologies are summarized in table 2.1.

Table 2.1: Existing consensus algorithm in current blockchain technologies.

Consensus Algorithm	Access-Type	Mining Approach	Decentraliz	Computing	Packet Overhead	Scalability	Energy Efficiency	Latency	Throughput	Limitations in IoT Applications
PoS	Public / Permissioned or Permissionless	Stack of locked assets and coins / Hash power	High	Medium	Low	Medium	Partial	Medium	Low	<ul style="list-style-type: none"> - Make the blockchain centralized - Pre-purchase of assets are required - Not suitable for IoT - Based on the monetary concept which is not applicable in IoT
PoC	Public / Permissionless	Idle disk space	High	Low	Low	High	Yes	High	Low	<ul style="list-style-type: none"> - Not rational choice for resource-constrained IoT - Waste disk space - High latency
PoA	Public / Permissionless	Solving difficult hash / Reputation	High	High	Low	High	Partial	Medium	Low	<ul style="list-style-type: none"> - Not Suitable - All network participants know about the miners - Pre-approved nodes only can perform mining - High delay
PoET	Private / Permissioned or Permissionless	Time-based based on Intel CPU power	Medium	Low	Low	High	Yes	Low	High	<ul style="list-style-type: none"> - Particular hardware requires for miners (Intel dependent) - IoT favorable (Low latency, high throughput) - Resource-efficient
AlgoRand	Public / Permissionless	Leader selection based on voting process	High	Low	High	Low	Partial	Medium	Low	<ul style="list-style-type: none"> - Low scalability due to voting - High latency - There is no monetary concept in IoT to assign weights
RepuCoin	Public / Permissionless	The reputation-based consensus / PoW-based	High	Medium	High	Low	Partial	High	Low	<ul style="list-style-type: none"> - Miners are known to the network - Low scalability due to voting
FBA	Public / Permissioned or Permissionless	Leader election based on quorum intersections	High	Low	High	Low	Partial	Medium	Low	<ul style="list-style-type: none"> - Scalability concerns due to finding intersection and structure the quorum - Need reducing the network overhead
Raft	Private / Permissioned	Randomized timers	High	Low	Low	Medium	Yes	Low	High	Need modification on restricted throughput
PBFT	Private / Permissioned	Based on Voting process	High	Low	Medium	Medium	Yes	Low	High	<ul style="list-style-type: none"> - Well suited for IoT applications - Low computational power and complexity - The network overhead and scalability concerns - significant energy consumption reduction

2.4 Hyperledger Fabric

Hyperledger Fabric (HLF) is a permissioned blockchain widely implemented by various enterprises [37]. HLF is built upon a pluggable consensus mechanism to address the specific application requirements. Practical byzantine fault tolerance is the most common consensus in HLF that can achieve consensus in hundreds of milliseconds. The permissioned blockchain environment provides the organization with a high degree of control, authorizes particular nodes to join the blockchain, participates in the consensus, and has access to the shared ledger. HLF supports a specific version of Smart Contracts called ChainCode [38]. A smart contract is defined as a self-executing piece of code to automate predefined tasks, including transferring products, services, or financial assets between various parties without an intermediary [39]. The smart contract concept can empower many IoT applications by designing and developing security mechanisms for IoT and defining different measures to protect IoT privacy based on smart contracts. These features are more beneficial in autonomous and automated IoT platforms with many devices distributed in a decentralized structure. Transactions are usually independent and carried out by different smart devices in IoT networks that can be managed and processed by different smart contracts more efficiently and securely [40].

HLF components are peers, orderer, chaincode, and policies. Peer nodes validate and perform various requests coming from network participants within the network as well as new block validation. The orderer is responsible for handling transactions (valid invoked requests), structuring a block based on received transactions, sending blocks for the verification process by peers, and appending them to the shared ledger's replica. Table 2.2 summarizes the well-known blockchain implementations and their key features.

2.4.1 Distributed Ledger Technology for Business Applications

This section presents the characteristics that are desired in a distributed ledger technology for creating business applications. Hyperledger fabric is suitable for creating distributed ledger technology-based business applications. Hyperledger Fabric is a distributed ledger technology for the business. The keyword here is the business that differentiated the Hyperledger Fabric from other blockchain technologies that are geared more toward the public domain. Two such dominant public domain distributed technology networks are the Ethereum and the Bitcoin

Table 2.2: Comparison of well-known blockchain implementation and their key features

Blockchain Implementation	Bitcoin	Ethereum	Hyperledger Fabric	R3 Corda	IoTA
Governance	Satoshi Nakamoto	Ethereum developers	Linux Foundation (IBM)	R3	iota.org
Operation Mode and Accessibility	Public Permissionless	Public Permissioned or Permissionless	Private Permissioned	Private Permissioned	Public Permissionless
Consensus Mechanism	PoW	Pow and PoS Ledger Level	Pluggable consensus Transaction Level PBFT	Pluggable consensus Transaction Level RAFT	Tangle
Decentralization	Yes	Yes	Partially	Partially	Partially
Privacy	Weak	Weak	High	High	Weak
Scalability	High	High	Partially	Partially	High
Throughput	Very Low	Low	High	High	High
Latency	10 Minutes	12 Seconds	100 ms	Not Verified	10 ms
Network-intensive	No	No	Yes	No	No
Compute-intensive	Yes	Partially	No	No	No
Smart contract	Limited	Yes Solidity	Yes Go, Java	Yes Kotlin, Java	No
Currency	Bitcoin	Ether Token via smart contract	None Tokens via CC	None	Iota

networks. The business application would require the distributed ledger system to have certain characteristics which are very different from the characteristics desired in a public domain distributed Ledger technology.

There are four characteristics that make Hyperledger fabric suitable for implementing DLT based business applications. Hyperledger Fabric is a permission network. It supports confidential transactions, and to participate, one does not need cryptocurrency, and it is programmable. With these characteristics, Hyperledger Fabric establishes trust, transparency, and accountability among the participants in the network. Hyperledger Fabric allows businesses to create permission networks. The Hyperledger fabric provides ways by which owners of the network can restrict who can access and do what on the network. It requires participants in the network to be known and join the network; the participants have to get permission from some authority.

Compare this with the public DLT platform such as Ethereum, where anyone can participate on the network as it is a permission-less network. Hyperledger Fabric provides ways by which the rules are assigned to the participant, and actions that each of those roles can take are restricted by way of access control list. Transactions are validated by a known set of validators that the participants trust since all participants are known in a business setting. It is easy to identify or establish those trust authorities or transaction validators. Compare this with a public network where everyone is anonymous, and there is a lack of trust among the participants. That is why in those public networks such as Ethereum, resource-intensive validation schemes are used. Another characteristic that is desired for business applications is the confidentiality of the

transactions. Not all transactions are desired by the business to be visible to all. Hyperledger fabric puts participants in control of the visibility of their transactions. Consider the scenario where A, B, and C are engaged in carrying out business on a DLT based application. Any transaction initiated by any of these entities will be visible to all of these participants. Now, if B and C would like to engage in some kind of business activity that requires them to restrict the visibility of the transaction to only two of them, then they can create a private channel to carry out such business. Any transaction initiated by B or C on this private channel will be visible only to be B and C. Businesses using Hyperledger fabric can create and participate on multiple such private channels or networks.

Hyperledger Fabric does not have any concept of cryptocurrencies. Now, Ethereum uses cryptocurrency to incentivize the distributed ledger network. Transactions on Ethereum are validated by miners who get paid in cryptocurrencies that they can exchange for the fee. This kind of transaction validation scheme is not needed in the case of a business DLT application. In other words, there is no need to incentivize the network using cryptocurrencies. Hyperledger Fabric allows participants to decide on who the validators would be and what kind of policies will be used for transaction validation.

Typical public domain transaction validation schemes such as proof of work which is very resource-intensive, are not applicable and are not needed for distributed ledger business applications. Hyperledger fabric is programmable by way of the construct called the chain code. Conceptually ChainCode is the same as the smart contract on other distributed ledger technologies. Businesses can use ChainCode to automate the business processes ChainCode sits next to the ledger, and participants of the network can execute the chain code in the context of a transaction that gets recorded in the ledger. Automation of business processes by way of chain code leads to higher efficiency, transparency, and greater trust among the participants.

2.4.2 Assets, ChainCode and Ledger

As discussed previously in the lecture on the distributed ledgers, assets represent some kind of value that can be exchanged on the blockchain systems. Any object of value in the real world may be represented as an asset on Hyperledger fabric as long as it can be represented digitally. On Hyperledger fabric, the asset representation may be JSON or in binary format. For example,

a simplified representation of the car will have two attributes in the JSON representation, the VIN number field that uniquely identifies the car and the owner of the car. The second field is that the owner can change as a result of a sale of the car. In effect, we are saying that the state of the asset may change over time. These state changes can occur on Hyperledger fabric only by way of well-defined transactions coded in ChainCode.

ChainCode defines the structure of the asset. It also defines the transactions that can be executed against the asset. It has all of the business logic needed for the transaction. In the case of the example of a car, there can be a function to sell the car defined in the ChainCode, and a call to this function will lead to the transfer of ownership of a specific car to the new owner. All transactions are recorded in a ledger. A ledger is a data structure that keeps track of all of these transactions. It also records the state changes taking place in the assets due to the execution of these transactions. And it is already known that the ledger and hyper ledger fabric are distributed. That is, all participants have a replica, a copy of the ledger.

2.4.3 Membership Services and Permissioned Network

What that means is that there is a need to assign identities to the participants in the network. Businesses deal with known entities. Businesses have B2B partners, for example, suppliers of raw material or purchaser of goods. In some industries, by law, businesses are supposed to interact with only known entities. For example, the banking industry. Banks must know the identity of every single customer it has. Then there are regulatory agencies that interact with the businesses. A distributed ledger technology-based application for businesses in effect would require support for managing identities on the distributed ledger network.

Unlike public networks such as Ethereum, anonymous access to blockchain applications built on Hyperledger is not allowed. Business application defines the roles that are assigned to the participant, and access is granted or restricted by way of these roles. An abstract service referred to as the membership service provider takes care of generating the credentials for the various participants. In the context of Hyperledger, member refers to a legally separate or independent entity. Identity in the hyper ledger network is managed by way of X509 certificates. When a participant identity is created, the certificate is issued to the participant. Anytime a transaction is initiated by the participant, certificates, private keys are used for signing the transaction, and

any component in the network can validate the authenticity of the transaction by using the participant's public key. Interestingly on Hyperledger, it is not only the participants that are issued the certificate. Even the infrastructure components are assigned an identity by way of certificates. This is to prevent a scenario where hackers can add a server to the network, for example, to disrupt the network or to make an attempt to manipulate the transactions. Every single infrastructure component in the Hyperledger network must have a valid certificate to become part of the network.

Members are legally separate entities, and so even they are assigned an identity by way of a certificate. Certificates follow the typical process of issuance and the revocation by the certification authorities in the network. Members can manage the identities within their organization. This aspect will remove the dependency on a single centralized certification authority. And this is achieved by way of implementing the concept of membership service providers where the member can use their certificate to create new valid identities that can participate on the network. So, in other words, a member can create a new participant certificate that associates the participants with the organization or the member by virtue of the certificate chain. Also, members can create the certificate for that infrastructure component. As a result, the hyper ledger fabric network can have one or more membership service provider components. Hyperledger is a permission network. All entities participating on the network are known and have an identity that is assigned by way of X 509 certificates. Certificates are issued to all participant's infrastructure components and members. Members are legally separate entities. These are the organizations that have decided to adopt Blockchain for process automation. Each of these members is assigned a certificate, and depending on their authority, they may be able to use an MSP to create participant and infrastructure component identity within their organization.

2.4.4 Nodes and Channels

The concept of Node is common in all blockchain technologies. Think of node as a communication endpoint in a blockchain network. Nodes connect to another Node, and that is how the blockchain network is formed. Nodes use some kind of peer-to-peer protocol for keeping the distributed ledger in sync across the network. In public blockchain networks such as Ethereum and Bitcoins, all nodes are equal, and the network looks like this in the case of Ethereum. To participate in these public networks, one just needs to download the node software generally

called Wallet, create an account and Execute the Node.

Things at Hyperledger are very different. Nodes are the communication entities of the Blockchain. Nodes need valid certificates to be able to communicate with the network, and the participants use the apps that connect to the network by way of the nodes. The participant's identity is not the same as the identity of the Node. When the participant executes or invokes a transaction, the participant's certificate is used for signing that transaction. Nodes certificate is used by the network to check if they should trust the Node or not. Let's say for the sake of discussion that this Node's certificate is either revoked or has expired. In that case, the transaction signed by a valid certificate held by the participant is broadcasted to the network, but the transaction will be rejected because the certificate that Node is using has expired or has been revoked. In Hyperledger Fabric, unlike the public domain Blockchain technologies such as Ethereum and Bitcoin, all nodes are not equal. HLF components and elements are presented in Figure 2.6.

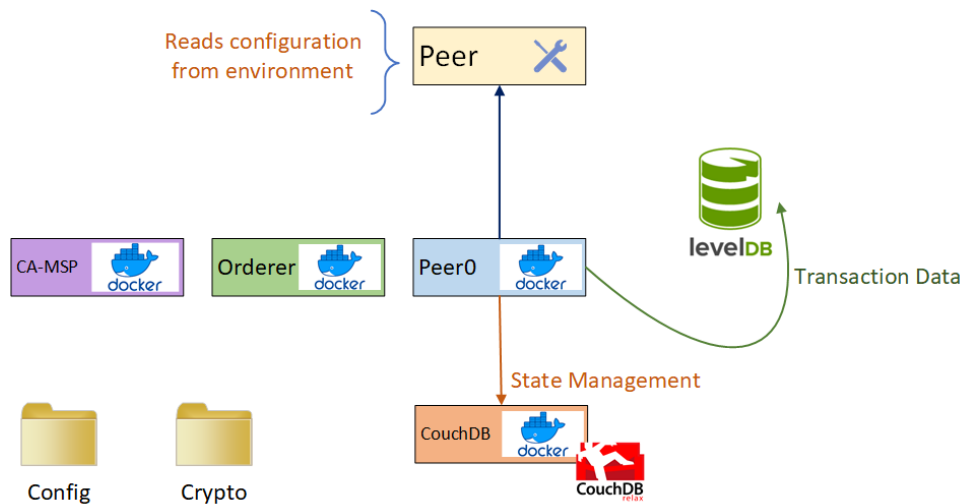


Figure 2.6: HLF components and elements.

There is three distinct types of nodes. The first one is the client node. This is the Node that applications use for initiating the transactions. Peers are the nodes that keep the ledger in-sync across the network. Orderers are the communication backbone for the blockchain network. They are responsible for the distribution of transactions. Members can participate on multiple Hyperledger blockchain networks. The transaction in each network is isolated, and this is made possible by way of what is referred to as the channel. Peers connect with the channels, and they can receive all the transactions that are getting broadcasted on that channel.

The channel has its own independent ledger. In other words, if there are two channels, there

are two different ledgers maintained in each of these channels, and there is no visibility for a peer connected to one channel into the ledger of another channel. Consider this example where there are five members, and they have decided to launch a blockchain network. Here as you can see, there is a single channel, and there is a ledger and the chain code that is available on that channel to all five members. When participants decided to have some kind of deal where they want their transactions to be private, so what they can do is they can create a private channel. The ledger and chain code for the private channel is independent and isolated from the ledger and chain code for the common channel.

2.4.4.1 Ledger Implementation

Hyperledger Fabric is a distributed ledger technology. All peers in the network have a copy replica of the ledger. Ledger has two parts transaction log and state database. In Fabric, there are two parts to the ledger. Transaction log that keeps track of all the transactions invoked against the assets. And then there is the state data. The state data is the representation of the current state of the asset at any point in time. As in the example of the car, let's say the dealer sells the car to person A, the transaction is added to the log, and state data reflects that A is the owner of the car. Now let's say person A sold the car to B. The state data reflects the new owner as B. And let's say again B sells to C; the state data reflects C as the owner.

The transaction log is immutable. But the state data is not immutable. This is sometimes a source of confusion. The term CRUD defines various actions includes create, retrieve, update and delete. There is a need to find out how CRUD applies to transaction logs and state data. One can create transaction records in the transaction log and retrieve the existing transaction records from the transaction log but cannot update an existing transaction record that is there in the log. In addition, the deletion of the transactions which have been added to the log is not possible. From the state data perspective, you can carry out any of these operations on the state data for an asset. It is mandatory to know how new transactions are created and how the CRUD operations are carried out on the state data. It all happens by way of the execution of the chain code. When the chain code is executed, it leads to the creation of transactions in the transaction log. And at the same time, depending on the code in the chain code, there may be a change in the state data. So, in other words, the CRUD operations are implemented in the code for the chain code. The transaction log is implemented using the levelDB. LevelDB is a

lightweight library for building key-value data stores. It is used in an embedded manner as part of the fabric peer implementation. That means that the levelDB is not launched as a separate process but is part of the peer process. It is a queryable and highly efficient implementation for the insertion of data or the creation of data.

Peers write the transactions to the level database. It is crucial to know that one cannot replace the level database with any other. It is a fixed implementation from the fabric peer implementation perspective. The state data consists of the key-value pairs that are versioned. The asset state is managed in storage variables identified by the keys, for example, and Key is equal to the owner. The value is represented by way of arbitrary blobs of binary objects. These blobs may be in JSON format. Ledger implementation is presented in Figure 2.7.

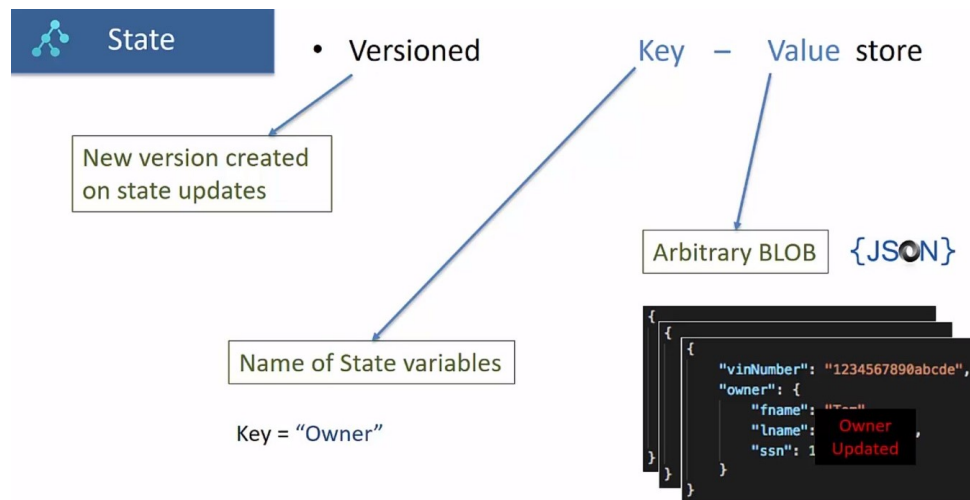


Figure 2.7: Ledger implementation.

Apart from the key value, the state data also has a version. When the state data is updated, the existing data is not over-written, which would cause the old data to be lost. Instead, a new version is created for the key-value pair and placed in the state database. For example the car has two attributes VIN number and the owner information. This is version one of the car's state data. If the car is sold by the current owner, so the owner data in the state is updated. So, at this point the version one of the data is there, and it's not overwritten. A version 2 of the state data is created for the owner and retained in the state database. Same way, version 3 will be created the next time a transaction is executed that updates the owner information. State data can have this representation - state name of the Key and then the version and value for that Key.

The start data by default is managed in LevelDB, which is the embedded database, like it has been explained in the case of the transaction log. The data will look like this for the car example. The VIN number, which never changed, stays at version 1, and then ownership of the car has changed over a period of time. So, version 1 was when the car was created by the manufacturer, and then version 2 is when the ownership of the car was transferred to the dealership. Assume the dealership Adam dealership and then when the dealership sold the car to John. Then there is a version 3 created in the levelDB. One thing to keep in mind is that chain code owns the data for these keys, so identifying a specific key is not just the key name but the chain code name and the key name. So, in other words, the access to the data is restricted to the owner chain code, and if there's another chain code that tries to access it won't be able to do that. Now, this is how it's implemented today in version 1.0.1. This may change in the future, wherein the other chain code may be allowed to access the chain code data that they don't own.

Both transaction log and state databases, by default, use levelDB, and levelDB support such simple queries. Now, if these simple queries are going against the transaction log, that should be fine because you are not going to go for complex queries against the transaction logs. It is crucial to know that when someone queries the transaction log, they will be querying it against the peer because the peer has the levelDB embedded within the peer process. Businesses depend on reports and business intelligence, and insights that they gather from the data to make their decisions. With simple queries that are available in levelDB, Businesses will be constrained because they need to write some complex queries to generate these reports and insights.

Unfortunately, levelDB will not work in those cases. The good news is that the Hyperledger fabric team understands this issue, and what they allow you to do is switch the levelDB with a more mature database with more flexibility around the querying capabilities. The state database is pluggable at the peer level. By default, it is levelDB which supports a simple query for key-value pairs, but you can replace it with couch database, which is a NoSQL database that allows one to execute complex queries, and all of this is done by the configuration of the peer.

2.4.5 Peers Nodes, Anchors, and Endorsers

Members in the blockchain network need to set up peers in their infrastructure for participating in the network. Say, for example, member A has set up three peer nodes. All of these peers

need to be configured with the appropriate cryptographic materials such as certificates and other information. Peers in the members organization receive transaction invocation requests from the clients within the organization. As transactions are created in the network, and new blocks get generated. These blocks are sent out to the peers by the ordering service, and peers receiving these blocks need to validate and update the ledger managed on the peer node. Inherently, this kind of architectural approach is highly scalable as there is no need for a centralized effort to scale the network or to scale the infrastructure.

Each member organization can look at their needs and set up the infrastructure based on their requirements. So, for example, member organization A has three peers, whereas B has decided to set up only two peers because that may be enough for their requirements. Member organizations can have multiple peers, but not all peers receive the block information from the Orderer; only the Anchor peer receives the blocks. Peers Nodes, Anchors, and Endorsers are presented in Figure 2.8.

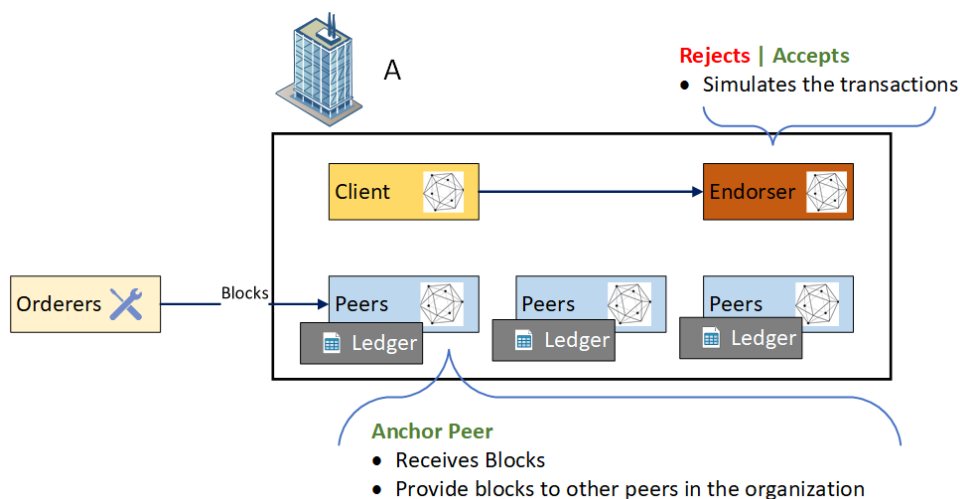


Figure 2.8: Peers Nodes, Anchors, and Endorsers.

What happens is when the anchor peer receives the block, it updates the other peers in the member organization. To avoid a single point of failure, an organization can create a cluster of anchor peers or more than one anchor peer. Anchor peers are set up and defined as part of the channel configuration, and the anchor peers are by default discoverable. Now what that means is that any peer that is marked as anchor peer is discoverable by the order and other anchor peers. Peers may be marked as the endorser, or they can take up the role of the endorser, in which case they are also known as the endorsing peer. A client sends the invocation requests to the endorsing peer. On receiving the request for the invocation, the endorsing peer validates

the transaction. For example, it checks if the end-user has used a valid certificate or not. If the validation checks out fine, then it simulates the chain code.

What that means is that the endorsing peer executes the chain code. Now, after the execution, it does not save the state of the chain code back to the blockchain as it is simply simulating the change. And you will understand much better how this process works as part of the transaction flow. So, for the time being, just think of simulation as execution, but the data that has changed in the chain code is not saved to the ledger at this point. At the end of the endorsement process, the endorser either rejects the transaction, and this may happen due to multiple reasons, for example, the security aspects didn't check out, or the execution failure or the endorser may respond back with an endorsed transaction request.

The primary objective of the endorsing peer or the endorser is to protect the network. Now when it has been explained as protect the network, it doesn't necessarily means protect from the intentional attack on the network, but it also means that it needs to protect the network from a misbehaving or misconfigured node on the network. And since every member organization is responsible for configuring their own nodes, which are peers, there is the possibility that a misconfigured node may be unintentionally added to the network.

Another advantage of this mechanism is that only the endorser needs to execute the chain code. And this will improve the overall scalability of the infrastructure. Since there is no need for all nodes to execute the chain code. The focus of this lecture was on the peer node. Every member organization needs to set up peers in the organization. Peers receive blocks from the network. There is a special kind of peer that is set up to receive the blocks, and that peer is known as the anchor. Anchor peer receives the blocks and then provides these blocks to the other peers in the organization. Then there is the endorser peer or the endorsing peer. The endorsing peer receives the transaction requests from the client. On receiving the transaction requests from the client, it simulates the transaction; that is, the transaction is executed. The chain code is executed, but the state of the chain code is not updated in the ledger. The endorser then rejects or accepts the transaction after it has carried out multiple validation checks. The primary objective of this endorsement mechanism is to protect the network from intentional as well as unintentional attacks.

2.4.6 Orderer Nodes

Endorsement policies are used by the client, the peer, and the orderer to ensure that the transactions are valid before they get added to the ledger across the network. The client is responsible for initiating the transactions, and the client does that by creating the transaction request and sending it to one or more endorsing peers. Clients connect to the endorser and send the transaction to be endorsed. It is important to know which peer should be used as an endorser, and the answer is it depends. It depends on the chain code. The chain code can associate a policy known as the endorsement policy, and this endorsement policy has two components which peers to use as endorsers. This is how the client comes to know which endorser it should connect to for getting the transaction endorsed. Another part of the endorsement policy is the criteria for the valid transactions. Figure 2.9 presents Orderer components.

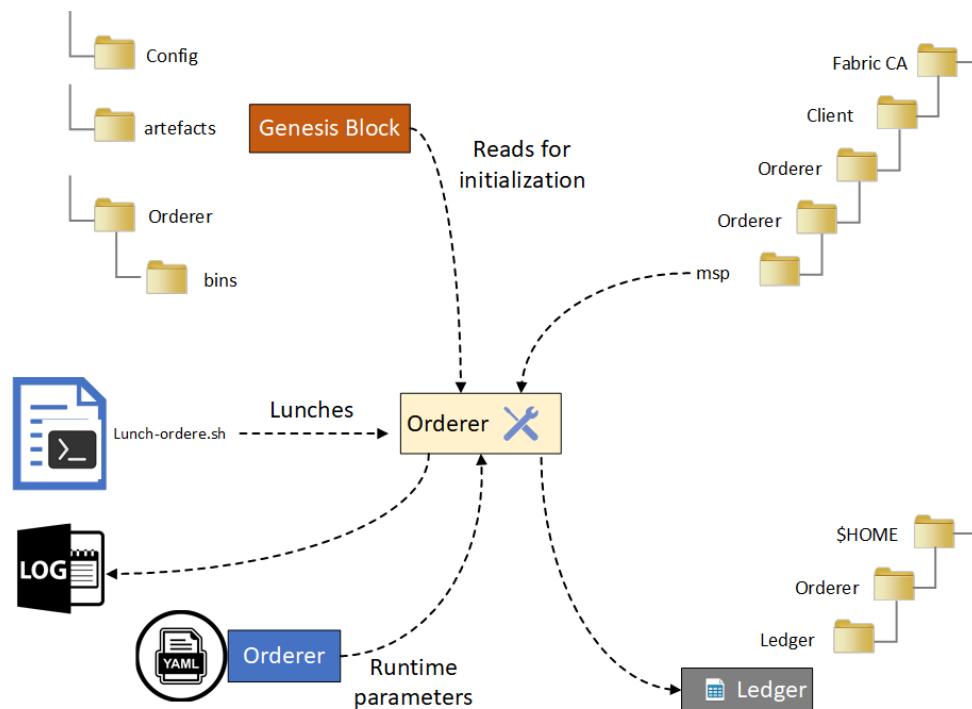


Figure 2.9: Peers Nodes, Anchors, and Endorsers.

In this example policies, three peers have been set up that have been assigned the role of endorsers. Clients must send the transaction request to all of these endorsing peers and then collect the responses. In the second part of the endorsement policy, there is a criterion for transaction validation. This is used by peers. In the criteria, the chain code developer defines either the number of endorsements needed for checking the validity of the transaction, or the

same thing can be specified as a percentage. For example, the policy may say that two out of three endorsers must endorse the policy to have the transaction considered valid. Hence, it may simply say 67 percent or above is the criteria used for marking the transaction as valid. In combination, one may use the logical expression with AND/OR or operators to define expressions that are used for checking the validity of the transaction.

The transaction will be considered valid if the organization 1 (Org1) endorser endorses the transaction, and if the Org1 member has not endorsed the transaction, then both or three endorsers should have endorsed the transaction request for it to be considered valid. Endorsement policies are optional. In other words, the chain code is not required to have an endorsement policy. The endorsement policy is specified at the time of deployment of the chain code, and the obvious question would be if the policy is not specified, then what is the default policy.

The default policy is that any peer from the client organization can endorse the transaction request. Peers are responsible for keeping the ledger in sync with the network. So, peers received the transactions that they need to add to the ledger, but before the peer adds the transaction to the ledger, it uses the endorsement policy to check the validity of that transaction. So, there are three high-level attributes that the peer checks for. The first is are all the endorsements valid. The second thing is checking the criteria. Is there enough endorsement for the transaction to be considered valid? The third thing is are the endorsements coming from the right sources. In this particular scenario, the endorsements coming from peer-1, peer-2, and peer-3. If there is a failure in any of these three checks then the transaction is considered invalid. After this check, the transaction is marked as failed if it has failed the validation and added to the transaction log. If the validation has passed, then it is marked as a successful transaction and added to the transaction log.

2.4.7 Membership Servers and Certification Authority

The membership services provider is an abstract component of the Hyperledger fabric system that provides the credentials to the clients and peers for them to participate in the Hyperledger Fabric network. The idea of abstraction is that alternate implementation of the membership service providers may be plugged in without impacting the core logic or the foundational components of the Hyperledger fabric network. The default MSP implementation is based on the public

key infrastructure (PKI). There are two main services that the MSP provides -authentication service and authorization service.

Authentication is wherein the user's identity is getting validated. In another word, is the user or peers certificate valid? That's for the MSP checks. Is the peer allowed to participate? This is again based on the certificate that the peer presents. The authorization referred to various actions that the user and peers can take. And the MSP again validates the authorization for the peers and users to carry out those actions. For example, can this user issue or create new identities, can the user deploy a chain code? So these are some examples where MSP plays a critical role in securing the Hyperledger fabric network. Now in PKI-based implementations, there is a need to manage the identity by way of certificates. Certificates are issued, validated, and revoked. This is where the certification authority fits into the picture. Hyperledger fabric development team has done great work of putting together a certification authority implementation that is available to one by default.

Certification Authority (CA) is a trusted party that affirms the identity of an entity. By signing the certificate containing the entity's public key and the key used for signing the certificate is the certification authorities own private key. There are two other authorities that are referred to in the context of certification authority the Registration Authority (RA) and the Validation Authority (VA). The process starts with the requester raising a request for issuance of identity or signing of the certificate. The requester sends the appropriate documentation to the registration authority. Once the registration authority has validated the identity, the registration authority informs the certification authority to issue a certificate. The certification authority issues the certificate by signing the certificate and then sending it back to the requester. The certification authority informs the validation authority about this new certificate so that anyone can check with the validation authority if the certificate is issued.

For a single certification authority for the whole of the network, the process of issuance of certificates will be very inefficient and may not be very cost-effective. The way Hyperledger Fabric works is that a root certificate is issued to each member in the network, and then the root certificate may be authorized to issue new identities so that the members in the network can manage the identities within their organizations. In other words, there are multiple certification authorities that can be set up in the Hyperledger Fabric network.

The node responsible for managing the certificates use the fabric-CA client to manage the

certificates on the fabric-CA server. Orderers and peers can validate the certificates using the interfaces exposed by the fabric-CA server. Apart from the CA client, there is also an SDK available for managing the certificates, and there is also a rest API interface available for the fabric-CA server.

The fabric-CA server manages the identities and certificates in the database. By default, the CA server uses the SQL light implementation, which obviously is not very robust. By way of configuration, one can change the CA server to use MySQL or Postgres, and it also supports the enterprise LDAP.

2.4.8 ChainCode Development

This section covers at a very high level the structure of the ChainCode (CC), how the chain code is developed, and the execution run-time for the ChainCode. ChainCode on Hyperledger fabric may be written in Go Lang, Java, and NodeJS. There are two parts to the ChainCode. The first part is the asset definition which is the digital representation of the asset. It is a structure or a class definition that one would create. In the case of NodeJS, the structure or the class for the asset is defined as part of Business Network modeling.

An example of the sample asset, which has two attributes asset ID and value. Once one has created the asset definition, one has to put together the transactions which will create the asset and manage the state of the assets. To do that, one would write the code for managing the state of the asset, and that code can be written in JavaScript in the case of node run time. The code in the transactions implements the business logic. This code carries out the typical operations against the various assets defined in the business network application. The create, retrieve, update and delete operations.

The developer writes the ChainCode in Java or Go Lang or NodeJS and then uses the common software practices to iteratively compile and test the code till one is satisfied that the code is working. As the next step, the developer deploys the ChainCode to the peer using a deployed transaction. As part of the deployment, the developer can also put together the endorsement policy for the chain code. The deployment transaction is propagated to the network, and once the deployment transaction is successful, the transaction log and the state data are got updated.

Participants in the network can use applications to invoke the chain code, and all such invocations are recorded in the transaction log. All the state changes are recorded in the state database. The deployment transaction deploys the ChainCode instance in its own container, and the execution or invocation of the chain code also happens within the independent containers for each of these ChainCode instances.

2.5 Hyperledger Fabric for Cellular-based IoT

This section presents various elements of the Hyperledger Fabric and its components integrated with the IoT network, including reference architecture and overall enterprise readiness. Open source, open architecture, and open standards of HLF give the flexibility to enhance the integration of Blockchain technology and IoT [38] and enables developers to tailor and integrate the system based on their needs. The specific characteristics of cellular-enabled IoT systems are obliged to comply with various technology governance standards and industry compliance. HLF, as an enterprise-ready piece of open-source software, powers the network adaption. HLF is a permissioned and private blockchain platform that is the right choice to be implemented for corporate use. It is developed in the Golang programming language, and the gRPC protocol facilitates the communication of different system elements. The Hyperledger Software Development Kits (SDKs) are available in Java, Node.js, and Golang.

2.5.1 Hyperledger Fabric Components

HLF is a modular and extensible architecture blockchain implementation, and these futures allow for alternative implementations to be plugged in and implemented with a modular framework with the IoT network system grows in complexity. The three main components of the HLF framework are the peer nodes, Membership Service Provider (MSP), and dedicated orderer service, which has been assigned to various nodes of the cellular-enabled IoT.

2.5.2 Membership Service Provider (MSP)

Different nodes within the network receive their associated certificate from the Certification Authority as a dedicated X509-based identification service. The MSP is a conceptual element

and can be organized based on the network design and specification. Moreover, any other service that provides X509-based PKI infrastructure can be implemented to issue identity certificates in different layers of the network.

2.5.3 Fabric Certification Authority

The Fabric network actors can be categorized into two main categories: human actors (such as admins, users) and machine actors (such as peers, orderers, and applications). Network components need an identity to be able to participate in the blockchain system based on the x509 certificate. The certificates holder information is appended into each issued certificate and holds some additional attributes that determine the roles and node privileges in the network. The certification authority (CA) maintains various certificates in the cellular-based IoT network and can be configured separately. In addition, any certification authority that issues x509 certificates can be implemented. The procedure is divided into a two-step process. In the first stage, the CA server creates (registrar) the node identity into the network and provides the authority holder's credentials. In the next step, the node as an identity holder enrolls the identity to obtain its x509 certificate. However, in a non-human actor scenario, the corresponding network admin enrolls and initializes the component.

2.5.4 Ordering Service

The ordering service is the backbone of the network's communication. The Orderer maintains the consistency of the ledger state across the entire network. The orderer functionality can be assigned to nodes, and the system can have more than one orderer. The Orderer establishes the Consensus in Fabric, and the Orderer service is responsible for providing the transaction's order. The Fabric provides a RAFT-based ordering service for production purposes and is considered in this study.

2.5.5 Peer Nodes

Peer nodes maintain the execution of the smart contracts (ChainCode) and the ledger maintenance. There are two particular peer nodes defined in the Fabric network: endorser peers and

anchor peers. Anchor peers are accessible from outside the defined organization; therefore, they receive the data blocks and distribute them to all peers. An organization can be structured to include a cluster of nodes (anchor peers) and address a single point of failure issues. In addition, peer nodes may act as endorsers or be assigned the role of an endorser peer. Client nodes send an invocation request for the ChainCode to the endorser peers. Upon receipt of the invocation request, the endorser peers simulate and validate the chaincode transaction.

2.5.6 Permissioned Network

The public blockchain facilitates a system in which participants download the software and begin transacting anonymously; however, it is not a suitable method to be implemented in business networks. Most enterprise networks does not need anonymity transactions within their system. Network participants are always identified by their identifiers and allocated various roles in the system. In contrast, the HLF is a permission blockchain network that assigns transactions to recognized identities and responsibilities. All nodes, users, and components need to be authenticated before participating in the network performed by MSP and CA entities. The HLF employs a public key infrastructure to approve and validate users and components.

2.5.7 Confidential Transactions

HLF provides the concept of dedicated channels. Various channels can be configured to separate the data flows. Channel capability enables transactions to be private between specified parties. Each channel can be associated with a specific ledger, and several channels may exist among consortium members connected to the same network.

2.5.8 Hyperledger Fabric Policies

The pluggable and modular features of the HLF blockchain enables participants to define several policies, different rules, decisions, and regulations that govern the consortium's operation and deployment. These characteristics facilitate decentralized decisions within the consortium blockchain environment. Numerous administrators from organization members vote by majority to make changes to the network that impact the consortium. By utilizing rules, the Hyperledger

Fabric technology enables decentralized administration. Figure 2.10 shows the considered architecture concept in this study. It has multiple parts; the main two parts in this structure are IoT layers and blockchain network. IoT layers are composed of various components: application, edge computing, network connectivity, data accumulation, data abstraction, and physical nodes. Several information can be exchanged between IoT networks and blockchain systems, and HLF blockchain deploys various communication interfaces to interact with IoT devices. This framework defines different types of peer nodes on the model, including endorser, non-endorser, orderer, and client nodes.

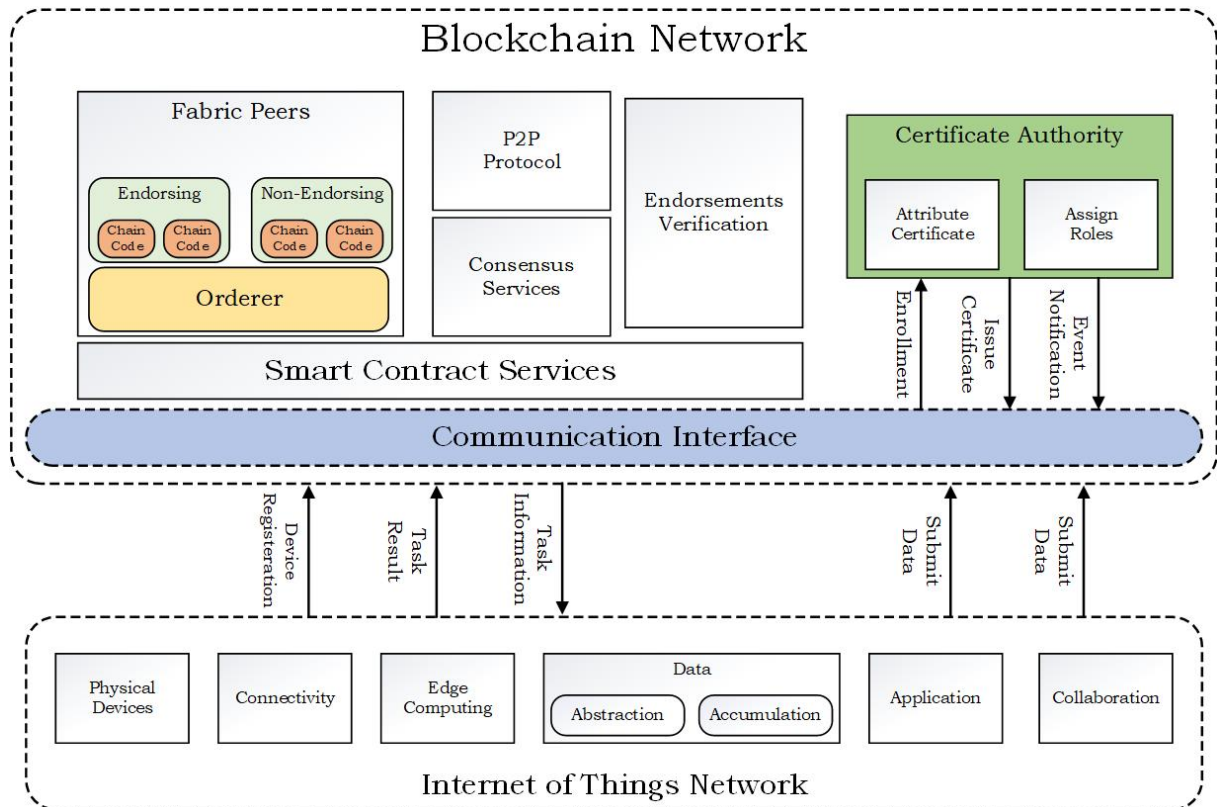


Figure 2.10: The high-level overview of the proposed architecture.

Chapter 3



Multi-Layer Blockchain-Based Security Architecture for Internet of Things

Chapter 3 includes the article, “Multi-Layer Blockchain-Based Security Architecture for Internet of Things” published in the MDPI journal of “Sensors”. This article is of open access type. It is republished in this thesis under the Creative Commons Attribution License. No special permission is required to reuse all or part of article published by MDPI, including figures and tables. For articles published under an open access Creative Common CC BY license, any part of the article may be reused without permission provided that the original article is clearly cited. Reuse of an article does not imply endorsement by the authors or MDPI.

The full text is included in the thesis has some modifications. This means that while the content is identical to the published article, there may be stylistic differences.

Article

Multi-Layer Blockchain-Based Security Architecture for Internet of Things

Houshyar Honar Pajooh ^{1,*}, Mohammad Rashid ¹, Fakhrul Alam ¹ and Serge Demidenko ²

¹ Department of Mechanical and Electrical Engineering, Massey University, Auckland 0632, New Zealand; M.A.Rashid@massey.ac.nz (M.R.); F.Alam@massey.ac.nz (F.A.)

² School of Science and Technology, Sunway University, Selangor 47500, Malaysia; SDemidenko@Sunway.edu.my

* Correspondence: h.pajooh@massey.ac.nz

Abstract: The proliferation of smart devices in the Internet of Things (IoT) networks creates significant security challenges for the communications between such devices. Blockchain is a decentralized and distributed technology that can potentially tackle the security problems within the 5G-enabled IoT networks. This paper proposes a Multi layer Blockchain Security model to protect IoT networks while simplifying the implementation. The concept of clustering is utilized in order to facilitate the multi-layer architecture. The K-unknown clusters are defined within the IoT network by applying techniques that utilize a hybrid Evolutionary Computation Algorithm while using Simulated Annealing and Genetic Algorithms. The chosen cluster heads are responsible for local authentication and authorization. Local private blockchain implementation facilitates communications between the cluster heads and relevant base stations. Such a blockchain enhances credibility assurance and security while also providing a network authentication mechanism. The open-source Hyperledger Fabric Blockchain platform is deployed for the proposed model development. Base stations adopt a global blockchain approach to communicate with each other securely. The simulation results demonstrate that the proposed clustering algorithm performs well when compared to the earlier reported approaches. The proposed lightweight blockchain model is also shown to be better suited to balance network latency and throughput as compared to a traditional global blockchain.

Keywords: internet of things; blockchain; hyperledger fabric; evolutionary clustering; security; scalability; authorization



Citation: Honar Pajooh, H.; Rashid, M.; Alam, F.; Demidenko, S.

Multi-Layer Blockchain-Based Security Architecture for Internet of Things. *Sensors* **2021**, *21*, 772.

[https://](https://doi.org/10.3390/s21030772)

doi.org/10.3390/s21030772

Received: 31 October 2020

Accepted: 20 January 2021

Published: 24 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Ubiquitous interconnected objects can be deployed through the Internet of Things (IoT) infrastructure using cloud platforms in a centralized network [1]. A wide variety of interconnected devices, including smart locks [2] and vehicles [3], can also implement decentralized solutions by employing the blockchain technology in a decentralized peer-to-peer manner [4]. Both of the models are capable of dealing with the challenges of providing privacy and security for networked devices in the IoT environment. Nevertheless, the constraints of limited resources, centralized control, scalability, overhead, latency, and throughput characterize the expected heterogeneity of IoT network devices [5].

In a centralized network structure, the server controls and enhances the performance of the devices [6]. However, centralized schemes have several drawbacks. The network with a large number of smart devices normally generates a tremendous volume of data. A cloud platform service-provider requires considerable network bandwidth as well as high-performance with regards to efficiency and storage [7,8]. Furthermore, there is always a risk of the centralized network key components failure leading to a serious (or complete) breakdown of the entire system [9,10]. The data that are collected by the central cloud storage often require further manipulation by a third-party. This potentially could lead to data leaks, thus compromising the end-user's privacy [11]. The external computing

resources coordination is another challenge for proving IoT security and performance in the centralized systems [9]. Therefore, most current centralized systems fail to provide entities with a guarantee of data reliability and privacy.

Most IoT devices are only able to communicate in short-range transmissions, due to their low-power wireless transmitters and receivers. IoT networks can benefit from utilizing the Multihop Cellular Network (MCN) concept [12] that facilitates significant shortening in signal coverage. The essence of MCN leveraged by distributed, decentralized blockchain technology can ensure the required high-security and credibility for the IoT network by addressing the drawbacks of the centralization servers [13]. Besides, it enhances the degree of trust between heterogeneous devices, and that can minimize the cost of conventional data-sharing platforms [14]. The formation of a large-scale network comprising of heterogeneous nodes is not as easy as traditional blockchain implementation needs high-performance nodes. A self-protection mechanism is also required, due to the distributed structure of IoT networks with a multitude of objects and devices [15].

The increased number of connected devices (in the order of million devices per sq km), heterogeneity of devices and vendors, interoperability, a vast amount of collected data and network traffic, requirements of large bandwidth capacity, communication latency, and trust are the major challenges within the new era of the 5G-enabled IoT. [16]. The new model of security should address the unique requirements of the 5G-enabled IoT and D2D (Device-to-Device) communication devices such as scalability, low latency, energy concerns, secure communication, and reliability. Blockchain technology, including bitcoin [17,18], have been implemented for security enhancement for a long time. Their technical value has been generally recognized. At the same time, their functionality support is still limited to simple transactional data storage. Furthermore, the blockchain is a viable option for supporting ultra-reliable low latency massive Machine Type Communication (mMTC) of resource-constrained IoT devices under the 5G networks for improved security and privacy.

This paper discusses a multilayer architecture that is based on a new clustering model suited for blockchain implementation to tackle the issues associated with implementation complexity and elaborate on the mechanism for securing IoT communication. The new network model that is based on multi-layer distributed blockchain can be regarded as an organic combination of the blockchain technology and clustering techniques that effectively utilize network clustering performance and capabilities, and significantly improve the overall security and reliability of the IoT. An adapted clustering algorithm has been developed to suit the IoT systems' blockchain implementation by considering the network performance metrics in defined cost functions. The IoT network clustering aims to reduce the network load, enhance coverage, and minimize the energy (as reflected in the distance) while leveraging the essence of MCNs. The multi-layer structure facilitates the detection of compromised entities within the entire network in each layer. Each transaction in the system needs to be verified by other participants by implementing a consensus algorithm. Blockchain is continuously monitored by the entire network participants, maintaining a copy of the blockchain ledger. Therefore, compromised nodes have no means of inserting fraudulent blocks into the public ledger without immediately being noticed by others. Thus, the multi-layer blockchain removes compromised entities from being a part of the system. This makes it impossible to compromise the integrity of records in the blockchain. Another crucial point is that the new multi-layer architecture allows for upgrading for the existing central cloud server. This makes large-scale deployments possible. Besides, a lightweight authorization and authentication process running in each cluster guarantees secure access to the network resources through implementing smart contracts.

The rest of the paper is organized, as follows. In Section 2, a literature review on the blockchain implementation in the IoT environment is introduced along with essential information on the blockchain and IoT security. Section 3 details the framework architecture and multi-layer system. Section 4 provides the proposed IoT blockchain framework implementation and associated results. The proposed clustering algorithm is based on the Genetic Algorithm (GA) and Simulated Annealing (SA) [19]. Section 5 illustrates the

challenges addressed by implementing the proposed system model. Finally, Section 6 draws a conclusion and presents future research directions.

2. Related Works

Fast-growing numbers of networked devices characterize modern IoT systems. Consequently, the amount of generated data by the connected devices is also escalating. This inevitably leads to security and privacy concerns. Security (along with computing and communication issues associated with IoT devices) is mainly due to the limited memory capacity and processing power of the devices [10].

2.1. Authentication and Authorization in IoT

Devices require authentication and authorization to enter the IoT system. These measures are considered as a critical juncture of network security [20]. Interconnected devices within the IoT environment are required for establishing secure communication with the aid of relevant authentication procedures. The authentication and authorization processes of the interconnected nodes and devices are traditionally maintained by a central authority in the IoT network based on the Public Key Infrastructure (PKI) [21]. Therefore, the process increases the authority center's workload significantly and it causes considerable delay due to a large number of requests [22]. To this end, several new authentication models have been proposed. The method that was proposed in [23] for the authentication and privacy is built up upon IP-Sec and Transport Layer Security (TLS). However, such a mechanism is not suitable for resource-constrained interconnected IoT devices due to the high demand for computational resources.

Research [24] develops an access management mechanism that is based on blockchain decentralized architecture in the IoT system. The proposed approach eliminates the centralized control server and implements the Proof of Concept (PoC) as a consensus algorithm. The development of a secure access control mechanism for IoT is presented in [25] in order to address the issues related to the distribution of access rights delegation. This approach uses the blockchain Ethereum technology to validate the identity of the entity. Research [26] proposes a framework with layers, intersect, and self-organization Blockchain Structures (BCS) to verify IoT entities. Model efficiency and security performance are analyzed in terms of storage efficiency, response time, and verification. Paper [27] highlights the concerns that are related to privacy and security of data authentication in IoT. The blockchain technology has been seen as a potential fabric for eliminating the central server concept, and distributed futures helps to address IoT challenges, such as device spoofing, false authentication, and lower reliability in data sharing. The authors in [28] propose a structure for security and authentication in IoT that is based on the blockchain. This proposal addresses the single-point-failure issue.

