future internet

MDPI

Article

Prototyping a Hyperledger Fabric-Based Security Architecture for IoMT-Based Health Monitoring Systems

Filippos Pelekoudas-Oikonomou [1,2,*], José C. Ribeiro [1], Georgios Mantas [2], Georgia Sakellari [2] and Jonathan Gonzalez [1]

[1] Evotel Informática S.A., 27400 Monforte de Lemos, Lugo, Spain; jose@evotel-info.com (J.C.R.); jgonzalez@evotel-info.com (J.G.)
[2] Faculty of Engineering and Science, University of Greenwich, Chatham Maritime, Kent ME4 4TB, UK; g.mantas@greenwich.ac.uk (G.M.); g.sakellari@greenwich.ac.uk (G.S.)
* Correspondence: filippos@evotel-info.com

Abstract: The Internet of Medical Things (IoMT) has risen significantly in recent years and has provided better quality of life by enabling IoMT-based health monitoring systems. Despite that fact, innovative security mechanisms are required to meet the security concerns of such systems effectively and efficiently. Additionally, the industry and the research community have anticipated that blockchain technology will be a disruptive technology that will be able to be integrated into innovative security solutions for IoMT networks since it has the potential to play a big role in: (a) enabling secure data transmission, (b) ensuring IoMT device security, and (c) enabling tamper-proof data storage. Therefore, the purpose of this research work is to design a novel lightweight blockchain-based security architecture for IoMT-based health monitoring systems leveraging the features of the Hyperledger Fabric (HF) Platform, its utilities. and its lightweight blockchain nature in order to: (i) ensure entity authentication, (ii) ensure data confidentiality, and (iii) enable a more energy-efficient blockchain-based security architecture for IoMT-based health monitoring systems while considering the limited resources of IoMT gateways. While security mechanisms for IoT utilizing HF do exist, to the best of our knowledge there is no specific HF-based architecture for IoMT-based health monitoring systems.

Keywords: Internet of Medical Things; blockchain; hyperledger fabric; healthcare

Citation: Pelekoudas-Oikonomou, F.; Ribeiro, J.C.; Mantas, G.; Sakellari, G.; Gonzalez, J. Prototyping a Hyperledger Fabric-Based Security Architecture for IoMT-Based Health Monitoring Systems. *Future Internet* **2023**, *15*, 308. https://doi.org/10.3390/fi15090308

Academic Editor: Matthew Pediaditis

Received: 3 August 2023
Revised: 24 August 2023
Accepted: 5 September 2023
Published: 11 September 2023

1. Introduction

In recent years, the Internet of Things (IoT) technology has emerged and grown rapidly, bringing significant benefits to the healthcare sector by transforming the healthcare industry and introducing the Internet of Medical Things (IoMT), where medical devices are interconnected so that anyone, anywhere, and at any time may have access to them [1]. The evolution and growth of IoMT networks can play a significant role in enhancing the quality of life of individuals by enabling IoMT-based health monitoring systems that deliver personalized and user-centric healthcare services despite time and location restrictions [2–4]. However, the high resource requirements of existing security solutions cannot be afforded by (i) the resource-constrained IoMT devices (e.g., bio-sensors), which are key components of IoMT-based health monitoring systems but characterized by limited processing power, storage capacity, and battery life, and/or (ii) the constrained environment in which the IoMT devices are deployed and interconnected using lightweight communication protocols [3,5–8]. Furthermore, the centralization approach commonly adopted by the state-of-the-art existing security frameworks is not easily applicable to IoMT-based health monitoring systems due to single point of failure issues that may render them vulnerable to different attacks, such as Denial of Service (DoS) attacks [7,9]. In addition, it is worth mentioning that conventional defense mechanisms cannot ensure tamper-proof data storage [10]. Therefore, it is clear that novel security mechanisms are urgently needed

so that the pressing security challenges of IoMT-based health monitoring systems, relying on IoMT networks, can be addressed in an effective and efficient manner. This must be performed while taking into consideration the inherent limitations of the IoMT networks and devices, due to their resource-constrained characteristics and the centralized nature of IoMT-based health monitoring systems, before these systems gain the trust of all involved stakeholders and reach their full potential in the healthcare market [6,8,11].

To this end, blockchain technology has been foreseen by the industry and research community as a disruptive technology that can be integrated into novel security solutions for IoMT-based health monitoring systems in order to: (a) enable IoMT devices to transmit data to each other in a secure manner (e.g., tamper-proof transmission of medical data, no risk of DoS attacks) [3], given its decentralized, autonomous, and cryptographically secure nature, (b) ensure IoMT device security, and (c) enable tamper-proof data storage [12–14].

However, despite the significant benefits that the integration of blockchain technology can bring to the current, centralized IoMT-based health monitoring systems by addressing security challenges related to single point of failure issues and data storage, the resource-constrained IoMT devices of these systems are still unable to afford complex and high energy-consuming blockchain operations (e.g., the mining process in Proof of Work (PoW)) because of their limited processing power, storage capacity, and battery life [15–18]. Consequently, it is essential for lightweight blockchain-based security mechanisms to be applied so that they can be efficiently supported by the resource-constrained IoMT devices within IoMT-based health monitoring systems. However, so far, and to the best of our knowledge [19], there are only few works on blockchain-based authentication and authorization mechanisms as well as on blockchain-based Intrusion Detection Systems that can be considered for protecting IoMT-based health monitoring systems, demonstrating the lack of proper works on lightweight blockchain-based security for IoMT-based health monitoring systems.

Towards this end, as a major initial step, the design and implementation of an energy-efficient blockchain-based security architecture for supporting the development of lightweight blockchain-based security mechanisms for IoMT-based health monitoring systems is of utmost importance. Our motivation lies in the lack of proper works on lightweight blockchain-based security for IoMT networks and particularly for IoMT-based health monitoring systems. Specifically, the present research work provides the following contributions:

1. The design and implementation of an energy-efficient blockchain-based security architecture that will rely on the Hyperledger Fabric (HF) platform, as it does not only ensure entity authentication (i.e., device authentication) and data confidentiality (i.e., confidentiality of the exchanged data) but also can enable a more energy-efficient blockchain-based security architecture for IoMT-based health monitoring systems compared to other popular blockchain platforms, such as Ethereum, which applies the consensus protocol of Proof of Work (PoW), which cannot be afforded by resource-constrained IoMT devices.
2. The performance evaluation of the proposed HF-based security architecture for IoMT-based health monitoring systems with a focus on the results in terms of latency (i.e., min (s), max (s), and average (s)), send rate (i.e., transactions per second), throughput (i.e., transactions per second), and memory usage (MB)). However, there are no similar architectures to the one proposed in the literature, and thus we cannot compare the evaluation results of this research work directly to evaluation results from other research works. Nevertheless, we compare our performance evaluation results to results from works on IoT networks for real-time healthcare applications and from a work on the performance evaluation of HF for use in IoT as presented in Section 6.

Following this introduction, the paper is organized as follows. In Section 2, we present a literature review of HF-based security mechanisms and HF security schemes for IoT. In Section 3, we give a brief overview of HF, its components, and its functionalities. In Section 4, the proposed HF-based security architecture (i.e., HF-based network) for IoMT-

based health monitoring systems, along with its main system components, is presented. In Section 5, we present the performance evaluation of the proposed HF-based architecture (i.e., HF-based network). Finally, Section 6 concludes the paper.

2. Related Work

In this section, we present a literature review of HF-based security mechanisms and schemes for IoT.

It is worth mentioning that, although HF-based mechanisms for IoT exist, there is a lack of HF-based security architectures for IoMT-based health monitoring systems, and for this reason, in this section, we will review HF-based mechanisms (authentication, authorization, access control, and data integrity) and HF-based security schemes for IoT that exist in the literature [19].

2.1. HF-Based Authentication and Authorization

D. Li et al. [20] proposed a blockchain-based authentication mechanism for IoT in order to eliminate the single point of failure. In their proposed research, they refer to the necessity of device authentication without the use of a central authority, which is used in the traditional Public Key Infrastructure mechanisms (PKI). Blockchain technology is suitable in this architecture and provides the decentralized network structure. The implementation of the blockchain-based authentication mechanism has been conducted with the use of Raspberry Pi devices and the HF platform.

Babu Erukala Suresh et al. in [21], proposed a distributed identity-based authentication scheme for IoT devices using permissioned blockchain. The proposed system uses blockchain as a distributed Private Key Generator (PKG), which eliminates the single point of failure and key escrow problem. The proposed authentication scheme has been implemented in HF.

Shohei Kakei et al. in [22], suggest a distributed authentication infrastructure that distributes trust points among multiple service providers and connects them through cross-certification. The proposed method creates a unified framework for regulating cross-certification in a unique way. The paper proposes a new decentralized trust model called Meta-PKI that aims to overcome the overconcentration of trust in Certification Authorities (CAs).

Siris V. et al. in [23], propose a model for decentralized authorization in constrained IoT environments that uses multiple authorization servers (ASs) and two blockchains, one for authorization and the other for payments. The proposed models aim to reduce transaction costs and delays compared to a single (public) blockchain. The paper investigates interledger mechanisms for securely linking transactions on two blockchains: a private or permissioned chain and a public chain. The evaluation of the proposed models has been carried out on two public Ethereum testnets: Rinkeby and Ropsten and HF. The evaluation considers the execution cost (gas), delay, and reduction in the amount of data that needs to be sent to IoT devices.

Houshyar Pajooh et al. in [24], propose and implement an integrated IoT system that uses HF to secure edge computing devices while providing traceability for the data generated by IoT devices. The proposed model addresses scalability challenges, processing power, and storage issues of IoT edge devices in the blockchain network. The paper also presents a lightweight mutual authentication and authorization model and an HF blockchain middleware module embedded in IoT gateways, ensuring secure data transactions for the IoT-distributed applications.

2.2. HF-Based Access Control

The authors of [25] present the development of an attribute-based access control (ABAC) mechanism using Hyperledger Fabric components for IoT devices in a blockchain network. The article uses Raspberry Pi 4 Model B based on ARM64 architecture as the IoT device and evaluates the HF blockchain implementation and access control mechanism on

the ARM64 architecture. The article demonstrates a real-world IoT-blockchain integration scenario, and through effective chaincode execution and testing, the authors successfully assess the ABAC mechanism using HF components.

Han Liu et al. [26] present the design and implementation of Fabric-IoT, a blockchain-based access control system for IoT that aims to solve the access control problem in IoT by providing dynamic access control management and record tracing using distributed architecture. The system is based on the HF platform and the Attribute-Based Access Control (ABAC) model.

D. H. Shih et al. in [27], propose an access control system for the Industrial Internet of Things (IIoT) based on blockchain technology and attribute-based access control (ABAC). The proposed access control system aims to provide decentralized, fine-grained, and dynamic access control management for IIoT to address security issues and ensure secure transactions between manufacturers.

