fabric_objectives fabric_architecture execution_phase ordering_phase validation_phase trust_fault_model

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University of Abderrahmane Mira – Bejaia

Faculty of Exact Sciences
Department of Computer Science
Networks and Security

MASTER THESIS

A Secure Hyperledger Fabric Network for IoT-Based eHealth Monitoring and Data Management

Submitted by
Ziad HEMMAR
Adem Fakhr Allah LOUAAR

Supervised by
Pr. Louiza BOUALLOUCHE MEDJKOUNE
Mrs. Nadjette REBOUH

Examination Committee
Mrs. Hania GADOUCHE — President
Mrs. Feriel CHERIFI — Examiner
Mrs. Souhila GHANEM — Examiner
Mrs. Zahia AZIZOU — Examiner

Academic Year: 2024–2025

Acknowledgments

I would like to express my deepest gratitude to all those who, in one way or another, have supported me throughout the completion of this thesis.

My sincere thanks go to my supervisors, **Pr. Louiza Bouallouche Medjkoune** and **Dr. Nadjette Rebouh**, whose guidance, encouragement, and unwavering support have been instrumental in shaping this work. Their expertise and constructive feedback have greatly enriched the quality of this research.

I am also grateful to the members of the jury for accepting to evaluate this thesis: **Mrs. Hania Gadouche** (President), **Mrs. Feriel CHERIFI**, **Mrs. Souhila Ghanem**, and **Mrs. Zahia Azizou** (Examiners). Their time, insights, and valuable suggestions are deeply appreciated and will undoubtedly contribute to my continued academic development.

I would like to extend my appreciation to the entire teaching staff for their commitment and for the knowledge they have imparted during my studies.

Finally, I warmly thank my friends and fellow students for their encouragement, support, and companionship throughout this academic journey.

Dedication 1

This thesis is dedicated with all my love and deepest gratitude to my family.

To my parents, whose sacrifices, guidance, and unconditional support have shaped the person I am today. Your faith in me has been my greatest source of strength.

To my siblings, for their constant encouragement, patience, and belief in my abilities, even in moments of doubt.

Your love, values, and resilience have been the foundation upon which this journey was built. This accomplishment is as much yours as it is mine.

Dedication 2

To my family — for your quiet love, your unseen sacrifices, and your unwavering belief in me, even when I couldn't believe in myself. Your strength carried me further than words can say.

To those who smiled at my progress, even when it was slow — your faith meant more than you know.

To those who held me up, reminded me to keep going, and never let me fall too far — thank you for your light.

I carry your impact in every word written here, and in every step forward.

Abstract

Smart healthcare now relies on connected medical devices (IoMT) that generate sensitive patient data. Protecting this data—while keeping it accessible to legitimate actors—demands privacy, fine- grained access control, and tamper-proof audit trails. This thesis presents a decentralized healthcare platform built on Hyperledger Fabric. Encrypted records are stored off-chain in Firebase Realtime Database, while on-chain metadata guarantees integrity and traceability. Role- and Attribute-Based Access Control combine to give patients sovereign control over their information. The prototype includes custom chaincode, a Node.js/Express backend, and a React frontend. Under a workload of 100 transactions per second the network sustained sub-200 ms end-to-end latency and zero transaction loss, confirming viability for real-time clinical use.
Keywords: Blockchain, Hyperledger Fabric, IoMT, Access Control, eHealth, Firebase.

Résumé

Les systèmes de santé intelligents reposent de plus en plus sur des dispositifs médicaux connectés (IoMT) générant des données sensibles. Assurer la confidentialité, un contrôle d'accès précis et une traçabilité fiable dans ce contexte représente un défi technologique majeur. Ce mémoire présente une plateforme décentralisée de gestion des données médicales, basée sur Hyperledger Fabric. Les enregistrements sont stockés de manière chiffrée hors chaîne via Firebase Realtime Database, tandis que les métadonnées sont inscrites sur la blockchain afin de garantir l'intégrité et la transparence des échanges. Un modèle combiné de contrôle d'accès basé sur les rôles (RBAC) et les attributs (ABAC) permet aux patients de conserver un contrôle total sur leurs données. Le prototype intègre des contrats intelligents, un backend sécurisé en Node.js/Express, et une interface utilisateur en React. Des tests sous charge (100 transactions par seconde) ont montré une latence inférieure à 200 ms et une stabilité complète, confirmant l'applicabilité du système en contexte réel.
Mots-clés : Blockchain, Hyperledger Fabric, IoMT, Contrôle d'accès, e-Santé, Firebase.

Contents

List of Figures

List of Tables

List of Abbreviations

ABAC	Attribute-Based Access Control
AES	Advanced Encryption Standard
API	Application Programming Interface
BFT	Byzantine Fault Tolerance
CA	Certificate Authority
CAPEX/OPEX	Capital Expenditure / Operational Expenditure
CID	Content Identifier
CFT	Crash Fault Tolerance
DCS	Decentralization, Consistency, and Scalability
DoS	Denial of Service
DPoS	Delegated Proof of Stake
ECG	Electrocardiogram
EHR	Electronic Health Record
GDPR	General Data Protection Regulation
HIT	Health Information Technology
HIoT	Healthcare Internet of Things
HIPAA	Health Insurance Portability and Accountability Act
HLF	Hyperledger Fabric
ID	Identifier
IoMT	Internet of Medical Things
IoT	Internet of Things
IPFS	InterPlanetary File System
MSP	Membership Service Provider
NGO	Non-Governmental Organization

OSN	Ordering Service Node
P2P	Peer-to-Peer
PBFT	Practical Byzantine Fault Tolerance
PoA	Proof of Authority
PoS	Proof of Stake
PoW	Proof of Work
PTM	Peer Transaction Manager
RBAC	Role-Based Access Control
RFID	Radio-Frequency Identification
RSK	RootStock
SC	Smart Contract
SDK	Software Development Kit
SHA-256	Secure Hash Algorithm 256-bit
TLS	Transport Layer Security
VM	Virtual Machine
VSCC	Validation System Chaincode
WSN	Wireless Sensor Network
X.509	X.509 Digital Certificates

List of Algorithms

General Introduction

Health is the foundation of personal well-being and social stability. It affects every aspect of human life—physical, mental, and economic. Maintaining good health requires not only access to quality care but also accurate information and timely intervention. In modern societies, healthcare is becoming increasingly dependent on technology to meet these needs efficiently and at scale.

Context and Motivation

Over the past decade, the healthcare sector has undergone a rapid digital transformation, driven by the proliferation of smart devices, electronic health records, and real-time monitoring technologies. As medical institutions increasingly rely on digital infrastructures, they collect, transmit, and store vast volumes of sensitive patient data. While this evolution has significantly enhanced diagnosis, treatment, and patient monitoring, it has also raised major challenges related to data security, access control, and interoperability.

Limitations of Traditional Architectures

Traditional healthcare systems, typically built on centralized architectures, are struggling to meet growing demands for trust, privacy, and accountability—especially in contexts involving wearable IoT medical devices and remote health monitoring. In such systems, data is generated continuously and must be handled with high security, precision, and flexibility.

Proposed Solution and Contributions

To address these limitations, this thesis explores the use of blockchain technology—specifically Hyperledger Fabric, a permissioned blockchain framework—as a secure and scalable foundation for healthcare data management. We propose a modular system architecture that combines Hyperledger Fabric with Firebase Realtime Database. In this design, encrypted medical records are stored off-chain, while metadata and access logs are recorded on-chain to ensure traceability, auditability, and integrity.

The proposed solution empowers patients with fine-grained control over who can access their data and under what conditions, enabling a shift toward patient-centric data sovereignty. The system also integrates a lightweight backend and modern frontend to handle user authentication, access control, and real-time performance metrics—ensuring both usability and scalability.

Overall, this work contributes a secure, extensible, and patient-centered framework tailored for healthcare environments where privacy, accountability, and interoperability are critical.

Thesis Structure

The thesis is organized as follows:

- **Chapter 1** introduces the core technologies, including the structure of healthcare systems, the role of IoT in medicine, and blockchain fundamentals, with a focus on Hyperledger Fabric.
- **Chapter 2** reviews and analyzes state-of-the-art blockchain-based healthcare solutions, identifying their limitations and presenting the proposed approach.
- **Chapter 3** details the system design and implementation, followed by performance evaluation results.
- Finally, the thesis concludes with a general conclusion and perspectives for future research and development.

Chapter 1

Definitions and Generalities

1.1 Introduction

Modern healthcare faces major challenges including chronic diseases, aging populations, and inequality in care access. Digital technologies like IoT, EHRs, and telemedicine are key to improving system performance and patient outcomes. These tools enable real-time data collection and smarter medical decisions. Yet, they raise serious concerns around security, interoperability, and trust. Blockchain, especially permissioned platforms like Hyperledger Fabric, offers solutions through decentralization and data integrity. This chapter sets the stage by examining healthcare structures and how digital and blockchain technologies reshape their future.

1.2 Healthcare System Structure and Challenges

This section examines the structure, key stakeholders, and institutional challenges faced by modern healthcare systems.

1.2.1 Definition and Structural Challenges

Healthcare systems, as defined by the World Health Organization, encompass "all activities whose primary responsibility is to promote, restore, and maintain health" [1]. This definition underscores the comprehensive and multifaceted nature of modern healthcare systems.

Structurally, healthcare systems typically consist of three main levels of service provision: primary care (basic health services and prevention), secondary care (specialist and hospital services), and tertiary care (advanced and complex medical interventions). These systems are supported by a variety of financing models, such as public funding, private insurance, and out-of-pocket payments, and are governed through administrative, legal, and policy frameworks established by health authorities and regulatory bodies (see figure 1.1).

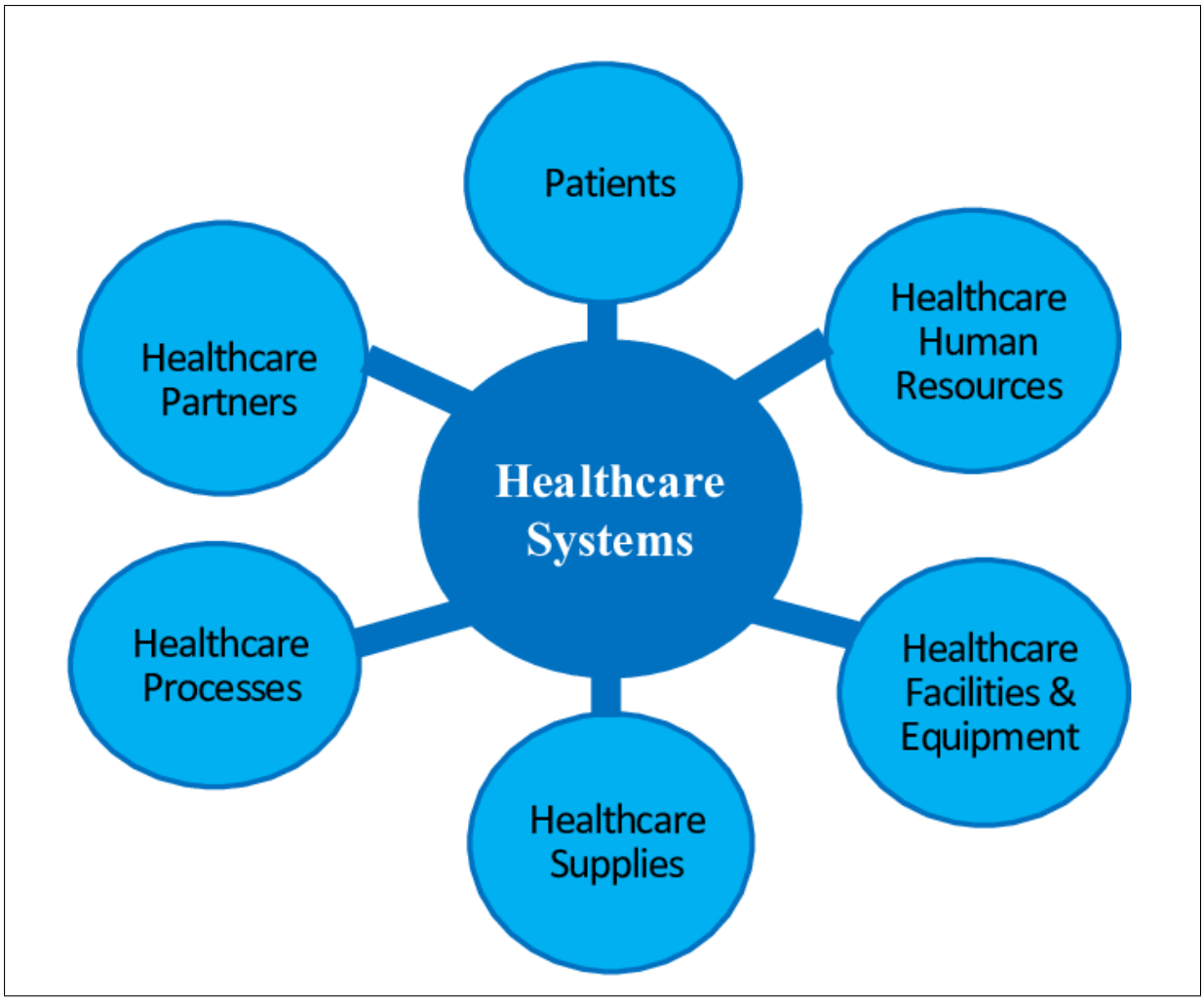

Figure 1.1: Structure of a Healthcare System

Operationally, they involve a wide network of stakeholders — including national governments, multilateral organizations, non-governmental organizations (NGOs), and private sector actors — that coordinate to deliver health services and shape policy directions [1].

This institutional complexity often results in overlapping responsibilities and fragmented service delivery, which can hinder efficiency. However, it also provides flexibility for different actors to complement one another and fill systemic gaps. As demographic trends, economic pressures, and epidemiological patterns evolve, healthcare systems worldwide are undergoing continuous reforms aimed at enhancing institutional coordination, equity, and health outcomes [1].

1.2.2 Digitalization of Healthcare

The digital transformation of healthcare has accelerated in response to global challenges like COVID-19, highlighting the need for accessible, remote, and cost-effective care, particularly for the elderly and chronically ill [2]. Mobile health (mHealth), telemedicine, and electronic health records (EHRs) have emerged as key technologies to support continuous,

remote patient monitoring and reduce the burden on physical healthcare infrastructures [2].

The Internet of Things **(IoT)** plays a crucial role by connecting wearable and implantable devices that collect real-time physiological data such as glucose levels, heart rate, temperature, etc. These tools enable early disease detection, chronic condition management, and smart home healthcare services [2, 3]. IoT integration into healthcare supports patient-centered approaches and allows quick interventions, remote monitoring, and improved outcomes through predictive analytics [3].

Electronic Health Records **(EHRs)** and Health Information Technology (HIT) have streamlined healthcare management by digitizing patient data and facilitating its secure exchange. These systems improve coordination among healthcare professionals, enhance clinical workflows, and support early detection and preventive care, especially for non-communicable diseases [4]. They, also, proved vital during the pandemic for resource management and disease tracking [4].

Despite their benefits, digital healthcare systems face significant challenges, including data privacy concerns, system interoperability issues, and vulnerability to cyber threats like ransomware and identity theft [3]. Ensuring secure transmission and storage of sensitive data remains a top priority, prompting the development of encryption protocols and cybersecurity strategies [2, 3].

1.3 Internet of Things in Healthcare

The Internet of Things (IoT) refers to a sophisticated network of uniquely identifiable and addressable devices equipped with sensors and communication technologies that collect, process, and exchange data over the Internet to deliver context-aware services [2]. This paradigm has evolved through the convergence of technologies such as real-time analytics, machine learning, embedded systems, and commodity sensors [5, 2]. A typical IoT infrastructure includes communication interfaces, RFID, wireless sensor networks (WSNs), cloud and fog computing, and advanced data processing algorithms [2]. In the healthcare domain—referred to as the Internet of Medical Things (IoMT)—IoT connects medical devices and healthcare systems to enable real-time monitoring, data-driven diagnostics, and remote patient management [6]. While IoMT offers transformative potential, it also raises critical concerns about data security and patient privacy due to the sensitivity of health information being transmitted across networks [6].