2.2. Blockchain-Based Frameworks for IoT Security and Privacy

Researchers have been developing blockchain technology to address the privacy and security challenges in the IoT as an alternative solution. The implementation of several privacy preservation strategies in blockchain-based IoT systems is discussed in [29]. These strategies include encryption, anonymization, private contract, mixing, and differential privacy. The authors of the research [30] review the blockchain technology and applications for IoT systems as well as a way the blockchain techniques can address the security challenges within the IoT systems. The lack of a comprehensive standard architecture, cloud server availability, capacity, susceptibility to manipulation, and cost limitations are highlighted as the critical challenges with the blockchain technology implementation in IoT [7,8].

Lightweight Scalable Blockchain (LSB) is presented in [31] in order to facilitate the privacy and security of the IoT devices. An overlay network is proposed to achieve decentralization and maintain end-to-end security and privacy with the blockchain-based framework implementation run by devices with robust computation capabilities. A new

Proof of Block and Trade (PoBT) consensus algorithm is proposed in [32] in order to address the challenges associated with integrating salable IoT networks and blockchain technology. The research aim is to reduce the computation time for the validation of trades and blocks. The work is also considered a ledger distribution mechanism to reduce the memory requirements of IoT devices. The study that is presented in [5] suggests using LSB to build the blockchain-based model on the modified consensus algorithm to minimize the Proof of Work (PoW) deployment complexity. Hence, the author replaced the PoW with a distributed trusted consensus algorithm. The proposal enhances the privacy and security of IoT networks in a decentralized manner. The research in [33] proposes a blockchain-based framework to address privacy, security, fault-tolerance, and autonomous behavior issues. The framework helps to assess the possible blockchain implementation through a decision structure for IoT and edge computing.

Data operations are performed in the blockchain system through smart contract implementation, including data gathering, invoking, transfer, storage. A new context-aware mechanism is proposed in [34] for blockchain-enabled IoT systems to facilitate the on-chain data allocation. The authors define a fuzzy logic mechanism to control the data and calculate the Rating of Allocation (RoA) value that is associated with each data request. The efficiency of the proposed mechanism is investigated in the blockchain-based cloud and fog architectures implementations.

2.3. *Permissioned Blockchain in IoT*

Hyperledger Fabric (HLF) [35], which is a distributed ledger technology, paves the way to leverage a trustful environment without central authority dependency while delivering a high degree of flexibility, scalability, and confidentiality. The consensus algorithm is an open architecture in HLF. It provides a flexible environment for modifying the configuration and increase the performance. A new authorization framework for an IoT network is proposed in [36] based on the HLF framework. The work focuses on enhancing the consensus algorithm by implementing the GA optimization. The aim is to attain the best configuration with input transactions and success rates as input parameters to the GA algorithm. The IoT data management and its traditional characteristics have been considered in [37]. The research proposes a permissioned blockchain-based decentralized trust management (BlockBDM) in order to address the security and trust problems of IoT big data management.

2.4. *Layer-Based IoT Blockchain*

A platform for facilitating secure communications for smart cities is proposed in [38]. The presented solution deploys a layer-wise security structure by integrating smart devices and blockchain technology. Paper [39] has proposed a multi-layer IoT blockchain-based solution that is specifically modelled for use in the medical field. The solution addresses computation and complexity issues of the blockchain implementation by converting IoT networks into decentralized multi-layer structures. The research presented in [40] proposes a hybrid network architecture for the smart city by leveraging the strength of emerging Software Defined Networking (SDN) and blockchain technologies. In order to achieve higher efficiency, the proposed architecture is divided into two parts: the core network and edge network. This model inherits the strength of both the centralized and distributed network architectures. In [41], the authors proposed a multi-level blockchain framework to enhance privacy and data security in IoT applications. The multi-level model focuses on improving the response time and resource utilization. The authors define mobile agents to perform the hash function, implement encryption, deploy aggregation, and decryption. The mobile agents are transferred between blockchain and IoT in order to accomplish the required tasks. A two-tier hierarchical blockchain framework for IoT is proposed in [42] for enhancing and measuring the scalability of a blockchain application in a IoT car rental system.

Some of the previous works discuss the multi-layer based blockchain approach for the integration of IoT and blockchain technology. The multi-layer based blockchain network model is introduced in [43] in order to overcome the challenges of conventional centralized network architecture. The proposed model reduces the difficulty of the blockchain deployment in IoT systems by dividing the network into a multi-level decentralized network. Hybrid IoT [44] is a new hybrid blockchain platform for IoT. It is based on the implementation of PoW and Byzantine Fault Tolerance (BFT) consensus algorithms. The proposed structure includes sub-blockchains and inter-blockchains. The BFT inter-connector platform connects two PoW sub-blockchains. An integrated blockchain-IoT is proposed in [45] in order to secure the digital system for healthcare. The work addresses the scalability challenges in the IoT system.

In [46], the authors propose a double-chain (alliance and private chain) model that considers the IoT environment for the data-sharing-transaction application. In the multi-layer model, the alliance chain processes the transactions. The transaction data record in the blockchain ledger is performed by the private chain that is deployed within each organization. The real blockchain system data is stored in an IPFS cluster server built by the alliance stores. Paper [47] proposes a hierarchical resource allocation framework based on the blockchain for edge computing. The presented model implements a smart contract-based hierarchical auction mechanism for solving resource allocation challenges for the IoT devices that are located beyond the coverage of Access Points. A blockchain-based multi-layer hierarchical architecture proposed in [48] facilitates the monitoring and managing of the Internet of Underwater Things (IoUT) on cloud data. Sensor nodes are clustered and organized based on selected residual energy cluster heads. The cluster head and node tracking are performed by using the Bloom filter. The gateways communicate by deploying a standard secret key, separated from another secret key that is used by the cluster head. Subsequently, the blockchain ledger stores the routed data. A fog layer smart gateway merged into the IoUT blockchain addresses the transaction preparation challenges, data routing to miners problems, and scalability issues [49]. The proposed model deploys a lightweight consensus mechanism to add blocks in the blockchain where the IoUT data are stored.

Unfortunately, the majority of the solutions proposed in the literature do not address the problems that are associated with the implementation of the blockchain technology in IoT systems, such as device authentication, low scalability, transaction delays, high computational resources for mining, and device heterogeneity. In our previous work [50], some of the challenges that are mentioned above are highlighted along with the discussion on the adoption of the blockchain technology in the IoT context. This article expands the implementation of the Lightweight Hyperledger Blockchain (LHB) technology and smart contracts to enhance the performance of the blockchain-IoT combination.

The heterogeneous IoT network lifetime improvement is achievable by implementing a clustering model along with a multi-layer structure. The clustering concept is the key to achieving the multi-layer architecture, where the cluster heads form the multi-layer structure. Clustering techniques for wireless networks and device-to-device (D2D) communications systems have been widely reported in the literature. They offer reduced energy consumption and higher throughput [51]. A self-clustering method is proposed in this work in order to identify Cluster Head (CH) nodes. Genetic algorithms considering various clustering factors, including geospatial ones (e.g., the distance between nodes, the base-station distance to nodes) and total network energy, are proposed. A fitness function simulating network changes and node movements within the network is optimized by deploying the SA methodology.

In the multi-layer architecture, devices in each layer have different computational capabilities and energy storage capacity. Consequently, different security strategies are proposed for individual layers. Each design is based on the blockchain. Even so, the blockchain implementation is modified to suit the devices of each particular layer. The key contribution of this research is three-fold:

1. A novel, lightweight, private multi-layer model is proposed for reducing the complexity of blockchain technology implementation while improving the network scalability. The proposed model is tailored to meet the requirements of IoT devices by adopting the blockchain technology to suit different layers of the IoT system. The simulation study shows that the proposed Hyperledger Fabric-based method outperforms a traditional blockchain solution, like the Ethereum, in terms of latency and throughput.
2. Clustering is one of the key steps of implementing the multi-layer architecture. Therefore, a new network clustering method is presented. It is based on the evolutionary computation that deploys multi-objective fitness functions that are relevant to heterogeneous IoT networks. The decentralized, fast, and self-clustering method divides the IoT network into clusters while considering the node mobility. The simulation results show that the proposed clustering algorithm outperforms existing solutions.
3. A novel method of authentication and authorization of IoT nodes is implemented in order to provide security for IoT devices and protect device communications through a multi-layer structure.

3. Multi-Layer Security Framework

The aim of the proposed network model is to provide a reliable trustful security mechanism for IoT networks while using the performance and capabilities of the cellular system. The intelligent clustering and machine learning approach based on Swarm Intelligence (SI) and Evolutionary Computation (EC) algorithms [19] is deployed in order to encode the multi-layer structure. This proposal provides a framework to facilitate the lightweight authentication and authorization of IoT networked devices (objects and nodes) based on the blockchain technology.

The proposed multi-layer network model divides the entire cellular-enabled IoT network into multiple tiers. Layer-1 consists of various clusters and IoT nodes. Layer-2 includes sink nodes and controlling devices, such as cluster heads. Layer-3 contains the base stations of a cellular network. All of the CHs, as cellular devices, have cellular connectivity with the 5G BSs, and, thus, via the BSs/D2D capability, also with other CHs. The BSs have the processing power (with appropriate servers and CPUs) to implement the decentralized blockchain mechanism at Layer-3. Figure 1 shows the overall system model.

Blockchain implementation can potentially lead to additional overhead and scalability issues [5]. The multi-layer network model, as shown in Figure 2, is proposed for minimizing the overhead, reduce delays, and response time, create associated channels to collect specific data, secure communication, and address the need for the network scalability. The first layer contains devices and nodes with a diverse range of computational capabilities and power resources. Locally registered devices use authentication and authorization services through a local authorization program that is run by the cluster heads in the IoT network. The second layer includes CH nodes, authority nodes, edge-computing nodes, and gateways. CH nodes can securely communicate in the blockchain environment that deploys a lightweight consensus. The local permissioned HLF blockchain is implemented in this layer. The last layer consists of BSs in cellular networks. This higher layer, which consists of resources with high computational power, can be arranged as a set of separate structures under the HLF blockchain [52]. Robust asymmetric cryptography mechanism deployment can be achieved in this layer. The security and privacy are guaranteed with the implementation of the Global Blockchain and sophisticated security approaches to the high-level layer (Layer 3).

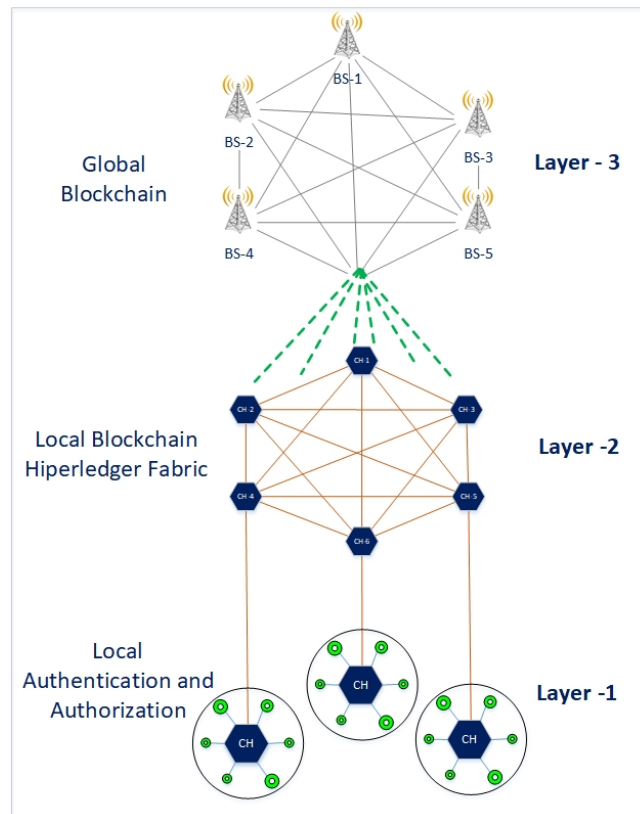


Figure 1. Multi-Layer model for Internet of Things (IoT) network.

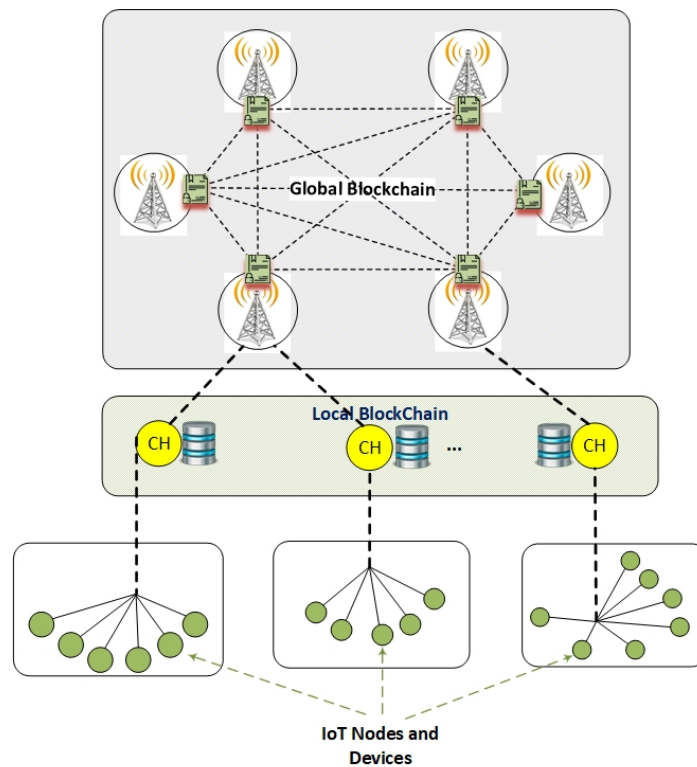


Figure 2. The network model based on a multi-layer structure that implements a local authorization service in the infrastructure level, a local blockchain, and public chain in the remaining two layers.

3.1. LAYER-1

This level includes IoT objects, nodes, and devices, as well as network elements for communications, network procedures, and protocols. Unsupervised hybrid clustering algorithms, as described in Section 4, convert the entire IoT network into multiple clusters and to form the layers. Each cluster is associated with a powerful device chosen as CH. IoT devices and nodes are geographically distributed non-uniformly. The devices are authenticated and authorized to the network through a local authorization and authentication services to guarantee the privacy and security inside each cluster.

The intra-cluster security and privacy are facilitated by local CH nodes acting as edge processing nodes, as shown in Figure 2. Such an approach can enhance the implementation efficiency while also addressing the issues that are associated with globally centralized cloud computing.

A lightweight session key is assigned to devices when they authenticate to associated CH nodes and establish communications. The session key period validation is carried out by the cluster heads in order to perform the authorization and authentication. The registration services and authentication management, as well as authorization, are also locally performed by the CH nodes to improve scalability and address device heterogeneity. CH nodes maintain the addition of new devices to the network through a local registration process. The cryptographic key distribution or session keys allow the node authentications. Less power-hungry cryptography is provided by edge computing. Alternatively, CH nodes for IoT devices with limited resources could provide long-term cached session keys (cryptographic keys).

Lightweight session keys are assigned by CH nodes in order to maintain the authorization of registered nodes as an authorization entity and authenticate them to the network. Symmetric keys and lightweight cryptography are proposed to tackle the scalability challenges and the limitation of IoT devices with constrained resources. CH nodes perform the following four tasks:

- a new node registration to the network as a new entity;
- session key (cryptographic key) distribution and assignment;
- communications management and initiation; and,
- secure communications management and establishment.

Symmetric key-wrapping encrypts the lightweight session keys, called the distribution keys. Every single communication is protected with a session key. The session key is a symmetric key that has a unique ID and a period of validity. The use of cryptographic keys (credential management) for encryption, message authentication, and decryption is managed by secure communication. Consequently, selected CH nodes are responsible for managing cryptographic keys. Figure 3 illustrates the overview of the authorization procedure.

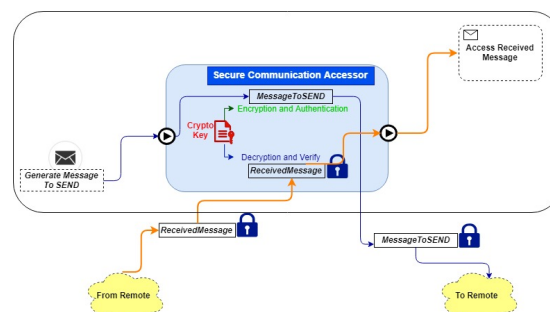


Figure 3. Local authorization service among IoT entity and Cluster Head (CH) nodes for secure communication.

3.2. LAYER-2

The second layer connects all of the selected CH nodes under the serving Base Station (BS) units. Cluster heads collect and forward data to the higher layer. All of the nodes in

the second layer run a private LBC in a distributed manner to achieve a consensus that is based on the defined consensus algorithm [5]. BS and CH units generate and verify blocks, handle communications with non-consensus devices and nodes within the same layer, and broadcast blocks to each cluster. Trusted nodes provide interfaces to subordinate and superior layers employing the blockchain protocol. The HLF blockchain platform is proposed for this layer.

The proposed network model needs to consider secured CH communications while addressing the resource-constrained and decentralized node distribution of the IoT network. The blockchain mining process is computationally intensive and time-consuming. Hence, it is not ideally suited for implementation in an IoT system. Therefore, the proposal suggests a lightweight, private, decentralized blockchain-based method for data communications built upon distributed consensus. It is important to consider the limitation (resource-constraint) of devices in the IoT system. Thus, a lightweight cryptographic mechanism is to be implemented.

In the proposed model, the interaction between CH nodes and other networked elements are based on local permissioned HLF blockchain [53] platform. HLF is a private permissioned blockchain and it is based on an execute-order-validate architecture. In this architecture, the transaction execution (via smart contract) is separated from transaction ordering in order to achieve better scalability and modular consensus implementations.

The main elements of the proposed model are organizations (base stations), IoT nodes and objects, ordering clusters, and peers (including endorser and committer, membership service providers (MSPs), and channels [54]), as shown in Figure 4.

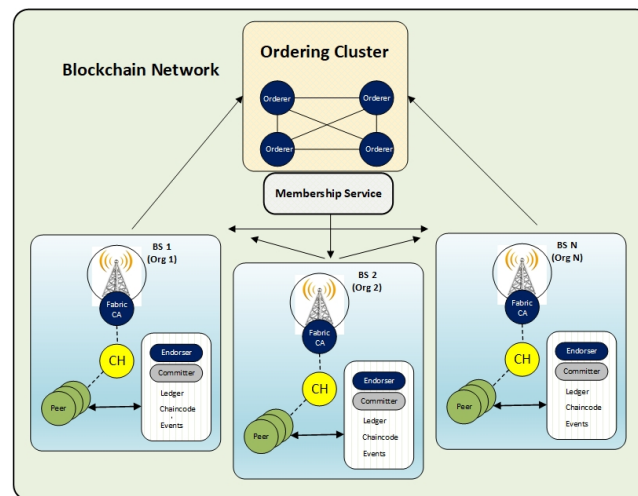


Figure 4. The network architecture of the IoT-enabled cellular system combined with the blockchain technology and smart contracts.

The peers maintain distributed ledger and execute transactions. A peer node can be an endorser or committer or both. The orderers are responsible for ordering all of the transactions in the network. In addition, orderers propose new blocks and seek consensus. Ordering service is a collection of orderers. All peers are committers by default. Ordering service sends the ordered state updates in the form of a block of transactions, and committers maintaining the ledger. The peer validates transactions of a new block, commits the changes locally as a copy of ledger, and updates the blockchain by appending it on the block. Peers also can be an endorser for endorsing transactions. An endorser executes the smart contract (ChainCode in HLF) and appends the results with its cryptographic signature (called endorsement) before sending it back to the client. In the proposed model, CH nodes can take the endorsers or committer roles, based on predefined roles.

MSP carries out the authentication services in the Hyperledger network. MSP has to verify the network nodes identity. The organizations are the logical representation of

the Hyperledger framework. They are responsible for the management of the network members with the help of MSP. Channels facilitate the various connections within the network between its different elements while using private or dedicated channels. The committers perform the validation and update of the shared ledger.

The Hyperledger Blockchain is implemented through various transactions for data collection and data transmission. Transactions are defined using smart contracts [54]. Base stations in the high-level layers provide an organization for the blockchain implementation. They are connected to CH and different nodes in the IoT system. The ordering cluster handles the transactions and queue orders while providing a shared channel for different peer-to-peer communications. Additionally, the ordering service performs messages broadcasting, including transactions, and creating transaction blocks. IoT devices send transactions to ordering clusters while using Ordering Service (OS) to make a block of transactions. Defining the IoT nodes in the blockchain network to have an endorser or committer role depends on various metrics, e.g., the network configuration. Aside from validation tasks and updating the blockchain state, the committer node is responsible for block addition to the blockchain ledger.

An IoT node becomes an endorser through submitting an endorsement request. This request is sent to the endorser node for approval and consistency monitoring. The process of consistency check proceeds with the smart contract execution. The endorser sends back the response to the associated IoT node requests and grants a specific read and writes access.

The transaction block creation is performed by ordering clusters through the OS. The transaction blocks are distributed to all CH nodes. The blockchain system in this level updates the ledger, and transactions are added to the ledger along with IoT node specifications. A copy of the Blockchain ledger is shared with all of the peers in the network after validation.

3.3. LAYER-3

This layer consists of a distributed networked collection of BS nodes acting as an organization owner. Base stations manage devices, generate data, and process requests in a cloud server manner. The trusted nodes in this layer have powerful computing resources with less power and processor limitations. Consequently, more robust asymmetric cryptography mechanisms are proposed for this level with the aid of the global blockchain.

The high-level layer consists of BS nodes that can perform independent mining tasks without reliance on the central authentication servers. The nodes in this layer are computationally powerful while forming a distributed network topology. Therefore, deploying a suitable global chain, such as the global Ethereum blockchain framework, along with more sophisticated security techniques is feasible. The deployment of asymmetric cryptography, such as Elliptic Curve Cryptography (ECC) [55], is an appropriate solution for this layer. The blockchain-based system implementation enhances the level of privacy and security while guaranteeing data integrity. The higher layers do not include any central node, while the devices are data independent. The blockchain network records the transaction exchange between the nodes of this layer. The cluster heads, base stations, and computing edge nodes initiate the globally distributed trust relationship service mechanism among other network members.

The peer-to-peer nature of the blockchain provides a suitable solution for a globally distributed security framework between different network entities, such as BS nodes. The communication among CH nodes and computing edge nodes is done through implementing the blockchain-based communication with the use of certificates. Smart contracts maintain the distribution of the certificates to perform a trustful communication within the blockchain system among different nodes in this layer. CH nodes are required to sign the certificates. The proposed blockchain-based model is enhancing the distributed trust between two CH nodes and related BS nodes when they collaborate for authorization of their

entities. It also enhances the trust, while an entity or IoT node establishes communication with other nodes in separate clusters under each BS.

The blockchain-based system maintains smart contract execution in order to avoid the requirement of using domain names and fixed addresses while the nodes establish the communications. The fixed addresses and domain names are not needed for the cluster heads in the proposed model for their communications with the edge devices as well as to execute smart contracts.

4. Framework Implementation

4.1. Network Self-Clustering

Figure 5 shows the clustering approach for the IoT network. Figure 6 shows the flow of the proposed network clustering algorithm. It can be seen that the clustering is done with the utilization of metaheuristic algorithms [56]. Metaheuristic algorithms are based on a close interaction between computational practices and optimization. The main advantage of these methods is that they are untrapped in local optimal points [57]. Therefore, these approaches seek all over the entire search space. Furthermore, the control within the metaheuristic algorithms is fully distributed among individuals (network nodes and participants). These individuals communicate with each other in a localized manner. The system response is robust, and the application for environment changes is fast [56–58].

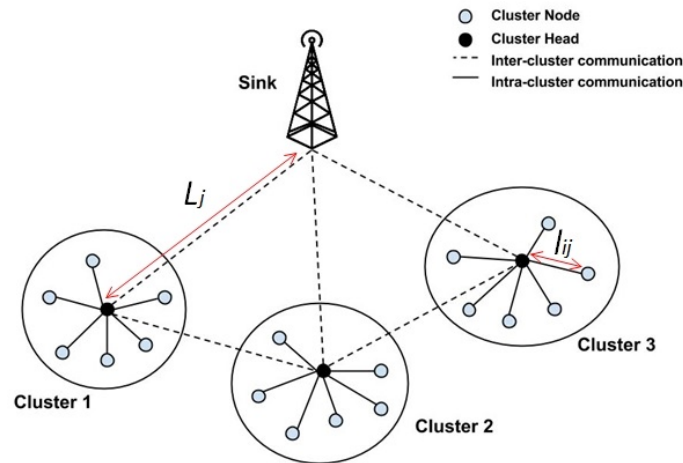


Figure 5. Network clustering scheme for cellular IoT network.

GA is a population-based algorithm that is a subset of metaheuristic methods. It shows a good global-based exploration performance for the search of a problem space [59]. Therefore, GA is proposed in this research for the heterogeneous IoT network clustering. Furthermore, a good local-based exploration mechanism within the search space is required in order to evaluate a single solution. While SA indicates very good performance in this manner, a hybrid mechanism (built upon GA and SA) is chosen in this paper to optimize the proposed IoT network clustering [60].

The clustering approach also reduces the latency and overhead in the IoT systems via the overall minimization of communication distances among IoT objects and selected cluster heads. With clustering, a lower number of nodes require long-distance transmissions to BS nodes. Therefore, the total energy consumption for the entire system is reduced, while the network coverage is enhanced [59–61]. The clustering-based approach helps to leverage the blockchain technology application efficiency by reducing the deployment complexity. The entire network is divided into non-overlapping clusters that are managed by the CH nodes. Other cluster members communicate with the CH nodes for data transmission.

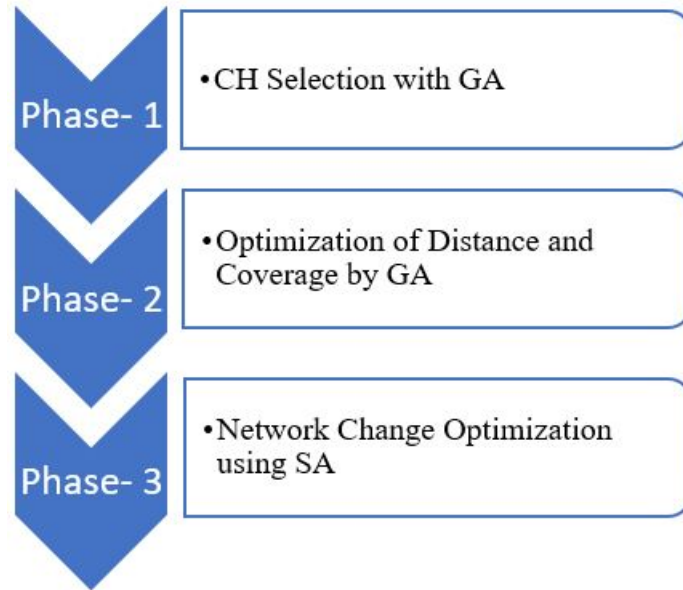


Figure 6. The flow of the Genetic Algorithm-GA-Simulated Annealing (GA-GA-SA) clustering algorithm.

The idea is to achieve the clustering through the deployment of the evolutionary computation algorithms within the network. In the proposed model, critical network attributes, such as the distance, network coverage, energy, and load, are the parameters considered for the clustering of the nodes.

A hybrid algorithm (consisting of a Genetic Algorithm and Simulated Annealing) is employed for the selection of a cluster head as well as for cluster optimization, as shown in Figure 6. The proposed self-clustering approach allows for avoiding a uniform distribution of nodes and clusters. This is done to model the heterogeneous nature of the IoT network. The total number of clusters as well as the number of nodes that belong to each cluster are not predefined. Besides, the proposed clustering enhances the flexibility of nodes deployment in the IoT network. The nonuniform distribution of nodes in each cluster is considered. Consequently, the lifetime of the entire network increases, while the energy dissipation among the CH nodes is more uniform.

4.1.1. GA Phase: CH Selection with Genetic Algorithm

The most critical factor in the IoT network design is satisfying the energy constraint. Longer network operation can be achieved through the shortening of communications and transmission links as well as by reducing the power consumption. Shorter communication links are achievable by grouping nodes into independent clusters. Such an approach facilitates the aggregation and forwarding of data, because each cluster member needs to exchange its information with the associated CH. The calculation of energy consumption uses the first order radio communication model [62]. The radio energy dissipation for transmitting or receiving a bit of data is equal to E_{elec} . Energy dissipation for transmitting n bits of data from the transmitter to the receiver node at the distance l can be calculated, as follows [63]:

$$E_{tx}(n, l) = E_{elec} \times n + E_{amp} \times n \times l^2, \quad (1)$$

while the energy dissipation volume in a node to receive n bits of data is formulated as:

$$E_{rx}(n) = E_{elec} \times n, \quad (2)$$

where, E_{elec} is the dissipation of radio energy and E_{amp} is transmission amplifier energy dissipation.

Consequently, the total energy requirements to send and receive n data bits between two nodes located at the distance l comprise two main elements. The first component is the energy for amplifying data, transmission, and receiving. The second component includes the energy for the data processing by the node. The current leakage is considered to be negligible in low voltage and high-frequency systems. Equation (3) denotes E_{ll} the total energy loss for the distances that are shown in Figure 5.

$$E_{ll} = \sum_{j=1}^k \sum_{i=1}^{m_j} \left[l_{ij}^2 + \frac{L_j^2}{m_j} \right], \quad (3)$$

where, L_j represents the distance between the cluster-head and computing edge node; the l_{ij} represents the distance between a node and its related cluster head (Figure 5); k represents a number of cluster heads; and, m represents the total number of nodes on the network.

Clustering is performed by considering the node residual energy, node distance from the BS, number of CH nodes, and CH distance from the other cluster heads (intra- and inter-cluster distances).

The cost function for the optimization problem considers the total transmission distance as a key metric that is to be minimized. Furthermore, the fitness function also takes the number of cluster heads into account while optimizing the network load. The following multi-objective cost function evaluates each individual node in the GAs algorithm:

Minimize:

$$cost(f_1) = \omega_1 \left(\frac{E_{dd}}{D} \right) + \omega_2 \left(\frac{CH_i}{m} \right) + \omega_3 \left(\frac{Load}{m} \right), \quad (4)$$

where, E_{dd} is the sum of CH distances to all individual nodes and the sum of the computing edge node distances to all CH nodes, CH_i indicates the number of cluster heads, D is the network scaling dimension, m is the total number of nodes, $Load$ is the max network load, and ω_1 , ω_2 , and ω_3 are predefined constant weights.

The goal is to attain a fewer number of CH nodes and enhance the energy. The weights ω_1 to ω_3 , with values between 0 and 1, represent the importance of the key metrics during the optimization procedure. Their values are chosen according to the importance of cost function factors [60]. GA minimizes the cost function at this stage. Figure 7 shows the initial phase of the clustering algorithm.

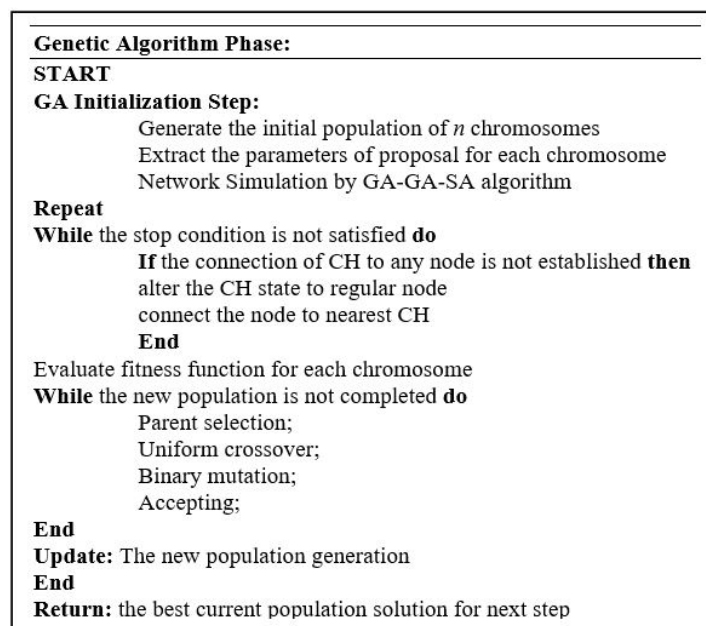


Figure 7. GA algorithm pseudocode.

4.1.2. Optimization of Distance and Coverage by GA

The GA approach helps to achieve distance optimization through its self-organized feature. The next GA starts with the final global solution of the first step GA as the initial solution, as outlined in Figure 6. This step can effectively formulate the mobility of different nodes. The proposed GA method also takes network coverage optimization into account.

The second GA phase enhances the GA solution of the first phase by a local search strategy. Optimizing the distance between CH to a node, and the CH to the sink or edge computing node results in the minimization of the total network energy dissipation. The distance-based equation is deployed in order to cluster the nodes in multi groups in the previous step and define the number of clusters through implementing the GA algorithm. The initial population for the current stage is generated from the best solution of the last phase. A multi-objective cost function is used in the GA optimization step. The distance is optimized while maximizing the coverage:

Minimize:

$$cost(f_2) = \omega_4(E_{II}) + \omega_5(1 - Coverage), \quad (5)$$

where, E_{II} is detailed in (3). Coverage shows the provided network coverage by nodes, ω_4 and ω_5 are predefined constant weights.

4.1.3. Network Changes Optimization Using SA

Simulated Annealing is a meta-heuristic algorithm that is chosen to perform the network changes, including node addition to, moving in, and removing from the IoT network. Generally, a random primary solution is required to start SA. However, in the proposed hybrid GA-GA-SA approach (Figure 6), the initial solution for SA is selected from the final GA global solution in the previous step. A local search strategy is deployed to improve the network changes by the SA algorithm. A new solution (called $Solution^{new}$) is generated at every iteration of SA that is located in the current solution ($Solution^{current}$) neighborhood area. The case $Cost^{new} < Cost^{current}$ means that the current solution is replaced by the new solution. Otherwise, the new solution can be accepted. The same cost function of the first GA (expression (4)) is considered for SA evaluation at each iteration. Figure 8 shows the SA algorithm.

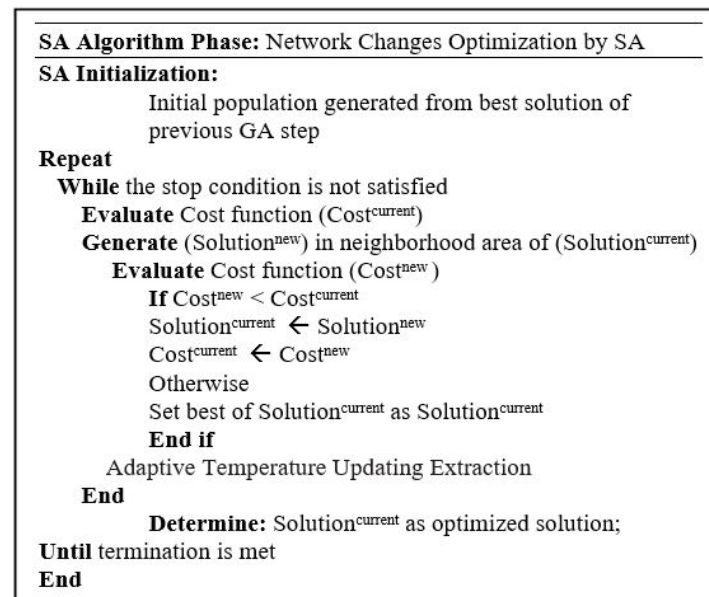


Figure 8. SA Algorithm pseudocode.

4.1.4. Clustering Results

In order to study the performance of the proposed clustering algorithms, a network environment for the IoT devices was simulated, as shown in Figure 5. It included 100 nodes randomly generated and distributed in a 2-D network. MATLAB 2018a was employed since it offered a reliable environment for clustering algorithms, facilitated a straightforward simulation of algorithms, so that the results could be ultimately compared. Table 1 provides the GAs parameters deployed in this scenario.

Table 1. GAs Parameter Settings.

GAs Parameters	Value
Population Size	30
Selection Type	Proportional Selection
Recombination Percentage	0.1
Crossover Percentage	0.5
Crossover Type	One-Point
Mutation Percentage	0.4
Mutation Rate	0.05
Generation Size	500

The GAs started from a specific number of individuals, termed population. Each individual in proposed GA algorithms was elevated while using combined cost functions presented in Equations (4) and (5). The network configuration changes were detected by the SA section and then optimized the network accordingly. Table 2 provides the SA parameters used in this scenario.

Table 2. SA Parameter Settings.

SA Parameters	Value
Max Iter SA	1000
T initial	0.001
T final	0.000
Pchange Max	0.05
Pchange Min	0.02

In the simulation, the energy loss per bit for transmitter or receiver (E_{elec}) was considered to be equal to 50 nJ/bit ($E_{elec} = 50$ nJ/bit), while the constant value for transmission amplifier was $E_{amp} = 0.1$ nJ/bit/m² which was in line with the reported work [64].

The proposed clustering algorithm was benchmarked against four following algorithms reported in the literature: ASLPR [60], ERA [65], FSFLA [64], and GAPSO [50].

Application-Specific Low Power Routing (ASLPR) is based on evolutionary algorithms adopted for Wireless Sensor Network (WSN) applications. It uses GA and SA for CH nodes selection. Energy-aware Routing Algorithm (ERA) is for cluster-based WSNs. The residual energy of the CH nodes and the intra-cluster distances is considered in ERA for cluster formation. Fuzzy Shuffled Frog Leaping Algorithm (FSFLA) employs the memetic Shuffled Frog Leaping Algorithm (SFLA) in order to optimize the Mamdani fuzzy rule-base table based on the application specifications. This protocol deals with node energy and intra-cluster distances as well as with network lifetime. Genetic Algorithm and Particle Swarm Optimization (GAPSO) [50] is proposed in order to form clusters in the IoT environment. All of the protocols were evaluated within the same simulated network environment. Each algorithm at the end of its optimization resulted in a different number of clusters and cluster heads.

The obtained simulation results indicate the effectiveness of the proposed clustering model as well as the efficiency of the algorithm to minimize the distances and the total network energy. Figure 9a illustrates the formed clusters for GA-GA-SA with the centrally

located base station with 100 nodes that were randomly distributed in a network coverage area of 150 (m) \times 150 (m). Figure 9b–d show that the proposed GA-GA-SA performs better when compared to the other algorithms by lowering the network load, minimizing the distances, and, therefore, increasing the network coverage.

4.2. Blockchain Implementation

4.2.1. Development Environment

We deployed simulation models in two different environments associated with each level of the multi-layer network in order to demonstrate the feasibility and practicability of the proposed blockchain framework. The first model implements the HLF blockchain in Layer-2 encompassing IoT devices, CH nodes (peers), APIs, and an organization. The global blockchain deployment simulator at Layer-3 is conducted to compare both Etherume and HLF metrics implementations. The simulation model at Layer-3 uses a workstation as the BS server running the blockchain applications.

The Layer-2 implementation environment was created in order to study the efficiency of the proposed blockchain framework of the multi-layer model, as illustrated in Figure 10. It also shows the means of connection between various entities consisting of IoT devices, IoT server, and blockchain network. The IBM Cloud was used to host development tools and technologies for implementing the IoT devices. IBM Watson IoT Platform [66] was chosen to host IoT devices and gateways. The Node-Red server provided communication between the IoT devices and servers while using the Constrained Application Protocol (CoAP). Physical nodes are simulated in the IBM IoT Watson platform and connected to related Cloud foundry services on the IBM Cloud. The IoT server is organized using a virtual environment that was integrated with various virtual nodes, and a lightweight permissioned HLF blockchain framework is utilized to grant the security for Layer-2. The HLF network within the experimental setup consists of four peers and an orderer node running as docker images using docker containers [67]. The open-source HLF (v1.4) blockchain framework was implemented and hosted by Linux foundations. The Ubuntu Linux 18.4 LTS is the operating system hosted by Intel Core i7-3770 @ 3.4 GHz processor and 16 GB memory. The docker environment is run by the docker engine (version 19.03.8). The configuration of docker images and containers is provided by the docker-compose (version 1.17.0) as the Integrated Development Environment (IDE).

A smart contract was installed and instantiated on peers nodes, and data storage was allocated in order to write a block of transactions to the blockchain ledger. The composer-playground is a web interface for designing and implementing smart contracts and managing transactions and assets. The composer Command Line Interface (CLI) provides an environment to deploy, implement, and execute smart contracts and related definitions by the developers. The peers were set up to use the CouchDB for managing the state data that can handle the complex queries against the transaction logs. The Chain Code (CC) was modelled as JavaScript Object Notation (JSON). The client application can invoke a CC to access the state database and perform various queries, such as put, get, and delete, through APIs. Different blockchain functions were defined by deploying a REST server to directly provide RESTful APIs that can be invoked while using a web client or a virtual device. The user can invoke relevant APIs using GET or POST to submit various transactions through HTTP requests. The REST server hosts the Fabric client application to communicate with the HLF network through google Remote Procedures Calls (gRPC) system deployment. All of the peers in the network have a copy replica of the ledger. The ledger has two parts transaction log and all the recorded state changes. The state data also consists of the key-value pairs that are version. All the state database changes are recorded in time order in the ledger, and the blocks are cryptographically linked together. The orderer node ensures ledger consistency by implementing the PBFT algorithm. The HLF framework supports the Execute-Order-Validate and Commit transaction model.

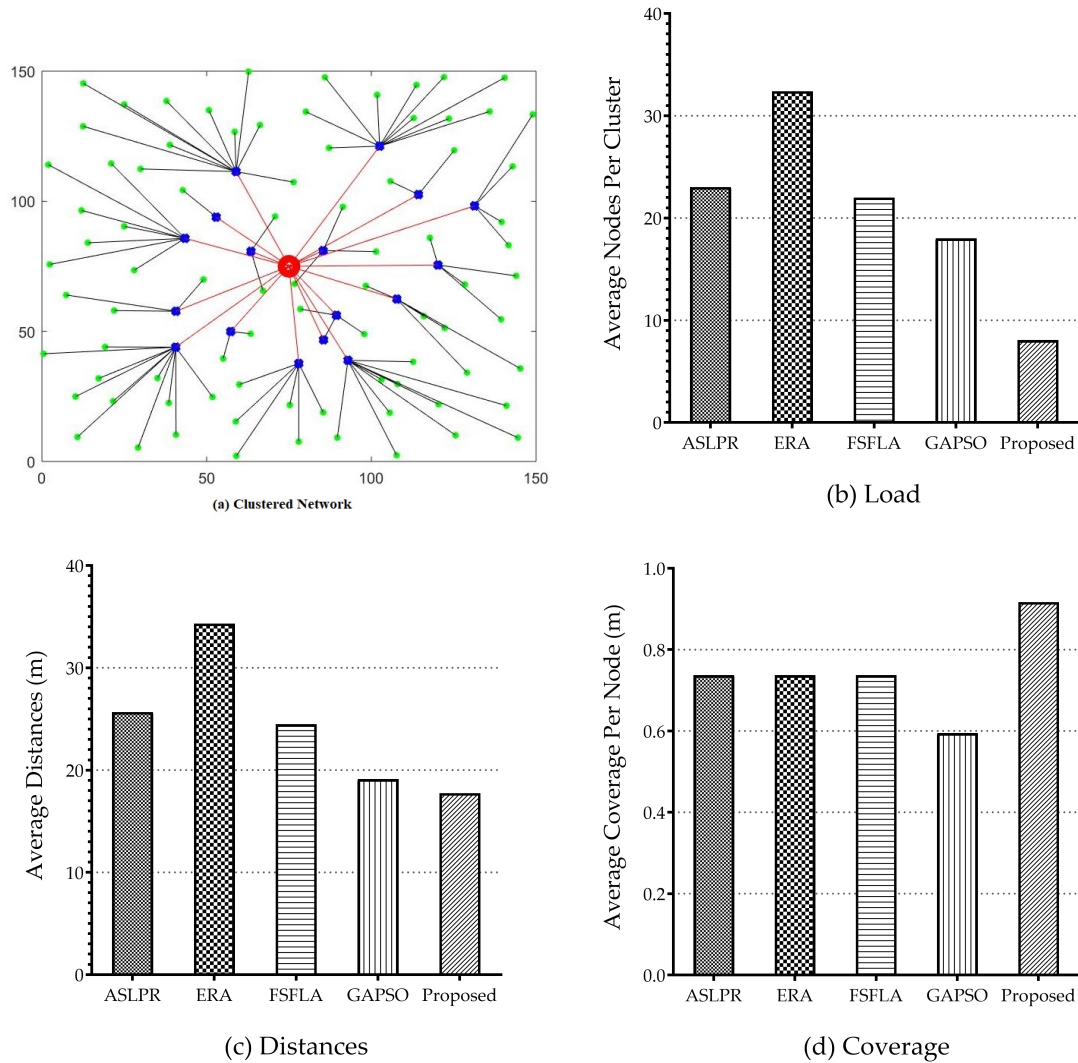


Figure 9. Performance of the proposed clustering algorithm. (a): clustered network and CH positions. (b–d): benchmarked performance in terms of load, distances, and coverage, respectively.

The Layer -3 simulation model was created while using a workstation as the BS server running the blockchain applications. This environment facilitated measuring the throughput and latency parameters of the Ethereum and Hyperledger private networks. The networks were set up in similar conditions and provided with a virtually generated workload. A distributed environment that includes the two blockchain networks was considered for the experimental set up. The simulation models used a workstation with Intel Core i7-3770 @ 3.4 GHz processor and 16 GB memory as the BS. For simplicity, the Etherume network was deployed with just one mining node. The experimental results are presented in Section 4.3.

4.2.2. Smart Contract for Modeling Transactions

The Hyperledger Composer [68] hosted the blockchain applications and facilitated the design and implementation of smart contracts as well as blockchain applications. A business network was deployed in the Hyperledger Composer through a set of open development tools. The members of business networks were participants. They could submit related transactions. The participants were the owners of IoT devices (CH and related BS nodes) with the management and access abilities for their devices. Assets were

services, devices, properties, and goods that were registered and stored within the network. In the reported study, the assets represented IoT devices, including sensors, actuators, or IoT nodes. Each device could be identified through the device ID, device type, device name, device owner, timestamp, event, and value. The presented nodes, including CH ones, were modeled as a different type of assets in the simulation. Transactions represented a logical process within the smart contracts. The implemented model stored the data checksum, data pointers, operations, and ownership of data in the blockchain ledger, while the actual data were held in a separate cloud-server or off-chain storage system. Smart contracts interacted with assets and participants. Besides, a smart contract could set various rules and conditions to perform multiple actions, such as read, create, update, or delete, within the blockchain network. The logical transaction operations were defined in smart contracts as transaction process functions. Smart contracts also included the queries definitions written in a bespoke query language to extract data from the blockchain network. The communication between the blockchain network, IoT device, and the web application was performed by REST APIs that were generated by the composer-rest-server.

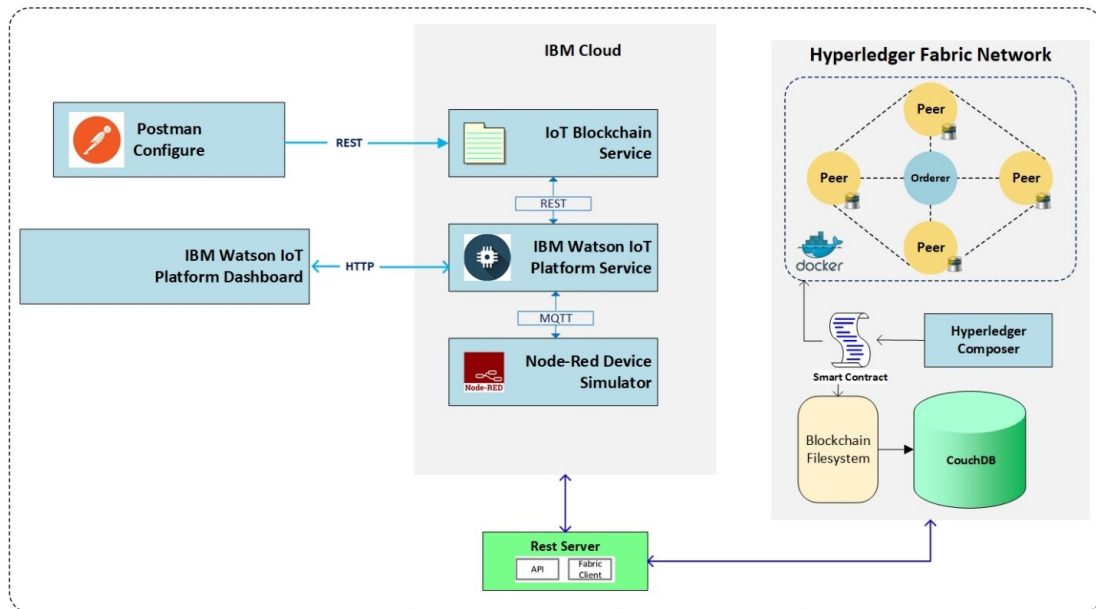


Figure 10. The implementation structure of the blockchain IoT framework.

4.3. Performance Evaluation

The primary objective of a blockchain application is to maintain a number of submitted transactions by the participants. The submitted transactions then proceed to the verification and ordering process, which results in a block generation and storing the transaction outcome on the blockchain ledger. The following metrics are presented by Hyperledger Performance and Scale Working Group [69] to measure the blockchain application performance:

- Transaction Throughput, i.e., the total number of committed transactions by the blockchain System Under Test (SUT) in a given time period in seconds.
- Transaction Latency, i.e., the amount of time that is taken for a transaction to be stored on the blockchain ledger.

The system was tested to evaluate the performance of the proposed model in terms of both the latency and throughput. The results were benchmarked against the parameters reported in the literature with the aim of demonstrating the efficiency of the designed framework. The evaluation was conducted while using the Hyperledger Caliper [70] to facilitate the specific blockchain configuration by the administrator.

In the proposed model, the latency represents the time that is required by CHs to verify new blocks. The block size is an essential factor that affects both node and network latency. The latency is measured by the time that the system requires to reach consensus after the node starts to detect the new block validations. The analysis of the system was conducted with a set of transactions, such as Open, Transfer, and Query. The results were provided for Hyperledger Fabric (proposed blockchain for Layer 2) and Ethereum (standard global blockchain). Table 3 presents the simulation results for evaluating latency and throughput for three different transaction types within HLF implementation. The average latency decreases by implementing a multi-layer model. In this model, only a portion of the nodes (i.e., CHs) is contributing to new blocks validation. Table 3 also presents the Ethereum implementation results. It can be seen that the proposed lightweight HLF blockchain is superior when compared to the Ethereum as a global blockchain technology.

Table 3. Hyperledger and Ethereum performance metric summary (H: Hyperledger Fabric, E: Ethereum).

Name	Send Rate (TPS)		Max Latency (s)		Min Latency (s)		Avg Latency (s)		Throughput (TPS)	
	H	E	H	E	H	E	H	E	H	E
Open	20.2	22.7	0.38	7.05	0.04	2.12	0.18	4.58	20.1	10
Query	10	10.2	0.07	0.02	0.01	0.01	0.01	0.01	10	10.2
Transfer	10	10.7	0.38	7.13	0.06	2.07	0.19	4.63	10	6.7

Despite security and privacy, latency and throughput are essential performance metrics when selecting an appropriate blockchain platform for IoT applications. The resource allocation for the blockchain network must be done in order to meet the latency requirements (for a given input load). A further experiment was conducted to analyze the SUT behavior consisted of multiple rounds of benchmarks with different transaction sending rates. The sending rates varied from 20 to 500 Transactions per Second (TPS), and 1000 transactions were generated for each benchmark to measure the maximum, average, and minimum transaction latency and throughput. Figure 11 presents the maximum, average, and minimum transaction latency for each round of experiments. The minimum latency remained below 1 s during the experiments, while the maximum latency proliferated as the send rate reached the 100 TPS. Figure 12 illustrates the transaction throughput results for varying transaction sending rate. The throughput remained around 100% while the sending rate was up to 110 transactions per second. A significant drop in the throughput was observed as the sending rate increased to 110 TPS, which was the maximum usable sending rate for the SUT.