The authors of [28] propose an Attribute-Based Access Control Model for Internet of Things using Hyperledger Fabric Blockchain (ABAC-HLFBC). By using smart contracts, they have implemented fine-grained and expressive access control. The ABAC model extracts the attributes of the subject, object, permission, and environment and transforms permission management into attribute management. The proposed model can provide a distributed and lightweight secure access control solution for IoT that overcomes traditional centralized access control issues.

J. Maeng et al. in [29], propose and implement a lightweight group management (H-LGM) model for IoT devices based on HF. The contribution of the proposed research work includes the proposal of lightweight rekeying to reduce the update overhead of group key (GK) and H-LGM, the use of GK to organize and manage groups, the use of an agent for lightweight rekeying, and the demonstration of H-LGM's effectiveness.

2.3. HF-Based Data Integrity

The authors of [30] propose a scheme to protect the identity privacy of IoT users by preventing identity leakage in transactions. The study uses the ring signature method to obscure the real identity of the user who proposes the transaction, and the aggregated signature method to shorten the time and space required for k signature verification to 1/k, improving the efficiency of the system. The correctness and efficiency of the scheme are also proved through theoretical analysis and experiments.

Ning Lu et al. in [31], present a proposed decentralized data integrity auditing scheme, called HF-Audit, that uses HF to establish two separate communication channels for User-TPA(third-party auditor)-CSP(cloud service provider). HF is chosen as the communication platform in HF-Audit because of its attributes such as tamper-proofing, access permission, anonymity, efficient processing, and private channel features.

The authors of [32] propose a secure data transfer scheme based on Hyperledger Fabric blockchain for enterprise data sharing in the Industrial Internet of Things (IIoT). The raw data collected by enterprises is encrypted and stored in the InterPlanetary File System (IPFS) network, while the keyword-index table is designed in HF blockchain to enable data sharing among enterprises. The framework comprises three layers: Hyperledger Network Layer, Client Layer, and Storage Layer. The proposed scheme constitutes a significant contribution to secure data sharing, privacy protection, and scalability in IIoT's applications.

The aforementioned related work is summarized in Table 1.

Table 1. HF-based mechanisms and schemes for IoT.

Reference	Type of Security	Key Characteristics
[20]	Authentication mechanism	Decentralization, simplicity, general application
[25]	Attribute-based access control (ABAC) mechanism for IoT devices	Development of ABAC mechanism, access control system design and policies, rules for network and channel settings, signature and Implicit Meta policies, resource access control lists (ACLs)
[24]	Authentication and Authorization model	Securing edge computing devices, traceability of IoT data, scalability, smart-contract queries
[29]	Lightweight group management (H-LGM) model	Group management for IoT devices, lightweight rekeying, group key (GK) utilization, agent-based rekeying
[26]	Blockchain-based access control system for IoT	Dynamic access control management, record tracing, Attribute-Based Access Control (ABAC) model, secure sharing of data resources, simplified sharing mode and storage structure, device access control policy based on ABAC
[27]	Access control system for Industrial Internet of Things (IIoT)	Decentralized, fine-grained, and dynamic access control management, attribute-based access control (ABAC), three smart contracts for policy management, resource URL storage, and access control
[21]	Distributed identity-based authentication scheme	Distributed PKG, elimination of single point of failure and key escrow problem, security protocol for IoT device authentication using IBE
[22]	Distributed authentication infrastructure	Distribution of trust points, cross-certification, decentralized trust model (Meta-PKI)
[30]	Privacy protection scheme with ring signature and aggregated signature	Identity privacy protection, prevention of identity leakage, ring signature method, aggregated signature method, accountability mechanism, three-role scheme architecture
[23]	Decentralized authorization model with multiple ASs and two blockchains	Multiple authorization servers (ASs), two blockchains for authorization and payments, interledger mechanisms, OAuth 2.0 delegated authorization framework, CBOR Web Token (CWT) format
[28]	Attribute-Based Access Control Model for IoT	Fine-grained and expressive access control, attribute-based permission management
[31]	Decentralized data integrity auditing scheme	Utilization of separate communication channels for User-TPA-CSP, mitigating risks of privacy breaches
[32]	Secure data transfer scheme	Utilization of IPFS for encrypted storage, design of Keyword-index, use of Chaincode and ECDSA for data integrity and mutual authentication

3. Hyperledger Fabric

In this section we present Hyperledger Fabric (HF) basic components and functionalities, as well as the reasons it constitutes a suitable platform for deploying our proposed architecture.

Hyperledger Fabric has been proposed by Androulaki et al. [33] and developed by Linux Foundation [34], and it is a distributed ledger platform for developing applications with modular architecture [35]. This platform provides pluggable consensus protocols (mainly PBFT-based) and a private-permissioned blockchain model. It is suitable for deploying IoT applications for stakeholders that partially trust each other. This implementation platform has low scalability due to the nature of PBFT algorithms and 33.33% (1/3) adversary tolerance. However, it provides high privacy and throughput and supports the development of smart contracts.

Support for pluggable consensus protocols is one of the platforms most distinguished features. This support makes it possible to tailor the platform more effectively to specific

use cases and trust models. For instance, fully byzantine fault-tolerant consensus may be deemed redundant and an undue drag on performance and throughput when it is deployed within a single business or when it is controlled by a trusted authority. In circumstances such as these, a consensus protocol that is crash fault-tolerant (CFT) may be more than sufficient; however, in a multi-party, decentralized use case, a consensus protocol that is byzantine fault-tolerant (BFT) may be necessary [36]. For all these reasons, HF can be a viable blockchain platform to develop upon lightweight blockchain-based security solutions for IoMT networks [15,37].

In HF, the workflow from the initiation of a transaction to the update of the ledger involves several stages, as shown in Figure 1. As a first step, a transaction is initiated by a client. Secondly, the transaction is sent along with other transactions into the endorsing peers. Then it is endorsed by the set of designated endorsing peers according to the specified endorsement policy. These peers simulate the execution of the transaction and generate a proposal response that includes the outcome and read/write sets. Following this, the transactions along with the collected endorsed proposal responses are sent to the ordering service. The ordering service establishes a consensus on the order of the transactions and assembles them into a block. Then the block is appended to the chain and transferred to the peers that update their local copy of the ledger and endorse the updates. The ledger is then updated across all peers, ensuring the consistency of the data among the network participants. This workflow guarantees a secure, transparent, and auditable process for updating the ledger in HF.

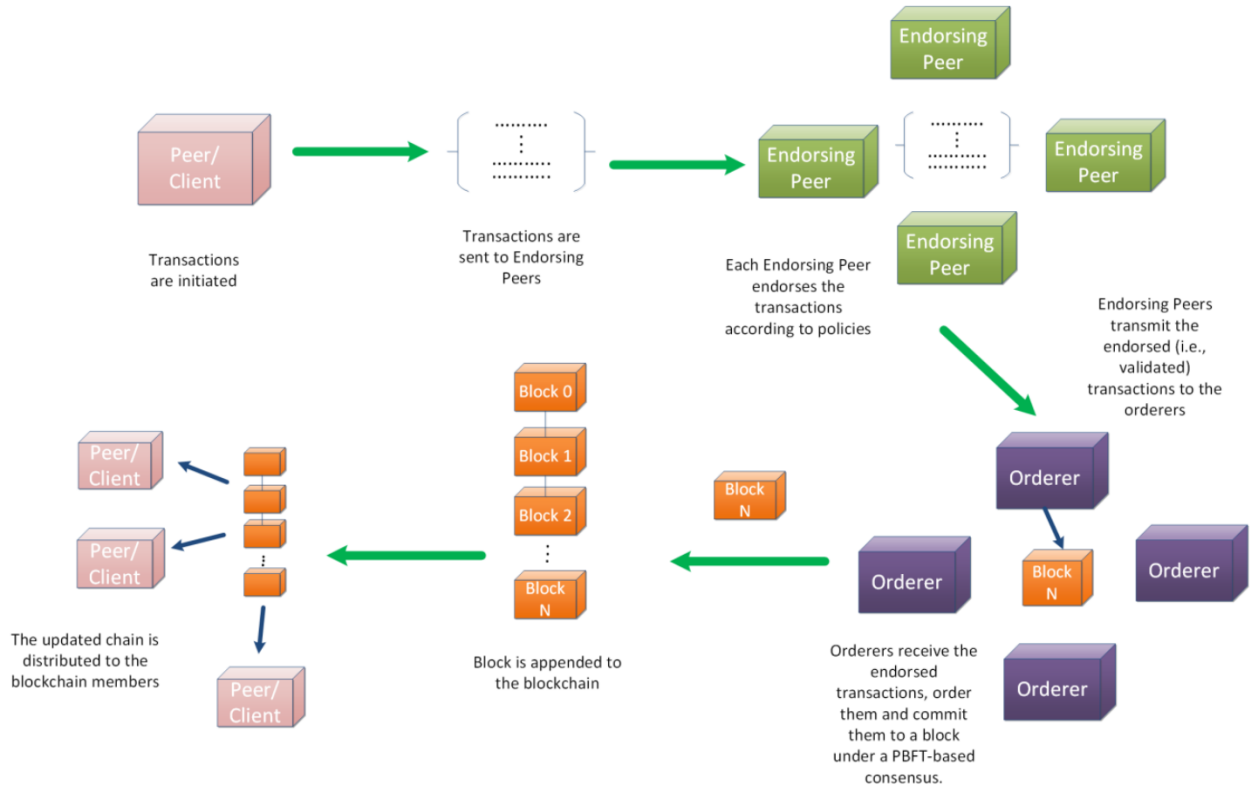


Figure 1. Transaction Workflow in HF.

The main components of HF are the following:

Blockchain Network: This can be understood as a collection of nodes that form a Peer-to-Peer (P2P) network in which every node shares a common distributed ledger and complies to the state of the ledger through a consensus protocol. In the case of HF, the blockchain network can provide, besides the distributive ledger, the feature of chaincode, a form of smart contract, that can be utilized to generate transactions that are then transmitted to

each peer node in the network and immutably recorded on their copy of the distributed ledger.

Peer: This is the main component of the blockchain network. Peers are the parts of the network where the blockchain ledger and the chaincode are hosted. Peers can also host SDK and APIs, through which network users can interact with applications and services. Peers are separated into two categories: (a) anchor peers and (b) endorsement peers. The former are responsible for distributing the blocks to the latter, while endorsement peers are responsible for endorsing the chaincode that is invoked by clients. Endorsement policies are pre-specified by the chaincode and define the number of peers that are needed to execute and endorse the specific chaincode.