1.3.1 Applications of IoT in Healthcare

The Internet of Medical Things (IoMT) applies IoT technologies in healthcare by connecting medical devices that wirelessly monitor and manage patient data. This integration enables key advancements such as remote monitoring, digitized health records, real-time material tracking, and telemedicine [5]. Telemedicine, in particular, allows healthcare providers to deliver care across distances, benefiting patients with limited mobility [5, 2]. IoT-based systems can continuously track vital signs, detect anomalies early, and improve chronic disease management, leading to reduced hospitalization rates [2]. Additionally, IoT supports smart healthcare environments through automated data collection and real-time tracking, enhancing both patient care and healthcare facility efficiency [5].

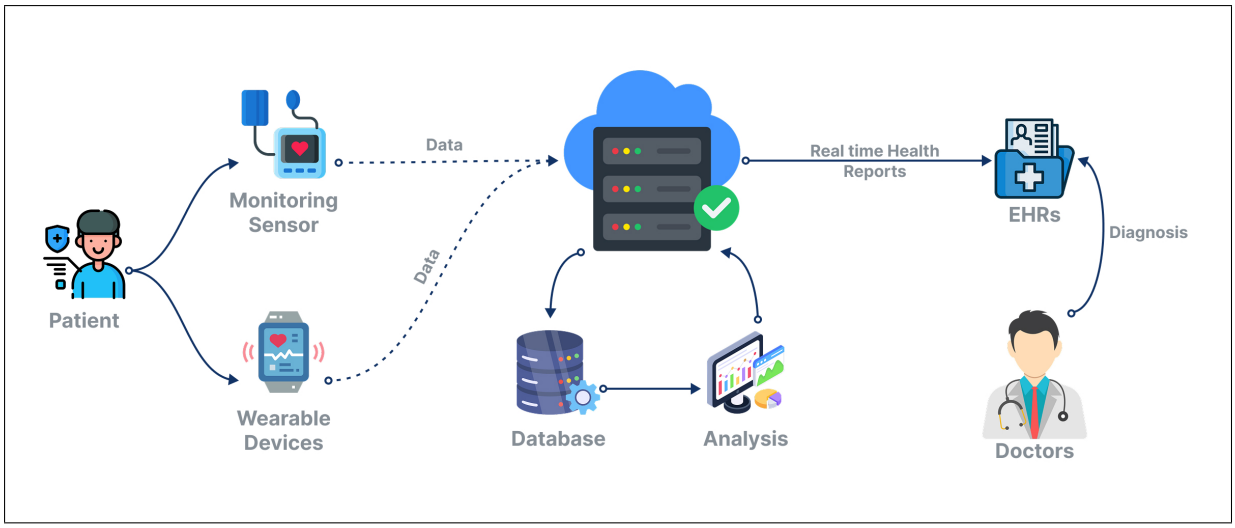

Figure 1.2: IoT-Based Healthcare Ecosystem

1.3.2 IoT Architecture in Healthcare

The typical architecture of Healthcare IoT (HIoT) systems is composed of four layers: The perception layer, the network layer, the middleware layer, and the application/business layer [2] (see figure 1.3).

- **Perception Layer:** This is the hardware layer that includes sensors and medical devices responsible for collecting physiological data from patients. These devices detect signals such as temperature, blood pressure, glucose levels, etc.
- **Network Layer:** This layer ensures secure data transmission between the devices and the healthcare systems using communication technologies like Wi-Fi, Bluetooth, Zigbee, 5G, etc.
- **Middleware Layer:** It serves as an intermediary between the hardware and application layers. It handles data storage, service requests, and device heterogeneity, allowing interoperability across various platforms.
- **Application/Business Layer:** This layer delivers healthcare services based on data analytics and processing. It generates alerts, diagnostic reports, and integrates data to support clinical decisions and patient management.

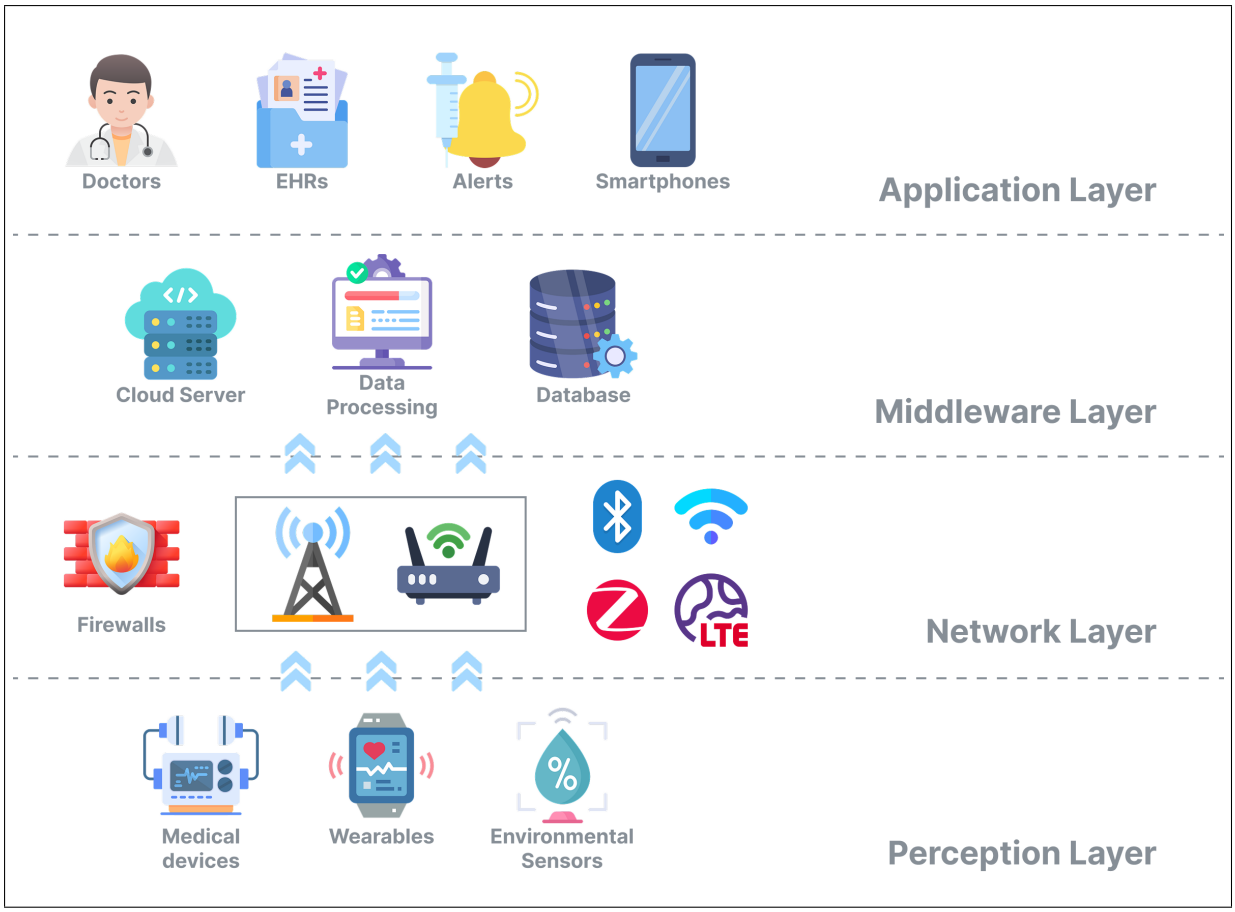

Figure 1.3: IoMT Layered Architecture

The workflow of a typical HIoT system follows three main steps: Data generation (through sensors and devices), data processing (via machine learning and analytics), and information consumption (for clinical decisions or automated actuation) [2]. For example, wearable devices can transmit patient vitals to a smartphone, which then forwards the data to a cloud server where it is analyzed. If anomalies are detected, alerts are sent to physicians or caregivers for immediate action.

1.3.3 Key Challenges and Limitations of IoT / HIT in Healthcare

While Healthcare IoT (HIoT) and Health Information Technology (HIT) promise breakthroughs in patient monitoring, remote diagnostics, and care automation, several structural, operational, and security limitations prevent scalable, resilient deployment. These challenges—documented across recent studies [4, 3, 7]—span interoperability, security, real-time reliability, and fault tolerance.

- **Interoperability & Standardization**. HIT and HIoT systems still lack unified data formats and protocols. Fragmented vendor ecosystems block seamless data exchange and system integration across hospitals and care platforms [4].
- **Security & Data Integrity**. Patient data remains vulnerable to breaches, tampering, and spoofing across sensor layers, communication protocols, and cloud in-

frastructures. Attacks on unencrypted or weakly authenticated channels can lead to compromised diagnoses and privacy violations [7]. The absence of system-wide zero-trust architectures and immutable audit trails remains a serious gap.

- **Fault Tolerance & System Resilience**. HIoT systems often lack the redundancy and failover mechanisms required for dependable operation. Node failures, packet loss, sensor drift, or cloud service interruptions can jeopardize critical services. Real-time health data pipelines must tolerate faults without degrading care continuity or triggering false alarms.

- **Scalability**. HIoT solutions that function in isolated testbeds often break down when deployed at national scale. Network congestion, device heterogeneity, and backend overloads challenge the feasibility of city-wide or nation-wide deployments [3, 7].

- **Real-Time Communication**. Clinical use cases—like arrhythmia detection or insulin dosing—demand ultra-low latency and deterministic behavior. Network delays, jitter, or packet reordering can render systems unsafe or medically useless [7].

- **Energy Constraints**. Wearables and implantable devices face hard energy limits. Continuous data collection, local computation, and wireless transmission rapidly drain power, limiting deployment in remote or resource-limited environments [7].

- **Massive Data Management**. The exponential growth of structured and unstructured health data (ECG streams, video feeds, clinical logs) stresses current storage, indexing, and retrieval systems. Data must remain accessible, consistent, and compliant with regulatory constraints such as GDPR and HIPAA [7].

- **Financial Barriers**. High CAPEX/OPEX for sensors, networks, secured cloud platforms, and maintenance slows down deployment in public or under-resourced health systems [4].

- **Workforce Digital Readiness**. Medical staff often lack training or willingness to adopt digital workflows. Technological friction, steep learning curves, and fear of system errors hinder integration into clinical routines [4].

- **Mobility & Continuous Access**. True remote care requires stable, cross-platform access to real-time health data. Network dropouts, lack of roaming protocols, or incompatible systems disrupt monitoring continuity and patient autonomy [3].

- **Real-World Testing & Deployment**. Many HIoT prototypes remain untested beyond labs. Simulation results rarely translate into real-world environments that involve noise, delays, user unpredictability, and adversarial scenarios [3].

Outlook. Security and fault-tolerance are no longer optional—they're foundational. Techniques like blockchain (for tamper-proof records), federated learning (for privacy-aware data use), and edge computing (for low-latency fault isolation) offer concrete mitigation paths. But without interoperable standards, robust cybersecurity, and system-level fault design, HIoT will remain a fragmented promise.

1.4 Blockchain Technology Overview

This section introduces the core principles of blockchain technology and sets the stage for its application in healthcare and IoT environments.

1.4.1 Fundamentals of Blockchain Technology

A blockchain is an *append-only*, *time-stamped* ledger in which each block contains: (i) a batched set of transactions, (ii) the cryptographic hash of the previous block, and (iii) a proof generated by a consensus algorithm [8], [9]. Because altering any historical block forces the attacker to recompute the chain's cumulative proof, the design offers *immutability*, *tamper-evidence*, and *ordered finality* [10]. Nodes maintain a local copy of the ledger, validate transactions, and broadcast new information peer-to-peer.

1.4.2 Distributed Ledger

At the core of any blockchain system lies a distributed ledger—a synchronized and append-only database replicated across multiple nodes in a network (see figure 1.4). Unlike traditional centralized databases, there is no single point of control or failure. All participants hold a copy of the ledger and update it via consensus protocols, ensuring consistency and fault tolerance even in adversarial conditions. The ledger records all validated transactions chronologically, enabling traceability, auditability, and tamper-resistance [8].

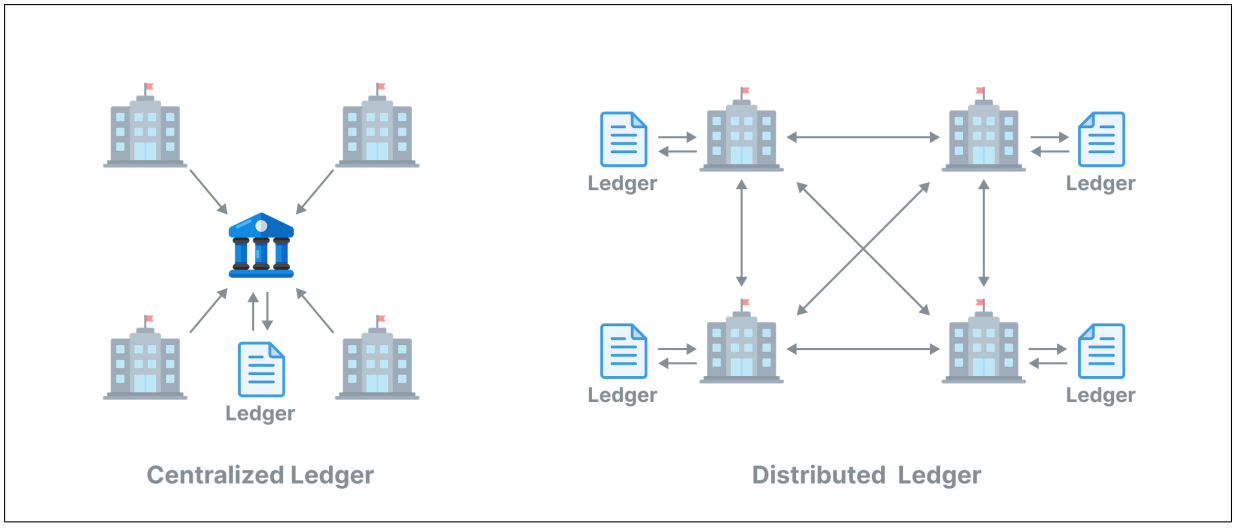

Figure 1.4: Distributed ledger technology.

1.4.3 Blockchain Network Components

A functional blockchain system relies on a coordinated set of components that handle data exchange, validation, storage, and execution—each playing a critical role in maintaining security, consistency, and decentralization [9] (see figure 1.5).

- **Nodes.** In a blockchain network, a node refers to an individual system that participates in the network. Nodes can serve different roles. A full node stores the

entire blockchain and is responsible for validating transactions. A publishing node is a type of full node that, in addition to validating transactions, also creates and publishes new blocks to the blockchain. In contrast, a lightweight node does not store or maintain a full copy of the blockchain and relies on full nodes to process and validate its transactions [8].

- **Transactions.** In a blockchain network, a transaction represents an interaction between parties, such as the transfer of cryptocurrency between users or the recording of activities involving digital or physical assets in business contexts. Each block in a blockchain can contain zero or more transactions, and in some implementations, publishing new blocks regularly—even without transactions—is essential for network security [8]. A typical cryptocurrency transaction includes inputs (referencing the source of digital assets and proving ownership through digital signatures) and outputs (specifying recipients, transferred amounts, and conditions for future use). Beyond transferring digital assets, transactions can also be used to record data or interact with smart contracts—for example, updating the status of a shipment in a blockchain-based supply chain system. Ensuring both the validity and authenticity of transactions is crucial, typically achieved through digital signatures verified using public-key cryptography[8]

- **Blocks.** In blockchain architecture, each block acts as a container for validated transactions and consists of two main components: the block header and the block body [8]. The header includes metadata such as the block height, timestamp, block size, nonce (in mining-based systems), and the hash of the previous block's header, which links blocks together to form a chain [9]. It also contains the hash of the current block's data, typically the Merkle root, ensuring data integrity.

  The block body holds a list of validated transactions, secured through digital signatures using asymmetric cryptography [8][9]. The chain structure is established by each block referencing the previous block's hash—any tampering alters the hash, breaking the chain and making unauthorized changes easily detectable. This structure underpins the immutability and security of the blockchain ledger [8].

- **Smart Contracts / Chaincode.** Smart contracts are self-executing programs that automatically enforce agreements when predefined conditions are met [9]. Introduced by Nick Szabo in the 1990s, they aim to reduce reliance on intermediaries, minimize exceptions, and digitally enforce terms [8].

  On a blockchain, smart contracts consist of code and data deployed via cryptographically signed transactions [8]. Once deployed, they are tamper-resistant and transparent, with all logic and outcomes recorded immutably on the ledger [8, 11]. These contracts must be deterministic—ensuring that identical inputs produce identical outputs across all nodes to maintain consensus [8].