This experiment only considered an individual client in the blockchain network to generate all the transactions. As expected, the performance of the blockchain network highly depends on the underlying hardware. The HLF provides a three-stage revolutionary architecture known as execute-order-validate, in which every stage depends on previously executed transactions.

Our experiments revealed that the proposed HLF-based blockchain model for IoT application could process up to 110 transactions per second while maintaining a 100% transaction throughput and an average latency of 500 milliseconds, with a maximum of 110 TPS, with throughput that is very close to 100%. A send rate of 100 TPS is sustainable, as the actual throughput is around 100%. However, increasing the send rate to 100 and 200 TPS only yields to a marginal throughput decrease. This lead to the conclusion that our setup can sustain a send rate of about 110 TPS. Therefore, our proposed architecture could support real-time provisioning of multiple 5G-enabled IoT applications without imposing any considerable latency to the process.

The maximum latency grows to nearly 15 s as the number of input transactions increases. This is due to resource restrictions of the containers that are allocated to the peer nodes. The minimum latency remains almost constant, as there are no high loads imposed

on the peer nodes at the beginning. Additionally, the blockchain configuration (e.g., the block size, the number of channels, ordering service, users, endorsing nodes) influences the latency. It can be observed that, in all cases, all of the transactions are successfully completed, i.e., no loss of transactions occurs.

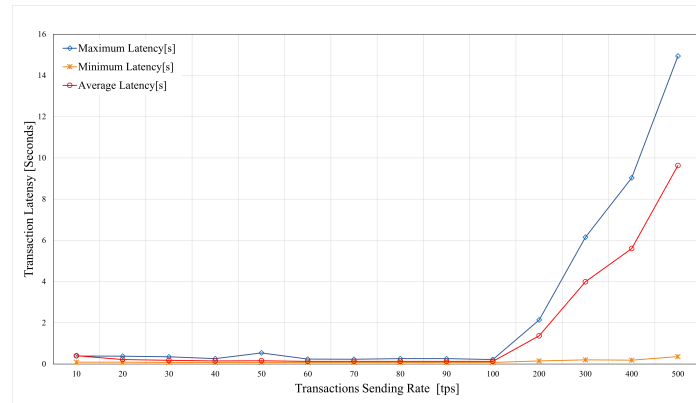


Figure 11. Latency vs. transaction sending rate.

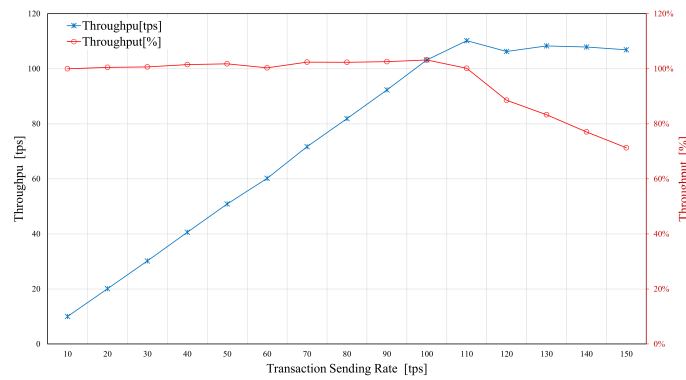


Figure 12. Throughput vs. transaction sending rate.

5. Security Analysis of the Framework

The proposed secured IoT multi-layer model that is based on Hyperledger Blockchain technology offers an overall superiority over the previous works reported in the literature, as illustrated by the metrics comparison that is given in Table 4.

Table 4. Security challenge comparison of blockchain applications in IoT systems.

Ref	IoT Application	Security Challenges				Implemented Consensus	Implemented Blockchain
		Framework Privacy	Heterogeneity and Flexibility	Authentication	Scalability		
[38]	Smart Grids, Smart Cities		Yes		Yes	PoW	Private
[71]	Microgrids, Smart Grids, Vehicle-to-Grids	Yes				PoW	Consortium
[72]	Microgrids, Smart Grid	Yes				PoC	Private
[73]	Big Data, eHealth	Yes			Yes	PoW	Public
[74]	Industrial IoT	Yes	Yes			PoW	Private
[75]	Smart Factory, Supply Chain		Yes			PoS	Consortium
[76]	Industrial IoT, Energy Harvesting networks	Yes	Yes			PoW	Consortium
[77]	eHealth	Yes				PoW	Public
[78]	Mobile edge computing, eHealth		Yes		Yes	PoC	Permissioned
[79]	Cloud computing, V2X	Yes		Yes		PoS	Consortium
[80]	Vehicular Edge Computing		Yes	Yes		PoW	Consortium
proposed	5G MBS	Yes	Yes	Yes	Yes	PBFT, PoC	Consortium

5.1. Framework Privacy

Contracts between different entities are recorded in the blockchain system. Therefore, privacy disclosure assessment is required. The identity of an object is encrypted, and the IoT address is recorded in the blockchain as the pseudonym of the entity. The domain name and fixed address for communication are not required, and the blockchain maintains tasks through running smart contracts, as discussed above. In the IoT network, IP address of an object is encrypted and recorded in the blockchain thus leading to the anonymity of the object. The contract context privacy is guaranteed by the Hash Code of the real context within the blockchain network while minimizing the risk of a privacy leak.

5.2. Heterogeneity and Flexibility

The proposed framework accommodates various configurations for system security in different scenarios. These include the IoT objects with limited resources, the security of sensitive information, high-risk, and broadcasting. The security configuration options can vary, due to the strength of cryptography techniques and characteristics of key lifetimes (strong crypto, short and long key lifetimes, and lightweight cryptography), key distribution mechanisms, the selection of different session keys, such as encryption and authentication, cached session keys, including of one, multiple, and unlimited keys, different owners of keys, and the stability of the fundamental protocols (TCP and UDP). Besides, a certain degree of flexibility is achievable by granting an option to a node or entity to connect or leave the system freely. Changes in the network are recorded in the blockchain through the distributed consensus process.

5.3. Authentication

The process of authentication is implemented in two parts: (1) local authentication and authorization process in the infrastructure layer and (2) rights to objects by smart contracts. The node requirements and respective rights are recorded in the blockchain that was implemented in different segments. The block summary consists of a contract summary. It is accessible at any time. The non-repudiation nature of this summary guarantees the interests of the object.

The multi-layer approach through the network clustering divides the entire IoT network into different tiers, as presented in Figure 2. This includes the local authentication services and globally distributed blockchain-based framework, while separating the external authority. Therefore, the effect of a local authentication service failure or attack to the network is limited to the compromised nodes, while the impact on the network is significantly reduced.

5.4. Scalability

The framework tackles the following scalability challenges: (1) high data traffic and (2) a massive number of IoT objects and devices. The multi-layer structure facilitates multiple cluster implementation and fulfills the scalability issues. Two different CH nodes can establish different secure communications on a client-server basis. CH establishes secure communication with the entities within the same cluster in order to avoid the overhead incrementation. When networked CH nodes start communicating within a framework that is based on the blockchain, the exchange of cryptographic keys is necessary before beginning the client-server communication by which further overhead is reduced.

6. Conclusions

This paper proposes a multi-layer security model for IoT devices functioning under multi-hop cellular networks based on distributed technology of the blockchain. The developed model provides a feasible solution to establish the decentralized application of the blockchain technology for the security of the cellular-enabled IoT network. The hybrid self-clustering EC algorithm, utilizing GA and SA, is developed to fragment the IoT network into clusters in order to provide the multi-layer structure and enhance the network

lifetime. Detailed system implementation is discussed, and the way the blockchain-based model can help to improve the IoT system authentication and authorization is elaborated. The model proposes the open-source HLF blockchain for deployment and verification. The multi-layer model enhances network security, lowers the processing load, and reduces network load and latency. The proposed implementation enhances the efficiency of the communications via the peer-to-peer nature of the blockchain communication and maps it to the device-to-device communication in cellular systems with improved integrity and security. The proposed solution tackles the IoT security challenges, including framework privacy, authentication, heterogeneity, and flexibility, as well as network scalability. The proposed hybrid clustering algorithm has been compared with four existing protocols. The simulations study demonstrates that the proposed algorithm outperforms the competitors in terms of various performance metrics, including network load, network coverage, and distances. The performance of the proposed multi-layer blockchain-based framework was evaluated. It was found that the lightweight blockchain was more effective than the global blockchain Ethereum.

The focus of our future work will be on the deployment of a practical scalable test-bed configured as MBS framework of IoT devices to study, analyze, and compare the performance in the real world environment.

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

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

Abbreviations

The following abbreviations are used in this manuscript:

ASLPR	Application-Specific Low Power Routing
BCS	Blockchain Structures
BS	Base Station
CC	ChainCode
CH	Cluster Head
CLI	Command Line Interface
CoAP	Constrained Application Protocol
D2D	Device-to-Device
EC	Evolutionary Computation
ECC	Elliptic Curve Cryptography
ERA	Energy-aware Routing Algorithm
FSFLA	Fuzzy Shuffled Frog Leaping Algorithm
GA	Genetic Algorithm
GAPSO	Genetic Algorithm and Particle Swarm Optimization
gRPC	google Remote Procedures Calls
HLF	Hyperledger Fabric
IDE	Integrated Development Environment
IoT	Internet of Things
LSB	Lightweight Scalable Blockchain
LHB	Lightweight Hyperledger Blockchain
LTS	Long Term Support
MCNs	Multihop Cellular Networks
MSP	Membership Service Providers

OS	Ordering Service
PKI	Public Key Infrastructure
PoBT	Proof of Block and Trade
PoC	Proof of Concept
PoW	Proof of Work
RoA	Rating of Allocation
SA	Simulated Annealing
SDK	Software Development Kit
SDN	Software Defined Networking
SFLA	Shuffled Frog Leaping Algorithm
SI	Swarm Intelligence
SUT	System Under the Test
TLS	Transport Layer Security
TSP	Transactions Per Second

References

1. Familiar, B. *Microservices, IoT, and Azure*; Springer: Berlin/Heidelberg, Germany, 2015.
2. Amoozadeh, M.; Raghuramu, A.; Chuah, C.N.; Ghosal, D.; Zhang, H.M.; Rowe, J.; Levitt, K. Security vulnerabilities of connected vehicle streams and their impact on cooperative driving. *IEEE Commun. Mag.* **2015**, *53*, 126–132. [[CrossRef](#)]
3. Ho, G.; Leung, D.; Mishra, P.; Hosseini, A.; Song, D.; Wagner, D. Smart locks: Lessons for securing commodity internet of things devices. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, Xi'an, China, 30 May–3 June 2016; ACM: New York, NY, USA, 2016; pp. 461–472.
4. Kshetri, N. Can blockchain strengthen the internet of things? *IT Prof.* **2017**, *19*, 68–72. [[CrossRef](#)]
5. Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. LSB: A Lightweight Scalable Blockchain for IoT security and anonymity. *J. Parallel Distrib. Comput.* **2019**, *134*, 180–197. [[CrossRef](#)]
6. Khan, M.A.; Salah, K. IoT security: Review, blockchain solutions, and open challenges. *Future Gener. Comput. Syst.* **2018**, *82*, 395–411. [[CrossRef](#)]
7. Karajeh, H.; Maqableh, M.; Masa' deh, R. Privacy and Security Issues of Cloud Computing Environment. In Proceedings of the 23rd IBIMA Conference Vision: 2020, Valencia, Spain, 13–14 May 2014; pp. 1–15.
8. Guerbouj, S.S.E.; Gharsellaoui, H.; Bouamama, S. A Comprehensive Survey on Privacy and Security Issues in Cloud Computing, Internet of Things and Cloud of Things. *Int. J. Serv. Sci. Manag. Eng. Technol.* **2019**, *10*, 32–44.
9. Ferrag, M.A.; Derdour, M.; Mukherjee, M.; Derhab, A.; Maglaras, L.; Janicke, H. Blockchain Technologies for the Internet of Things: Research Issues and Challenges. *IEEE Internet Things J.* **2019**, *6*, 2188–2204. [[CrossRef](#)]
10. Yang, Y.; Wu, L.; Yin, G.; Li, L.; Zhao, H. A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet Things J.* **2017**, *4*, 1250–1258. [[CrossRef](#)]
11. Mistry, I.; Tanwar, S.; Tyagi, S.; Kumar, N. Blockchain for 5G-enabled IoT for industrial automation: A systematic review, solutions, and challenges. *Mech. Syst. Signal Process.* **2020**, *135*, 106382. [[CrossRef](#)]
12. Al-Fuqaha, A.; Guizzani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [[CrossRef](#)]
13. Wang, S.; Ouyang, L.; Yuan, Y.; Ni, X.; Han, X.; Wang, F. Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 2266–2277. [[CrossRef](#)]
14. Moin, S.; Karim, A.; Safdar, Z.; Safdar, K.; Ahmed, E.; Imran, M. Securing IoTs in distributed blockchain: Analysis, requirements and open issues. *Future Gener. Comput. Syst.* **2019**, *100*, 325–343. [[CrossRef](#)]
15. Granjal, J.; Monteiro, E.; Silva, J.S. Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1294–1312. [[CrossRef](#)]
16. Li, S.; Da Xu, L.; Zhao, S. 5G Internet of Things: A survey. *J. Ind. Inf. Integr.* **2018**, *10*, 1–9. [[CrossRef](#)]
17. Lao, L.; Li, Z.; Hou, S.; Xiao, B.; Guo, S.; Yang, Y. A survey of IoT applications in blockchain systems: Architecture, consensus, and traffic modeling. *ACM Comput. Surv.* **2020**, *53*, 1–32. [[CrossRef](#)]
18. Decker, C.; Seidel, J.; Wattenhofer, R. Bitcoin meets strong consistency. In Proceedings of the 17th International Conference on Distributed Computing and Networking, Singapore, 4–7 January 2016; pp. 1–10.
19. Du, K.L.; Swamy, M. Search and optimization by metaheuristics. In *Techniques and Algorithms Inspired by Nature*; Birkhauser: Basel, Switzerland, 2016.
20. Zhang, Z.; Cho, M.C.Y.; Wang, C.; Hsu, C.; Chen, C.; Shieh, S. IoT Security: Ongoing Challenges and Research Opportunities. In Proceedings of the 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, Matsue, Japan, 17–19 November 2014; pp. 230–234. [[CrossRef](#)]
21. Cooper, D.; Santesson, S.; Farrell, S.; Boeyen, S.; Housley, R.; Polk, W.T. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *RFC* **2008**, *5280*, 1–151.
22. Oriwoh, E.; Conrad, M. 'Things' in the Internet of Things: towards a definition. *Int. J. Internet Things* **2015**, *4*, 1–5.

23. Ukil, A.; Bandyopadhyay, S.; Pal, A. Iot-privacy: To be private or not to be private. In Proceedings of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS 2014), Toronto, ON, Canada, 27 April–2 May 2014; pp. 123–124.
24. Novo, O. Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT. *IEEE Internet Things J.* **2018**, *5*, 1184–1195. [[CrossRef](#)]
25. Pal, S.; Rabehaja, T.; Hill, A.; Hitchens, M.; Varadharajan, V. On the integration of blockchain to the internet of things for enabling access right delegation. *IEEE Internet Things J.* **2019**, *7*, 2630–2639. [[CrossRef](#)]
26. Qu, C.; Tao, M.; Zhang, J.; Hong, X.; Yuan, R. Blockchain based credibility verification method for IoT entities. *Secur. Commun. Netw.* **2018**, 2018. [[CrossRef](#)]
27. Kumar, N.M.; Mallick, P.K. Blockchain technology for security issues and challenges in IoT. *Procedia Comput. Sci.* **2018**, *132*, 1815–1823. [[CrossRef](#)]
28. Li, D.; Peng, W.; Deng, W.; Gai, F. A Blockchain-Based Authentication and Security Mechanism for IoT. In Proceedings of the 2018 27th International Conference on Computer Communication and Networks (ICCCN), Hangzhou, China, 30 July–2 August 2018; pp. 1–6. [[CrossRef](#)]
29. Hassan, M.U.; Rehmani, M.H.; Chen, J. Privacy preservation in blockchain based IoT systems: Integration issues, prospects, challenges, and future research directions. *Future Gener. Comput. Syst.* **2019**, *97*, 512–529. [[CrossRef](#)]
30. Dai, H.; Zheng, Z.; Zhang, Y. Blockchain for Internet of Things: A Survey. *IEEE Internet Things J.* **2019**, *6*, 8076–8094. [[CrossRef](#)]
31. Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. Blockchain for IoT security and privacy: The case study of a smart home. In Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Big Island, HI, USA, 13–17 March 2017; pp. 618–623.
32. Biswas, S.; Sharif, K.; Li, F.; Maharjan, S.; Mohanty, S.P.; Wang, Y. PoBT: A lightweight consensus algorithm for scalable IoT business blockchain. *IEEE Internet Things J.* **2019**, *7*, 2343–2355. [[CrossRef](#)]
33. Pahl, C.; El Ioini, N.; Helmer, S. A Decision Framework for Blockchain Platforms for IoT and Edge Computing. In Proceedings of the International Conference on Internet of Things, Big Data and Security, Funchal, Portugal, 19–21 March 2018.
34. Yáñez, W.; Mahmud, R.; Bahsoon, R.; Zhang, Y.; Buyya, R. Data Allocation Mechanism for Internet-of-Things Systems with Blockchain. *IEEE Internet Things J.* **2020**, *7*, 3509–3522. [[CrossRef](#)]
35. Fabric, W.T.H. Available online: <http://hyperledger-fabric.readthedocs.io/en/release-1.4/> (accessed on 15 October 2020).
36. Klaokliang, N.; Teawtim, P.; Aimtongkham, P.; So-In, C.; Niruntasukrat, A. A Novel IoT Authorization Architecture on Hyperledger Fabric with Optimal Consensus Using Genetic Algorithm. In Proceedings of the 2018 Seventh ICT International Student Project Conference (ICT-ISPC), Nakhon Pathom, Thailand, 11–13 July 2018; pp. 1–5. [[CrossRef](#)]
37. Zhaofeng, M.; Lingyun, W.; Xiaochang, W.; Zhen, W.; Weizhe, Z. Blockchain-Enabled Decentralized Trust Management and Secure Usage Control of IoT Big Data. *IEEE Internet Things J.* **2019**, *7*, 4000–4015. [[CrossRef](#)]
38. Biswas, K.; Muthukkumarasamy, V. Securing Smart Cities Using Blockchain Technology. In Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Sydney, Australia, 12–14 December 2016; pp. 1392–1393. [[CrossRef](#)]
39. Chendeb, N.; Khaled, N.; Agoulmine, N. Integrating Blockchain with IoT for a Secure Healthcare Digital System. In Proceedings of the 8th International Workshop on ADVANCES in ICT Infrastructures and Services (ADVANCE 2020), Cancun, Mexico, 27–29 January 2020; pp. 1–8.
40. Sharma, P.K.; Park, J.H. Blockchain based hybrid network architecture for the smart city. *Future Gener. Comput. Syst.* **2018**, *86*, 650–655. [[CrossRef](#)]
41. Mbarek, B.; Jabeur, N.; Pitner, T. Mbs: Multilevel blockchain system for IoT. *Pers. Ubiquitous Comput.* **2019**, 1–8. [[CrossRef](#)]
42. Oktian, Y.E.; Lee, S.G.; Lee, H.J. Hierarchical multi-blockchain architecture for scalable internet of things environment. *Electronics* **2020**, *9*, 1050. [[CrossRef](#)]
43. Li, C.; Zhang, L.J. A blockchain based new secure multi-layer network model for Internet of Things. In Proceedings of the 2017 IEEE International Congress on Internet of Things (ICIOT). IEEE, Honolulu, HI, USA, 25–30 June 2017; pp. 33–41.
44. Sagirlar, G.; Carminati, B.; Ferrari, E.; Sheehan, J.D.; Ragnoli, E. Hybrid-iot: Hybrid blockchain architecture for internet of things-pow sub-blockchains. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018; pp. 1007–1016.
45. Badr, S.; Gomaa, I.; Abd-Elrahman, E. Multi-tier blockchain framework for IoT-EHRs systems. *Procedia Comput. Sci.* **2018**, *141*, 159–166. [[CrossRef](#)]
46. Xuan, S.; Zhang, Y.; Tang, H.; Chung, I.; Wang, W.; Yang, W. Hierarchically Authorized Transactions for Massive Internet-of-Things Data Sharing Based on Multilayer Blockchain. *Appl. Sci.* **2019**, *9*, 5159. [[CrossRef](#)]
47. Lin, H.; Yang, Z.; Hong, Z.; Li, S.; Chen, W. Smart Contract-based Hierarchical Auction Mechanism for Edge Computing in Blockchain-empowered IoT. In Proceedings of the 2020 IEEE 21st International Symposium on “A World of Wireless, Mobile and Multimedia Networks” (WoWMoM), Cork, Ireland, 31 August–3 September 2020; pp. 147–156.
48. Uddin, M.A.; Stranieri, A.; Gondal, I.; Balasurbramanian, V. A Lightweight Blockchain Based Framework for Underwater IoT. *Electronics* **2019**, *8*, 1552. [[CrossRef](#)]

-
49. Saputro, M.Y.A.; Sari, R.F. Securing IoT network using lightweight multi-fog (LMF) blockchain model. In Proceedings of the 2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Bandung, Indonesia, 18–20 September 2019; pp. 183–188.
 50. Rashid, M.A.; Pajooh, H.H. A Security Framework for IoT Authentication and Authorization Based on Blockchain Technology. In Proceedings of the 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), Rotorua, New Zealand, 5–8 August 2019; pp. 264–271. [CrossRef]
 51. Aslam, S.; Alam, F.; Hasan, S.F.; Rashid, M.A. Performance Analysis of Clustering Algorithms for Content-Sharing Based D2D Enabled 5G Networks. In Proceedings of the International Telecommunication Networks and Applications Conference 2019 (ITNAC2019), Auckland, New Zealand, 27–29 November 2019.
 52. Foundation, L. Hyperledger White paper. *Hyperledger* **2016**, *v2.0.0*, 1–19.
 53. De Angelis, S.; Aniello, L.; Baldoni, R.; Lombardi, F.; Margheri, A.; Sassone, V. PBFT vs. proof-of-authority: Applying the CAP theorem to permissioned blockchain. In Proceedings of the Italian Conference on Cyber Security, Milan, Italy, 6–9 February 2018.
 54. Cachin, C. Architecture of the hyperledger blockchain fabric. In Proceedings of the Workshop on Distributed Cryptocurrencies and Consensus Ledgers, Chicago, IL, USA, 25–29 July 2016; Volume 310.
 55. Joglekar, J.; Bhutani, S.; Patel, N.; Soman, P. Lightweight Elliptical Curve Cryptography (ECC) for Data Integrity and User Authentication in Smart Transportation IoT System. In *Sustainable Communication Networks and Application*; Karrupusamy, P., Chen, J., Shi, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 270–278.
 56. Altay, E.V.; Alatas, B. Performance Comparisons of Socially Inspired Metaheuristic Algorithms on Unconstrained Global Optimization. In *Advances in Computer Communication and Computational Sciences*; Springer: Singapore, 2019; pp. 163–175.
 57. García, J.; Crawford, B.; Soto, R.; Astorga, G. A clustering algorithm applied to the binarization of Swarm intelligence continuous metaheuristics. *Swarm Evol. Comput.* **2019**, *44*, 646–664. [CrossRef]
 58. Sabet, S.; Shokouhifar, M.; Farokhi, F. A comparison between swarm intelligence algorithms for routing problems. *Electr. Comput. Eng. Int. J.* **2016**, *5*, 17–33.
 59. Elhoseny, M.; Yuan, X.; Yu, Z.; Mao, C.; El-Minir, H.K.; Riad, A.M. Balancing Energy Consumption in Heterogeneous Wireless Sensor Networks Using Genetic Algorithm. *IEEE Commun. Lett.* **2015**, *19*, 2194–2197. [CrossRef]
 60. Shokouhifar, M.; Jalali, A. A new evolutionary based application specific routing protocol for clustered wireless sensor networks. *AEU-Int. J. Electron. Commun.* **2015**, *69*, 432–441. [CrossRef]
 61. Eleburuike, I.O.; Adekunle, S.S. *Energy Efficient Wireless Sensor Network Using Hierarchical Routing Technique*; Blekinge Institute of Technology: Karlskrona, Sweden, 2010.
 62. Heinzelman, W.R.; Chandrakasan, A.; Balakrishnan, H. Energy-efficient communication protocol for wireless microsensor networks. In Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Maui, HI, USA, 7 January 2000. [CrossRef]
 63. Guru, S.M.; Hsu, A.; Halgamuge, S.; Fernando, S. An Extended Growing Self-Organizing Map for Selection of Clusters in Sensor Networks. *Int. J. Distrib. Sens. Netw.* **2005**, *1*, 227–243. [CrossRef]
 64. Fanian, F.; Rafsanjani, M.K. Memetic fuzzy clustering protocol for wireless sensor networks: Shuffled frog leaping algorithm. *Appl. Soft Comput.* **2018**, *71*, 568–590. [CrossRef]
 65. Amgoth, T.; Jana, P.K. Energy-aware routing algorithm for wireless sensor networks. *Comput. Electr. Eng.* **2015**, *41*, 357–367. [CrossRef]
 66. Nykyri, M.; Kuisma, M.; Kärkkäinen, T.J.; Hallikas, J.; Jäppinen, J.; Korpinen, K.; Silventoinen, P. IoT Demonstration Platform for Education and Research. In Proceedings of the 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), Helsinki, Finland, 22–25 July 2019; Volume 1, pp. 1155–1162. [CrossRef]
 67. Docker, I. Docker. 2017. Available online: <https://www.docker.com/what-docker> (accessed on 30 September 2020).
 68. Composer, H. Hyperledger Composer Documentation. *Linux Found.* **2018**. Available online: <https://hyperledger.github.io/composer/latest/introduction/introduction.html> (accessed on 1 September 2020).
 69. Performance, H.; Group, S.W. Hyperledger Blockchain Performance Metrics. White paper. 2018; Volume 1. Available online: https://www.hyperledger.org/wpcontent/uploads/2018/10/HL_Whitepaper_Metrics_PDF_V1 (accessed on 30 September 2020).
 70. Caliper, H. Hyperledger Caliper Architecture. *Electronic Article*. 2019. Available online: https://hyperledger.github.io/caliper/docs/2_Architecture (accessed on 15 September 2020).
 71. Kokoris-Kogias, L.; Gasser, L.; Khoffi, I.; Jovanovic, P.; Gailly, N.; Ford, B. Managing identities using blockchains and CoSi. In Proceedings of the 9th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2016), Darmstadt, Germany, 19–22 May 2016.
 72. Aitzhan, N.Z.; Svetinovic, D. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Trans. Dependable Secur. Comput.* **2016**, *15*, 840–852. [CrossRef]
 73. Wang, K.; Shao, Y.; Shu, L.; Zhu, C.; Zhang, Y. Mobile big data fault-tolerant processing for ehealth networks. *IEEE Netw.* **2016**, *30*, 36–42. [CrossRef]
 74. Wan, J.; Li, J.; Imran, M.; Li, D. A blockchain-based solution for enhancing security and privacy in smart factory. *IEEE Trans. Ind. Informatics* **2019**, *15*, 3652–3660. [CrossRef]
 75. Lu, Q.; Xu, X. Adaptable blockchain-based systems: A case study for product traceability. *IEEE Softw.* **2017**, *34*, 21–27. [CrossRef]
-

76. Li, Z.; Kang, J.; Yu, R.; Ye, D.; Deng, Q.; Zhang, Y. Consortium blockchain for secure energy trading in industrial internet of things. *IEEE Trans. Ind. Informatics* **2017**, *14*, 3690–3700. [[CrossRef](#)]
77. Esposito, C.; Santis, A.; Tortora, G.; Chang, H.; Choo, K. Blockchain: A panacea for healthcare cloud-based data security and privacy? *IEEE Cloud Comput.* **2018**, *5*, 31–37. [[CrossRef](#)]
78. Rahman, M.A.; Hossain, M.S.; Loukas, G.; Hassanain, E.; Rahman, S.S.; Alhamid, M.F.; Guizani, M. Blockchain-based mobile edge computing framework for secure therapy applications. *IEEE Access* **2018**, *6*, 72469–72478. [[CrossRef](#)]
79. Liu, H.; Zhang, Y.; Yang, T. Blockchain-enabled security in electric vehicles cloud and edge computing. *IEEE Netw.* **2018**, *32*, 78–83. [[CrossRef](#)]
80. Kang, J.; Yu, R.; Huang, X.; Wu, M.; Maharjan, S.; Xie, S.; Zhang, Y. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet Things J.* **2018**, *6*, 4660–4670. [[CrossRef](#)]

Chapter 4


Hyperledger Fabric Blockchain for Securing the Edge IoT

Chapter 4 includes the article, “Hyperledger Fabric Blockchain for Securing the Edge Internet of Things” published in the MDPI journal of “Sensors”. This article is open access and has been republished in this thesis under the Creative Commons Attribution License. No special permission is required to reuse all or part of article published by MDPI, including figures and tables. For articles published under an open access Creative Common CC BY license, any part of the article may be reused without permission provided that the original article is clearly cited. Reuse of an article does not imply endorsement by the authors or MDPI.

The full text is included in the thesis has some minor modifications. This means that while the content is identical to the published article, there may be stylistic differences.

Article

Hyperledger Fabric Blockchain for Securing the Edge Internet of Things

 Houshyar Honar Pajooh ^{1,*} , Mohammad Rashid ¹ , Fakhrul Alam ¹  and Serge Demidenko ^{1,2} 

¹ Department of Mechanical and Electrical Engineering, Massey University, Auckland 0632, New Zealand; m.a.rashid@massey.ac.nz (M.R.); f.alam@massey.ac.nz (F.A.)

² School of Science and Technology, Sunway University, Subang Jaya 47500, Malaysia; sdemidenko@Sunway.edu.my

* Correspondence: h.pajooh@massey.ac.nz; Tel.: +64-21440684

Abstract: Providing security and privacy to the Internet of Things (IoT) networks while achieving it with minimum performance requirements is an open research challenge. Blockchain technology, as a distributed and decentralized ledger, is a potential solution to tackle the limitations of the current peer-to-peer IoT networks. This paper presents the development of an integrated IoT system implementing the permissioned blockchain Hyperledger Fabric (HLF) to secure the edge computing devices by employing a local authentication process. In addition, the proposed model provides traceability for the data generated by the IoT devices. The presented solution also addresses the IoT systems' scalability challenges, the processing power and storage issues of the IoT edge devices in the blockchain network. A set of built-in queries is leveraged by smart-contracts technology to define the rules and conditions. The paper validates the performance of the proposed model with practical implementation by measuring performance metrics such as transaction throughput and latency, resource consumption, and network use. The results show that the proposed platform with the HLF implementation is promising for the security of resource-constrained IoT devices and is scalable for deployment in various IoT scenarios.

Keywords: Internet of Things; hyperledger fabric; smart contract; security and privacy; data provenance; edge computing



Citation: Honar Pajooh, H.; Rashid, M.; Alam, F.; Demidenko, S. Hyperledger Fabric Blockchain for Securing the Edge Internet of Things. *Sensors* **2021**, *21*, 359. <https://doi.org/10.3390/s21020359>

Received: 7 December 2020
Accepted: 5 January 2021
Published: 7 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Internet of Things (IoT) [1] technologies are associated with the significant growth of generated, collected and used data. At the same time, with the rapid involvement of distributed heterogeneous devices, various aspects of traditional IoT applications and platforms face challenges in security, privacy, data integrity, and robustness [2]. The blockchain has emerged as an innovative engine that can facilitate reliable and transparent data transactions. It has been widely applied to traditional sectors, including finance, commerce, industry, and logistics.

Most IoT platforms and applications depend on centralized architecture by connecting to cloud servers via gateways. Unfortunately, this leads to severe security and privacy risks. Wireless communication between sensor nodes and IoT gateways might also be very susceptible to attack. Cloud servers are potential targets for Distributed Denial-of-Service (DDoS) attacks resulting in significant infrastructure collapse [3]. Moreover, the centralized server solution introduces a single point of failure risk to the entire system.

Networked devices in an IoT system are heterogeneous in terms of their security requirements and resource availability. Resource-constrained devices operate in an open environment that increases the risks of physical and wireless accessibility by adversaries. RSA (Rivest–Shamir–Adleman) [4] and ECC (Elliptic Curve Cryptography) [5] are the two most popular key cryptosystems. However, computing RSA is time-consuming due to the modular exponentiation involved. Similarly, point multiplication in ECC relies on

modular multiplication, which is computation-intensive thus resulting in a prolonged operation. The computational complexity of conventional security techniques such as SSL (Secure Sockets Layer) [6] and its successor, TLS (Transport Layer Security), make them not suitable for IoT devices. The SSL/TLS approach supported by CRL (Certificate Revocation List) creates scalability challenges for IoT applications. Homomorphic encryption [7] is very useful in protecting the privacy of users. However, the homomorphic encryption may be slow thus requiring special implementation techniques to speed up the execution. The ideal solution must provide data security and integrity while handling vast traffic and being attack-resistant. Furthermore, lightweight, scalable, transparent access control are to be associated with such a model. Blockchain is regarded as a promising solution to provide decentralized accountability and an immutable approach that can be used to overcome the aforementioned problems in heterogeneous scenarios [1]. It offers great security features while providing high transparency and enhancing efficiency. Meanwhile, it can also improve data traceability and eliminate third-party intervention at a lower cost.

Thanks to the development of edge computing platforms, data generated by the IoT devices can be transferred to the edge gateways for further process and analysis. At the same time, cloud-centric services are not suitable for the edge computing applications due to the limited network bandwidth, security, and data privacy. When applied to the edge computing systems, the blockchain provides a feasible solution to protect IoT data from being tampered [8]. It is a general distributed, decentralized, and peer-to-peer system that guarantees data integrity and consistency within existing industrial domains. Ethereum [9] is a common blockchain service showing intrinsic characteristics of distributed applications (dApps) over the blockchain network such as decentralization, anonymity, and auditability. However, common blockchain platforms (e.g., Ethereum) require tremendous computational power, making the integration of IoT nodes challenging.

The blockchain is an emerging technology playing a vital role in storing information and securing IoT systems and devices [10]. Although the blockchain is a promising application to solve IoT privacy and security challenges of current centralized systems, lots of IoT devices are constrained to perform complex operations due to their limited power of CPU, restricted data storage, and constrained battery resources. Furthermore, existing consensus algorithms in blockchain-based networks such as the Proof of Work (PoW) [11] cannot be implemented on devices with limited computing resources. The mining process described as taking decisions by all the nodes in peer-to-peer networks, requires considerable computational capabilities. Smart contracts present another promising application of blockchain technology that can distributively enforce various access control policies in IoT applications in the real-world scenarios. The data provenance plays a decisive role in the security and privacy of IoT systems. Additionally, the integrity of all generated data by IoT devices can be ensured by private blockchain technology.

In this paper, a blockchain-enabled edge computing approach is proposed and implemented for the IoT network with an open-source Hyperledger Fabric (HLF) blockchain platform. HLF is the best fit for this study because of its lower processing complexity (fewer number of transactions). Moreover, the transactions there can be performed in parallel while using various validators. Additionally, the processing is made more efficient by employing the fast RAFT [12] consensus algorithm. Finally, it provides a channel mechanism for private communication and private data exchange between members of a consortium. Moreover, all the HLF programs run in the docker [13] containers providing a sandbox environment that separates the application program from the physical resources and isolates the containers from each other to ensure the application's security. A layer-wise security architecture is designed according to the capabilities of different nodes and functionality to fit the scalable IoT applications. The infrastructure includes Base Stations (BS), Cluster Heads (CH), and IoT devices facilitating access control policies and management. Mutual authentication and authorization schemes for IoT devices are proposed and implemented with the aim to ensure the security of the interconnected devices in the scalable IoT platform. The local authentication is used for ordinary IoT devices connected

to CHs (edge IoT gateways), while the blockchain service provides the authentication of the IoT edge gateways i.e., the edge IoTs. The practical end-to-end lightweight HLF prototype for IoT applications is deployed on the embedded edge IoT hardware built upon the ARM64 CPU-based Raspberry Pi to validate the feasibility of the proposed design. HLF docker images are customized to fit with the IoT gateways. The Fabric client facilitates the request and query of transactions through invoking ChainCodes (CC) in IoT gateways. Off-chain data storage and blockchain distributed data storage are employed to support the architecture data traceability. HLF is implemented to act as a medium for multiple device interactions while exchanging information. Moreover, the blockchain maintains a global computation state. The distributed data storage is secure, and it has a large capacity. The data processing confidentiality and efficiency are guaranteed by implementing external off-chain computations. An HLF blockchain middle-ware module embedded in the IoT gateways ensures secure data transactions for the IoT distributed applications. The performance metrics such as throughput, latency, resource consumption and network use of the proposed model are evaluated using the edge IoT devices and x86-64 commodity virtual hardware.

The following distinct contributions are made in this work:

1. A novel architecture for the security and privacy of IoT edge computing using a permissioned blockchain is proposed. The proposed architecture considers 5G-enabled IoT technologies for node communications. The architecture is suitable for real-world IoT systems due to the developed ChainCodes that facilitate storage and retrieval of data in a tamper-proof blockchain system. Moreover, blockchain-based data traceability for 5G-enabled edge computing using the HLF is designed to provide auditability of the IoT metadata through a developed NodeJS client library.
2. The adaptability of the Hyperledger Fabric for ARM architecture of the edge IoT devices is improved by modifying official docker images from the source as there are no official or public images of HLF to support the 64-bit ARMv8 architecture.
3. A lightweight mutual authentication and authorization model is designed to facilitate a secure and privacy-preserving framework for IoT edge that protects the sensor nodes' sensitive data through a permissioned fabric platform. Furthermore, it provides trust for the IoT sensors, edge nodes, and base stations by the private blockchain. This is achieved by using the edge nodes to record the IoT data in an immutable and verifiable ledger to guarantee metadata traceability and auditability.
4. Performance characteristics of the proposed architecture blockchain in terms of throughput, transaction latency, computational resources, network use, and communication costs are experimentally evaluated in two network setups.

The rest of the paper is organized as follows. In Section 2, a review of the related works is presented. Section 3 presents the main characteristics of blockchain technology. Section 4 describes the proposed HLF model implementation and elaborates on the details of the system design. In Section 5 the profiling and analysis are presented, including results from real-life IoT applications. Finally, Section 6 presents the conclusion and directions for future work.

2. Related Work

2.1. IoT Overview

In general terms, IoT is a collection of physical devices, computers, servers, and small objects embedded within a network system [14]. Some of the most prominent IoT application areas are smart homes [15] and smart cities [16], vehicular systems [17], and smart healthcare networks [18]. All these systems are highly distributed. The evolution from the conventional cloud-centric architecture has been accelerated by the emergence of the edge computing technologies [19,20]. A unified standard classification is defined to ensure the consistency of the development and structures of IoT. It includes four layers: service layer, platform layer, network layer, and device layer [21]. A comprehensive review of security attacks towards Wireless Sensor Networks (WSNs) and IoT is presented

in [22]. The study also provides the techniques for prevention, detection, and mitigation of those attacks.

IoT systems normally include many interconnected IoT devices generating a massive amount of data. Meanwhile, IoT devices normally have limited capabilities in terms of the CPU processing performance, memory capacity, and battery energy volume. Therefore, they can be characterized as having restricted ability to resist various cyber-attacks. This leads to issues associated with insufficient security and potential compromising of privacy. New technologies have been developed to address the IoT's decentralization challenges with the blockchain being among the most promising of them.

2.2. IoT Blockchain

Most IoT applications are prone to problems such as system failure and data leakage. Blockchain technology can mitigate these problems by providing better security and scalability for IoT applications. However, there are many challenges associated with the actual implementation of the approach. They are associated with tasks distribution between IoT devices as well as with the limited capabilities of the IoT devices such as computational performance, memory capacity, power resources. Numerous research works on blockchain technology focus on coping with these challenges to adopt blockchain in IoT [23–25].

Many distributed and decentralized IoT systems have adopted blockchain technology to provide trust [26], security [27], data management [28], fault-tolerance [29], as well as peer-to-peer and interoperable transactions [30]. The application scope of blockchain platforms can be divided into three main types depending on the way they manage user credentials: (i) public or permissionless blockchain, (ii) private or permissioned blockchain, and (iii) consortium blockchain. Blockchains that anonymous nodes can join, read data, and participate in transactions with equivalent status are public blockchains. In contrast, private or consortium blockchains are based on permissions and different types of nodes. Some nodes need to be authenticated to perform specific actions [31].

Scalability is the major challenge in the integration of blockchain and IoT systems. Many research works have addressed the scalability issues within Bitcoin's architecture [32]. Smart contracts are promising solutions to facilitate the integration of distributed IoT systems and blockchain technology. However, their performance and scalability are directly linked to overall blockchain system performance [33]. Multiple IoT applications recently adopted blockchain for digital payment, smart contract services [34], and data storage [35]. Nonetheless, continuous developments have shown that new technologies can bring significantly higher scalability and degree of performance to next-generation blockchain systems.

The layer-based IoT blockchain frameworks are proposed in the literature to cope with the scalability challenges in IoT systems while providing higher performance and security. The layer-wised structure is a promising solution to smart cities' security by integrating smart devices and blockchain technology [36]. A hybrid-network architecture is seen to leverage the strength of emerging Software Defined Network (SDN) and blockchain technologies in a multi-layer platform [37]. Layer-based blockchain can potentially address the IoT systems' challenges such as response time and resource consumption [38]. This approach can further facilitate the integration of blockchain technology in IoT systems by tackling the complexity of blockchain implementation in the layer-based model [39].

Security challenges associated with the cyber-physical systems (CPSs) of smart cities are reviewed in [40] and adoption of distributed anomaly detection systems by CPSs of smart cities is proposed. A permissioned private blockchain-based solution in the context of the Industrial IoT (IIoT) is proposed in [41] to secure the encrypted image. This approach stores the cryptographic pixel values of an image on the blockchain, ensuring the image data privacy and security. The state of the art in industrial automation is presented in [42] to provide a better understanding of the enabling technologies, potential advantages and challenges of Industry 4.0 and IIoT. Also, it covers the cyber-security related needs of IIoT users and services.

2.3. Blockchain for Mobile Edge Computing

Several pieces of research have considered the integration of blockchain technology and edge computing layer over the past few years. Multiple works have focused on enabling secure and efficient distributed edge computing [43,44]. Such integration targets security enhancement. It also uses blockchain technology to develop access control policies for various applications at the edge [45–47]. Other works [48,49] investigated the edge resource management by implementing the blockchain. Distributed robotic system automation was also considered [50]. The integration of blockchain significantly benefits the security of edge computing [51]. Permission blockchain and Distributed Ledger Technology (DLT) embedded with identity management bring benefits to address many challenges by adding a resilience layer while network traffic integrity is guaranteed against malicious diversion and traffic manipulation. Network resource manipulation and fraudulent use of shared resources are avoidable through the blockchain-enabled resource management. Moreover, the blockchain provides a higher degree of security for the automotive sector [48] and the healthcare sector at the edge [52]. Blockchain is applied to provide a decentralized authentication model in edge and IoT environments [53]. The blockchain application is further explored to enhance the privacy, integrity, and authentication between IoT, mobile edge computing, and cloud in telehealth systems connected with 5G and IoT [54]. An HLF-based blockchain architecture is proposed in [55] for healthcare monitoring applications. The authors in [56] highlighted the importance and benefits of fog computing for IoT networks. The study also provides a comprehensive investigation of hardware security to fog devices through an enriched literature review. A model based on HLF blockchain is proposed in [57] as a service to answer IoT systems' specific requirements, including low hardware, storage, and networking capabilities.

2.4. Blockchain for Data Sharing and Traceability

Digital signatures and Message Authentication Code (MAC) are two standard methods to identify data lineage and origin. However, these cryptographic techniques are not able to provide comprehensive data provenance [58]. Furthermore, the key management in a heterogeneous IoT network with data sourced from different nodes is complicated. Although logging-based methods can facilitate data transmission and system events monitoring, they cannot efficiently track data in distributed IoT systems [59]. Blockchain technology has been widely considered for data provenance within a distributed system such as IoT. Data operations are embedded in the blockchain transactions to provide the data provenance [60]. ProvChain [61] is a distributed and decentralized blockchain-based data provenance architecture to provide verifiability and data integrity in cloud environment. A blockchain network records the data operations as the provenance of data in the blockchain transactions while the system stores the data record in a local ledger. Smart contracts can automate the blockchain-enabled provenance systems without the off-chain verification [62]. A function for tracing the data deviation is designed into smart contracts with built-in access rules to protect data privacy in a distributed ledger [63]. SmartProvenance [64] is the blockchain-based distributed data provenance system that facilitates the verification of provenance records and provides trustworthy data and provenance collection using smart contracts and the Open Provenance Model (OPM). The blockchain is proposed to ensure secure and trustworthy industrial operations [65]. The complexity of blockchain implementation causes various limitations in deploying the aforementioned provenance techniques in IoT systems. Existing works on data provenance are computationally complex and pose a hardware cost. Therefore, these methods are not feasible for resource-constrained IoT systems with limited CPU performance, memory size, and power capacity.

Despite the benefits that blockchain brings to IoT applications, there are resource constraints and scalability challenges associated with the integration [2,66,67]. Generally, the blockchain demands substantial computational power for the mining process in Proof of Work (PoW), low latency, and high bandwidth. IoT devices with low processing power

are not capable of performing the blockchain mining process. The data encryption process is frequently happening in blockchain systems. The computationally intensive process of blockchain drains the low power capacity of IoT devices. The size of the blockchain ledger increases continuously while the storage capacity of most IoT devices is low. Storing a copy of the full blockchain ledger for IoT devices is not feasible as it requires a large memory capacity. With Bitcoin, the blockchain storage size rests at over 200 GByte while for Ethereum it is around 1.5 TByte. New block generation and agreement reaching in the blockchain require the nodes to exchange information through the consensus process frequently. The consensus process and information exchange need high bandwidth and low latency. However, the bandwidth of IoT devices is normally strictly limited.

One common concern about the blockchain system is associated with the need for achieving high scalability in a blockchain network [68]. The problem with such a large blockchain size is centralization risk. Most IoT systems have a very high number of interconnected devices. In addition, IoT networks frequently change to suit different applications by adding or removing IoT devices. Therefore, a solution is required to address the IoT system scalability challenges. Moreover, the limitations in the processing power and storage capacity of IoT devices in the blockchain network are also to be resolved. Addressing these challenges is the main focus of this paper.

3. Blockchain Overview

Satoshi Nakamoto, first implemented a decentralized digital currency in 2009 [69]. The blockchain can be described as a distributed ledger consisting of immutable and verifiable transactions. All network participants share a replica of the ledger in the network. Integrity, immutability, transparency, non-repudiation and equal rights are the main properties of the blockchain systems.

Bitcoin [70] is known as the most popular blockchain platform. PoW is used in Bitcoin to perform ownership management and tracking coins owner via implementing public-key cryptography with a consensus algorithm. The consensus algorithm is executed when a new block is introduced to the previous block to guarantee the reliability and validity of all transactions. The nodes will reach a consensus when 51% of the nodes are truthful.

IOTA [71] is a distributed ledger designed for IoT to facilitate the value and data exchange. A machine-to-machine communication is facilitated by the Tangle protocol capable of forming micro-payment systems. Additionally, it establishes IOTA network, which is a set of Tangle graphs. This set constitutes the ledger to store transactions submitted by the network nodes. The process of block validation leads to making a decision and adding a new block to the blockchain.

3.1. Consensus Algorithm

Li et al. [72] reviewed the most common consensus algorithms in the existing blockchain systems. These consensus mechanisms are PoW, Proof of Stake (PoS), Practical Byzantine Fault Tolerance (PBFT), Delegated Proof of Stake (DPoS), Proof of Authority (PoA), Proof of Elapsed Time (PoET), and Proof of Bandwidth (PoB).

PoW is the widest deployed consensus algorithm [73] that was first introduced by Bitcoin. The nodes use computational power to compete in finding the nonce value. This process is called mining. The difficulty level for PoW is adjustable when the number of participants increases to manage the block's average processing time. Higher difficulty results in a lower number of blocks. No user should take more than 50% of the processing power to avoid controlling the system by just one user.

PoS [74] was introduced to address the vast energy consumption issues associated with the competing process in PoW. No competition is employed in the PoS algorithm. The network selects a node as a validator (so-called a transaction validator node). The node is chosen in advance to be a part of the Proof of Stake and attend a similar process of difficulty adjustment as PoW. If the validator does not validate the transaction, the network

sets the next node as a validator, and the process continues until any node validates the transaction. PoS deploys CASPER protocol to perform the consensus process.

PoA [74] algorithm is based on a chosen set of trusted nodes (known as Authorities). This consensus algorithm is a Byzantine Fault Tolerant (BFT) variation. The chain becomes a part of the permanent records when most authority nodes (for example at least $N/2 + 1$) signs off the chain. This procedure facilitates the creation of a permissioned chain and is associated with a lighter exchange of messages.

Hyperledger [75], introduced in 2016 by the Linux Foundation, is the most successful and the most popular permissioned blockchain in the industrial and IoT domains. The designed permissioned blockchains for enterprise ecosystems deploy the RAFT Consensus Protocol [12], which is a better fit because it is more straightforward and less resource consuming. Figure 1 shows the process of the RAFT consensus protocol and block creation considered in this study.

Kafka [76] and RAFT are the same types of consensus that use Crash Fault Tolerant (CFT) for ordering service implementation. They can tolerate up to $N/2$ system failures. RAFT follows a “leader and follower” approach. There a leader node is dynamically elected among the ordering nodes in a channel (this collection of nodes is known as the “consenter set”), and the followers replicate its decisions. However, RAFT’s ordering service deployment is easier and more manageable than Kafka-based ordering services from the configuration to the process’s speed. Additionally, the RAFT configuration originates directly from the orderer (unlike the Kafka case, which cannot be configured directly from orderer services and must create a Zookeeper cluster to enable the state machine replication process). The comprehensive design facilitates different organizations to contribute nodes to a more distributed ordering service.

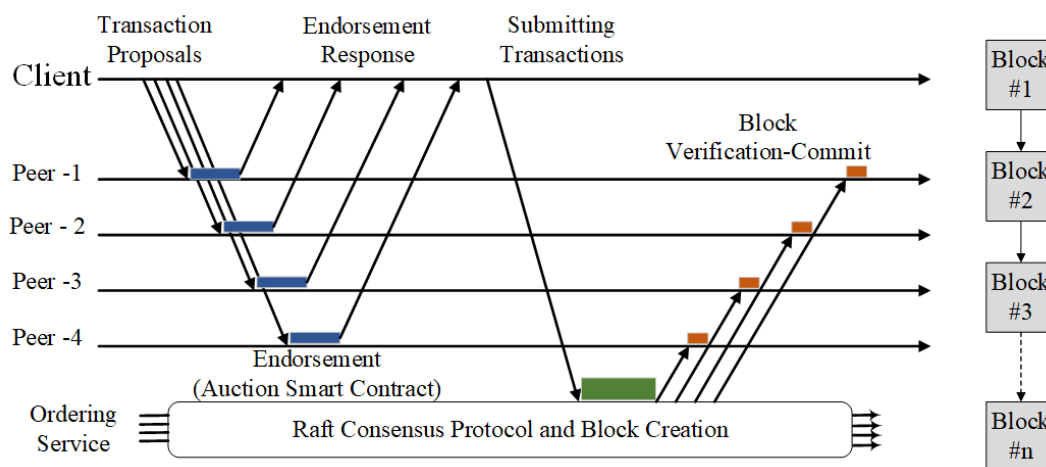


Figure 1. Overview of the RAFT consensus protocol and block creation.