Ordering Service: Different from the permissionless blockchains (e.g., Bitcoin, Ethereum) that come to consensus with a probabilistic process, HF uses the orderer node that, as the name indicates, orders the transactions. The group of ordering nodes compose the ordering service. After the ordering of the transaction, the deterministic consensus of the Hyperledger Fabric follows. Ordering is taking place in the specific nodes, and it is separated from the endorsing of transactions that takes place in peers. HLF provides three implementations of ordering service: Solo, Kafka, and Raft [38].

Certificate authority (CA): This is a tool that, as the name indicates, generates certificates for admins, users, peers, orderers, or applications in the form of an X.509 certificate [39] to identify the aforementioned blockchain network entities. Besides the identity for the entities that is issued by CA, CA also defines the privileges of the entities over the network.

Chaincode: This is the piece of code that acts as an application and provides functionalities to the established blockchain network, and it is carried in a Docker container. For this implementation we are going to used node.js language for chaincode implementation, but in other cases chaincode can be written in programming languages such as Go or Java.

Channels: These provide the communication between the nodes of the network. They comprise organizations, peers for each member, and the distributed ledger, as well as the chaincode. Channels are the places where transactions are proposed and handled. In Hyperledger fabric a node can participate in more than one channel and transmit information and data privately.

Endorsement policies: Endorsement policies specify the number of peers on a channel that are necessary to execute the chaincode of a transaction and endorse the results of this execution for the transaction to be credited as valid. As part of the transaction validation process performed by the peers, each validating peer verifies that the transaction has the correct number of endorsements.

Membership Service Provider (MSP): MSP is a component of HF that abstracts membership activities. An MSP detaches all cryptographic processes and protocols underlying certificate issuance, certificate validation, and user authentication and allows peers to validate incoming transaction requests from clients and sign transaction outcomes. An MSP can establish its own concept of identity, as well as the rules by which identities are managed and authenticated.

As described, HF's unique qualities make it a highly scalable system for permissioned blockchains that supports changeable trust assumptions, enabling the platform to accommodate a vast array of industrial use cases (e.g., banking, supply chain and more) from which one of them is on the scope of this research work: healthcare. The lightweight nature of HF, as well as the features that were presented in this section, is what makes this platform a suitable choice for our HF-based security architecture for IoMT-based health monitoring systems, keeping into consideration both the design needs as well as the resource constraint nature of IoMT nodes.

The support for private transactions is one of the key reasons why HF is ideal for IoT networks. HF enables private transactions between specific parties, preserving the confidentiality of sensitive data while offering modular architecture, which allows it to be tailored to the exact requirements of an IoT network. Scalability is another significant property of HF that makes it suited for IoT networks. The volume of data created by an

IoT network grows in direct proportion to the number of devices in the network. HF is built to manage massive amounts of data and can be horizontally scaled to accommodate more devices and transactions. Finally, the consensus method of HF is well-suited for IoMT networks as it ensures that all network participants agree on the state of the ledger, which is critical in an IoMT network because numerous devices may perform transactions at the same time.

In conclusion, Hyperledger Fabric's support for private transactions, modular architecture, scalability, and consensus mechanisms make it an appropriate blockchain platform for IoT networks [12,19,33].

4. Proposed HF-Based Security Architecture

This section gives an overview of the architecture of a typical IoMT-based health monitoring system where the proposed blockchain-based security architecture will be considered to be deployed for supporting the development of lightweight blockchain-based security mechanisms in IoMT-based health monitoring systems. Then, an overview of the proposed blockchain-based security architecture, relying on the HF platform, along with a description of its main components, are given.

4.1. Overview of an IoMT-Based Health Monitoring Systems

In this Section, we present the architecture of an IoMT-based health monitoring system. Typically, the architecture of an IoT-based health monitoring system is divided into three domains: (1) perception domain; (2) network domain; and (3) cloud domain, as shown in Figure 2.

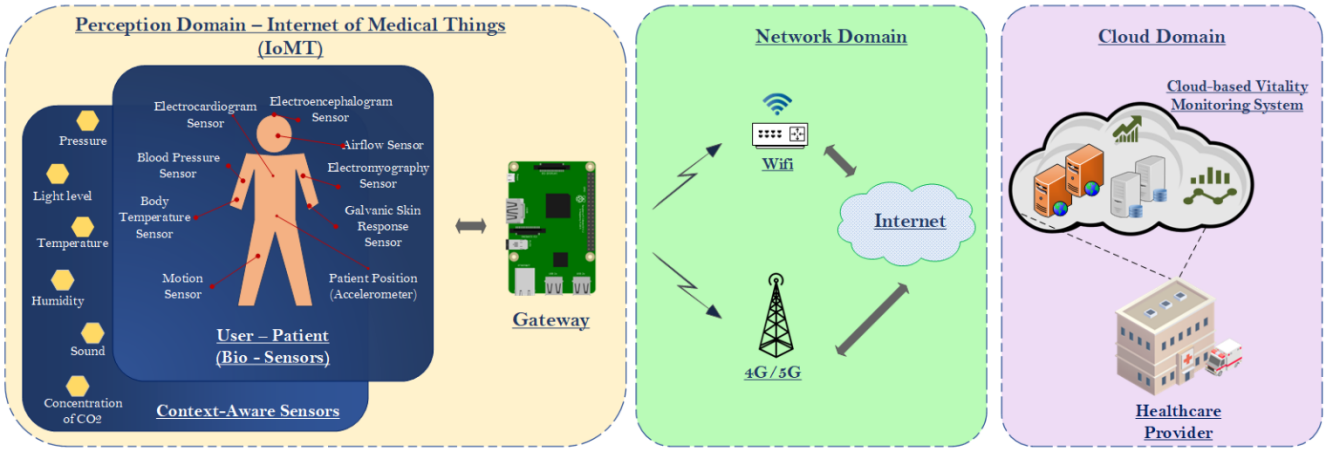


Figure 2. IoMT-based health monitoring system.

(a) Perception Domain: It is possible to understand this as the device layer in the ITU-T standard model [40] and as the IoMT edge network of the IoMT-based health monitoring system. The perception domain interacts with objects (e.g., physical things) via the IoMT devices (e.g., sensors, actuators, etc.) that are located on the IoMT edge network. The primary function of the perception domain is to connect things to the IoMT edge network. It is also responsible for measuring, gathering, and managing the information that is provided by these devices via IoT devices transmitting the information that has been gathered to higher domains through domain interfaces. Finally, the perception domain is home to the biosensors that are in charge of gathering the user/patient's vital signs as well as the context-ware sensors that are in charge of gathering context information from the user/patient's surrounding environment (e.g., air pressure, humidity, sound, etc.).

(b) Network Domain: This is the transmission domain, and in the IoMT-based health monitoring system architecture described above, it is implemented as the middle domain. The function of the network domain is to collect the data that has been obtained by the perception domain and to establish the routes that will be taken in order to send the data

that has been received to the Cloud domain via integrated networks. This particular sphere incorporates a wide range of tools and methods of communication, including Wi-Fi, 4G/5G, and the Internet, amongst others. In addition, it is the responsibility of the network domain to supply the relevant data services, such as data aggregation and fog computing.

(c) Cloud Domain: This then makes use of the data it has received from the network domain in order to give relevant cloud-based services or operations to the user/patient, to healthcare professionals, or to other authorized individuals (e.g., authorized relatives of the user/patient). For instance, the provided cloud-based services may include services such as the monitoring and assessment service of the user/patient's health status on the side of the healthcare professionals, the storage service where the received data are stored to databases in the cloud domain, and the analysis service to evaluate received data in order to provide predictions about the future state of sensing devices at the perception domain.

4.2. Overview of the Proposed HF-Based Security Architecture

The proposed HF-based security architecture for IoMT-based health monitoring systems is presented in Figure 3, and aims to: (i) achieve entity authentication (i.e., device authentication) as the HF platform allows the deployment of permissioned blockchain networks, (ii) enable IoMT devices to transmit data to each other in a secure manner (i.e., confidentiality of the exchanged data) due to the functionality of the HF platform to permit secure communication, via channels, between specific participants, (iii) eliminate single point of failure issues (e.g., no risk of DoS attacks), and (iv) achieve more secure (i.e., tamper-proof) data storage given its decentralized, autonomous, and cryptographically secure nature. The HF platform can enable a more energy-efficient blockchain-based security architecture for IoMT-based health monitoring systems, compared to other popular blockchain platforms such as Ethereum, as HF does not apply the consensus protocol of Proof of Work (PoW) that cannot be afforded by IoMT devices (e.g., medical sensors) due to their limited processing power, storage capacity, and battery life.

The proposed architecture can include multiple HF organizations (i.e., N HF organizations) that are interconnected through peers or orderers running on devices, such as gateways and servers, and operating under the HF platform on an HF-based network. In the context of HF, each organization is a member (e.g., healthcare provider, patient) that is eligible or invited to join the HF blockchain network. In addition, each organization can include multiple peers or orderers of different perception or cloud domains (e.g., K Perception domains in Organization 1 and M Cloud domains in Organization N). Owing to the functionalities of HF, an organization can contain Certificate Authorities (CAs), responsible for the entity authentication, orderers, or peers, and each of these entities can be interconnected with others through private channels inside the HF network.

Each organization in the proposed HF-based security architecture consists of the following key HF components as also shown in Figure 3:

Peer is the main component of the HF network that permits the hosting of (i) the blockchain ledger, (ii) the chaincode, and (iii) the endorsement policies, while permitting the communication of the user with the HF network through the SDK. A peer can be hosted on a gateway of a Perception Domain or on a server of a Cloud Domain of the proposed HF-based security architecture.

Orderer is a component responsible for managing and ordering the transactions that are added to the blockchain. An orderer is responsible to ensure the timely and accurate ordering of patient data and other transactions on the blockchain. The orderer could be responsible for receiving transactions, such as patient data and other healthcare-related information, from the gateways in the network and then validating the transactions to ensure that they meet the network's consensus rules. The validated transactions would then be added to the blockchain to maintain the integrity and consistency of the ledger. Similar to a peer, an orderer can be hosted on a gateway of a Perception Domain or on a server of a Cloud Domain of the proposed HF-based security architecture.

