Abstract

Internet of Things (IoT) is responsible for connecting billions of devices across the world, enabling convenient communication for a wide range of applications. Privacy and Security with minimal performance degradation are crucial to preserving the integrity of such communications, thus forming a compelling research objective. In enterprise IT, identity and access management (IAM) refers to the way toward distinguishing and controlling individual organization elements' duties and access rights to applications. Identity Management is a central function of any IoT ecosystem. Many approaches to develop such functionality utilize a centralized methodology, which lead to privacy issues and a single point of failure. Consortium blockchain technology serves as a potential solution to tackle the limitations of centralized networks. Increased latency of blockchain systems when subjected to heavy load serves as a major drawback to the implementation of these systems. The fog computing paradigm is introduced here to provide the identity management services at the edge of the network to reduce latency. Hence, this work proposes a consortium - blockchain based identity management technique for Internet of Things (IoT) and describes the associated protocols, to address privacy, security and scalability issues in a centralized identity management system. In addition, the system leverages the Fog computing paradigm to address latency and performance issues during heavy load scenarios. In addition, this system is applied to a healthcare scenario where sensors are registered and authenticated at the Fog layer using a decentralized ledger and then the required information is transmitted and stored for further analysis. Evaluation of this work is done by measuring latency and throughput for different payload sizes and query and invocation actions.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
Declaration		*i*
Acknowledgements		*ii*
Abstract		*iii*
List of Figures		
List of Tables		
1	**INTRODUCTION**	
	1.1 General Introduction…….……...8	
	1.2 Problem Statement…….……..8	
	1.3 Objectives of the project…….…….9	
	1.4 Project deliverables…….…....9	
	1.5 Current Scope…….10	
	1.6 Future Scope……...10	
2	**PROJECT ORGANIZATION**	
	2.1 Software Process Models……11	
	2.2 Roles and Responsibilities……..12	
3	**LITERATURE SURVEY**	
	3.1 Introduction……13	
	3.2 Related Works with the citation of the References…….14	
	3.3 Conclusion of Survey……..27	
4	**PROJECT MANAGEMENT PLAN**	
	4.1 Schedule of the Project (Represent it using Gantt Chart).....28	
	4.2 Risk Identification……...28	
5	**SOFTWARE REQUIREMENT SPECIFICATIONS**	
	5.1 Product Overview……...31	
	5.2 External Interface Requirements……....31	
	5.2.1 User Interfaces…….....31	
	5.2.2 Hardware Interfaces…….31	
	5.2.3 Software Interfaces……..…31	
	5.2.4 Communication Interfaces……..….32	
	5.3 Functional Requirements……..…..32	
	5.3.1 Admin and IoT Sensor Nodes Registration……..……..32	

5.3.2 User Authentication…….……33
5.3.3 Verification of Digital Signature…….……33
5.3.4 Data Exchange between IoT device and the User…….……..33

6 **DESIGN**
6.1 Introduction…….34
6.2 Architecture Design…….35
6.3 Graphical User Interface……..35
6.4 Class Diagram and Classes……..36
6.5 Sequence Diagram……...38
6.6 Data flow diagram……...39
6.7 Conclusion……...……39

7 **IMPLEMENTATION**
7.1 Tools Introduction……..40
7.2 Technology Introduction…….40
7.3 Overall view of the project in terms of implementation…….41
7.4 Explanation of Algorithm and how it is been implemented……..43
7.5 Information about the implementation of Modules……45
7.6 Conclusion…….47

8 **TESTING**
8.1 Introduction……...48
8.2 Testing Tools and Environment…….48
8.3 Test cases……49

9 **RESULTS & PERFORMANCE ANALYSIS**
9.1 Result Snapshots……51
9.2 COMPARISION results tables……..58
9.3 Performance analysis – graphs, tables etc…….60

10 **CONCLUSION & SCOPE FOR FUTURE WORK**
10.1 Findings and suggestions……...63
10.2 Significance of the Proposed Research Work……64
10.3 Limitation of this Research Work……..65
10.4 Directions for the Future works…….65

REFERENCES

APPENDICES:
1 Software manual (if required in the project)
2 Data Set
3 Publications
4 Certificates

Chapter I

1.Introduction

1.1 General Introduction

The expected exponential growth of IoT devices in future years raises management issues to be resolved. Cloud computing may not be adequate for a massive scale while fog computing brings the ability to manage the distribution of controllability and manageability. Fog computing is the concept of a network fabric that stretches from the outer edges of where data is created to where it will eventually be stored, whether that's in the cloud or in a customer's data center. Fog is another layer of a distributed network environment which is closely associated with cloud computing and the internet of things (IoT). Blockchain is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system. A blockchain is essentially a digital ledger of transactions that is duplicated and distributed across the entire network of computer systems on the blockchain. Blockchain-based technology has raised hypes in implementing applications in trust-less environments. We thus propose a blockchain-based architecture for scalable control of IoT devices. Moreover, smart contracts are developed to facilitate the ledger update process.

1.2 Problem Statement

This work proposes a consortium - blockchain based IoT identity management technique and describes the associated protocols, in order to address privacy, security and scalability issues in a centralized identity management system. The fog computing paradigm is introduced here to provide the identity management services closer to the IoT devices in order to improve performance and reduce latency. In addition, this system is applied to a healthcare scenario where we register and authenticate sensors at the Fog layer using a decentralized ledger and then the required information is transmitted and stored for further analysis. Evaluation of the work is done by measuring latency and throughput for different payload sizes and query and invocation actions.

1.3 Objectives of the Project

The objectives of the work are as follows:

- Proposal of a consortium blockchain based IoT identity management technique.
- Presentation of architecture of blockchain system along with description of the protocols for registration and authentication.
- Addressing issues in centralized systems while showing the need for distributing authority of the identity management system to groups of organizations.
- Analyzing the impact of separating the main functionalities of an identity management system on its performance.
- Application of fog computing to sensors in order to reduce cloud processing time.
- Implementation of a prototype of a secure healthcare system showing the viability of the architecture proposed.
- Evaluation of the results for different payload sizes and query and invocation actions.
- Comparison of results obtained with related works.

1.4 Project Deliverables

Identity management forms a critical piece of security infrastructure for any large organization that involves the interaction of several users across widespread client systems. The scaling and integrity requirements of identity management models are strict. Thus, we explore blockchain-based identity management solutions and decentralized identity frameworks. We thus provide:

- An implementation of a blockchain-based identity model for a secure healthcare system using Hyperledger Fabric, the distributed operating system for permissioned blockchain, and RAFT, a consensus algorithm responsible for establishing reliability between consortium members.
- The associated protocols for registration and authentication
- An incorporation of fog computing in order to reduce the latency and make the system more responsive and increase availability, while ensuring security and privacy for data transfer.

1.5 Current Scope

The term "blockchain" is no longer jargon, and the protocol has been applied to a wide range of applications. Blockchain is a decentralized digital ledger that generates a chain of immutable timestamp blocks. Each block is added to the ledger after a mining phase that is validated by the participating nodes. Blockchain removes the need for a trustworthy third party with its distributed record-keeping and authentication scheme. The internet of things, or IoT, emerged in the last decade with the aim of increasing automation in systems where all nodes, machines, and sensor networks are connected everywhere. IoT has made life easier with smart transportation, smart healthcare, smart agriculture, and other smart city approaches. As billions of devices are connected to ever-expanding networks, security remains a major concern. Heavy cryptographic methods are difficult to implement since most devices have limited resources. BC tends to be a solution for IoT security problems, according to recent reports. The current IoT ecosystem is a centralized environment in which devices are controlled, detected, and authenticated from a single location, posing a scalability problem. As a consequence, BC provides a distributed authentication and management system capable of enforcing privacy and security. BC deviates from the centralized system by providing automotive security, authentication, and trust management in both distributed and decentralized IoT ecosystems. Thus, the aim of our research is to propose a consortium blockchain-based IoT identity management system that is further enhanced by using fog computing to perform authentication and verification off chain, resulting in lower latency and overcoming the disadvantages of blockchain-based IoT systems

The scope of this work is limited to:

- Development of a proof-of concept prototype which shows viability of the architecture for business production
- Manual registration of IoT devices to the consortium blockchain network

1.6 Future Scope

This project can be further extended to include a large number of peers and deploy the proposed system in large-scale IoT scenarios. An automatic process to register different

objects can be provided to ease the management of large connected objects. Using a multi-factor authentication method further improves the privacy and security of the identities.

Chapter II

2. Project Organization

2.1 Software Process Model

The Iterative and Incremental SDLC Model is a circular blended technique. The initial planning stage is followed by a small number of steps which are performed repeatedly to improve the system incrementally with the completion of each cycle. The Iterative and Incremental model necessitates comprehensive interaction with users. This model is used in this case since most of the requirements are known up front but are expected to evolve over time, The requirements of the healthcare system are prioritized and it introduces a technology which in our proposed system is an implementation of a blockchain-based identity model for a secure healthcare system using Hyperledger Fabric, the distributed operating system for permissioned blockchain, and RAFT.

The Incremental cycles used to develop the Smart Healthcare System include, requirements adjustment, knowledge acquisition, knowledge engineering, validation and testing and deployment.

Fig. 2.1 SDLC Iterative and Incremental Model

2.2 Roles and Responsibilities

Shruthi Raju	● Project Head ● Identifying suitable consensus algorithm for the use case ● Identifying a suitable platform for implementation, designing the nodes, designing the block chain instance.
S Spandana	● Documentation of the Project.
Ritu Sinha	● Implementation of the authentication and building the APIs which includes generating key pairs and addresses. ● Data authentication through digital signatures and hashes, etc. ● Implementation of the consensus algorithm
Srishti Bijjur	● Design the Admin and user interface ● Strategizing the entire project through work flow structures and estimation techniques ● Testing and performance evaluation. ● Sensor data generation

Chapter III

3. Literature Survey

3.1 Introduction

The job of identity management is expanding to include smart devices, actuators and sensors due to the rapid increase of the Internet of Things. The job entails granting access to IoT resources and applications as well as monitoring sensors. Cloud computing to tackle this exponential increase has been found wanting due to the increased latency and the strain placed on resource constrained IoT devices in order to maintain a communication channel. Thus fog computing enables a solution here.

The fog layer extends the cloud to be closer to the devices that generate and act on data. These devices, called fog nodes, can be deployed in a location with a network connection. Any device with storage, computing and networking capabilities can act as a fog node. Due to its proximity to IoT devices in comparison with the cloud, it keeps sensitive information inside the network and reduces latency.

Blockchain is a system of recording information in a way that employs decentralized hashing and cryptographic methods in order to ensure that the history of the data stored is unalterable. A blockchain is essentially a digital ledger of transactions that is duplicated and distributed across the entire network of computer systems on the blockchain. Blockchain enables transparency of data, risk reduction, and detection of fraudulent transactions and attempts to alter data. It thus is used in many applications in order to provide security in a scalable manner.

The number of blockchain-based solutions that aim to deliver on the principles of identity management is growing rapidly. In the meantime, Distributed ledgers (DLTs) and blockchain technology are two examples of distributed ledgers. Often gaining traction in business settings, widening beyond the scope of "early adopter" and "technology hype". In order to

help the fast-growing IoT network, we propose a blockchain-based architecture for scalable control of IoT devices. Moreover, smart contracts are developed to facilitate the ledger update process. Fog computing is employed in order to reduce latency and provide easier authentication.

3.2 Related Work

Alrawais et.al [1] first discusses the major security and privacy challenges in IoT. It then proposes a mechanism that employs fog to improve the method of distribution of certificate revocation information among IoT devices for enhancement of security. Here, a bloom filter to create a short list that can effectively reduce the certificate revocation list size with acceptable overhead. Fog nodes are used as a gateway to verify revocation status in case of false positives of the bloom filter. The main shortcomings of this scheme is the resource consumption on IoT devices such as storage and communication overhead for IoT devices. The scheme of this system is presented in Fig 3.2.1.

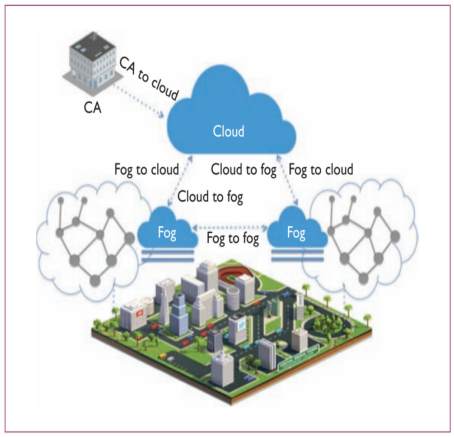


Fig 3.2.1 An overview of the scheme presented by [1]

Harishchandra Dubey et.al [2] proposes a system called Fog Data, which employs fog computing in order to enhance the big data analysis for telehealth applications. It is a cascade of three sub-systems, namely a body sensor network (BSN) for data acquisition, a Fog gateway computer for onsite processing, and the cloud server for storage and back-end analysis. The methods used for analysis of healthcare data on Fog Computer include GNU Zip Compression, Clinical Speech Processing Chain and Dynamic Time Warping. Fog Data

does not use speech features like shimmer, jitter and sensory pleasantness that are useful for quantification of perceptual speech quality in addition to loudness and fundamental frequency for speech disorders. There may also be issues regarding the sensitivity of medical data. The scheme of this software is shown in Fig 3.2.2.

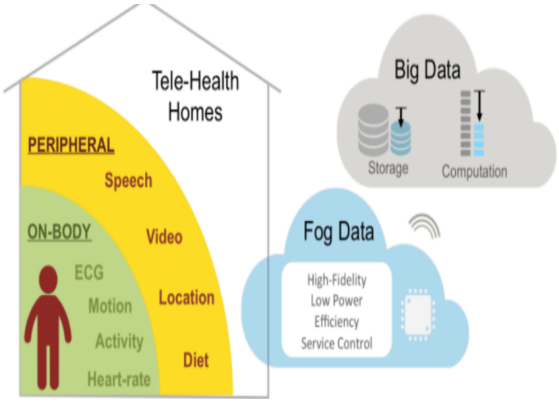


Fig 3.2.2 FogData scheme presented by [2]

L. Zhang et.al [3] proposes a network model here divides the IoT into two layers - edge and high level layers. Edge layers act as an interface to high-level layers and provide an enhanced level of network efficiency as the number of devices in a single edge layer are less and the devices are geographically concentrated. The high-level layers implement the wide area networking feature of IoT. This network model enhances the efficiency of the application of blockchain technology, and also makes the entire network resistant to risks. The future work aims at the analysis of uncertain problems arising in practical applications and the improvement of the network model. Fig 3.2.3 depicts the layers of the network model of the proposed system.

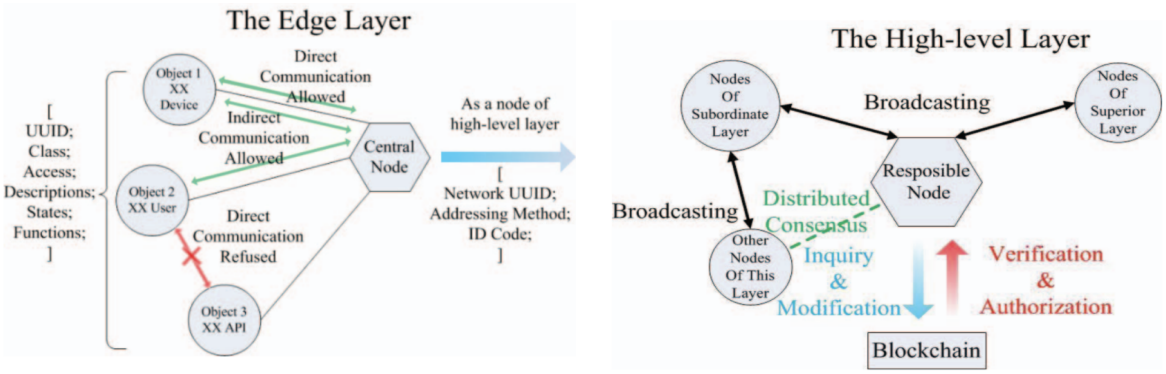


Fig 3.2.3 The layers of the network model depicted in [3]

T. Nguyen Gia *et al*. [4] presents a health monitoring system which enables monitoring of ECG signals of a patient and provides continuous analysis at a low cost. The architecture consists of IoT sensor nodes, fog gateways, and a back-end cloud. Signals are collected using the sensor nodes and then transmitted to the fog gateways via an nRF module which is completely customizable while having a low cost and low power consumption. Based on the information obtained, preprocessing may or may not be performed and the data is then transferred to the Cloud. When using AES-256, the latency of the sensor node and the system increases. Thus if more sophisticated algorithms are applied at the sensor node, latency can increase drastically. Thus security of the system is jeopardized and the system is unable to apply advanced cryptographic techniques. The modules presented in this system are illustrated in Fig 3.2.4.