The process is initiated by sending the transaction proposals to the blockchain peers. A transaction proposal consists of various values, IoT metadata as well as other blockchain-related contents. The client application is responsible for starting the process and then transaction broadcasting to each blockchain member organizations’ peers. Once the peers receive the transactions, they activate the endorsement process by executing the Chain-Code implementing authentication and authorization mechanism. The transaction is then endorsed and returned as the signed transaction. When all peers have endorsed the transaction based on the endorsement policy, the next step includes sending the transaction to the ordering service when the consensus is reached (i.e., RAFT in our case). The last step is encompassing the creation of the final block and committing it to the ledger.

3.2. Smart Contracts

Smart contracts are executable distributed programs to facilitate, execute, and enforce the terms of an agreement on a decentralized consensus tamper-proof and typically self-enforcing through automated execution [77]. The smart contracts are simply executable scripts that are filed on the blockchain with a specific address.

Smart contracts are triggered by transactions to execute and perform operations based on recorded instructions. They are installed and instantiated on blockchain participants. HLF is programmable by a construct called ChainCode (CC). Conceptually, CC is the same as the smart contract on other distributed ledger technologies. CC sits next to the ledger. Participants of the network can execute CC in the context of a transaction that is recorded in the ledger. Automation of business processes through CC leads to higher efficiency, transparency, and greater trust among the participants. Smart contracts allow decision automation thus making them suitable for IoT applications.

4. Hyperledger Fabric IoT System Model

4.1. Overall Design

The network model proposed in this work is based on blockchain technology as an individual application integrated with edge computing to provide security, identity management, and authentication. This study builds on the model introduced in our previous work [78] using a multi-layer platform approach and the Lightweight Hyperledger Blockchain technology along with smart contracts to enhance the performance of the blockchain-IoT combination. The whole network is divided into several layers and sub-networks. The devices in each layer have different computational capabilities and energy storage capacity. As a result, different security approaches are proposed for individual layers based on the blockchain. However, the blockchain implementation is modified to suit the devices of each particular layer. These layers are Base Station (BS) nodes, Cluster Head (CH) nodes (edge layer), and IoT devices. In the current work, we propose an additional layer—Off-Chain Storage servers—to enhance the data storage of IoT devices. Moreover, it facilitates the system performance improvement as the increase in the shared ledger size causes system performance degradation. The Hyperledger Blockchain platform is considered to be a potential solution to cope with scalability challenges while distributed programs are defined to facilitate various tasks and transactions [79]. However, the blockchain implemented in the embedded edge gateways provides reliable connectivity considering sufficient power and computational resources requirements. Figure 2 shows the conceptual framework of the proposed IoT Blockchain platform. The presented model encompasses interconnected IoT devices, Edge IoT nodes (CHs), client application nodes, external data storage, and IoT servers orchestrated in the peer-to-peer blockchain-based network to form a multi-layer blockchain model.

4.2. Multi-Layer IoT Blockchain Network

4.2.1. Layer-1

A cluster of IoT devices is collected under each CH, a service agent for that cluster. This layer is the external service interface, in which IoT devices collect sensing data, perform local computing, and send results for storage and further analysis. CH nodes register the identity of each connected IoT device by implementing a smart contract. Each IoT device has a unique address within the IoT system. Each IoT node exists only in one cluster. The nodes in this layer have limited power, computational performance, and storage resources.

4.2.2. Layer-2

Cluster heads at Layer-2 are responsible for data routing, security management (such as local authentication and authorization procedures), and network management. Beyond the aforementioned responsibilities, the IoT blockchain service is running in this layer to provide blockchain technology services and form a distributed system. The IoT devices' identity management, communications, and consensus processes are run in this layer

within the peer-to-peer blockchain network. The blockchain also handles the shared distributed ledger across all participants. Furthermore, this layer handles consensus algorithms and smart contract services to form data consistency and traceability.

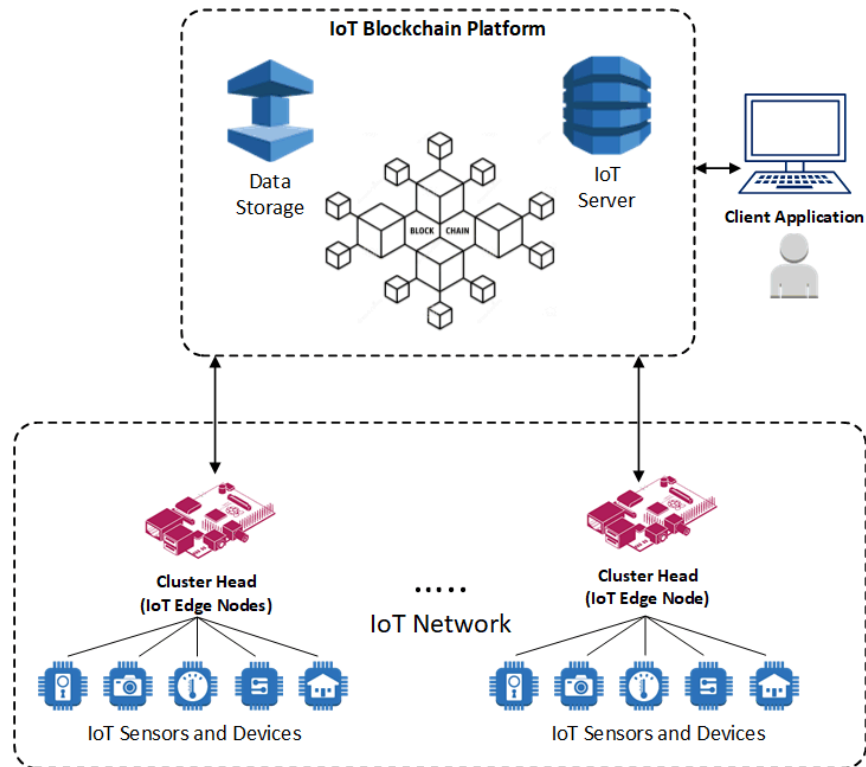


Figure 2. Conceptual framework of the integrated IoT blockchain platform.

A client application node across the network can have granted access to invoke various blockchain behaviors. Various ledger modifications are enabled by running smart contracts installed and instantiated in all peer nodes or selected peer nodes. The CH nodes running local authentication and authorization mechanism are directly connected to BS nodes. ChainCodes provide deployment, query, and invocation services. The API rest server can act as an interface by the client application with modifying the network-related operations and behaviors. Furthermore, the application client performs transaction submission to the blockchain. Therefore, various services can be defined within the blockchain network, including user enrollment and authentication and IoT device registrations. The IoT device authentication and authorization need to be carried on before transaction submission. The local authentication and authorization process manages this procedure. Consequently, a registered participant can sign a transaction using its private keys.

Data queries are enabled through CC, which is an executable logic hosted by peer nodes. Additionally, it facilitates appending data from data stored in the ledger. CC and related functionalities are mirrored across all peer nodes. CC deployment can be done to a specific number of peers to address the scalability issues. Therefore, parallel execution can be supported, which is resulted in an overall increase in system performance. The client application performs several operations, including storing the data checksum, data pointers, and data ownership in the blockchain. The actual data is stored in an external data storage, which is off-chain.

4.2.3. Layer-3

In general, this layer is consistent with the current centralized cellular network encompassing Base Station nodes while the cloud server manages the process requests and

data generated from various devices. Powerful devices in this layer can choose to use a non-symmetric encryption algorithm for data transmission. Layer-3 provides connectivity and wide area networking capabilities for the edge nodes. Network in the Layer-3 is decentralized, and BS units are distributed. The nodes trust the BSs in the system while they can access public networks.

4.2.4. Layer-4

This layer is designed for storing sensed data by the IoT devices as well as enabling big data analytic applications for further analysis. It is generally done off-chain. It stores the actual data, while the blockchain ledger data includes data checksum, pointers, and data ownership. The blockchain world state is stored in a database such as LevelDB or CouchDB. The stored data in be queried and traced by a file ID in the blockchain. This method provides data provenance and data consistency between the edge nodes.

4.3. Local Authentication and Authorization of IoT Devices in Layer-1

Identity of IoT devices is registered and stored in the shared ledger. Each IoT device can join only one cluster. The registration request is sent to CH. It includes the required information such as IoT node ID, cluster identity, and timestamp. CH runs the smart contract in the local blockchain to perform the IoT device registration. The mutual authentication model is designed to provide the security of IoT devices with limited resources. The role of CH is to register the IoT devices as well as locally authenticate and authorize IoT entities. It also interacts with other cluster heads to form a secure communication between entities through the implemented blockchain network.

The entire process is orchestrated in a smart contract to form an Authentication and Authorization ChainCode. The CC is installed and instantiated by the blockchain peer to perform the IoT blockchain local authentication procedure. This process is illustrated in Figure 3.

Authentication of the IoT devices consists of a few steps: the discovery of devices, key exchange, authentication, and data encryption. These procedures consider two network entities: the CLIENT (IoT sensor nodes) and SERVER (an edge computing gateway or intermediary node). It is noteworthy that the authentication of the IoT devices implements the exchange of keys using Diffie-Hellman Ephemeral (DHE) for the collection of session keys or secret keys. The following six steps describe the local Authentication of IoT devices.

- Step 1 The first step starts with the CLIENT sending a package to the SERVER to establish a "connection". For visualization purposes, this package contains the "HELLO CLIENT" character string.
- Step 2 The answer from the SERVER to the CLIENT with the "HELLO SERVER" string. With that, the connection is established. For better performance, it is suggested to use chain bits for establishing the connection.
- Step 3 The CLIENT generates a pair of asymmetric keys consisting of the public key ($K^{C_{pub}}$) and the private key ($K^{C_{priv}}$). For the key generation, an Initialization Vector (IV) is required with random values guaranteeing the distinction between the generated keys. Then, a packet is sent to the SERVER containing: the CLIENT's public key ($K^{C_{pub}}$); a value such as "challenge-response" generated by the CLIENT; a character string F_{dr} defining the "challenge-response".
- Step 4 The SERVER generating a pair of asymmetric keys: the public key ($K^{S_{pub}}$) and the private key ($K^{S_{priv}}$). In sequence, the SERVER receives the CLIENT's package and responds with another package containing its public key ($K^{S_{pub}}$) and the response to the "challenge-response" calculated from the F_{dr} function. The F_{dr} is a mathematically predefined function that can be sum, subtraction, or multiplication applied to the value of IV received.
- Step 5 The CLIENT calculates Diffie-Hellman values. A new package consisting of the obtained DH value (DH^C), the parameters g and p used in the calculation, a new value of IV (iv^C), and the value of IV obtained from the SERVER applied

to the function $F_{dr}(F(iv^S))$ will be sent to the SERVER. Moreover, a summary function (Hash) for all these data and its result is encrypted with the CLIENT key ($K^{C_{priv}}$). It is then included in the package. The whole package is then encrypted with the public key of the SERVER ($K^{S_{pub}}$). The encryption guarantees the data confidentiality.

Step 6 The SERVER performs the calculation of the Diffie-Hellman values from the information coming from the CLIENT. The SERVER then performs the same actions as done by the CLIENT in step 5. It sends the resulting package to the CLIENT at the end of the process. With that, both the parties have a common key: the session key (DH_K).

After exchanging the keys, the client and the server can exchange encrypted data with a symmetric key (DH_K), which can last for the session.

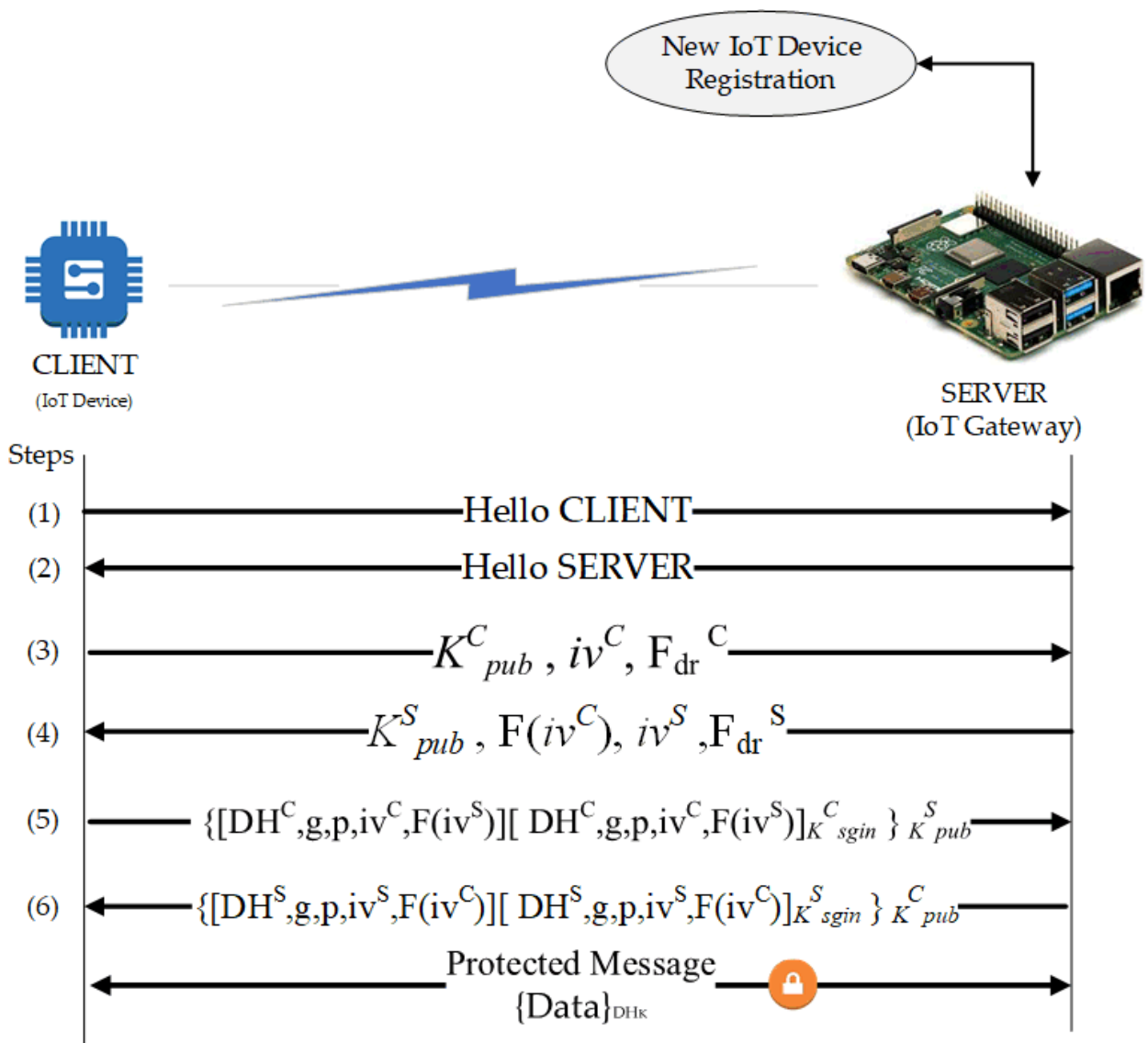


Figure 3. Local authentication flow.

4.4. Secured IoT Blockchain for Edge Computing Nodes in Layer-2

The proposed model as illustrated in Figure 4 encompasses the blockchain as part of the individual applications of the edge computing layer to provide security, data traceability, identity management, and privacy. A blockchain orchestrates a decentralized database that allows applications to trace the history of appended transactions to a shared ledger.

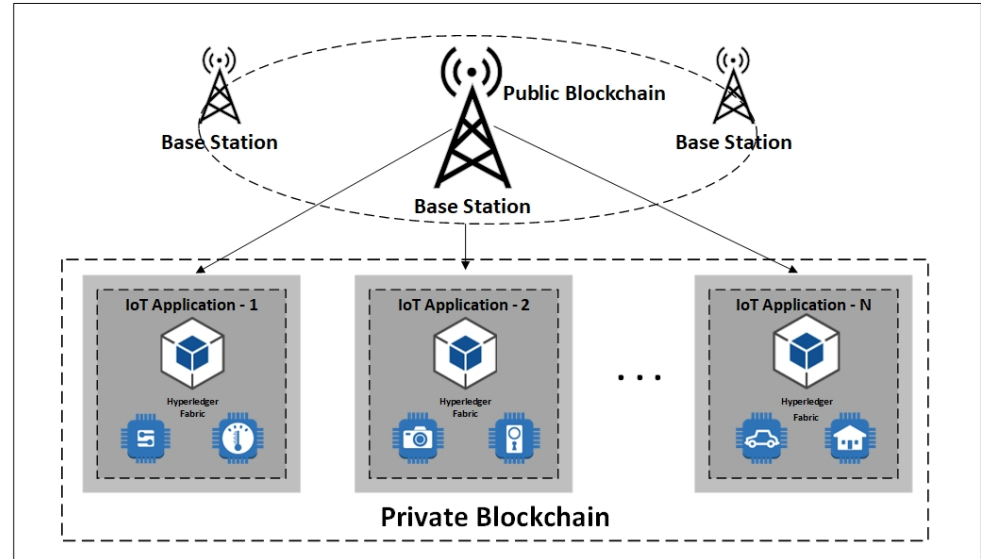


Figure 4. Blockchain-based edge services.

The main component of the proposed model in this layer is HLF blockchain framework running on the docker containers and integrated client library. The storage component is designed in a separate layer to store the actual collected data off-chain. The client library initiates the operations and communicates with other elements. The seamless provenance of metadata storage is enabled while the data checksums are recorded in a tamper-proof blockchain ledger.

4.4.1. Nodes in IoT Edge Hyperledger

There are three distinct types of nodes in HLF: Peer, Orderer, and Client. The client is the node that applications use for initiating the transactions. Client nodes perform issuing transactions to the peers, collecting proposal responses, and sending blocks for ordering. Peers are the nodes that interact with the blockchain ledger and endorse transactions through running CC. Peers are the nodes that keep the ledger in-sync across the network.

Orderers are the communication backbone for the blockchain network. They are responsible for the distribution of transactions. Furthermore, the orderer nodes are accountable for the validity and verification of responses. Moreover, the order nodes form new blocks from grouped transactions when the consensus is achieved.

Peers nodes update the ledger after the blocks are generated. Members can participate in multiple Hyperledger Blockchain networks. Transactions in each network are isolated, and this is made possible by way of what is referred to as a channel. Peers connect with the channels that can receive all the transactions that are getting broadcasted on those channels. The transaction flow is presented in Figure 5.

There are two particular types of peer nodes: Anchor and Endorser. These peers need to be configured with appropriate cryptographic materials, such as certificates. Peers in the member's organization receive transaction invocation requests from the clients within the organization. Once transactions are created in the network and new blocks get generated, they are sent out to the peers by the ordering service. Peers receiving these blocks need to validate and update the ledger. This is managed on the peer node. Inherently, this

architectural approach is highly scalable as there is no need for a centralized effort to scale the network or scale the infrastructure.

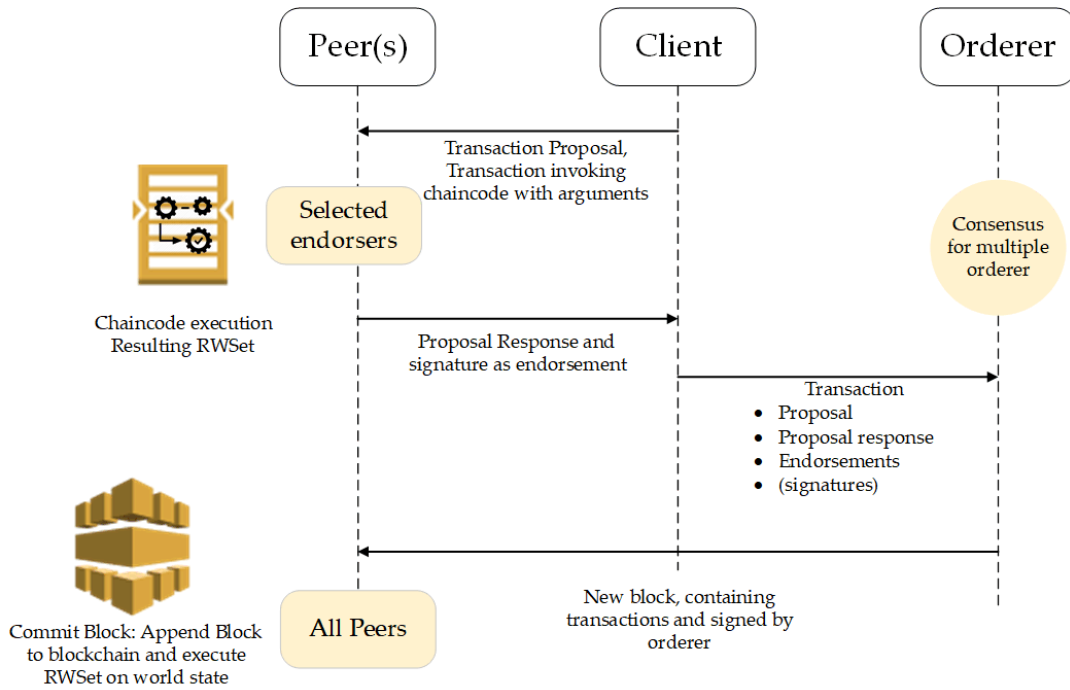


Figure 5. Proposed HLF network transaction flow.

Each member organization can look at their needs and set up the needed infrastructure based on their requirements. Member organizations can have multiple peers. However, not all peers receive the block information from the Orderer—only the relevant anchor peer receives them. To avoid a single point of failure, an organization can create a cluster of the anchor peers. The anchor peers are set up and defined as part of the channel configuration. The anchor peers are by default discoverable. Peers may be marked as the endorsers or take up the endorser’s role (known as the endorsing peers). A client sends the invocation requests to the endorsing peer. On receiving the request for the invocation, the endorsing peer validates the transaction. For example, it checks whether the end-user has used a valid certificate. If the validation checks out fine, then it simulates CC.

A set of IoT edge nodes is configured to run HLF processes through Docker. Network participants run the peer process and maintain the blockchain ledger by receiving various transaction proposals. The peer process is the main component of the HLF network while hosting CC and the ledger. Network’s efficiency can be enhanced by increasing the number of running peers. However, one peer node per organization is normally sufficient. The ordering service handles blocks of ordering tasks and validates the proposed blocks by peers with a deterministic consensus algorithm. The proposed model can be enhanced through the multiple Orderers approach for fault tolerance using RAFT [12] or Kafka [76] methods.

4.4.2. ChainCode in IoT Edge

Each peer participating in HLF networks keeps a copy of the ledger. The ledger consists of the blockchain and world state. Each block contains packed transactions, ordered and broadcasted by ordering service based on peer proposals. The world state database keeps the latest state in key or value form. CC is a program (smart contract) that is written to read and update the ledger state. Its operation is the process of deploying a well-developed CC onto a fabric network (channel) such that client applications can invoke CC functions. CC deployment (lifecycle ChainCode) includes: (i) install CC to

selected peers, (ii) instantiate CC to a channel and specify an endorsement policy as well as initial function arguments when needed. After the deployment, invoking the ChainCode functions is accessible.

One enhancement in HLF is that the CC governance becomes decentralized. The CC package does not need to be identical across channel members. This means that organizations can extend the CC to include additional validation. Lifecycle CC includes steps in which member organizations can explicitly participate in the ChainCode deployment. The current design implements ChainCodes to manage IoT devices' identity connected to edge gateways, store, and retrieve data from the blockchain ledger. The checksum of all collected data objects is stored in the ledger. Moreover, the location of data and the data ownership (authenticated ID) are considered to be recorded. This approach enables the system to track the data location and verify the integrity of the data. Using the certificate for invoking the transaction, the system records who and when edited or stored an item. The data lineage traceability is enabled by recording the references of the items used to generate it. The client library facilitates the ledger's interaction to perform various functions, storing and querying the provenance information. The proposed model implements multiple endorsing nodes to ensure running the CC in a lightweight environment.

Part of the ChainCode design includes running the authentication and authorization processes for security, privacy, and identity management. Furthermore, CC tracks the owner of performed operations on data. The Client Identity (CID) CC library [58] introduced in HLF v1.1 is used in this research to save a userID issued by the Certificate Authority (CA).

4.4.3. Certificate Authority

Membership Services Provider (MSP) is an abstract component of the HLF system that provides clients' and peers' credentials to participate in the Hyperledger Fabric network. The default MSP implementation is based on the Public-Key Infrastructure (PKI). There are two primary services provided by MSP: authentication and authorization. In PKI-based implementations, there is a need to manage the identity by way of certificates. The certificates are issued, validated, and revoked by the CA.

Each component needs to be authenticated and identified before accessing the fabric network. In a typical case, a user is issued with a digital certificate that includes proper information associated with that user. Fabric CA is the Certificate Authority developed by HLF serving a CA role. Once the Fabric CA is up and running, it can issue new certificates with the request's specific requirement. Fabric CA can be accessed using Fabric-CA Client or Fabric SDK, both from HLF. Digital Certificate is issued by CA that is trusted by the fabric network. The user's operation is then accepted and processed by the fabric network. The digital certificate can be issued when crypto material is generated with Cryptogen and Configtxgen binaries, or more commonly, generated through registration and enrollment on CA. The current design implements Hyperledger's CA docker image, customized to provide persistent certificate database storage. The fabric-CA implementation has two parts: fabric-CA server and fabric-CA client. Members are issued a root certificate that they can use for issuing their own identities within their organizations. Thus, the Hyperledger fabric network can have one or more certificate authorities to manage the certificates.

4.4.4. Ledger Implementation

HLF is a distributed ledger technology. All peers in the network have a copy replica of the ledger. The ledger has two parts: a transaction log and state database. The transaction log keeps track of all the transactions invoked against the assets. The state data are a representation of the current state of the asset at any point in time. The transaction log is implemented using the LevelDB, that is a lightweight library for building a key-value data store. It is embedded and used as part of the fabric peer implementation. Unfortunately, the LevelDB does not provide a capability for creating and executing complex queries. However, one can replace the state database (which is implemented in the LevelDB) with

CouchDB that supports the creation of complex queries. Therefore, the state database is pluggable at the peer level. The transaction log is immutable. At the same time, the state data are not immutable. The creation of records in the transaction log is possible, as well as the retrieving of existing transaction records from the transaction log. However, it is not possible to update a current transaction record that is present in the log while it is possible to delete any of the transactions added to the log. From the state data perspective, create, retrieve, update, and delete operations can be carried out on the state data for an asset. The ledger implementation in the proposed model is shown in Figure 6.

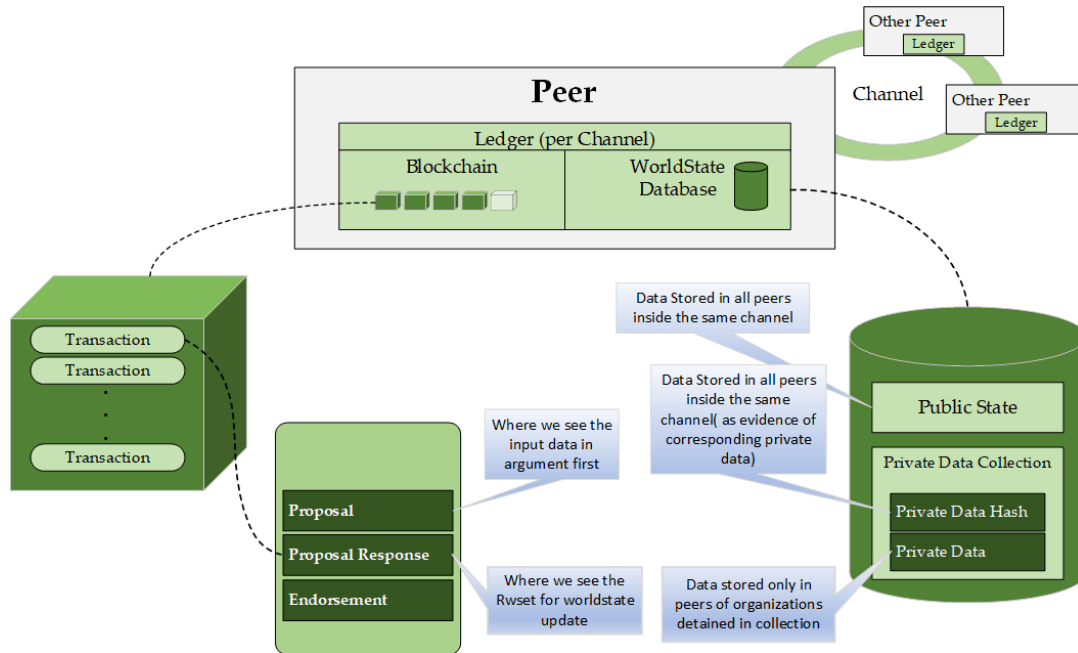


Figure 6. Ledger implementation flow.

4.5. Base Station Nodes with High Computational Power in Layer-3

BS node’s main functionality includes several tasks such as nodes management under each base station, collecting and aggregating the received data from sensing nodes, processing, analyzing, and storing the received data. As an organization manager, BS is trusted by other network participants. CH nodes (edge IoT devices) first need to be initialized and authenticated by BS before joining the network. Base stations can connect to public networks or clouds as they have robust computing and storage resources. In a public blockchain, nodes build trust in a decentralized manner through a consensus algorithm. Running public blockchain within resource constraint IoT nodes is not feasible due to the lack of needed massive capacity and time for the frequent authentication process. The unified authentication scheme is presented in Layer-2 to facilitate the joining process for nodes in a local private blockchain framework. The current hybrid design proposes a public blockchain for base stations in Layer-3 of the network model. Cluster head nodes are registered and authenticated with BS nodes through implementing the smart contracts. The node’s identity information is recorded in a public blockchain ledger.

4.6. Layer-4 Off-Chain Storage

Implementation of Distributed Ledgers Technologies (DLT) with blockchain is limited in terms of the amount of data stored in their ledger. The size of the shared ledger is growing incessantly, causing the system performance degradation. The solution to this challenge in the proposed design includes the use of off-chain storage. The blockchain in Layer-2 stores only the metadata’s provenance while the actual generated IoT data

are stored in non-blockchain-based storage. This amount is a small fraction of the total generated data by the IoT devices. The data checksums are computed, stored, and verified with the blockchain records to ensure the integrity and immutability of the stored IoT data. The CC functions and the ledger functionality are independent of the off-chain storage choice. However, quick adding multiple storage (or other) resources is possible based on system requirements.

The current design implements SSHFS [80] as shared storage, while Raspberry Pi are employed as CHs (edge IoT devices). Thus, the choice of external shared storage needs to be aligned with the ARM64 architecture of the Raspberry Pi system. The SSHFS is a FUSE-based user-space client. It allows mounting a remote filesystem using SFTP as an underlying protocol through SSH. Most SSH servers enable and support the SFTP protocol and provide access by default. Performance evaluation of distributed storage services in the community network shows that SSHFS is comparable with other network file systems [81]. Moreover, the system enhancement is achievable with a more resilient distributed file system such as Open AFS [82] or cloud-based services such as Amazon EFS [83].

5. Performance Evaluation

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

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

The detailed desperation of Hyperledger performance metrics is documented in the Hyperledger Performance and Scale Working Group white paper [84].

5.1. Experimental Setup and Implementation

The experimental setup consists of two different environments of the same network. The first network was set up and run on virtual desktop nodes. The other system included Raspberry Pi (RPi) devices acting as IoT edge nodes. These RPi were chosen as IoT cluster heads and were connected to several small IoT sensors.

The virtual desktop setup had five virtual machines running on VMware virtual platform environment: 5 Intel(R) Xenon(R) Gold 5220 CPU@202GHz 2C2T. All nodes run Ubuntu 18.04. The official Hyperledger Fabric (version 1.4) framework was deployed as an underlying blockchain application. HLF is a permissioned open-source blockchain architecture designed for the enterprise ecosystem. Figure 7 shows the system under test high-level architecture.

The same network setup was implemented on four RPi Broadcom BCM2711 Quad-core Cortex-A72 (ARM v8) 64-bit SoC@1.5GHz devices, and one virtual desktop used as CA server. RPi nodes run the Debian 64-bit OS and nodes interconnected in a peer-to-peer network thus forming a distributed and decentralized network. Because the official HLF framework cannot be run on Raspberry Pi devices, the docker images for ARM64 architecture has been modified to support running the HLF on the RPi nodes.

Measurements on both the networks were taken enabling a comparison between the architectures. The two system setups encompass devices with dissimilar capabilities. That helped to better understand the system performance and devices' capabilities in different scenarios of running the HLF platform. Docker containers consisted of blockchain components that were orchestrated by the Docker Swarm and deployed across the network of nodes. A client was considered to be load-generating one that could submit transactions into the system, and invoke transactions and system behaviors from it.

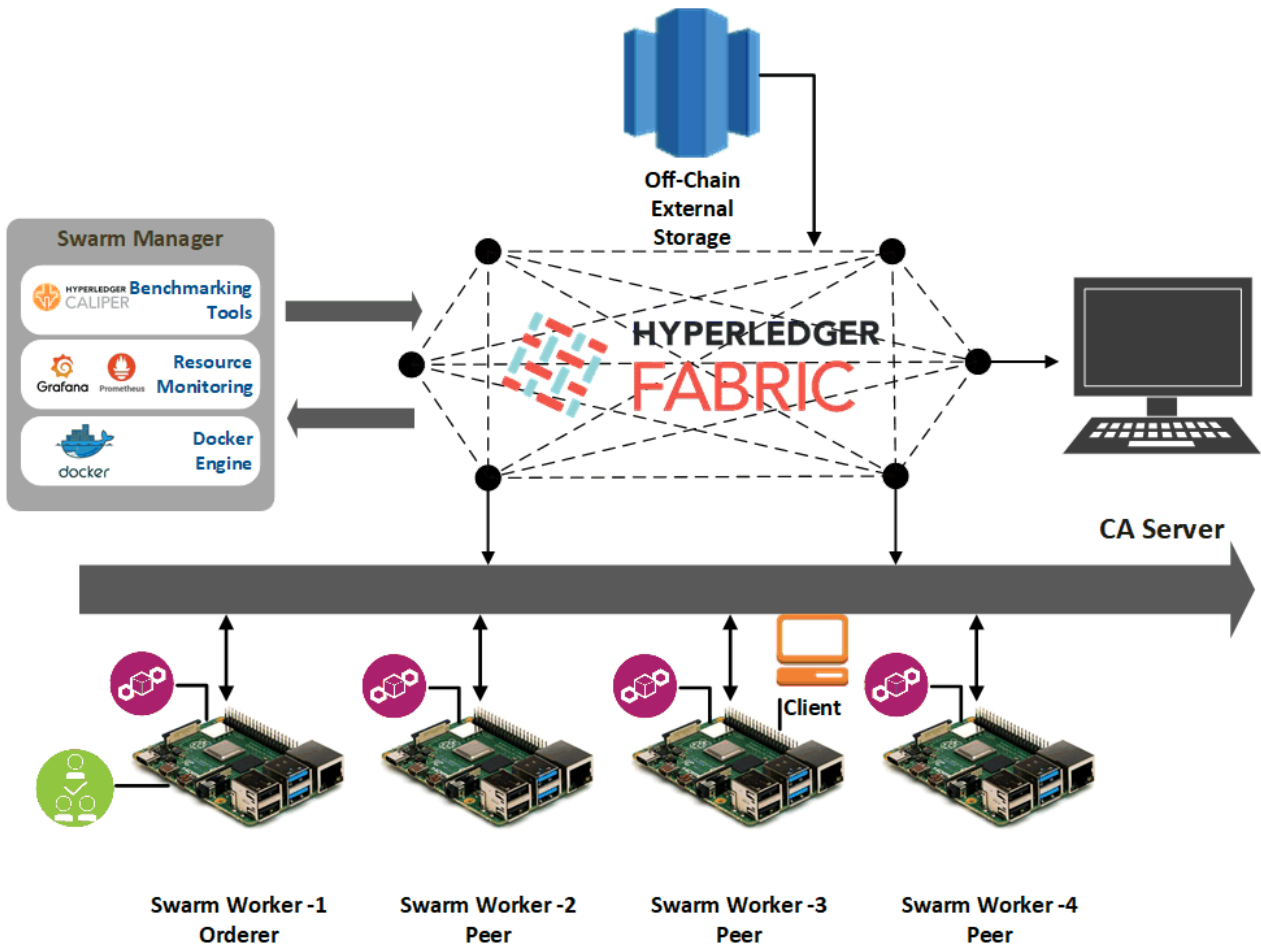


Figure 7. Experimental setup and system under test.

5.2. System Configurations

The system configurations encompass various tasks while taking into account also configuring system dependencies. They included Docker composes configuration, docker swarm setup, loading needed certificates and different scripts, CC configurations, external off-chain storage setting, various network access, modifying Docker images for RPi, etc. Many issues were coming from unsupported 64-bit RPi images, including software, libraries, and kernel issues. A shared Docker swarm network was implemented to manage and deploy multiple Docker containers to edge IoT nodes. Docker composes and related compose files were the central point for configuring containers deployment, modifying variables, initializing scripts, and testing the fabric network. Docker images were built to suit the RPi 64-bit ARMv8 architecture as the HLF does not officially support ARM architecture.

5.3. Transaction Throughput

Transaction Throughput is a performance metric defined by the Hyperledger Performance and Scale Working Group [84]. This metric represents the number of transactions processed by blockchain, leading to writing the outcome in a distributed ledger within a specific time. For this purpose and to measure the throughput, multiple rounds of benchmark applications were run on the top of the implemented HLF network with varying transaction batches. The corresponding time for each transaction and batch were measured through the benchmark application. The total time and average time were found to determine the response times and the number of transactions per minute.

5.3.1. Desktop Measurements

The throughput measurement was conducted by submitting several transactions together while varying load intensity levels. Figure 8a indicates exponential growth in the throughput with the batch sizes increase until it reaches its peak around 3500 transactions. Larger batch sizes can help the system to order more messages within the same block while it is submitted in the same timeout. Furthermore, Figure 8a indicates that many blocks are required to be filled up quickly to achieve higher throughput. The maximum number of transactions performed by the implemented virtual environment system was around 3500 transactions per minute, the peak system throughput. It is essential to consider that these large batch sizes were generated to evaluate the system performance. The system was limited to 58 transactions per second (approximately 3500 transactions per minute) due to the hardware capability of the virtual desktop.

Transactions response time is illustrated in Figure 8b. The response time increased with the growth in batch size. A large number of transactions caused system congestion—more transactions needed to be handled by peers and verified by the Orderer. Therefore, the individual transaction response time increased accordingly. As shown in Figure 8b, the transactions were handled quickly at the beginning of the process. However, the response time increased with the growth in the number of transactions in the queue to be handled and verified.

With the increase in the transaction arrival rate, the throughput increased linearly as expected until it flattened out at the peak point. This was because the number of ordered transactions waiting in the queue during the validation phase grew rapidly while subsequently affecting the commit latency. It shows that the validation phase was a bottleneck in the system performance. An increase in the number of Orderer nodes and validation peers could address this challenge.

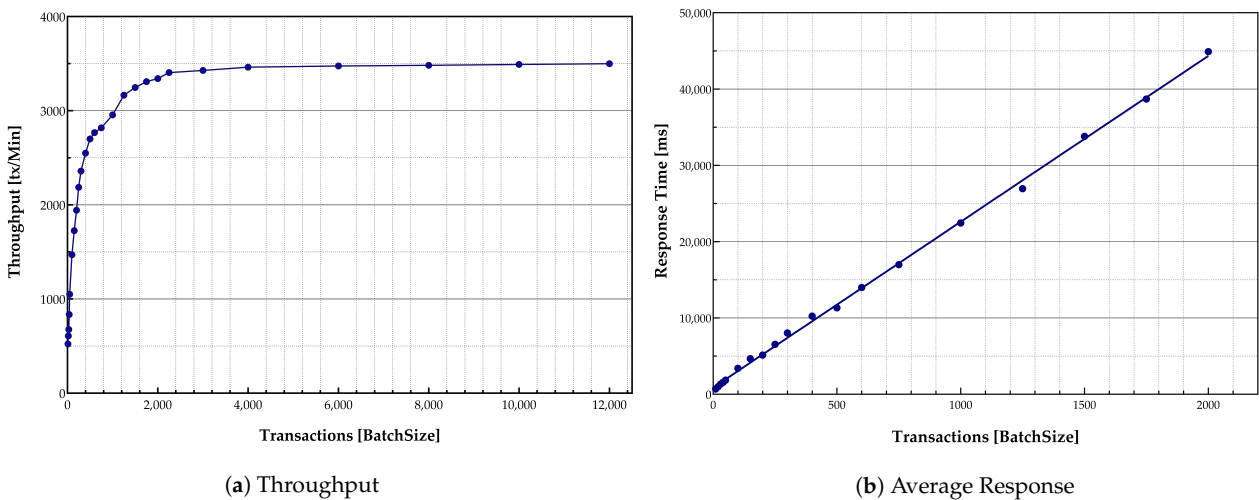


Figure 8. Effects of transaction sizes on the throughput and average response times in Desktop setup.

5.3.2. Raspberry Pi Measurements

The same system evaluation was performed in the environment consisting of RPi devices so to compare with the results obtained while using the virtual desktop setup.

The results that are shown in Figure 9a,b confirm the same trend as was observed previously while using the desktop setup. The maximum throughput peak happened around 750 transactions batch size per minute (i.e., 12 per second), which is lower than the results for the virtual desktop case. Moreover, the higher response times than in the desktop version were observed. The peak throughput occurred in the batch sizes

around 750 transactions per minute due to constraints of RPi devices in terms of the CPU capabilities.

The blockchain distributed ledger may be limited due to the amount of data stored in the blockchain system. The growth in the shared ledger causes degradation in the performance. To address this issue, the provenance of data was kept in the HLF ledger. External storage was dedicated in layer-4 of the proposed model to store the data verified by immutable blockchain records.

It should be noted that the results show satisfactory performance for the system in general. However, it is expected that the same results could be achieved by adding more clients to the system. Most of the restrictions, in this case, are related to the client's hardware on which the applications are run and are related to the peer nodes' limitations. The results show that storing information and recording data in the ledger do not affect the system performance any much. However, the limitations are mostly related to the time required to perform these operations as it should be done in a sequence, thereby affecting bandwidth and response times.

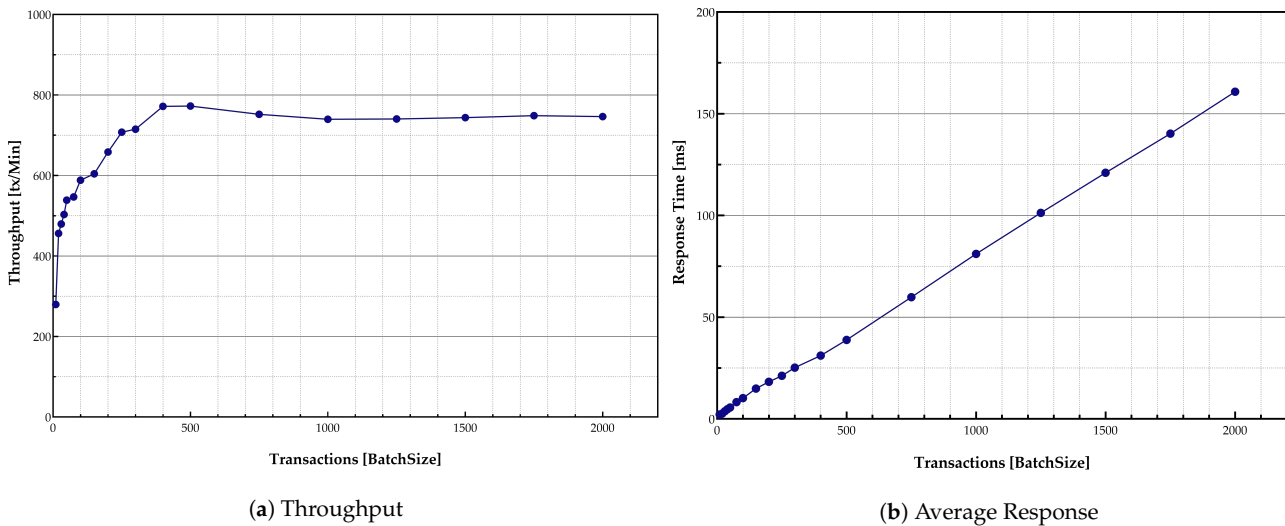


Figure 9. Effects of transaction sizes on the throughput and average response times in Raspberry Pi setup.

5.4. Transactions Latency

Transaction Latency indicates the time between the invoking of a transaction by a client and recording the transaction on the ledger. In the experimental setup, the measurements of a single transaction latency were performed by an application that sent a defined number of transactions to the HLF network while recording the individual transaction time, total average time, and corresponding statistical metrics. The results are shown in Figure 10 for CC Operation latency are the average of 100 separate operations.

Table 1 presents the results for operator SET in both desktop and RPi setup. It is evident from Table 1 that in the case of operator SET, the Raspberry Pi setup measurements were worse than those associated with the Desktop setup. The reason for this can be found in the standard deviation of related measures. The results of throughput measurements in the case of Raspberry Pi show a lot of fluctuations compared to the desktop option. It can be explained as the capability difference between the two implementations. Indeed, it took 2109 ms to submit a transaction and confirm it by running the HLF on the Desktop setup, while the time for Raspberry Pi was about 2348 ms. The Retrieving operations time for GET operators was about 100 ms in both cases. The results for RPi indicate more delays compared to the desktop environment. When the number of ordered transactions waiting in the verification process queue during the validation phase increased, it significantly increased the commit latency. Therefore, a validation phase can be considered to be a

bottleneck. However, the increase in the number of involved peers also causes higher latency. Furthermore, the experiments indicate that for real applications such as IoT to achieve lower transaction latency, the use of a smaller block size with a low transaction rate would be needed. In contrast, the higher transaction rates need a larger block size to achieve higher throughput and lower transaction latency.

Table 1. Statistics analysis of SET ChainCode latency.

Setup Environment	Avg	Std	Med	Max	Min
Desktop	2109	42.5	2105	2518	2103
RPi	2348	252	2306	4029	2204

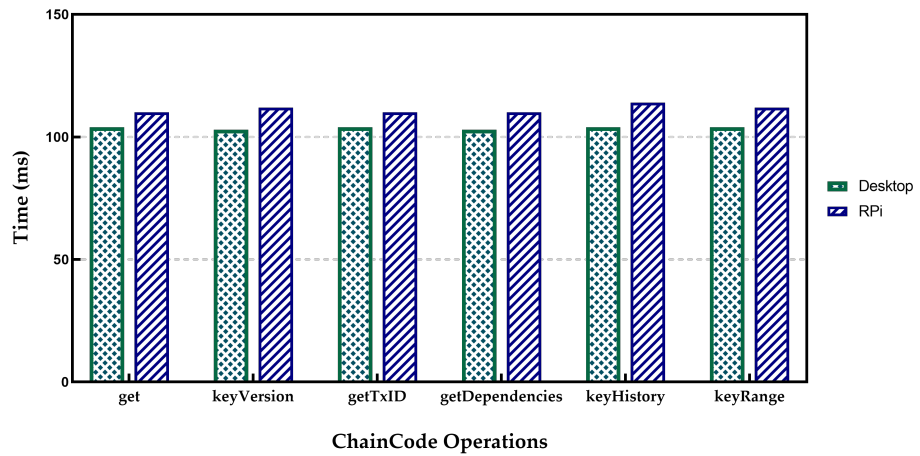


Figure 10. Latency for all ChainCode operation.

The experiment was further developed with multiple rounds of the benchmark to submit transactions with different sending rates starting from 10 to 500 transactions per second (TPS) for different block sizes. The experiment aimed to measure the maximum, average, and minimum transaction latency and transaction throughput. The results are presented in Figure 11. The minimum latency remained below 1 s during the experiments, while the maximum latency proliferated as the send rate reached 100 TPS.

5.5. Resource Consumption

Resource measurements encompass CPU computational capability, memory, and network use. The measurements carried out with varying load levels employed edge, middle, and large load cases. The operation of storing various data sizes in the network was performed with different transactions to calculate the resource consumption. The volumes were different for desktop and Raspberry Pi network setups due to hardware limitations and RPi devices' capability.

5.6. CPU and Memory Use Measurements

The CPU and memory activities were measured with the Psrecord utility [85] by attaching the processes' pid and submitting transactions with varying data sizes. Psrecord is an open-source monitoring tool that can record real-time metrics in time-series databases. The Psrecord monitors and records a defined process. The specific usage is recorded by the Psrecord tool up to a maximum of 400% of maximum system use. The result for Orderer and ChainCode processes indicates that the resource consumption of these two processes was negligible. The Peer nodes consumed most of the memory and CPU resources. This was because the verification of the transaction and smart contracts by peer nodes required high CPU usage. Therefore, the investigation mainly dealt with the peer process and client application processes.

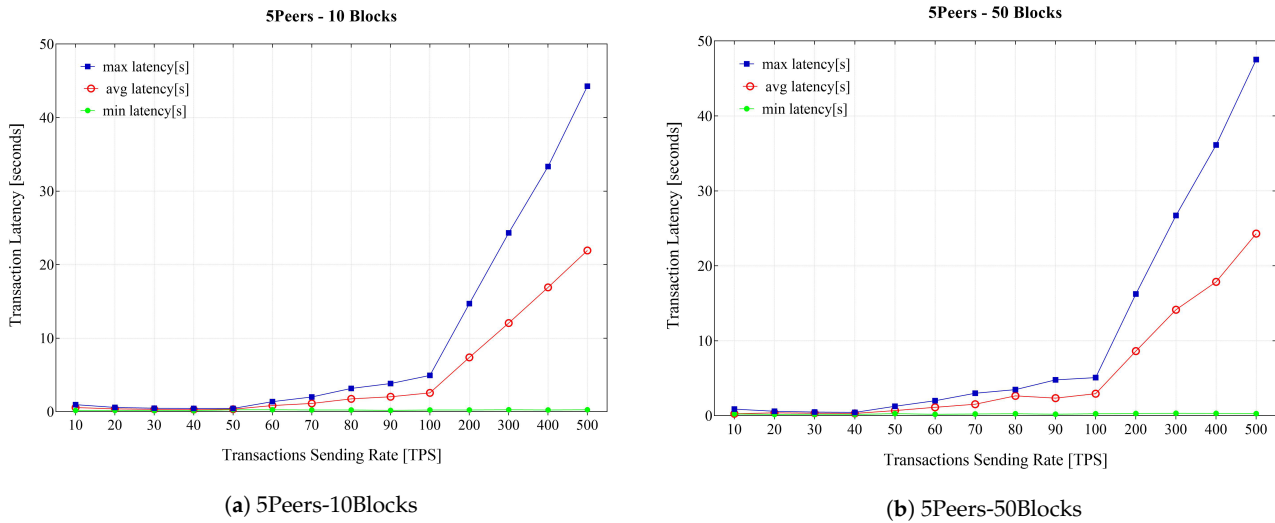
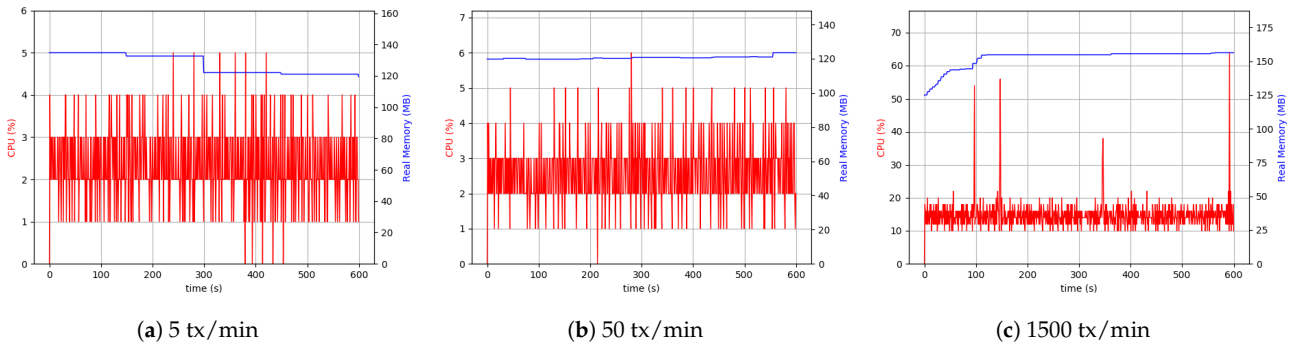


Figure 11. Latency vs. transaction sending rate.