  Smart contracts typically follow four phases [11]:

  1. **Creation** – Stakeholders convert agreements into executable logic.
  2. **Deployment** – The contract is uploaded to a blockchain (e.g., Ethereum, Hyperledger Fabric), becoming immutable.

3. **Execution** – When conditions are met, contract functions trigger automatically, updating the blockchain state [8, 11].
4. **Completion** – Assets are distributed or actions finalized, with all events permanently recorded [11].

Different blockchain platforms implement smart contracts in varying ways. In Ethereum, users pay "gas" to execute contract code, which prevents abuse and ensures resource efficiency. In contrast, permissioned networks like Hyperledger Fabric may avoid execution fees and instead rely on access controls and participant trust models [8].

- **Network Layer.** The communication backbone of a blockchain network, responsible for the peer-to-peer (P2P) exchange of transactions, blocks, consensus messages, and metadata. Typically implemented using a gossip-based dissemination protocol, this layer ensures that all valid data reaches all participating nodes efficiently and redundantly, even in the presence of unreliable links or partial failures. It is designed to resist topology-based attacks such as Sybil (where a single adversary spawns many fake nodes) and Eclipse attacks (where a node's view is isolated and manipulated). Mechanisms like peer reputation scoring, random peer sampling, encrypted channels (TLS), and authentication via digital certificates (e.g., in Hyperledger Fabric) further reinforce the trustworthiness and resilience of this layer. In permissioned networks, the network layer may also enforce access control and segregated communication via channels or subnets [8].

- **State Database.** A high-performance key–value store (such as LevelDB or CouchDB) that maintains the latest snapshot of the blockchain's world state. While the blockchain itself records the full historical log of transactions, the state database offers a current, query-efficient representation of smart contract variables (e.g., account balances, asset ownership, medical record pointers). When a smart contract executes, it reads and writes to this database rather than scanning the entire chain history. The state is updated deterministically during block validation, ensuring consistency across all nodes. In permissioned platforms like Hyperledger Fabric, this database can also support rich JSON-based queries and pluggable privacy-preserving features for regulated data (e.g., GDPR-compliant selective disclosure) [8, 11].

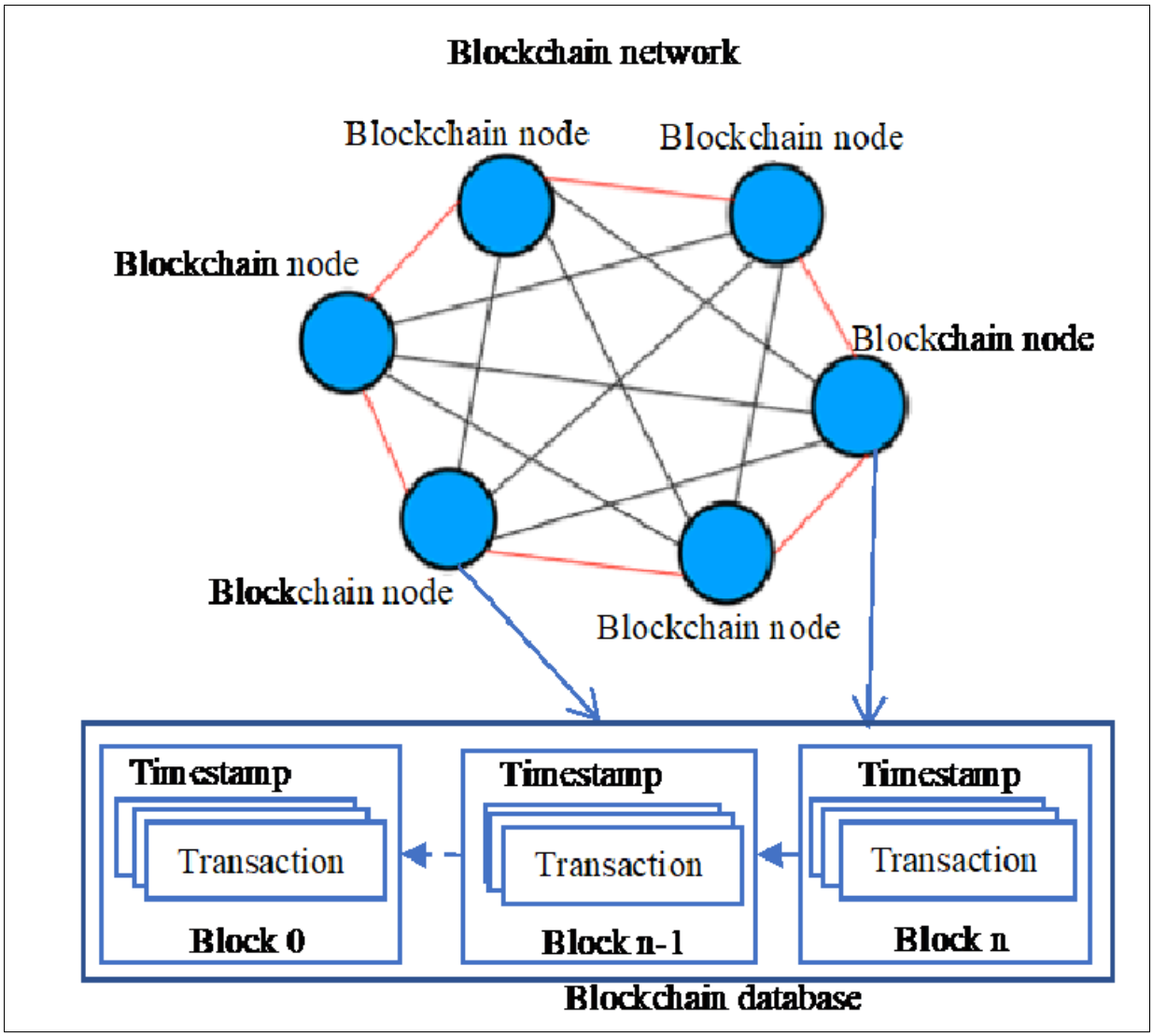

Figure 1.5: Blockchain-architecture.

1.4.4 Cryptographic Foundations

- **Hash Functions** are essential to blockchain, ensuring data security and enabling block linkage. A cryptographic hash function produces a fixed-size digest from input data of any length. Even a minor input change generates a vastly different output, exposing any tampering [8]. These functions are preimage resistant, second preimage resistant, and collision resistant, making it infeasible to reverse inputs or find two inputs with the same hash [8]. In blockchains, each block includes the hash of the previous block's header, forming a secure, chronological chain. Altering any block invalidates all subsequent hashes, preserving ledger integrity [8, 12]. Transactions and timestamps are also hashed with the previous block's hash, enabling independent verification of the chain's integrity by any node [12]. This ensures decentralized trust and immutability.

- **Asymmetric-Key Cryptography** (public key cryptography) underpins security and trust in blockchain systems. It uses a public-private key pair: the public key is

shared openly, while the private key remains secret. Anyone can encrypt data using the public key, but only the private key holder can decrypt it, ensuring confidentiality. Conversely, signing data with a private key allows anyone with the public key to verify the sender's identity and the data's integrity [8]. In blockchain, this enables users to digitally sign transactions—proving ownership and authorization without exposing their private key [8]. Digital wallets, such as those in Bitcoin, use the public key as a visible address, while the private key authorizes transactions securely [13].

1.4.5 Consensus Mechanisms

A consensus mechanism is a foundational component of blockchain systems, enabling distributed nodes to agree on a single version of the ledger without a centralized authority. This mechanism is crucial for maintaining consistency, preventing fraud (such as double spending), and ensuring secure value transfers across untrusted networks [13].

Several types of consensus mechanisms exist, each with different trade-offs in performance, scalability, and trust models. The most widely adopted include Proof of Work (PoW), Proof of Stake (PoS), and Proof of Authority (PoA).

- **Proof-of-Work (PoW).** PoW was the first consensus mechanism introduced by Satoshi Nakamoto in Bitcoin to enable decentralized agreement in a trustless environment [13]. In PoW, nodes (miners) compete to solve a computationally intensive cryptographic puzzle. The first to solve it earns the right to publish the next block and is rewarded with cryptocurrency. The puzzle typically involves finding a nonce such that the hash of the block's contents falls below a target value [8]. For example, Bitcoin adjusts this difficulty approximately every 2016 blocks to maintain an average block time of 10 minutes [8].

  This approach makes it easy to verify valid blocks but expensive to produce them, helping to prevent Sybil attacks by tying block production capability to computational power rather than the number of identities [8]. However, the high resource consumption of PoW systems—particularly in terms of electricity and hardware—has motivated the development of alternative consensus mechanisms [8].

  Ethereum and RSK are examples of platforms that use PoW, requiring significant computational effort to validate blocks [11].

- **Proof-of-Stake (PoS).** PoS introduces an energy-efficient alternative to PoW by tying block validation rights to the amount of cryptocurrency a user has staked in the system [8]. The underlying assumption is that users with more at stake are more incentivized to act honestly to protect their investment. Instead of solving a puzzle, validators are selected to publish blocks based on various schemes: random selection proportional to stake, coin age, or multi-round voting [8].

  Unlike PoW, PoS avoids extensive resource consumption, making it more sustainable for the environment. Some blockchains using PoS eliminate block creation rewards, relying solely on transaction fees as validator incentives [8].

  A common variant is Delegated Proof of Stake (DPoS), used in EOS, where users vote for a fixed number of delegates who are responsible for validating and adding new blocks [11].

- **Proof of Authority (PoA).** , also referred to as Proof of Identity, is primarily used in permissioned blockchain networks. Instead of staking coins or solving puzzles, nodes validate blocks based on their real-world identities and reputations [8]. These identities are verified and registered within the blockchain network, and block production is limited to a small set of trusted entities.

  Publishing nodes maintain their ability to validate blocks by acting in the network's best interest; dishonest behavior can lead to loss of reputation and exclusion from future participation [8]. Because of its centralized trust assumptions, PoA is typically used in private or consortium blockchains where participants are known and vetted.

1.4.6 Types of Blockchain

Blockchain systems can be classified based on access control, governance, and trust assumptions, each suited to different application domains and security requirements.

Permissionless (Public) Blockchains. Permissionless blockchains allow anyone to join the network, validate transactions, and interact with the ledger without prior approval. These open, decentralized systems rely on consensus algorithms like Proof of Work (PoW) and Proof of Stake (PoS), which require participants to expend or lock in resources (e.g., computing power or tokens) to validate blocks. This deters malicious activity by making attacks costly and inefficient [8].

Notable examples include Bitcoin and Ethereum, which emphasize transparency and trustlessness—anyone can verify transactions independently. However, their openness also introduces security challenges, making strong consensus protocols essential [13].

Permissioned (Private and Consortium) Blockchains. Permissioned blockchains limit participation to authorized entities. These can be private, controlled by a single organization, or consortium-based, managed by a group of trusted participants [8][13].

In private blockchains, access and validation are restricted, and consensus is often centralized or semi-centralized. This model sacrifices decentralization but offers faster transactions and stronger privacy controls. Because validators are known and trusted, consensus does not require energy-intensive processes; misbehavior is managed through governance and access revocation [8][13].

Consortium blockchains. They offer a hybrid approach, where a predefined group collectively governs participation and validation. They often use consensus methods like multi-signature schemes or voting among nodes, balancing trust, performance, and security for inter-organizational collaboration [13].

1.4.7 Blockchain Challenges

Despite its potential, blockchain technology faces several technical, operational, and regulatory challenges that limit its adoption in critical sectors like healthcare.

- **Performance and Scalability.** Blockchain networks, especially those using Proof of Work (PoW), struggle with low throughput and high latency (e.g., Bitcoin processes only 6–10 transactions per second), mainly due to computational demands and limited block sizes [14]. While PBFT protocols improve energy efficiency, they scale poorly due to message complexity. According to the DCS theorem, blockchain systems must trade off between decentralization, consistency, and scalability [14].

- **Privacy Concerns.** Though pseudonymous, blockchain is not private. Public access to transaction histories and wallet balances allows potential de-anonymization through analysis of public keys and user behavior, raising concerns for sensitive applications [14].

- **Interoperability Barriers.** Another major issue is lack of interoperability. With over 6,500 blockchain projects on GitHub using different programming languages, consensus protocols, and privacy models, most blockchains operate in isolated ecosystems that cannot interact with one another [14]. This fragmentation inhibits the development of integrated applications and slows the overall progress of blockchain adoption across industries.

- **High Energy Consumption.** The PoW algorithm, widely used in early blockchain systems like Bitcoin, is energy-intensive, consuming more electricity annually than many countries. This creates environmental sustainability concerns, with a single transaction generating up to 274.29 kg of CO_2 [14]. Although alternatives like Proof of Stake (PoS) and sharding offer improved energy efficiency and scalability, they often trade off decentralization or security [15].

- **Security Vulnerabilities.** Early-stage blockchain systems are vulnerable to attacks such as selfish mining and 51% attacks, which threaten network integrity [14]. Additional risks arise from flaws in smart contracts and digital wallets [15].

- **Development Costs and Economic Viability.** While blockchain reduces intermediary costs, high initial development expenses and performance limitations make it less attractive in sectors where traditional databases already perform efficiently unless transparency and immutability are critical [15].

- **Regulatory and Legal Challenges.** The decentralized nature of blockchain undermines traditional centralized regulatory frameworks, making governments wary of its economic implications. In some countries, such as Iran, Morocco, and Pakistan, cryptocurrencies are banned, and regulatory uncertainty remains a significant barrier to adoption [14]. Furthermore, associations with illicit activities such as money laundering have tainted blockchain's public image, prompting stricter oversight that could stifle innovation [14].

1.4.8 Blockchain in healthcare

Blockchain technology presents significant potential for addressing critical challenges in healthcare, particularly in the areas of data management, security, and interoperability. Its decentralized structure and immutability make it ideal for ensuring secure data exchange across disparate healthcare systems without compromising patient privacy [16].

The ability of blockchain to support access control, audit trails, and patient-controlled healthcare records strengthens both the accuracy and reliability of electronic health records (EHRs) [17]. Additionally, blockchain enhances data integrity through cryptographic hashing and immutability, preventing unauthorized alterations to sensitive medical information [18].

In healthcare applications, permissioned blockchain frameworks such as Hyperledger Fabric have been effectively used to implement access control mechanisms through membership services and digital certificates, offering both flexibility and security in managing sensitive medical data [18]. The use of smart contracts allows automation of complex processes such as real-time patient monitoring and clinical trial data tracking, reducing costs and minimizing human intervention [16, 18]. Moreover, blockchain supports healthcare transformation towards a more patient-centric model by enabling secure data sharing and improved provenance of medical activities [17].

Among the available blockchain frameworks, Hyperledger Fabric is particularly well-suited for healthcare due to its modular, permissioned architecture, which aligns with the sector's strict privacy, compliance, and interoperability requirements.

1.5 Hyperledger Fabric

As one of the most mature and widely adopted permissioned blockchain frameworks, **Hyperledger Fabric** stands out due to its modular architecture, strong focus on confidentiality, and suitability for enterprise environments. Developed under the Linux Foundation's Hyperledger project, Fabric provides the foundation for building secure, scalable, and customizable blockchain solutions that address the limitations of earlier platforms.

1.5.1 Overview of Hyperledger Fabric

Hyperledger Fabric definition

Hyperledger Fabric is an open-source blockchain platform designed specifically for permissioned blockchain networks. Unlike public blockchains—where anyone can participate anonymously—Fabric operates in a network of known and identified participants who share a common goal but may not fully trust each other. It provides a secure and flexible infrastructure for building distributed-ledger applications with enhanced confidentiality, scalability, and resilience[19, 20, 21].

Fabric is often described as **a distributed operating system for permissioned blockchains**[22], enabling the execution of distributed applications (smart contracts) written in standard programming languages. This capability distinguishes Fabric from many other blockchain platforms that require domain-specific languages or restrict smart contract execution environments[23].

Key Objectives and Use Cases

Hyperledger Fabric addresses several limitations found in earlier permissioned blockchain platforms that adopt the traditional *order–execute* architecture, where every transaction must be executed sequentially and deterministically by every participant. These limi-

tations include lack of modular consensus, rigid trust models, restricted programming environments, and confidentiality challenges[23].

Fabric introduces a novel **execute–order–validate** transaction-processing paradigm that improves performance, flexibility, and confidentiality by allowing transactions to be executed (endorsed) by a subset of peers before ordering and final validation[23]. This design supports parallel execution, non-deterministic smart contracts, and tailored endorsement policies, enabling real-world applications to enforce business logic and trust assumptions more naturally.