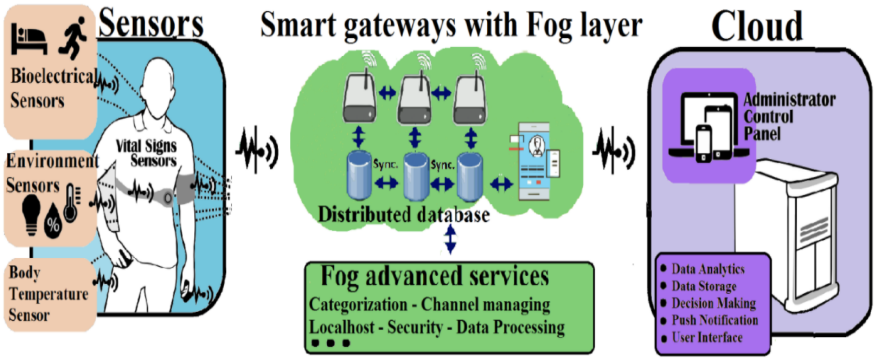


Fig 3.2.4 Depiction of the modules presented by [4]

Parminder Singh et.al. [5] presents a Blockchain and Fog-based Architecture Network for Internet of Everything applications in smart cities. The system consists of two layers. The first layer contains various sensor nodes which are connected to each other and to a fog node. Blockchain provides communication security at this layer. The second layer consists of the fog nodes and is specifically employed to counteract high traffic from devices while reducing latency of responses. This work can be applied to other use cases such as video chatting, 5G/6G streaming applications etc. The system architecture of this work is shown in Fig 3.2.5.

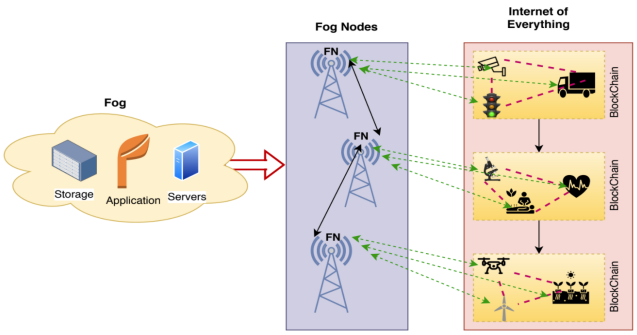


Fig 3.2.5 System architecture of [5]

Omar Ahmad Sghaier and Otman Basir et.al. [6] presented a semi-decentralized Blockchain-based identity management system that provides features of identity creation and authentication. Two smart contracts were described for validation to function like registrar and management contracts. The distributed ledger of Blockchain is used, with smart contracts on top, to build the global registry with specific rules to enable authorized and identifiable entities of the creation of the device identity. The proposed approach defines the identity management framework throughout the device life cycle. This work does not discuss the issues related to scalability and the limitations due to the latency in transactions processing which will speed up its adoption in enterprise applications if they are solved. The overview of the work is shown in Fig 3.2.6.

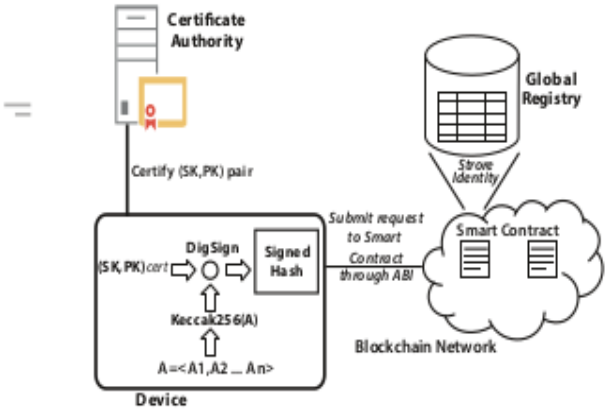


Fig 3.2.6 Identity Creation and Registration Overview presented by [6]

Anang Hudaya Muhamad Amin et.al [7] presented a blockchain based identity management system and integrated it with face recognition in an IoT-cloud application. The authors have

chosen a campus wide printing service as a use case where face recognition is required to use the service. Once a user's login information is digitally verified via immutable blockchain, a smart contract is deployed to keep track of the user's printing activities. The validation of user identity allows access to printing services. The integration of blockchain in this service has decentralized the IDM, automated the service, and kept the service inexpensive and transparent. One main issue is that there has been an increasing demand for outsourcing IoT data to clouds for storage and computation services due to the constrained resources of IoT devices. This dynamic incorporation has brought a series of new challenging security concerns such as identity and access control, authentication, system integrity. The service explained in this work is illustrated in Fig 3.2.7.

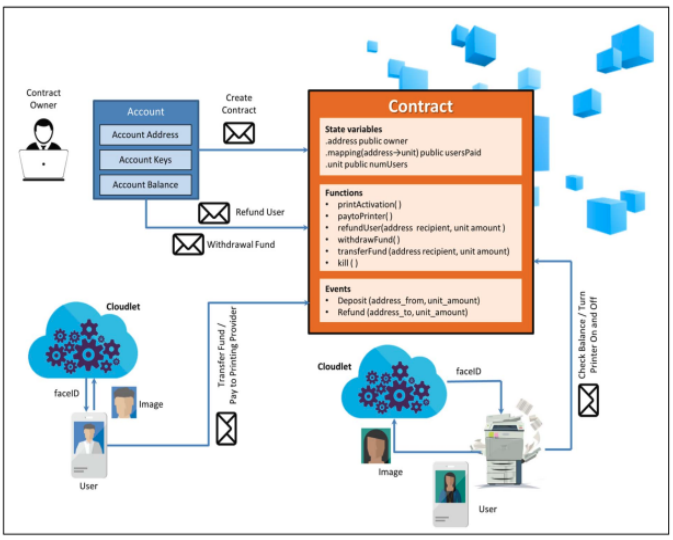


Fig 3.2.7 Decentralized Campus Wide Printing Service with Blockchain-based IDM presented by [7]

In the health sector the privacy of the patients' data has to be maintained. Shobana G and M Suguna et.al. [8] proposed a system, "Insurance Management in Healthcare Sector" that uses blockchain combined with identity management to access the identity of a person when authorized by the person. The insured amount, on successful verification, will then be transferred to the policy holder or the hospital who wishes to claim it. As a result, the insurance claim can reach the policyholder who has initiated the claim process with proof of work. The insurance company which adopts blockchain technology will help in fraud detection in claiming the amount from them. A drawback of this work is the latency

associated with public blockchains such as Ethereum. The architectural flow of the system is shown in Fig 3.2.8.

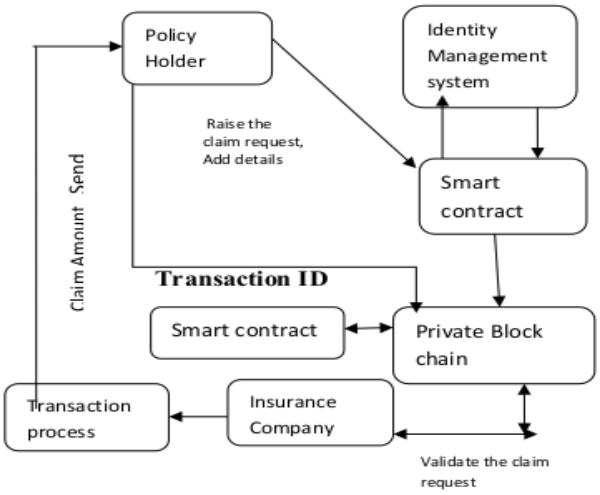


Fig 3.2.8 Architectural flow of Insurance Management System presented by [8]

Zhu Xiaoyang and Youakim Badr et.al. [9] introduced a hybrid architecture for the Internet of Things, combining fog computing to ensure security in the trustless IoT environment. A fog computing security architecture called FOCUS has been proposed, in which each fog node provides services relying on the blockchain technology and the social network paradigm. A smart home testbed has been set up to test the FOCUS. The access control part has been prototyped using SELinux combined with the proposed identity framework and deployed FOCUS on the fog node which is a SELinux enabled system (Debian) running in a Raspberry PI. The proposed fog computing security architecture provided guidelines to secure the IoT. The future work aims at bringing privacy protection to the blockchain identity ledger using zero knowledge proof. Fig 3.2.9 depicts the prototype scenario of this work.

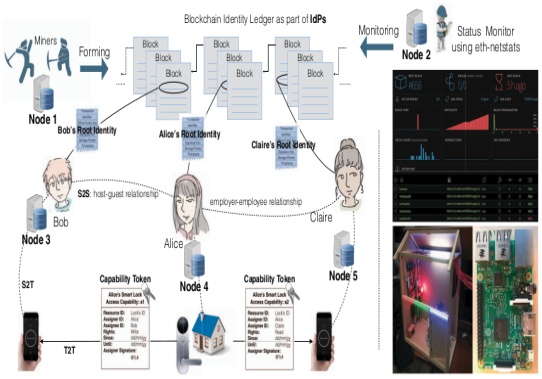


Fig 3.2.9 FOCUS Prototype Scenario presented by [9]

Weiwei Li et.al. [10] proposed a system to build a lightweight and secure blockchain-based IoT identity management system. The authors have integrated a permissioned blockchain, access control that is attribute-based and identity-based signature. The IoT system was divided into IoT domains that function differently. Multiple blockchain nodes were present due to the establishment of blockchain ledger (local) for each domain. The authors then used a policy decision point selection algorithm to select multiple IoT devices so that real-time distributed policy decisions can be achieved. The resource usage and execution of tasks are monitored by the IoT devices. The system architecture is depicted in Fig 3.2.10. It was found that the system was more secure in terms of attacks. But the issue with this proposed system is that as the number of policy decision points increased, the time taken to reach consensus increased exponentially.

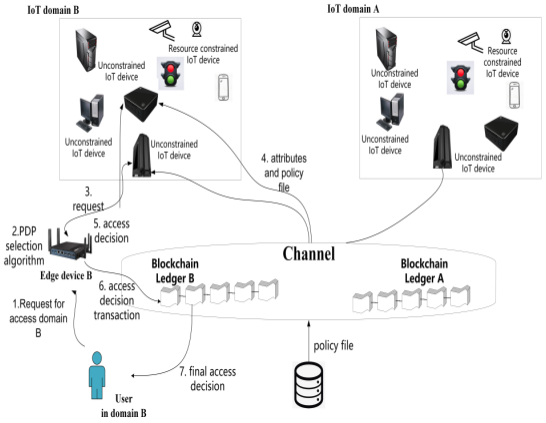


Fig 3.2.10 Blockchain-based IoT Access Control System Architecture presented by [10]

Cícero A. Silva et.al. [11] proposes a system to facilitate better management of the patient's records. This is achieved by storing a subset of the patient's information records closer to the data sources and application. The proposed architecture ensures the security of the patients' data through use of access control techniques. The access times and standard deviations were evaluated and concluded that blockchain with fog and cloud improves the system availability and performance in the analyzed scenario. The main limitations of this system is that, even though including block chain provides better security, it represents a threat since it adds an impact to the time of access to medical records. The average time increased only 42.53 ms compared to the environment that did not use this technology, with 14.74% increase. This

can be looked into in future works. The architectural view of the system is shown in Fig 3.2.11.

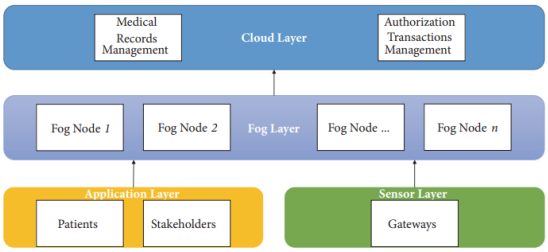


Fig 3.2.11 Architecture's layer decomposition view presented by [11]

O. Bouachir et.al. [12] discusses blockchain and fog computing required to improve cyber physical systems, in terms of quality of service, data storage, computing and security. concluded with solved security and reliability. The main advantages provided by blockchain are the principal issues faced by ICPS. Fog nodes are in close proximity to IIOT devices where surveillance could be weak and therefore this proposed system could be helpful in attacks such as denial of services, man in the middle, etc. In the future, the authors plan to use the developed tool to test their system for various challenges. Most block chains can process a limited number of transactions simultaneously. Hence security and data integrity issues will be focused on. Further, the authors intend to improve the energy consumption. ICPS-based fog and block chain require novel algorithms, applications and interfaces. Hence, the authors also focus on developing advanced algorithms. The system overview is presented in Fig 3.2.12.

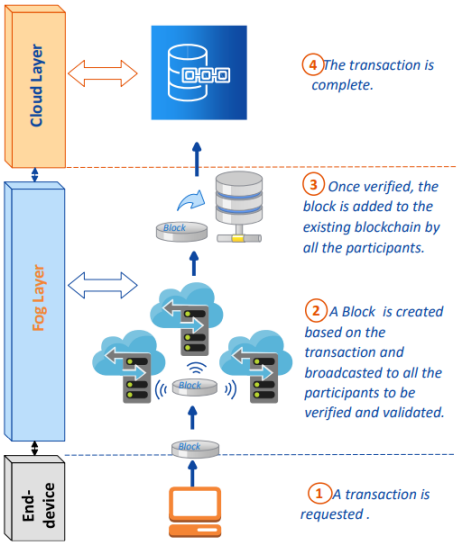


Fig 3.2.12 Blockchain for Fog and IIoT presented by [12]

Awaisi, Kamran and Hussain et.al. [13] presents a Fog-based architecture for IOT-based healthcare management systems. The proposed architecture is validated by conducting experiments in iFogSim. The proposed system was applied to the system values for network usage and latency for fog based and cloud-based scenarios were calculated and compared. It was deciphered that the proposed architecture reduces the above calculated values of network and latency to a significant level to manage and protect the identities of the user safely, the SHA512 and elliptical or cryptography techniques were applied. The shortcomings of the work include, lack of a fault tolerance system. If the system encounters a fault, the ease of access to records might get enabled and cause breaches. Hence, delegation to authorities can be better taken care of. Fig 3.2.13 depicts the overview of the system.

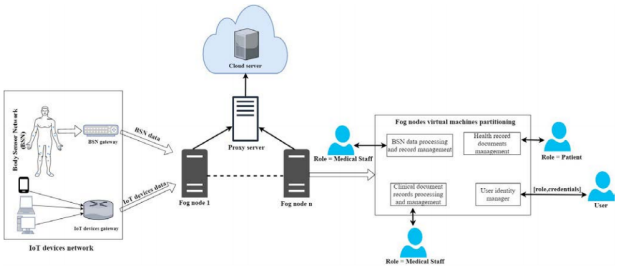


Fig 3.2.13 Blockchain for Fog and IIoT of [13]

Wang, Haoyu, and Wang et al. [14] suggest a model that uses the blockchain's reward and punishment system to improve fog nodes. The resource contribution was numerically evaluated using MATLAB R2018b for the proposed scheme. The optimum resource contribution strategy (t), the fog cluster's satisfaction level, and the income of fog nodes all increase as the game progresses. In future work, the authors suggest modelling and analysing the proposed architecture from various viewpoints and visions (cloud service providers or users), as well as other applications, in order to solve the problem of fog computing in the field of resource management. The structure of the system is shown in Fig 3.2.14.

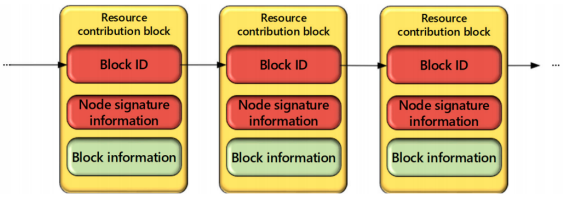


Fig 3.2.14 The Chain structure for the proposed system by [14]

M. H. Ziegler et.al. [15] proposes an architecture design which utilizes the plasma framework to combine blockchain technology and fog computing and evaluate the performance of its prototype.To evaluate the proposed system, two test scenarios were conducted. One for the client and one for the operator a node. In the first scenario, As a result, the CPU reaches its maximum usage during the active emission of transactions. Second scenario shows slightly lower utilization of the CPU. Future research on Plasma focuses on minimizing the size of proofs, on fastening the finality with state channels and on exploring the usage of zero-knowledge proofs. This development will enable improved efficiency which is necessary to run real world IoT applications and constitutes a subject of the author's future research. Fig 3.2.15 depicts the interaction with the ethereum blockchain.