5.6.1. Desktop Setup

Evaluation of the CPU and memory use by the involved process provided a comprehensive view of the overhead and the impact on the device hardware. Therefore, a series of measurements were conducted to analyze resources' consumption, including the resources of the network, CPU, and memory of the involved devices. Peer, Orderer, ChainCode, and application client processes were involved. The experiment was initiated by sending 3000 transactions per minute each of 1 KByte. The initial measurements indicated a high dependency on peer and client processes to the data sizes and throughput. However, Orderer and ChainCode processes used a small CPU capacity percentage (about 9%) and memory (approximately 16 MByte and 33 MByte). Due to that fact, the evaluation and analysis were focused more on peer and client processes' usage of resources. With lower load sizes, the peer processes showed similar behavior. When increasing the throughput, the peer process used a higher CPU percentage (about 20%), and memory usage at around 150 MByte. The client process used approximately 40% of the CPU capacity continuously and used 120 MByte of memory. The reason for this can be attributed to multiple processes in the client. It mainly involves connecting to a peer for each transaction, invoking CC and related operators, performing related transactions, executing the proposal requests and responses related to ordered transactions. The use of resources is also increased if the client uses external storage. In this case, it needs to calculate the checksums stored in the ledger as well as storing the data in external storage. These experiments were carried out with the highest possible load amount (in the real-world scenarios, these values would be significantly lower). The results are presented in Figure 12.

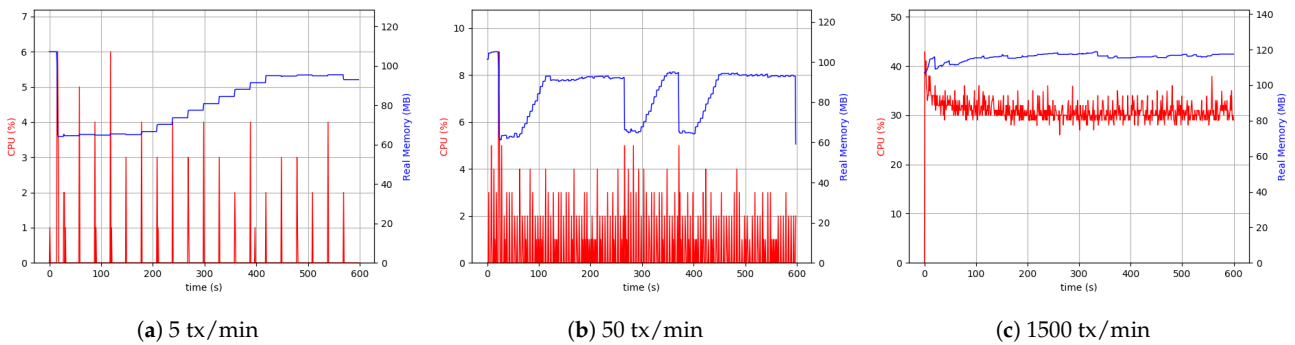
Similar to the scenario with the client process, the peer process used about 40% of CPU capacity and 150 MByte of memory. One of the key elements in any HLF network is a peer node and its related processes, playing a vital role in ordering transactions. The peer node plays the role of a response coordinator to all components and from them while Peers must keep the ledger coordinated across the HLF network. Peers connect with the channels, and they can receive all the transactions that are getting broadcasted on that channel. Peer nodes' measurements show more resource consumption than the orderer, ChainCode, and clients to synchronize with other components in the HLF network. To better evaluate and analyze peer and client processes' behavior, the consumption of resources at different data size levels with three separate throughputs were investigated. The different levels selected were low throughput and large data size (small), low throughput and small data size (medium), and high throughput and small data size (large).



(a) 5 tx/min (b) 50 tx/min (c) 1500 tx/min

Figure 12. CPU and memory use for varying data sizes for peer process in the Desktop setup.

The results are plotted in Figure 13 for CPU and Memory use of peer and client application processes over 10 min span with sampling per second. As seen in the plots, the peer process required a higher CPU use for the larger load with 30% increase. Similarly, the use of memory was higher, as the peer process must handle more transactions. To evaluate the client process performance and related applications, external storage was added to assess its impact on CPU and memory use. From the low number of transactions and up to many transactions, these values were sampled (Figure 13). Larger files needed more CPU and memory levels. Finally, it can be concluded that the client process can be influenced by the file size and the level of the load intensity to handle.



(a) 5 tx/min (b) 50 tx/min (c) 1500 tx/min

Figure 13. CPU and memory use for varying data sizes for client process in the Desktop setup.

5.6.2. Raspberry Pi Setup

Following up with analyzing the use of the resources, the RPi system setup was tested. It is crucial to acknowledge that the RPi hardware was less capable and had hardware limitations. Therefore, it was necessary to pay attention to the data sizes sent through and the number of transactions. Consequently, we considered the maximum number of transactions to be 500 per minute.

As is evident from Figure 14, the difference between 5 transactions per minute and 50 transactions per minute cases was more visible than the desktop setup. The continuation of the comparisons led to the conclusion that with the same throughput, the RPi uses more CPU resources (4 to 5 times more), which was interpreted as a hardware restriction inherent to RPi devices. Although it was not possible to make a comprehensive comparison between 500 transactions (tx) per minute in the case related to RPi setup and 1500 tx per minute related to desktop setup, as shown in Figure 14, the CPU usage and memory were approximately the same in both the cases.

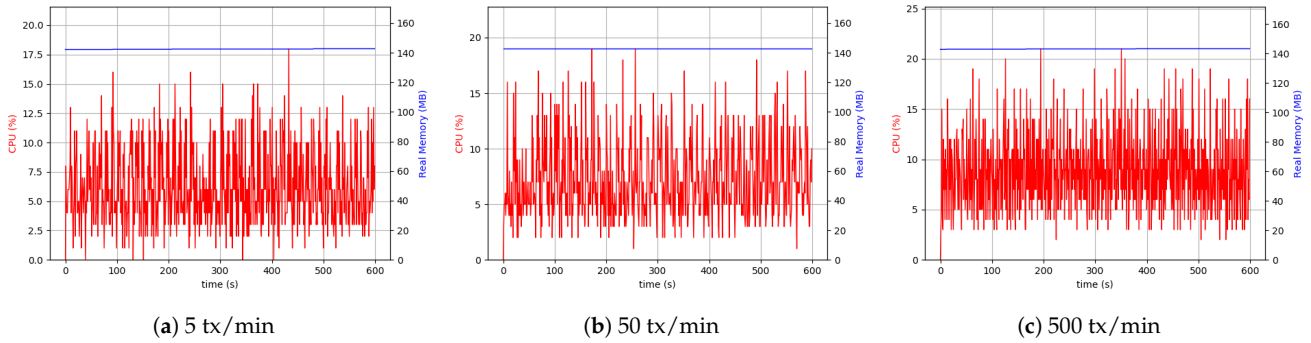


Figure 14. CPU and memory use for varying data sizes for peer process in RPi setup.

Similarly, the same measurements were performed for the client application process in the RPi setup. In this case, external data storage was considered. Figure 15 shows the results of the experiment. The higher usage of CPU was due to the difference in device-related clock rate in each of the separate setups. The peer process memory consumption was higher in the RPi setup compared to the desktop one. This can be found in peer process behavior in handling transactions. In both the setups in the client application process, the level of memory use was similar. However, in all cases, the use of 200 MByte to 300 MByte of memory was sufficient, and it was not considered the system's main limitation. The Desktop setup's resource consumption with a realistic transaction load size of around 50 KByte every five seconds was around 5% CPU and 15% in RPi.

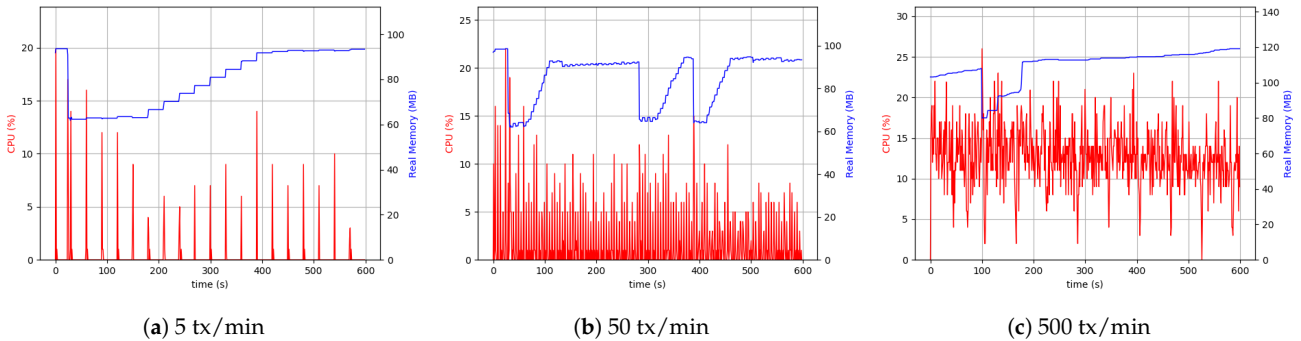


Figure 15. CPU and memory for varying data sizes for client process in RPi setup.

5.7. Network Use Measurements

To assess the consumption of available network resources and to check the network overhead, launching the peer node and client application node locally could be employed to send the transactions to the orderer, other peers, and external data storage. If the peer node is launched locally, it allows us to monitor ledger updates. At the same time, all transmitted traffics between different involved participants can be checked. Furthermore, it would be possible to have an overview of all the factors of the transmitted data.

To measure and analyze network traffic, the Speedometer utility running on the Linux environment [86] was used. Speedometer measured the sent and received network traffic over a specific network interface. All other network activities were disabled. The HLF network and external storage-related communication processes were monitored using the iftop Linux monitoring tool to measure network traffic accurately. The experiments were initiated without running any processes such as the Docker, and only the process run by the operating system to be monitored was allowed. The results show that baseline 3–5 KByte/s data can be written off to others as the network traffic.

With running the HLF, significant changes in network traffic were detectable. Figure 16 displays that with the onset of the peer process, network traffic increased by about five times compared to the baseline mode. In this case, there were no transactions between peers. The main reason for this was the beginning of the communication between peer process and network components, to have ledger consistency and reaching a synchronization through the gossip protocol. For further analysis and finding out how network resources would be affected by offered load, different offered load levels were engaged, and various modes were evaluated with and without external storage resources. The relevant results are presented in Figure 17.

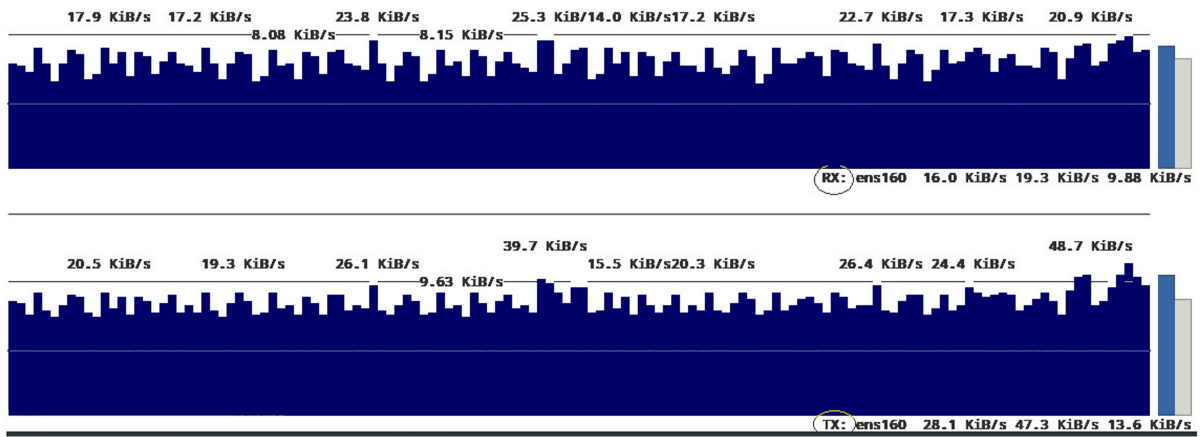


Figure 16. Network use for peer process with no transactions.

The results show that receiving and sending traffic to perform transactions every 5 s occupies something about 1–40 KByte/s spectrum. Involving an external storage source significantly increases traffic and increases its range to about 100 KByte/s. This increase was also visible in the incoming traffic and indicated by the file storage’s confirmation in the shared folder. Further increase in the number of transactions would increase the sent and received traffic.

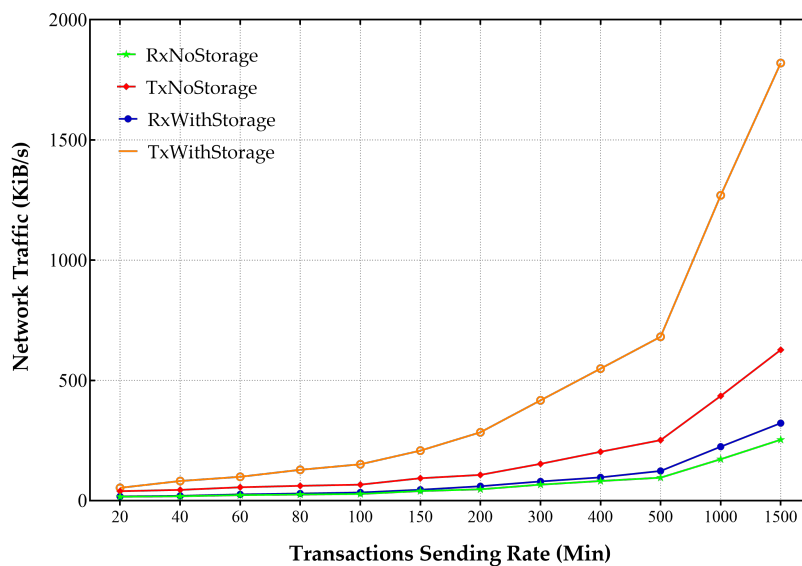


Figure 17. Network use vs. load sizes with/without external storage.

6. Conclusions

Providing security to massive interconnected IoT devices while ensuring the scalability of IoT systems with minimum resource requirements is a challenging problem. Additionally, the heterogeneity and diversity of connected devices within the IoT realm make it even more challenging. Therefore, the interoperability, identity, and privacy of IoT systems need to be guaranteed securely. The existing centralized solutions, such as a cloud-centric model, are costly. Moreover, these solutions' latency is also noticeable. Furthermore, the single point of failure issue is a considerable risk to the security of the centralized solutions. Blockchain technology is a promising solution to provide security for IoT devices while leveraging trust and interoperability.

This paper presented an implementation of the Hyperledger Fabric Blockchain platform as a permissioned blockchain technology integrated with edge IoTs to test and analyze the performance of the proposed Blockchain-based multi-layer IoT security model. The presented proof of concept was implemented using two different environment setups on the Raspberry Pi devices and VMware Virtual desktops. The performance metrics such as transaction throughput, transaction latency, computational resources, and network use of the implemented networks, were evaluated. The implemented prototype facilitates the record of sensing data by IoT devices (metadata) in a tamper-proof and transparent blockchain-based framework to provide data traceability. Moreover, the framework's security is guaranteed by implementing a layer-wise blockchain approach and local authentication process for IoT nodes in each cluster. The client application is developed with the help of Hyperledger Node SDK where various Hyperledger ChainCodes help to perform local authentication and authorization. Moreover, they facilitate the record of file pointers to provide checksums traceability and data validation.

The presented findings indicate a significantly optimal throughput for IoT applications. Peers and clients' processes are the primary source of resource consumption in the network. The Orderer and ChainCode use fewer resources compared to the peer process. Experimental results show a significant increase in throughput of approximately six times compared to the optimal scale implementation of HLF. The Desktop setup's resource consumption with a realistic transaction load size of around 50 KByte every five seconds is around 5% CPU and for the RPi setup is around 15% CPU. Peer and client processes are the primary resource consumers in HLF as our measurements indicate an average of 40% to 50% CPU consumption respectively at full load, while these measurements for the Orderer process and ChainCode use an average of about 10% of CPU resources. The deployed model could retrieve a single record in 100 ms. However, the use of the built-in ChainCode queries allows retrieving 10 dependent IoT records in 102 ms. The empirical results all indicate low overhead for running the proposed model.

Further work will consider the deployment of the proposed model in larger-scale IoT scenarios significantly increasing the number of peers for the empirical analysis of the system performance for both overall and detailed Fabric performance metrics, including throughput, latency, block size, endorsement policy, and scalability.

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

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors acknowledge Massey University for the resources provided to conduct this research. H.H.P. also acknowledges the support received through the Massey University Doctoral Scholarship.

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

Abbreviations

The following abbreviations are used in this manuscript:

BFT	Byzantine Fault Tolerant
BS	Base Station
CA	Certification Authority
CC	ChainCodes
CH	Cluster Head
CID	Client Identity
CRL	Certificate Revocation List
CPSs	Cyber-Physical Systems
dApps	distributed Applications
DDoS	Distributed Denial-of-Service
DHE	Diffie-Hellman Ephemeral
DLTS	Distributed Ledger Technologies
DPoS	Delegated Proof of Stake
ECC	Elliptic Curve Cryptography
HLF	Hyperledger Fabric
IoT	Internet of Things
IIoT	Industrial IoT
MAC	Message Authentication Code
MSP	Membership Services Provider
NFS	Network File Systems
OPM	Open Provenance Model
PBFT	Practical Byzantine Fault Tolerance
PoA	Proof of Authority
PoB	Proof of Bandwidth
PoET	Proof of Elapsed Time
PoS	Proof of Stake
PoW	Proof of Work
RPi	Raspberry Pi
RSA	Rivest-Shamir-Adleman
SDN	Software Defined Networking
SSL	Secure Sockets Layer
TSL	Transport Layer Security
WSNs	Wireless Sensor Networks

References

1. Ali, M.S.; Vecchio, M.; Pincheira, M.; Dolui, K.; Antonelli, F.; Rehmani, M.H. Applications of blockchains in the Internet of Things: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 1676–1717. [[CrossRef](#)]
2. Lao, L.; Li, Z.; Hou, S.; Xiao, B.; Guo, S.; Yang, Y. A survey of IoT applications in blockchain systems: Architecture, consensus, and traffic modeling. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–32. [[CrossRef](#)]
3. Javaid, U.; Siang, A.K.; Aman, M.N.; Sikdar, B. Mitigating IoT device based DDoS attacks using blockchain. In Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, Munich, Germany, 15 June 2018; pp. 71–76.
4. Zhou, X.; Tang, X. Research and implementation of RSA algorithm for encryption and decryption. In Proceedings of the 2011 6th International Forum on Strategic Technology, Harbin, China, 22–24 August 2011; Volume 2, pp. 1118–1121.
5. Suárez-Albela, M.; Fernández-Caramés, T.M.; Fraga-Lamas, P.; Castedo, L. A practical evaluation of a high-security energy-efficient gateway for IoT fog computing applications. *Sensors* **2017**, *17*, 1978.
6. Oppliger, R. *SSL and TLS: Theory and Practice*; Artech House: Norwood, MA, USA, 2016.
7. Caudhari, A.; Bansode, R. Securing IoT devices generated data using homomorphic encryption. In *Intelligent Computing and Networking*; Springer: Singapore, 2020; pp. 219–226.
8. Hou, L.; Zheng, K.; Liu, Z.; Xu, X.; Wu, T. Design and prototype implementation of a blockchain-enabled LoRa system with edge computing. *IEEE Internet Things J.* **2020**. [[CrossRef](#)]

9. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
10. Panarello, A.; Tapas, N.; Merlino, G.; Longo, F.; Puliafito, A. Blockchain and IoT integration: A systematic survey. *Sensors* **2018**, *18*, 2575. [[CrossRef](#)]
11. Johansen, S.K. *A Comprehensive Literature Review on the Blockchain as a Technological Enabler for Innovation*; Department of Information Systems, Mannheim University: Mannheim, Germany, 2018; pp. 1–29.
12. Ongaro, D.; Ousterhout, J. In search of an understandable consensus algorithm. In Proceedings of the 2014 USENIX Annual Technical Conference (USENIXATC 14), Philadelphia, PA, USA, 19–20 June 2014; pp. 305–319.
13. Merkel, D. Docker: Lightweight Linux containers for consistent development and deployment. *Linux J.* **2014**, *2014*, 2.
14. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [[CrossRef](#)]
15. Stojkoska, B.L.R.; Trivodaliev, K.V. A review of Internet of Things for smart home: Challenges and solutions. *J. Clean. Prod.* **2017**, *140*, 1454–1464. [[CrossRef](#)]
16. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of Things for Smart Cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [[CrossRef](#)]
17. Queralta, J.P.; Gia, T.N.; Tenhunen, H.; Westerlund, T. Collaborative mapping with ioe-based heterogeneous vehicles for enhanced situational awareness. In Proceedings of the 2019 IEEE Sensors Applications Symposium (SAS), Sophia Antipolis, France, 11–13 March 2019; pp. 1–6.
18. Mutlag, A.A.; Abd Ghani, M.K.; Arunkumar, N.A.; Mohammed, M.A.; Mohd, O. Enabling technologies for fog computing in healthcare IoT systems. *Future Gener. Comput. Syst.* **2019**, *90*, 62–78. [[CrossRef](#)]
19. Qingqing, L.; Yuhong, F.; Queralta, J.P.; Gia, T.N.; Tenhunen, H.; Zou, Z.; Westerlund, T. Edge computing for mobile robots: multi-robot feature-based lidar odometry with FPGAs. In Proceedings of the 2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU), Kathmandu, Nepal, 4–6 November 2019; pp. 1–2.
20. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Yhings J.* **2016**, *3*, 637–646. [[CrossRef](#)]
21. Lee, S.K.; Bae, M.; Kim, H. Future of IoT networks: A survey. *Appl. Sci.* **2017**, *7*, 1072. [[CrossRef](#)]
22. Butun, I.; Österberg, P.; Song, H. Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures. *IEEE Commun. Surv. Tutor.* **2019**, *22*, 616–644. [[CrossRef](#)]
23. Bahga, A.; Madiseti, V.K. Blockchain platform for industrial internet of things. *J. Softw. Eng. Appl.* **2016**, *9*, 533–546. [[CrossRef](#)]
24. Huh, S.; Cho, S.; Kim, S. Managing IoT devices using blockchain platform. In Proceedings of the 2017 19th International Conference on Advanced Communication Technology (ICACT), Pyeongchang, Korea, 19–22 February 2017; pp. 464–467. [[CrossRef](#)]
25. Sharma, P.K.; Singh, S.; Jeong, Y.; Park, J.H. DistBlockNet: A Distributed Blockchains-Based Secure SDN Architecture for IoT Networks. *IEEE Commun. Mag.* **2017**, *55*, 78–85. [[CrossRef](#)]
26. Song, J.C.; Demir, M.A.; Prevost, J.J.; Rad, P. Blockchain design for trusted decentralized IoT networks. In Proceedings of the 2018 13th Annual Conference on System of Systems Engineering (SoSE), Paris, France, 19–22 June 2018; pp. 169–174.
27. Qian, Y.; Jiang, Y.; Chen, J.; Zhang, Y.; Song, J.; Zhou, M.; Pustišek, M. Towards decentralized IoT security enhancement: A blockchain approach. *Comput. Electr. Eng.* **2018**, *72*, 266–273. [[CrossRef](#)]
28. Ayoade, G.; Karande, V.; Khan, L.; Hamlen, K. Decentralized IoT data management using blockchain and trusted execution environment. In Proceedings of the 2018 IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, USA, 6–9 July 2018; pp. 15–22.
29. Su, P.H.; Shih, C.S.; Hsu, J.Y.J.; Lin, K.J.; Wang, Y.C. Decentralized fault tolerance mechanism for intelligent IoT/M2M middleware. In Proceedings of the 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, Korea, 6–8 March 2014; pp. 45–50.
30. Chen, J. Devify: Decentralized internet of things software framework for a peer-to-peer and interoperable iot device. *ACM SIGBED Rev.* **2018**, *15*, 31–36. [[CrossRef](#)]
31. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017; pp. 557–564.
32. Karame, G. On the security and scalability of bitcoin’s blockchain. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 1861–1862.
33. Scherer, M. Performance and Scalability of Blockchain Networks and Smart Contracts. Master’s Thesis, Umeå University, Umeå, Sweden, 2017.
34. Christidis, K.; Devetsikiotis, M. Blockchains and smart contracts for the internet of things. *IEEE Access* **2016**, *4*, 2292–2303. [[CrossRef](#)]
35. Peng, Z.; Wu, H.; Xiao, B.; Guo, S. VQL: Providing query efficiency and data authenticity in blockchain systems. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW), Macao, China, 8–12 April 2019; pp. 1–6.
36. Biswas, K.; Muthukkumarasamy, V. Securing Smart Cities Using Blockchain Technology. In Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Sydney, Australia, 12–14 December 2016; pp. 1392–1393. [[CrossRef](#)]

-
37. Sharma, P.K.; Park, J.H. Blockchain based hybrid network architecture for the smart city. *Future Gener. Comput. Syst.* **2018**, *86*, 650–655. [[CrossRef](#)]
 38. Mbarek, B.; Jabeur, N.; Pitner, T. Mbs: Multilevel blockchain system for IoT. *Pers. Ubiquitous Comput.* **2019**, 1–8. [[CrossRef](#)]
 39. Xuan, S.; Zhang, Y.; Tang, H.; Chung, I.; Wang, W.; Yang, W. Hierarchically Authorized Transactions for Massive Internet-of-Things Data Sharing Based on Multilayer Blockchain. *Appl. Sci.* **2019**, *9*, 5159. [[CrossRef](#)]
 40. Butun, I.; Österberg, P. Detecting intrusions in cyber-physical systems of smart cities: Challenges and directions. In *Secure Cyber-Physical Systems for Smart Cities*; IGI Global: Goteborg, Sweden, 2019; pp. 74–102.
 41. Khan, P.W.; Byun, Y. A Blockchain-Based Secure Image Encryption Scheme for the Industrial Internet of Things. *Entropy* **2020**, *22*, 175. [[CrossRef](#)]
 42. Butun, I. *Industrial IoT: Challenges, Design Principles, Applications, and Security*; Springer Nature: Cham, Switzerland, 2020.
 43. Zhu, H.; Huang, C.; Zhou, J. Edgechain: Blockchain-based multi-vendor mobile edge application placement. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 25–29 June 2018; pp. 222–226.
 44. Queralta, J.P.; Qingqing, L.; Gia, T.N.; Truong, H.L.; Westerlund, T. End-to-End Design for Self-Reconfigurable Heterogeneous Robotic Swarms. *arXiv* **2020**, arXiv:2004.13997.
 45. Dai, Y.; Xu, D.; Maharjan, S.; Chen, Z.; He, Q.; Zhang, Y. Blockchain and deep reinforcement learning empowered intelligent 5G beyond. *IEEE Netw.* **2019**, *33*, 10–17. [[CrossRef](#)]
 46. Xiong, Z.; Zhang, Y.; Niyato, D.; Wang, P.; Han, Z. When mobile blockchain meets edge computing. *arXiv* **2017**, arXiv:1711.05938.
 47. Rahman, M.A.; Hossain, M.S.; Loukas, G.; Hassanain, E.; Rahman, S.S.; Alhamid, M.F.; Guizani, M. Blockchain-Based Mobile Edge Computing Framework for Secure Therapy Applications. *IEEE Access* **2018**, *6*, 72469–72478. [[CrossRef](#)]
 48. Samaniego, M.; Deters, R. Using blockchain to push software-defined IoT components onto edge hosts. In Proceedings of the International Conference on Big Data and Advanced Wireless Technologies, Blagoevgrad, Bulgaria, 10–11 November 2016; pp. 1–9.
 49. Samaniego, M.; Deters, R. Virtual Resources & Blockchain for Configuration Management in IoT. *J. Ubiquitous Syst. Pervasive Netw.* **2018**, *9*, 1–13.
 50. Queralta, J.P.; Qingqing, L.; Zou, Z.; Westerlund, T. Enhancing Autonomy with Blockchain and Multi-Access Edge Computing in Distributed Robotic Systems. In Proceedings of the Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 20–23 April 2020.
 51. Soldani, D. 5G and the Future of Security in ICT. In Proceedings of the 2019 29th International Telecommunication Networks and Applications Conference (ITNAC), Auckland, New Zealand, 27–29 November 2019; pp. 1–8.
 52. Ferrer, E.C.; Rudovic, O.; Hardjono, T.; Pentland, A. Robochain: A secure data-sharing framework for human-robot interaction. *arXiv* **2018**, arXiv:1802.04480.
 53. Ma, Z.; Meng, J.; Wang, J.; Shan, Z. Blockchain-based Decentralized Authentication Modeling Scheme in Edge and IoT Environment. *IEEE Internet Things J.* **2020**. [[CrossRef](#)]
 54. Hewa, T.; Braeken, A.; Ylianttila, M.; Liyanage, M. Multi-Access Edge Computing and Blockchain-based Secure Telehealth System Connected with 5G and IoT. In Proceedings of the 8th IEEE International Conference on Communications and Networking (IEEE ComNet'2020), Hammamet, Tunisia, 28–30 October 2020.
 55. Attia, O.; Khoufi, I.; Laouiti, A.; Adjih, C. An Iot-blockchain architecture based on hyperledger framework for healthcare monitoring application. In Proceedings of the 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Canary Islands, Spain, 24–26 June 2019; pp. 1–5.
 56. Butun, I.; Sari, A.; Österberg, P. Hardware Security of Fog End-Devices for the Internet of Things. *Sensors* **2020**, *20*, 5729. [[CrossRef](#)]
 57. Pešić, S.; Radovanović, M.; Ivanović, M.; Tošić, M.; Ikočić, O.; Bošković, D. Hyperledger Fabric Blockchain as a Service for the IoT: Proof of Concept. In Proceedings of the International Conference on Model and Data Engineering, Toulouse, France, 28–31 October 2019; pp. 172–183.
 58. Heller, B.; Sherwood, R.; McKeown, N. The controller placement problem. *ACM SIGCOMM Comput. Commun. Rev.* **2012**, *42*, 473–478. [[CrossRef](#)]
 59. Suen, C.H.; Ko, R.K.; Tan, Y.S.; Jagadpramana, P.; Lee, B.S. S2logger: End-to-end data tracking mechanism for cloud data provenance. In Proceedings of the 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, Melbourne, Australia, 16–18 July 2013; pp. 594–602.
 60. Salman, T.; Zolanvari, M.; Erbad, A.; Jain, R.; Samaka, M. Security services using blockchains: A state of the art survey. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 858–880. [[CrossRef](#)]
 61. Liang, X.; Shetty, S.; Tosh, D.; Kamhoua, C.; Kwiat, K.; Njilla, L. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, Spain, 14–17 May 2017; pp. 468–477.
 62. Neisse, R.; Steri, G.; Nai-Fovino, I. A blockchain-based approach for data accountability and provenance tracking. In Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, 29 August–1 September 2017; pp. 1–10.
-

63. Demichev, A.; Kryukov, A.; Prikhodko, N. The approach to managing provenance metadata and data access rights in distributed storage using the hyperledger blockchain platform. In Proceedings of the 2018 Ivannikov Ispras Open Conference (ISPRAS), Moscow, Russia, 22–23 November 2018; pp. 131–136.
64. Ramachandran, A.; Kantarcioglu, M. Smartprovenance: A distributed, blockchain based dataprovenance system. In Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, Tempe, AZ, USA, 19–21 March 2018; pp. 35–42.
65. Latif, S.; Idrees, Z.; Ahmad, J.; Zheng, L.; Zou, Z. A blockchain-based architecture for secure and trustworthy operations in the industrial Internet of Things. *J. Ind. Inf. Integr.* **2020**, *21*, 100190.
66. Atlam, H.F.; Alenezi, A.; Alassafi, M.O.; Wills, G. Blockchain with internet of things: Benefits, challenges, and future directions. *Int. J. Intell. Syst. Appl.* **2018**, *10*, 40–48. [[CrossRef](#)]
67. Dorri, A.; Kanhere, S.S.; Jurdak, R. Blockchain in internet of things: Challenges and solutions. *arXiv* **2016**, arXiv:1608.05187.
68. Buterin, V. A Next-Generation Smart Contract and Decentralized Application Platform. *White Pap.* **2014**, *3*, 37.
69. Nakamoto, S.; Bitcoin, A. *A Peer-to-Peer Electronic Cash System*; Bitcoin: 2008, Volume 4. Available online: <https://bitcoin.org/bitcoin.pdf> (accessed on 30 December 2020).
70. Singhal, B.; Dhameja, G.; Panda, P.S. *Beginning Blockchain: A Beginner's Guide to Building Blockchain Solutions*; Springer: New York, NY, USA, 2018.
71. Popov, S. The Tangle. IOTA Whitepaper.pdf. 2017. Available online: <https://iota.org> (accessed on 15 November 2020).
72. Li, X.; Jiang, P.; Chen, T.; Luo, X.; Wen, Q. A survey on the security of blockchain systems. *Future Gener. Comput. Syst.* **2020**, *107*, 841–853. [[CrossRef](#)]
73. Tschorsch, F.; Scheuermann, B. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2084–2123. [[CrossRef](#)]
74. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [[CrossRef](#)]
75. Cachin, C. Architecture of the hyperledger blockchain fabric. In Proceedings of the Workshop on Distributed Cryptocurrencies and Consensus Ledgers, Chicago, IL, USA, 25 July 2016; Volume 310.
76. Castro, M.; Liskov, B. Practical Byzantine fault tolerance. In Proceedings of the OSDI, New Orleans, LA, USA, 25–26 February 1999; Volume 99, pp. 173–186.
77. Metcalfe, W. Ethereum, Smart Contracts, DApps. In *Blockchain and Crypt Currency*; Springer: Singapore, 2020; pp. 77–93.
78. Rashid, M.A.; Pajooh, H.H. A Security Framework for IoT Authentication and Authorization Based on Blockchain Technology. In Proceedings of the 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), Rotorua, New Zealand, 5–8 August 2019; pp. 264–271. [[CrossRef](#)]
79. Valenta, M.; Sandner, P. Ethereum, Hyperledger Fabric and Corda, FSBC Working Paper. June 2017. pp. 1–8. Available online: http://explore-ip.com/2017_ComparisonofEthereumHyperledgerCorda.pdf (accessed on 30 December 2020).
80. Hoskins, M.E. Sshfs: Super easy file access over ssh. *Linux J.* **2006**, *2006*, 4.
81. Rajgarhia, A.; Gehani, A. Performance and extension of user space file systems. In Proceedings of the 2010 ACM Symposium on Applied Computing, Sierre, Switzerland, 22–26 March 2010; pp. 206–213.
82. Milicchio, F.; Gehrke, W.A. OpenAFS. *Distributed Services with OpenAFS: for Enterprise and Education*; Springer: Rome, Italy, 2007; pp. 81–147.
83. Mukherjee, S. Benefits of AWS in Modern Cloud. 2019. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3415956 (accessed on 30 November 2020).
84. Hyperledger Performance and Scale Working Group, Hyperledger Blockchain Performance Metrics. 2018. Available online: Available online: https://www.hyperledger.org/wpcontent/uploads/2018/10/HL_Whitepaper_Metrics_PDF_V1 (accessed on 15 November 2020).
85. Robitaille, T. Psrecord: Record the CPU and Memory Activity of a Process. 2017. Available online: <https://github.com/astrofrog/psrecord> (accessed on 15 November 2020).
86. Ward, I. Speedometer 2.8. 2015. Available online: <http://excess.org/speedometer/> (accessed on 15 November 2020).

Chapter 5

Experimental Performance Analysis of Scalable Distributed HLF in a Large Scale IoT Testbed

Chapter 5 includes the article, “Experimental Performance Analysis of Scalable Distributed HLF in a Large Scale IoT Testbed” submitted to the IEEE IoT journal. This article is open access and has been restructured in this thesis under the Creative Commons Attribution License. No special permission is required to reuse all or part of article published by IEEE, including figures and tables. For articles published under an open access Creative Common CC BY license, any part of the article may be reused without permission provided that the original article is clearly cited. Reuse of an article does not imply endorsement by the authors or IEEE.

The full text is included in the thesis has some minor modifications. This means that while the content is identical to the published article, there may be stylistic differences.

Experimental Performance Analysis of Scalable Distributed Hyperledger Fabric in a Large Scale IoT Testbed

Houshyar Honar Pajooh, *Member, IEEE*, Mohammad A. Rashid, *Senior Member, IEEE*, Fakhrul Alam, *Senior Member, IEEE*, and Serge Demidenko, *Fellow, IEEE*

Abstract—Blockchain technology with its decentralization characteristics, immutability, and traceability is well-suited for facilitating secure storage, sharing, and management of data of decentralized Internet of Things (IoT) applications. Despite the increasing development of blockchain platforms, there is still no comprehensive method for adopting the blockchain technology on IoT systems due to the blockchain’s limited capability to process substantial transaction requests from a massive number of IoT devices. Hyperledger Fabric (HLF) is a popular open-source permissioned blockchain platform hosted by the Linux Foundation. This article reports a comprehensive empirical study that measures HLF’s performance and identifies potential performance bottlenecks to better meet the requirements of blockchain-based IoT applications. The study considers the implementation of HLF on distributed large scale IoT systems. First, we present a model for monitoring the performance of the HLF platform that addresses the overhead challenges while delivering more details on system performance and better scalability. Then, we implement the framework to evaluate the impact of varying network workloads on the performance of the blockchain platform in a real large-scale distributed environment. In particular, the performance of the HLF is evaluated in terms of throughput, latency, network size, scalability, and the number of peers serviceable by the platform. The experimental results indicate that the proposed framework can provide detailed real-time performance evaluation of blockchain systems for large scale IoT applications.

Index Terms—Blockchain, hyperledger fabric, smart contract, performance, throughput, latency, scalability.

I. INTRODUCTION

BLOCKCHAIN, an innovative technology that originated from Bitcoin [1], encompasses a list of continuously growing data and transaction records, called blocks that are cryptographically linked and secured. Peers maintain the blockchain in a peer-to-peer (P2P) transaction platform, where transactions are recorded in a period of time and packaged into a block by peers to join the blockchain ledger. Blockchain offers a decentralized network with records being tamper-resistant and traceable. Numerous blockchain-based decentralized applications have emerged with the widespread development of the technology. As a secure and unalterable architecture, blockchain is a promising paradigm to address the

needs of availability, confidentiality, and integrity for IoT applications [2]. The integration of the blockchain to the Internet of Things (IoT) is a challenging enrichment that can guarantee the privacy, security, trust, and data reliability of conventional IoT applications. The feasibility of such blockchain-based IoT systems has been extensively explored recently [3], [4]. Nonetheless, the time-consuming consensus process is the primary bottleneck of adopting blockchain technology on IoT applications. Besides, the IoT systems are varied in terms of the number of generated requests where applications generate thousands of transactions per second (TPS).

Distributed ledger technologies (DLTs) enable the storage of information securely and accurately using a set of cryptographic primitives. Once stored, the information becomes immutable. Hyperledger Fabric (HLF), a form of permissioned distributed ledger technology (DLT), helps enterprises build their specific DLT solutions more efficiently and securely. Some of the key enhancements of HLF such as ensuring data privacy, better ChainCode application patterns, externally launched ChainCodes and modified docker images pave the way for secure DLT solutions (including the decentralized ChainCode - Smart Contract). Implementing HLF facilitates managing channels, ChainCodes, policies, certificate authorities, etc., within the IoT environments where devices operate, and services provided in multiple segments or among different layers.

HLF architecture facilitates a permissioned blockchain environment. The HLF system performance is enhanced through implementing a highly modular framework and pluggable consensus. It can also provide privacy for a broad range of implementation solutions (e.g., IoT networks) while meeting the specific needs of IoT applications. Furthermore, a pluggable consensus approach improves the latency of finality and confirmation. Scalability, throughput, robust cryptographic security, latency and resource consumption are some of the major challenges while moving from traditional DLT to HLF solution. The security arrangements of distributed IoT systems can be established and maintained through deploying HLF with various features offered by different versions of the Fabric implementation. Such security requirements include system integrity, authentication and authorization, data privacy, system security, device authentication, storage, key management, access controls and revocations, device enrollment, user privacy, identity management, user authentication, and authorization and synchronization of software upgrades. Besides, Fabric

H. Honar Pajooh, M. Rashid, F. Alam are with Department of Mechanical & Electrical Engineering, School of Food and Advanced Technology, Massey University, Auckland 0632, New Zealand (e-mail: h.pajooh@massey.ac.nz).

S. Demidenko and F. Alam are with School of Science and Technology, Sunway University, 47500 Selangor, Malaysia.

offers the implementation of restricted networks and controlled access to user data within the IoT systems.

Blockchain applications have gained significant attention from both industry and academia in recent years [5], [6]. Such applications are noticeable in different domains ranging from public services [7], finance [8], smart hospitals [9], smart manufacturing [10], supply chains [11], energy trading [12] to the new era of IoT [13]. Despite the development and implementation of many blockchain projects, there are still concerns associated with the blockchain platforms' throughput, latency, and ability to scale [14].

Performance computation and evaluation represent significant challenges for current blockchain systems [15], [16], particularly during the execution of complex smart contracts. Technical challenges in adapting blockchain systems are throughput, latency, scalability, size, bandwidth, security, wasted resources, usability, as well as versioning, and hard forks [17]. Therefore, it is crucial to evaluate the real-time performance of the blockchain platforms. The performance of the blockchain system can be considered as overall performance and detailed performance. The overall performance, including the throughput and latency, can help to find the ideal blockchain system that could fit real-world application scenarios. However, the detailed performance computation reveals performance bottlenecks and provides detailed information on the entire process.

The primary focus of this work is to present an adaptive framework to enhance blockchain-based IoT system's performance in a distributed environment with many peers. The aim is to improve the transaction throughput by redesigning the consensus protocols while the security and immutability need to be accomplished by the implemented blockchain platform. Blockchain parameters affect the performance, security, and adaptability of the system. It becomes more complicated when choosing an optimal configuration in the IoT systems with vast amounts of small and resource constraint devices. The parameters need to be validated and tested before deploying the blockchain IoT systems to determine the limitations and possible bottlenecks within the implementation.

The evaluation of the blockchain system's overall performance has been studied widely in the literature [18]–[21]. However, various process stages need more detailed performance measurements as the overall metrics cannot reflect the detailed performance. There is a lack of metrics to measure and monitor the detailed performance of blockchain systems. Moreover, the framework scalability and real-time monitoring overhead need to be comprehensively studied. Besides, we need to monitor the system performance in a large scale distributed environment. Thus, the way of performance monitoring and the selection of metrics to monitor are the main challenges for blockchain's performance measurements.

The Fabric network is orchestrated by various components, including endorsers, ordering service, and committers. It constitutes different transaction processing phases consisting of the endorsement phase, ordering phase, validation, and commit phase. Therefore, Fabric encompasses various configurable parameters, such as block size, channels, endorsement policy, and state databases. Finding the right set of values for this

range of parameters is the main challenge in adapting an efficient blockchain system. A comprehensive performance analysis needs to find out the optimal block size to achieve higher throughput, lower latency while considering a more efficient type of endorsement policy in a distributed platform.

This study considers a detailed real-time performance computation model for Hyperledger Fabric (HLF) blockchain systems to address the aforementioned challenges. A comprehensive review of blockchain research topics [22] shows that latency, throughput, and scalability are the primary limiting characteristics. The selected performance computation model facilitates collecting real-time performance data by analyzing logs and the daemon process. Besides, the study considers the following factors:

- 1) Distribution. The Performance model needs to consider the real distributed scenario on a large scale.
- 2) Scalability. The performance monitoring framework requires to be extended easily to compute the performance of added new peers to the Fabric system.
- 3) Detailed monitoring. The comprehensive analysis of logs could reveal more detail data associated with various stages of the Fabric system.
- 4) Minimum overhead. The performance computation model should fit real-time monitoring and have a negligible performance impact on the running of blockchain systems.

We aim to demonstrate a model for studying the impact of the different blockchain network workloads on the performance of the first long-term support release of the Hyperledger Fabric platform, v.1.4 [16], [23]. The system performance is evaluated by implementing varying transactions, at different rates, and block sizes in a large-scale distributed platform. Performance of the Fabric network being evaluated includes the latency (in seconds), throughput (in tps), and scalability (network size and the number of participants peers) as well as the endorsement policy. The followings are the main contributions of this paper:

- 1) Conducted comprehensive experimental analysis of several HLF performance metrics to outline detailed system performance and configuration guidelines to show how various network configurations affect overall system performance.
- 2) Proposed a scalable model and tested for real-time performance computation of the Fabric platform, considering lower overheads and better scalability. The experiments were conducted on networks of different sizes with peers running on multiple hosts to identify the major limitations of the HLF network and performance bottlenecks.
- 3) Provided comprehensive metrics measurements to analyze and monitor the impact of system configurations (e.g., number of transactions, block sizes, endorsement policies, network size) on the HLF performance, especially the scalability on a distributed environment, highlighting the possibilities and limitations of HLF implementation in large-scale IoT networks.

To the best of authors' knowledge, the experimental study

reported in this research study is the first of its kind in terms of its focus and comprehensiveness. Specifically, it performs the scalability analyses for the permissioned HLF blockchain framework for IoT in a distributed computing infrastructure, identifies the bottlenecks to the scalability, and offers possible solutions for addressing them in the context of real-world IoT implementation. Furthermore, this work determines the impact of various blockchain metrics on the system's latency and resource consumption. This makes it different from the existing studies that do not consider the measurement of the relevant resource utilisation/requirements (e.g., memory and CPU consumption) with an increasing number of peers in the system. In addition, the interdependency between the scalability of the permissioned HLF blockchain networks and their resource usage was investigated.

The rest of the paper is organized as follows. In Section II, related works in performance evaluation of HLF blockchain platforms are presented. Section III provides an overview of the HLF blockchain technology and the target platforms in this study. Section IV presents the methodology for evaluating Fabric implementations, the key configuration metrics, and the experimental setup. Then, a discussion of the results and their implications are covered in Section V. Finally, Section VI concludes this paper.

II. RELATED WORKS

The empirical analysis of DLT has been well documented in the literature to evaluate the performance of the implemented blockchain systems, such as Hyperledger and Ethereum. Although the empirical analysis does not provide standardized results, this method is flexible in terms of parameterization. This approach could find potential bottlenecks and show how to optimize the system performance based on self-designed experiments. A well-controlled test environment can facilitate comparing the performance of different private blockchain platforms and various versions of a particular blockchain system. Besides, some studies were conducted to evaluate the performance of various hash and encryption algorithms from the data layer in the conceptual model of the blockchain.

The throughput and latency of HLF v1.0 were studied in [24] by deploying an experimental approach using the Caliper [25] benchmarking tool. The study explored the impact of different transactions and various ChainCodes parameters on transaction latency and throughput under micro-workloads. The authors evaluated the performance of Fabric characteristics by implementing a varying number of ChainCodes, channels, and peers. The sensitivity of HLF v1.0 throughput to the Orderer setting was highlighted in the evaluation results. Furthermore, the results showed that the HLF v1.0 committer was incapable of handling the transaction process in parallel using multiple vCPUs that could be considered as a bottleneck in system performance.

An experimental performance evaluation of two different versions of HLF (v0.6 and v1.0) was conducted by Nasir et al. in [26] to analyze the execution time, throughput, latency, and scalability through implementing various workloads to the system and node scales. The inclusive results showed that HLF

v1.0 consistently outperformed HLF v0.6 across all measured key performance metrics.

The study in [27] evaluated the blockchain latency, throughput, and scalability of the network with varying block sizes, peer CPU, SSD vs. RAM disk, and various peers. The obtained results identified that the end-to-end throughput of HLF v 1.1 could go up to 3500+ transactions per second (TPS) in certain widespread deployment configurations. Besides, the latencies were about a few hundred ms while scaling well to 100+ peers.

Authors in [28] conducted a study to find the performance bottlenecks of HLF v1.0 with varying performance metrics consisting of endorsement policies, different block sizes, number of channels, resource allocation, and state database choices (LevelDB vs. CouchDB). The experimental results listed the major system bottlenecks such as endorsement policy verification, sequential policy validation of transactions in a block, and state validate and commit transactions (with CouchDB). The authors outlined optimization solutions to overcome the aforementioned bottlenecks.

Geyer et al. [29] investigated the performance of HLF in terms of the underlying communication network using Caliper through the configuration of network parameters, including latency and packet loss in a dedicated testbed. The work examined the impact of transaction rate, network properties, block size, ChainCode, local network impairment, and block size. The experimental results indicated the transactions' validation as a significant contributor to the transaction latency in HLF.

In [30], the authors evaluated the performance of the HLF v1.4 platform's in terms of transaction throughput, latency, and scalability with various network workloads such as the number of transactions, transaction type, and transaction rate.

The impact of malicious behavior on the transaction throughput and latency of HLF performance was explored in the study conducted by Wang [31] with designed multiple malicious behavior patterns. The results showed significant degradation in the system performance due to the attacks' delays and due to keeping some replicas out of working.

Shi et al. [32] empirically investigated the performance of the Sawtooth platform. The blockchain platform performance was examined in terms of consistency, stability, and scalability with various workloads and configurations. The authors provided an adjustable optimization approach to configure parameters consists of the scheduler and maximum batches per block.

Implementing permissioned blockchains technology such as HLF on large-scale IoT systems is not a trivial task and is associated with a number of challenges. Unfortunately, it has been just partially covered by the existing research studies and technology practices. Aiming to address this deficiency, this work is reporting the results of performing a comprehensive experimental study focusing on the challenges associated with scalability and performance.

The scalability problem refers to the blockchain system handling an expanding number of peers while continuing to stay operational. The results presented in this study are based on the experimental study involving a large-scale practical setup. They highlight the blockchain network's scalability issue and demonstrate that it is significantly impacted by the

hardware configuration, blockchain system architecture, and complexity of smart contracts' operations. The performance issue was tackled through the extensive experimental study involving a large-scale distributed computing infrastructure (5 to 100 peers), measuring various performance metrics of a popular permissioned blockchain framework (HLF) under varying conditions.

Unfortunately, the above-cited and other earlier research reports did not provide detailed performance computation of real distributed large scale HLF implementation with different number of peers. Although they tried to configure the HLF platform with a different number of organizations, peers, etc., most consider the performance evaluation within a single host machine and not in a distributed environment. Also, the impact of network size on the fabric system's overall and detailed performance in different process stages were not investigated comprehensively. Moreover, the aforementioned research methods could not meet the low overhead requirement and have poor scalability. Motivated by this research gap, we conducted a comprehensive analysis of the HLF blockchain in a scalable and real distributed environment hosted by various virtual machines. The main objective is to compare the two different environments for implementing HLF, including Single-host and Multiple-Host. Addressing these challenges, we propose a detailed real-time performance computation and evaluation model for HLF blockchain systems.