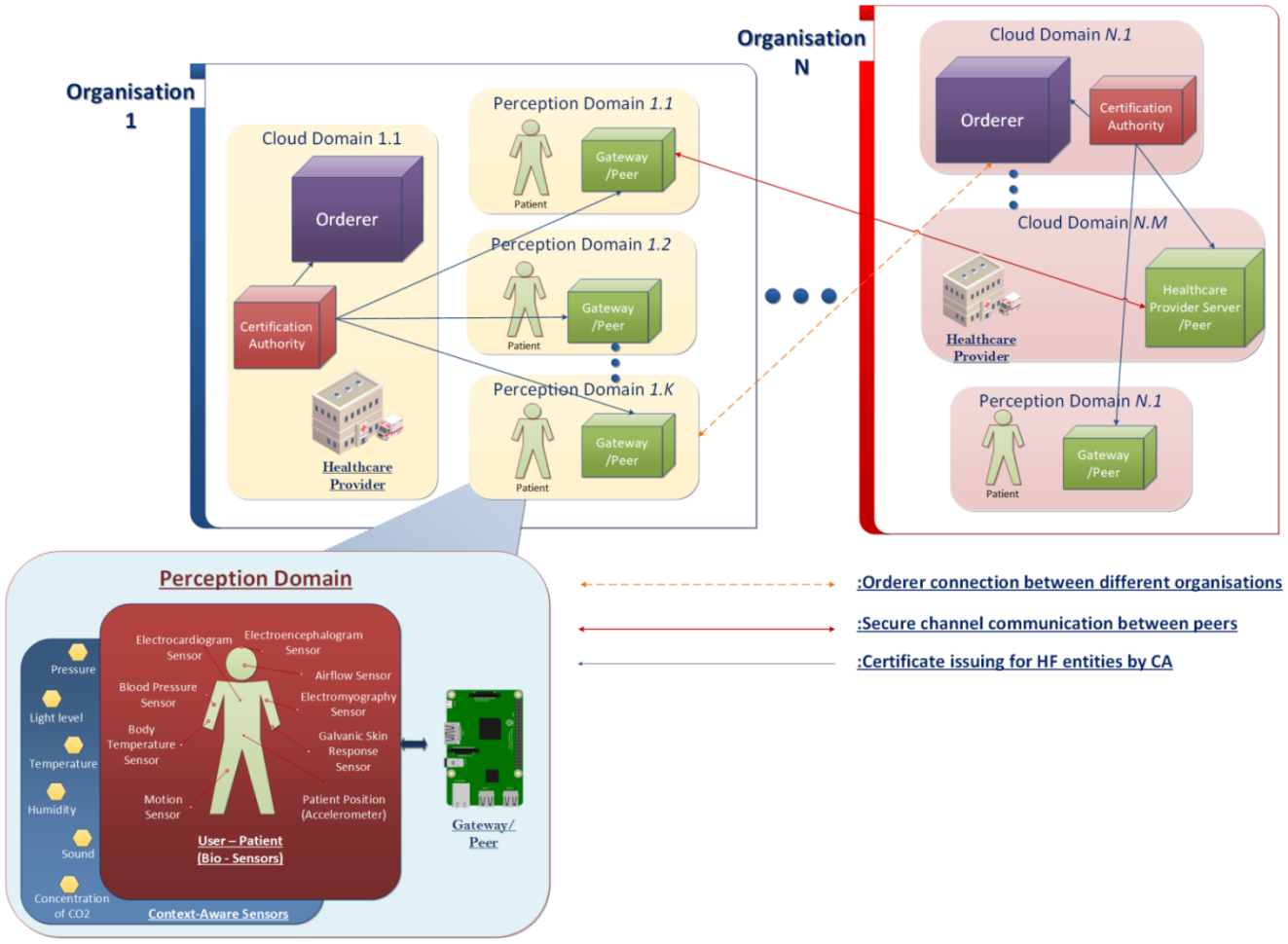


Figure 3. Overview of the HF-based security architecture for IoMT-based health monitoring systems.

Certificate Authority (CA) generates certificates for the entities of the network, such as admins, users, peers, orderers, or applications, that can be used to authenticate them. Thus, the generated certificates are used to ensure the legitimacy of the entities of the network so that they can communicate with each other and transmit data in a secure manner.

5. Simulation Set-Up of the Proposed HF-Based Security Architecture

In this section, we present the simulation set-up of the proposed architecture. In this simulation, we deployed an instance of the proposed HF architecture presented in Section 4 on a Linux virtual environment on a Virtual Machine (VM) where the HF platform and the necessary perquisite software (e.g., Docker, golang) were installed. The parameters of the VM are shown below in Table 2:

Table 2. Virtual Environment Parameters.

Feature	Specifications
Operating System	Ubuntu 22.04.2 AMD 64
RAM	4096 MB
Hyperledger Fabric	2.2
Docker	20.10.21
Node.js	v.12.22.9

The deployed HF network that represents the instance of the proposed architecture consists of the following components, as shown in Figure 4:

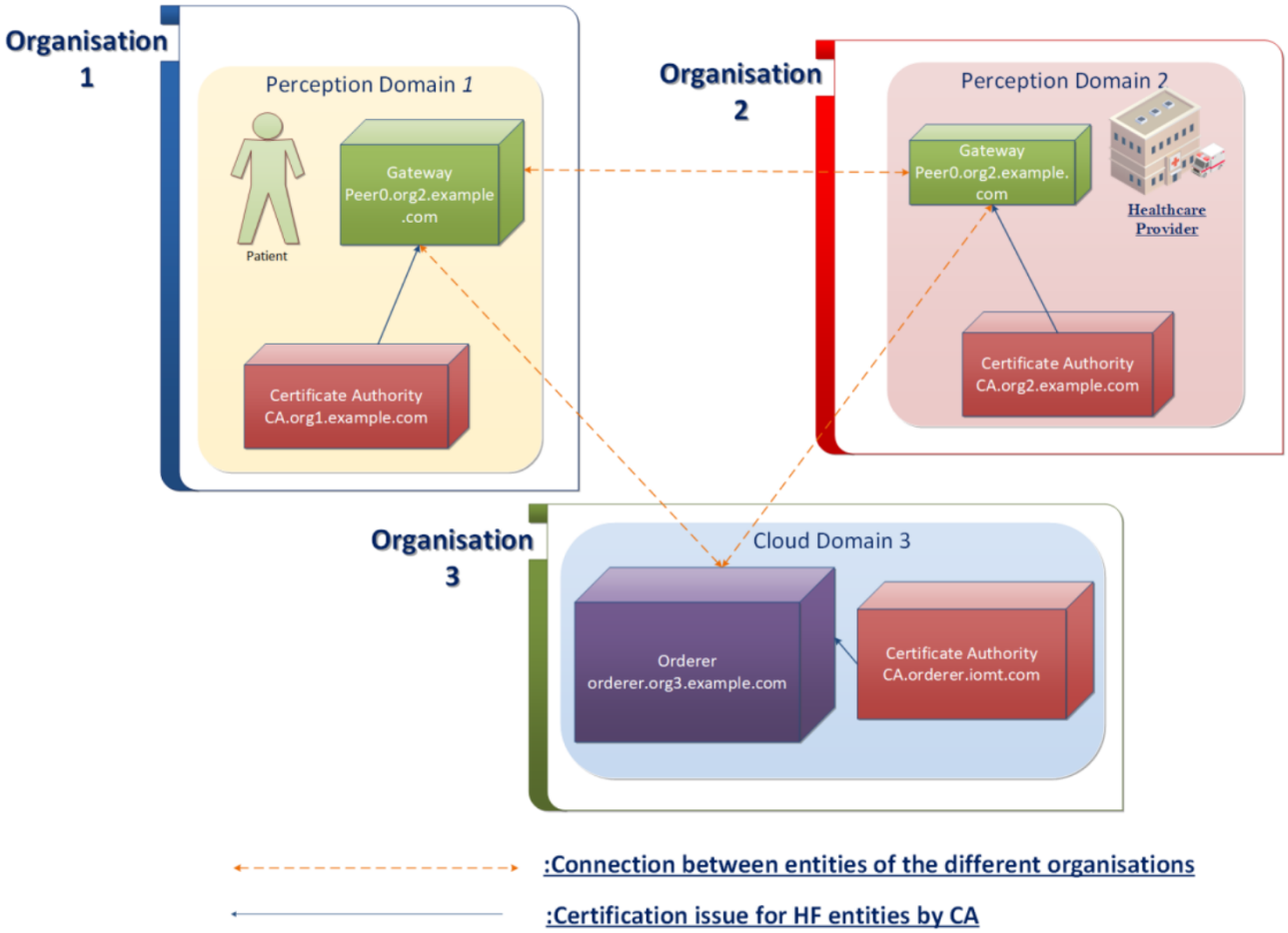


Figure 4. The deployed HF network.

- *Organization 1*, *peer0.org1*, representing the gateway of the patient in its respective perception domain;
- *Organization 2*, *peer0.org2*, representing the gateway of a healthcare provider that is part of the network;
- *Organization 3*, *orederer.org3*, representing the ordering service;
- A *CA* is included in each Organization (e.g., *ca_org1*, *ca_org2*, *ca_orderer*).
- We have also deployed the following necessary components:
- a *cli* that allows us to interact with the peer in order to connect it to a channel and initiate transactions;
- a *CouchDB*, which is the state database used in HF to store the current world state of the blockchain network; and
- the containers that include the chaincodes (i.e., *create_asset*, *read_asset*) that will run during the simulation base on which the evaluation results in terms of latency (i.e., min (s), max (s), and average (s)), send rate (i.e., transactions per second), throughput (i.e., transactions per second), and memory usage (MB) are collected.

The process for the simulation-set of the proposed network includes the following steps:

Step 1. As a first step, we create the Fabric CA servers by initiating the corresponding Docker containers, with the use of the HF binaries, in order to generate all the necessary certificates for the orderers and peers, as shown in Figure 5.

Step 2. After setting up the Fabric CA servers, we proceed to generate the cryptographic material for the network (i.e., TLS, MSP certificates). According to the HF documentation [41], we initially enroll the CA server by generating an admin certificate

which subsequently will be used to issue certificates and enroll the peers and the orderer in each organization, as well as to create a channel configuration that will define the initial state of the network. In Figures 6 and 7, we present an example of the enrolment of the CA admin and the org1 admin, respectively.

Step 3. After generating the certificates, as a next step we define the initial configuration of the network. This step contains the configuration of the orderer and the peers that is achieved through the creation of the *docker-compose.yaml* file that contains the necessary configuration parameters.

Step 4. Once the initial network configuration is defined, we proceed with the creation of a genesis block that contains information about the network, including the cryptographic materials, the policies, and other configuration settings defined and created in previous steps. The genesis block is created with the use of the *configtxgen* binary provided by the HF platform. Additionally, in this step we create also the *CouchDB* containers used as ledger state databases. Figure 8. presents the creation of the genesis block.

Step 5. This step involves the definition of the channel configuration, including the participating organizations and their respective peers as well as the orderer. In addition, in this step, the channel is created as shown in Figure 9. Once the channel is created, each peer can join the channel to establish communication within the HF network.

Step 6. In this step, we access the peer *cli* using the prompt command *docker exec -it cli*. Through the peer *cli*, we initiate the connection of the peer to the channel, as shown in Figure 10.

Step 7. This step involves the installation of the chaincode. At this point, we pack the chaincode into a Docker container, install it on the peers, and instantiate it on the channel. Then the initial state of the ledger is created and stored in the *CouchDB*. Figure 11 depicts the initiation of the installation of the chaincode in the channel, while Figure 12 depicts the final step of the process which is the successful approval of the chaincode to the channel and the commitment of the chaincode by both peers.