Today, Fabric is widely adopted across various industries, with use cases ranging from trade logistics, food safety, contract management, identity management, to financial services such as securities trading and settlement. More than 400 prototypes and production systems utilise Fabric's modular architecture to meet diverse requirements for secure, scalable, and confidential distributed ledgers[23].

1.5.2 Architecture and Components

Fabric is a distributed operating system for permissioned blockchains, designed to run distributed applications written in general-purpose programming languages (e.g., Go, Java, Node.js). It maintains a secure, append-only, replicated ledger and does not include any built-in cryptocurrency [23].

Fabric introduces an "execute–order–validate" architecture, diverging from the traditional order–execute approach. A Fabric distributed application consists of two main elements:

- **Smart Contract (Chaincode)** — contains the application logic executed during the *execution phase*. Chaincode, which can be developed by untrusted parties, plays a central role in application logic. There are also system chaincodes used to manage the blockchain system and its parameters[23].

- **Endorsement Policy** — applied during the *validation phase*. Defined by trusted administrators, these policies cannot be altered by untrusted developers. A typical policy specifies required endorsers using monotone logical expressions such as (A $\wedge B$) $\vee C$. Custom policies can support arbitrary logic [23].

A Fabric network comprises identified nodes, ensured by a modular Membership Service Provider (MSP). Nodes assume three roles[23]:

- **Clients** — submit transaction proposals, coordinate execution, and broadcast transactions for ordering.

- **Peers** — execute transaction proposals and validate transactions. All peers maintain the ledger (a hash chain of transactions) and the state (the current ledger snapshot). Only endorsing peers execute proposals, as specified by the chaincode policy.

- **Ordering Service Nodes (OSNs)** — form the ordering service responsible for establishing total transaction order. OSNs do not access application state nor participate in execution or validation, making Fabric's consensus mechanism highly modular.

Fabric supports multiple channels, each forming a separate blockchain connected to the same ordering service. Channels have distinct peer memberships and isolated transaction ordering. Some deployments may implement per-channel access control.

1.5.3 Transaction Lifecycle in Hyperledger Fabric

The following section outlines the Transaction Lifecycle in Hyperledger Fabric, detailing the execution, ordering, and validation phases that govern how transactions are processed within the network.

Execution Phase

In the *execution phase*, clients sign and send transaction proposals to the *endorsers* specified by the endorsement policy of the target chaincode. Each proposal contains the client's identity (via MSP), the operation to execute, parameters, chaincode ID, a nonce, and a derived transaction ID[23].

Endorsers simulate the proposal by executing it on their *local state* without synchronising with others. Chaincode runs in an isolated Docker container, and results are *not* persisted to the ledger during simulation. The peer transaction manager (PTM) maintains the blockchain state as a *versioned key–value store*. Chaincodes cannot directly access other chaincodes' state but may invoke them within the same channel, subject to permissions[23].

Simulation yields a "readset" (keys read with versions) and a "writeset" (state changes). Each endorser returns an *endorsement*—a signed message containing these sets—back to the client. The client collects endorsements until the policy is satisfied, requiring identical read/write sets. It then submits the transaction to the ordering service[23].

Design Note. Because endorsers execute independently, high-contention workloads can produce divergent outputs, causing endorsement failure. Unlike *primary–backup* replication in synchronised databases[24], Fabric trades this complexity for architectural simplicity, assuming blockchain applications exhibit low contention.

Ordering Phase

The client assembles the endorsed transaction and sends it to the ordering service[23]. The ordering service provides *atomic broadcast* per channel, establishing a total order of transactions. Transactions are batched into blocks, forming a hash-chained sequence[23]. Key API calls include `broadcast(tx)` and `deliver(s)`.

Safety Properties.

- **Agreement** — all correct peers deliver identical blocks for each sequence number.
- **Hash Chain Integrity** — successive blocks link via cryptographic hashes.
- **No Skipping** — blocks are delivered sequentially without gaps.
- **No Creation** — only client-broadcast transactions appear in blocks.

Liveness. If a correct client broadcasts a transaction, all correct peers eventually deliver a block containing it.

Ordering implementations may provide additional liveness and fairness guarantees[25]. Fabric disseminates blocks to all peers via a "gossip service" compatible with both CFT and BFT ordering services.

Discussion. The ordering service *does not* execute or validate transactions, making Fabric the first system to completely separate consensus from execution and validation. Duplicate transactions are filtered later during validation, simplifying ordering logic.

Validation Phase

Upon receiving a block, peers perform three steps[23]:

1. **Endorsement Policy Evaluation** (parallel) — the Validation System Chaincode (VSCC) checks endorsements; non-compliant transactions are marked invalid.
2. **Read–Write Conflict Check** (sequential) — compares each transaction's readset versions with the current ledger state; mismatches mark the transaction invalid.
3. **Ledger Update** — appends the validated block and applies writesets of valid transactions to the peer's world state.

A bitmask of valid transactions is stored to facilitate future state reconstruction. The VSCC supports monotone logical expressions over endorsers; custom VSCCs can be configured statically.

Design Note. Fabric records *invalid* transactions for auditability, unlike Bitcoin or Ethereum. Being permissioned, Fabric can detect and react to DoS attempts, for example by blacklisting offending clients or imposing transaction fees[26, **?**].

Trust and Fault Model

Hyperledger Fabric supports flexible trust and fault assumptions:

- **Clients** — potentially malicious or Byzantine.
- **Peers** — grouped into organisations; a peer trusts peers in its own organisation but not those in others.
- **Ordering Service** — implementations range from centralised (development) to CFT clusters to BFT clusters[27, 28].

Applications define their trust assumptions via endorsement policies, independently of the ordering service's trust model.

1.6 Conclusion

This chapter explored the intersection of digital technologies and healthcare, focusing on IoT and blockchain as enablers of secure, interoperable, and efficient health systems. It outlined the structure and challenges of healthcare systems and how IoT applications enhance patient care and data collection. Blockchain technology was introduced as a solution for trust, transparency, and integrity in health data exchange. Key blockchain components and consensus mechanisms were analyzed, along with their applicability in healthcare. The chapter concluded with a deep dive into Hyperledger Fabric, a modular permissioned blockchain tailored for enterprise use. Together, these technologies pave the way for a more patient-centric, secure, and resilient healthcare infrastructure.

Chapter 2

From Existing Solutions to Proposed Contribution: A Critical Review

2.1 Introduction

Blockchain technology has emerged as a promising solution to long-standing challenges in healthcare data management, particularly concerning trust, privacy, and secure data sharing among multiple stakeholders. Recent research efforts have proposed a variety of blockchain-based architectures aiming to improve the integrity and confidentiality of electronic health records (EHRs), as well as to enable decentralized control and fine-grained access management. Despite these advancements, many proposed solutions remain limited by their reliance on public blockchain platforms, simplistic access control models, or simulated environments that fail to reflect the complexity of real-world deployments. In this chapter, we critically review a selection of recent studies that represent the current state of the art in blockchain-based healthcare systems. Through this analysis, we identify the key limitations that motivate the design choices made in our proposed architecture, which is presented in the following chapter.

2.2 Critical Review of the Literature

This section provides a structured review of four key contributions to securing healthcare data using blockchain technology. The first study explores a public solution based on Ethereum for managing electronic health records. The remaining three works focus on private solutions, primarily using Hyperledger Fabric, applied to clinical systems, healthcare supply chains, and IoT-based health monitoring. This distinction highlights the architectural choices, strengths, and limitations of public versus private blockchain approaches in the healthcare context.

2.2.1 Ethereum Blockchain for Electronic Health Records: Securing and Streamlining Patient Management

Mole and Shaji (2024) in [29] propose a decentralized framework for managing electronic health records (EHRs) using the Ethereum blockchain and InterPlanetary File System

(IPFS). The system leverages Ethereum's immutability to secure access logs and uses IPFS for distributed off-chain storage of patient data, combining transparency with scalability.

The conceptual design highlights several theoretical advantages of decentralized healthcare architectures. However, the practical implementation remains limited in scope. The authors utilize Ganache, a local Ethereum test environment, which fails to simulate the conditions of real-world public or permissioned blockchain networks. As a result, critical aspects such as gas cost variability, transaction confirmation latency, and network congestion are not evaluated. This undermines the system's credibility as a deployable solution in production environments.

From a security and access control perspective, the system incorporates Role-Based Access Control (RBAC) to regulate permissions. While functional in basic scenarios, RBAC lacks the flexibility to support the dynamic and context-aware access requirements common in healthcare environments—such as varying roles, emergency overrides, or condition-based policies. Although the

potential use of Attribute-Based Access Control (ABAC) is acknowledged, it remains unimplemented, leaving a gap in adaptive data governance.

The inclusion of IPFS enables off-chain storage, which is an essential component for handling large- scale medical data. However, the system architecture appears monolithic, with no support for modular deployment, multi-tenant isolation, or cross-organization scalability. These are critical features in healthcare, where data sensitivity varies across stakeholders (e.g., hospitals, labs, pharmacies), and workflows require isolated yet interoperable subsystems.

Furthermore, the design does not address compliance with regulatory standards such as GDPR or HIPAA. There is no implementation of consent revocation, auditability features, or support for patients' right to be forgotten—requirements that are central in any real-world health data management system.

In conclusion, the article makes an important contribution to the conceptual exploration of Ethereum-based healthcare systems. It successfully demonstrates the feasibility of using blockchain and IPFS for decentralized EHR management. However, limitations in deployment realism, access control flexibility, and modularity reduce its immediate applicability. Future work would benefit from testing under realistic network conditions, integration of fine-grained access policies, and alignment with data protection regulations to better position blockchain as a reliable backbone for healthcare data sharing.

2.2.2 The Case of Hyperledger Fabric as a Blockchain Solution for Healthcare Applications

This paper [30] examines the use of Hyperledger Fabric as a permissioned blockchain framework for healthcare systems, emphasizing its potential advantages in terms of security, compliance, and scalability. The authors present a performance-focused analysis, arguing that private blockchain architectures—such as Fabric—offer greater control, lower latency, and better governance compared to public alternatives.

While the paper effectively demonstrates Fabric's modular architecture and customizable consensus mechanisms, several key limitations reduce its applicability in real-world healthcare environments.

First, the evaluation lacks modeling of real-time or high-frequency data scenarios typical in clinical and IoT-enabled monitoring systems. The experimental setup is restricted

to batch-based operations and does not assess how Fabric handles continuous, latency-sensitive data streams—a critical requirement in hospital and IoMT contexts.

Second, the architecture assumes a homogeneous computing environment. The absence of heterogeneous device simulation—ranging from low-power IoT sensors to high-performance servers—means that important constraints such as variable processing power, intermittent connectivity, and bandwidth limitations are overlooked.

Third, the proposed permissioning model relies solely on static Role-Based Access Control (RBAC) using X.509 certificates. There is no implementation or discussion of more dynamic and context- aware mechanisms, such as Attribute-Based Access Control (ABAC), which are often required in healthcare for managing sensitive data access based on patient status, location, or emergency conditions.

In addition, the study does not include failure testing under realistic stress conditions. Scenarios such as network partitioning, node crashes, transaction overload, or endorsement service failures are not addressed, leaving the system's fault tolerance and operational robustness unverified.

Finally, the prototype uses a monolithic channel design in which all types of data—patient records, supply chain events, and administrative logs—are processed within the same Fabric channel. This coupling may reduce modularity, hinder scalability, and complicate data governance in multi- organization deployments.

In conclusion, the paper provides a solid foundational analysis of Hyperledger Fabric's potential benefits for healthcare applications. However, its limited treatment of real-time processing, heterogeneous system behavior, access control granularity, and failure resilience suggests that further research is needed to validate Fabric's readiness for mission-critical healthcare infrastructures.

2.2.3 IoT-Driven Blockchain to Manage the Healthcare Supply Chain and Protect Medical Records

Rizzardi et al. (2024) [31] propose an IoT-driven blockchain framework that leverages Hyperledger Fabric to manage both healthcare data and supply chain workflows. Their implementation demonstrates core blockchain functionalities and emphasizes architectural integration, including

evaluations conducted on both high-end computers and Raspberry *Pi* devices. This highlights an effort to assess the system across different computing environments.

However, despite this dual testing approach, the experimental setup remains largely simulated and does not reflect the complexity of real-world healthcare environments. The reliance on emulated interactions and the limited role of actual IoT devices hinder a complete understanding of system behavior under dynamic, heterogeneous, and latency-sensitive conditions. The use of Raspberry *Pi 3* as a representative edge node may also fall short of capturing the performance and security requirements for scalable medical IoT deployments. Key metrics—such as energy consumption, consensus overhead, and transaction failure rates—are not discussed, leaving questions regarding the framework's efficiency in resource-constrained scenarios.

From a design standpoint, the use of Role-Based Access Control (RBAC) via X.509 certificates provides a functional baseline for managing permissions. Nevertheless, the absence of finer-grained policies such as Attribute-Based Access Control (ABAC) limits adaptability to context-aware healthcare operations, where access rules may vary based on

patient status, location, or role. The architecture also adopts a monolithic organizational model that binds patient records and supply chain processes within the same channel and smart contract logic, which may hinder modularity and future extensibility.

Furthermore, while the system handles on-chain data interactions effectively, it does not elaborate on off-chain storage mechanisms—an essential consideration for handling large volumes of sensitive medical data. The lack of discussion on hybrid storage strategies or compliance with data protection regulations like GDPR weakens its deployment potential. Lastly, although some performance metrics are evaluated, the experiments do not include stress testing under high transaction loads, network failures, or concurrent interactions—scenarios critical for validating robustness in live healthcare ecosystems.

better align with the realities of healthcare data sharing and IoT integration. In conclusion, while the proposed system demonstrates foundational capabilities of blockchain integration in healthcare, it falls short in deployment realism, access flexibility, and resilience evaluation. Future iterations would benefit from modular design, hybrid storage mechanisms, and performance testing under realistic network and operational stress conditions.

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

This paper [32] presents a prototype of a security architecture built on Hyperledger Fabric, intended to secure data flows within Internet of Medical Things (IoMT) health monitoring systems. It utilizes Fabric's permissioned blockchain structure and smart contract features to manage patient- related transactions and preserve an immutable audit trail.

The prototype demonstrates the potential of permissioned blockchains in healthcare applications. However, its deployment is limited to a controlled testing environment—specifically, Docker containers running on a single virtual machine. This setup falls short of representing the heterogeneous, latency-sensitive, and bandwidth-variable nature of real-world IoMT infrastructures. The lack of distributed deployment weakens the relevance of performance claims under actual operational constraints.

Performance testing is conducted using Hyperledger Caliper, which offers a baseline for system throughput and latency. Nonetheless, critical operational dimensions—such as fault tolerance, device mobility, network resilience, and responsiveness under varying loads—are not evaluated. These omissions are significant in the context of health monitoring systems, where real-time reliability is non-negotiable.

Functionally, the prototype's smart contracts implement only basic operations (e.g., asset creation and retrieval) without any integrated access control, privacy logic, or dynamic authorization mechanisms. In sensitive environments like healthcare, where regulatory compliance requires strict role separation and data confidentiality, the absence of such controls limits the practical applicability of the system.

Additionally, the article does not explore alternative blockchain platforms, permissionless models, or even non-blockchain-based security frameworks. Nor does it present comparative benchmarks that would help situate its approach within the broader ecosystem of distributed ledger technologies and secure data architectures.

In conclusion, the paper contributes to the growing body of research on blockchain in healthcare by delivering a Fabric-based prototype for IoMT contexts. However, the limited deployment realism, absence of privacy enforcement, and narrow evaluation scope

suggest that further work is necessary to validate the architecture for real-world medical applications.

2.3 Comparative Table

The table 2.1 summarizes and contrasts the key features and limitations of the four reviewed blockchain-based healthcare solutions across core technical criteria.