III. HYPERLEDGER FABRIC

HLF was the first consortium blockchain platform established by the Linux Foundation [33]. HLF supports arbitrary smart contracts (known as ChainCodes [34]) implemented in general-purpose programming languages like Go, Java, and Nodejs. Other existing blockchain platforms deploy smart contracts written in a Domain-Specific Language (DSL) such as Ethereum's Solidity to avoid non-deterministic operations. Smart contracts enable a range of HLF applications across various industrial domains. Novel Execute-Order-Validate architecture for transactions and a pluggable consensus protocol differentiates the fabric from other blockchain platforms. The existing blockchain platforms use conventional order-execute architecture to facilitate the ordering of the transactions based on consensus protocol, and then each peer executes transactions in sequence order. The execution phase increases the network latency as the peers need to scrutinize all transactions in the block and execute them [27]. Key components in the fabric network are peer nodes, clients, and ordering service nodes belonging to different organizations. Each network entity in the fabric network has an identity assigned by a Membership Service Provider (MSP) [35].

Hyperledger Fabric is a permissioned blockchain platform where the network participants are exposed to each other and fully trusted. Nevertheless, the fabric network could be structured based on the governance model constructed, so trust exists between the participants. Blockchain applications are orchestrated and deployed based on participating organizations within the consortia. Nodes (or peers) host the blockchain and perform smart contract execution as well as mutually

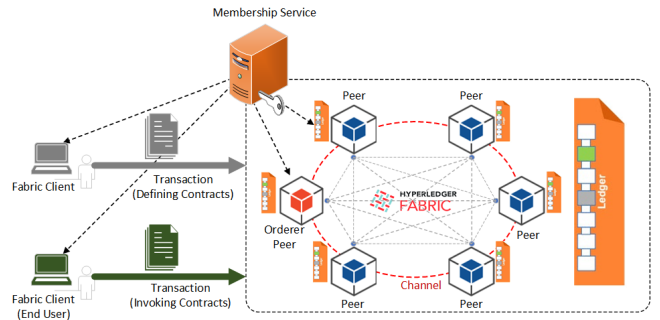


Fig. 1. HLF network system architecture with major components [27]

maintain the ledger's state. The concept of the channel in the fabric helps implement the shared ChainCode by all entities or develop private deployment. ChainCodes could be privately shared and deployed to a group of peers and is not accessible by other peers. The data and ChainCode are only available to the participants joining the same channel. The fabric network needs to authenticate and identify the peers through generated cryptographical materials. Therefore, particular channel members can be authenticated in this way. The ordering service (OS) performs the ordering of the accepted transactions by the fabric network on a per-channel basis. The overall Hyperledger Fabric system architecture is presented in Figure 1.

A. Transaction Flow in Hyperledger Fabric

Hyperledger Fabric employs the Execute-Order-Validate and Commit transaction model. Figure 2 shows the transaction flow in HFL blockchain platform that involves three steps: Endorsement, Ordering, and Validation phases. The transactions are ChainCode invocations' running on the Docker [36] container. Thus, it helps to separate them from other running ChainCodes on the same peer and the fabric codes. HLF is a distributed ledger technology where fabric peers keep a copy replica of the ledger. The ledger has two parts: the transaction log and state data. The transaction log keeps track of all the invoked transactions, while the state data represents a current state of the asset at any point in time. Various operations could be carried out on the state data by way of executing the ChainCode. ChainCode execution leads to creating transactions in the transaction log. It could also lead to changes in the state data. The transaction log is implemented using the levelDB that is a lightweight library for building a key-value data store embedded in fabric peer implementation. The state data consists of the key-value pairs that are versions. The state database is pluggable at the peer level. LevelDB supports a simple query for key-value pairs. However, it can be replaced with the CouchDB database, a NoSQL database that allows executing complex queries.

Participating organizations, their MSPs, and peers' identities need to be established before submitting any HLF network transactions. The channel needs to be initialized on the Orderer network with corresponding organization MSPs, and organization peers join the channel and initialize the ledger. Finally, the required ChainCodes need to be installed on the channel.

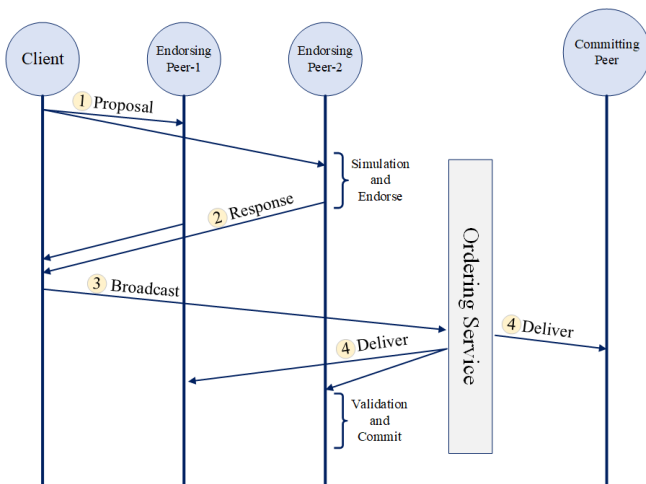


Fig. 2. Fabric transaction flow.

1) *Phase 1: Endorsement Phase* : Client applications using the fabric software development kit (SDK) create a request proposal and submit the transactions to endorsing peers based on the endorsement policy embedded in the ChainCode. The client puts a cryptographic signature to the proposal and sends it on the same channel where the ChainCode is deployed. The endorsing peers execute the proposed transactions and deterministic output received by corresponding endorsing. Nevertheless, all validity checks are carried on by endorsing peers before the execution ensures the client's authenticity and authority. The endorsement response includes the cryptographic materials, response value, and the read-write sets generated as a result of ChainCode execution. The endorsing peer signs the transaction response with the peer's identity through a system ChainCode called ESCC and sends a proposal response to the client. The ledger remains unchanged at this point. The client application collects endorsements from multiple endorsing peers, according to ChainCodes' defined endorsement policy, and sends them to the ordering service.

2) *Phase 2: Ordering Phase*: The ordering service performs transaction verifications. It orders them per channel. The next step is to deliver the ordered transactions by ordering service. All peers need to see the transaction in the same order known as block formed by ordering service and communicated to all peers. The ordering service deploys one of the developed ordering algorithms, such as SOLO, Kafka, or Raft [27]. Solo ordering algorithm consists of a single ordering service node (that controls all network ordering transactions), runs on one node, and is used for development. Kafka is a more distributed algorithm. It deploys the Kafka cluster to create and consume transactions. Thus, it provides crash fault tolerance (CFT) and is recommended in production deployments. Raft is similar to Kafka as it follows the leader-follower model. It has the advantage of being CFT. The setup of the Raft is rather straightforward. The ordering service can perform access control permissions of the client nodes to check whether they can broadcast or receive blocks on a particular channel.

3) *Phase 3: Validation Phase* : Once the block is sent to all peer nodes through either the ordering service provider or gos-

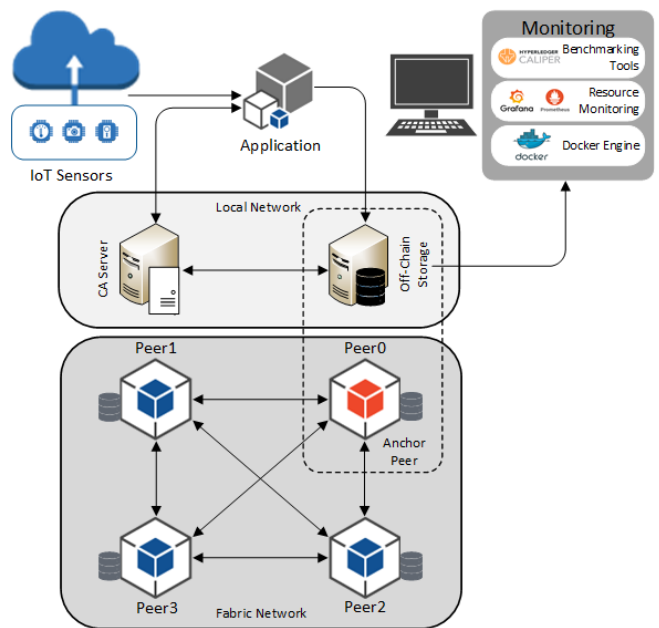


Fig. 3. HLF-based distributed system model (the peers and off-chain storage are separated into a scalable platform).

sip protocol (based on gRPC [37]), the transactions need to be validated. The validation process identifies and rejects invalid transactions. Therefore, only valid transactions are committed and updated in both ledger and world state. The validation phase consists of two consecutive steps: endorsement policy evaluation (using validation system ChainCode (VSCC)) and read-write conflict check (multi-version concurrency control (MVCC)). If there is a mismatch in versions, and the endorsement policy is not satisfied, the transactions are marked as invalid and their effects are omitted. Valid transactions are the ones that match the read set versions. The peer notifies the client about the success or failure of the transaction. The final step encompasses the ledger update. All the peers commit the identified valid transactions in the fabric network, and its write set is updated in the world state. Thus, in Fabric V1, each transaction goes through three phases: endorsement, ordering, and validation.

IV. CONFIGURATION PARAMETERS AND KEY METRICS

This work aims to study the performance of the scalable HLF in a distributed environment with a different number of nodes and various conditions to analyze how different parameters affect the system performance. We limit our study to a detailed analysis of a few parameters while other aspects are covered in general terms to understand and identify the interplay of network components. Therefore, the primary focus is on studying the overall performance from the peer's perspective. At the same time, the Orderer and Gossip effect on our experiment is eliminated as they have been kept static. The overall system under the test (SUT) and related components are presented in Figure 3. The shown model includes a single-channel HLF network with one client running benchmarking tools and one anchor peer.

A. Key Parameters Definition

Several key parameters are considered in this study. The first of them is a block size. An Orderer orchestrates transactions in batches. It then delivers them to peers in a block with the aid of the Gossip protocol. Each peer processes one block of received transactions at a time. The Orderer performs the cryptographical process per block to verify the Orderer signature, while the endorsement signature verification process is handled per transaction. The block size variation influences the throughput and latency. Therefore, we study the effect of various block sizes in conjunction with transaction sending rate. Note that we assume that all transactions are of the same complexity and independent of each other.

Endorsement policies play a vital role in controlling the number of executions of a transaction and signing transactions before submission to the Orderer. So, the transaction can successfully be validated by the VSCC phase. Validation confirms that a transaction's endorsements meet the endorsement policy for that ChainCode, i.e., read/write set does not conflict with simultaneous updates that were committed before. Time required for endorsement policy to collect and evaluate transactions is affected by its complexity and requiring more resources.

Channel provides an environment where a group of peers creates a separate transactions ledger accessible only by members. However, a peer can join multiple channels and therefore to maintain various ledgers. The channels process order, and deliver transactions independently, even though on the same peers. The number of the channels and their functionality directly impact system performance and scalability.

The routine verification process and signature computation by peers as a part of system ChainCodes need significant CPU and network resources. Besides, running user defined ChainCodes by endorsing peers during transaction submissions create extra loads on system. Our design considers a network having low latency and high bandwidth.

B. Performance Metrics

This paper considers a distribution ledger technology (DLT) system level to model HLF performance. The performance evaluation metrics document published by Hyperledger Performance and Scalability Working Group [38] provides precise performance metrics applicable across various DLT platforms. We use and refine their definitions for our experiments and analysis.

1) *Transaction Throughput*: Deployment, execution, and invoking of smart contracts in different blockchain systems occur at different speeds. We need to monitor the transaction throughput that is defined as a number of transactions per second (TPS) and indicates the rate of committing valid transactions by a fabric network in a defined period of time. For HLF network with a single channel, we consider the measurement at a single peer. However, we further extend our experiments to multiple peers (up to 100) in our model and experimental analysis. The formal mathematical description of transaction throughput can be obtained from the following formula:

$$TPS_i = \frac{\text{Count}(T_x \text{ in } (t_s, t_e))}{t_e - t_s} (\text{transactions/s}), \quad (1)$$

where, T_x is the total number of submitted transactions, t_e is the last block commit time, and t_s is the initial transaction submission time. The transaction throughput of N peers is calculated by taking the average:

$$\overline{TPS} = \frac{\sum_i TPS_i}{N} (\text{transactions/s}), \quad (2)$$

2) *Transaction Latency*: When the transaction is sent to the network, it takes some time to be confirmed by the system. Therefore, there is a gap between submitting a transaction, committed in a block, and being accepted by all peers. Transaction latency is the amount of time taken from the point the transaction is submitted to the point when the transaction is confirmed and committed with the result being available across the network. This metric is measured per transaction. However, in most cases, the experiment provides various statistics on overall transactions such as high, average, low, and standard deviations. We check the transaction confirmation at a single peer and multiple peers with various load levels in our analysis. The computed end-to-end latency consists of three components: endorsement latency, ordering latency, and commit latency [39]. During a period started at t_s and ended at t_e , the transaction sent to the peer is shown by Tx_{input} , and $Tx_{confirmed}$. It presents the confirmed transaction's transaction action. The average latency (AL) of the peer i can be computed by the following equation:

$$AL_i = \frac{\sum_{Tx} (t_{Tx_{confirmed}} - t_{Tx_{input}})}{\text{Count}(T_x \text{ in } (t_s, t_e))} (\text{transactions/s}), \quad (3)$$

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

$$\overline{AL} = \frac{\sum_i AL_i}{N} (\text{transactions/s}), \quad (4)$$

3) *Network Size and Scalability*: The implemented HLF network's ability to support increasing the number of participants is computed in this study. Network size indicates the number of validating peers participating in consensus in the System Under Test (SUT). Network size is presented to show the total number of nodes actively participating in the HLF blockchain network.

4) *Block Size*: Block size presents the number of transactions per block, and it is described by three variables: the maximum transaction count, absolute maximum byte, and preferred maximum bytes. The transactions are batched as a block. Our study further expands the analysis to include multiple blocks (10 blocks and 50 blocks) in batches. It also studies effects of different batch sizes on HLF systems.

C. Test Environment

The primary goal of our study is to benchmark the performance of distributed HLF implemented on multiple machines.

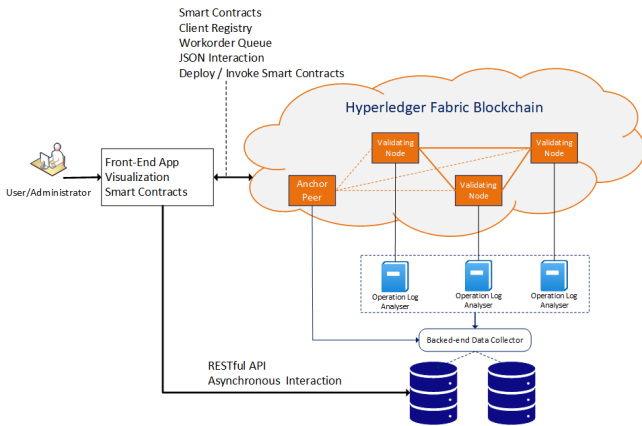


Fig. 4. Experimental setup and components for performance evaluation.

TABLE I
THE SYSTEM UNDER TEST PARAMETERS AND METRICS.

Parameters	Values
Transactions	1 KV write (1-w) of size 20 bytes
Channels	1 Channel
World StateDB	LevelDB
Peer Resources	Up to 100 vCPU, 3.3 GHz, 10 GiB, Low to Moderate Network Performance
Block Size	30 transactions per block
Batch Timeout	1000 ms
Tx Sending Rate	5 – 500 (tps)
Number of Blocks	10, 50

Therefore, we conduct an in-depth study of HLF core components and benchmark fabric performance for IoT applications. The throughput and latency of the system under test are computed by varying configuration parameters listed in section IV-A and IV-B.

Figure 4 depicts the experimental setup model that was used in all our experiments. A permissioned HLF network was set up where with one organization that includes several peers in each scenario. The ordering service is run on a separate node, and a single channel was implemented. The ChainCode was deployed on the channel to facilitate the assigned tasks.

Our setup deployed a private HLF blockchain network in a controlled distributed environment. To achieve realistic results, we deployed several Amazon AWS EC2 instances as an underlying network of nodes. Their parameters are given in Table I. Each instance runs on its Virtual Machine (VM). All VMs belong to the same subnetwork to diminish the effect of network latencies within the experiments. We conducted the same investigation several times with different values of peers and nodes. KV is stand for Key Value (KV), representing a value as a transaction to be sent to the blockchain network.

We deployed the Hyperledger Fabric (version 1.4) blockchain framework to run the blockchain application. It is an open-source permissioned blockchain platform designed for enterprise applications. Virtual machine instances host Hyperledger Caliper [25], a benchmark tool to measure multiple blockchain performance. The Caliper also runs on client and monitoring instances to broadcast transactions on the HLF channel. The network consisted of numerous peers (from 5

TABLE II
NETWORK NODES AND LOAD SIZES.

Parameters	Values
Transactions Sending Rate	10, 20, 30, ...,100, ..., 500 (tps)
Number of Peers	5, 10, 20, ..., 100
Block Size	10, 50

peers per organization and up to the maximum of 100 peers) that are run on scalable network infrastructure. The blockchain components were deployed as a Docker container. Docker Swarm was used to orchestrate and manage the containers spread across the network of VMs. All nodes had the Ubuntu 18.04 LTS operating system.

We deploy the Hyperledger Caliper, as the standard open-source benchmarking tool recommended by the Hyperledger community. Further analysis on collected log files and data have been conducted using Microsoft PowerBI [40] and Origin Pro [41]. Besides, to monitor Hyperledger Fabric Docker Containers, we used Prometheus & Grafana [42].

The Proof of Work (PoW) consensus shows its robustness and, due to its pseudo-anonymous nature, it has been considered the most secure option for crypto currencies applications. However, in the enterprise ecosystems such as IoT networks and telecom environments, they appear redundant as the blockchain participants are already known to each other. Therefore, permissioned blockchains are designed for enterprise systems that use more straightforward and less resource-consuming consensus protocols, such as Raft [39]. Hence, we consider implementing the Raft consensus protocol in this study.

V. RESULTS AND DISCUSSIONS

This section presents the impact of various key metrics parameters on the performance of the Fabric network. The throughput and transaction latency shown here have resulted from multiple benchmarking runs. The averages of various runs also have been computed.

A. Impact of Transaction Sending Rate - Single Host

Figure 5 plots the average throughput for various block sizes over different transaction sending rates in a single-host setup. Figure 6 presents the average latency over the same transaction sending rates. Table II lists the multiple parameters used in this experiment including the transaction sending rate, block size, and peers. Each experiment started with sending transactions from 10 tps up to 500 tps.

The average latency remained below 1 second throughout the experiments till it reached around 100 tps. The throughput scaled linearly with the increase in the transaction sending rate, and it flattened out at about 100 tps showing the highest usable rate of the system. When the load was increased beyond the peak point, the performance started to degrade. Additionally, when the number of ordered transactions waiting in the verification process queue by VSCC during the validation phase increased, it significantly increased the commit latency. Therefore, a validation phase can be considered as a bottleneck

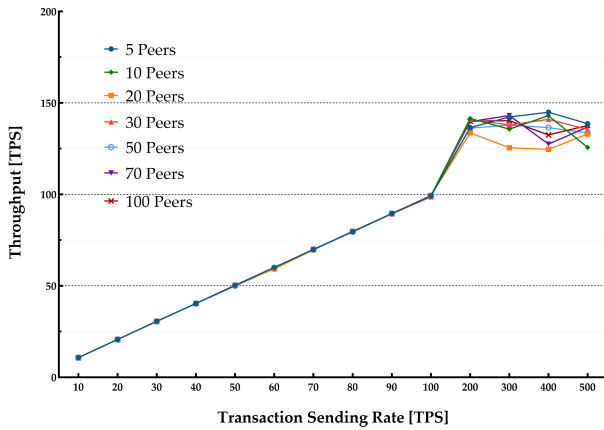


Fig. 5. Transactions sending rate vs throughput – single Host.

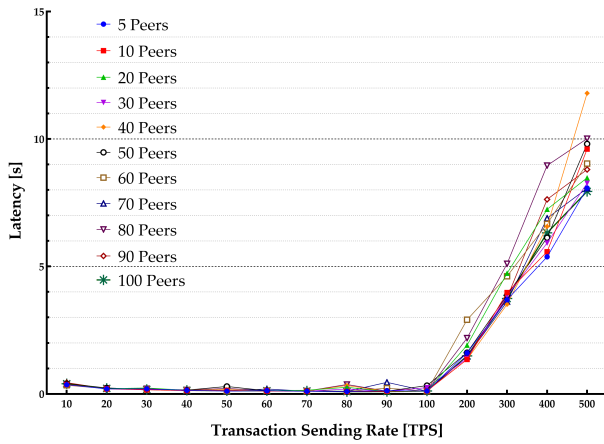


Fig. 6. Transactions sending rate vs latency- single Host.

in the blockchain system thus causing a significant delay. However, the blockchain system depends on SUT hardware capabilities. Besides, the increase in the number of involved peers also increases the latency.

Hyperledger Fabric relies on Docker-based architecture, and all the components of your hyperledger network run in separate containers with no visibility of the neighboring containers. To make them communicate, they create a network, and each container attaches itself to it. It can be found in the docker-compose-cli.yml. In a single host scenario, all containers run on a single machine, while the multi-host setup includes various nodes that start from 1 to 100, but the measurements are done in few steps. By default, Compose sets up a single network for your app.

Furthermore, during the experiments, we observed an increase in the transaction sending rate, leading to a small increase in average CPU utilization by about 10%. The CPU was mainly used during the validation phase of a block by VSCC. It was experimentally found that, for real applications (such as IoT) to achieve lower transaction latency, we needed to use smaller block sizes with low transaction rates. In contrast, higher transaction rates needed larger block sizes to

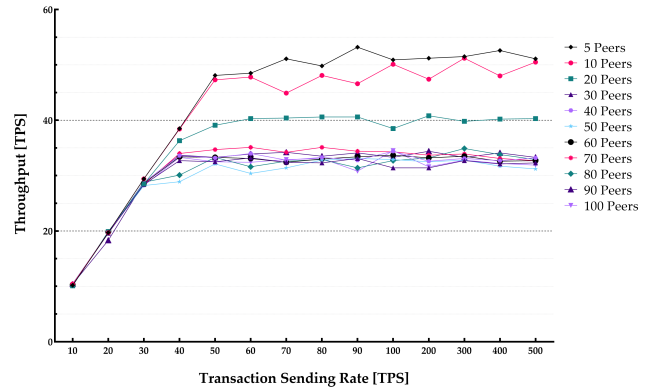


Fig. 7. Transactions sending rate vs throughput for multiple host arrangement.

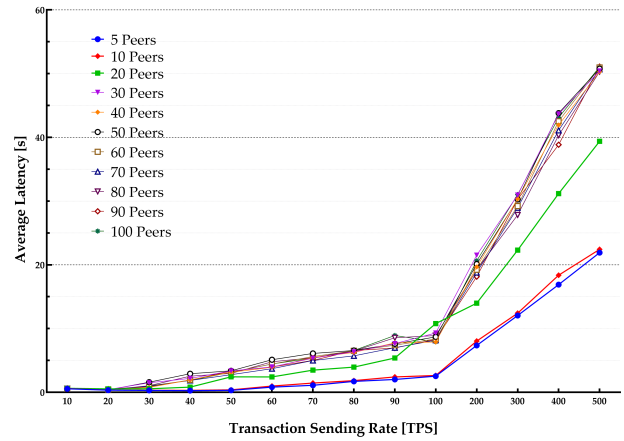


Fig. 8. Transactions sending rate vs latency for multiple host arrangement.

get higher throughput and lower transaction latency.

B. Impact of Transaction Sending Rate - Multiple Host

Participating entities run peers within the consortium. For larger consortiums, each organization is considered a partner; preferably, they would run at least one peer to contribute to the network. To achieve the real-world production environment, we deployed an overlay network of nodes and implemented the peer nodes in a distributed system. In that setup, we explored the effect on system performance when consortiums' size was growing. The endorsement policy was configured to help peers endorsing a transaction on a single channel setup within the consortium and to include the transaction output on the blockchain.

Each peer was installed separately in a VM hosted on a commercial cloud provider infrastructure and contributed to the HLF network. The experiment consisted of multiple benchmark rounds with changing transaction sending rates (from 10 tps to 500 tps). We generated various transactions for each benchmark run and submitted them to the HLF network to compute the maximum, average, and minimum transaction latency, and transaction throughput. Besides, we run the experiments with various block sizes in each section

(10 Blocks and 50 Blocks). Figures 7 and 8 show the latency and throughput measurements for all workloads.

Our observation can lead to a conclusion that for a lower transaction rate of 100 tps, more involved peers yield a lower throughput and incur higher latencies when the consortium size increases. The results are also slightly different for various block sizes, showing that changing the block from 10 to 50 results in higher latency and lower throughput. The throughput increased linearly as it can be predicted with an increase in the transaction sending rate until it flattened out at the saturation point, as shown in Figure 8. The results indicate that an increase in the number of peers leads to reaching a lower transaction sending rate peak point. The results for latencies were as predicted.

C. Impact of Endorsement Policy

The first experiments considered the ChainCode endorsement policy where peers run on a single host and one peer from the organization endorsed all transactions. Such a representation is for a basic ChainCode implementation between various entities where the organizations have the same authority to control the incoming requests. However, we analyzed the case when the clients sent the transactions to a number of endorsers with response coming from multiple endorsing peers. Hence, we studied the impact of different endorsement policies on the average latency and throughput with varying sending rates. We configured the endorsement policy such that a single peer from an organization endorsed the transactions. Then we expanded the experiments by involving more peers endorsing the transactions.

Figures 9 and 10 present the resulting throughput and average latency measurements. The results show that both cases have almost the same latency trend till they reach the peak point. Beyond the peak point, the transaction latency with a single host endorsement indicates a better performance. The throughput increases linearly in all the cases, albeit with different peak points for different configurations. The throughput gets flatten after reaching the peak point, which is not the same for different metric configurations and network setups. The bottleneck can be seen as this version of HLF does not utilize all available CPU cores within participating peers to commit transactions in parallel. Therefore, with higher sending rates, the throughput plateaus to the maximum value that the system can offer.

D. Impact of Block Size - Multiple Host

The block size dictates the number of transactions batched in a block at the Orderer and delivered to peers through gossip protocol. The ordering service controls the creation of blocks from the transactions using various parameters such as BatchSize and Batch Timeout. The BatchTimeout indicates the amount of the Orderer waiting time before creating a block regardless of how many transactions are included. We analyzed the effect of varying block size on the throughput and latency with different transaction sending rates.

There is a slight increase in the latency for a block size as the transaction sending rate increases closer to the peak

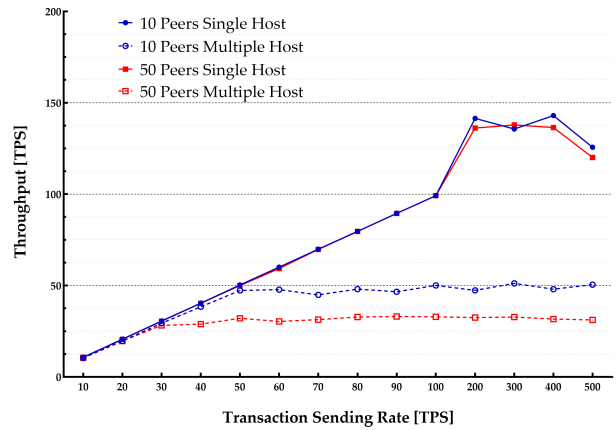


Fig. 9. Transactions sending rate vs throughput for various endorsement policies.

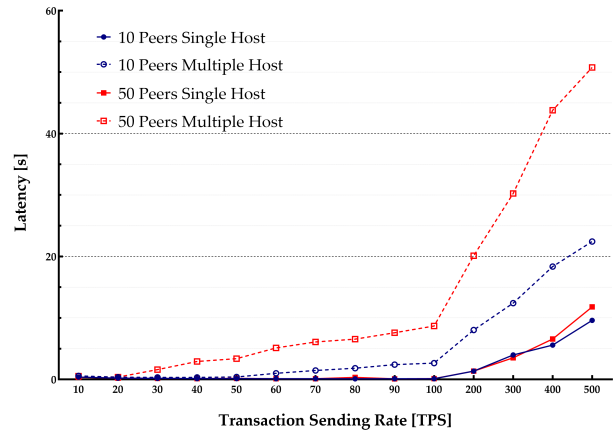


Fig. 10. Transactions sending rate vs latency for various endorsement policies.

point. For smaller block sizes and higher sending rates, blocks are generated faster before the block timeout. Therefore, transaction waiting time decreases at the ordering service node. The increase in the transaction sending rate means more transactions in a block, and so the time taken by the validation and commit phase is increased accordingly.

The results shown in figures 11 and 12 indicate that performance optimization can be achieved with an increase in block sizes. Therefore, the Orderer BatchSize can significantly influence the system throughput. Besides, the results show that smaller block numbers reduce the throughput. This metric can be specified during the Orderer bootstrap and can be dynamically altered based on various system applications and the total system load. With the increase in the number of peers, the larger block sizes show less impact on the throughput. Additionally, having transactions with a smaller BatchSize diminishes throughput as a greater number of blocks and block events are required to be generated.

This fact suggests that when the transaction sending rate is expected to be lower than the peak point, we need to use a smaller block size to achieve a lower transaction latency for

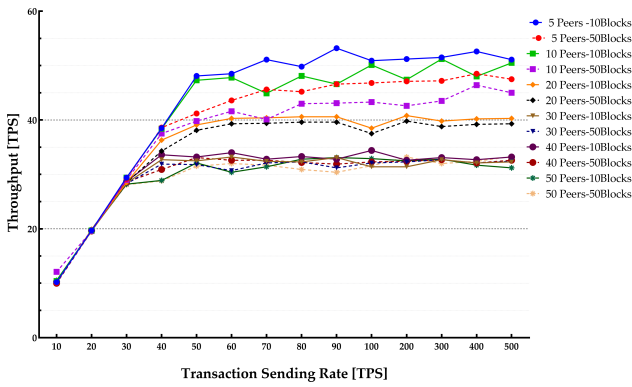


Fig. 11. Impact of block sizes on system throughput.

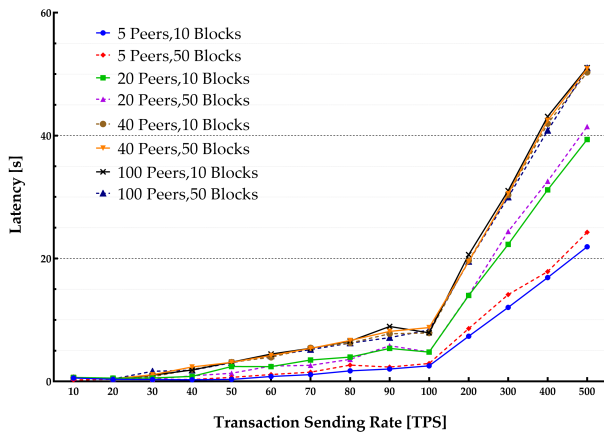


Fig. 12. Impact of block sizes on transactions latency.

applications. On the other hand, when the transaction sending rate is predicted to be high, we need to employ a larger block size to gain a higher throughput and a lower latency.

E. Impact of Network Size

In this experiment, we studied the impact of the network size increasing and including more peers. Our investigations included one organization and one Orderer service node. All transactions were directed to the same Orderer for the validation process. The number of channels remained the same as in the previous experiments, and we run all ChainCodes on one channel. The membership provider was responsible for the permission of entities within the same organization. The main goal was to study the effect of peer node number increase on the total throughput and average latency.

We studied the average latency and throughput for network sizes having from 5 to 100 peers. The endorsement policy setting consisted of a single channel and single Orderer within the consortium setup to perform endorsement of a transaction to be stored on the blockchain. Figures 13 and 14 show the throughput and average latencies for transactions with varying send rates up to 2500 tps.

We observed that for a lower transaction rate below 100 tps, an increase in the number of peers yielded a lower throughput

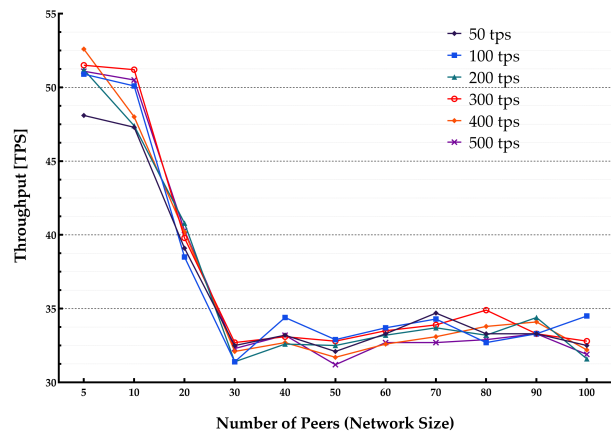


Fig. 13. Impact of network size on the system throughput.

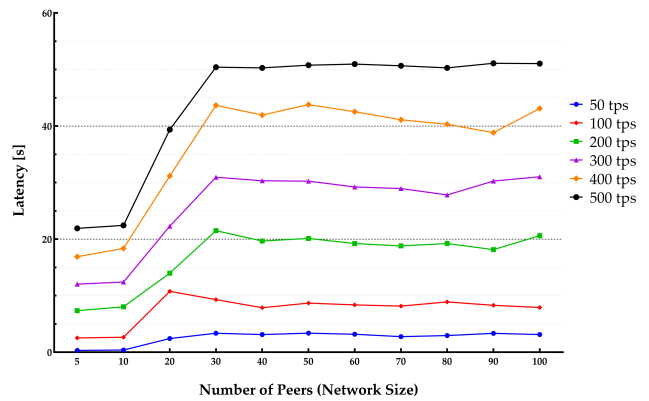


Fig. 14. Impact of network size on transactions latency.

and higher latency when the network size increased. The results slightly varied for different block sizes, showing that changing the block from 10 to 50 results in higher latency and lower throughput.

F. Resource Consumption

In this part, we studied the various system resource consumption, including CPU, Memory, Disk, and Network Traffic I/O. Measurements of the blockchain system resource consumption are crucial for blockchain users or managers by providing them with an overview of the blockchain system performance. Note that the major part of CPU resource consumption is accrued during the execution of chain codes. The amount of CPU consumption is mainly affected by the business logic implemented in the contract. Complex contracts including encryption and loops normally consume more CPU resources. Besides committing the block, the hash of the world state computation also consumes much of CPU resources.

Memory consumption occurs when the virtual machine or docker loads the account data from the world state during the contract execution and opens up some arrays. The hard disk storage space is separated by the blockchain program for storing data, including a world state. Therefore, it uses I/O

resources when maintaining the blockchain operations such as block committing and contract execution. Keeping every peer in the same state within different blockchain systems is supported by deploying a different consensus protocol. The consensus protocol performs appending transactions in the network and transfers the block data while consuming the network flow. The experiment encompassed sending multiple transaction batches to calculate the metrics mentioned above. The results are presented in Figures 15-19.

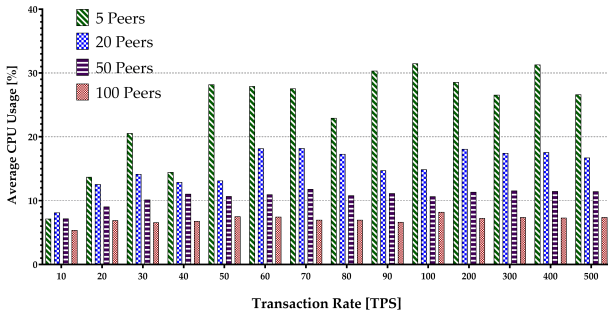


Fig. 15. Peers average CPU usage.

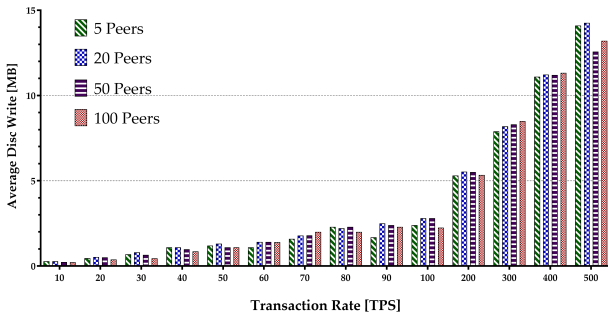


Fig. 16. Peers average disk write usage.

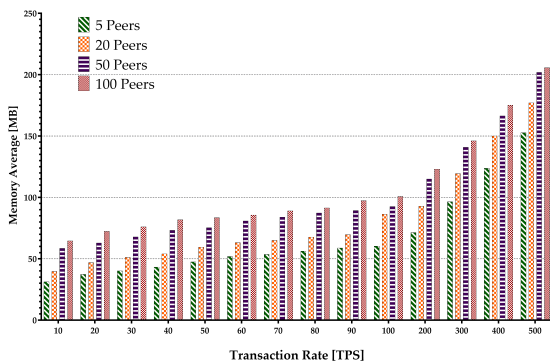


Fig. 17. Peers average memory consumption.

Figure 15 depicts the average peer CPU utilization. With the increase in the number of the network peers, more CPUs are involved, and less load is imposed on individual CPU in general. This result in the decrease in an average CPU

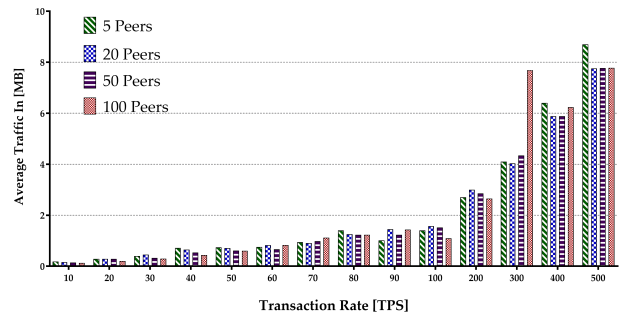


Fig. 18. Peers average network traffic In.

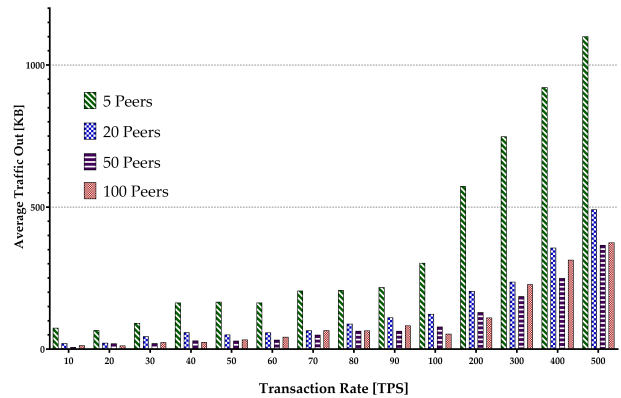


Fig. 19. Peers average network traffic Out.

utilization. Additionally, the growth in batch sizes increases the CPU usage until it flattens around the peak. The average disk write consumption shown in Figure 16, indicating the linear growths with both the number of peers and the batch sizes. Figure 17 shows similar patterns of about average memory utilization by peers in the network. However, the figure indicates that the maximum batch size suitable for the system is around 100 tps. The network In/Out traffic is presented in Figures 18 and 19, respectively. The average In traffic increases with the increase in the number of peers and batch sizes. However, the Out traffic shows a higher growth with increased batch sizes, with fewer peers involved in the blockchain system.

G. Summary of Performance Analysis

For analysing the scalability and performance of the Fabric platform in a distributed architecture, we evaluated the scenarios where there are multiple hosts in the network (from 5 hosts to 100 hosts). We conducted experiments of the same configuration on a single host environment. The Fabric introduced the multiple organization concept in Fabric v1.0. Fabric v1.4 is applied to a multi-host network of cloud-based instances and analyzed by measuring the system performance metrics consisting of throughput, latency, network size, resource consumption, and endorsement policies. The results are compared with single-host deployment. The single host deployment shows better performance for most of the metrics,

but this scenario is not applicable in production. However, the performance of multi-host schemes can be increased with multi-organization deployment in which each organization has its dedicated Orderer.

For the single-host, the HLF platform can handle a maximum of 25000 transactions. However, as shown in the results, the execution time is decreased when the number of transactions is more than 100 transactions resulting in a decrease in the throughput and the latency. Single host implementation has higher throughput and lower latency compares to multiple host deployment. For the multiple-host, regardless of the number of peers in the network, the HLF network can handle around 25000 transactions. However, the number of concurrent transactions is limited and highly dependent on the number of peers in the HLF network. The execution time is higher than the single-host deployment, resulting in docker deployment in the large scale network. The throughput is lower, and the latency is higher compare to the single host deployment. Fabric cannot instantiate more than 20 endorsing peers on local machines in both scenarios and need to be instantiated manually.

VI. CONCLUSION

This paper presented a detailed experimental performance analysis for the scalable Hyperledger Fabric blockchain platform in a distributed large network, with varying numbers of nodes and workloads. We proposed a scalable framework for precise and real-time monitoring of HLF systems. It offers significantly lower overhead and more details about the various HLF system metrics compared with previous approaches. We conducted a comprehensive performance analysis and evaluation of the well-known HLF blockchain systems with different network configurations, network load levels, node numbers, and batch sizes. The system performance evaluation is shown in terms of throughput, latency, block size, network size, endorsement policy, CPU usage, memory consumption, disk write, In/Out traffic and scalability. The experimental results indicate the feasibility of the proposed framework. However, the throughput, latency, and scalability of a blockchain framework depend on hardware configuration, blockchain network design, and smart contract complexity operations.

The throughput of the system linearly increases below the transaction rate of about 100 tps for single-host configuration, and around 50 tps for multi-host. After the peak point, the throughput degrades, and transaction latencies increase showing that the system throughput is sensitive to the Orderer setting. The results also indicate that the change in the number of endorsements influences the performance metrics significantly. The smaller number of endorsements results in better performance. Unfortunately, this also brings security vulnerabilities to the system due to weakening its anti-collision properties. One of the main bottlenecks is how the committing peers utilize multiple vCPUs presented in the system to do parallel transactions that need to be optimized to improve the system performance. The size of transactions significantly impacts throughput and transaction latencies. The system scalability demonstrates promising results. A more extensive network size

needs to consider the optimized number of endorsements per ChainCode to a smaller subset of peers to achieve a better performance.

Future work will consider implementing the updated version of HLF and evaluating the use of real-time transactional data, as well as exploring more test cases, such as analyzing the impact of having multiple Orderers on the overall system performance. Besides, other system configurations such as multiple fabric organizations and the increase in the number of endorsement peers may be further explored.

REFERENCES

- [1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.
- [2] A. Boudguiga, N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey, "Towards better availability and accountability for iot updates by means of a blockchain," in *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2017, pp. 50–58.
- [3] H. Honar Pajoo, M. Rashid, F. Alam, and S. Demidenko, "Hyperledger fabric blockchain for securing the edge internet of things," *Sensors*, vol. 21, no. 2, p. 359, 2021.
- [4] —, "Multi-layer blockchain-based security architecture for internet of things," *Sensors*, vol. 21, no. 3, p. 772, 2021.
- [5] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, vol. 107, pp. 841–853, 2020.
- [6] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.
- [7] L. Chettri and R. Bera, "A comprehensive survey on internet of things (iot) toward 5g wireless systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2019.
- [8] Y. Guo and C. Liang, "Blockchain application and outlook in the banking industry," *Financial Innovation*, vol. 2, no. 1, pp. 1–12, 2016.
- [9] F. Jamil, S. Ahmad, N. Iqbal, and D.-H. Kim, "Towards a remote monitoring of patient vital signs based on iot-based blockchain integrity management platforms in smart hospitals," *Sensors*, vol. 20, no. 8, p. 2195, 2020.
- [10] Z. Shahbazi and Y.-C. Byun, "Integration of blockchain, iot and machine learning for multistage quality control and enhancing security in smart manufacturing," *Sensors*, vol. 21, no. 4, p. 1467, 2021.
- [11] —, "A procedure for tracing supply chains for perishable food based on blockchain, machine learning and fuzzy logic," *Electronics*, vol. 10, no. 1, p. 41, 2021.
- [12] F. Jamil, N. Iqbal, S. Ahmad, D. Kim *et al.*, "Peer-to-peer energy trading mechanism based on blockchain and machine learning for sustainable electrical power supply in smart grid," *IEEE Access*, vol. 9, pp. 39 193–39 217, 2021.
- [13] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE ACCESS*, vol. 4, pp. 2292–2303, 2016.
- [14] M. Pilkington, "Blockchain technology: principles and applications," in *Research handbook on digital transformations*. Edward Elgar Publishing, 2016.
- [15] C. Dannen, *Introducing Ethereum and solidity*. Springer, 2017, vol. 318.
- [16] Hyperledger fabric. Accessed: 2021-January-15. [Online]. Available: <https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html>.
- [17] M. Swan, *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc., 2015.
- [18] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 1085–1100.
- [19] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance evaluation of blockchain systems: A systematic survey," *IEEE Access*, vol. 8, pp. 126 927–126 950, 2020.
- [20] M. Dabbagh, K.-K. R. Choo, A. Beheshti, M. Tahir, and N. S. Safa, "A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities," *Computers & Security*, vol. 100, p. 102078, 2021.

- [21] M. Dabbagh, M. Kakavand, M. Tahir, and A. Amphawan, "Performance analysis of blockchain platforms: Empirical evaluation of hyperledger fabric and ethereum," in *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAJET)*. IEEE, 2020, pp. 1–6.
- [22] J. Yli-Huomo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology?—a systematic review," *PLoS one*, vol. 11, no. 10, p. e0163477, 2016.
- [23] Smart contracts. Accessed: 2020-November-15. [Online]. Available: <https://hyperledger-fabric.readthedocs.io/en/release-1.4/smartcontract/smartcontract.html>.
- [24] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat, and S. Chatterjee, "Performance characterization of hyperledger fabric," in *2018 Crypto Valley conference on blockchain technology (CVCBT)*. IEEE, 2018, pp. 65–74.
- [25] H. S. W. Group. Hyperledger blockchain performance metrics. Accessed:2020-November-10. [Online]. Available: <https://www.hyperledger.org/wpcontent/uploads/2018/10/HL-Whitepaper-Metrics-PDF-V1>
- [26] Q. Nasir, I. A. Qasse, M. Abu Talib, and A. B. Nassif, "Performance analysis of hyperledger fabric platforms," *Security and Communication Networks*, vol. 2018, 2018.
- [27] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Eneyart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.
- [28] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2018, pp. 264–276.
- [29] F. Geyer, H. Kinkelin, H. Leppelsack, S. Liebold, D. Scholz, G. Carle, and D. Schupke, "Performance perspective on private distributed ledger technologies for industrial networks," in *2019 International Conference on Networked Systems (NetSys)*. IEEE, 2019, pp. 1–8.
- [30] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: throughput, latency and scalability," in *2019 IEEE international conference on blockchain (Blockchain)*. IEEE, 2019, pp. 536–540.
- [31] S. Wang, "Performance evaluation of hyperledger fabric with malicious behavior," in *International Conference on Blockchain*. Springer, 2019, pp. 211–219.
- [32] Z. Shi, H. Zhou, Y. Hu, S. Jayachander, C. de Laat, and Z. Zhao, "Operating permissioned blockchain in clouds: A performance study of hyperledger sawtooth," in *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE, 2019, pp. 50–57.
- [33] L. Foundation, "Hyperledger white paper," *Hyperledger*, vol. v2.0.0., p. 1–19, 2016, accessed: 2021-January-10. [Online]. Available: <https://github.com/hyperledger/hyperledger/wiki/Whitepaper-WG>.
- [34] What is chaincode? Accessed: 2021-January-30. [Online]. Available: <https://hyperledger-fabric.readthedocs.io/en/release-1.4/chaincode.html>.
- [35] Membership Service Providers (MSP). Accessed: 2020-November-10. [Online]. Available: <https://hyperledger-fabric.readthedocs.io/en/release-1.4/membership/membership.html>.
- [36] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.
- [37] Why grpc? Accessed: 2021-January-10. [Online]. Available: <http://grpc.io/>
- [38] "W. t. h. fabric. <http://hyperledger-fabric.readthedocs.io/en/release-1.4/>." [Online]. Available: <http://hyperledger-fabric.readthedocs.io/en/release-1.4/>
- [39] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, 2014, pp. 305–319.
- [40] B. Power, "Microsoft powerbi platform," 2020.
- [41] Originpro. Accessed: 2021-January-30. [Online]. Available: <https://www.originlab.com/>.
- [42] P. Authors, "Prometheus-monitoring system & time series database," 2017.



Houshyar Honar Pajooh (StudentM'16) received his ME (Hons) from the University of Auckland, New Zealand in Electrical & Electronic Engineering in 2016. He is currently pursuing a Ph.D. at Massey University. His research interests include Blockchain, network security and the internet of things.



Systems.

Mohammad A. Rashid (M'16-SM'19) is a Senior Lecturer at the Department of Mechanical & Electrical Engineering, Massey University, New Zealand. He received MSc Eng degree in Electronics Engineering specialising in Engineering Cybernetics Systems from the Wroclaw University of Science and Technology in 1978 and PhD from the University of Strathclyde, UK in 1986. His research interests include Embedded Systems, Internet of Things, Network Access Protocols, IoT Security and Blockchain, Big Data Analytics, ERP/Enterprise



Smart Sensors and Precision Instrumentation by leveraging his expertise in Wireless and Visible Light Communication, IoT and Signal Processing. A/Professor Alam is an Associate Editor of IEEE Access and sits on the IEEE Conference Technical Program Integrity Committee (TPIC). He is also a member of the Institution of Engineering & Technology (IET) and the Association for Computing Machinery (ACM).



Serge Demidenko (M'91-SM'94-F'04) is Professor and Dean of the School of Science and Technology, Sunway University, Malaysia. He is also associated with the Department of Mechanical & Electrical Engineering, School of Food and Advanced Technology, Massey University, New Zealand as a postgraduate supervisor. He graduated in Computer Engineering from the Belarusian State University of Informatics and Radio Electronics, and received Ph.D. from the Institute of Engineering Cybernetics of the Belarusian Academy of Sciences.

Chapter 6

IoT Big Data Provenance Scheme Using Blockchain on Hadoop Ecosystem

Chapter 6 includes the article, “IoT Big Data provenance scheme using blockchain on Hadoop ecosystem” published in the Journal of Big Data. This article is open access and has been republished in this thesis under the Creative Commons Attribution License 4.0. Authors of articles published in Journal of Big Data retain the copyright of their articles and are free to reproduce and disseminate their work.

The full text is included in the thesis has some minor modifications. This means that while the content is identical to the published article, there may be stylistic differences.

RESEARCH

Open Access



IoT Big Data provenance scheme using blockchain on Hadoop ecosystem

Houshyar Honar Pajooh^{1*} , Mohammed A. Rashid¹, Fakhurul Alam^{1,2} and Serge Demidenko^{1,2}

*Correspondence:

h.pajooh@massey.ac.nz

¹ Department of Mechanical & Electrical Engineering, School of Food and Advanced Technology, Massey University, Auckland 0632, New Zealand
Full list of author information is available at the end of the article

Abstract

The diversity and sheer increase in the number of connected Internet of Things (IoT) devices have brought significant concerns associated with storing and protecting a large volume of IoT data. Storage volume requirements and computational costs are continuously rising in the conventional cloud-centric IoT structures. Besides, dependencies of the centralized server solution impose significant trust issues and make it vulnerable to security risks. In this paper, a layer-based distributed data storage design and implementation of a blockchain-enabled large-scale IoT system are proposed. It has been developed to mitigate the above-mentioned challenges by using the Hyperledger Fabric (HLF) platform for distributed ledger solutions. The need for a centralized server and a third-party auditor was eliminated by leveraging HLF peers performing transaction verifications and records audits in a big data system with the help of blockchain technology. The HLF blockchain facilitates storing the lightweight verification tags on the blockchain ledger. In contrast, the actual metadata are stored in the off-chain big data system to reduce the communication overheads and enhance data integrity. Additionally, a prototype has been implemented on embedded hardware showing the feasibility of deploying the proposed solution in IoT edge computing and big data ecosystems. Finally, experiments have been conducted to evaluate the performance of the proposed scheme in terms of its throughput, latency, communication, and computation costs. The obtained results have indicated the feasibility of the proposed solution to retrieve and store the provenance of large-scale IoT data within the Big Data ecosystem using the HLF blockchain. The experimental results show the throughput of about 600 transactions, 500 ms average response time, about 2–3% of the CPU consumption at the peer process and approximately 10–20% at the client node. The minimum latency remained below 1 s however, there is an increase in the maximum latency when the sending rate reached around 200 transactions per second (TPS).