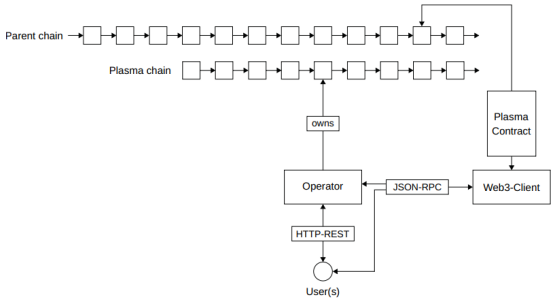


Fig 3.2.15 Users and an operator interact with the Ethereum parent chain via a Web3-client, presented by [15]

Tanweer Alam et.al. [16] utilizes Fog Computing with IoT apparatus on the network edge. These IoT devices use blockchain technology to connect, exchange, and transfer information among themselves. Some of these devices are given a special status, and are known as 'Miners', and are responsible for verifying transactions in the network. Once a transaction is verified by a Miner, the block is added to the previously existing blockchain, and these changes are propagated throughout the network. To execute the blockchain, the Sumeragi algorithm is used on the Hyperledger Fabric platform. The performance of such a framework was tested for various numbers of IoT nodes, and results noted. The author notes that the aim of this paper is to formulate a strategy for authentic and secure connections between a network of IoT nodes. Unfortunately this paper does not provide an open source repository to share this technique with researchers. It also does not disclose detailed implementation

details, but it does describe the results of the experiment. The system architecture of the proposed work is shown in Fig 3.2.16.

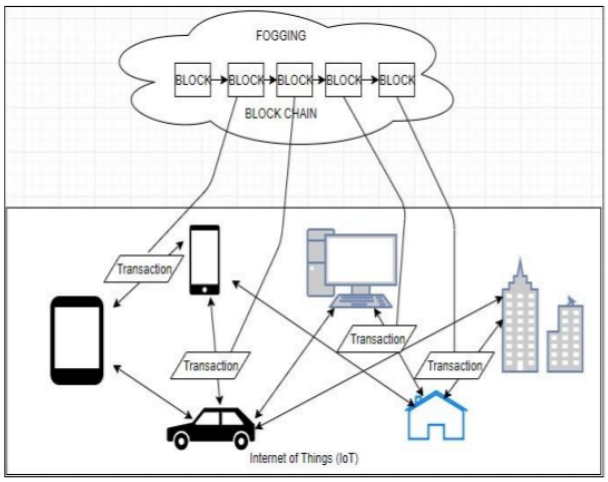


Fig 3.2.16 IoT-Fog system architecture proposed by [16]

A. Muthanna et.al. [17] proposes a three tiered system of which the first layer represents the device layer, which contains all IoT devices and sensor devices. The second layer enables fog computing, by deploying fog nodes to handle data forwarded by dedicated IoT nodes, and allows for edge monitoring, classification, and analyzing capabilities. Results from the second layer are forwards to the topmost third layer, named as the cloud layer. The entire network communicates through two communication protocols, namely SDN (software defined networking) and the blockchain. Using SDN gives network administrators fine grained control over Fog and IoI nodes, through the use of programmable APIs. This paper employs software defined networking to implement it's proposed architecture. The disadvantage of software defined architecture, or SDN is that it eliminates the use of physical routers and switches. This is a potential vulnerability, and can leave your network open to attacks by malware and intruders. Fig 3.2.17 depicts the system architecture of the proposed work.

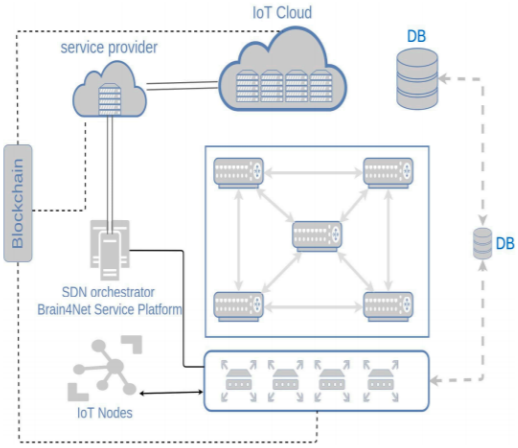


Fig 3.2.17 The proposed system architecture presented by [17]

S. Tuli et.al. [18] A novel approach to detection of heart disease, utilizing deep-learning and IoT has been proposed in this paper. This lightweight framework is named HealthFog, and provides a generic system architecture for development of ensemble deep learning on fog computing. The FogBus framework has been used for interaction of IoT devices and the edge network with the remote cloud, and analysis conducted based on pre-existing heart patient data. Such a system can provide key insights to medical practitioners with high accuracy, low latency, and low energy consumption. FogBus uses blockchain technology to ensure secure communication, along with authentication and encryption techniques to preserve the integrity of sensitive patient data. This paper can improve accuracy, by undertaking training with various ensemble learning methods and increased variety of data. The architecture of the proposed system is shown in Fig 3.2.18.

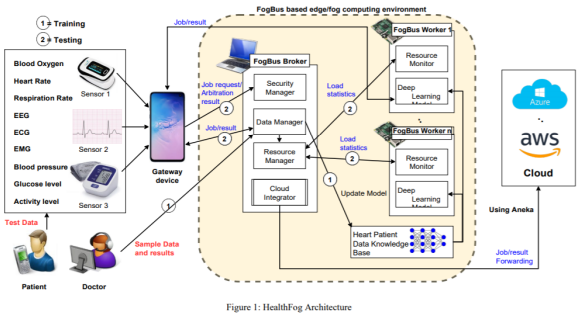


Fig 3.2.18 The proposed system architecture presented by [18]

L. Hang et.al. [19] proposes a system that begins by discussing the difficulties in implementing blockchain technologies on embedded IoT devices, due to resource constraints. To solve this issue, a decentralized scheme is presented to facilitate communication between

end-user devices and the blockchain network, using user-friendly interfaces implemented with front-end web technologies. In order to achieve a lightweight system, in this architecture IoT devices need not download the full node, as consensus processes (Byzantine Fault Tolerance Algorithm) are executed remotely. Advantages of this approach include easy access to resources and management of resources within a blockchain network enabled by a web-based framework. The authors note that research into the convergence of IoT and the blockchain is still in its infancy, and further investigation is required. Pertaining to the proposed architecture, future research directions aim at testing the interoperability of the proposed system with different IoT frameworks. Fig 3.2.19 depicts the system architecture presented in this work.

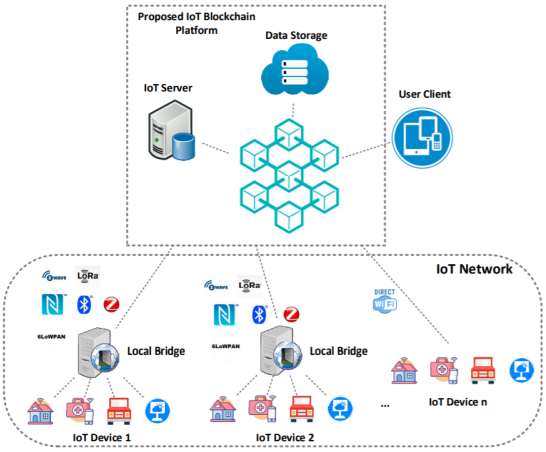


Fig 3.2.19 The proposed system architecture presented by [19]

H. L. Cech et.al. [20] explores the communication of sensor data using the blockchain integrated into an existing low-cost fog computing environment called HCL-BaFog. HCL-BaFog includes SDN (software defined networking) functionality, and it can use Docker Swarm to distribute virtualized functions among nodes working in any mode. Since embedded IoT nodes are frequently resource constrained, the fog gateway incorporates Multichain functionality. Multichain is an open source platform to deploy blockchain applications, and provides an API and command line interface. The authors proceed to describe a procedure for sharing of real time sensor data, by storing only the hash of the data in the blockchain to reduce storage demands. This approach uses an open source blockchain called Multichain. Multichain does not support smart contracts, and is not a private, permissible blockchain. This opens up our system to security vulnerabilities. This paper also

uses SDN, which we have discussed the security implications of previously. Fig 3.2.20 illustrates the architecture of the proposed system.

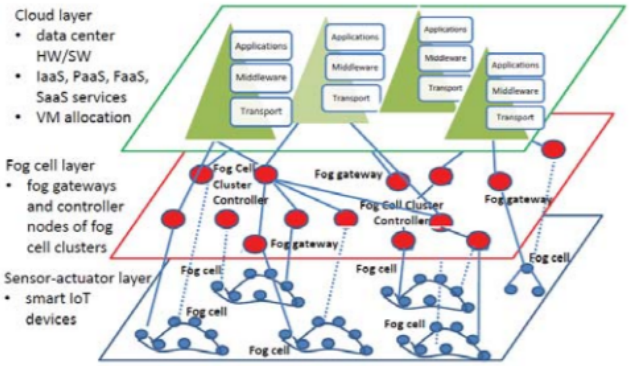


Fig 3.2.20 The proposed system architecture presented by [20]

3.3 Conclusion of Survey

An extensive body of literature exists exploring the intersection of IoT, blockchain, and fog computing. However, it is lacking in the subsequent areas:

- Several solutions employ the use of public, permissionless, open-source blockchain solutions such as Multichain, and Ethereum. Not only are public blockchains extremely slow, they are also vulnerable to malware and intruders. Additionally, Multichain does not support smart contracts.
- Software Defined Networking (SDN) is applied to implement approaches in a few of the papers discussed. Although SDN provides a programmable networking interface, it removes the need for physical switches and routers. Without the firewall capabilities of physical switches/routers, security becomes a huge burden, with the SDN acting as a single point of failure.
- Certain implementations seen place a strain on the limited resources of IoT devices which results in a detrimental performance in a real time setting. This is in the form of a communication and storage overhead.
- Due to the exponentially increasing number of IoT devices, if they all make requests directly to the cloud server it results in a very high latency which is unacceptable for certain real time applications. It may also lead to overload and crashing of the cloud server.

We thus plan to address these gaps through our work in the following ways:

- Hyperledger Fabric: A public blockchain like Ethereum can be joined in any location by any entity. However, IoT networks are also required in applications which must not be accessible to the public, and thus private blockchains are preferred here. This technology provides many advantages which are as follows: All members must be approved and verified in order to access the network, thus preventing the joining of malicious actors; All transactions are private; It uses relatively lesser computing power due to its smaller size; Since all members are trusted, simpler consensus mechanisms can be implemented.
- Fog Computing: Moving heavy computation to the Fog layer, reduces the computational burden on resource constrained IoT devices and cloud services and improves the responsiveness of the system while reducing latency.

Chapter IV

4. Project Management Plan

4.1 Schedule of the Project

A Gantt Chart is a timeline that is used as a project management tool to illustrate how the project will run. It shows individual tasks, their durations and the sequencing of these tasks. The overall timeline of the project and the expected completion date for our project, is represented in the figure below.

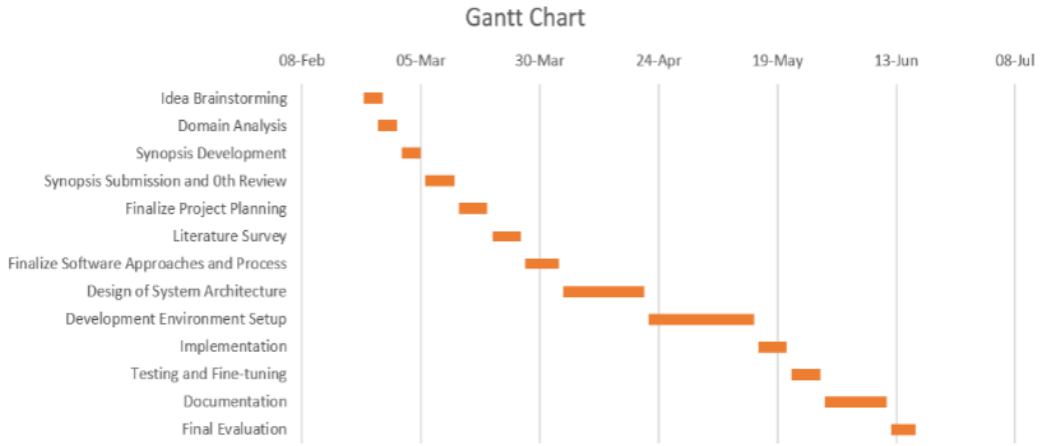


Fig 4.1.1 Gantt Chart for our project.

4.2 Risk Identification

The following points summarise the risks identified in the implementation of this system.

4.2.1 Increased Power and Latency

Integrating blockchain with identity management may result in increased latency and power consumption as it requires time for information processing and validating.

Mitigation Steps:

We add fog nodes which are solely dedicated to the purpose of providing authentication and security services. These enable the formation of a gateway for authentication to take place off chain, thus reducing the period of interaction required with the blockchain. This ensures that the increase in power and latency is acceptable as a tradeoff to the increased security of the system.

4.2.2 Dynamic Identity

As most of the attributes change over time, the blockchain identity management system becomes less efficient if the identities are dynamic.

Mitigation Steps:

In our system, it is the responsibility of the admin to manually register the IoT nodes and perform the required fog device and user access control mappings. Thus dynamic identities must be handled by the organization by constantly keeping track of the devices involved and their unique identifiers.

4.2.3 Failure of Fog Node

The fog node acts as a gateway between the blockchain and the end user for authentication and identity management. Thus the failure of the fog node results in failure of connection between the modules which impairs the process of authentication and user verification.

Mitigation Steps:

Redundancies can be introduced in the system to ensure that there is always a dedicated connection between the blockchain, end users and IoT nodes. There must also be a system in place to ensure that the organization receives alerts in the case a fog node failure occurs.

4.2.4 The Inherent Nature of Blockchains

The nature of blockchain makes the data stored almost impossible to be altered. This may result in GDPR (eg. "right to be forgotten") breaches thus making adoption of the system problematic.

Mitigation Steps:

The definitions of data must be implemented clearly and control can be implemented to ensure that sensitive data is stored off chain with only device identities and revocation statuses being stored on the blockchain.

4.2.5 Data might be input incorrectly or incompletely:

The data which is entered into the blockchain may contain errors which may not be able to be altered due to the inherent nature of blockchain technology. This might result in further problems and errors in the authentication procedure.

Mitigation Steps:

The admins responsible for the onboarding of data onto the blockchain can be subjected to audit and the organizations involved in this situation can be provided the required audit to ensure compliance to norms established.

Risk	Probability	Impact	Experience	Mitigation Plan
Increased power and latency	High	Medium	High	Addition of fog nodes to enable off chain authentication and connection
Dynamic identity	Medium	High	Medium	Provide checks on the admin during registration and mapping of IoT devices

Failure of Fog Node	High	Medium	Medium	Introduce redundancies to ensure constant connection and notify in case of failure of a single fog node
Nature of blockchains	High	Medium	High	Clear data definition and rules must be established
Data input	High	Medium	Medium	Subjection to audit to ensure compliance to data input rules

Table 4.2 Risk Identification and Mitigation steps

Chapter V

5. Software Requirements and Specifications

5.1 Product Overview

We aim to develop an implementation of a block chain-based identity management system for secure healthcare. Blockchain platforms such as Hyperledger fabric and a consensus algorithm called RAFT consensus algorithm is used to achieve the above. The objectives of the product include developing and integrating the secure health system, ensuring the veracity of the architecture as well as testing the performance and evaluating the results obtained. The scope of the project is limited to developing a proof-of-concept prototype which checks if the architecture is in line for business production and registration of the various devices to the consortium Block chain network. The goal of the implementation is to address fault tolerance, security and sensitivity issues in a centralized system. The benefits of Digital identity management and authentication include network infrastructure security at all levels, identity verification and authentication of all the participants and uniform patterns of

authorization to access electronic health information. It can be used in managing electronic medical records, protection of healthcare data, personal health record data management etc.

5.2 External Interface Requirements

5.2.1 User Interfaces

The graphical user interface consists of a mobile application. The user can enter the IP address of the node to view the status of the IoT health sensors. The GUI shows the status of the sensor along with the graphical representation of the data.

5.2.2 Hardware Interfaces

The hardware requirements has been identified as follows:

Component	Minimum Requirement
Processor	CPU single-core or dual-core
RAM	2 GB
Hard Disk	25 GB HDD Free Space

Table 5.2.2 Hardware Interfaces

5.2.3 Software Interfaces

The software requirements has been identified as follows:

a. Operating System – Ubuntu 18.04 Desktop

b. Hyperledger Fabric - for distributed identity system based on blockchain technology

c. ProjectLibre - for Gantt Chart

d. Star UML - for Data Flow diagram

5.2.4 Communication Interfaces

On receiving the authorization request from the user, the fog node and smart contracts acts as a communication interface by verifying and authorizing the users to access the IoT node and enables data transfer.