Table 2.1: Comparative Analysis of Blockchain-Based Healthcare Systems

Criteria \ Work	**Mole & Shaji (2024)**	**HLF for Healthcare Apps**	**Rizzardi et al. (2024)**	**IoMT-Based Fabric Prototype**
Blockchain Platform	Ethereum (Ganache simulator)	Hyperledger Fabric	Hyperledger Fabric	Hyperledger Fabric
Access Control	RBAC only, ABAC mentioned (not implemented)	None	RBAC via X.509	None
Off-chain Storage	IPFS	Not specified	Not discussed	Not implemented
Realistic Deployment	Simulated with Ganache	Basic blockchain ops only	Simulated on Raspberry Pi + desktop	Docker on single VM
Performance Evaluation	None or limited to gas cost discussion	Latency, throughput	Some metrics (no stress test)	Caliper metrics only
Privacy-Enhancing Techniques	Not discussed	Not discussed	Not discussed	Not discussed
Scalability & Extensibility	Lacks modularity	Lacks scaling design	Limited extensibility	Not extensible
Patient-Centric Data Sovereignty	Not enforced	Not discussed	Not enforced	Not enforced

2.4 Comparative Discussion and Research Gaps

The reviewed literature demonstrates a growing interest in blockchain-enabled healthcare systems, with varying emphases on data integrity, access control, scalability, and integra-

tion with IoT infrastructures. However, a deeper comparison reveals persistent structural and architectural limitations that hinder real-world adoption.

A central divide is observed between public and private blockchain approaches. Mole and Shaji [29] employ Ethereum and IPFS for EHR management, offering transparency and decentralization, but suffering from performance unpredictability and limited access control expressiveness. In contrast, the Fabric-based systems [30, 31, 32] prioritize permissioned environments and governance but often fail to model realistic healthcare workflows or dynamic access requirements.

Across all four works, access control models remain underdeveloped. Three rely solely on static Role-Based Access Control (RBAC), and none implement Attribute-Based Access Control (ABAC), despite its relevance in healthcare settings where permissions may depend on time, context, or patient-defined policies. This omission is particularly limiting in dynamic environments such as hospitals, where emergency overrides or conditional access rules are essential.

Deployment realism also remains a common weakness. Most prototypes are tested in isolated or simulated environments—such as local Ethereum testnets, Docker-based Fabric setups, or Raspberry Pi clusters—without distributed, heterogeneous, or latency-sensitive deployments. As a result, critical challenges such as network resilience, fault tolerance, and real-time responsiveness remain largely unaddressed.

Another notable gap is the absence of modular architectures. Reviewed systems often adopt monolithic channel and chaincode designs, bundling disparate components such as patient data, administrative logs, and supply chain events into a single Fabric channel. This tightly coupled structure limits scalability and maintainability in multi-organizational networks.

Moreover, off-chain storage—an essential requirement for handling large-scale healthcare data—is either missing or poorly integrated. While Mole and Shaji leverage IPFS, others either neglect storage scalability or fail to anchor off-chain content securely. Additionally, data protection regulations such as GDPR or HIPAA are rarely operationalized beyond theoretical mention, with little to no support for consent revocation, audit trails, or data erasure.

In summary, while prior works highlight the potential of blockchain in healthcare, they fall short in addressing core technical and regulatory challenges. There is a clear need for a system that integrates fine-grained and context-aware access control, supports modular deployment, enables secure off-chain data handling, and provides empirical performance insights under real-world conditions. The following section introduces our contribution designed to fill these gaps.

2.5 Our contribution: Hyperledger Fabric–Based Architecture for Secure and Modular Healthcare Data Management

Despite significant progress in blockchain-based healthcare systems, existing approaches continue to face structural and practical limitations. Common gaps include simplistic access control schemes, overreliance on simulated environments, lack of modularity, inadequate off-chain storage integration, and insufficient emphasis on patient-centric data

sovereignty. These issues are especially critical in IoT-driven and privacy-sensitive clinical settings, where data flows are continuous, access policies dynamic, and compliance requirements strict.

In response to these gaps, our proposal introduces a Hyperledger Fabric–based healthcare architecture designed around real-world constraints and modular governance. The system enables secure and decentralized exchange of medical data between a patient organization and a doctor organization, built on a permissioned blockchain with fine-grained access logic.

A central strength of the design lies in its hybrid access control model:

- **Role-Based Access Control (RBAC)** restricts access based on authenticated user roles (e.g., doctor, patient), offering baseline permission enforcement.
- **Attribute-Based Access Control (ABAC)** introduces contextual flexibility, allowing patients to define rules based on time, purpose, or specific conditions, and to dynamically grant or revoke access.

To ensure scalability and storage efficiency, the architecture adopts a *hybrid data model*: medical records are encrypted and stored off-chain, with their integrity anchored on-chain using a cryptographic hash and IPFS-style content identifiers. All sensitive data is encrypted via symmetric cryptography, with encryption keys held and managed exclusively by the patient. Secure key transmission is handled over TLS sessions, with patient-driven revocation and rotation mechanisms reinforcing confidentiality and forward secrecy.

Recognizing the limitations of simulation-heavy literature, the implementation includes a built-in *performance monitoring layer* using Prometheus and Grafana. This setup provides real-time observability of key metrics—block processing time, chaincode execution latency, and endorsement throughput—under varied transaction loads. This performance visibility is essential for assessing readiness in live environments.

By combining modular smart contract architecture, dual-layer access control, cryptographic rigor, and transparent performance analysis, the system addresses several of the core limitations found in prior work. While certain advanced features (e.g., zero-knowledge proofs, integration with national health data infrastructures) remain out of scope, the proposed framework provides a secure, adaptable, and extensible foundation for real-world healthcare data exchange.

2.6 Conclusion

This chapter critically examined four blockchain-based healthcare solutions, highlighting their strengths and recurring limitations. Common gaps include weak access control, unrealistic testing environments, limited modularity, and insufficient support for patient data sovereignty. Ethereum- and Fabric-based systems both show potential but often lack deployment readiness. In contrast, our proposed architecture addresses these issues through a modular design, hybrid RBAC–ABAC access control, encrypted off-chain storage, and real-time performance monitoring. Built on Hyperledger Fabric, it prioritizes privacy, scalability, and practical integration. The approach provides a strong foundation for secure medical data exchange. The following chapter focuses on implementation and evaluation.

Chapter 3

Contribution and Implementation

3.1 Introduction

This chapter presents the technical implementation of our blockchain-based healthcare data management system, building upon the critical analysis of existing solutions discussed in Chapter 2. We introduce a comprehensive architecture that combines Hyperledger Fabric's permissioned blockchain with off-chain storage in Firebase, creating a secure and scalable platform for health monitoring. The system emphasizes patient control through robust access management mechanisms, including both Role-Based and Attribute-Based Access Control (RBAC/ABAC), while maintaining data privacy through client-side encryption. Our design addresses key limitations in current approaches by providing verifiable data integrity, real-time access monitoring, and modular extensibility for future healthcare applications.

3.2 System Overview

This section outlines the objectives and presents the overall system architecture, highlighting key components and interactions.

3.2.1 Objectives

Our system is designed to offer a secure and privacy-focused way to manage healthcare vital records by combining blockchain technology with external (off-chain) storage. At its core, the goal is to give patients more control over their personal health data while making sure that only trusted medical professionals can access or update those records in a transparent and trackable way.

More specifically, the system aims to:

- **Improve Security:** By using Hyperledger Fabric, we ensure that all references to patient data and access logs are tamper-proof. This means once a transaction is recorded, it can't be changed without detection.
- **Protect Privacy:** Instead of storing sensitive medical data directly on the blockchain, we keep it encrypted in an off-chain database (Firebase Real-Time Database). The

blockchain only stores hashes and metadata, which keeps personal information safe from public exposure.

- **Put Patients in Control:** Patients will have ownership over their health data. They can decide who sees what and when, and they'll be able to track all access in real time through clear audit trails.
- **Control Access Based on Identity:** Access is tightly managed using digital certificates from the blockchain network. Only verified users — like registered doctors or patients — will be able to read or update data, based on their roles.
- **Ensure Transparency and Accountability:** Every action involving patient records — whether it's creating, viewing, editing, or deleting — is recorded permanently. This helps build trust and also supports legal and regulatory compliance.
- **Support Growth and Compatibility:** The system is designed to scale easily as more users join and more data is collected. It also supports different storage platforms, making it flexible and future-proof.

3.2.2 General Architecture

The architecture of our system, illustrated by figure 3.1, is built to securely manage healthcare vital records while keeping patients at the center of control. It brings together several components that work in coordination to handle the complete data lifecycle—from the moment a record is created to when it's accessed by an authorized party. The sequence diagram in 3.2 illustrates all partners and actions involved in our system.

The data originates from wearable devices (such as fitness bands or health monitors) that continuously capture vital signs like heart rate, temperature, or blood oxygen levels. These readings are transmitted in real time or at regular intervals to the patient's personal device—typically a smartphone or personal computer—where initial processing and encryption take place.

At the front end, both patients and doctors use a web application to interact with the system. When a patient uploads a new health record, the data received from the wearable device is first encrypted on the client side to ensure that privacy is preserved from the very beginning.

Instead of placing these sensitive data directly on the blockchain, they are stored off-chain using Firebase Realtime Database, a cloud-based NoSQL storage solution. Firebase is well-suited for this application as it enables real-time synchronization, scalability, and secure access control. To preserve privacy and integrity, the health data is encrypted before being uploaded, and only a hash of the data along with its reference ID is stored on the blockchain. This strategy reduces blockchain storage overhead while maintaining verifiability and external data integrity.

Behind the scenes, the web application connects to a backend server that handles authentication, authorization, and business logic. This backend communicates with the Hyperledger Fabric blockchain to store a cryptographic hash of the health record along with a reference ID that points to the encrypted data stored in Firebase Realtime Database. This approach ensures that the data's origin and integrity can be verified without exposing any private medical content on the blockchain.

When a doctor needs access to a patient's record, they must first receive explicit consent. The system checks their permissions using blockchain-based identity management before granting access. Once verified, the backend retrieves the reference to the encrypted data from Firebase Realtime Database, fetches it, and then decrypts it for the authorized doctor to view.

3.2.3 Hybrid Access Control and Key Management

A central pillar of the implementation is the hybrid access control model, combining Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC), supported by a decentralized key management scheme. This structure ensures that patient data remains confidential, access policies are dynamically enforceable, and data sovereignty is preserved by design.

RBAC and ABAC in Chaincode Logic

Role-Based Access Control (RBAC): RBAC offers baseline permission enforcement by assigning predefined roles to network participants (e.g., `patient`, `doctor`, `admin`) during certificate enrollment. Each digital identity issued by the Hyperledger Fabric Certificate Authority (CA) includes role attributes, retrievable via the client identity library in chaincode using `cid.GetAttributeValue()`. For instance:

- `Doctors` can access patient data only when explicitly authorized.
- `Patients` manage their own data and define access policies.
- `Admins` handle identity registration and system audits.

Attribute-Based Access Control (ABAC): To enable context-aware control, ABAC allows patients to define dynamic access conditions using metadata such as time limits, purpose of use, and organizational affiliation. Access policies are expressed as JSON objects and evaluated within chaincode during every access attempt. This enables highly granular and revocable permissions, for example granting a doctor access for 48 hours for a diagnosis, but automatically revoking it afterward.

Granting and Revoking Access

Access is granted through a transaction initiated by the patient. This transaction includes the recipient's identity, an encrypted symmetric key used to decrypt the associated medical record, and the corresponding policy object. The symmetric key is encrypted using the recipient's public key and stored on-chain along with the access metadata and content identifier of the encrypted data.

Revocation is similarly initiated by the patient. A transaction marks the corresponding policy as revoked, ensuring that further access attempts are denied by the smart contract. If necessary, key rotation is performed automatically, and the affected data is re-encrypted with a new symmetric key, rendering prior decryption keys obsolete.

Decentralized Key Management

All medical data is encrypted on the client side using symmetric encryption algorithms prior to being uploaded to off-chain storage. A unique key is generated per record to ensure isolation and minimize risk in the event of compromise.

Key distribution is handled in a decentralized manner. The patient encrypts each symmetric key with the recipient's public key before uploading it to the blockchain as part of the access grant. Communication between clients and the blockchain network is secured through TLS channels to prevent interception.

To maintain forward secrecy and adaptability, the system supports key rotation either on-demand or upon revocation. When keys are rotated, affected records are re-encrypted with the new key, and access control policies are updated accordingly. Private keys are securely stored within the client's local keystore, ensuring that cryptographic control remains exclusively with the data owner.

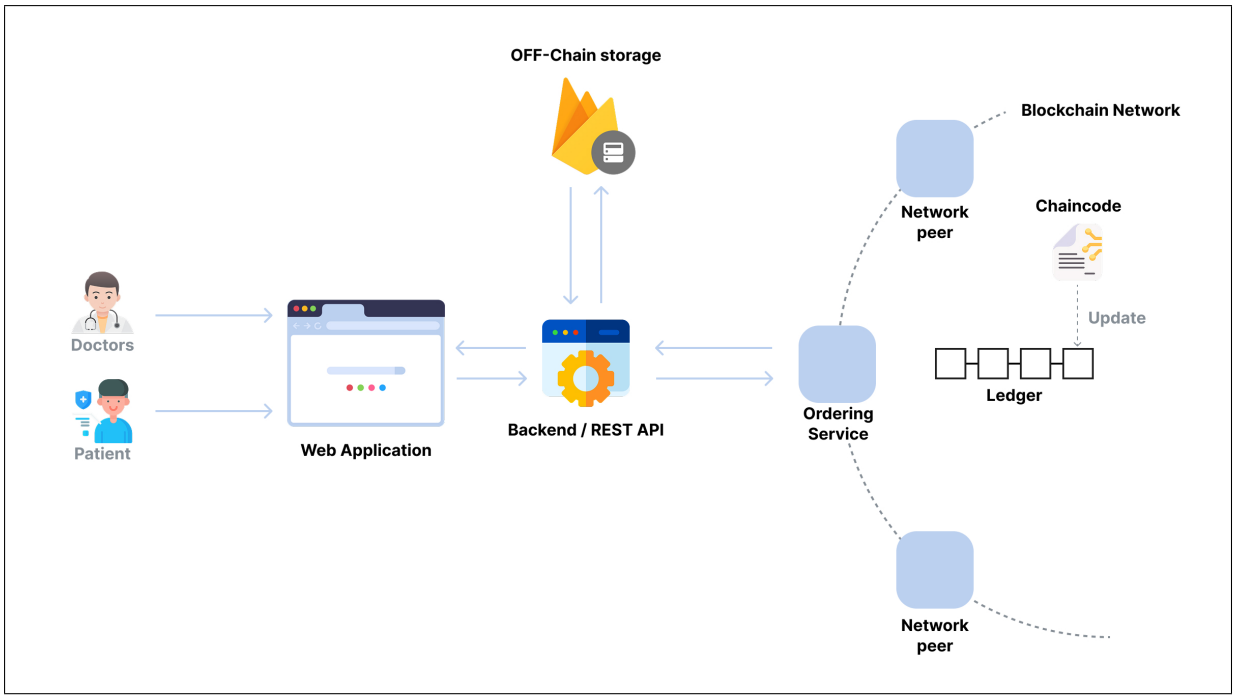

Figure 3.1: System Architecture for Secure Healthcare Data Management.

Every action—such as creating, updating, or managing access to a record—is permanently recorded on the blockchain to ensure transparency and accountability. This design cleanly separates responsibilities: the blockchain is used for verification, auditing, and access control, while Firebase Realtime Database handles the secure and efficient off-chain storage of actual medical data.