Keywords: Internet of Things, Hyperledger fabric, Blockchain, Big Data, Data provenance, Hadoop, Traceability

Introduction

Over the past decades, data generated by the massive implementation and use of the *Internet of Things (IoT)* have been growing exponentially. The global Big Data market size has been projected to grow from USD 138.9 billion in 2020 to USD 229.4 billion by 2025 [1]. This unprecedented increase of data acquisition across many fields [2] (such

as healthcare, manufacturing, retail, logistics, transportation, etc.) allows for gaining meaningful in-depth insights. The extraction of meaningful insights from Big Data (e.g., volume, velocity, and representation) require a robust structure to facilitate the data storage, analysis, and processing in a secure, distributed, and scalable manner [3]. Big Data Analytics is an emerging field dealing with processing and analyzing vast data volumes [4]. The tremendous increase of data volumes within the Big Data ecosystems requires a robust solution to ensure information integrity so the correct knowledge can be derived from the analysis.

Blockchain offers a promising architecture for distributed large data storage and protection. A group of nodes and users within a blockchain network works cooperatively to structure the public ledger that contains the validated and recorded transactions as blocks. The data in IoT applications can be stored in off-chain storage (Big Data systems) while the data pointer to the off-chain storage can be kept in the blockchain system. When a data entity from the big data system is required, the blockchain accesses the specific data entity through a trusted environment. The user authentication is maintained by the distributed blockchain miners instead of a third-party auditor or a trusted centralized server. This study considers the decentralized storage for IoT data as off-chain Big Data system in a distributed manner while an entity can easily locate the address through the blockchain system. The third-party auditor and centralized trusted server are eliminated and the access to IoT data is managed by the blockchain nodes. The blockchain also manages the authentication of the users. The proposed work provides the accountability and tractability of IoT data where activities such as data modification and data access can be recorded in the blockchain.

Various approaches have been put forward to implement blockchain in several real-world applications. The Secured Map Reduce (SMR) is a security and privacy layer between HDFS and MR Layer (Map Reduce) introduced in [5]. The research work promotes data sharing for knowledge mining and address the scalability issues of privacy. The state-of-the-art security and privacy challenges in big data as applied to healthcare industry is reviewed in [6], The research work explores the security and privacy issues of big healthcare data and discussed ways in which they may be addressed. A permissioned blockchain is deployed for Halal supply chain to maintain secure transactions where the proposed model considers the transaction speed and rate to transfer data in effective manner [7].

Big Data analytics operating on cloud-based systems has been exploited widely. It has become the technology norm in extracting data-driven knowledge. The existing sensor-based IoT ecosystems (formed from integrating cloud-based Big Data analytics and wireless technologies) span a broad range of applications such as smart homes, smart cities, smart healthcare, etc. However, practical integration of IoT and Big Data systems face many issues such as security and privacy, non-interoperability, scalability, data traceability, and management. This hinders the true potential of such systems. Security concerns associated with data privacy, integrity, safety mechanisms, and quality could negatively affect Big Data systems applications [8]. The existing solutions fail to address the Big Data auditing challenges in cloud platforms efficiently. This could lead to security issues in computing-based Big Data storage. The multi-layer blockchain paves the way to address the privacy and security of IoT through a layer-based structure [9].

Local authentication and authorization are deployed to ensure the security of small IoT devices. *Hyperledger Fabric (HLF)* blockchain platform is a feasible approach to address the security and privacy challenges of edge computing devices within the IoT ecosystem. It can also provide the data provenance for generating data from IoT devices within the HLF and off-chain storage [10]. The details of implementing a layer-based blockchain model in an IoT environment including mathematical modelling and assumptions are described in our previous works [9, 10] along with the implementation of lightweight authentication mechanism for constrained IoT devices within the blockchain platform.

The most critical challenges and issues associated with various Big Data applications and techniques are security and privacy, infrastructure scalability, data interpretation, intelligence, real-time data processing, and data management. Among them, security and privacy are considered to be the most important [11]. The verification and integrity of user data within an untrustworthy infrastructure provided by a *cloud service provider (CSP)* is another critical challenge. Big Data characteristics consisting of variety, volume, and veracity raise concerns about efficient Big Data security mechanisms [12]. The aforementioned concerns and issues require investigating a robust mechanism that can verify the integrity of the outsourced data for Big Data storage in the cloud environment.

The majority of existing solutions incorporate *third-party auditor (TPA)* programs to maintain data integrity based on log files. This process increases the required storage size as well as communication and computation overheads. At the same time, it also brings many security concerns. Various solutions and practices have been explored to preserve data confidentiality and provide information security. In recent times the blockchain technology has received significant attention from many researchers as a promising solution to provide security and privacy in Big Data systems. The blockchain is defined as a number of nodes joined in a peer-to-peer manner maintained by the *distributed ledger technology (DLT)*. The study [13] considers the blockchain to enable efficient data collection and secure data sharing in a reliable and safe industrial IoT environment. The integration of blockchain with edge computing servers facilitates the security of the data collection process from IoT devices and the integrity of collected data [14]. Blockchain provides a robust structure for efficient and secure data collection in mobile ad-hoc networks [15]. Besides, the blockchain framework ensures data immutability, non-repudiation services, and network management capabilities. The decentralized architecture of the blockchain and its unique advantages make it a promising solution for securing big data services and protecting data privacy. Nonetheless, direct implementation of the blockchain technology on existing auditing systems is not practical [16]. The performance of blockchain system is degraded with the increase of the volume of data stored in the ledger. Accordingly, the deployment of the centralized auditing program into the decentralized blockchain network is challenging.

IoT systems face challenges in performing various identity management functions, maintenance of the trustworthiness of data, access control to data within the network, and detection of abnormal behaviours. Data provenance is a solution to tackle these challenges. It includes recording information about data operations, and data origins as well as analyzing the data history from their source to the current state. Blockchain offers a distributed data storage. It can be deployed to provide data provenance for various applications by recording data operations from blockchain transactions. Embedding

the data provenance (enriched by blockchain technology) into Big Data applications enhances system security and privacy while ensures data availability. The blockchain-enabled data provenance mechanism for Big Data applications in IoT systems guarantees data verifiability and integrity. This is because the data operations are recorded in the form of the transaction by every block in the blockchain network. Different devices within the IoT edge cloud architecture impose various trust concerns on the systems. Hence, a provenance mechanism is applicable to record the origin of multiple sensor data to meet these concerns [17]. Nonetheless, the blockchain-based provenance system scalability can be enhanced by integrating the high capacity of Big Data systems such as the *Hadoop Distributed File System (HDFS)*. Smart contracts combined with cryptographic methods maintain the task automation within the blockchain network. The integration of smart contracts helps to build up a secure environment for the IoT Big Data applications through a comprehensive data provenance management system. This study aims to provide the data provenance, integrity, traceability, and accountability for a large volume of data generated by a very large number of IoT devices and stored in a secure and verifiable Big Data ecosystem.

Blockchain technology paves the way to provide security and privacy for large-scale IoT data storage as well as to enhance the decentralized storage application, eliminate the centralized trust server, facilitate data traceability and accountability. Although many research efforts attempted to address the security and privacy challenges of Big Data systems, the authors are not aware of any studies on the application of blockchain technology that would comprehensively address the data traceability and data provenance for Big Data systems on large-scale IoT environments. The main goal of this paper is to address the outlined research gaps by implementing the HLF blockchain framework to maintain data provenance and auditing on Big Data systems within the large-scale IoT network without third-party auditing interventions. HLF blockchain is deployed to enhance data security by implementing mutual authentication and overcoming communication and computation overheads. In summary, the main contributions of this work are as follows.

- 1) HLF blockchain scheme is developed to provide secure data storage for Big Data systems in a large-scale IoT network. The proposed model maintains data privacy preservation, ensures a secure connection to a Big Data system through the HLF network, and guarantees data collection security. The centralized trust server is eliminated through implementing the HLF blockchain technology.
- 2) A two-layer security framework is proposed that involves HLF blockchain and a Big Data system. Trusted entities are linked to HLF, and third-party auditing parties are eliminated to reduce the compromised auditor's risk. The network scalability is enhanced by incorporating edge computing to maintain IoT data computation as well as to collect and forward data to the blockchain and off-chain storage.
- 3) A model is proposed to store the lightweight verification checksums and data pointers in the blockchain ledger to reduce the communication and computation overheads. The HLF blockchain performs data provenance while the actual metadata are stored in off-chain storage after being verified by the blockchain. Extensive experiments were conducted through a prototype implementation on a Hadoop system to

evaluate the performance of the proposed scheme in terms of throughput, response time, latency, communication, and computation cost.

The rest of the paper is organized as follows. "[Background](#)" introduces blockchain technology, security settings, Big Data systems, and the primary settings of the model. An overview of relevant state-of-the-art literature sources for the different data provenance solution approaches is presented in "[Related works](#)" section. In "[System model and architecture](#)" section, the proposed model is extended to protect large-scale IoT data storage. The system implementation is presented in "[System implementation](#)" section. Detailed model analysis and performance evaluations are presented in "[Results and discussions](#)" section. Finally, the findings are summarized in "[Conclusion](#)" section, along with outlining potential future research directions.

Background

Blockchain and Big Data systems are the two main components of the blockchain-enabled IoT data provenance framework. Blockchain provides a security and privacy basis. It guarantees the authorization and authentication for data owners and users with specific access and allows them to perform data analysis. Meanwhile, blockchain records storing the lightweight verification tags on the blockchain ledger to maintain the verifiability, integrity, and traceability of data are stored in off-chain storage. An overview of the technologies used in the framework is presented below.

Blockchain

Blockchain, as an open-source digital distributed ledger, is one of the most prevalent innovations broadly deployed in various areas [18]. Nodes within the distributed blockchain network communicate in a *peer-to-peer (P2P)* manner while the need for a centralized authority is eliminated. Blocks are the list of records wherein stored information is encrypted. All transactions are cryptographically marked and verified by all other participants holding replicas of the entire ledger and records. Thus, all records are immutable, tamperproof, synchronized, and cannot be changed when stored in the blockchain. Blockchain platforms can be categorized into three types: *private*, *public*, and *consortium*. Public or permissionless blockchains such as Bitcoin and Ethereum [19] allow all entities to join the network without restrictions while anonymous participants can perform the verification process. On the contrary, participants are required to get permission to join the private blockchain or permissioned blockchain network while the blockchain is limited to the authorized participants belonging to an organization or group of organizations. Only selected nodes within the blockchain consortium can perform the verification process (Hyperledger Fabric [20] and Ripple [21]). In consortium blockchains, a specific group of nodes has access to the public ledger. The blockchain architecture is partially decentralized. Here, the consensus process is maintained by all participants based on specific rules. The key features of blockchain are as follows.

- 1) *Immutable* Blockchain (with its permanent and unalterable characteristics) provides an immutable framework where each node has a copy of the ledger. Transactions are

verified and validated by nodes before adding to the ledger. Participants are not able to make alterations to the data stored in the blockchain ledger.

- 2) *Distributed* The deployed standard protocols in blockchain facilitate orchestrating blocks build upon the group of transactions verified by participants based on a predefined set of rules. Besides, blockchain can synchronize and distribute the data among multiple networks.
- 3) *Decentralized* The architecture of blockchain eliminates the need for a central authority. The governance is done by the group of nodes maintaining the ledger. Network participants hold a copy of all transactions and record their replicated data using private keys. Thus, the risk of single-point failure vulnerability is eliminated.
- 4) *Consensual* Data consistency within the blockchain framework is maintained by the associated consensus algorithms, and the blockchain operation relies on that. The consensus process decides to select a group of active nodes and remove the false and corrupted transactions added to the ledger. Maintaining transaction data integrity is achieved when all nodes agree by executing consensus algorithms.
- 5) *Anonymous* All users communicate in the blockchain network in a P2P fashion. Users' identities cannot be disclosed while the encoded transaction details are visible to all participants.
- 6) *Secure* High degrees of security are provided by the immutable and decentralized blockchain through deploying various cryptography techniques. Each set of data is uniquely identified by implementing hash functions on information and robust fire-wall algorithms to protect the framework against unauthorized access. The data are tamperproof as each block in the ledger holds its associated hash information and the previous block hash data.
- 7) *Traceable* The blockchain transactions are digitally signed and time-stamped thus facilitating data traceability and auditability. Every block is permanently connected to its previous block enabling the data owner to trace the data within the blockchain framework.

Big Data systems

Big Data is a definition for large data sets that traditional data processing systems cannot efficiently interpret, collect, process, and manage using conventional methods and mechanisms. Big Data typically have the 4-V attributes, consisting of *volume*, *velocity*, *variety*, and *veracity* [22]. Many challenges are associated with data volume processing, such as modularity, imbalance, dimensionality, data nonlinearity, bias and variance, and computing availability. Variety indicates the collected data types, which are naturally heterogeneous and involved structured data, unstructured data, multi-structured data, and semi-structured data. The velocity presents the data generation speed while the veracity indicates the quality of data generated from various sources.

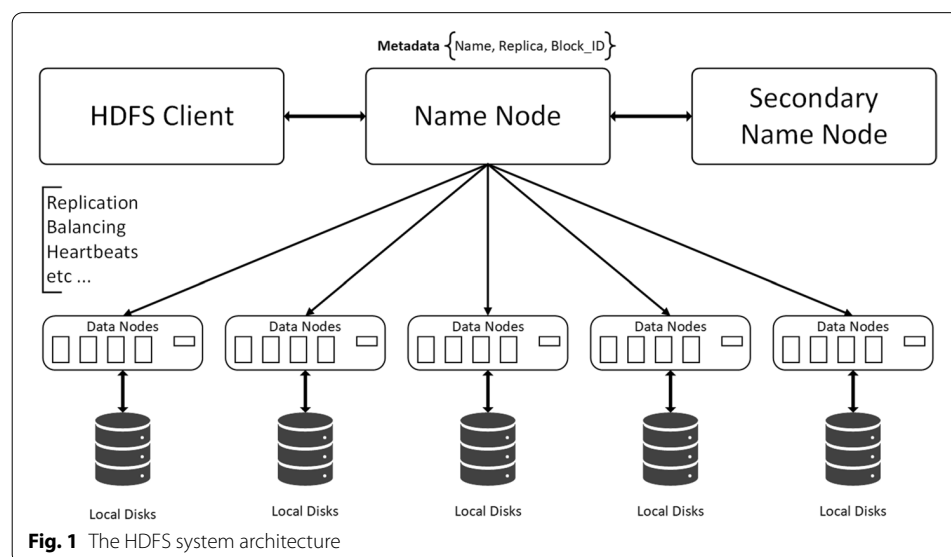
Big Data analytical systems facilitate knowledge extraction from multiple datasets for various purposes. The extracted information can be used in many applications, including smart cities, smart grids, e-health, logistics, transportations, mobile and wireless communications. The most popular data analytics frameworks in the industry are Hadoop [23, 24], MongoDB [25], Spark [26], and Storm [27].

Hadoop ecosystem

Hadoop is a framework to manage an orchestration of a cluster of computers with distributed processing based on the *MapReduce* programming model. Two main components of the Hadoop system are *MapReduce* (for parallel and distributed processing) and *Hadoop Distributed File System (HDFS)* (as storage of data in a distributed file system). Nodes within the Hadoop architecture are classified into *Master* and *Slave* ones. The master node performs the data collection and maps them to the respective slave nodes. The slave nodes maintain read/write operations in the file system and carry out the block creation, block deletion, and replication based on the name node rules. The master node then records all the operation results. The final results are formed by combining the sub results. The MapReduce determines the master node as a job tracker and the slave nodes as task trackers. Therefore, the job scheduling, subtask distribution, and fault tolerance associated functions take place in the master node.

HDFS helps to address the storage challenges of Big Data sets by distributing the enormous volume of data among various computing resources and machines called the Hadoop cluster. The main components of the HDFS architecture are the *Name Node* (referred to as a master node), the *Data Nodes* (slave nodes), and the *Secondary Name Node* (the name node backup). The HDFS distributed system architecture is illustrated in Fig. 1.

MapReduce programming framework is implemented within the Hadoop system to perform the Big Data processing. The framework maintains the processing of large data sets in a parallel and distributed manner across the Hadoop cluster architecture. Map phase and Reduce phase are the two distinct tasks of the MapReduce process. The data process is happening in all machines with the Hadoop cluster. It is known as the Map phase. Combining the outcomes and forming the final results are referred to as the Reduce phase. MapReduce is one of the core pieces of Hadoop that performs big data analytics along with HDFS and *Yet Another Resource Negotiator (YARN)*. YARN is a technology sitting under the hood to manage all the resources of the cluster and to



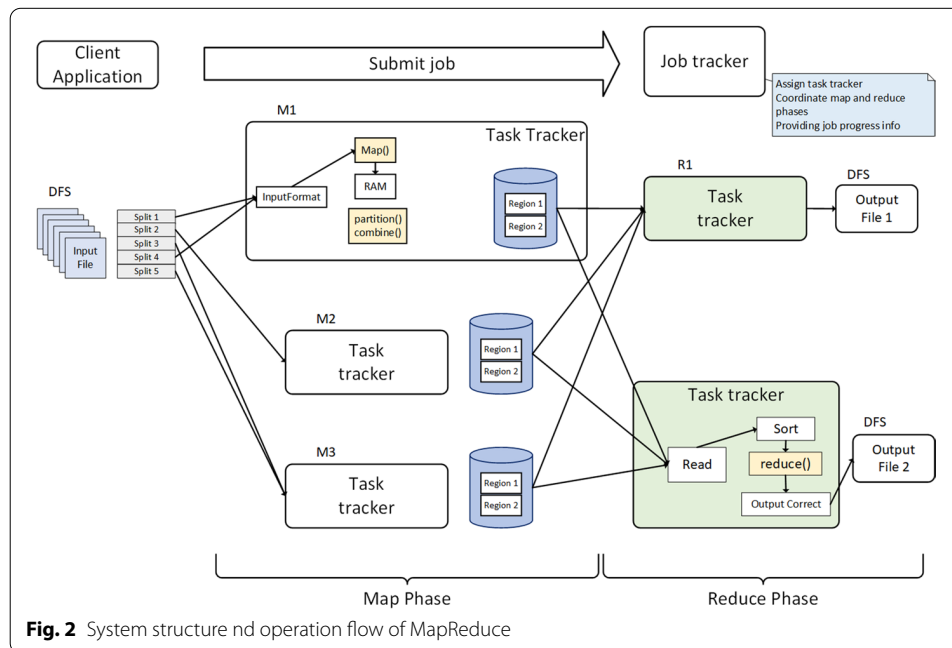
assign computational resources for application execution. *NodeManager*, *AppManager*, and *Container* are the components of YARN. Figure 2 presents the detailed operations and system structure of MapReduce.

Related works

Data provenance and data integrity have been considered as critical elements of the security requirements for big data analytics in IoT-based solutions. They enable users to check the integrity of stored data in outsourced storage. Data provenance enhanced with blockchain technology is a promising solution to provide the trustworthiness of stored data through immutable and tamperproof information about the data origin and history of data records.

Blockchain-based data provenance in IoT

AgriBlockIoT [28] is a fully decentralized blockchain-based model to maintain the data traceability for Agri-Food supply chains. The mechanism provides immutable, fault-tolerance, and auditable records of the whole supply chain system from production to consumption. The history of the purchased product is recorded in the blockchain system thus enabling effective data retrieval for consumers. The proposed system collects data provenance including the data origins and operations performed on the data. The trust concerns coming from various IoT edge devices in cloud infrastructure are addressed by a provenance mechanism to record sensor data and origins of the related entities [17]. The provenance system structure is based on a combination of IoT edge devices organized with a blockchain network. Blockchain transactions are used to record all actions within the ledger with data provenance. *Physical Unclonable Functions (PUFs)* are utilized in BlockPro [29] to facilitate the data provenance and data integrity to achieve secure IoT environments with the help of the Ethereum blockchain and smart contracts.



PUFs produce unique hardware fingerprints for each device and deploy them to find data provenance and identify the data source.

A distributed database based on the blockchain was designed to guarantee data verifiability and integrity called ProvChain [30]. The Ethereum blockchain and two smart contracts are applied to maintain a decentralized digital ledger to ensure data integrity and prevent data tampering attacks. Data operations are stored in the local ledger while the blockchain records the provenance entry. The provenance retrieval from the blockchain network is maintained by a *Provenance Auditor (PA)* that keeps the local database. Research [31] reports an extensible and secure IoT data provenance framework based on a layered architecture consisting of smart contracts and Ethereum blockchain implemented for a wide range of IoT applications. Shreya Khatal et al. [32] propose decentralized storage called *Fileshare* for file sharing within the secure environment based on blockchain to ensure the integrity and ownership of shared files. The introduced *Decentralized Application (DApp)* is built upon the Ethereum blockchain framework while smart contracts utilize a distributed file system in the data layer.

Blockchain-based data verification

Blockchain technology has recently attracted many researchers in various fields, including cloud data storage and data integrity, application of edge and fog computations, provenance, etc. [33]. The foundation of cloud storage systems is formed based on data storage. The challenges of storing cloud data securely are being investigated in [34, 35] and addressed by deploying blockchain techniques. Restructuring the history of data associated with each data operation or scientific result is a critical data provenance element. Nonetheless, it facilitates data management more efficiently in different applications such as scientific data and high-quality web data management. Works [36, 37] investigate embedding the data provenance mechanism into blockchain transactions to address the collection and verification issues.

Computing services with low latency and higher bandwidth are in place by applying the new edge and fog computing techniques with shared resources. Edge and fog computing security can be further enhanced by integrating the emerging blockchain technology to establish a trusted decentralized environment. Blockchain technology has been considered to ensure the privacy-preserving for applications on edge platforms. Besides, data storage and resource allocation applications are deployed using blockchain at the edge and fog computing level while the architectural security is further improved [38–41]. Although the existing works attempt to enhance the data provenance mechanisms and edge computing applications by replacing some functionalities with blockchain technology, they still rely on centralized entities with significant limitations and additional overheads generated from deploying the centralized entities.

Research [42] proposes a framework for data integrity based on a blockchain for peer-to-peer (P2P) cloud storage. Data integrity verification is deployed using rational sampling approaches to establish sampling verification effectively. A fixed third-party auditor is deployed to perform the integrity verification of operation logs based on blockchain in the cloud [43]. This method brings third-party auditor security drawbacks into the system while the computing and communication overhead being quite considerable. A certificate-less public verification scheme against procrastinating auditors with the aid

of blockchain technology is proposed in [44]. The main idea is based on recording each verification by auditors in the form of blockchain transactions. Moreover, certificate-less cryptography is deployed in the scheme to overcome certificate management issues.

Implementing blockchain in a distributed large scale IoT environment with Big data storage providing the protection is a challenging task. The most significant issue is providing a light-weight authentication mechanism to manage the identities of users and IoT devices through a blockchain system. Most of the research works consider authentication and other security primitives in a centralized server. Besides, providing a secure channel where data provenance and accountability can be maintained without intervention from third party and trusted central server is a major limitation of the previous works. In our work, we take advantage of light-weight authentication model to achieve effective and efficient authentications for the users and IoT identities. The works cited in the literature review suffer from scalability problem that has been addressed in our proposed scheme with multi-layer blockchain approach extending to Hadoop database as off-chain storage of the underlying database. Hadoop is a distributed and scalable Big Data storage and supports random, real-time read/write access to Big Data. Furthermore, the majority of the previous studies consider the Cloud-IoT environment in which a large number of users and devices share data through the cloud computing infrastructure. However, the cloud can not provide a scalable platform and suffers from a lack of supporting a vast number of users. Therefore, the existing research works are limited to a certain number of devices and users. These challenges have been addressed in our proposed framework. In addition, our work considers processing massive data in IoT devices through lightweight algorithms to overcome the limitation of energy efficiency and processing performance of the current approaches.

System model and architecture

Blockchain technology and Big Data integration have been considered as potential solutions to address large-scale real-world problems. The exponential growth in the generated data presents its own security and privacy challenges and issues associated with data sources reliability and data sharing. The challenges of the Big Data ecosystem can be answered using unique features of blockchain technology such as decentralized storage, transparency, immutability, and consensus mechanisms. The integration of them can further enhance Big Data security and privacy, improve data integrity, provide fraud prevention, facilitate real-time data analytics, expand data sharing, enhance data quality, and streamline data access.

This work aims to develop a blockchain-enabled public data provenance and auditing model in the Big Data ecosystem (Hadoop ecosystem) to provide a more efficient and secure framework than the reported solutions. Blockchain offers a decentralized database that records the history of all transactions appended to the shared ledger and enhances data traceability. The information inside the Big Data ecosystem in many applications is shared with multiple workers and writers, while most of them may be non-trusting participants. Blockchain as a resilient framework is a feasible solution to eliminate a third-party intermediary, provide automated interactions among multiple transactions in the shared database, and enhance auditability. To achieve the goal of the research, a distributed provenance tracking architecture is designed while deploying an

application built on top of the HLF and Hadoop ecosystem. The proposed data provenance model aims to identify the way the data was derived and to provide data *confidentiality*, *integrity*, and *availability*. The HLF permissioned blockchain having registered members offers the above-required functionalities.

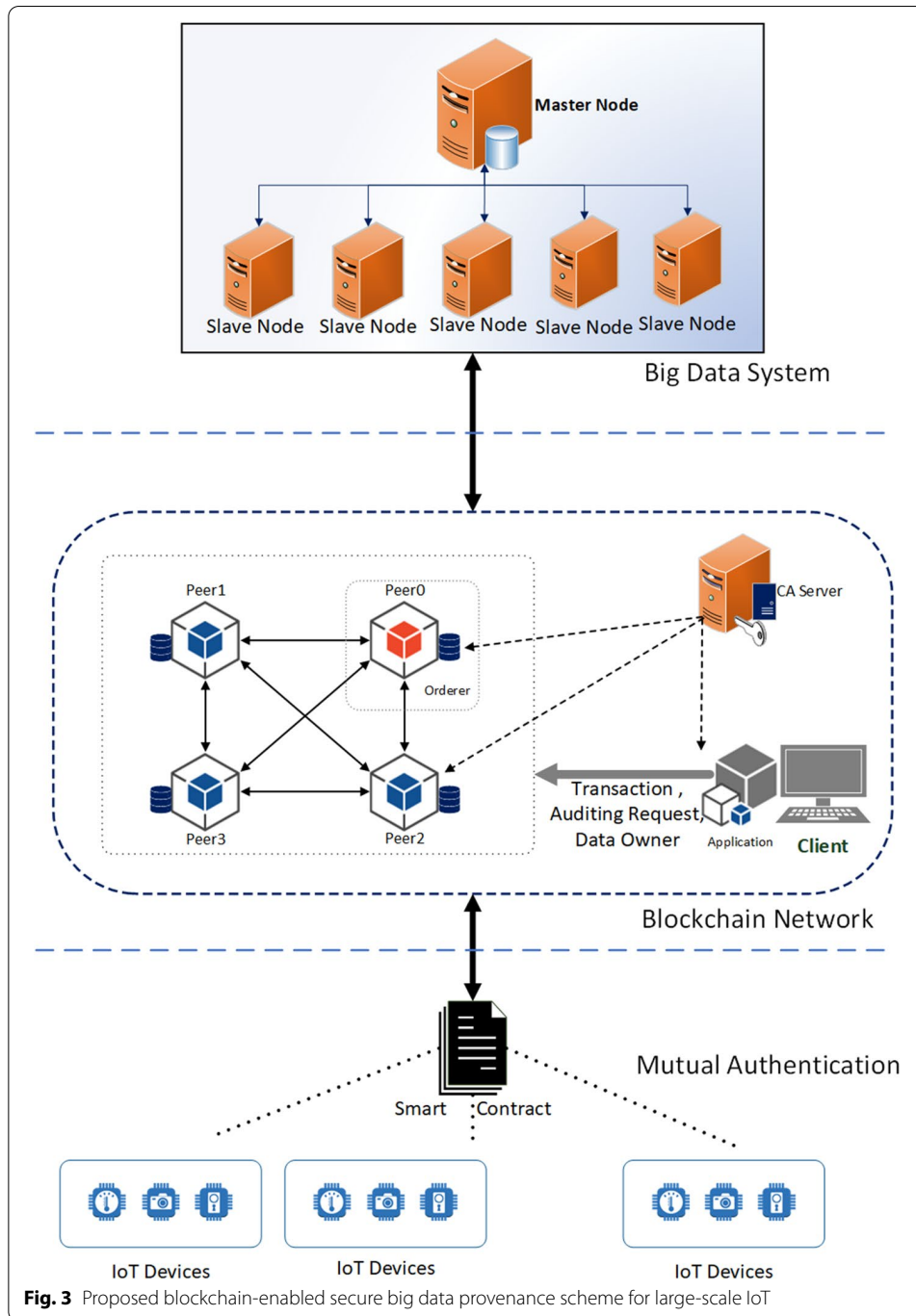
The blockchain-based high-level scheme

The architecture of the system includes three layers: a blockchain layer, a Big Data system (off-chain storage) layer, and an authentication provider layer. The first component is the HLF network implemented and running in *Docker containers* [45] and associated client libraries for multiple interactions with the HLF. The second component is the Big Data system (Hadoop ecosystem, in this case) operating as off-chain storage. The communication with other parts of the system is managed through a client library to initiate and perform multiple tasks and operations. The proposed model aims to provide seamless records of provenance data in a tamperproof and immutable blockchain ledger, ensuring data storage and access to a pluggable Big Data storage service. HLF blockchain framework is deployed to record all provenance data entries. Multiple data operations can be stored within each block in the system. Data operations, as well as invoke and data querying, are recorded in the blockchain ledger. The identity of devices needs to be registered and stored in the shared ledger before running the HLF blockchain. Each gateway collects data from connected devices and sends them to a higher level for verification and storage. The registration request is sent to the gateways (IoT applications to edge IoT nodes). It includes the required information such as device ID, gateway identity, and timestamp. The gateway runs the *ChainCode (smart contract)* in the local blockchain to perform the device registration. The mutual authentication model is designed and implemented to provide the device's authentication before joining the network to ensure a secure and trusted environment. Secure communications between entities can be then established through the implemented blockchain network.

Figure 3 demonstrates the blockchain-based data provenance system model. It verifies data integrity by finding the location of the data item and associated checksum. The lineage of data for new items is accessible via storing the references to the data items deployed to create it. Data operations are recorded to have good visibility on the time and the clients who stored the data items or manipulated the data object. The records are maintained based on the certificate ID used to invoke the transaction. Such a design provides a data provenance framework based on the HLF blockchain offering data security, privacy, and auditability for Big Data systems.

Hyperledger framework

Multiple HLF processes are orchestrated and configured to be run on different nodes using Docker containers. Nodes in the HLF network run a peer process and maintain the shared ledger through various transaction proposals. The client library initiates the transaction proposal using the HLF software development kit functions, which are cryptographically signed with a certificate generated by the *Certificate Authority (CA)*. The critical element in the HLF framework is the peer process. It holds a replica of the shared ledger by running the ChainCode. Running more peers helps



to achieve higher performance. At the same time, one peer per organization is sufficient to run the HLF network. The ordering service handles the block ordering (based on deterministic consensus protocol) and validates the blocks that peer processes have proposed. The single-orderer architecture is considered in the reported model using the built-in HLF RAFT consensus algorithm. Figure 3 presents the proposed

blockchain-enabled secure Big Data provenance scheme for a large-scale IoT, including the HLF framework components.

Hadoop data storage

The distributed shared ledger implemented through the HLF blockchain has limitations in terms of data storage. The HLF performance degrades with the growth in the ledger's size resulting from increasing the blockchain platform's shared ledger. The proposed model stores the provenance of data in the shared ledger (a small portion of the metadata). The actual metadata is placed in the Hadoop ecosystem to tackle the issues mentioned earlier. In this way, the data is stored in the off-chain storage. The data checksums are computed to perform the data verification and integrity checks. Hence, the HLF blockchain can verify stored data integrity by comparing the immutable recorded information in the shared ledger with the checksum of stored data in the Hadoop system. A ChainCode is developed to facilitate these operations running in each peer node within the HLF network. The built-in client library sends the data checksum and provenance data. The file-store operators are not needed and thus are eliminated. A flexible distributed Hadoop ecosystem is introduced as a pluggable storage solution to accommodate secure and verified data. The client facilitates the data invocation process by putting the data in the Hadoop storage and sending information to the blockchain for verification. The data query operations initiate the ledger side to acquire the location and address. Then the data is retrieved from the Hadoop storage.

ChainCode

The ChainCode runs on the peer nodes, maintains the data query, and appends data on the information stored in the shared ledger. It is the main component in the model with several functionalities to automate the tasks within the blockchain platform. All peer nodes have access to the functionalities implemented in the ChainCode. Storing and retrieving data from the shared ledger are automated by the ChainCode. The proposed design considers storing checksums of all data objects, data addresses and locations, information about workers who stored the data, information on creating an object, data lineage, timestamp, certificate ID, and additional fields that can be customized for various data structures (e.g., JSON structure). The process starts with the ChainCode functions (invoked as parameters associated with the data) to begin storing data in the HLF ledger. A specific function is designed in the ChainCode to perform the data retrieving functionalities. The data can be queried based on the data items assigned to a particular stored key and the data iterations. The query of data collections is provided through various query definitions within the ChainCode, either by key range or the iteration history.

Client library

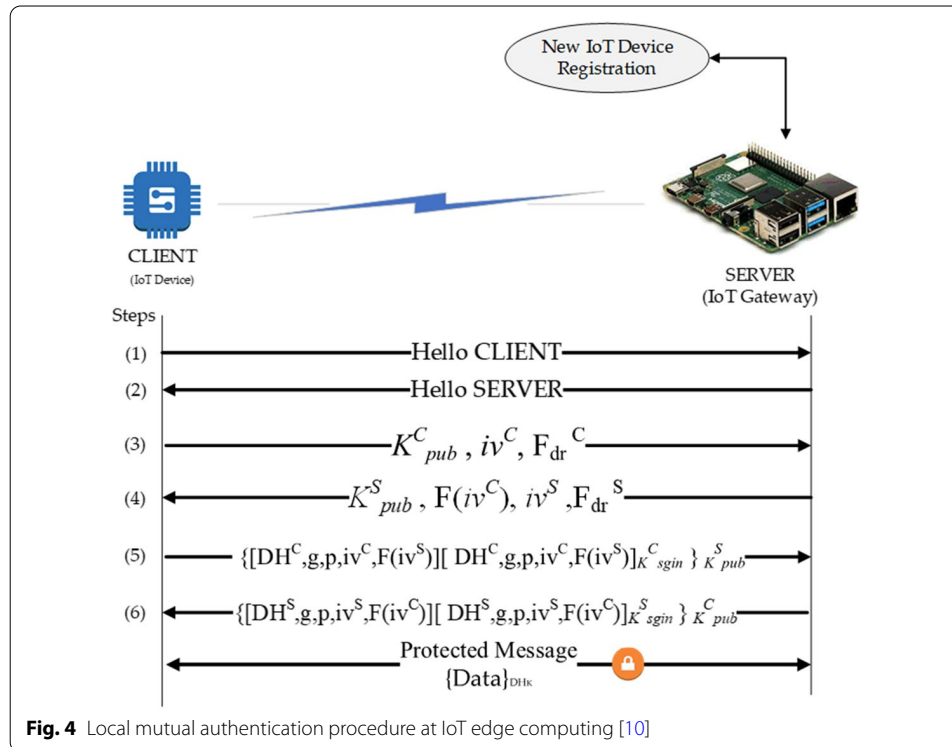
The client library is developed and built using the *Software Development Kit (SDK)* [46] to interact with the HLF blockchain platform for data verification and provenance operations. The client library is a core element operating as a middleware for all other applications that need to interact with blockchain and store data or record the provenance data. The client application communicates with both blockchain and Hadoop systems for various operations as different distributed workers. It can be integrated into the peer

node or work as a separate node within the HLF blockchain system. Several client nodes with their associated client applications can run the HLF blockchain as an overlay network in the background and perform various tasks relevant to Big Data analytics. The client applications control the fraction of data stored in the HLF shared ledger and the metadata that need to be stored in the Big Data system.

Edge computing

The edge computing device is a central node to implement the blockchain-based IoT Big Data storage scheme. It offloads the tasks from small IoT devices and maintains significant energy savings. Besides, it performs the associated computations, manages data storage, and relays transactions and messages for IoT devices. The edge computing node contains the IoT device identification and authentication information. It stores the identification information of all interconnected IoT devices and provides a pair of keys for each device to perform the authentication through implementing a lightweight mutual authentication protocol.

The authentication procedure is shown in Fig. 4 for each IoT device to the edge computing server. The generated messages and transactions by IoT devices are managed and created by the IoT edge node. The HLF blockchain framework runs on the edge computing nodes, and the edge server conducts signing valid transactions, including the IoT device signature. The sensitive data are then verified to be ready for storage in the Hadoop ecosystem while the data checksums and related operation tags are stored in the HLF blockchain. The edge servers collect all verified and trusted data and send them to the Hadoop distributed file system. The collection of data from interconnected IoT



devices is a continuous process. The locations and addresses of the stored data are determined in the HLF blockchain for further verification and traceability operations. The details of the authentication and authorization process and the procedures to implement it within the layer-based structure are fully covered in the earlier research [10].

System implementation

The interaction with the ledger in HLF is possible through executing the defined ChainCode. The ChainCode is responsible for storing the data provenance and handling various data queries. Hence, the system implementation starts with defining specific ChainCode operations consisting of storing data provenance, querying the lineage of data, and retrieving data lineage. The IoT applications require a lightweight ChainCode to be implemented on endorsing peers to address the limitation of IoT devices in terms of their communication capacity, storage, and processing power. The access of the ChainCode to external resources is limited to ensure that the ChainCode can provide the same results for all endorsers. The ChainCode is designed to support different operations associated with data provenance and traceability of data within the ledger and the attached off-chain storage. The ChainCode specific operations in the proposed system include storing the data provenance related to an item, querying item checksums, retrieving an object with the associated transaction ID, extracting the version of an object based on its transaction ID, retrieving the lineage of the data item, retrieving the history of a data objects, querying the key-range of the list of items, retrieving the provenance information, and providing a specific version of an object and the related transaction ID. The main implementation concern is to make the ChainCode lightweight that can address the limitations of IoT devices and allocate a significant part of functionalities to the client applications. The implemented system consists of distributed peer nodes that are at the centre of communications between network elements and the off-chain storage (Big Data ecosystem).

The performance of the proposed model was evaluated for the system throughput, response time, latency, and resource consumption (memory, CPU, network) metrics. The evaluation was further expanded to cover the scalability of the distributed large-scale IoT network environment. The measurements were conducted by implementing a benchmark application on top of the *node package manager (NPM)* libraries run on the client node. Besides, various Linux-based tools and utilities were deployed to monitor the system's performance. To emulate a large number of IoT devices, the customized *Locust* [47] was deployed on an independent server interconnected with the edge computing devices in the same LAN. The experiments were conducted by emulating 100–2000 IoT devices connected directly to the edge IoT servers to send messages and transactions. A maximum number of 500 IoT devices was considered to be managed by each edge device. The edge server stored the identification of all connected IoT devices and authenticated them within a trusted HLF environment by implementing a mutual authentication scheme described in "[Edge computing](#)" section.

The performance analysis was carried out for the proposed model for various workloads and environment parameters. Moreover, a diverse set of interaction performances was observed to explore the improvement or degradation caused by different

parameters and configurations of the model. Several benchmarking applications build on NPM libraries run on client nodes were employed to perform the benchmarking processes.

Often, stakeholders need to find out which benchmarking model is suitable for their applications and particular use cases since different methods differ in terms of involved parameters and phases. To address this challenge, the HLF performance guidelines and HLF performance metrics documented in the Hyperledger Performance and Scale Working Group white paper [48] were considered to conduct the benchmarking of HLF V1.4. Real-time data reporting was deployed and statistic data on resource utilization were collected and monitored.

Experiment setup

The setup of the developed system prototype consists of five units of ARM-based *Raspberry Pi (RPI)* 4B, a client-server, and a Hadoop system as off-chain storage. The hardware and software specifications of RPis are summarized in Table 1. The RPis, client-server, and Big Data ecosystems were interconnected within the same LAN. The peer docker containers run on each RPi, and one node was assigned as the orderer node. Unofficial docker images of HLF version 1.4 were modified and established on each RPi device. The docker images were compiled to suit the ARM64 architecture of the RPi. Performance measurements were conducted by a client desktop computer using a client application build on the NPM libraries. The client application was developed using HLF node SDK. The ordering type (using the solo type of order in HLF) indicated that the consensus was achievable by a single ordering node implementing a sorting algorithm. New block generation was done based on specific parameters that have been defined in the client application.

The Hadoop cluster was configured with one master node and five slave nodes. The cluster was equipped with 48 CPU core and 35 TB local storage. The details of the Hadoop cluster configuration and associated software are presented in Table 2. The cluster had dedicated switches. It worked in the same networking structure. The same as mentioned in "[Blockchain-based data verification](#)" section, Yarn maintained the resource management. It facilitated resource monitoring for active nodes including the job details and correspondent histories. The HDFS was configured in the master node (name node), secondary name node, and five worker nodes (data nodes). Figure 3 shows the proposed model and the system under test architecture used for the performance measurements.

Table 1 Raspberry hardware specifications

Type of device	CPU cores	Memory
Raspberry Pi Computer Model B	Broadcom BCM2711 Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz	4GB LPDDR4
Client node	Intel (R) core(TM) i-7-6700 CPU @3.4 GHz	8 GB

Table 2 Hadoop cluster experimental setup and specifications

Node configuration	Hardware	Specifications
Server configuration	Processor	2.9 GHz
	Main memory	64 GB
	Local storage	10 TB
Node configuration	CPU	Intel® Xenon® CPU E3-1231 v3@ 3.40 GHz
	Main memory	32 GB
	Number of nodes	5
	Local storage	6 TB each, 30 TB Total
	CPU cores	8 each, 40 total
Software	Operating System	Ubuntu 16.04.2 (GNU/Linux 4.13.0-37-generic x86_64)
	JDK	1.7.0
	Hadoop	2.4.0
	Spark	2.1.0
Workload	Varying data sizes and batch sizes	Submitted by the HLF client application

Results and discussions

This section presents the results attained after implementing the proposed model and sending various batch sizes and workloads to evaluate the performance of the entire architecture. We have assessed the performance of the proposed model by measuring multiple parameters consisting of the system throughput, response time, latency, and resource consumption (memory, CPU, network) metrics. Each measurement was conducted with some repeats, and the average obtained results were plotted in each graph.

Throughput and response time measurements

The benchmark application was developed on top of the client library to generate transaction batches to the network. A timer was associated with each transaction. In addition, the timer was allocated for every set of transactions. The benchmark application calculated the response time of a transaction and the total average time while considering the number of successful and failed transactions for various data set and batch sizes. The benchmark application was powered with the ability to store the data in the HLF ledger or the Hadoop system. The performance evaluation was initiated by several transactions submitted together. The results indicated that the throughputs and response times were affected by the size of the data. However, the impact was not significant if the data provenance and the transaction tags were only stored in the blockchain ledger. As stated earlier, the provenance of data was stored in the blockchain ledger, and the actual metadata were placed in the Hadoop ecosystem off-chain storage.

The performance was affected when the Hadoop system was involved in storing the metadata (since the client application needed to consider the time for calculation of data checksums, operation tags, and the time to store the data in the Hadoop system). Figure 5 illustrates a degradation in the performance with the growth in the batch sizes.

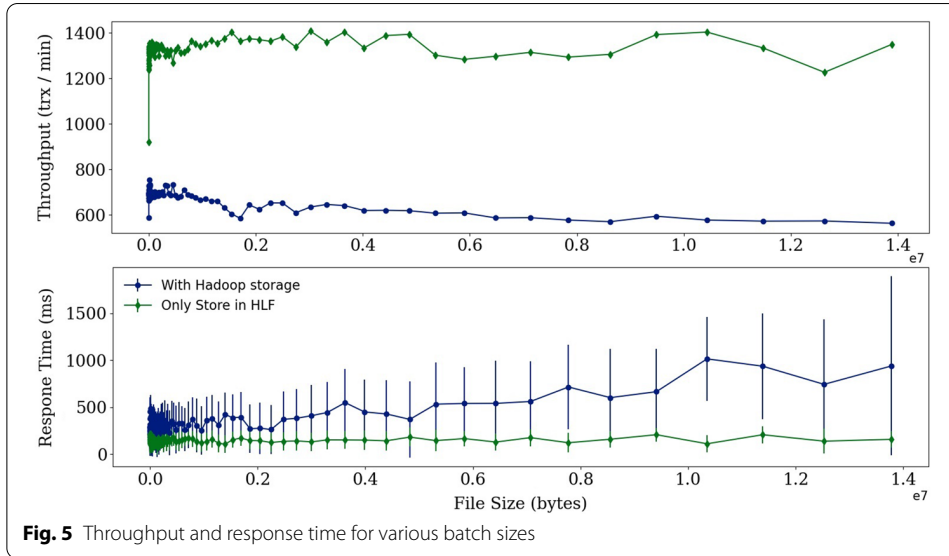


Fig. 5 Throughput and response time for various batch sizes

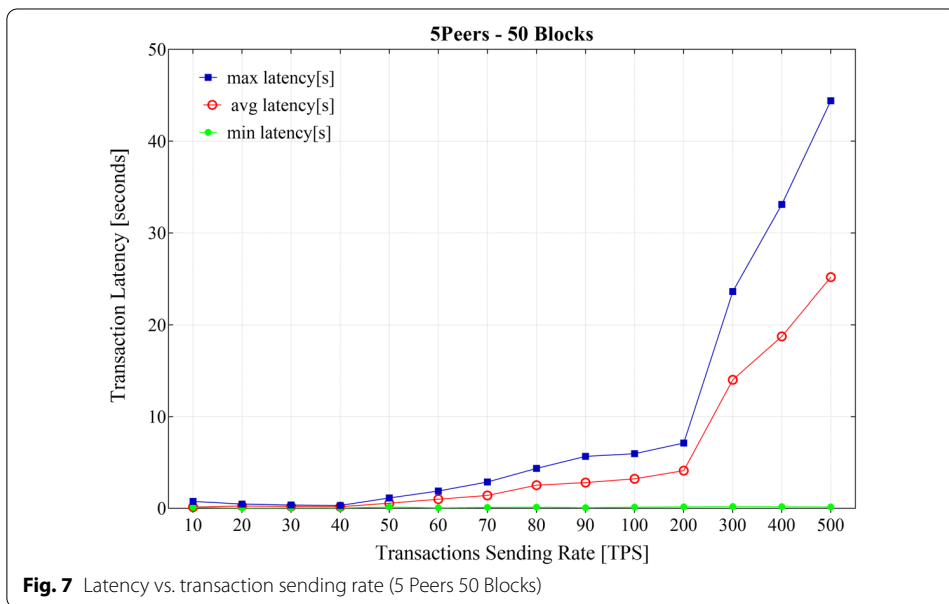
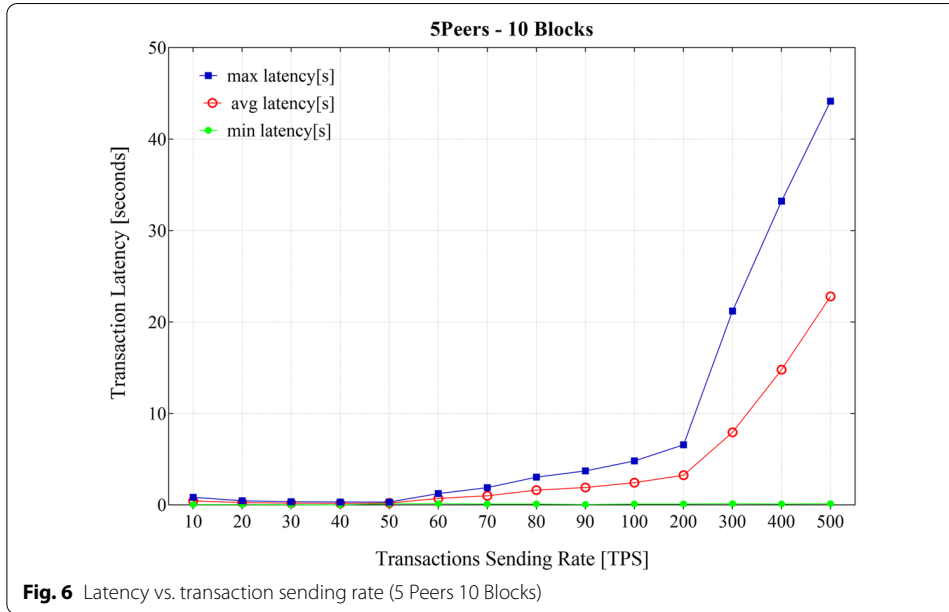
The obtained results (600 transactions per minute and 500 ms average response time) can be considered as very promising according to the HLF performance guidelines and HLF performance metrics documented in the *Hyperledger Performance and Scale Working Group* white paper [48]. One of the main limitations came from the client node's hardware capabilities and the peer process hardware constraints. The HLF employed the *Execute-Order-Validate* and *Commit* transaction model. Therefore, the system needed to perform the required operations for each data object, resulting in degradation in the throughput and increased response time. Besides, the system needed to consider the time for storing the data provenance in the HLF ledger, calculating the checksum of data objects, and storing the metadata in the Hadoop system.

To address the challenge, the network was made to include multiple clients. More endorsers were required to improve the overall throughput and response time performance. With the small number of transactions, the throughput was slightly lower, while the increase in the number of transactions led to some rise in the throughput. At the same time, it could be noted that the throughput was approximately constant for a certain number of transactions.

The latency measurements were done by running multiple rounds of the benchmark to submit various transactions with different sending rates (from 10 *Transactions-per-Second* (TPS) to 500 TPS) for different block sizes. The goal was to measure the maximum, average, and minimum transaction latency.

The results indicate that during the experiments, the minimum latency remained below 1 s. However, there was an increase in the maximum latency when the sending rate reached around 200 TPS. This was due to the rise in the number of ordered transactions waiting in the verification process queue during the validation phase that significantly increased the commit latency.

Since the system setup deployed a solo-orderer configuration, other orderer types needed to be employed along with different configurations. Consequently, the validation phase was considered as being a bottleneck in the overall system performance.



Hence, there was a need to deploy a smaller block size with a low transaction rate for IoT applications to have lower transaction latency. In contrast, higher transaction rates needed larger block sizes to achieve higher throughput and lower transaction latency (the results of latency measurements for various block and batch sizes are presented in Figs. 6 and 7). It happened mainly due to the increasing waiting time for transactions in the ordering services.

A potential optimization solution to overcome this drawback is to process transactions in parallel with sharding. However, the effect of transaction conflicts needs to be considered. Besides, in order to achieve lower transaction latency, it is recommended to use a

lower block size (along with a lower arrival transaction rate) than the default block size. Hence, with a higher transaction arrival rate than, a higher (than the default value) block size is recommended.