5.3 Functional Requirements

5.3.1 Admin and IoT Sensor Nodes Registration

The system must be able to register the admin and the new IoT sensor device when a request is made by providing a unique identifier ID.

5.3.2 User Authentication

When the user makes an authentication request to access the IoT node, the blockchain must authorize or reject the request. The credentials are checked if they match with the contents in the distributed ledger. A token is sent to the user and the fog node if the authorization is successful.

5.3.3 Data Exchange between IoT Device and the User

Once the user is authenticated and verified, the system must open a connection between the sensor and the user for exchange and storage of data for further analysis.

Chapter VI

6. Design

6.1 Introduction

Each entity possesses a unique identifier which is used for registration to the network. The user sends a payload containing its credentials and other required details to the fog node for registration. The fog node interfaces with the blockchain and a digital identity certificate is generated using Fabric Certification Authority, which is followed by the credentials being registered on the blockchain channel. If the entity wants to then gain access to network resources, they must use their unique identifier to request authentication. The required payload is sent to the fog node which interfaces with the blockchain network. The network checks if the entity is registered on the blockchain. If not, the request is rejected. If the entity is registered, then the request is recorded on the blockchain, and the success acknowledgement is sent to the entity following which it can gain access to network resources. The entire blockchain network is placed at the fog layer thus allowing a reduction in latency while interfacing with the blockchain.

6.2 Architectural Design

An Architecture Diagram is a representation of the entire workflow and steps of a project in the form of flow charts or graphs. It gives an overview of the entire project in a single Diagram which includes the project's principles, elements and components.

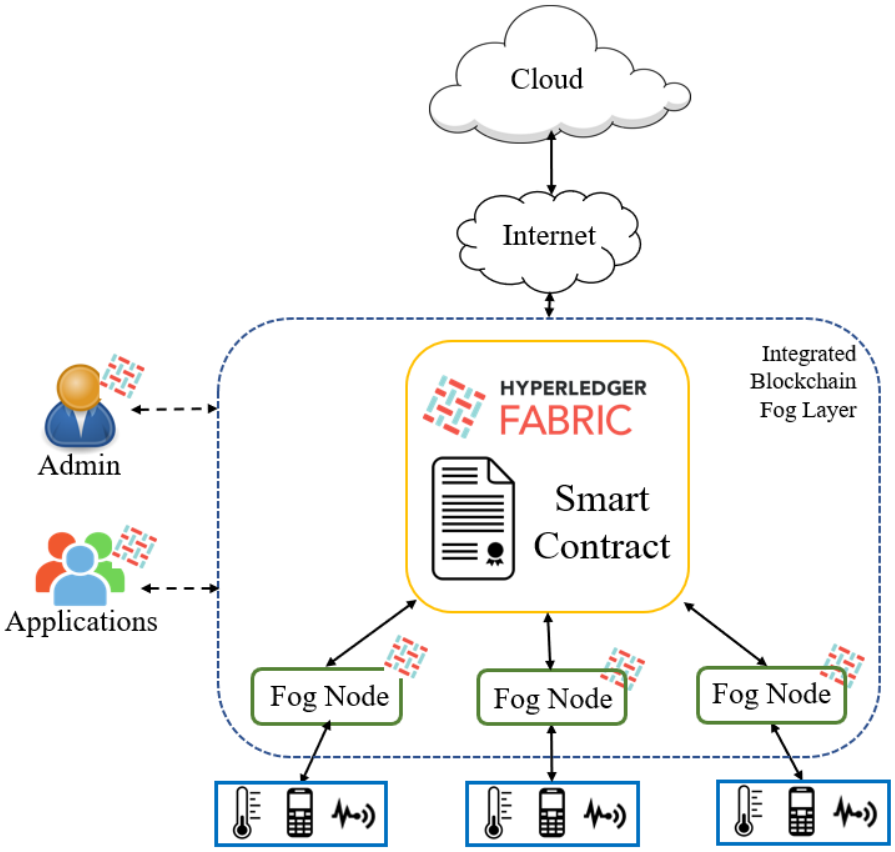


Fig 6.2 Architectural Design for Consortium Blockchain Based IoT Identity Management

6.3 Graphical User Interface

User Interface Design refers to the design of the user interface for software, websites or applications. The following figures represent an example of the user interface provided to analyze the statistics of various channels and understand the underlying blockchain network.

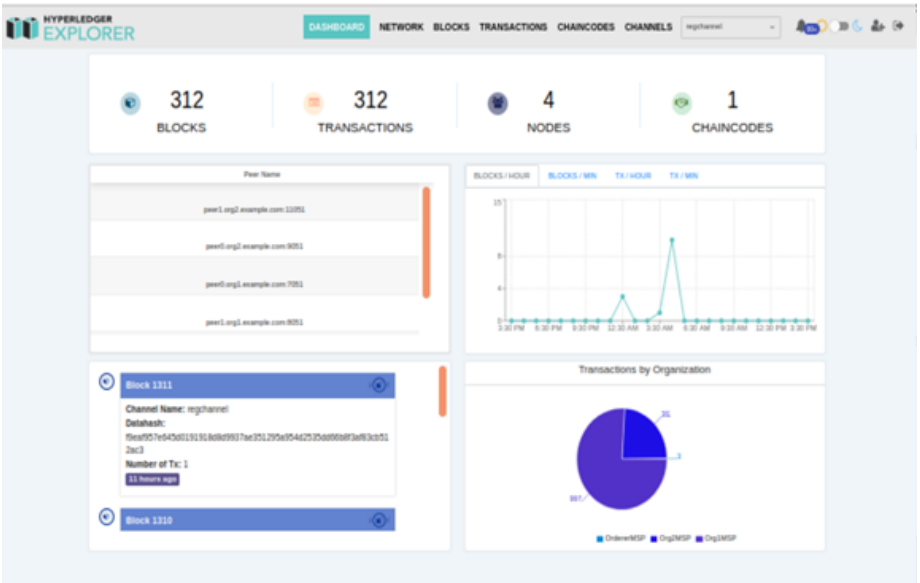


Fig 6.3.1 User interface for registration channel

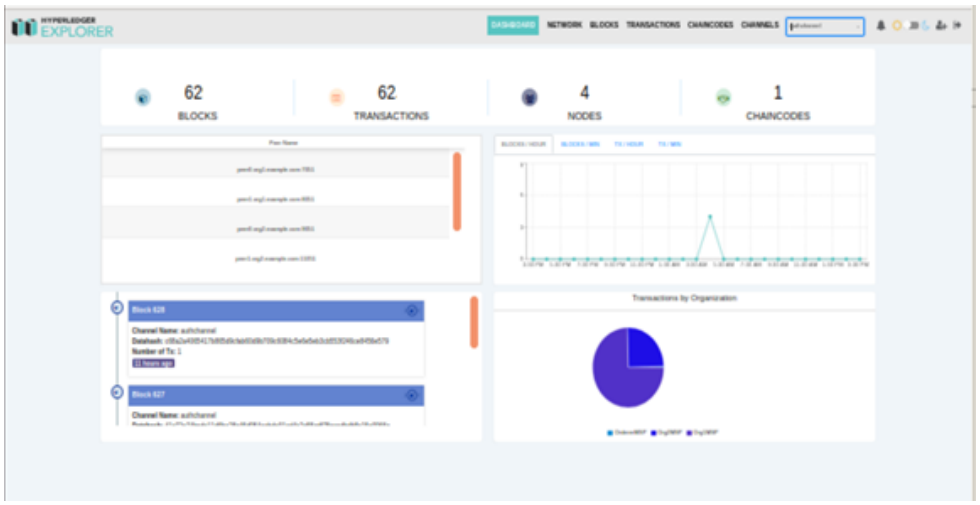


Fig 6.3.2 User interface for authentication channel.

6.4 Class Diagram and Classes

These diagrams are used to represent the various aspects of a system, through structures called 'classes'. Classes help us define the characteristics and functions of objects, and the relationships between them. Classes also enable us to display different kinds of inheritance between data structures. They are beneficial in translating design models into actual programmable code.

- Upper Section: The upper section encompasses the name of the class.

- Middle Section: The middle section constitutes the attributes, which describe the quality of the class.
- Lower Section: The lower section contains methods or operations. It demonstrates how a class interacts with data.

The diagram below represents the class diagram of the proposed system.

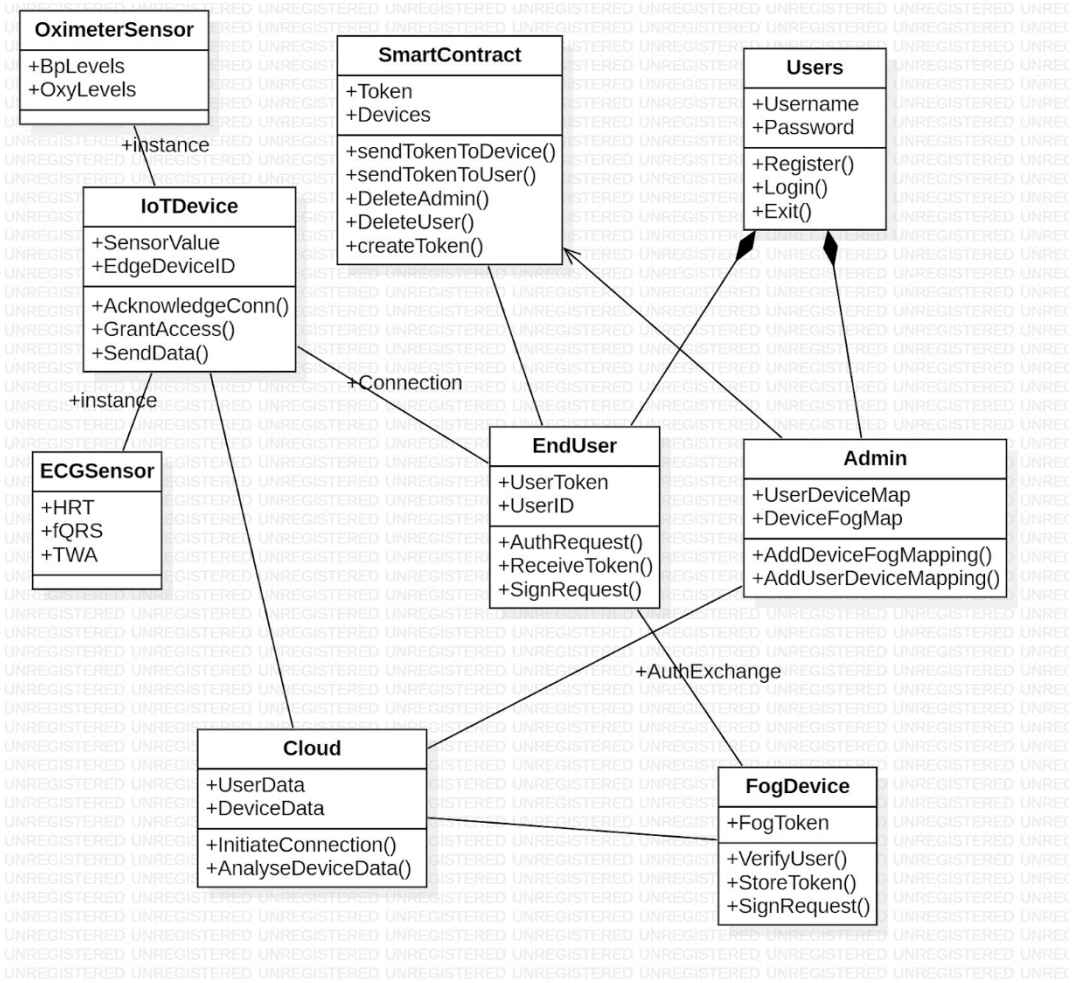


Fig 6.4 Class Diagram for the proposed system.

6.5 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

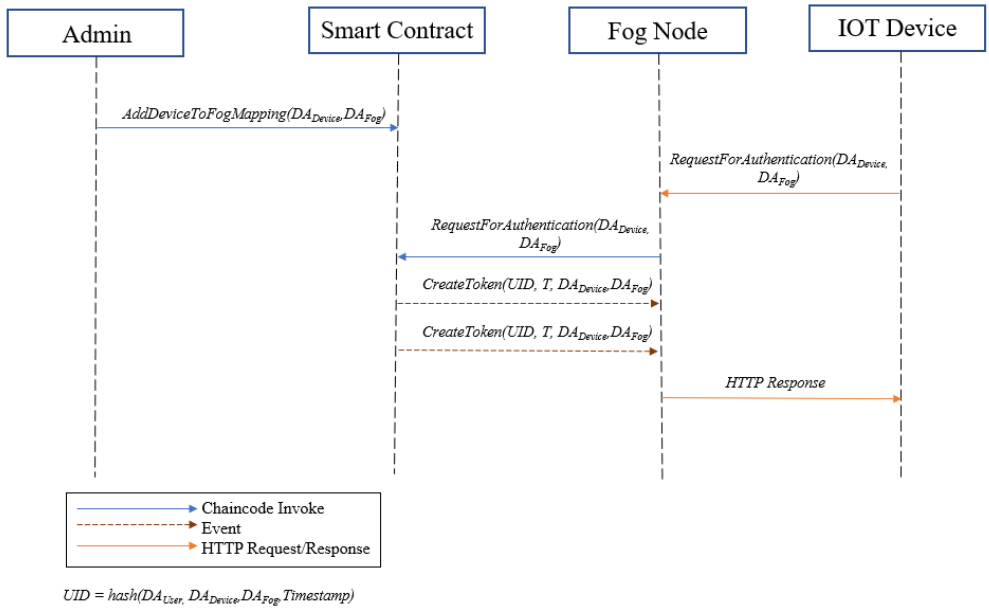


Fig 6.5 Sequence diagram for authentication function of the proposed system.

6.6 Data Flow Diagram

A data flow diagram shows the manner in which data moves through an interaction or framework. It incorporates data sources and outputs, data stores, and the different sub cycles the information travels through. DFDs are made utilizing standardized symbols and notations to portray different elements and their connections. The objective of the data flow diagram is to depict the boundaries of a system and the scope of the work. It enables easy visualization of contents and represents a detailed and well explained diagram of system components. The data flow diagram may be used to perform a system or software at any level of abstraction. The diagram below represents the data flow diagram of the proposed work.

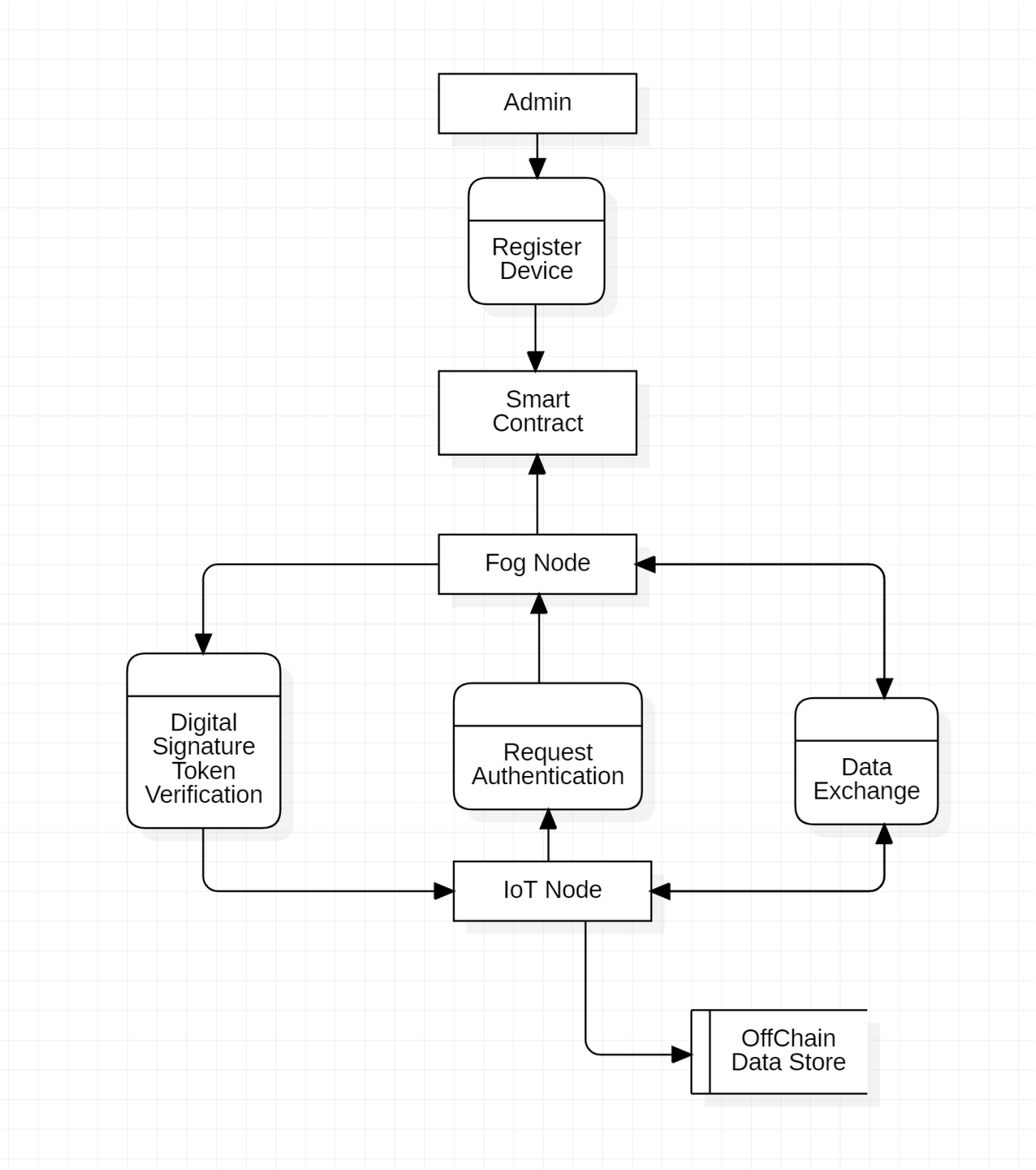


Fig 6.6 Data Flow Diagram for the proposed system.