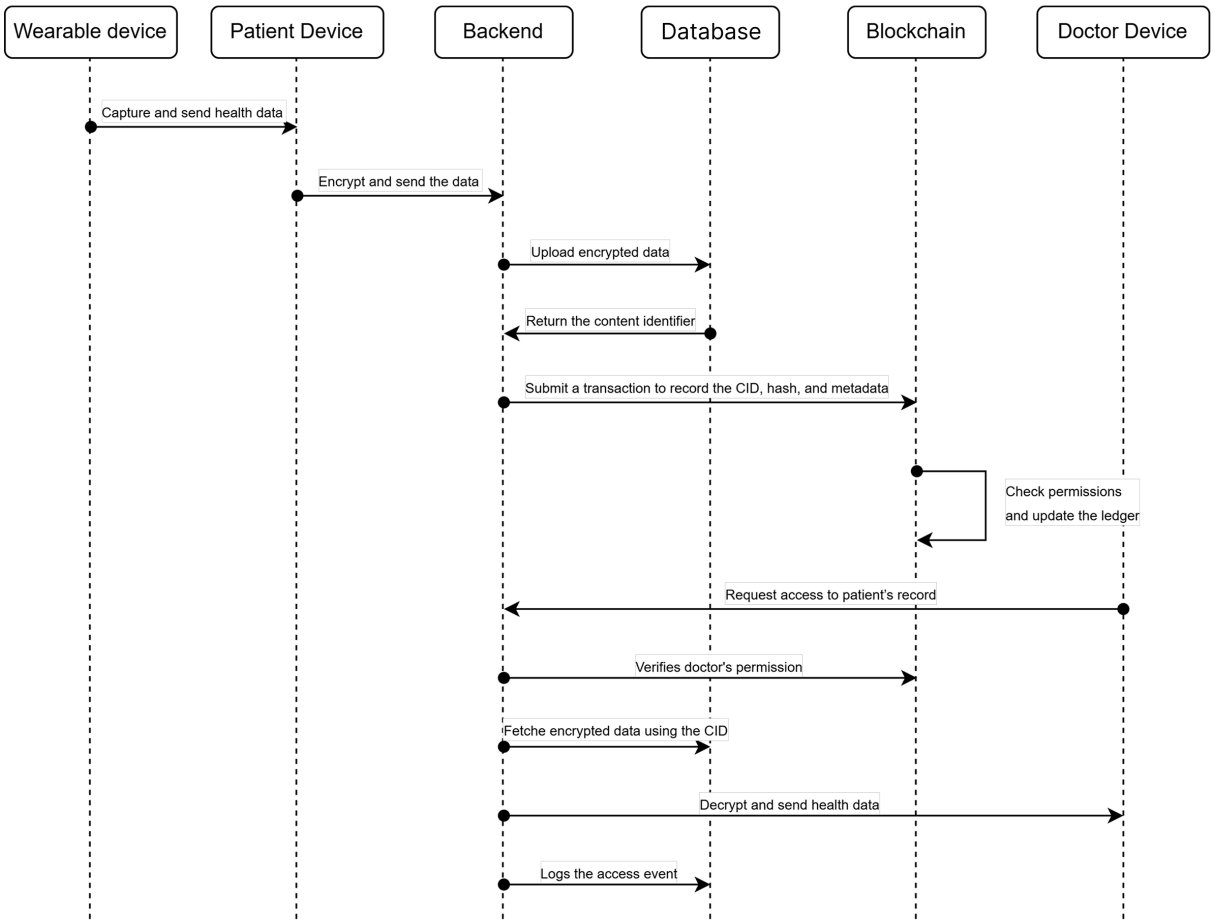

Figure 3.2: Sequence diagram representing the overall workflow and interactions among the system's components

3.3 Technology Stack

This section details the core technologies used, including the blockchain platform, smart contracts, off-chain storage, backend, frontend, and supporting tools.

3.3.1 Blockchain Platform (Hyperledger Fabric)

Our system uses Hyperledger Fabric as the core blockchain platform. As a permissioned blockchain framework, Fabric enables fine-grained, role-based access control over health record references and metadata. It provides modular components for consensus, membership services, and smart contracts (chaincode), ensuring that only authorized participants (patients, doctors, and healthcare organizations) can interact with sensitive data on the network. Fabric's support for private channels and organizations also enables robust data segregation and compliance with privacy regulations. [1]

[1] `https://hyperledger-fabric.readthedocs.io/en/release-2.5/`

3.3.2 Chaincode

Chaincode, which implements the business logic for creating, reading, updating, and deleting health record references on the blockchain, is developed in *Go*. *Go* [2] is the primary and most mature language for Hyperledger Fabric chaincode, offering high performance and robust concurrency. All access control rules and permission checks are enforced in Go chaincode, ensuring secure and reliable execution of transactions within the Fabric network.

3.3.3 Off-chain Storage

To handle the actual storage of encrypted health data, the project uses *"Firebase Realtime Database"*, a cloud-hosted NoSQL database that supports real-time data synchronization. Instead of storing large and sensitive files directly on the blockchain, encrypted health records are uploaded to Firebase. Only a cryptographic hash and metadata—such as a generated identifier and timestamp—are stored on-chain. This approach ensures scalability and privacy, while allowing blockchain to serve as a trusted source of reference and integrity verification.[3]

3.3.4 Backend

This subsection covers the backend architecture built with "Node.js" and "Express.js", integrating the "Hyperledger Fabric SDK" for the blockchain interaction.

Node.js

The backend is implemented using Node.js, a lightweight and efficient JavaScript runtime built on Chrome's V8 engine. Node.js allows for high-concurrency and non-blocking I/O, making it suitable for handling API requests and blockchain interactions.[4]

Express.js

Express.js is used as the web framework for the backend, providing a robust set of features to build RESTful APIs, manage routing, and handle HTTP requests and responses.[5]

Hyperledger Fabric SDK for Node.js

The backend interacts with the Fabric blockchain using the official SDK, which enables secure submission of transactions, retrieval of data, and management of user identities and certificates.[6]

[2] `https://go.dev/doc/`
[3] `https://firebase.google.com/docs/database`
[4] `https://nodejs.org/docs/latest/api/`
[5] `https://expressjs.com/`
[6] `https://hyperledger.github.io/fabric-sdk-node/main/module-fabric-network.html`

3.3.5 Frontend

The user-facing web application is built with *"React.js"*, a popular JavaScript library for creating dynamic and responsive user interfaces. *React* enables efficient state management and component-based architecture, providing a seamless experience for patients and doctors as they interact with their health data.[7]

3.3.6 Other Tools and Libraries

This subsection highlights supporting technologies, focusing on Docker and Docker Compose for containerization and service orchestration.

Docker & Docker Compose

The entire Hyperledger Fabric network, including peer nodes, orderers, and certificate authorities, is containerized using Docker. Docker Compose is used to orchestrate multi-container deployment, simplifying setup, scaling, and management of the blockchain infrastructure.[8]

3.4 Detailed Implementation

This section provides an in-depth look at the system's components, including network setup, chaincode logic, backend and frontend design, performance evaluation, and possible improvements.

3.4.1 Network Components

The Hyperledger Fabric network in our project consists of the following main components, as depicted in Figure 3.3:

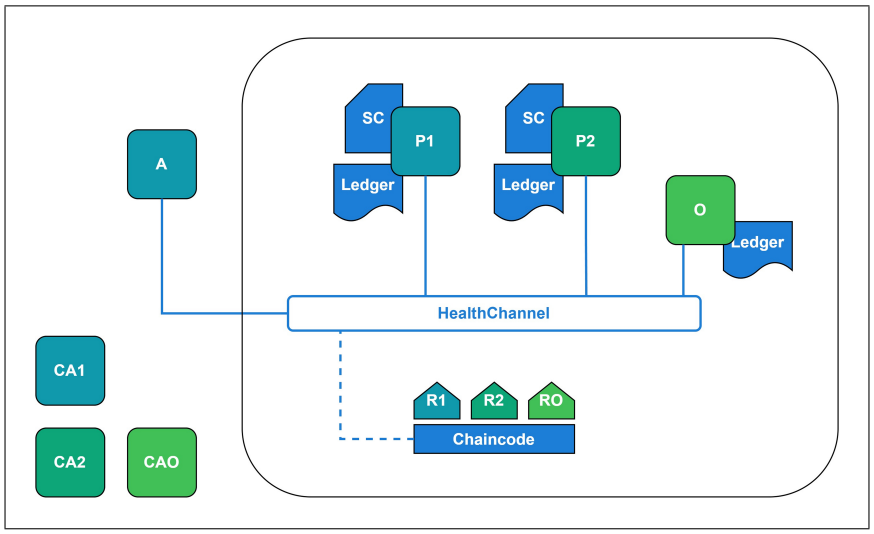

Figure 3.3: Our Fabric network architecture

[7] https://react.dev/
[8] https://docs.docker.com/

- **Organizations:** The network is structured around two organizations, each representing a distinct stakeholder. For this implementation:
  - **Org1 (R1):** Represents the patient-side organization.
  - **Org2 (R2):** Represents the healthcare provider or doctor-side organization.
- **Certificate Authorities (CAs):** Each organization maintains its own Fabric CA, responsible for enrolling users and administering digital certificates. This ensures strong, cryptographically-backed identity management for both users and peers.
- **Client application (A):** Also called a client SDK or application layer, it is the user-facing component that interacts with the blockchain network on behalf of end-users (e.g., patients, doctors, or administrators). It serves as the bridge between human users and the Fabric network.
- **Peer Nodes (P1 and P2):** Each organization hosts at least one peer node. These are the main network participants responsible for endorsing transactions, executing chaincode (smart contracts), and maintaining the distributed ledger (world state and blockchain history).
- **Ordering Service (O):** The ordering service (using the Raft consensus protocol) is responsible for collecting endorsed transactions, ordering them into blocks, and distributing these blocks to all peer nodes for commitment. This ensures consistency and fault tolerance across the network.
- **Channel (HealthChannel):** A single application channel is created to provide a private communication layer between the participating organizations. All chaincode transactions, ledger updates, and state changes related to health records occur within this channel.
- **Chaincode (Smart Contracts (SC)):** Business logic for health record management—including record creation, retrieval, update, deletion, and permission enforcement—is implemented as chaincode written in Go, deployed on the channel and executed by endorsing peers.

3.4.2 Launching the Network

To initialize the Hyperledger Fabric network for this project, we used an automated shell script that brings up all necessary components. This script first launches a fabric network with Certificate Authorities enabled, creates the application channel, and deploys the chaincode with the required endorsement policy. Upon execution, all peer, orderer, and CA services are started as Docker containers, making the network ready for operation (see figure 3.4).

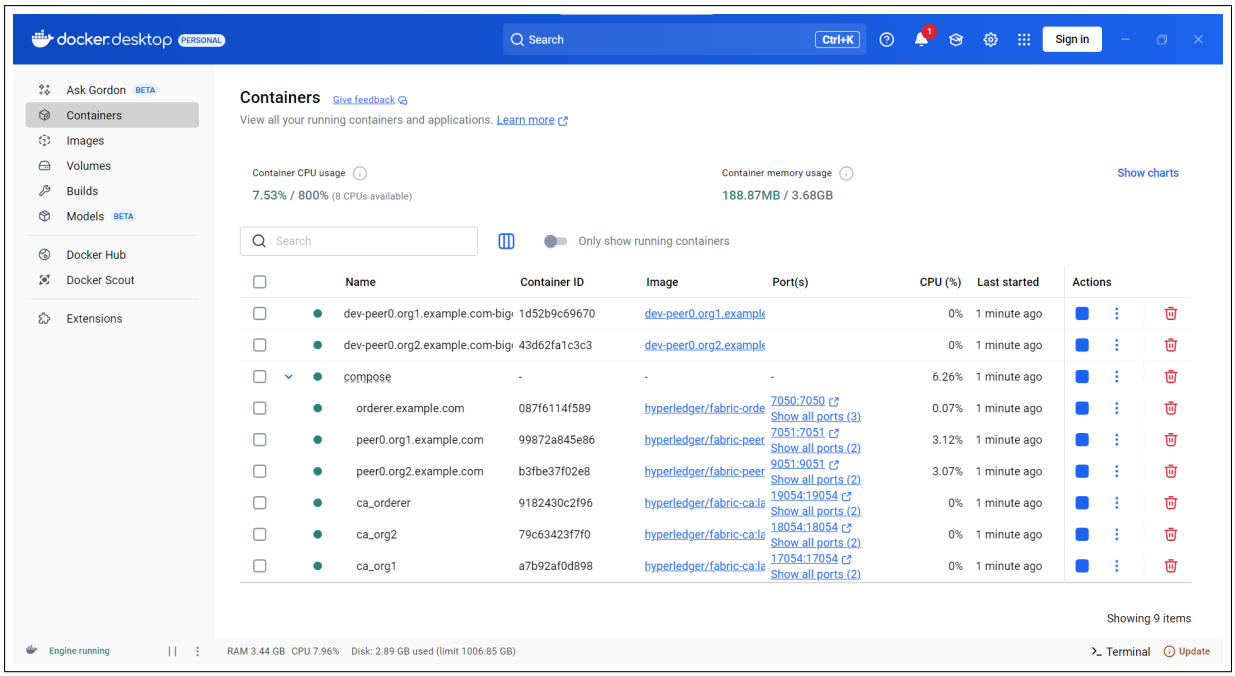

Figure 3.4: Docker Desktop showing all Hyperledger Fabric network containers running after startup

3.4.3 Chaincode Logic

The smart contract developed for our system defines the essential operations for managing health record lifecycles. These operations include record creation, access control, and metadata retrieval, all executed securely on the Hyperledger Fabric network. Each function plays a specific role in enforcing patient control, ensuring data confidentiality, and maintaining auditability across the system.

Create a Record

The algorithm 1 enables patients to create and register metadata of encrypted off-chain health data. It ensures only the record owner can initiate this action.

Algorithm 1 CreateRecord

Require: `recordID`, `patientID`, `dataType`, `cid`, `hash`
Ensure: Record is created only by the patient and saved on ledger

```
clientID ← GetClientID
if clientID ≠ patientID then
    return Error("Unauthorized")
end if
if RecordExists(recordID) then
    return Error("Record already exists")
end if
record ← { "recordID": recordID, "patientID": patientID, "dataType": dataType, "cid": cid, "hash": hash, "accessList": [patientID] }
PutState(recordID, record)
```

Grant Access

The algorithm 2 allows patients to explicitly grant access to specific doctors, reinforcing user-controlled data sharing.

Algorithm 2 GrantAccess

Require: `recordID`, `doctorID`
Ensure: Only patient can authorize doctor access
1: `record` ← ReadRecord(`recordID`)
2: `clientID` ← GetClientID
3: **if** `clientID` ≠ `record.patientID` **then**
4: **return** `Error("Unauthorized")`
5: **end if**
6: **if** `doctorID` ∈ `record.accessList` **then**
7: **return** `Error("Already authorized")`
8: **end if**
9: Append `doctorID` to `record.accessList`
10: PutState(`recordID`, `record`)

Revoke Access

The algorithm 3 allows patients to revoke previously granted permissions, maintaining privacy and control over sensitive data.

Algorithm 3 RevokeAccess

Require: `recordID`, `doctorID`
Ensure: Only patient can revoke access
1: `record` ← ReadRecord(`recordID`)
2: `clientID` ← GetClientID
3: **if** `clientID` ≠ `record.patientID` **then**
4: **return** `Error("Unauthorized")`
5: **end if**
6: Remove `doctorID` from `record.accessList`
7: PutState(`recordID`, `record`)

Get Record Metadata

The algorithm 4 provides authorized users with metadata required to retrieve and verify off-chain health data (from Firebase), while logging access attempts for auditability.

Algorithm 4 GetRecordMetadata

Require: `recordID`
Ensure: Only authorized users can read metadata
1: `record` ← ReadRecord(`recordID`)
2: `clientID` ← GetClientID
3: **if** `clientID` ∉ `record.AccessList` **then**
4: **return** Error: Unauthorized access
5: **end if**
6: logAccess(`recordID`, `clientID`, "VIEW")
7: **return** record metadata (CID, Hash, etc.)

3.4.4 Backend Architecture and Logic

The backend of our system serves as the intermediary layer between the user-facing frontend and the Hyperledger Fabric blockchain network. Built with *Node.js* and *Express.js*, it exposes a set of *RESTful APIs* that handle health record operations, enforce access control, and interact securely with the blockchain network through the Hyperledger Fabric SDK. It ensures this by adopting a clear folder structure, implementing robust API functionality, and enforcing strict security and identity management mechanisms.

Folder Structure.

The backend directory consists of the following key files and components:

- `app.js`: The main entry point that defines API endpoints for record management and access control.
- `fabricService.js`: Handles interactions with the Fabric network using the Fabric SDK, such as connecting to the gateway, retrieving contracts, and accessing the user wallet.
- `config.js`: Stores configuration for organization-specific paths, including connection profiles and wallet directories.
- `enrollAdmin.js` and `registerUser.js`: Utility scripts for registering and enrolling network users with the Certificate Authorities (CAs).
- `wallet/`: A local file-based wallet for storing user credentials and identities.

API Functionality.

The backend exposes multiple endpoints that correspond directly to the chaincode functions deployed on the Fabric network. These endpoints serve both patients and doctors, handling all blockchain transactions and queries through the smart contract interface. The following is an overview of the main routes:

- `POST /record`: Allows a patient to submit a new health record by sending metadata (such as CID and hash of encrypted data). Upon submission, the backend calls the `CreateRecord` chaincode function. Immediately afterward, the system grants access to a default doctor (e.g., `doctora`) by calling `GrantAccess`, enabling seamless monitoring.
- `GET /record/:id`: Enables authorized users to retrieve record metadata stored on-chain. The backend checks access permissions by invoking `GetRecordMetadata`, and returns Firebase-related metadata (such as a content identifier and hash) only if the requester is present in the access list.
- `POST /grant-access`: Allows a patient to manually grant access to a doctor. It forwards the request to the chaincode function `GrantAccess`, modifying the record's access list accordingly.
- `POST /revoke-access`: Enables patients to revoke a doctor's access to a specific record. This is enforced through the `RevokeAccess` chaincode function, ensuring full control over data visibility.