Step 8. Finally, as the chaincode is installed in the peers, the user can interact with the state of the ledger and initiate transactions through the peer *cli* by querying the chaincode.

```
LOCAL_VERSION=2.4.6
DOCKER_IMAGE_VERSION=2.4.6
CA_LOCAL_VERSION=1.5.5
CA_DOCKER_IMAGE_VERSION=1.5.5
Generating certificates using Fabric CA
Creating network "fabric_test" with the default driver
Creating ca_org2    ... done
Creating ca_org1    ... done
Creating ca_orderer ... done
```

Figure 5. Creating CAs (step 1).

```
Enrolling the CA admin
+ fabric-ca-client enroll -u https://admin:adminpw@localhost:7054 --caname ca-org1 --tls.certfiles /home/filippos/go/src/github.com/filippos/fabric-samples/test-network/organizations/fabric-ca/org1/ca-cert.pem
2023/04/10 14:29:10 [INFO] Created a default configuration file at /home/filippos/go/src/github.com/filippos/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2023/04/10 14:29:10 [INFO] TLS Enabled
2023/04/10 14:29:10 [INFO] generating key: &{A:ecdsa S:256}
2023/04/10 14:29:10 [INFO] encoded CSR
2023/04/10 14:29:10 [INFO] Stored client certificate at /home/filippos/go/src/github.com/filippos/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/msp/signcerts/cert.pem
2023/04/10 14:29:10 [INFO] Stored root CA certificate at /home/filippos/go/src/github.com/filippos/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2023/04/10 14:29:10 [INFO] Stored Issuer public key at /home/filippos/go/src/github.com/filippos/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/msp/IssuerPublicKey
2023/04/10 14:29:10 [INFO] Stored Issuer revocation public key at /home/filippos/go/src/github.com/filippos/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/msp/IssuerRevocationPublicKey
```

Figure 6. Enrolling the CA admin (step 2).

```
Generating the org admin msp
+ fabric-ca-client enroll -u https://org1admin:org1adminpw@localhost:7054 --caname ca-org1 -M /home/filippos/go/src/github.com/filippos/fabric-samples
/test-network/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp --tls.certfiles /home/filippos/go/src/github.com/filip
pos/fabric-samples/test-network/organizations/fabric-ca/org1/ca-cert.pem
2023/04/10 14:29:12 [INFO] TLS Enabled
2023/04/10 14:29:12 [INFO] generating key: &{A:ecdsa S:256}
2023/04/10 14:29:12 [INFO] encoded CSR
2023/04/10 14:29:12 [INFO] Stored client certificate at /home/filippos/go/src/github.com/filippos/fabric-samples/test-network/organizations/peerOrgani
zations/org1.example.com/users/Admin@org1.example.com/msp/signcerts/cert.pem
2023/04/10 14:29:12 [INFO] Stored root CA certificate at /home/filippos/go/src/github.com/filippos/fabric-samples/test-network/organizations/peerOrgan
izations/org1.example.com/users/Admin@org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2023/04/10 14:29:12 [INFO] Stored Issuer public key at /home/filippos/go/src/github.com/filippos/fabric-samples/test-network/organizations/peerOrganiz
ations/org1.example.com/users/Admin@org1.example.com/msp/IssuerPublicKey
2023/04/10 14:29:12 [INFO] Stored Issuer revocation public key at /home/filippos/go/src/github.com/filippos/fabric-samples/test-network/organizations/
peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp/IssuerRevocationPublicKey
```

Figure 7. Enrolling the org1 admin (step 2).

```
Generating channel genesis block 'mychannel.block'
/home/filippos/go/src/github.com/filippos/fabric-samples/test-network/../bin/configtxgen
+ configtxgen -profile TwoOrgsApplicationGenesis -outputBlock ./channel-artifacts/mychannel.block -channelID mychannel
2023-04-10 14:29:23.858 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2023-04-10 14:29:23.910 WEST 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: etcdraft
2023-04-10 14:29:23.916 WEST 0003 INFO [common.tools.configtxgen.localconfig] completeInitialization -> Orderer.EtcdRaft.Options unset, setting to tic
k_interval:"500ms" election_tick:10 heartbeat_tick:1 max_inflight_blocks:5 snapshot_interval_size:16777216
2023-04-10 14:29:23.916 WEST 0004 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: /home/filippos/go/src/github.com/filippos/
fabric-samples/test-network/configtx/configtx.yaml
2023-04-10 14:29:23.917 WEST 0005 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2023-04-10 14:29:23.922 WEST 0006 INFO [common.tools.configtxgen] doOutputBlock -> Creating application channel genesis block
2023-04-10 14:29:23.922 WEST 0007 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
+ res=0
```

Figure 8. Creating the genesis block (step 4).

```
Creating channel mychannel
Using organization 1
+ osnadmin channel join --channelID mychannel --config-block ./channel-artifacts/mychannel.block -o localhost:7053 --ca-file /home/filippos/go/src/git
hub.com/filippos/fabric-samples/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --client-cert /home/filip
pos/go/src/github.com/filippos/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/tls/server.crt
--client-key /home/filippos/go/src/github.com/filippos/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.exa
mple.com/tls/server.key
+ res=0
Status: 201
{
        "name": "mychannel",
        "url": "/participation/v1/channels/mychannel",
        "consensusRelation": "consenter",
        "status": "active",
        "height": 1
}

Channel 'mychannel' created
```

Figure 9. Channel creation (step 5).

```
Joining org1 peer to the channel...
Using organization 1
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=0
2023-04-10 14:29:30.119 WEST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2023-04-10 14:29:30.331 WEST 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
Joining org2 peer to the channel...
Using organization 2
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=0
2023-04-10 14:29:33.408 WEST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2023-04-10 14:29:33.734 WEST 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
Setting anchor peer for org1...
Using organization 1
Fetching channel config for channel mychannel
```

Figure 10. Joining *peer0.org1* into the channel (step 6).

```
Using docker and docker-compose
deploying chaincode on channel 'mychannel'
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: basic
- CC_SRC_PATH: ../asset-transfer-basic/chaincode-go/chaincode/
- CC_SRC_LANGUAGE: go
- CC_VERSION: 1.0
- CC_SEQUENCE: 1
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: NA
- DELAY: 3
- MAX_RETRY: 5
- VERBOSE: false
```

Figure 11. Chaincode installation process (step 7).

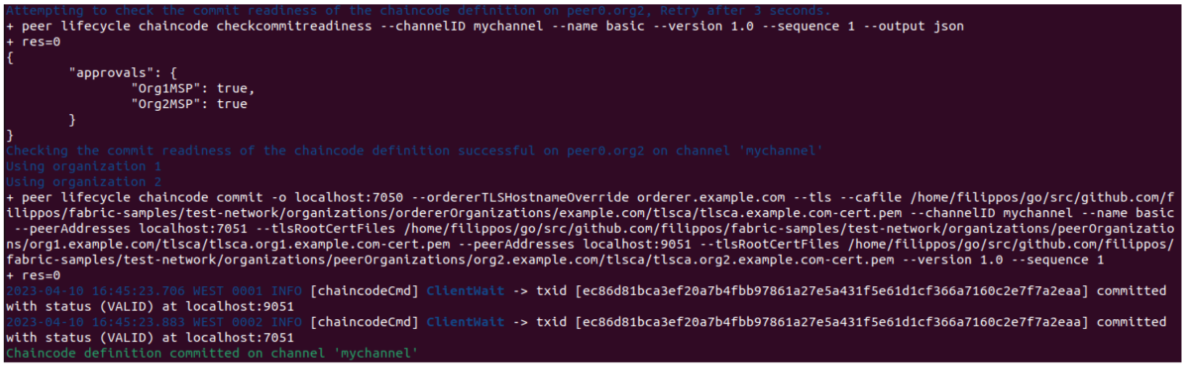


Figure 12. Chaincode approved and committed on channel (step 7).

In Figure 13 we present the containers deployed for the purpose of the HF network simulation:

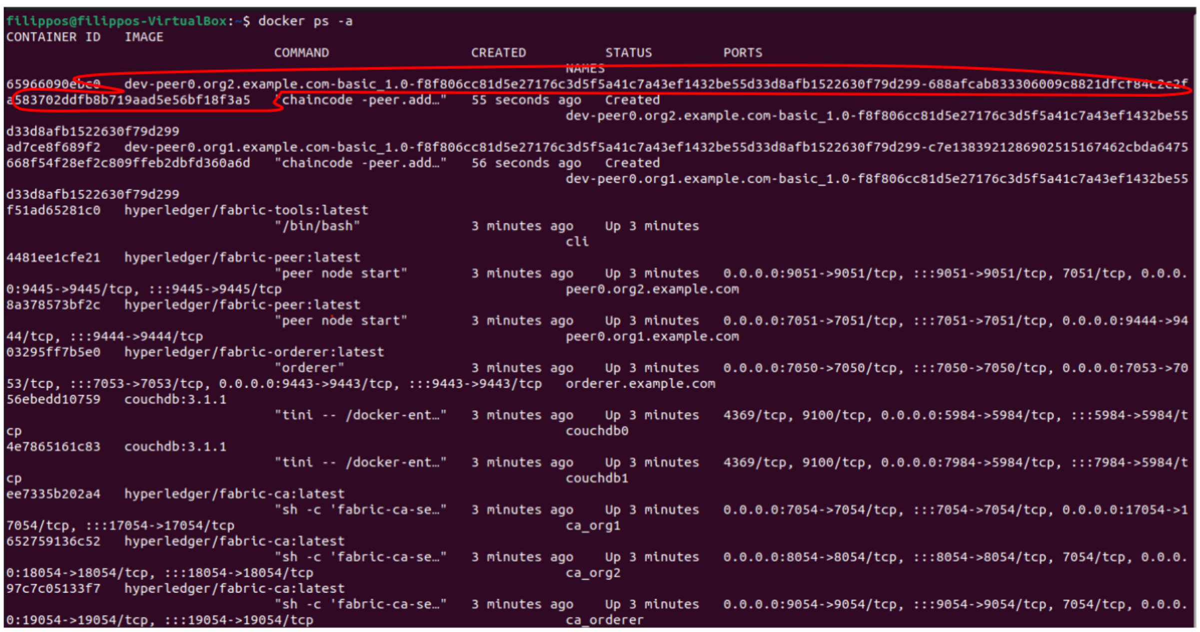


Figure 13. Docker containers for the performance evaluation.

It is worth mentioning that the aforementioned HF architecture can be secure and effective against eavesdropping, spoofing, and masquerading attacks due to the deployed Certificate Authority that provides TLS and MSP certificates to the nodes (i.e., peers, orderers) connected to the network. These certificates are necessary in order for a node to be able to securely communicate, transmit, or receive information in the network. Therefore, a non-certified, by the deployed CA, node is unable to participate in the network.