6.7 Conclusion

The aim of this project is design and develop a consortium - blockchain based IoT identity management technique and describes the associated protocols, in order to address privacy, scalability and security issues in a centralized identity management system. We employ fog computing in order to provide lower latency and a system with higher availability. We also

aim to provide a graphical user interface which is easy to use and provides a comfortable feel.

Chapter VII

7. Implementation

7.1 Tools Introduction

7.1.1 Docker swarm

Docker swarm is a tool used for container orchestration, where a group of different containers such as virtual machines that are configured together to form a cluster can be deployed amongst multiple host machines. This work utilizes docker swarm to construct the network between the four virtual machines that is the fog nodes in the distributed set up

7.1.2 Docker

Docker is an open source platform which can be used to develop an application, ship an application as well as run different applications. Docker helps Attenuate the dilatory between implementation and production. This work utilizes Docker swarm to orchestrate a Blockchain network between the different virtual machines existing in our system

7.1.3 Ubuntu

Ubuntu 18.04 is used to run the fog nodes. Our implementation consists of four VMs or fog nodes.

7.1.4 Hyperledger explorer

Hyperledger Explorer is a graphical user interface friendly tool for visualizing Blockchain implementations and operations. It is supported with a rich user interface which is used to represent, depict, query, view and deploy the various transactions that occurred in the permissioned Blockchain platform. In our implementation the GUI shows transactions per

minute in the Blockchain authentication channel information, etc. Latency and throughput can also be viewed.

7.2 Technology Introduction

7.2.1 Fog computing

For computing is an infrastructure that brings the data closer to the cloud. Leveraging the fog computing technology in combination with Blockchain application can be exploited for its advantage of Blockchain having a decentralized, modular structure. In such a System multi various devices can be incorporated into the Blockchain platform via fog computing which greatly increases the business efficiency and at the same time reduces operational expenses.

7.2.2 Python

This is a general-purpose programming language with which the client code has been written. The clients in our project include 2 organizations and each organization has 2 peers and each peer represents one VM/Fog node.

7.2.3 Golang

This is an open-source programming language that has been used in our implementation to write the chaincodes. In our implementation, there is a registration chaincode that collects data during the registration phase and an authentication chaincode that accesses and verifies the authenticity of the data.

7.2.4 Hyperledger Fabric

Hyperledger Fabric is an open-source permissioned blockchain platform developed for usage in enterprise settings that offers some important advantages over other prominent DLT or blockchain platforms. Since it is a private blockchain it does not provide general public access to data which is an important consideration in a healthcare scenario.

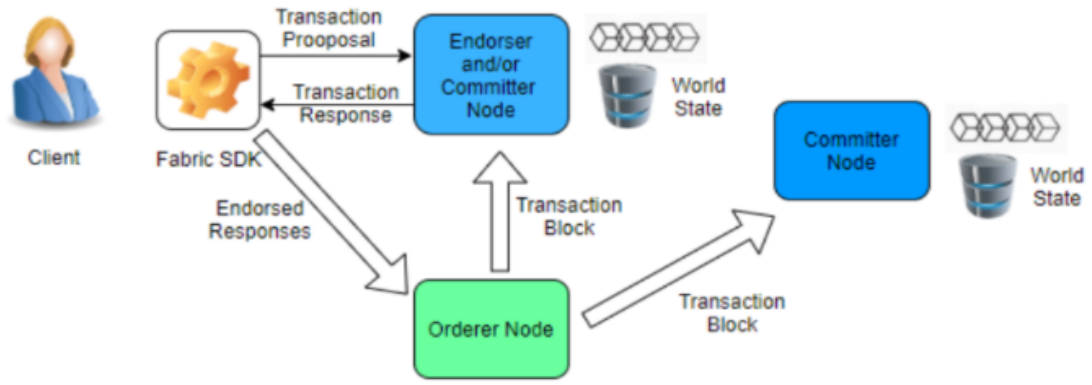


Fig 7.2.4 Hyperledger Fabric Architecture

Hyperledger fabric architecture consists of different components. These include:

1. Membership service provider (MSP): MSP is the module that delineates the rules in which identity is authenticated, validated, and given access to a certain network. It also defines the roles of various components of the fabric network.

2. Nodes: These are the communication entities that are the only logical functions that allow users to generate various services. Nodes of different kinds can be run on one physical server. There are three types of nodes. These are,
    - The client node - The client node is the one that invokes the transaction to the endorsers and broadcasts these transaction proposals to the orderer node/ordering service.
    - The peer node - This peer node commits the transactions and saves the state and copies of the ledger. These are fundamental to the fabric because they host smart contracts (chain codes) and ledgers. It receives the updates in the form of transaction blocks from the orderer/ordering service.
    - The Orderer Node - The orderer node performs the transaction ordering by implementing the consensus protocol. It is responsible for running the communication service and these nodes together form an ordering service. Hyperledger supports three consensus methods which are Solo, Kafka, and RAFT. For our implementation, we have used Raft consensus algorithm.
3. Chaincode: Defines a series of executable application logic, that is stored or contained in the world state. World state is a database that stores the state of the ledger. The

application logic in the chain code governs the functioning of all the transactions. The chaincode or smart contract is hosted by leveraging the container methods/technology by the fabric network.

4. Fabric SDK: This part of the system allows for communication between the applications and the fabric blockchain network. It facilitates an API which is used to query the ledger contents and to submit the transactions to the ledger.

7.3 Overall view of the project in terms of implementation

This section presents an implementation of the proposed blockchain based IoT identity management system. The aim of this implementation was to address the gaps our team deduced from an extensive literature survey of existing and related works. The high level view of our architecture is based on utilized Fog Computing for supporting the Blockchain. In this model, our tiers consist of client devices at the bottom level, fog nodes at the middle tier, and the cloud unit at the topmost level. These tiers interact with each other to support the many functions of identity management. Blockchain support is provided at the Fog nodes using Hyperledger Fabric, which is a modular blockchain framework that acts as a foundation for developing blockchain-based products. Each Fog node is run on a Ubuntu 18.04 virtual machine, with docker capabilities installed. The client is run on an OS that supports compilation and execution of Python 3.7 programs. The cloud unit is an Amazon EC2 instance, running Ubuntu 18.04 operating system.

7.4 Explanation of the Algorithm and how it is been implemented

In order to provide registration and authentication functionality, our team developed a custom chaincode, which is used to implement the business logic of how applications interact with the ledger. The logic applied here is simple and the dependency involved is the authentication function required to fetch information from the registration ledger. This method adopts the use of digital certificates which are generated using Fabric CA. We use unique object identifiers to obtain information for a particular identity. The entity can register itself to a particular organization by using this unique identifier. The protocols used are described as follows:

- Registration - The device is provided a unique ID and the admin registers it to the blockchain network. Once a request for registration is received by the network, the blockchain checks if the device has been previously registered. If not, it registers the device to the network. The identifier is created by combining the metadata of the transaction and the identifier of the device sent with the request.
- Authentication - Once a request for authentication is received, the network checks the registration ledger for the existence of the identity. If so, it records the requests and sends an authentication success acknowledgement.
- Revocation - This involves removing the device permissions to access the network resources. This is a relatively less frequent function.

An essential facet of implementing Hyperledger fabric is the ordering service. It features a node called the orderer which is responsible for the ordering of transactions. This node in conjunction with the other orderer nodes forms the ordering service. The design of Hyperledger Fabric depends on deterministic consensus algorithms, where a transaction block which has been validated by any peer is considered to be correct and final. There are several ordering service implementations that are configurable to use with Hyperledger Fabric, including Kafka, Raft, and Solo. In this work, we have adopted the Raft algorithm to execute our ordering service. Raft is a crash fault tolerant ordering service, which was designed in order to address the issues in implementation and understanding of the Paxos Algorithm. It follows a "leader and follower" model, where a leader node is elected, and its actions are replicated by all the follower nodes. It is capable of handling up to half of the systems' failures. The failure of the leader node can also be handled by this system provided there is a majority. This is easier to set up and manage in comparison to Kafka-based ordering services, and their design allows various organizations to contribute nodes to a distributed ordering service. Raft provides a better fit as it is relatively less resource consuming and more straightforward. Each organization can set up its own orderer node, leading to a more decentralized system.

The Raft protocol, by virtue of its design, provides the following safety against malfunctioning of the consensus:

- Leader election safety – At most one leader is elected for one term

- Log Matching safety - if multiple logs contain an entry with an identical index and term, those logs are guaranteed to be indistinguishable in all entries up through to the given index.
- Leader completeness – The log entries committed in each term will always appear in the logs of the leaders following the said term.
- State Machine safety – if a server has applied a specific log entry to its state machine, then, at that point no other server in the server cluster can apply a different command for the same log.
- Leader is Append-only – A leader node can only append new commands to its log
- Follower node crash – When the follower node crashes, all the requests sent to this node are ignored and it cannot take part in the leader election process. When the node restarts, it's log its synced with the leader node.

7.5 Information about the implementation of Modules

7.5.1 Layer 1: IoT devices

This layer consists of a group of IoT. They are grouped and can access a particular node in the Fog Layer. These devices have very low computational power. This layer simply sends requests to the fog controller and collects data, thus forming the external interface. These devices are assumed to have a unique address which is used for registration and authentication. The fog node which interfaces with the device registers and authenticates it by implementing a smart contract. This module was implemented as a simple Python script which sends requests to the fog nodes for registration, authentication and revocation of identity.

7.5.2 Layer 2 : Fog Layer implementing Blockchain functionality

This layer is responsible for the management of identity of devices at the IoT layer. It serves requests from the connected devices such as registration, authentication and revocation of

access. The Hyperledger Fabric permissioned blockchain also runs a network of peers at this layer, thus forming a distributed system performing blockchain services. It also handles smart contract services and consensus algorithms.Smart contracts are installed and instantiated on all or selected peers. Chaincodes deployed provide invocation and query services. The node contains an API server which is used to interface with the blockchain.IoT devices mapped to the node can invoke blockchain functionalities by sending requests to an API server on the node.

Chaincode deployment can be on multiple peers, thus allowing parallelized execution and improving scalability.

This module was implemented by using 4 Linux Ubuntu 18.04 virtual machines with 2GB RAM. An overlay network was created over these machines using Docker Swarm. The services were deployed on various machines using Docker stack and configuring placement of services. Hyperledger Fabric peers, orderers and cli were deployed as Docker containers to form the blockchain network. It thus provides an environment that separates the application program from the physical resources and isolates the containers from each other to ensure the application's security.

Fabric-CA serves as the certification authority, with a separate CA being deployed as a container for each organization. The Fabric-CA issues, validates and revokes certificates for identity management.

A Nodejs API server is also developed and containerized which can interface with the blockchain peer and thus respond to requests sent by the client involving ledger modification and querying.

7.5.3 Layer 3 : Public Network (Cloud)

This layer is responsible for management of fog controller nodes and providing wide area networking capabilities. These have the highest computational power and can be used to aggregate data from various nodes. This data then can be stored and analyzed to produce further results. Fog controllers can be registered and authenticated at this layer and then information transfer can be initiated. This would be a relatively less frequent process and thus time delay for this can be tolerated in the system.

7.5.4 Visualization Module

Hyperledger Explorer is a user -friendly web application which enables us to view, query and invoke blocks. It enables us to browse underlying activity in the blockchain. It can be used by creating docker containers for the application and the database or can be directly built from the codebase. In our experiment we deploy Hyperledger Explorer as docker containers. This interface can then be used to understand the network and peers involved.

7.6 Conclusion

We implemented the identity management model using the distributed blockchain platform, Hyperledger fabric, which is a permissioned blockchain. The fog nodes were developed using Ubuntu and a network was orchestrated between the fog nodes using Docker swarm. Other tools such as Python for client code and Golang for Chaincodes were also used to achieve the expected outcome. Fog computing is chosen since its geographical proximity is more to the users hence reducing latency and enabling fast response. Fog layer also provides high security and power efficiency. RAFT consensus algorithm is used since it is a production level consensus protocol justifying proof of concept for the production environment. Hyperledger fabric is the chosen blockchain platform since it is a permissioned blockchain that allows only limited authorization which is ideal in case of a healthcare scenario.

Chapter VIII

8. Testing

8.1 Introduction

Testing is integral to determining the speed, security, and stability of a framework. There are many types of performance tests that simulate various possible user scenarios to understand the behaviour of systems.

To ensure our architecture and implementation met the standards of modern day IoT and Fog applications, the architecture proposed in this work was tested rigorously against several test conditions and parameters. Both the authentication and registration channels were evaluated for transaction time as well as RTT (Round Trip Time), against varying payload sizes. For each channel and payload size, 60 trials were conducted, resulting in a total of 960 trials.

8. 2 Testing Tools and Environment

In order to test such a system along preferred parameters, were sent on a periodic basis through a Python script. This script allowed us to capture key insights for each transaction, with accurate timestamp data, as well as generate simulated payloads as per the required size. Hyperledger Explorer was utilized before and after each transaction to verify the increase in the number of blocks. Python 3.7 was employed as the runtime environment for this script.

8.3 Test Cases

Our proposed framework consists of two separate channels for authentication and registration. Each channel allows us to perform both invoke and query functions. Test cases were formulated based on the channel, function, and payload size.

Use Case	Test Case	Description	Procedure	Expected Result
UC01: Client wants to register to an organization.	TC01: Client sends the registration request to Fog node.	Client submits the registration POST request with UserID and Organization name data to Fog node invoke endpoint.	Client supplies it's UserID and the name of the preferred organization, the body of the POST request is constructed and it it sent to the Fog node endpoint.	The client is successfully registered to the organization and receives a 200 status code response from the API Server.
UC02: Client wants to register a device to fog mapping.	TC02: Client sends the mapping request to Fog node.	Client submits the mapping registration POST request with device address, fog address, and other details to the Fog node query endpoint.	Client supplies the device address, fog address and the name of the preferred organization, the body of the POST request is constructed and it it sent to the Fog node endpoint.	The device fog mapping is successfully recorded to the ledger and client receives a 200 status code response from the API Server.
UC03: Client wants to verify that it has been registered.	TC03: Client sends the verification request to Fog node.	Client submits the verification GET request with device address, fog address, and other details to the Fog node invoke endpoint.	Client supplies the device address, fog address and the name of the preferred organization, the body of the GET request is constructed and it it sent to the Fog node endpoint.	The device fog mapping is successfully recorded to the ledger and client receives a 200 status code response from the API Server. If registration does not exist, client receives a 404.
UC04: Client wants to request authentication with the ledger.	TC04: Client sends the authentication request to Fog node.	Client submits the authentication invoke request with device address, fog address and other details to the Fog node invoke endpoint.	Client supplies the device address, fog address, and the name of the preferred organization, the body of the POST request is	The client is successfully authenticated and client receives a 200 status code response from the API Server.

			constructed and it it sent to the Fog node endpoint.	
UC05: Client wants to verify authentication with the ledger.	TC05: Client sends the authentication verification request to Fog node.	Client submits the authentication verification query request with device address, fog address and other details to the Fog node invoke endpoint.	Client supplies the device address, fog address and the name of the preferred organization, the body of the GET request is constructed and it it sent to the Fog node endpoint.	The client is receives a 200 status code if it is authenticated, otherwise it receives a 404 error from the API Server.