Security and Identity.

Each API request is contextually tied to a user identity, determined via the "userId" parameter and validated against the organization's wallet. If the identity is missing, unauthorized, or not enrolled, the request is rejected with an appropriate error. This ensures only registered and verified participants (patients or doctors) can execute transactions or view data.

3.4.5 Frontend Application

The frontend of our system is a web-based interface developed using *React.js*, a modern JavaScript library known for its component-based architecture and high responsiveness. The interface is designed to support two primary user roles: Patients and doctors. Each user type has a dedicated dashboard that allows secure interactions with the healthcare data stored off-chain in **Firebase** and referenced on the blockchain through the backend APIs.

Directory Structure.

The frontend project is organized as follows:

- `App.js`: The entry point of the application that manages routing and authentication state.
- `components/`: Contains individual React components for patient and doctor dashboards as well as the login interface.
- `firebase/config.js`: A firebase configuration file used to connect to Firebase Authentication and Realtime Database.

- `utils/crypto.js & hash.js`: Handle encryption, decryption, and hashing of patient health data on the client side prior to storage or transmission.

In the following, we present the execution windows, starting with the authentication screen, followed by the patient and doctor interfaces.

Login Interface.

Upon visiting the application, users are presented with a login page, illustrated by 3.5, where they enter their username and password. Authentication is handled via *Firebase Authentication*, which maps the user to their corresponding blockchain certificate stored in the backend's wallet. Depending on their role (patient or doctor), users are redirected to their respective dashboards.

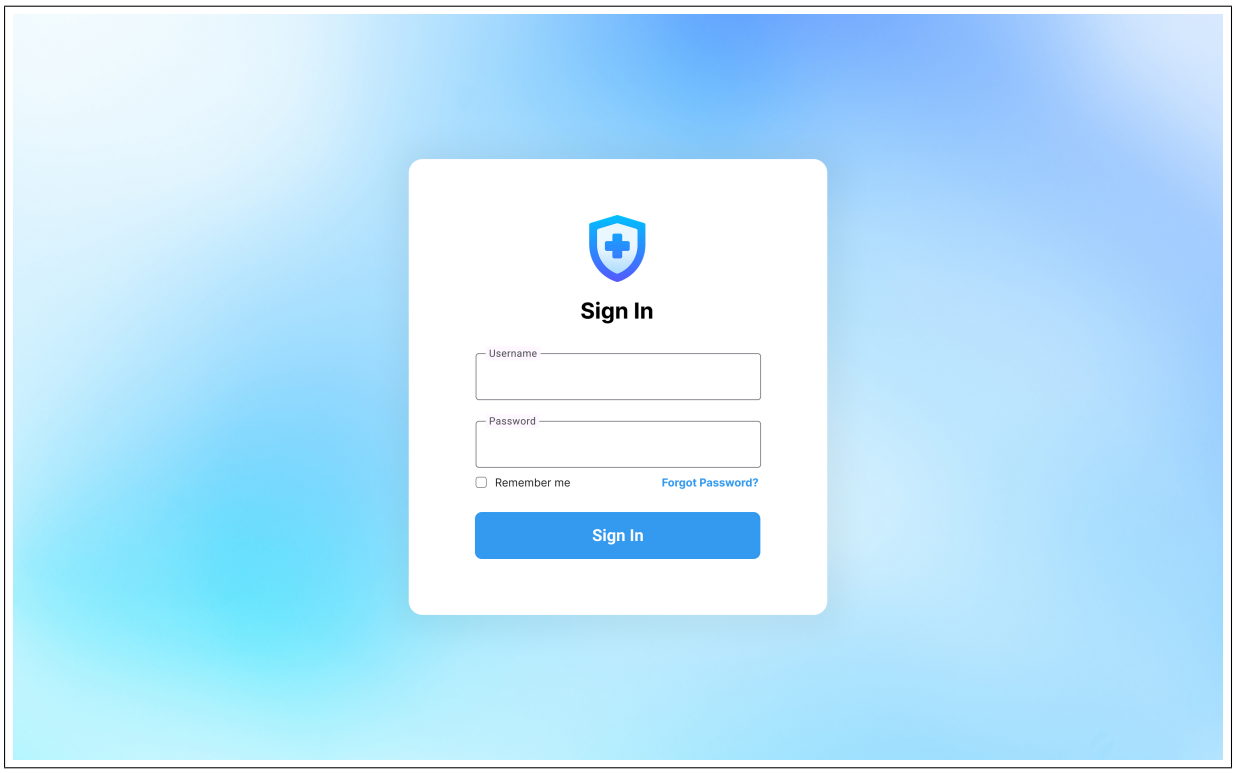

Figure 3.5: User login screen with Firebase-backed authentication.

Patient Dashboard.

Patients can view their previously submitted records and generate new ones (see figure 3.6). Each new health record includes simulated vitals (heart rate and blood pressure), which are encrypted in the browser using AES and hashed using SHA-256. The encrypted data is stored in Firebase Realtime Database, while only the metadata (timestamp, hash, and Firebase CID) is sent to the backend and recorded on the blockchain.

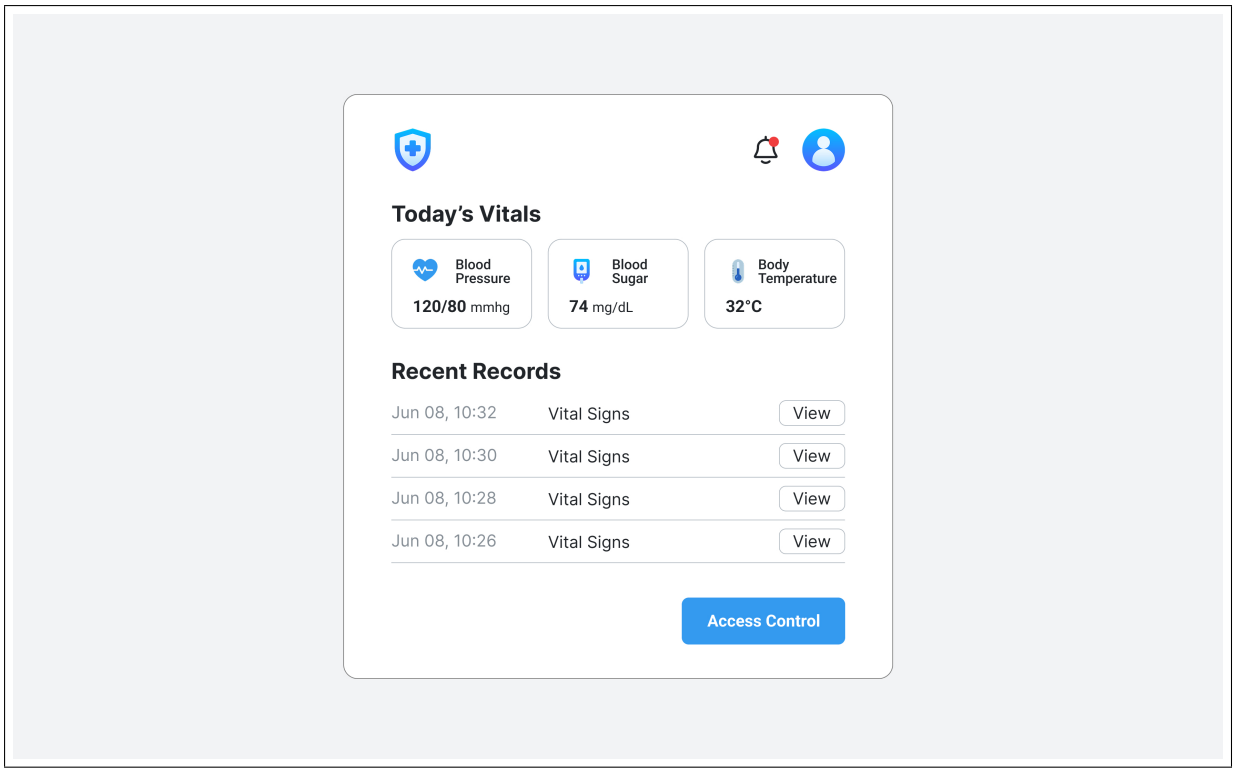

Figure 3.6: Patient Dashboard: View and submit new health records.

Doctor Dashboard.

Doctors who have been granted access can view patient records in real-time. The dashboard, presented in figure 3.7, continuously fetches record metadata from the backend and uses the CID to retrieve encrypted data from Firebase. Once retrieved, the doctor-side application performs:

1. Data decryption using the shared secret key.
2. Hash computation to verify the data against the blockchain metadata.
3. Status display (OK or Tampered) based on the verification.

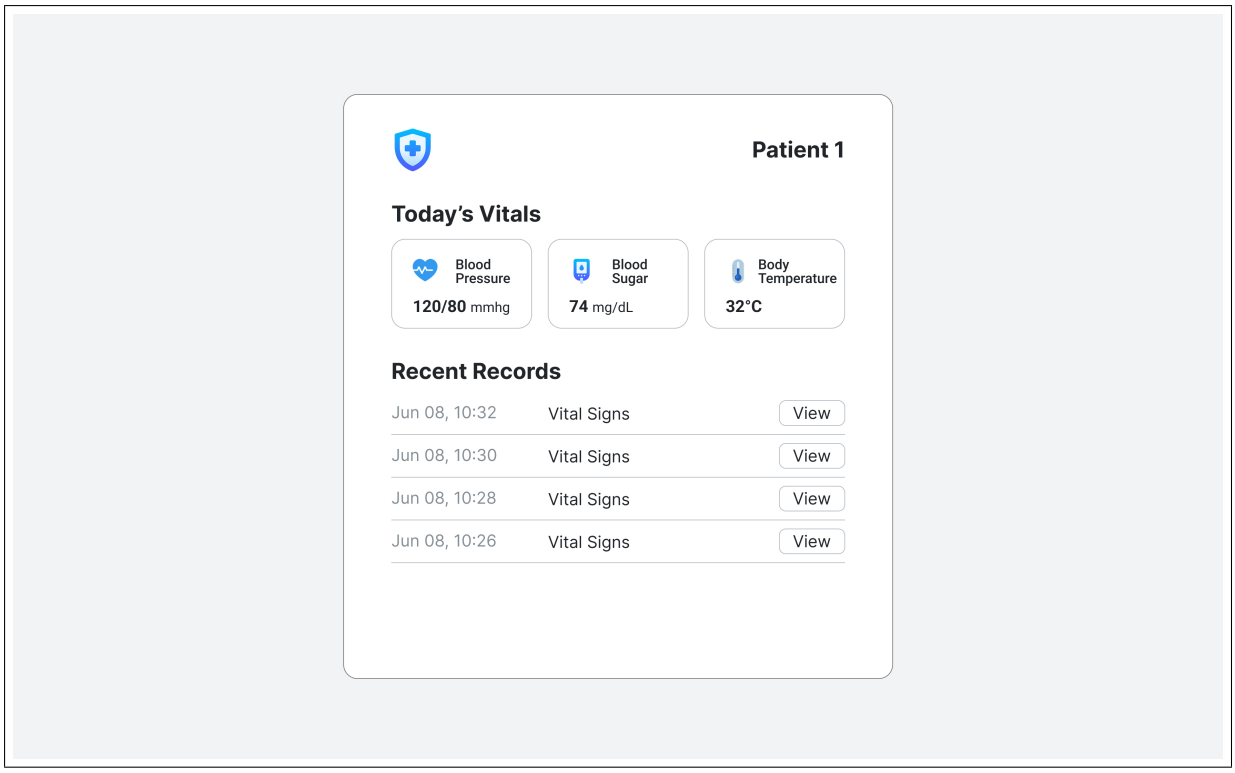

Figure 3.7: Doctor Dashboard: Real-time vitals monitoring and integrity check.

Client-side Security.

To minimize trust assumptions and improve data privacy, all sensitive operations—such as encryption, decryption, and hashing—are performed on the client side. This means that:

- No raw health data is exposed to the backend or blockchain.
- Only the encrypted payload and a hash (used for verification) are shared off-device.
- Data tampering attempts can be detected by comparing hashes stored on-chain with hashes recomputed from Firebase data.

3.4.6 Performance Evaluation and Discussion

To assess the performance of our blockchain-based healthcare monitoring system, we leveraged the built-in monitoring tools available in Hyperledger Fabric's test network. Specifically, we used a real-time Grafana dashboard configured through Prometheus, which provides visibility into blockchain-related metrics at both the peer and the system levels. Additionally, we executed a custom test script that simulates 100 complete transaction workflows—including record creation, access granting, and data retrieval—with short delays between each to emulate semi-realistic usage. This allowed us to observe the chaincode execution, the block commit health, and the overall network stability under consistent load.

We focused on three main categories of metrics:

Ledger Metrics These metrics give insight into the state and performance of the ledger itself:

- **Block Processing Time**: Measures the time taken to validate, order, and commit a block to the ledger. It reflects the efficiency of the consensus and commit process.
- **Blockchain Height**: Indicates the total number of blocks appended to the ledger. It grows linearly with successful transaction activity and is used to track ledger progression.

Endorser Metrics These metrics focus on the proposal and endorsement phase of the transaction lifecycle:

- **Successful Proposal Duration**: The average time taken by endorsing peers to process and approve transaction proposals. Lower values reflect faster endorsement responses.
- **Successful Proposals**: The total number of transaction proposals that were successfully endorsed by peers. This indicates the responsiveness and availability of endorser peers.

Chaincode Metrics These metrics evaluate the performance and workload of the chaincode execution layer:

- **Request Duration**: Measures the time taken to process a chaincode request from invocation to response. It reflects the efficiency and responsiveness of the smart contract logic under load.
- **Requests Received**: Indicates the total number of chaincode invocation requests handled by the peer. It gives insight into the chaincode activity level and transaction load.

These metrics collectively provide a comprehensive view of the system's performance and responsiveness under load. Based on our tests, the system maintained stable behavior with acceptable block processing and endorsement times, indicating that the architecture is scalable and suitable for moderate healthcare data workloads.

Simulation Results

This subsection presents the results of the simulation, focusing on both ledger and endorser metrics collected through the monitoring tools. These results reflect the system's behavior under load and provide insight into performance, responsiveness, and stability.

Ledger Metrics

The figure 3.8 presents metrics related to block processing efficiency and ledger growth.

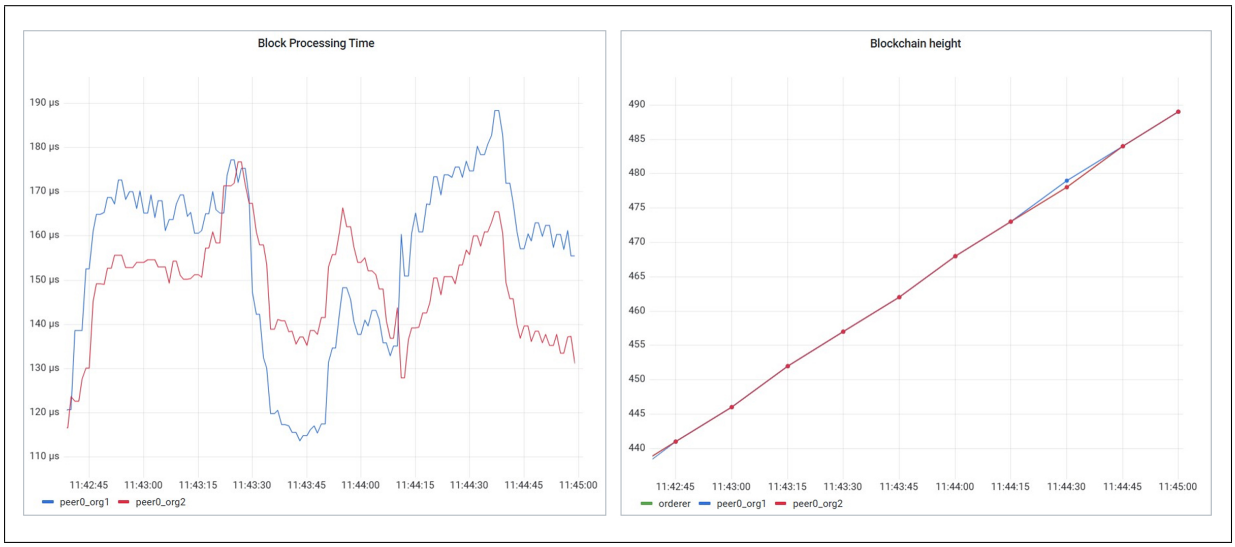

Figure 3.8: Ledger Metrics: Block Processing Time and Blockchain Height over time

- *Block Processing Time.*, Refers to the duration taken by each peer to process a block after receiving it from the ordering service. In our tests, this value ranged between 110 μs and 190 μs for both *peer0.org1* and *peer0.org2*, indicating low latency overall. However, occasional fluctuations suggest *jitter* in processing time, which could stem from load variations or endorsement delays. For a real-time healthcare system, reducing this inconsistency—by tuning peer configurations or improving hardware—would improve system reliability.