6. Performance Evaluation

To evaluate the deployed HF-network, as described in Section 5, we have used the Hyperledger Caliper blockchain benchmark tool [42]. Caliper is an open-source benchmark tool, hosted by the Linux Foundation, that enables users to evaluate the performance of different blockchain platforms and consensus algorithms in a standardized environment. It provides a set of predefined use cases and performance metrics while allowing users to define custom scenarios and execute them in order to evaluate them and collect data such as transaction throughput, response time and resource consumption. Caliper functions as a benchmark tool of various blockchain platforms, such as Hyperledger Fabric, Sawtooth, and Ethereum. Caliper can also function under various workloads and configurations [43,44].

In the context of this performance evaluation, we have assumed the test scenario where two peers are interconnected via a shared channel, as described in Section 5. These peers, which were developed in Section 5, are designed to correspond to the characteristics of two gateways within our proposed HF-based architecture. For the purpose of our performance evaluation, we have evaluated the network by collecting results in terms of latency (i.e., min (s), max (s), average (s)), send rate (i.e., transactions per second), throughput (i.e., transactions per second), and memory usage (MB). Regarding parameters, we measured network performance based on (i) "assets" (i.e., stored values) stored in the blockchain, and (ii) "workers" performing processes in parallel during the benchmark. The "workers" parameter in Caliper specifies the number of processes that will take place during the benchmark. Each "worker" process runs on a separate thread or CPU core, allowing the benchmark to use multi-core CPUs and distribute the workload across multiple threads. The use of multiple "workers" improves the benchmark's performance and scalability by allowing it to process transactions in parallel. However, it is worth noting that increasing the number of "workers" also increases memory usage. In general, the optimal number of "workers" for a given Caliper benchmark will depend on the specific blockchain network and workload being tested, as well as the available system resources.

The selection of "assets" parameter for evaluating network performance allows us to test the network's efficiency as the number of stored objects increases. On the other hand, the quantity of "workers" directly relates to the scalability potential of the architecture, showcasing the network's ability to handle elevated transaction volumes simultaneously. As a result, as stored objects and parallel transactions significantly impact the performance of a HF network, the results of the performance evaluation are directly relevant to real-world scenarios involving HF-based architectures. The conclusions drawn from these results are highly applicable to real-world implementation.

In our case, we have experimented with different values of the "workers" parameter on the network in order to evaluate the network and identify its optimal settings. We have experimented in instances of 1, 2, 5, 10, 15, and 20 "workers", while we have kept the number of the assets at 5. The "asset" parameter defines the number of assets (i.e., stored values) that are included in the ledger. We have experimented in instances of 5, 10, 25, 50, 75, 100, and 150 assets in each performance evaluation iteration, and we have maintained the number of "workers" at 4.

We have tested the following two chaincode functions in the present evaluation: (i) *create_asset* and (ii) *read_asset*. The *create_asset* function creates an asset and appends it to the ledger according to a corresponding asset template. On the other hand, the *read_asset* function reads the assets stored in the ledger.

As we plan, as future work, to integrate a Collaborative Intrusion Detection System (CIDS) in the proposed HF network architecture, the template of the created assets in the *create_asset* function is based on possible outputs that an CIDS may produce in order to be stored in the ledger. An example of the asset template for CIDS values stored in the ledger is shown in Figure 14 below:

In order to ensure the accuracy and reliability of the evaluation results derived from the utilization of the Caliper benchmarking tool, a comprehensive validation approach was undertaken. The process included the configuration of our HF-based network within a simulated environment, designed to emulate real-world settings. Moreover, the configuration details of the simulation, such as hardware specifications, network topology, and software versions, have been documented with the aim to facilitate the replication and validation of the experiment by fellow researchers.

Subsequent to the configuration phase, a series of multiple test runs were executed. This approach was adopted to determine any anomalies or performance variations in isolated test instances. Employing multiple test runs supported the accuracy of the conclusions drawn from the results, enhancing the reliability of the outcomes. Nevertheless, due to the fact that our experiments were conducted in a virtual environment, potential variations in the performance of the proposed HF-based architecture in a real-world context may occur.

```
class MyWorkload extends WorkloadModuleBase {
constructor() {
super();
this.txIndex = -1;
this.modelParams = []; // Array to store model parameters
this.evaluationMetrics = {}; // Object to store evaluation metrics
this.alerts = []; // Array to store alerts
this.logEntries = []; // Array to store log entries
}
```

Figure 14. Asset template on the *create_asset* function.

In the following section, we present the evaluation results in terms of latency (i.e., min (s), max (s), average (s)), send rate (i.e., transactions per second), throughput (i.e., transactions per second), and memory usage (MB). The evaluation results were generated using the MATLAB R2018b software based on the values collected by Caliper.

6.1. Latency

Latency is the time delay between the initiation of the transaction and its commitment to the blockchain.

In the case of the "assets" parameter, we have experimented in instances of 5, 10, 25, 50, 75, 100, and 150 assets in each performance evaluation iteration. We observe that, as we increase the number of assets, the latency in each evaluation iteration remains constant. Notably, in the case of *create_asset* function (Figure 15a), the average latency remains below 25 s. Similarly, for the of *read_asset* function (Figure 16a), the average latency remains stable, below 0.05 s, while we observe the higher value of the maximum latency of 0.3 s. This stable latency performance, while asset numbers are increasing, underscores the network's resilience and its capacity to efficiently handle a growing asset load.

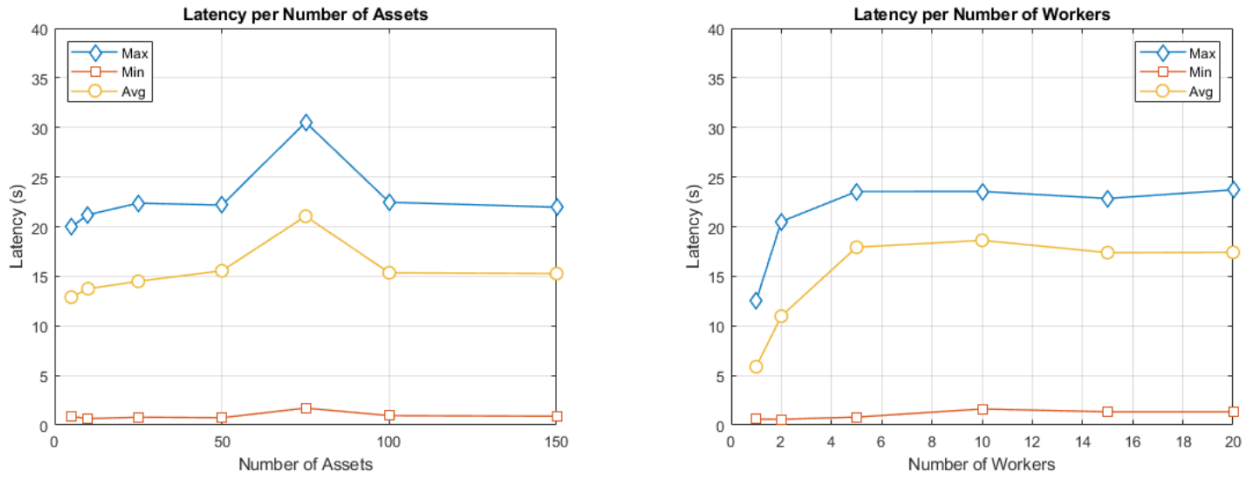
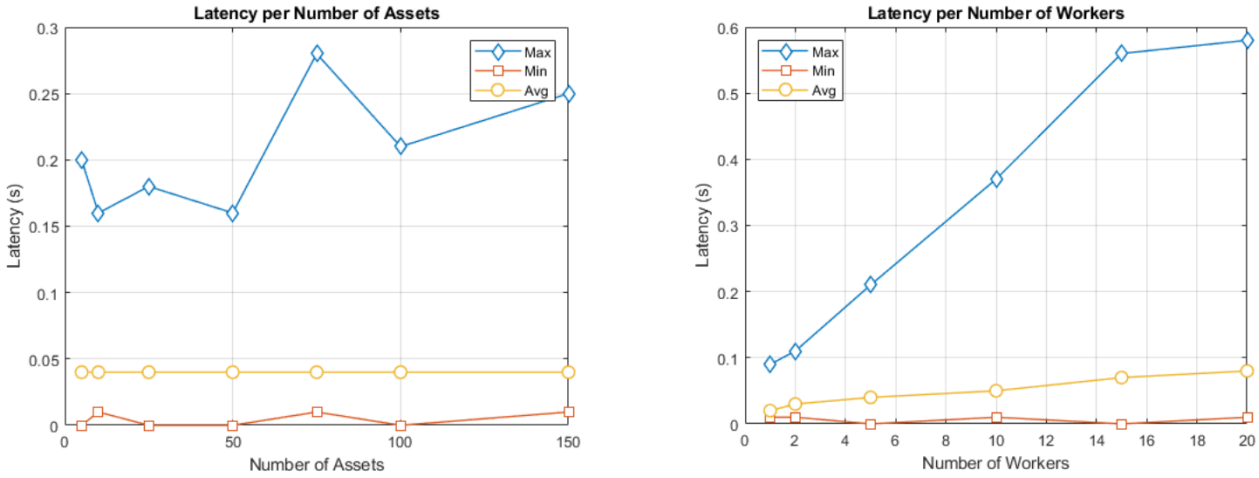


Figure 15. Network Latency (s), (**a**) per "asset" (**b**) per "worker", for *create_asset*.



(**a**) (**b**)

Figure 16. Network Latency (s), (**a**) per "asset" (**b**) per "worker", for *read_asset*.