Chapter IX

9. Results and Performance Analysis

9.1 Result Snapshots


```
{
        "@context": [
                "www.example.com/v1",
                "www.example.com/credentials"
        ],
        "type": "VerifieableIdentity",
        "purpose": "register",
        "verifiableCredential": [{
                "deviceID": "uuid:vd8s710e-b7b1-194m-177b-3d918310hdn21",
                "deviceType": "sensor",
                "issuer": "Org1MSP",
                "issuanceDate": "2021-05-06T20:23:28Z",
                "credentialTitle": {
                        "deviceName": {
                                "en-us": "Oximeter sensor",
                                "purpose": "oxygen level"
                        },
                        "measureType": "numeric",
                        "units": "sp02",
                        "measureDisplay": {
                                "en-us": "sp02"
                        },
                        "measureUnitSuffix": {
                                "en-us": "%"
                        }
                }
        }]
}
```


Fig 9.1.1 Example of sensor payload

```
ubuntu@ubuntu-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
ubuntu@ubuntu-VirtualBox:~$ docker node ls
ID                            HOSTNAME            STATUS    AVAILABILITY   MANA
GER STATUS   ENGINE VERSION
4lpizn9rwlarxdzyud66nszvj *   ubuntu-VirtualBox   Ready     Active         Lead
er           20.10.2
lj8f447v18t75nsze0c1f8cwz     ubuntu-VirtualBox   Ready     Active
             20.10.2
md3vgkvpuhoqo1ipcatrwl8e5     ubuntu-VirtualBox   Ready     Active
             20.10.2
w5nr282mpnwk40zunqz3u3uvl     ubuntu-VirtualBox   Ready     Active
             20.10.2
ubuntu@ubuntu-VirtualBox:~$ docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
b55ea24fbb92   bridge            bridge    local
32021c4c40fa   docker_gwbridge   bridge    local
0c220ac61754   host              host      local
n94o8oidhkq8   ingress           overlay   swarm
9e51259c3366   none              null      local
ep0aky7yntqg   test              overlay   swarm
ubuntu@ubuntu-VirtualBox:~$
```

Fig 9.1.2 Representation of nodes connected in a Docker Swarm with an overlay network

```
y:24h0m0s} Idemix:{IssuerPublicKeyfile:/etc/hyperledger/fabric-ca-server/Issuer
PublicKey IssuerSecretKeyfile:/etc/hyperledger/fabric-ca-server/msp/keystore/Is
suerSecretKey RevocationPublicKeyfile:/etc/hyperledger/fabric-ca-server/IssuerR
evocationPublicKey RevocationPrivateKeyfile:/etc/hyperledger/fabric-ca-server/m
sp/keystore/IssuerRevocationPrivateKey RHPoolSize:100 NonceExpiration:15s Nonce
SweepInterval:15m}}
2021/06/04 23:21:41 [DEBUG] DB: Getting identity 2BMBRC3A
2021/06/04 23:21:41 [DEBUG] DB: Login user 2BMBRC3A with max enrollments of 1 a
nd state of 0
2021/06/04 23:21:41 [DEBUG] DB: identity 2BMBRC3A successfully logged in
2021/06/04 23:21:41 [DEBUG] DB: Getting identity 2BMBRC3A
2021/06/04 23:21:41 [DEBUG] Requested expiry '0001-01-01 00:00:00 +0000 UTC' is
 before the CA certificate expiry '2021-05-27 19:50:00 +0000 UTC'. Will use CA
cert expiry
2021/06/04 23:21:41 [DEBUG] Processing sign request: id=2BMBRC3A, CommonName=2B
MBRC3A, Subject=<nil>
2021/06/04 23:21:41 [DEBUG] Request is not for a CA signing certificate
2021/06/04 23:21:41 [DEBUG] Checking CSR fields to make sure that they do not e
xceed maximum character limits
2021/06/04 23:21:41 [DEBUG] Finished processing sign request
2021/06/04 23:21:41 [DEBUG] DB: Getting identity 2BMBRC3A
2021/06/04 23:21:41 [DEBUG] Attribute extension being added to certificate is:
&{ID:[1 2 3 4 5 6 7 8 1] Critical:false Value:7b226174747273223a7b2268662e41666
6696c696174696f6e223a226f7267312e6465706172746d656e7431222c2268662e456e726f6c6c
6d656e744944223a2232424d4252433341222c2268662e54797065223a22636c69656e74227d7d}
2021/06/04 23:21:41 [DEBUG] Adding attribute extension to CSR: &{ID:[1 2 3 4 5
6 7 8 1] Critical:false Value:7b226174747273223a7b2268662e416666696c696174696f6
e223a226f7267312e6465706172746d656e7431222c2268662e456e726f6c6c6d656e744944223a
2232424d4252433341222c2268662e54797065223a22636c69656e74227d7d}
```

Fig 9.1.3 Registration of Identity with Fabric CA

```
mmitted block [1305] with 1 transaction(s) in 150ms (state_validation=50ms bloc
k_and_pvtdata_commit=24ms state_commit=68ms) commitHash=[0b7cd3085059b1a20f08f9
2a198519fd8309b14c8bd11f9e2de99e4ec0a7bcbc]
2021-06-04 23:06:23.172 UTC [kvledger] CommitLegacy -> INFO 8ca [regchannel] Co
mmitted block [1306] with 1 transaction(s) in 165ms (state_validation=22ms bloc
k_and_pvtdata_commit=23ms state_commit=110ms) commitHash=[a62be98897fb79c5b331a
38238fb2aa2a8f9f47a3e968a347f4407651663ac0a]
2021-06-04 23:06:37.251 UTC [kvledger] CommitLegacy -> INFO 8d7 [regchannel] Co
mmitted block [1307] with 1 transaction(s) in 72ms (state_validation=11ms block
_and_pvtdata_commit=27ms state_commit=28ms) commitHash=[bc61798e024a8c1ed7622da
5911f010766764d6bf1e08dced15a912abfae5cd3]
2021-06-04 23:06:39.777 UTC [kvledger] CommitLegacy -> INFO 8dd [regchannel] Co
mmitted block [1308] with 1 transaction(s) in 144ms (state_validation=41ms bloc
k_and_pvtdata_commit=16ms state_commit=79ms) commitHash=[53ad4cecc7ee591e3a0d04
8ce1fd0d27bc1157e0baafde8e8786d788d5386f89]
2021-06-04 23:06:52.327 UTC [kvledger] CommitLegacy -> INFO 8eb [regchannel] Co
mmitted block [1309] with 1 transaction(s) in 126ms (state_validation=16ms bloc
k_and_pvtdata_commit=18ms state_commit=80ms) commitHash=[74a4162eedfe44ecc383bc
c35a5e1dd1af89c44c202a1f8561cb2caabafc7e24]
2021-06-04 23:06:54.835 UTC [kvledger] CommitLegacy -> INFO 8f2 [regchannel] Co
mmitted block [1310] with 1 transaction(s) in 118ms (state_validation=8ms block
_and_pvtdata_commit=19ms state_commit=85ms) commitHash=[1bad5ed991c5ca23bfb7e97
617c5dfdfacb66d898e8b9419dbb3eaacc4acf120]
2021-06-04 23:21:45.712 UTC [kvledger] CommitLegacy -> INFO c52 [regchannel] Co
mmitted block [1311] with 1 transaction(s) in 394ms (state_validation=53ms bloc
k_and_pvtdata_commit=50ms state_commit=278ms) commitHash=[33c30811750fb212a3356
8a2466c6f82a93ac586217574d09f94e2fd065c982e]
```

Fig 9.1.4 Registration of identity using blockchain (register ledger)

```
ubuntu@ubuntu-VirtualBox:~$  docker logs -f 33614a1b1e9d 2>&1 | grep "authchann
el] Committed"
2021-06-04 23:04:10.165 UTC [kvledger] CommitLegacy -> INFO 899 [authchannel] C
ommitted block [625] with 1 transaction(s) in 524ms (state_validation=320ms blo
ck_and_pvtdata_commit=25ms state_commit=161ms) commitHash=[b81609915fec246c0066
066f4fa2a5ee5e60599934502c10256dc6833f84db9e]
2021-06-04 23:06:26.328 UTC [kvledger] CommitLegacy -> INFO 8d1 [authchannel] C
ommitted block [626] with 1 transaction(s) in 175ms (state_validation=2ms block
_and_pvtdata_commit=13ms state_commit=138ms) commitHash=[13b799bf28e5a449b09de4
f2402a8443ed7309b2cd9f13267099b94e18f5d22b]
2021-06-04 23:06:42.339 UTC [kvledger] CommitLegacy -> INFO 8e4 [authchannel] C
ommitted block [627] with 1 transaction(s) in 82ms (state_validation=0ms block_
and_pvtdata_commit=35ms state_commit=39ms) commitHash=[8309800317016df8254c3912
01f774d033d06afeac6fa702222022147bb3f956]
2021-06-04 23:06:57.443 UTC [kvledger] CommitLegacy -> INFO 8f7 [authchannel] C
ommitted block [628] with 1 transaction(s) in 106ms (state_validation=0ms block
_and_pvtdata_commit=27ms state_commit=72ms) commitHash=[d76a8b51e6f1d5adf3179df
5ac28661eeb54e37eb94817b339dad8452929c530]
```

Fig 9.1.5 Performing authentication on separate channel

```
IP: http://192.168.0.105:4000
Sent Register for userId TC817P

Register() took 0.8214421272277832 to run

{'wallet': {'providerRegistry': {'providers': {}}, 'store': {'storePath':
'/usr/src/app/wallet'}}}
Sent addDeviceFogMapping for D:d742 F192.168.0.105 UserAddr: TC817P

addDeviceFogMapping() took 3.4391541481018066 to run

200
Sent addDeviceFogMapping for D:d742 F192.168.0.105 UserAddr: TC817P

addUserDeviceMapping() took 3.235823154449463 to run

200
Sent requestAuthentication for D:d742 F192.168.0.105 UserAddr: TC817P

requestAuthentication() took 3.182769298553467 to run

{'UID': '6dac1833280f512643e560096a4240c95e28edd5', 'user': 'TC817P',
'device': 'd742', 'fog': '192.168.0.105', 'devicePayload':
'new_device_payload'}
```

Fig 9.1.6 Output generated on client IoT device

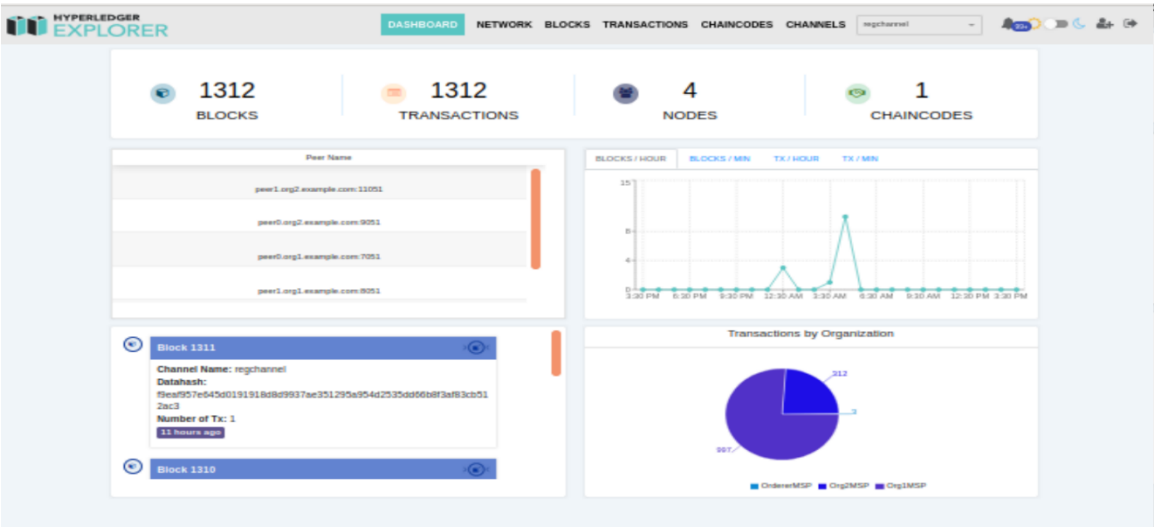


Fig 9.1.7 Registration channel statistics viewed using Hyperledger Explorer interface

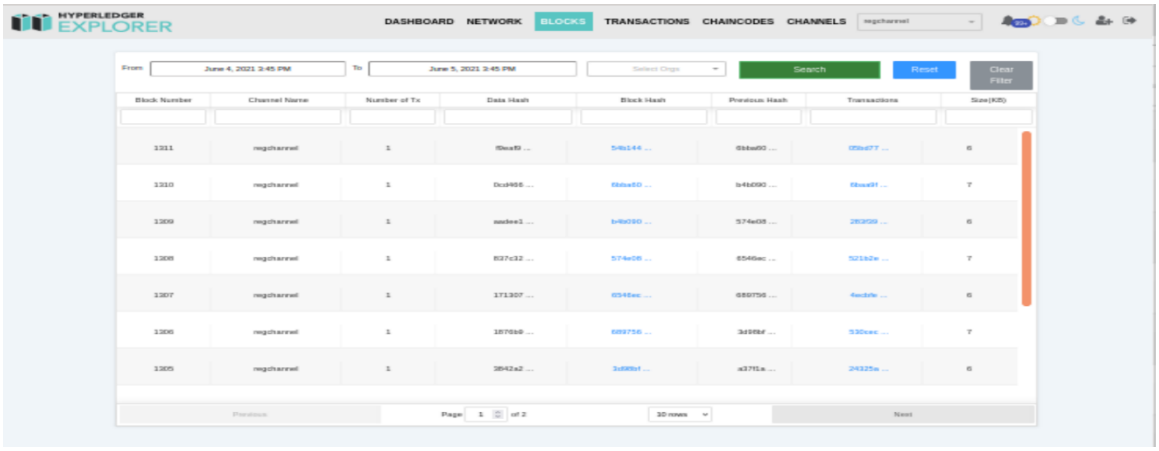


Fig 9.1.8 User interface for viewing list of blocks present on registration channel

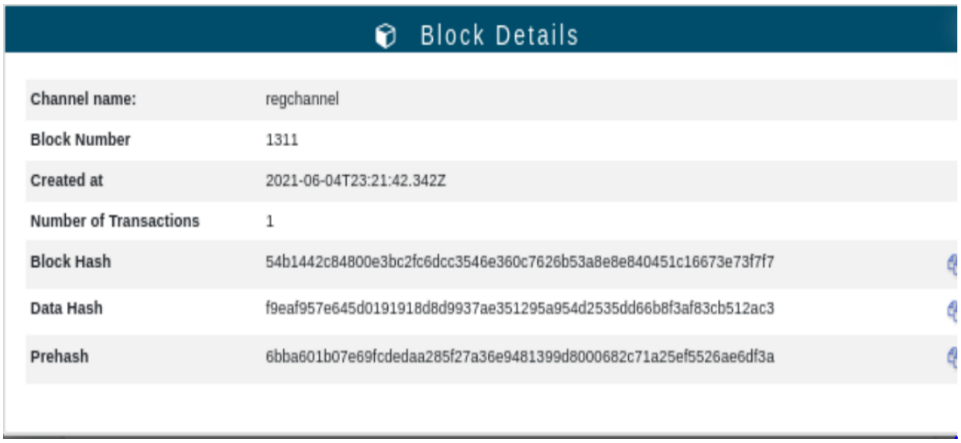


Fig 9.1.9 Example of registration block details viewable on interface

Fig 9.1.10 Transaction details for registration functionality

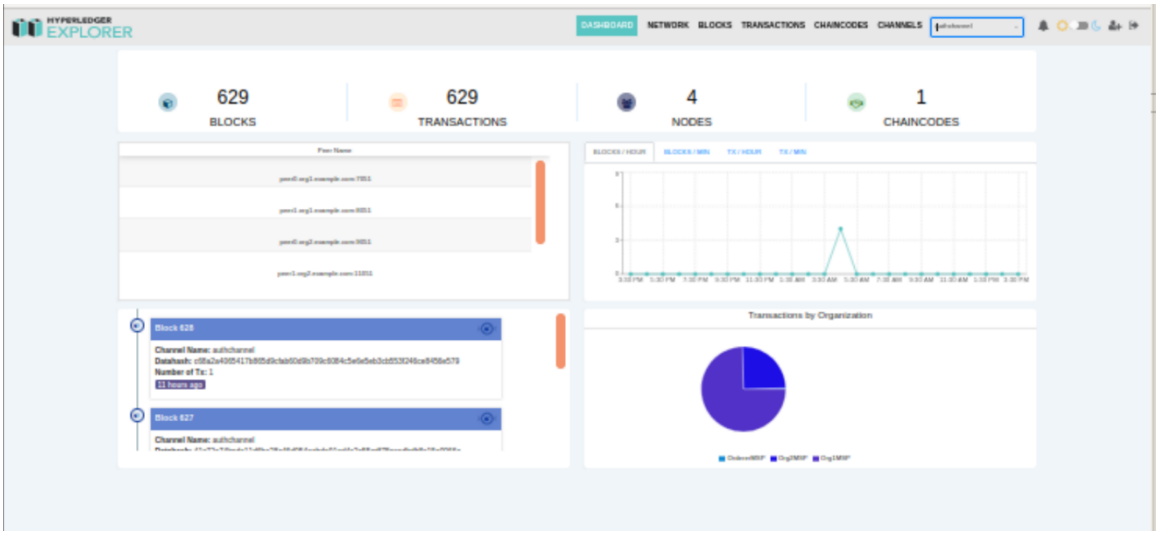


Fig 9.1.11 Authentication channel statistics viewed using Hyperledger Explorer interface