Large scale IoT environment evaluations

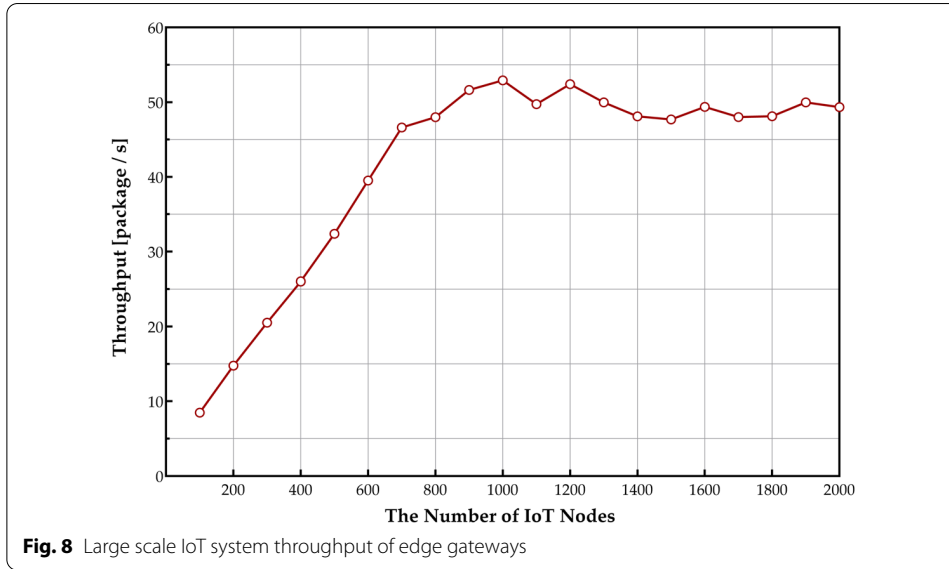
The data collected from massive IoT devices were managed by Edge computing and IoT gateways as a middleware between the IoT sensors and Big Data systems as well as application services. The data provenance tracking was maintained to ensure the quality of shared data. The process consisted of the identity, validity, and lineage of data. Hence, edge devices could save energy for small sensors by offloading work, improving the bandwidth, and decreasing the latency. The edge devices performed preprocessing tasks and compression, resulting in significant energy saving for IoT devices. The traffic evaluations demonstrated a constant range between 20 KB/s and 30 KB/s in the idle state when there were no transactions between peer nodes and 100 KB/s during maximum load where the maximum amount of transactions were exchanged. The increase in the number of the orderer and endorser peers could improve the performance through the gossip protocol configuration. However, the results indicate that the proposed provenance model was promising for application in large-scale IoT networks with many trusted IoT sensors and devices. The generated ChainCode queries were able to retrieve 10 linked IoT records in 104 ms.

To further explore the system's performance in a large-scale IoT environment, a set of experiments were conducted with varying numbers of IoT devices connected to each edge device. By implementing a large-scale IoT environment, the impact of CPU utilization and throughput on the system were explored. The experimental environments included from 100 to 2000 IoT devices distributed equally between IoT gateways (RPI). All devices needed to be authorized before initiating communication with the network and other participants. The procedure of mutual authentication is described in "[Edge computing](#)" section. The increase in the number of IoT devices caused growth in the processing time. That was addressed by modifying the HLF configurations and adding more orderers and endorsers based on specific applications.

The system throughput and CPU utilization are illustrated in Figs. 8 and 9. Figure 8 shows a linear growth in the system throughput until it reaches the maximum load (around 1000 IoT devices). It can be seen as a result of gradually increased resource allocation by the system until all resources were fully utilized. As depicted in Fig. 9, the CPU utilization was increased, reflecting higher resource utilization by the system. After the peak point, the throughput stabilized. The CPU was mainly used during the validation phase of a generated block. Therefore, modifying the configuration of HLF in terms of the batch timeout, maximum message count, and block size can result in better performance.

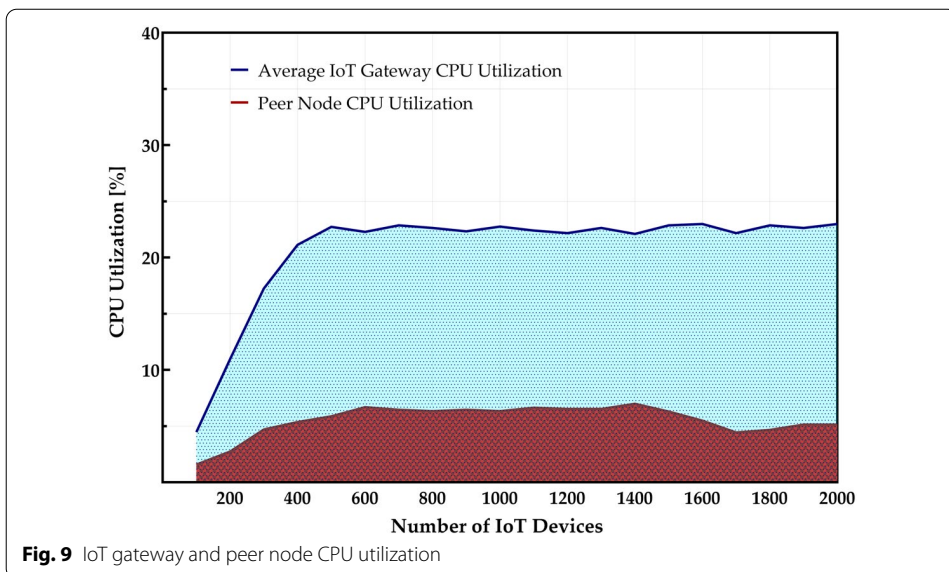
Data provenance and tracking resource consumption

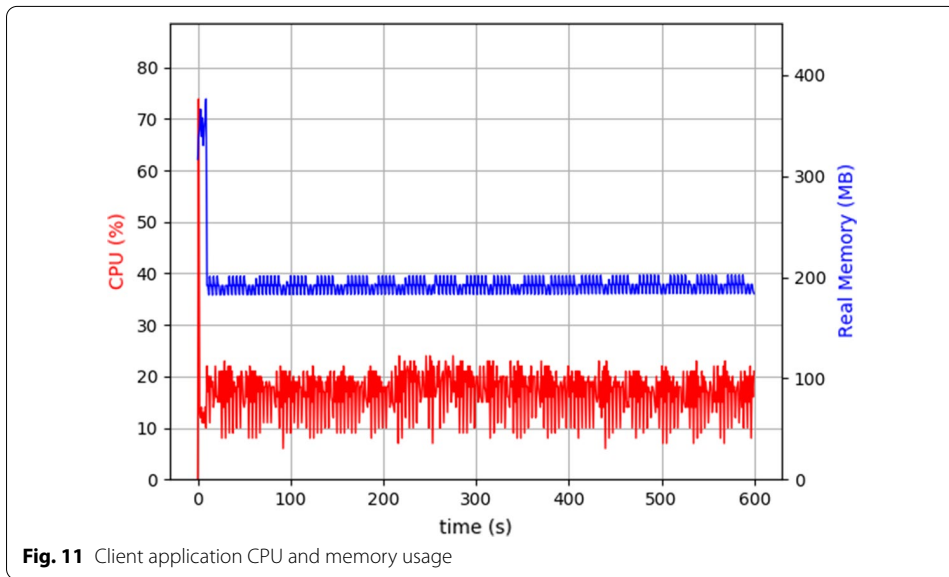
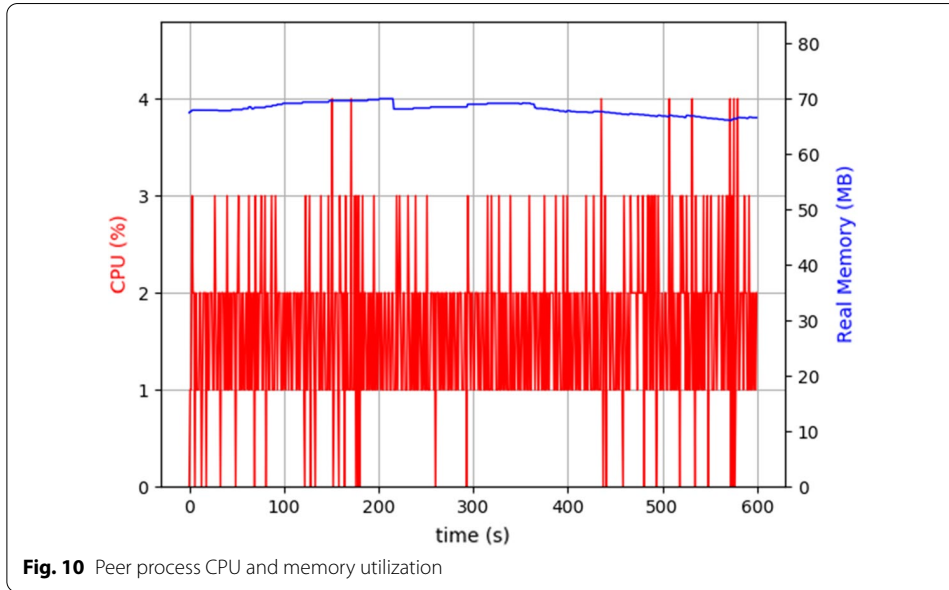
To further evaluate the performance of the proposed system, the federated machine learning technique [49] was considered for application across the distributed set of network participants while providing a collection of models, training, and test data sets. The framework was implemented in a way that facilitated data provenance and metadata



tracking. The *ImageAI* [50] library was implemented. Training and data sets were provided in the framework. The process was initiated with storing the model. It included the following steps: data checksum computation, storing the metadata in a big data system, and maintaining the transactions by the client application library through the HLF blockchain to record the data checksum and files locations. Storing 100 models of 100 MB (the models were created using the ImageAI library) was successfully performed in around 2.3 s.

The resource consumption measurements results are presented in Figs. 10 and 11. The results show that the CPU consumption was slightly (2–3%) affected in the peer process during the model storage. The client application consumed more CPU capacity—approximately 10 to 20%. The reason for that can be found in the range of operations that needed to be handled by the client applications: computing data checksums,





storing the metadata in the Big Data system as well as storing information within the HLF blockchain. It indicates that this network model can easily be deployed on low-cost IoT devices (e.g., RPi) for real-world applications.

The client application profiling can reveal more details of the CPU consumption. Figure 12 indicates that the majority of CPU consumption by the client process was due to checksum computation and data storing in the big data system. Storing the HLF blockchain information by the client process and garbage collection occupied a small amount (some 6% and 7%, correspondingly). As mentioned in "System model and architecture" section, the client application needed to follow the sequential operations resulting in more CPU consumption. These limitations can be addressed by implementing the calculation of checksum in parallelized manner.

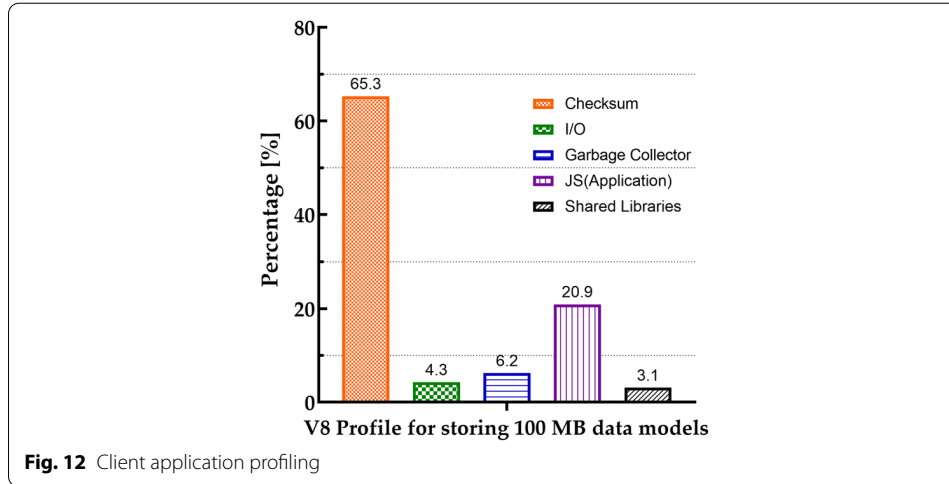


Fig. 12 Client application profiling

The network traffic measurements were conducted to explore the network overhead impact by storing the models generated by ImageAI every 2.3 s (this was for storing 100 models of 100 MB). The proposed provenance data framework was able to store the data provenance information, including data checksum, the data location, operation tags, data owner information, and some other optional parameters about the batch size. The system also provided the history of data models, tracked training datasets, and tested dataset provenance. Therefore, the lineage of successful transactions could be traced, and the model could be verified through the system. The previous measurements (associated with the utilisation of CPU, memory, and profiling) indicate that the system limitations were mainly due to the size of the file, checksum calculation, and network transfer. Figure 13 shows a low overhead for storing 100 MB data objects. Storing large files could be considered as a limitation that impacts the

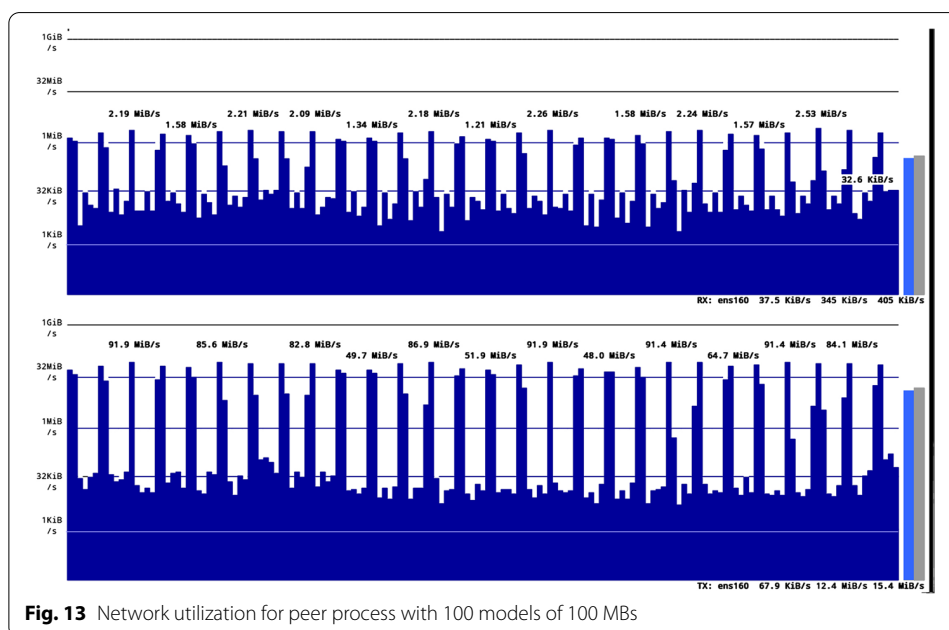


Fig. 13 Network utilization for peer process with 100 models of 100 MBs

network traffics. Hence, in such cases, the optimized solution would be to store the data provenance in the HLF. Files of large size (e.g., megabytes range) posed additional loads on the client nodes due to various resource-consuming operations such as checksum computation. The statistical results indicate no abrupt or anomalies in the network performance with changes in the system configurations. The network performance was mainly dependent on the traffic input and output. There was a progressive response to traffic changes—the observed increases in the network performance were primarily related to the growth in the network traffic thus indicating the normal network behaviour.

Conclusion

This paper proposes a blockchain-enabled secure framework for large-scale IoT data storage in a Big Data system environment. Edge computing is considered to be merged to facilitate the management of the authentications of the small IoT devices and perform data storage. A lightweight mutual authentication scheme is deployed to perform authorization and authentication of IoT devices in blockchain-based IoT applications.

The paper presents the detailed implementation of the proposed security scheme to provide the data provenance, data integrity, traceability, and auditability of IoT data in the Hadoop system as off-chain storage. The proposed model offers tamper-proof and transparent records spread across a collection of distributed peers by developing a provenance scheme using blockchain. The model also overcomes the high communication and computation overheads associated with storing large volumes of IoT data in centralized cloud storage. The proposed model eliminates the need for third-party auditing and a centralized server.

The results of the experimental research show the throughput of about 600 transactions per minute and 500 ms of the average response time. Peer and client processes were the primary resource consumers in HLF. The measurements showed about 2–3% of the CPU capacity consumption at the peer process, and approximately 10–20% at the client node. The minimum latency remained below 1 s during the experiments. However, there was an increase in the maximum latency when the sending rate reached around 200 TPS.

This study shows that the proposed scheme is a promising solution for a large-scale IoT network. Moreover, extensive experimental results demonstrate that the proposed model can be deployed to track provenance metadata with competitive throughput and latency while maintaining low computation and communication overheads. Integrating the proposed scheme with a distributed database such as Apache Cassandra to store transaction data with more detailed performance evaluations and developing a sharding-based consensus that handles the network partitions are future research directions.

The future works may include developing a framework to support more features including MQTT-based communication between blockchain, IoT sensors and Hadoop off-chain storage to store transaction data. Besides, the future works could categorize IoT data types and match them with feasible frameworks within the Hadoop ecosystem through integration with the proposed blockchain model.

Acknowledgements

Not applicable.

Authors' contributions

HHP was the main contributor of this work. He has done the literature review, experiments, data collection, prepare results, and drafted the manuscript. MR worked closely with HHP as the principal research supervisor, review, analyze, and manuscript preparation. FA and SD helped to improve the final paper. All authors read and approved the final manuscript.

Funding

This work was not funded.

Availability of data and materials

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

¹Department of Mechanical & Electrical Engineering, School of Food and Advanced Technology, Massey University, Auckland 0632, New Zealand. ²School of Science and Technology, Sunway University, 47500 Sunway, Selangor, Malaysia.

Received: 5 May 2021 Accepted: 20 August 2021

Published online: 30 August 2021

References

1. Marketsandmarkets: "Big Data Market by Component, Deployment Mode, Organization Size, Business Function (Operations, Finance, and Marketing and Sales), Industry Vertical (BFSI, Manufacturing, and Healthcare and Life Sciences), and Region - Global Forecast to 2025" (Accessed on 20 January 2021). online: <https://www.researchandmarkets.com/r/8ww41e>
2. Dedeoglu V, Jurdak R, Dorri A, Lunardi R, Michelin R, Zorzo A, Kanhere S. Blockchain technologies for iot. In: *Advanced Applications of Blockchain Technology*, pp. 55–89. Springer, ??? 2020.
3. Gantz J, Reinsel D. Extracting value from chaos. IDC iVIEW. 2011;1142(2011):1–12.
4. Pouyanfar S, Yang Y, Chen S-C, Shyu M-L, Iyengar S. Multimedia big data analytics: a survey. *ACM Comput Surv (CSUR)*. 2018;51(1):1–34.
5. Jain P, Gyanchandani M, Khare N. Enhanced secured map reduce layer for big data privacy and security. *J Big Data*. 2019;6(1):1–17.
6. Abouelmehdi K, Beni-Hessane A, Khaloufi H. Big healthcare data: preserving security and privacy. *J Big Data*. 2018;5(1):1–18.
7. Surjandari I, Yusuf H, Laoh E, Maulida R. Designing a permissioned blockchain network for the halal industry using hyperledger fabric with multiple channels and the raft consensus mechanism. *J Big Data*. 2021;8(1):1–16.
8. Baig MI, Shuib L, Yadegaridehkordi E. Big data adoption: state of the art and research challenges. *Inf Process Manag*. 2019;56(6):102095.
9. Honar Pajooh H, Rashid M, Alam F, Demidenko S. Multi-layer blockchain-based security architecture for internet of things. *Sensors*. 2021;21(3):772.
10. Honar Pajooh H, Rashid M, Alam F, Demidenko S. Hyperledger fabric blockchain for securing the edge internet of things. *Sensors*. 2021;21(2):359.
11. Deepa N, Pham Q-V, Nguyen DC, Bhattacharya S, Prabadevi B, Gadekallu TR, Maddikunta PKR, Fang F, Pathirana PN. A survey on blockchain for big data: Approaches, opportunities, and future directions. *arXiv preprint arXiv:2009.00858* 2020.
12. Rawat DB, Doku R, Garuba M. Cybersecurity in big data era: from securing big data to data-driven security. *IEEE Trans Ser Comput*. 2019.
13. Liu CH, Lin Q, Wen S. Blockchain-enabled data collection and sharing for industrial iot with deep reinforcement learning. *IEEE Trans Ind Inf*. 2018;15(6):3516–26.
14. Xu X, Zhang X, Gao H, Xue Y, Qi L, Dou W. Become: blockchain-enabled computation offloading for iot in mobile edge computing. *IEEE Trans Ind Inf*. 2019;16(6):4187–95.
15. Liu G, Dong H, Yan Z, Zhou X, Shimizu S. B4sdc: a blockchain system for security data collection in manets. *IEEE Trans Big Data*. 2020.

16. Yang R, Yu FR, Si P, Yang Z, Zhang Y. Integrated blockchain and edge computing systems: a survey, some research issues and challenges. *IEEE Commun Surv Tutor*. 2019;21(2):1508–32.
17. Pahl C, El Ioini N, Helmer S, Lee B. An architecture pattern for trusted orchestration in iot edge clouds. In: 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), 2018; 63–70. IEEE.
18. Agiwal M, Roy A, Saxena N. Next generation 5g wireless networks: a comprehensive survey. *IEEE Commun Surv Tutor*. 2016;18(3):1617–55.
19. Wood G, et al. Ethereum: a secure decentralised generalised transaction ledger. *Ethereum project yellow paper*. 2014;151(2014):1–32.
20. Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, De Caro A, Enyeart D, Ferris C, Laventman G, Manevich Y, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In: *Proceedings of the Thirteenth EuroSys Conference*, 2018; 1–15.
21. Schwartz D, Youngs N, Britto A, et al. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*. 2014;5(8):151.
22. Jindal A, Kumar N, Singh M. A unified framework for big data acquisition, storage, and analytics for demand response management in smart cities. *Future Gener Comput Syst*. 2020;108:921–34.
23. Cutting MCD. "Apache Hadoop." <http://hadoop.apache.org>. Accessed 15 Feb 2021.
24. Borthakur D. The hadoop distributed file system: architecture and design. *Hadoop Proj Website*. 2007;11(2007):21.
25. MongoDB: "MongoDB A complete data framework." <https://www.mongodb.com/>. Accessed 20 Feb 2021.
26. Spark: "Apache Spark™ is a unified analytics engine for large-scale data processing." <https://spark.apache.org/>. Accessed 15 Feb 2021.
27. Storm: "Apache Storm." <https://storm.apache.org/>. Accessed 15 Feb 2021.
28. Caro MP, Ali MS, Vecchio M, Giuffreda R. Blockchain-based traceability in agri-food supply chain management: a practical implementation. In: 2018 IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany), 2018; 1–4. IEEE.
29. Javaid U, Aman MN, Sikdar B. Blockpro: Blockchain based data provenance and integrity for secure iot environments. In: *Proceedings of the 1st Workshop on Blockchain-enabled Networked Sensor Systems*, 2018; 13–18.
30. Liang X, Shetty S, Tosh D, Kamhoua C, Kwiat K, Njilla L. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2017; 468–477. IEEE.
31. Sigwart M, Borkowski M, Peise M, Schulte S, Tai S. A secure and extensible blockchain-based data provenance framework for the internet of things. *Personal and Ubiquitous Computing*, 2020;1–15.
32. Soldatos J, Kefalakis N, Hauswirth M, Serrano M, Calbimonte J-P, Riahi M, Aberer K, Jayaraman PP, Zaslavsky A, Žarko IP, et al. Openiot: Open source internet-of-things in the cloud. In: *Interoperability and Open-source Solutions for the Internet of Things*, 2015;13–25. Springer.
33. Yang C, Chen X, Xiang Y. Blockchain-based publicly verifiable data deletion scheme for cloud storage. *J Netw Comput Appl*. 2018;103:185–93.
34. Li J, Wu J, Chen L. Block-secure: blockchain based scheme for secure p2p cloud storage. *Inf Sci*. 2018;465:219–31.
35. Zhu L, Wu Y, Gai K, Choo K-KR. Controllable and trustworthy blockchain-based cloud data management. *Future Gener Comput Syst*. 2019;91:527–35.
36. Liang X, Shetty SS, Tosh D, Njilla L, Kamhoua CA, Kwiat K. Provchain: blockchain-based cloud data provenance. *Blockchain for Distrib Syst Secur*. 2019;69.
37. Tosh D, Shetty S, Liang X, Kamhoua C, Njilla LL. Data provenance in the cloud: a blockchain-based approach. *IEEE Consumer Electr Mag*. 2019;8(4):38–44.
38. Gai K, Wu Y, Zhu L, Xu L, Zhang Y. Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks. *IEEE Internet Things J*. 2019;6(5):7992–8004.
39. Tuli S, Mahmud R, Tuli S, Buyya R. Fogbus: a blockchain-based lightweight framework for edge and fog computing. *J Syst Softw*. 2019;154:22–36.
40. Ren Y, Leng Y, Cheng Y, Wang J. Secure data storage based on blockchain and coding in edge computing. *Math Biosci Eng*. 2019;16(4):1874–92.
41. Muthanna A, A Ateya A, Khakimov A, Gudkova I, Abuarqoub A, Samouylov K, Koucheryavy A. Secure and reliable iot networks using fog computing with software-defined networking and blockchain. *J Sensor Actuator Netw*. 2019;8(1):15.
42. Yue D, Li R, Zhang Y, Tian W, Peng C. Blockchain based data integrity verification in p2p cloud storage. In: 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), 2018; 561–568. IEEE.
43. Wang J, Peng F, Tian H, Chen W, Lu J. Public auditing of log integrity for cloud storage systems via blockchain. In: *International Conference on Security and Privacy in New Computing Environments*, 2019; 378–387. Springer.
44. Zhang Y, Xu C, Lin X, Shen XS. Blockchain-based public integrity verification for cloud storage against procrastinating auditors. *IEEE Trans Cloud Comput*. 2019.
45. Docker I. Docker. *linea*. [Junio de 2017]. Disponible en: <https://www.docker.com/what-docker> 2017.
46. Hyperledger: "Hyperledger fabric client sdk for node.js." <https://github.com/hyperledger/fabric-sdk-node> Accessed 25 Feb 2021.
47. Locust: Locust: an open source load testing tool. <https://locust.io/>. Accessed 1 Mar 2021.
48. Performance H, Group SW. "Hyperledger Blockchain Performance Metrics." https://www.hyperledger.org/wp-content/uploads/2018/10/HL_Whitepaper_Metrics_PDF_V1.01.pdf. Accessed: 15 February 2020.
49. Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: concept and applications. *ACM Trans Intell Syst Technol (TIST)*. 2019;10(2):1–19.
50. Moses O. Adams Manual: Tire Models, using the Fiala handling force model. <https://github.com/OlafenwaMoses/ImageAI> Accessed 1 Feb 2021.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Chapter 7

Conclusion and Future Research Directions

Security methods that only work on a centralized trust server basis cannot exploit the advantage of emerging cloud and edge computing and make a single point of failure problems in the system. Resource-constrained IoT devices are not capable of deploying fully distributed approaches due to the additional overhead on individual IoT devices. The work considers protecting the security and privacy of IoT infrastructure by implementing a locally centralized authentication and authorization of IoT devices. The IoT edge computing nodes are globally distributed in a trustful environment facilitated by implementing blockchain technology.

7.1 Conclusions

In this thesis, a Multi-layer Blockchain System (MBS) is proposed to provide the security and privacy of the 5G-enabled IoT network in a decentralized and distributed structure. In the multi-layer architecture, devices in each layer have different computational capabilities and energy storage capacities. Consequently, different security strategies are proposed for individual layers. The blockchain implementation is modified to suit the devices of each particular layer. The clustering concept is the key to achieving the multi-layer architecture, where the cluster heads form the multi-layer structure. A self-clustering method is proposed in this work to identify Cluster Head (CH) nodes. It offers reduced energy consumption and higher throughput. Genetic

algorithms considering various clustering factors, including geospatial ones (e.g., the distance between nodes, the base-station distance to nodes) and total network energy, are proposed. A fitness function simulating network changes and node movements within the network is optimized by deploying the SA methodology.

By leveraging emerging network frameworks built on IoT edge computing and distributed blockchain system, the proposed architecture reaches much higher availability which is a critical measure in the safety of the IoT system. The availability is guaranteed even under failures of local authorization entities running on the edge layer. We expect the possibility of integrating the heterogeneous IoT devices and infrastructures, ranging from sensor objects to electric power grid control systems, into the distributed multi-layer architecture enriched by implementing the blockchain to facilitate various security alternatives. The proposed approach enables large-scale IoT and D2D communications deployment to address the challenges associated with increasing data traffic.

The main achievements and contributions of this thesis are summarized as follows:

1. Proposed a multi-layer security model for IoT devices functioning under multi-hop cellular networks based on distributed technology of the blockchain. The suggested model provides a feasible solution to establish the decentralized application of blockchain technology to secure the cellular-enabled IoT network. The hybrid self-clustering EC algorithm, utilizing GA and SA, is developed to fragment the IoT network into clusters in order to provide the multi-layer structure and enhance the network lifetime. The multi-layer model improves network security, lowers the processing load, and reduces network load and latency. The proposed implementation enhances the efficiency of the communications via the peer-to-peer nature of the blockchain communication and maps it to the device-to-device communication in cellular systems with improved integrity and security. The proposed solution tackles the IoT security challenges, including framework privacy, authentication, heterogeneity, flexibility, and network scalability. The proposed hybrid clustering algorithm has been compared with four existing protocols. The simulation results show that the proposed algorithm outperforms the competitors in terms of various performance metrics, including network load, network coverage, and distances. The performance of the proposed multi-layer blockchain-based framework was evaluated. It was found that the lightweight blockchain was more effective than the global blockchain Ethereum.

The main achievements from this part of the research are:

- A novel method of authentication and authorization of IoT nodes is implemented to provide IoT devices security and protect device communications through a multi-layer structure.
 - Proposed a novel, lightweight, private multi-layer model for reducing the complexity of blockchain technology implementation while improving network scalability. The proposed model is tailored to meet the requirements of IoT devices by adopting blockchain technology to suit different layers of the IoT system. The simulation study shows that the proposed Hyperledger Fabric-based method outperforms a traditional blockchain solution, like the Ethereum, in terms of latency and throughput.
 - The qualitative and quantitative performance evaluations were undertaken to analyze the proposed solution. Simulation results indicate that the multi-layer framework approach improves latency, throughput, processing time, and packet overhead compared to existing blockchain approaches.
2. Presented an implementation of the secured HLF blockchain platform as a permissioned blockchain technology integrated with edge IoTs to test and analyze the performance of the proposed blockchain-based multi-layer IoT security model. The proposed approach aimed to provide security to massively interconnected IoT devices while ensuring the scalability of IoT systems with minimum resource requirements. Besides, the heterogeneity and diversity of connected devices within the IoT realm are considered. The presented proof of concept was implemented using two different environment setups on the Raspberry Pi devices and VMware Virtual desktops. The performance metrics, such as transaction throughput, transaction latency, computational resources, and network use of the implemented networks, were evaluated. The presented findings indicate a significantly optimal throughput for IoT applications. Peers' and clients' processes are the primary source of resource consumption in the network. The Orderer and ChainCode use fewer resources compared to the peer process. Experimental results show a significant increase in throughput of approximately six times compared to the optimal scale implementation of HLF. The empirical results all indicate low overhead for running the proposed model.

The main achievements from this part of the research are:

- A novel architecture for the security and privacy of IoT edge computing using a per-

- missioned blockchain is proposed. The proposed architecture considers 5G-enabled IoT technologies for node communications. The architecture is suitable for real-world IoT systems due to the developed ChainCodes that facilitate storage and retrieval of data in a tamper-proof blockchain system. Moreover, blockchain-based data traceability for 5G-enabled edge computing using the HLF is designed to provide auditability of the IoT metadata through a developed NodeJS client library.
- The adaptability of the Hyperledger Fabric for ARM architecture of the edge IoT devices is improved by modifying official docker images from the source as there are no official or public images of HLF to support the 64-bit ARMv8 architecture.
 - Designed a lightweight mutual authentication and authorization model to facilitate a secure and privacy-preserving framework for IoT edge that protects the sensor nodes' sensitive data through a permissioned fabric platform. Furthermore, it provides trust for the IoT sensors, edge nodes, and base stations by the private blockchain. This is achieved by using the edge nodes to record the IoT data in an immutable and verifiable ledger to guarantee metadata traceability and auditability.
 - Performance characteristics of the proposed approach in terms of throughput, transaction latency, computational resources, network use, and communication costs are experimentally evaluated in two network setups. The findings indicate a significantly optimal throughput for IoT applications. Peers' and clients' processes are the primary source of resource consumption in the network. The Orderer and ChainCode use fewer resources compared to the peer process. Experimental results show a significant increase in throughput of approximately six times compared to the optimal scale implementation of HLF.
3. Proposed a detailed experimental performance analysis for the scalable HLF blockchain platform in a distributed large network with varying numbers of nodes and workloads. Presented a scalable and distributed framework for precise and real-time monitoring of HLF systems. It offers significantly lower overhead and more details about the various HLF system metrics. We conducted a comprehensive performance analysis and evaluation of well-known HLF blockchain systems with different network configurations, network load levels, node numbers, and batch sizes. The system performance evaluation is shown in terms of throughput, latency, block size, network size, endorsement policy, CPU usage, memory consumption, disk write, In/Out traffic, and scalability. The experimental results

indicate the feasibility of the proposed framework. However, the throughput, latency, and scalability of a blockchain framework depend on hardware configuration, blockchain network design, and smart contract complexity operations. The findings demonstrate that the system throughput is sensitive to the Orderer setting. The results also indicate that the change in the number of endorsements influences the performance metrics significantly. The smaller number of endorsements results in better performance. It is shown that one of the main bottlenecks is how the committing peers utilize multiple vCPUs presented in the system to do parallel transactions that need to be optimized to improve the system performance. The size of transactions significantly impacts throughput and transaction latencies.

The main achievements from this part of the research are:

- Considered a detailed and real-time performance computation model for HLF blockchain systems to address the challenges mentioned earlier. A comprehensive review of blockchain research topics shows that latency, throughput, and scalability are the primary limiting characteristics. The selected performance computation model facilitates collecting real-time performance data by analyzing logs and the daemon process.
 - A comprehensive experimental analysis of various HLF performance metrics is carried out to explore overall and detailed system performance and provide the system configuration guidelines to attain the maximum performance. This research work proposes a scalable model for real-time performance computation of the Fabric platform, considering lower overheads and better scalability. The experiments are deployed in a distributed network with a varying number of peers and different network sizes to identify the significant performance bottlenecks.
 - Studied the comprehensive metrics measurements to measure and monitor the impact of system configurations (e.g., number of transactions, block sizes, endorsement policies, network size) on the HLF performance, especially the scalability. The results show the possibilities and limitations of HLF implementation in large-scale IoT networks. This study could benefit various application domains in selecting the best blockchain platforms that would fit specific applications.
4. Proposed a layer-based distributed data storage design and implementation of a blockchain-enabled large-scale IoT system. The need for a centralized server and a third-party auditor

has been eliminated by leveraging HLF peers that perform transaction verifications and records audits in a big data system with the help of blockchain technology. The HLF blockchain facilitates storing the lightweight verification tags on the blockchain ledger. In contrast, the actual metadata are stored in the off-chain big data system to reduce the communication overheads and enhance data integrity. Additionally, a prototype has been implemented on embedded hardware showing the feasibility of deploying the proposed solution in IoT edge computing and big data ecosystems. Finally, experiments have been conducted to evaluate the performance of the proposed scheme in terms of throughput, latency, communication, and computation costs. The obtained results have indicated the feasibility of the proposed solution to retrieve and store the provenance of large-scale IoT data within the big data ecosystem using the HLF blockchain.

The main achievements from this part of the research are:

- HLF blockchain scheme is developed to provide secure data storage for big data systems in a large-scale IoT network. The proposed model maintains data privacy preservation, ensures secure connection to a big data system through the HLF network, and guarantees data collection security. The centralized trust server is eliminated through implementing the HLF blockchain technology.
- A two-layer security framework is proposed that involves HLF blockchain and a big data system. Trusted entities are linked to HLF, and third-party auditing parties are eliminated to reduce the compromised auditor's risk. The network scalability is enhanced by incorporating edge computing to maintain IoT data computation as well as to collect and forward data to the blockchain and off-chain storage.
- A model is proposed to store the lightweight verification checksums and data pointers in the blockchain ledger to reduce the communication and computation overheads. The HLF blockchain performs data provenance while the actual metadata are stored in off-chain storage after being verified by the blockchain. Extensive experiments were conducted through a prototype implementation on a Hadoop system to evaluate the performance of the proposed scheme in terms of throughput, response time, latency, communication, and computation cost.

7.2 Future Research Directions

With the rapid growth in blockchain integration into IoT ecosystems, there are still challenges that need to be addressed to enhance the security environment of IoT and D2D devices with blockchain technology. The multi-layer framework enables increasing the scalability of blockchain by forming layers and clusters where only cluster heads maintain the blockchain operations, including storing transactions and blocks. In this thesis, the clustering techniques are implemented based on evolutionary algorithms to form the clusters overlay in distribute manner. However, the clusters in the IoT network can be organized based on other parameters and features that can improve the performance of the blockchain. One possible solution could deploy the average number of transactions that may further enhance the load balancing on each cluster head. Besides, the blockchain-based clustering algorithm needs to reduce the delay in adding or removing a CH to the network.

The local authentication and authorization, which has been deployed in layer-based architecture, provide a trustful environment while reducing the processing overhead for new block verifications. The risk of generating fake transactions is minimized within this ecosystem. However, the generalized approach is required to analyze the minimum number of CH within the system to eliminate security risks in the blockchain. Additionally, the multi-layer implementation can also be extended to study the security of the system against a wide-ranging possible threats and attacks.

This work has implemented a prototype of the HLF blockchain in real-world edge IoT settings to highlight the system performance and limitations. As a future research direction, developing the layer-based architecture on top of other existing blockchains, e.g., Ethereum, can be further explored, enabling the users of particular blockchain solutions to use their specific IoT applications.

Blockchain protection against malicious nodes or miners who modify the ledger or remove transactions needs to be investigated. This issue still exists in centralized cloud servers where malicious nodes can maintain a copy of private user data. A robust solution to address this challenge may potentially enhance blockchain privacy.

Automating some of the CH operations through implementing smart contracts can enhance

the distributed control within the integrated blockchain-based IoT system while reducing the processing and packet overheads. Besides, a consensus mechanism is required to manage these smart contracts, which need to be further investigated on how to achieve the consensus among the affected nodes.

This work has proposed a layer-based distributed data storage design and implementation of a blockchain-enabled large-scale IoT system to store the lightweight verification tags on the blockchain ledger without relying on trusted third parties. This approach mitigates the trust challenges associated with the centralized server solutions. The blockchain-based solution has the potential to address security, transparency, anonymity, and auditability challenges. The work studied the performance of the proposed model with the big data ecosystem by implementing a prototype using edge computing devices and several Raspberry Pi devices. As future work, comprehensive deployment of the proposed model can be considered for large-scale real-world scenarios to explore all fundamental features of the model. Furthermore, the anonymous authentication mechanism and routing algorithms may also be studied.

Appendix A

Statement of Contribution Forms (DRC 16)



MASSEY UNIVERSITY
GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Houshyar Honar Pajooch	
Name/title of Primary Supervisor:	Dr Mohammad AbdurRashid	
Name of Research Output and full reference:		
Honar Pajooch, H., Rashid, M., Alam, F., & Demidenko, S. (2021). Multi-Layer Blockchain-Based Security Architecture for Internet of Things. <i>Sensors</i> , 21(3), 772. doi:10.3390/s21030772		
In which Chapter is the Manuscript /Published work:	Chapter 3	
Please indicate:		
<ul style="list-style-type: none"> The percentage of the manuscript/Published Work that was contributed by the candidate: 	80%	
and		
<ul style="list-style-type: none"> Describe the contribution that the candidate has made to the Manuscript/Published Work: 	The candidate performed the experiments with assistance from the second author. He produced the first draft of the article based on suggestion from the co-author supervisors regarding the narrative and results that was presented.	
For manuscripts intended for publication please indicate target journal:		
Candidate's Signature:	Houshyar Honar Pajooch	<small>Digitally signed by Houshyar Honar Pajooch DN: cn=Houshyar Honar Pajooch, c=NZ, o=Massey University, ou=School of Food and Advanced Technology, email=h.pajooch@massey.ac.nz Date: 2021.09.27 21:12:02 +02'00'</small>
Date:	27/09/2021	
Primary Supervisor's Signature:	Mohammad Rashid	<small>Digitally signed by Mohammad Rashid DN: cn=Mohammad Rashid, c=NZ, o=Massey University, ou=MEE at F&AT, email=m.a.rashid@massey.ac.nz Date: 2021.09.29 10:26:51 +13'00'</small>
Date:	29/09/2021	

(This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis)



MASSEY UNIVERSITY
GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Houshyar Honar Pajooch	
Name/title of Primary Supervisor:	Dr Mohammad AbdurRashid	
Name of Research Output and full reference:		
Honar Pajooch, H., Rashid, M., Alam, F., & Demidenko, S. (2021). Hyperledger Fabric Blockchain for Securing the Edge Internet of Things. <i>Sensors</i> , 21(2), 359. doi:10.3390/s21020359		
In which Chapter is the Manuscript /Published work:	Chapter 4	
Please indicate:		
<ul style="list-style-type: none"> The percentage of the manuscript/Published Work that was contributed by the candidate: 	80%	
and		
<ul style="list-style-type: none"> Describe the contribution that the candidate has made to the Manuscript/Published Work: 	The candidate performed the experiments with assistance from the second author. He produced the first draft of the article based on suggestion from the co-author supervisors regarding the narrative and results that was presented.	
For manuscripts intended for publication please indicate target journal:		
Candidate's Signature:	Houshyar Honar Pajooch	<small>Digitally signed by Houshyar Honar Pajooch DN: cn=Houshyar Honar Pajooch, c=NZ, o=Massey University, ou=School of Food and Advanced Technology, email=h.pajooch@massey.ac.nz Date: 2021.09.27 21:13:19 +02'00'</small>
Date:	27/09/2021	
Primary Supervisor's Signature:	Mohammad Rashid	<small>Digitally signed by Mohammad Rashid DN: cn=Mohammad Rashid, c=NZ, o=Massey University, ou=MEE at F&AT, email=m.a.rashid@massey.ac.nz Date: 2021.09.29 10:28:58 +13'00'</small>
Date:	29/09/2021	

(This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis)



MASSEY UNIVERSITY
GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Houshyar Honar Pajooch	
Name/title of Primary Supervisor:	Dr Mohammad AbdurRashid	
Name of Research Output and full reference:		
Honar Pajooch, H., Rashid, M.A., Alam, F. et al. Experimental Performance Analysis of Scalable Distributed Hyperledger Fabric in a Large Scale IoT Testbed. IEEE IoT Journal		
In which Chapter is the Manuscript /Published work:	Chapter 5	
Please indicate:		
<ul style="list-style-type: none"> The percentage of the manuscript/Published Work that was contributed by the candidate: 	80%	
and		
<ul style="list-style-type: none"> Describe the contribution that the candidate has made to the Manuscript/Published Work: 	The candidate performed the experiments with assistance form the second author. He produced the first draft of the article based on suggestion form the co-author supervisors regarding the narrative and results that was presented.	
For manuscripts intended for publication please indicate target journal:		
IEEE Internet of Things Journal		
Candidate's Signature:	Houshyar Honar Pajooch	<small>Digitally signed by Houshyar Honar Pajooch DN: cn=Houshyar Honar Pajooch, c=NZ, o=Massey University, ou=School of Food and Advanced Technology, email=h.pajooch@massey.ac.nz Date: 2021.09.27 21:16:07 +02'00'</small>
Date:	27/09/2021	
Primary Supervisor's Signature:	Mohammad Rashid	<small>Digitally signed by Mohammad Rashid DN: cn=Mohammad Rashid, c=NZ, o=Massey University, ou=MEE at F&AT, email=m.a.rashid@massey.ac.nz Date: 2021.09.29 10:29:52 +13'00'</small>
Date:	29/09/2021	

(This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis)



MASSEY UNIVERSITY
GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Houshyar Honar PajooH	
Name/title of Primary Supervisor:	Dr Mohammad AbdurRashid	
Name of Research Output and full reference:		
Honar PajooH, H., Rashid, M.A., Alam, F. et al. IoT Big Data provenance scheme using blockchain on Hadoop ecosystem. J Big Data 8, 114 (2021).		
In which Chapter is the Manuscript /Published work:	Chapter 6	
Please indicate:		
<ul style="list-style-type: none"> The percentage of the manuscript/Published Work that was contributed by the candidate: 	80%	
and		
<ul style="list-style-type: none"> Describe the contribution that the candidate has made to the Manuscript/Published Work: 	The candidate performed the experiments with assistance form the second author. He produced the first draft of the article based on suggestion form the co-author supervisors regarding the narrative and results that was presented.	
For manuscripts intended for publication please indicate target journal:		
Candidate's Signature:	Houshyar Honar PajooH	<small>Digitally signed by Houshyar Honar PajooH DN: cn=Houshyar Honar PajooH, c=NZ, o=Massey University, ou=School of Food and Advanced Technology, email=h.pajooH@massey.ac.nz Date: 2021.09.27 21:16:46 +02'00'</small>
Date:	27/09/2021	
Primary Supervisor's Signature:	Mohammad Rashid	<small>Digitally signed by Mohammad Rashid DN: cn=Mohammad Rashid, c=NZ, o=Massey University, ou=MEE at F&AT, email=m.a.rashid@massey.ac.nz Date: 2021.09.29 10:30:26 +13'00'</small>
Date:	29/09/2021	

(This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis)

Additional References

- [1] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, “Smart locks: Lessons for securing commodity internet of things devices,” in *Proceedings of the 11th ACM on Asia conference on computer and communications security*, 2016, pp. 461–472.
- [2] M. Amoozadeh, A. Raghuramu, C.-N. Chuah, D. Ghosal, H. M. Zhang, J. Rowe, and K. Levitt, “Security vulnerabilities of connected vehicle streams and their impact on cooperative driving,” *IEEE Communications Magazine*, vol. 53, no. 6, pp. 126–132, 2015.
- [3] J. Fruhlinger, “The mirai botnet explained: How teen scammers and cctv cameras almost brought down the internet,” *CSO Online*, 2018.
- [4] T. Dierks and E. Rescorla, “The transport layer security (tls) protocol version 1.2,” 2008.
- [5] A. F. Skarmeta, J. L. Hernandez-Ramos, and M. V. Moreno, “A decentralized approach for security and privacy challenges in the internet of things,” in *2014 IEEE world forum on Internet of Things (WF-IoT)*. IEEE, Conference Proceedings, pp. 67–72.
- [6] H. Gross, M. Hölbl, D. Slamanig, and R. Spreitzer, “Privacy-aware authentication in the internet of things,” in *International Conference on Cryptology and Network Security*. Springer, Conference Proceedings, pp. 32–39.
- [7] A. Ukil, S. Bandyopadhyay, and A. Pal, “Iot-privacy: To be private or not to be private,” in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, Conference Proceedings, pp. 123–124.
- [8] M. S. Hossain, G. Muhammad, S. M. M. Rahman, W. Abdul, A. Alelaiwi, and A. Alamri, “Toward end-to-end biometrics-based security for iot infrastructure,” *IEEE Wireless Communications*, vol. 23, no. 5, pp. 44–51, 2016.

- [9] S. R. Moosavi, T. N. Gia, A.-M. Rahmani, E. Nigussie, S. Virtanen, J. Isoaho, and H. Tenhunen, "Sea: a secure and efficient authentication and authorization architecture for iot-based healthcare using smart gateways," *Procedia Computer Science*, vol. 52, pp. 452–459, 2015.
- [10] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "Iot security: ongoing challenges and research opportunities," in *2014 IEEE 7th international conference on service-oriented computing and applications*. IEEE, Conference Proceedings, pp. 230–234.
- [11] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, Conference Proceedings, pp. 839–858.
- [12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Report, 2019.
- [13] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.
- [14] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control," *Journal of medical systems*, vol. 40, no. 10, pp. 1–8, 2016.
- [15] E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, and C. Weinhardt, "A blockchain-based smart grid: towards sustainable local energy markets," *Computer Science-Research and Development*, vol. 33, no. 1, pp. 207–214, 2018.
- [16] K. R. Özyilmaz, M. Doğan, and A. Yurdakul, "Idmob: Iot data marketplace on blockchain," in *2018 crypto valley conference on blockchain technology (CVCBT)*. IEEE, Conference Proceedings, pp. 11–19.
- [17] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International workshop on open problems in network security*. Springer, Conference Proceedings, pp. 112–125.
- [18] 2017. [Online]. Available: <https://hyperledger-fabric.readthedocs.io/en/release-1.1/>
- [19] 2021. [Online]. Available: <http://www.linuxfoundation.org>

- [20] S. Huh, S. Cho, and S. Kim, “Managing iot devices using blockchain platform,” in *2017 19th international conference on advanced communication technology (ICACT)*. IEEE, Conference Proceedings, pp. 464–467.
- [21] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, and Y. Manevich, “Hyperledger fabric: a distributed operating system for permissioned blockchains,” in *Proceedings of the thirteenth EuroSys conference*, Conference Proceedings, pp. 1–15.
- [22] 2021. [Online]. Available: <https://www.blockchain.com/charts/blocks-size>
- [23] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, “Lsb: A lightweight scalable blockchain for iot security and anonymity,” *Journal of Parallel and Distributed Computing*, vol. 134, pp. 180–197, 2019.
- [24] S. Haber and W. S. Stornetta, “How to time-stamp a digital document,” in *Conference on the Theory and Application of Cryptography*. Springer, 1990, pp. 437–455.
- [25] N. Szabo, “Formalizing and securing relationships on public networks,” *First monday*, 1997.
- [26] M. Abramowicz, “Cryptocurrency-based law,” *Ariz. L. Rev.*, vol. 58, p. 359, 2016.
- [27] R. C. Merkle, “A digital signature based on a conventional encryption function,” in *Conference on the theory and application of cryptographic techniques*. Springer, Conference Proceedings, pp. 369–378.
- [28] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, “Blockchain challenges and opportunities: A survey,” *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.
- [29] J. Debus, “Consensus methods in blockchain systems,” *Frankfurt School of Finance and Management, Blockchain Center, Tech. Rep*, 2017.
- [30] S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, “Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain,” 2018.
- [31] A. Baliga, “Understanding blockchain consensus models,” *Persistent*, vol. 4, pp. 1–14, 2017.
- [32] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, “Repucoin: Your reputation is your power,” *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1225–1237, 2019.

- [33] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling byzantine agreements for cryptocurrencies,” in *Proceedings of the 26th Symposium on Operating Systems Principles*, Conference Proceedings, pp. 51–68.
- [34] D. Mazieres, “The stellar consensus protocol: A federated model for internet-level consensus,” *Stellar Development Foundation*, vol. 32, 2015.
- [35] D. Ongaro and J. Ousterhout, “In search of an understandable consensus algorithm,” in *2014 USENIX Annual Technical Conference (USENIXATC 14)*, Conference Proceedings, pp. 305–319.
- [36] O. Dib, K.-L. Brousmiche, A. Durand, E. Thea, and E. B. Hamida, “Consortium blockchains: Overview, applications and challenges,” *International Journal On Advances in Telecommunications*, vol. 11, no. 1&2, pp. 51–64, 2018.
- [37] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, “Blockchain-enabled smart contracts: architecture, applications, and future trends,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266–2277, 2019.
- [38] C. Cachin, “Architecture of the hyperledger blockchain fabric,” in *Workshop on distributed cryptocurrencies and consensus ledgers*, vol. 310. Chicago, IL, Conference Proceedings.
- [39] V. Buterin, “A next-generation smart contract and decentralized application platform,” *white paper*, vol. 3, no. 37, 2014.
- [40] X. Wang, X. Zha, W. Ni, R. P. Liu, Y. J. Guo, X. Niu, and K. Zheng, “Survey on blockchain for internet of things,” *Computer Communications*, vol. 136, pp. 10–29, 2019.