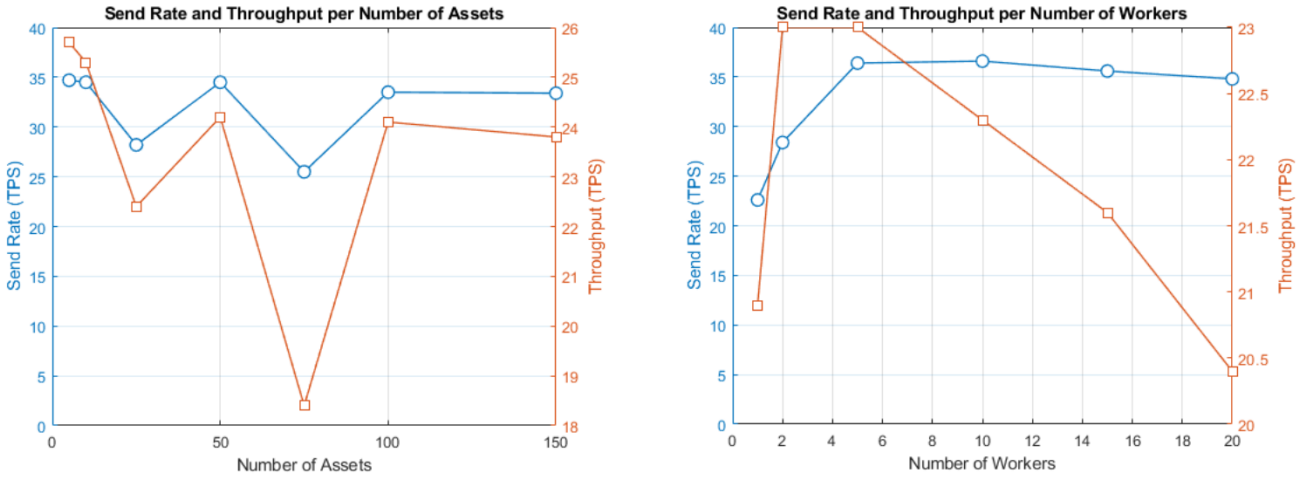
In the case of the "workers" parameter, we have experimented in instances of 1, 2, 5, 10, 15, and 20 "workers" in each performance evaluation iteration. The results demonstrate a steady increase in the maximum latency that is proportional to the increase in the value of the "workers" parameter in the case of *read_asset* (Figure 16b). However, the maximum latency does not surpass the 0.6 s and the average latency remains constantly below 0.1 s. On the other hand, as there will be more parallel processes available to process transactions and respond to requests, latency is kept at a constant value and less than 25 s in the case of the *create_asset* (Figure 15b). It is worthwhile to note that the average latency in the case of *create_asset* (Figure 15b) sees an increase up to the number of 5 "workers" and then remains stable below 20 s for the rest of the iterations. This observation reaffirms the adaptability and stability of the network under varied worker loads.

Due to the fact that there are no similar architectures to the proposed one in the literature, we cannot compare the evaluation results of this research work directly to evaluation results from other research works. However, we compared our performance evaluation results to results from works on IoT networks for real-time healthcare applications [45,46] and from a work on the performance evaluation of HF for use in IoT [38]. We observed similar values (i.e., minimum latency of 0.1 s [45] and 4 s [38]) and, thus, our performance evaluation of the proposed architecture has demonstrated satisfactory results in terms of latency in the cases of *create_asset* and *read_asset* functions. The values of latency are considered adequately low and within an acceptable range, indicating that the proposed HF architecture responds to requests in a timely manner.

6.2. Send Rate and Throughput

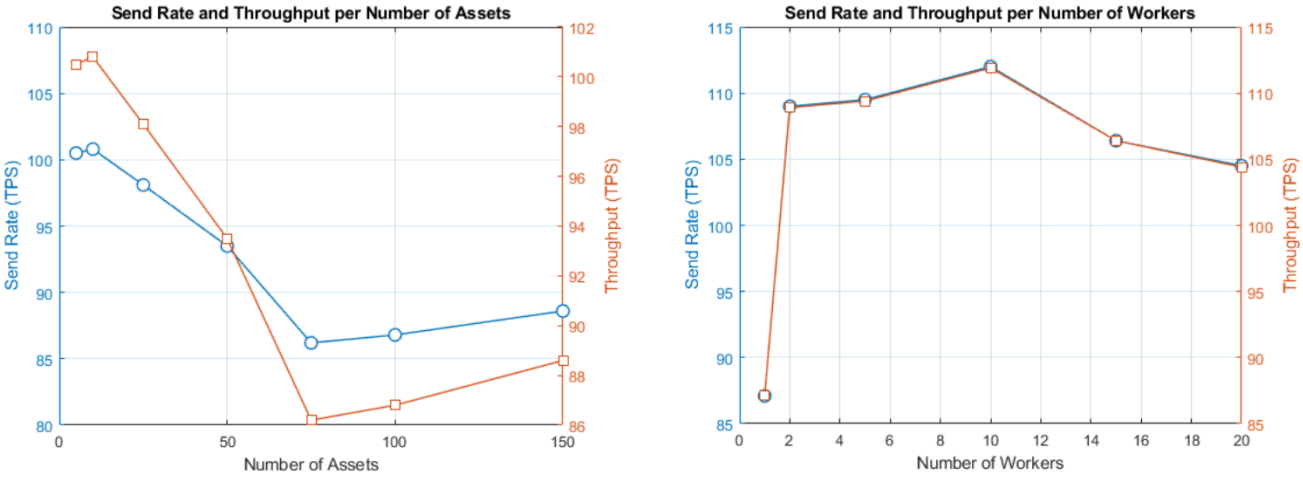
Send rate refers to the rate at which transactions can be submitted to the network for processing and it is measured in transactions per second (TPS). In contrast, *throughput* refers to the rate at which transactions can be successfully processed by a blockchain network within a given time period, typically measured in transactions per second (TPS).

In the case of the number of "assets", we can observe a decrease in the send rate as well as in the throughput proportional to the increase in the number of "assets" in both functions (i.e., *create_asset*, *read_asset*) (Figures 17a and 18a). However, even in the maximum value of 150 assets (Figure 18a), the simulation performs with a send rate of 91 TPS and a throughput of 89 TPS which can be considered an acceptable value.



(**a**) (**b**)

Figure 17. Send rate (TPS) and throughput (TPS), (**a**) per "asset" (**b**) per "worker", for *create_asset*.



(**a**) (**b**)

Figure 18. Send rate (TPS) and throughput (TPS), (**a**) per "asset" (**b**) per "worker", for *read_asset*.

In the case of the "workers" parameter, we observe that the value of the send rate and throughput is increased up to 110 TPS and then reached the value of 105 TPS, in the maximum number of the "workers" parameter (i.e., 20), in the case of *read_asset* function (Figure 18b). On the other hand, we observe a decrease in throughput while increasing the number of "workers" in the *create_asset* function (Figure 17b). We can conclude that, while reading the assets on the ledger, by increasing the number of the "workers" parameter, we can increase the throughput, while distributing the workload more evenly and increasing the total number of transactions that can be processed in a given time period.

Overall, the performance evaluation has demonstrated satisfactory results in terms of send rate and throughput, compared to a work on the performance evaluation of HF for use in IoT [38], in both cases (i.e., *create_asset*, *read_asset*) and we can conclude that the proposed HF architecture is capable of handling a significant amount of traffic without any distinguishable performance degradation and, thus, being suitable for the case of IoMT-based health monitoring systems.

6.3. Peer Memory Usage

Peer memory usage refers to the amount of memory consumed by a peer node in the blockchain network to store and manage ledger data, execute smart contracts, and maintain ledger state.

By increasing the "workers" parameter as well as the number of "assets", we observe an increase in the peer memory usage (Figure 19a,b). We can observe a maximum value of memory usage of 205 MB, which is approximately 4.99% of the total available memory of the gateway (i.e., 4096 MB). This is a feasible value for a peer node to function properly on a lightweight gateway (e.g., Raspberry Pi 4 [47] with up to 8 GB RAM).

(**a**) (**b**)

Figure 19. Peer memory usage (MB), (**a**) per "asset" (**b**) per "worker".

7. Conclusions and Future Work

In this paper, an HF-based security architecture for IoMT-based health monitoring systems has been proposed and deployed. With the proposal of this novel architecture, we have tried to address the security issues in the field of IoMT-based health monitoring systems and have tried to fill the gap caused by the lack of lightweight blockchain-based security architectures for IoMT-based health monitoring systems. The detailed description of its main components (i.e., perception domain, HF blockchain network), along with the interactions among them, have also been provided. In addition, a deployment of the simulation set-up of the proposed HF-based security architecture (i.e., HF-based network) for IoMT-based health monitoring systems has been described and evaluated in terms of latency (i.e., min (s), max (s), and average (s)), send rate (i.e., transactions per second), throughput (i.e., transactions per second), and memory usage (MB). However, due to the fact that there are no similar architectures to the proposed one in the literature, we cannot compare our evaluation results directly to evaluation results from other works. Nevertheless, we compared our results to results from works on IoT networks for real-time healthcare applications [45,46] and from a work on the performance evaluation of HF for use in IoT [38]. In this regard, our results demonstrate a good performance of the proposed architecture, allowing its use in resource-constrained environments, such as the IoMT, and enabling more advanced functionalities to be implemented on top of this architecture. It is worth noting that our experiments were conducted in a virtual environment, and potential variations in the performance of the proposed HF-based architecture in a real-world context may occur. Furthermore, the proposed architecture can be secure and effective against security attacks due to the functionalities provided by HF (i.e., TLS, MSP). As future work, we plan to develop and evaluate a Collaborative Intrusion Detection System (CIDS) for IoMT-based health monitoring systems, based on the developed HF-based security architecture, that will be responsible for collecting behavioral information (i.e., log files)

from the IoMT devices, information from the IoMT network traffic, and for detecting malicious incidents in such systems.

Author Contributions: Conceptualization, F.P.-O., J.C.R., G.M., G.S. and J.G.; methodology, F.P.-O., J.C.R., G.M., G.S. and J.G.; investigation, F.P.-O., J.C.R., G.M., G.S. and J.G.; resources, F.P.-O., J.C.R., G.M., G.S. and J.G.; writing—original draft preparation, F.P.-O., J.C.R., G.M. and G.S.; writing—review and editing, F.P.-O., J.C.R., G.M., G.S. and J.G.; visualization, F.P.-O., J.C.R. and G.M.; supervision, G.M., G.S. and J.G.; funding acquisition, J.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research work has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876487. The JU receives support from the European Union's Horizon 2020 research and innovation program and Finland, Spain, Italy, Germany, Czech Republic, Belgium, and the Netherlands.

Data Availability Statement: Not applicable.

Acknowledgments: Author José Ribeiro would like to acknowledge the REACT project, which has received funding from European Union's Horizon Europe research and innovation program under the Marie Sklodowska-Curie grant agreement No 101069053 (REACT).