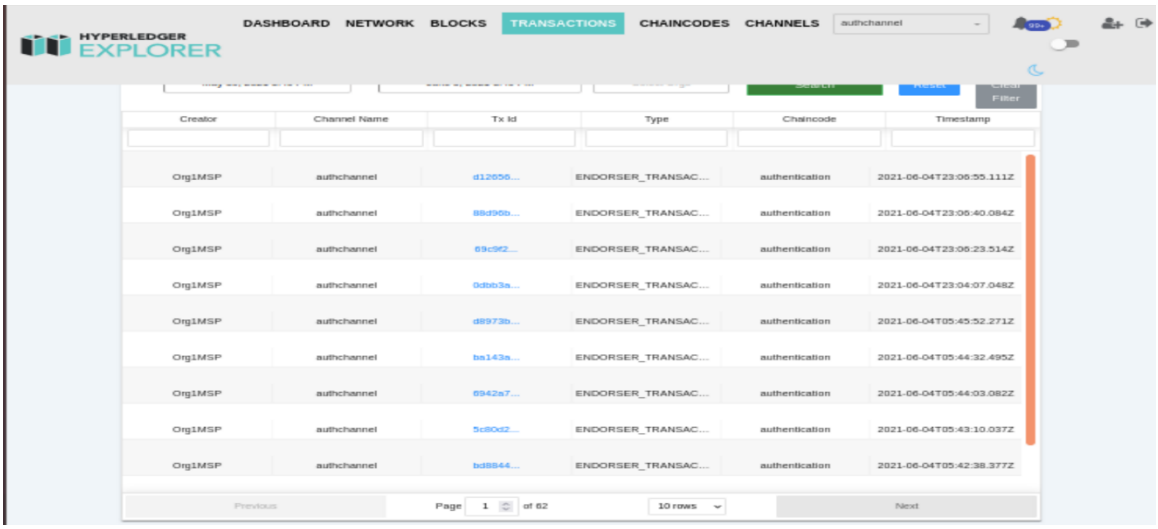


Fig 9.1.12 User interface for viewing list of blocks present on authentication channel

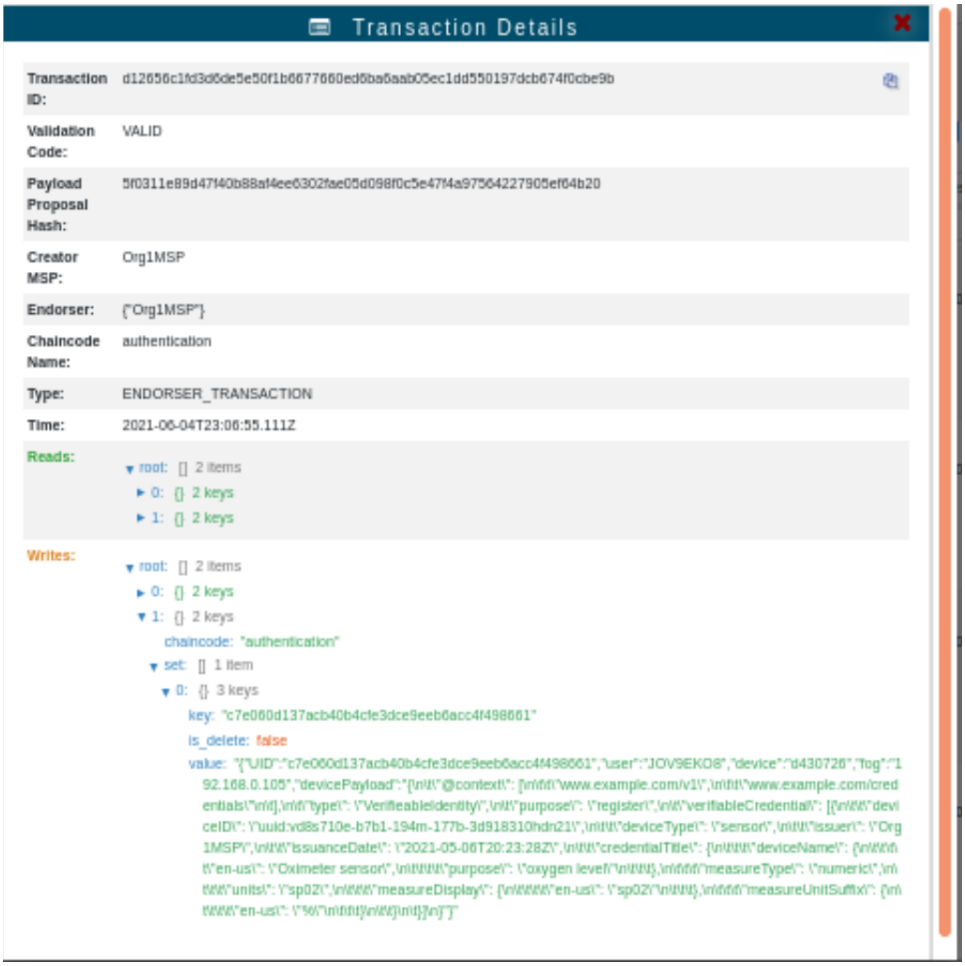
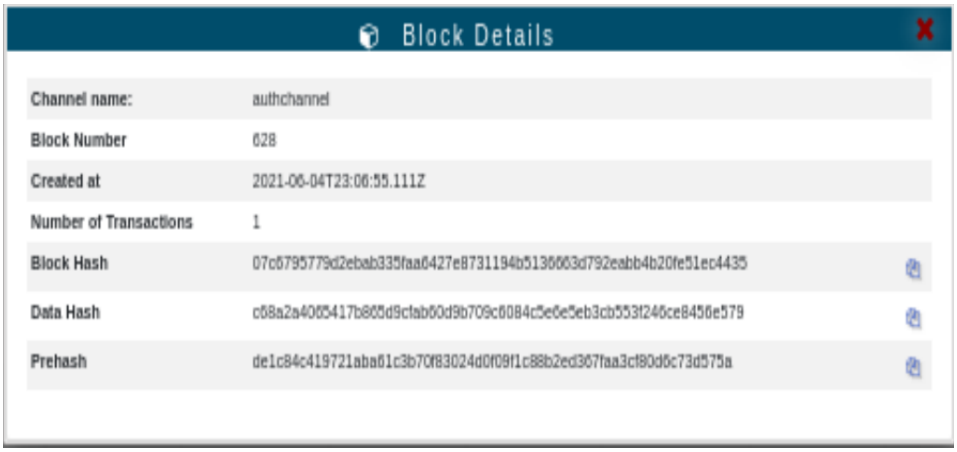


Fig 9.1.13 Transaction details for authentication functionality



Block Details

Channel name:	authchannel
Block Number	628
Created at	2021-06-04T23:06:55.111Z
Number of Transactions	1
Block Hash	07c6795779d2ebab335faa6427e8731194b5136663d792eabb4b20fe51ec4435
Data Hash	c68a2a4065417b865d9c1ab60d9b709c6084c5e6e5eb3cb553f246ce8456e579
Prehash	de1c84c419721aba61c3b70f83024d0f09f1c88b2ed367faa3cf80d6c73d575a

Fig 9.1.14 Example of authentication block details viewable on interface

9.2 Comparison with related works and results

Query and invocation times were measured for different payload sizes for both registration and authentication channels. 60 trials were conducted for each payload size. The results of the analysis are presented in Table 9.2.1. It can clearly be seen that the transaction latency is dependent on payload size and it affects both throughput and latency. Trials were conducted to analyse the service latency of the system by using client server to send requests to the assigned fog nodes with varying payload. The results of these trials are depicted in Table 9.2.2.

		Payload Size (in MB)	Average (ms)	Min (ms)	Max (ms)	TPS
Registration	Invoke	0.35	175.15	101	310	6
Registration	Invoke	0.5	229.866667	108	324	4
Registration	Invoke	1	241.866667	118	335	4
Registration	Invoke	1.5	243.65	117	350	4
Registration	Query	0.35	69.65	22	202	14
Registration	Query	0.5	76.5833	22	259	13
Registration	Query	1	139.4	51	243	7
Registration	Query	1.5	141.5	59	282	7
Authentication	Invoke	0.35	198.41667	105	302	5
Authentication	Invoke	0.5	244.8	144	342	4
Authentication	Invoke	1	218.21667	122	326	5
Authentication	Invoke	1.5	249.3333	111	387	4
Authentication	Query	0.35	84.31667	22	233	12
Authentication	Query	0.5	85.21667	30	263	12
Authentication	Query	1	143.2	50	238	7
Authentication	Query	1.5	173.7833	63	283	6

Table 9.2.1 The payload size (PLS), transaction action, the average time of the trials, minimum and maximum of the transaction latency, and throughput (transaction per second) for registration and authentication ledgers

		Payload Size (in MB)	Average	Min	Max
Registration	Invoke	0.35	2.149371016	0.2851443291	3.3597139
Registration	Invoke	0.5	2.168229012	1.065559277	4.954394102
Registration	Invoke	1	2.791633019	1.632224649	5.632884979
Registration	Invoke	1.5	3.201848284	1.037397176	5.428032875
Registration	Query	0.35	0.7325008667	0.38099	1.938812
Registration	Query	0.5	0.7477107833	0.377013	2.303852
Registration	Query	1	0.9823700167	0.446843	2.24896
Registration	Query	1.5	0.9849132833	0.426885	2.732391
Authentication	Invoke	0.35	2.1218755	1.074985	3.366099
Authentication	Invoke	0.5	2.248452127	1.029971361	5.301445723
Authentication	Invoke	1	3.02275585	1.115124	5.724133
Authentication	Invoke	1.5	3.197353725	2.006488562	6.453481197
Authentication	Query	0.35	0.5682287667	0.31285	1.151666
Authentication	Query	0.5	0.8117199167	0.428156	2.306834
Authentication	Query	1	0.9141895833	0.417438	2.44627
Authentication	Query	1.5	1.11977485	0.677872	2.705791

Table 9.2.2 The payload size (PLS), transaction action, the average time of the trials, minimum and maximum of the transaction latency, and throughput (transaction per second) for communication with Fog node and receiving response

Works	BC Platform	Domain	Computing Power	Block Size	Trial	Transaction Time
Blockchain for IoT Security and Privacy	Cooja Simulator	Smart home	-	16–36 Bytes	-	69 ms
Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers	Ethereum	-	Intel Core i5 12 GB RAM	-	100	Create: 27.25 ms Verify: 120.17 ms
Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers	Chainspace	-	-	Create: 1.38 KB, Verify: 1.76 KB	10,000	Create: 12.1 ms, Verify: 19.3 ms
IoTChain: A three-tier blockchain-based IoT security architecture	Contiki 2.7 Simulator	-	-	0.8 MB	-	Write: 6 s, Read: 9 s
Authenticating health activity data using distributed ledger technologies	IOTA protocol	Health activities	Intel i7–7700 HQ @ 2.80 GHz	500 Char	300	Create: 386 ms, Attach: 12.8s
Authenticating health activity data using distributed ledger technologies	IOTA protocol	Health activities	ARMv7 Processor rev 5 (v7l)	500 Char	300	Create: 6.07 s, Attach: 16.6s
Our Work	Hyperledger Fabric	-	Linux Ubuntu 18.04	1 MB	60	Reg-Invoke: 241.867 ms, Reg-Query 139.4 ms, Auth-Invoke: 218.2667 ms Auth-Query: 143.2 ms

Table 9.2.3 Comparisons of Related work. This Table performs a comparison of various blockchain based identity management systems with the work presented in this report. Most previous implementations have used a public blockchain like Ethereum which shows a massive problem in scalability and possesses a high transaction cost. IOTChain was implemented using a simulator and possessed a transaction time of 6 to 9 seconds which is not conducive to an IoT identity management system for exponentially increasing devices. The IOTA protocol possesses a single point of failure due to the centralized nature of the system, as the consensus node is handled by IOTA. Other works also use smaller payloads and thus transaction latency may increase to a large extent with higher payload sizes. We thus provide a secure, fast decentralized system which is capable of handling large payload sizes.

9.3 Performance Analysis (Graphs)

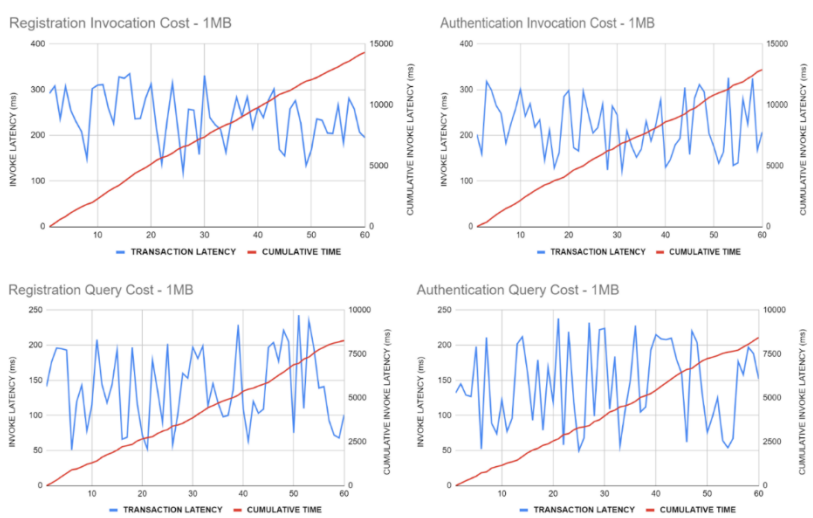


Fig 9.3.1 Registration and Authentication Invocation and Query Costs for payload of 0.35MB

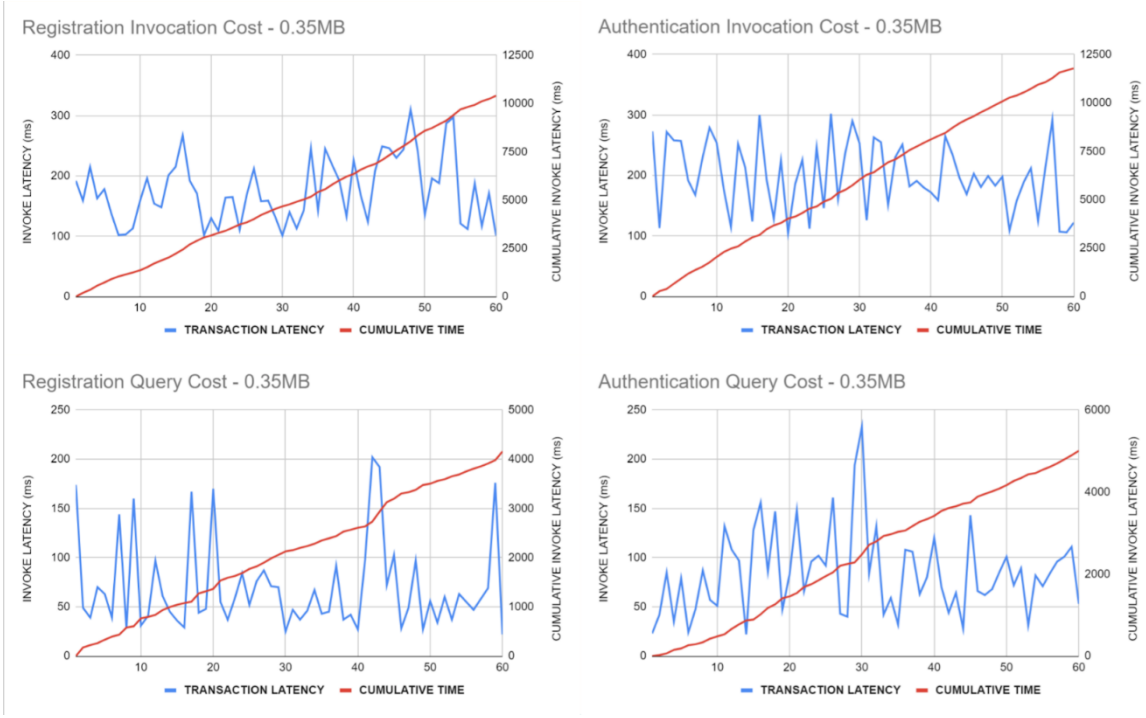


Fig 9.3.2 Registration and Authentication Invocation and Query Costs for payload of 0.5MB

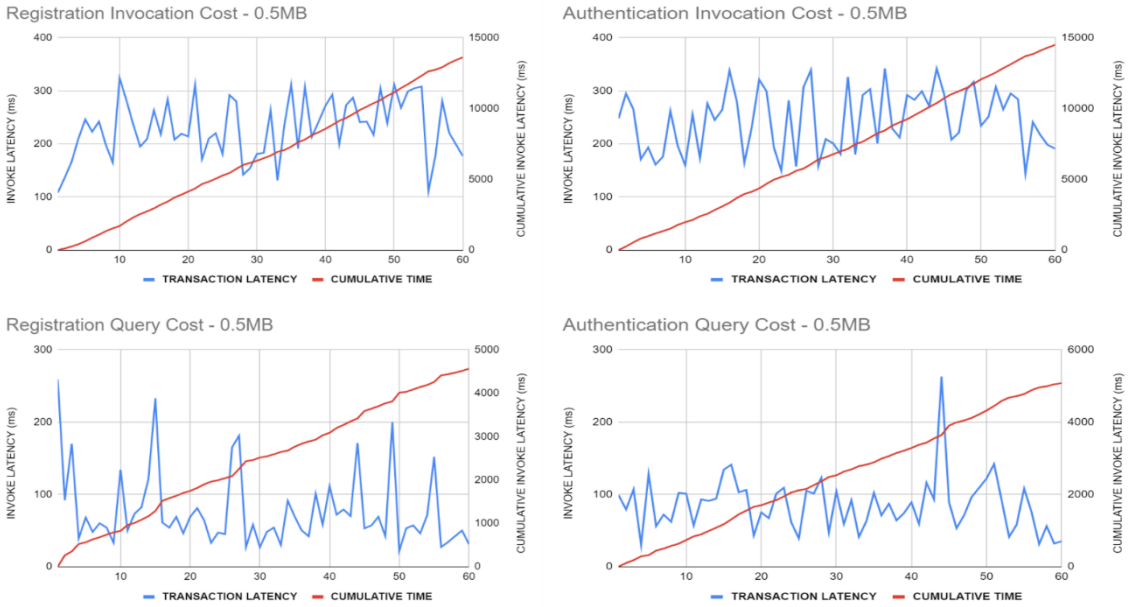


Fig 9.3.3 Registration and Authentication Invocation and Query Costs for payload of 1MB

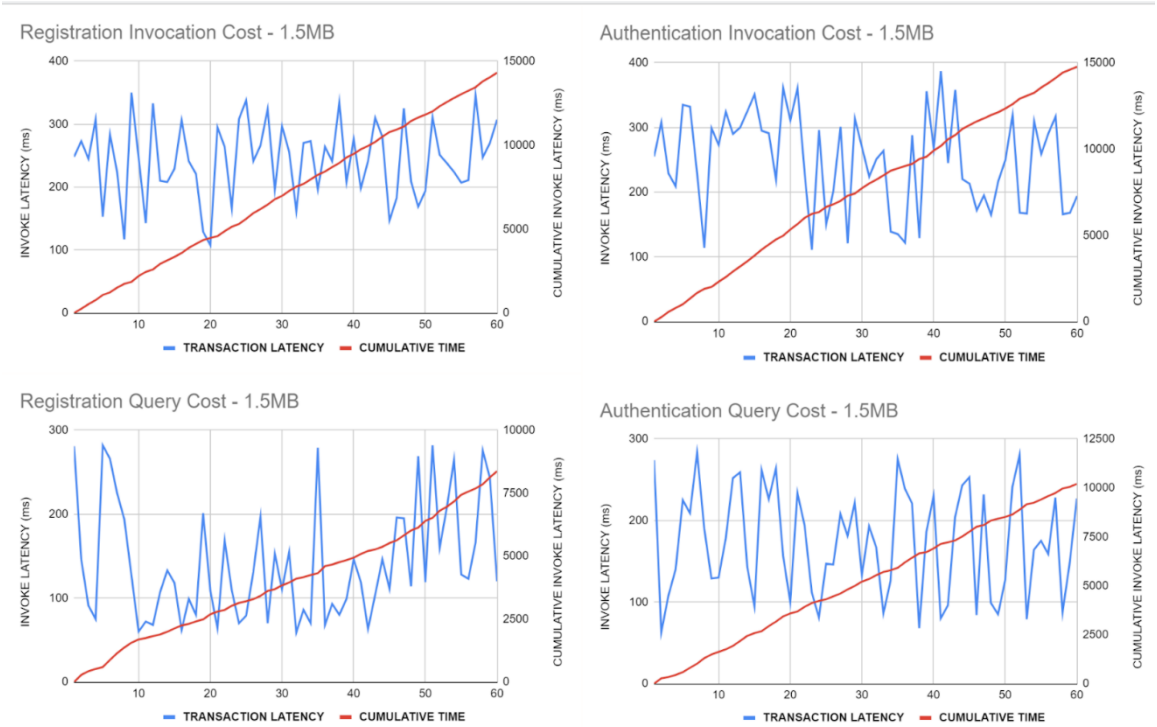


Fig 9.3.4 Registration and Authentication Invocation and Query Costs for payload of 1.5MB

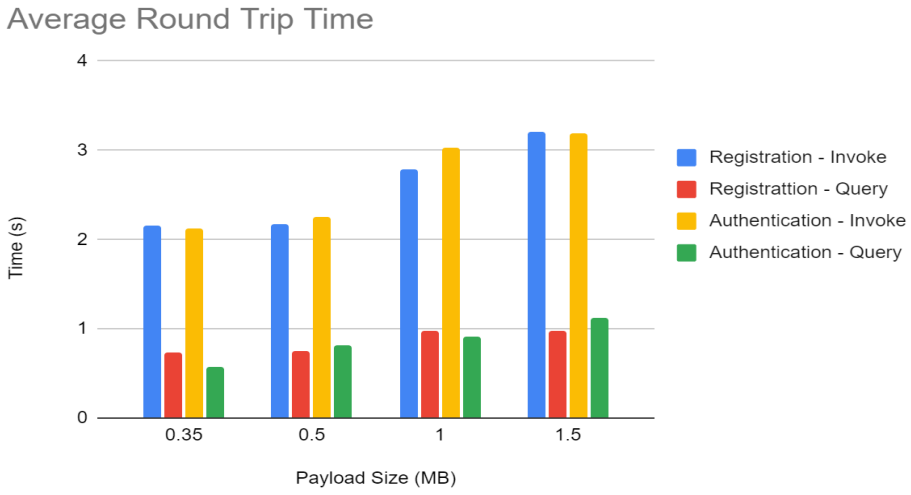


Fig 9.3.5 Average round trip time for various identity management operations

Chapter X

10. Conclusion and scope for future work

10.1 Findings and Suggestions

Our Project implements a consortium - blockchain-based IoT identity management approach that addresses scalability, privacy, and security issues in a centralized identity management system, particularly in a healthcare scenario. The findings are presented as below:

- The presented findings indicate a significantly optimal output for IoT applications. Peers' and clients' processes are the primary resource of consumption in the network, with the Orderer and the chaincode use relatively fewer resources.
- The use of fog computing coupled with blockchain has helped reduce latency and total round trip time as well as produced a better throughput for transactions per minute. The deployment of the blockchain at the fog layer enables the system to be distributed and thus further improves scalability.
- The use of a permissioned blockchain, which is a private blockchain, prevents data access from the general public and allows only limited authorization which is advantageous for a healthcare system. Hyperledger Fabric is also suited for enterprise use and has shown good scalability performance. Thus the use of this technology proves advantageous.
- In our implementation, the key functions of identity management which are registration and authentication are executed on 2 different ledgers, the aim of which is to meet the scalability of the fast-growing environment by ordering the transactions and minimizing the wait time. Separation of ledgers supports simultaneous execution of authentication and registration functions.
- It can be seen that query time is less than time for invocation which could be a benefit of splitting the ledgers.
- The Raft consensus algorithm enables an easy setup when compared to Kafka and has a better fault tolerant system compared to Solo protocol. It can be used in a production

environment and thus this work provides a proof of concept for production environments.

- Simulators of any kind were not used in the process of input data collection. Hence, The collected data was more reliable and provided better accuracy in the results.

Based on our findings, We particularly espouse and suggest the use of such a system i.e a system with multiple ledgers, in a healthcare scenario since hospitals and other healthcare facilities are flooded with patients. For example, while a sensor device X is executing a registration transaction, a sensor Y can execute an authentication transaction simultaneously and hence, avoid mismanagement and facilitate smooth patient treatment. We also strongly suggest using Raft consensus protocol for blockchain solutions that aim for a large scale production environment and suggest using this blockchain platform as a foundation to blockchain solutions which aim for usage within private enterprises.

10.2 Significance of the Proposed Research Work

Provision of security to an exponentially increasing number of IoT devices while ensuring scalability in resource constrained environments forms a major research area. Another problem is the diversity of the devices, thus requiring a system which can ensure interoperability and privacy within the IoT realm. Cloud based centralized solutions possess very high latency and cost. Blockchain thus provides a promising solution to IoT identity management due to its decentralized nature. The placement of the distributed blockchain at the fog layer which is geographically proximate to IoT devices enables lower latency and higher availability of the system. Hyperledger Fabric is a private and permissioned blockchain where organizations must enroll using a Membership Service Provider. It is suitable for enterprise use and is highly scalable in nature. It is capable of providing high throughput and low latency while ensuring privacy of transactions. This system has been implemented using Hyperledger Fabric and clearly describes the different identity management functions and the lifecycle of the entity in the network. This system also analyzes the effectiveness of separating the various identity management functions and analyzes the effect on latency. The system has been evaluated and shows good performance and a significantly optimal throughput. Thus this work demonstrates a proof-of-concept for the system proposed to be deployed on a large scale production environment.

10.3 Limitations of this Research Work

- There are a large number of devices and different fog nodes which are present in this architecture. These fog nodes may be present in an environment which is less secure. By imposing fake IP addresses, hackers can easily gain access to these vulnerable nodes. There is also the increased risk of infiltration of the device and the company by corrupted data files. This causes a vulnerability towards Man-in-the-Middle attacks.
- Maintenance is more complicated in fog architectures in comparison to cloud architectures due to the distribution of controllers and storages across various locations in the network.
- Each person today generally possesses more than one device and each service may have thousands of connected sensors. This causes difficulty in terms of management of these objects. Providing an automated method for registration of these different objects is essential to serve the fast growth of the IoT environment.

10.4 Directions for the Future Works

Technology has been rampant in the last decade. In Today's world, every person owns more than 2 devices. These devices consist of many apps and services and each of these services make use of a number of sensors. These sensors are linked together in a large distributed environment and is becoming arduous for humans to keep a track and manage these objects and services. An automatic registration of sensors considering the level of trust becomes a necessity to ease the complications. Considering the fact that the registration could become automated in a few years, multi-level authentication becomes extremely important. It would require sensors and objects to provide two or more verification factors to gain access to the resources they are looking for. This ensures security as well. Identity management has already become a need in various industries and aspects such as Big data centers, Privileged account management, Mobile device management, etc. In the near future, Identity management is expected to be a huge contributor to Cyber security as well.

References

[1] – A. Alrawais, A. Alhothaily, C. Hu and X. Cheng, "Fog Computing for the Internet of Things: Security and Privacy Issues," in *IEEE Internet Computing*, vol. 21, no. 2, pp. 34-42, Mar.-Apr. 2017

[2] – Dubey, Harishchandra & Constant, Nicholas. (2015). Fog Data: Enhancing Telehealth Big Data Through Fog Computing. ACM BigData 2015 The Fifth ASE International Conference on Big Data

[3] – C. Li and L. Zhang, "A Blockchain Based New Secure Multi-Layer Network Model for Internet of Things," *2017 IEEE International Congress on Internet of Things (ICIOT)*, Honolulu, HI, USA, 2017, pp. 33-41

[4] – T. Nguyen Gia *et al*., "Low-cost fog-assisted health-care IoT system with energy-efficient sensor nodes," *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Valencia, Spain, 2017, pp. 1765-1770

[5] – Singh, Parminder & Nayyar, Anand & Kaur, Avinash & Ghosh, Uttam. (2020). Blockchain and Fog Based Architecture for Internet of Everything in Smart Cities. Future Internet. 12. 1-12

[6] – Omar, Ahmad Sghaier, and Otman Basir. "Identity management in IoT networks using blockchain and smart contracts." In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 994-1000. IEEE, 2018

[7] – Ahmad, Nazrul M., Siti Fatimah Abdul Razak, Subarmaniam Kannan, Ibrahim Yusof, and Anang Hudaya Muhamad Amin. "Improving identity management of cloud-based IoT applications using blockchain." In *2018 International Conference on Intelligent and Advanced System (ICIAS)*, pp. 1-6. IEEE, 2018.

[8] – Shobana, G., and M. Suguna. "BlockChain Technology towards Identity Management in Health Care Application." In 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), pp. 531-535. IEEE, 2019

[9] – Zhu, Xiaoyang, and Youakim Badr. "Fog computing security architecture for the internet of things using blockchain-based social networks." In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1361-1366. IEEE, 2018

[10] – Sun, Shuang, Rong Du, Shudong Chen, and Weiwei Li. "Blockchain-Based IoT Access Control System: Towards Security, Lightweight, and Cross-Domain." IEEE Access (2021)

[11] – Cícero A. Silva, Gibeon S. Aquino, Sávio R. M. Melo, Dannylo J. B. Egídio, "A Fog Computing-Based Architecture for Medical Records Management", Wireless Communications and Mobile Computing, vol. 2019

[12] – O. Bouachir, M. Aloqaily, L. Tseng and A. Boukerche, "Blockchain and Fog Computing for Cyberphysical Systems: The Case of Smart Industry," in Computer, vol. 53, no. 9, pp. 36-45, Sept. 2020

[13] – Awaisi, Kamran & Hussain, Shahid & Ahmed, Mansoor & Khan, Arif & Ahmed, Ghufran. (2020). Leveraging IoT and Fog Computing in Healthcare Systems. IEEE Internet of Things Magazine. 3. 52-56

[14] – Wang, Haoyu & Wang, Lina & Zhou, Zhichao & Tao, Xueqiang & Pau, Giovanni & Arena, Fabio. (2019). Blockchain-Based Resource Allocation Model in Fog Computing. Applied Sciences

[15] – M. H. Ziegler, M. Großmann and U. R. Krieger, "Integration of Fog Computing and Blockchain Technology Using the Plasma Framework," 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Seoul, Korea (South), 2019, pp. 120-123

[16] – Tanweer Alam. IoT-Fog: A Communication Framework using Blockchain in the Internet of Things,Authorea. June 08, 2020

[17] – A. Muthanna et al., "Secure and reliable IoT networks using fog computing with software-defined networking and blockchain", J. Sensor Actuator Netw., vol. 8, no. 1, pp. 15, 2019.

[18] – S. Tuli, N. Basumatary, S. S. Gill, M. Kahani, R. C. Arya, G. S. Wander, et al., "HealthFog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated IoT and fog computing environments", Future Gener. Comput. Syst., vol. 104, pp. 187-200, Mar. 2020.

[19] – L. Hang and D.-H. Kim, "Design and implementation of an integrated IoT blockchain platform for sensing data integrity", Sensors, vol. 19, no. 10, pp. 22-28, 2019.

[20] – H. L. Cech, M. Grossmann and U. R. Krieger, "A fog computing architecture to share sensor data by means of blockchain functionality", Proc. IEEE Int. Conf. Fog Comput. (ICFC), pp. 31-40, Jun. 2019