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

References

1. Zachos, G.; Essop, I.; Mantas, G.; Porfyrakis, K.; Ribeiro, J.C. An Anomaly-Based Intrusion Detection System for Internet of Medical Things Networks. *Electronics* **2021**, *10*, 2562. [CrossRef]
2. Karavatselou, E.; Fengou, M.A.; Mantas, G.; Lymberopoulos, D. Profile management system in ubiquitous healthcare cloud computing environment. In *Broadband Communications, Networks, and Systems, Proceedings of 9th International EAI Conference, Broadnets 2018, Faro, Portugal, 19–20 September 2018*; Springer: Berlin/Heidelberg, Germany, 2019; Volume 263, pp. 105–114. [CrossRef]
3. Papaioannou, M.; Karageorgou, M.; Mantas, G.; Sucasas, V.; Essop, I.; Rodriguez, J.; Lymberopoulos, D. A Survey on Security Threats and Countermeasures in Internet of Medical Things (IoMT). *Trans. Emerg. Telecommun. Technol.* **2020**, *33*, e4049. [CrossRef]
4. Del-Valle-Soto, C.; Valdivia, L.J.; López-Pimentel, J.C.; Visconti, P. Comparison of Collaborative and Cooperative Schemes in Sensor Networks for Non-Invasive Monitoring of People at Home. *Int. J. Environ. Res. Public Health* **2023**, *20*, 5268. [CrossRef] [PubMed]
5. Essop, I.; Ribeiro, J.C.; Papaioannou, M.; Rodriguez, J.; Zachos, G.; Mantas, G. Generating datasets for anomaly-based intrusion detection systems in iot and industrial iot networks. *Sensors* **2021**, *21*, 1528. [CrossRef] [PubMed]
6. Gope, P.; Hwang, T. BSN-Care: A Secure IoT-Based Modern Healthcare System Using Body Sensor Network. *IEEE Sensors J.* **2016**, *16*, 1368–1376. [CrossRef]
7. Seliem, M.; Elgazzar, K. BIoMT: Blockchain for the internet of medical things. In Proceedings of the 2019 IEEE International Black Sea Conference on Communications and Networking, BlackSeaCom 2019, Sochi, Russia, 3–6 June 2019.
8. Alsubaei, F.; Abuhussein, A.; Shiva, S. Security and Privacy in the Internet of Medical Things: Taxonomy and Risk Assessment. In Proceedings of the 2017 IEEE 42nd Conference on Local Computer Networks (LCN), Singapore, Singapore, 9–12 October 2017; pp. 112–120. [CrossRef]
9. Sicari, S.; Rizzardi, A.; Grieco, L.A.; Coen-Porisini, A. Security, privacy and trust in Internet of things: The road ahead. *Comput. Netw.* **2015**, *76*, 146–164. [CrossRef]
10. Khezr, S.; Moniruzzaman, M.; Yassine, A.; Benlamri, R. Blockchain technology in healthcare: A comprehensive review and directions for future research. *Appl. Sci.* **2019**, *9*, 1736. [CrossRef]
11. Zhang, M.; Raghunathan, A.; Jha, N.K. Trustworthiness of medical devices and body area networks. *Proc. IEEE* **2014**, *102*, 1174–1188. [CrossRef]
12. Khan, M.A.; Salah, K. IoT security: Review, blockchain solutions, and open challenges. *Future Gener. Comput. Syst.* **2018**, *82*, 395–411. [CrossRef]
13. Wang, X.; Zha, X.; Ni, W.; Liu, R.P.; Guo, Y.J.; Niu, X.; Zheng, K. Survey on blockchain for Internet of Things. *Comput. Commun.* **2019**, *136*, 10–29. [CrossRef]
14. Cui, Z.; Xue, F.; Zhang, S.; Cai, X.; Cao, Y.; Zhang, W.; Chen, J. A Hybrid BlockChain-Based Identity Authentication Scheme for Multi-WSN. *IEEE Trans. Serv. Comput.* **2020**, *13*, 241–251. [CrossRef]
15. Pelekoudas Oikonomou, F.; Ribeiro, J.; Mantas, G.; Bastos, J.; Rodriguez, J. A Hyperledger Fabric-based Blockchain Architecture to Secure IoT-based Health Monitoring Systems. In Proceedings of the 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), Athens, Greece, 5–8 September 2021.
16. Dorri, A.; Kanhere, S.S.; Jurdak, R. Blockchain in Internet of Things: Challenges and Solutions. *arXiv* **2016**, arXiv:1608.05187.

17. Lao, L.; Li, Z.; Hou, S.; Xiao, B.; Guo, S.; Yang, Y. A survey of IoT applications in blockchain systems: Architecture, consensus, and traffic modeling. *ACM Comput. Surv.* **2020**, *53*, 1–32. [CrossRef]
18. Liyanage, M.; Braeken, A.; Kumar, P.; Ylianttila, M. *IoT Security: Advances in Authentication*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2019; ISBN 1119527929.
19. Pelekoudas-oikonomou, F.; Zachos, G.; Papaioannou, M.; De Ree, M.; Ribeiro, J.C.; Mantas, G.; Rodriguez, J. Blockchain-Based Security Mechanisms for IoMT Edge Networks in IoMT-Based Healthcare Monitoring Systems. *Sensors* **2022**, *22*, 2449. [CrossRef]
20. Li, D.; Peng, W.; Deng, W.; Gai, F. A blockchain-based authentication and security mechanism for IoT. In Proceedings of the The 27th International Conference on Computer Communication and Networks (ICCCN 2018), Hangzhou, China, 30 July–2 August 2018. [CrossRef]
21. Babu, E.S.; Dadi, A.K.; Singh, K.K.; Nayak, S.R.; Bhoi, A.K.; Singh, A. A distributed identity-based authentication scheme for internet of things devices using permissioned blockchain system. *Expert Syst.* **2022**, *39*, e12941. [CrossRef]
22. Kakei, S.; Shiraishi, Y.; Mohri, M.; Nakamura, T.; Hashimoto, M.; Saito, S. Cross-Certification towards Distributed Authentication Infrastructure: A Case of Hyperledger Fabric. *IEEE Access* **2020**, *8*, 135742–135757. [CrossRef]
23. Siris, V.A.; Dimopoulos, D.; Fotiou, N.; Voulgaris, S.; Polyzos, G.C. Decentralized authorization in constrained IoT environments exploiting interledger mechanisms. *Comput. Commun.* **2020**, *152*, 243–251. [CrossRef]
24. Pajooh, H.H.; Rashid, M.; Alam, F.; Demidenko, S. Hyperledger fabric blockchain for securing the edge internet of things. *Sensors* **2021**, *21*, 359. [CrossRef]
25. Iftekhar, A.; Cui, X.; Tao, Q.; Zheng, C. Hyperledger fabric access control system for internet of things layer in blockchain-based applications. *Entropy* **2021**, *23*, 1054. [CrossRef]
26. Liu, H.; Han, D.; Li, D. Fabric-iot: A Blockchain-Based Access Control System in IoT. *IEEE Access* **2020**, *8*, 18207–18218. [CrossRef]
27. Shih, D.H.; Wu, T.W.; Shih, M.H.; Chen, G.W.; Yen, D.C. Hyperledger Fabric Access Control for Industrial Internet of Things. *Appl. Sci.* **2022**, *12*, 3125. [CrossRef]
28. Shammar, E.A.; Zahary, A.T.; Al-Shargabi, A.A. An Attribute-Based Access Control Model for Internet of Things Using Hyperledger Fabric Blockchain. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 6926408. [CrossRef]
29. Maeng, J.; Heo, Y.; Joe, I. Hyperledger Fabric-Based Lightweight Group Management (H-LGM) for IoT Devices. *IEEE Access* **2022**, *10*, 56401–56409. [CrossRef]
30. Yanhui, L.; Jianbiao, Z.; Salman Pathan, M.; Yijian, Y.; Puzhe, Z.; Maroc, S.; Nag, A. Research on identity authentication system of Internet of Things based on blockchain technology. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 10365–10377. [CrossRef]
31. Lu, N.; Zhang, Y.; Shi, W.; Kumari, S.; Choo, K.K.R. A secure and scalable data integrity auditing scheme based on hyperledger fabric. *Comput. Secur.* **2020**, *92*, 101741. [CrossRef]
32. Chen, C.; Yang, J.; Tsaur, W.J.; Weng, W.; Wu, C.; Wei, X. Enterprise Data Sharing with Privacy-Preserved Based on Hyperledger Fabric Blockchain in IIOT's Application. *Sensors* **2022**, *22*, 1146. [CrossRef]
33. Androulaki, E.; Barger, A.; Bortnikov, V.; Muralidharan, S.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Murthy, C.; Ferris, C.; et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018. [CrossRef]
34. Projects—Linux Foundation. Available online: https://www.linuxfoundation.org/projects/ (accessed on 8 June 2022).
35. Hyperledger Fabric—Hyperledger Foundation. Available online: https://www.hyperledger.org/use/fabric (accessed on 24 January 2022).
36. Introduction—Hyperledger-Fabricdocs Main Documentation. Available online: https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html (accessed on 8 June 2022).
37. Oikonomou, F.P.; Mantas, G.; Cox, P.; Bashashi, F.; Gil-Castineira, F.; Gonzalez, J. A Blockchain-based Architecture for Secure IoT-based Health Monitoring Systems. In Proceedings of the 2021 IEEE 26th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Porto, Portugal, 25–27 October 2021; pp. 1–6. [CrossRef]
38. Shalaby, S.; Abdellatif, A.A.; Al-Ali, A.; Mohamed, A.; Erbad, A.; Guizani, M. Performance Evaluation of Hyperledger Fabric. In Proceedings of the 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), Doha, Qatar, 2–5 February 2020; pp. 608–613. [CrossRef]
39. X.509: Information Technology—Open Systems Interconnection—The Directory: Public-Key and Attribute Certificate Frameworks. Available online: https://www.itu.int/rec/T-REC-X.509-201910-I/en (accessed on 8 June 2022).
40. *ITU-T Y.2060*; ITU ITU-T Recommendation Database. Telecommunication Standardization Sector: Geneva, Switzerland, 2012.
41. Membership Service Providers (MSP)—Hyperledger-Fabricdocs Main Documentation. Available online: https://hyperledger-fabric.readthedocs.io/en/release-2.2/msp.html (accessed on 22 June 2022).
42. GitHub-Hyperledger/Caliper: A Blockchain Benchmark Framework to Measure Performance of Multiple Blockchain Solutions. Available online: https://github.com/hyperledger/caliper (accessed on 2 April 2023).
43. Hyperledger Caliper | Caliper Is a Blockchain Performance Benchmark Framework, Which Allows Users to Test Different Blockchain Solutions with Predefined Use Cases, and Get a Set of Performance Test Results. Available online: https://hyperledger.github.io/caliper/ (accessed on 14 July 2022).
44. Choi, W.; Hong, J.W.K. Performance Evaluation of Ethereum Private and Testnet Networks Using Hyperledger Caliper. In Proceedings of the 22nd Asia-Pacific Network Operations and Management Symposium, APNOMS, Tainan, Taiwan, 8–10 September 2021; pp. 325–329. [CrossRef]

45. Mondragón-Ruiz, G.; Tenorio-Trigoso, A.; Castillo-Cara, M.; Caminero, B.; Carrión, C. An experimental study of fog and cloud computing in CEP-based Real-Time IoT applications. *J. Cloud Comput.* **2021**, *10*, 32. [CrossRef]
46. Haghi Kashani, M.; Madanipour, M.; Nikravan, M.; Asghari, P.; Mahdipour, E. A systematic review of IoT in healthcare: Applications, techniques, and trends. *J. Netw. Comput. Appl.* **2021**, *192*, 103164. [CrossRef]
47. Raspberry Pi 4 Model B Specifications—Raspberry Pi. Available online: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ (accessed on 28 July 2023).