- *Blockchain Height.* Tracks the number of successfully committed blocks over time. All nodes, including the orderer and both peers, showed a smooth and linear increase with no signs of rollback or consensus failure. This confirms stable and continuous block generation throughout the evaluation period.

Endorser Metrics

The figure 3.9 presents metrics related to endorsement responsiveness and proposal handling efficiency.

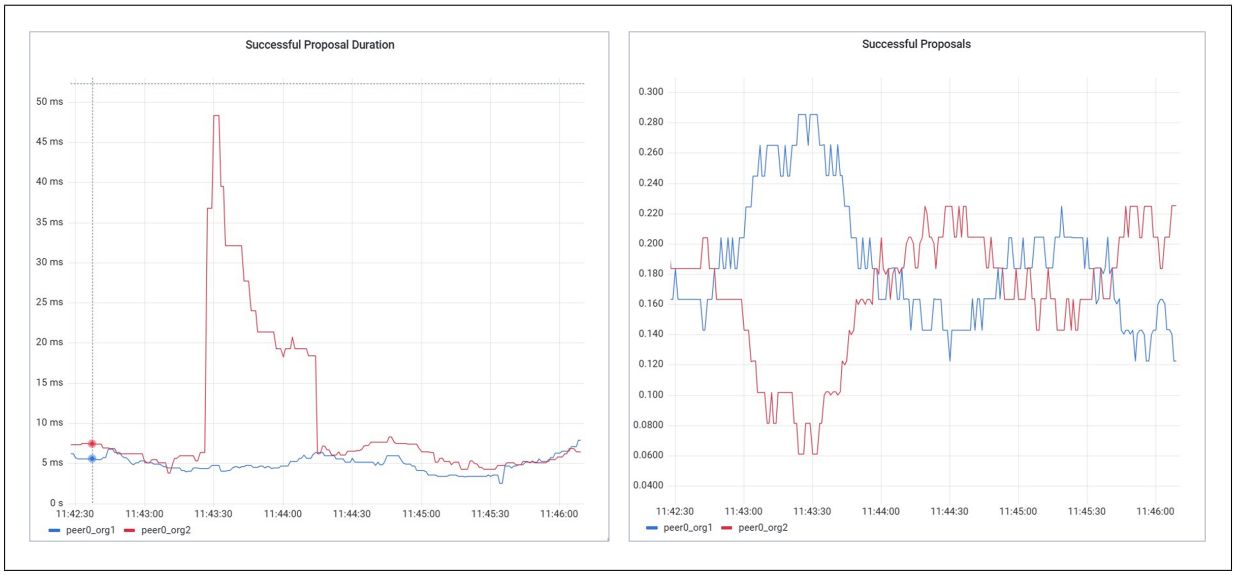

Figure 3.9: Endorser metrics: Proposal duration and throughput trends across peers

- *Successful Proposal Duration.* Represents the time required to complete a successful transaction proposal during the endorsement phase. During the evaluation, durations peaked around *45 ms* before stabilizing below *15 ms*. The initial spike may be attributed to chaincode cold starts or temporary peer overloads. After this warm-up phase, performance remained stable, indicating efficient handling of endorsement tasks.

- *Successful Proposals & Requests Completed.* Measure the rate at which endorsements and transaction completions occur. Values fluctuated between *0.1* and *0.25*, reflecting varying load. Although this variation is expected in benchmarking environments, it highlights the importance of stress testing the system under increased user/device activity to ensure future scalability.

Chaincode Metrics

The figure 3.10 presents metrics related to chaincode request handling and execution efficiency.

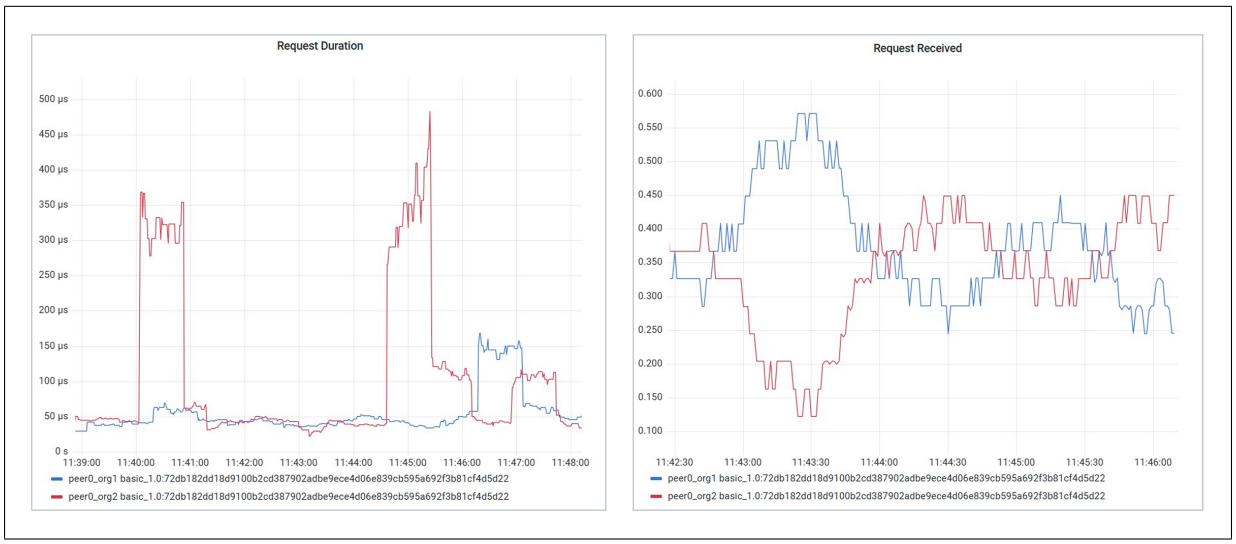

Figure 3.10: Chaincode performance: Request latency and peer request load trends

- *Request Duration.* Measures the total time taken for a client transaction, covering the full cycle from proposal to commit. Most durations remained under 300 µs, with occasional spikes exceeding 450 µs. These spikes may be caused by temporary network delays or the internal logic of the chaincode. As the chaincode enforces access control and verification, optimizing its execution—such as avoiding redundant `GetState()` calls—can help reduce such latency.

- *Request Received.* Indicates how many transaction requests received by each peer. The graph show periodic oscillations, suggesting generally good load distribution with minor throughput variation. While this is acceptable, consistent dips may signal momentary throttling or peer-level delays that should be monitored.

Results Discussion

The performance evaluation results demonstrate that our system maintains stability and responsiveness under moderate load, validating the effectiveness of the architectural choices. The ledger metrics, particularly the consistent *Block Processing Time* and steady increase in *Blockchain Height*, indicate that the ordering and committing processes are functioning efficiently without bottlenecks or delays. This confirms that the consensus mechanism and peer synchronization are correctly configured and able to support continuous transaction flow.

The endorser metrics reinforce this observation, with a high count of *Successful Proposals* and low *Proposal Duration*, showing that endorsing peers respond reliably and promptly. This reflects both the low computational cost of the chaincode functions and the proper distribution of endorsement policies.

Regarding the chaincode metrics, the *Request Duration* remained low and stable, suggesting that the smart contract logic is lightweight and optimized. The volume of *Requests*

Received aligns with the number of simulated transactions, confirming that no calls were dropped or blocked. These results justify the choice of minimalistic, modular chaincode and a clear separation of logic between on-chain validation and off-chain data handling.

Overall, the system's performance confirms that the combination of Hyperledger Fabric, well-structured backend APIs, and off-chain storage enables efficient data flow, controlled access, and robust processing without compromising scalability. Future stress testing under higher concurrency can further validate the architecture's resilience and identify upper performance bounds.

3.5 Conclusion

The implementation described in this chapter successfully demonstrates a working prototype of a blockchain-enabled healthcare data management system that balances security, privacy, and usability. Through careful system design and performance evaluation, we have shown how distributed ledger technology can facilitate secure health record sharing while maintaining patient sovereignty over sensitive data. While the current implementation achieves its core objectives, we identify several areas for future enhancement, including improved scalability for IoT device integration and expanded functionality for comprehensive healthcare workflows. This work establishes a foundation for further development of decentralized healthcare solutions that can meet the complex requirements of modern medical data ecosystems.

General Conclusion

The evolution of digital healthcare systems brings both opportunities and significant challenges-particularly around the secure, private, and efficient handling of medical data. In this thesis, titled *"A Secure Hyperledger Fabric Network for IoT-Based eHealth Monitoring and Data Management"* we proposed a decentralized platform that enhances data transparency, ensures integrity, and empowers patients with control over their information, while enabling secure and traceable interactions among healthcare actors.

To achieve this, we designed and implemented a modular system that combines Hyperledger Fabric for storing verifiable on-chain metadata with Firebase Realtime Database for managing encrypted off-chain records. The architecture incorporates both Role-Based and Attribute-Based Access Control models, allowing fine-grained, patient-driven data governance while maintaining full system accountability.

Through a structured methodology-spanning literature review, architecture design, chaincode development, full-stack implementation, and performance evaluation-we validated the system's feasibility. Simulated testing confirmed good responsiveness, consistent behavior, and architectural robustness under load. Beyond the technical output, the project also provided valuable real-world experience in blockchain development, system security, and healthcare informatics.

In conclusion, our work lays a solid foundation for the development of secure, decentralized, and patient-centered healthcare platforms. It shows that permissioned blockchains-when thoughtfully integrated with off-chain storage and fine-grained access control-can address critical challenges in digital health infrastructure and open promising pathways for future innovation.

However, due to time and scope constraints, certain features such as the drug supply chain and pharmacy integration were deferred. These aspects, along with technical enhancements, form the basis for future work.

Future Work

To evolve this prototype into a production-grade system, the following directions are proposed:

- **Performance Optimizations:** Improve chaincode efficiency, fine-tune endorsement policies, and allocate more computing resources to reduce latency and increase throughput. Full stress-testing and real-world deployments should be conducted to evaluate scalability under heavy load.
- **Monitoring Enhancements:** Integrate automated monitoring and alerting (e.g., with Prometheus and Grafana) to support real-time diagnostics, anomaly detection, and proactive maintenance.

- **Feature Expansion:** Add modules for appointment scheduling, digital prescriptions, insurance claims, and unified medical histories. Introduce analytics dashboards for patients and care providers.
- **Pharmacy Organization Integration:** Extend the network to include a third organization representing pharmacies. This would enable a secure, blockchain-based drug dispensation and supply chain system, improving traceability, reducing fraud, and ensuring regulatory compliance.
- **Broader Stakeholder Onboarding:** Explore integration with external systems (e.g., hospital EHRs, pharmacy software), and address interoperability, legal, and governance issues for real-world deployment.

Bibliography

[1] H. Durrani, "Healthcare and healthcare systems: inspiring progress and future prospects," *Mhealth*, vol. 2, p. 3, 2016.

[2] M. H. Kashani, M. Madanipour, M. Nikravan, P. Asghari, and E. Mahdipour, "A systematic review of iot in healthcare: Applications, techniques, and trends," *Journal of Network and Computer Applications*, vol. 192, p. 103164, 2021.

[3] S. P. Bhavnani, J. Narula, and P. P. Sengupta, "Mobile technology and the digitization of healthcare," *European heart journal*, vol. 37, no. 18, p. 1428, 2016.

[4] C. A. Okolo, S. Ijeh, J. O. Arowoogun, A. O. Adeniyi, and O. Omotayo, "Reviewing the impact of health information technology on healthcare management efficiency," *International Medical Science Research Journal*, vol. 4, no. 4, pp. 420–440, 2024.

[5] K. O. M. Salih, T. A. Rashid, D. Radovanovic, and N. Bacanin, "A comprehensive survey on the internet of things with the industrial marketplace," *Sensors*, vol. 22, no. 3, p. 730, 2022.

[6] G. Bigini, V. Freschi, and E. Lattanzi, "A review on blockchain for the internet of medical things: Definitions, challenges, applications, and vision," *Future Internet*, vol. 12, no. 12, p. 208, 2020.

[7] P. Ratta, A. Kaur, S. Sharma, M. Shabaz, and G. Dhiman, "Application of blockchain and internet of things in healthcare and medical sector: applications, challenges, and future perspectives," *Journal of Food Quality*, vol. 2021, no. 1, p. 7608296, 2021.

[8] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," Tech. Rep., 2019.

[9] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International journal of web and grid services*, vol. 14, no. 4, pp. 352–375, 2018.

[10] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

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

[12] M. Di Pierro, "What is the blockchain?" *Computing in Science & Engineering*, vol. 19, no. 5, pp. 92–95, 2017.

[13] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, "Everything you wanted to know about the blockchain: Its promise, components, processes, and problems," *IEEE Consumer Electronics Magazine*, vol. 7, no. 4, pp. 6–14, 2018.

[14] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *Ieee Access*, vol. 7, pp. 117 134–117 151, 2019.

[15] M. Andoni, V. Robu, D. Flynn, S. Abram, D. Geach, D. Jenkins, P. McCallum, and A. Peacock, "Blockchain technology in the energy sector: A systematic review of challenges and opportunities," *Renewable and sustainable energy reviews*, vol. 100, pp. 143–174, 2019.

[16] T. A. Telia and F. Masoodi, "Blockchain in healthcare: Challenges and opportunities," in *Proceedings of the 2nd International Conference on IoT Based Control Networks and Intelligent Systems (ICICNIS 2021)*, India, 2021, cluster University Srinagar and University of Kashmir. [Online]. Available: mailto:mtawseef805@gmail.com

[17] M. Hölbl, M. Kompara, A. Kamišalić, and L. Nemec Zlatolas, "A systematic review of the use of blockchain in healthcare," *Symmetry*, vol. 10, no. 10, p. 470, 2018.

[18] A. Hasselgren, K. Kralevska, D. Gligoroski, S. A. Pedersen, and A. Faxvaag, "Blockchain in healthcare and health sciences—a scoping review," *International journal of medical informatics*, vol. 134, p. 104040, 2020.

[19] "Hyperledger fabric," http://github.com/hyperledger/fabric, accessed: 2025-05-25.

[20] "Hyperledger," http://www.hyperledger.org, accessed: 2025-05-25.

[21] "The linux foundation," http://www.linuxfoundation.org, accessed: 2025-05-25.

[22] A. S. Tanenbaum, "Distributed operating systems anno 1992. what have we learned so far?" *Distributed Systems Engineering*, vol. 1, no. 1, p. 3, 1993.

[23] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," pp. 1–15, 2018.

[24] B. Kemme and G. Alonso, "A new approach to developing and implementing eager database replication protocols," *ACM Transactions on Database Systems (TODS)*, vol. 25, no. 3, pp. 333–379, 2000.

[25] C. Cachin, R. Guerraoui, and L. Rodrigues, *Introduction to reliable and secure distributed programming.* Springer Science & Business Media, 2011.

[26] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Computing Surveys (CSUR)*, vol. 22, no. 4, pp. 299–319, 1990.

[27] J. Sousa, A. Bessani, and M. Vukolic, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2018, pp. 51–58.

[28] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with bft-smart," in *2014 44th Annual IEEE/IFIP international conference on dependable systems and networks*. IEEE, 2014, pp. 355–362.

[29] J. S. Mole and R. Shaji, "Ethereum blockchain for electronic health records: securing and streamlining patient management," *Frontiers in Medicine*, vol. 11, p. 1434474, 2024.

[30] M. Antwi, A. Adnane, F. Ahmad, R. Hussain, M. H. ur Rehman, and C. A. Kerrache, "The case of hyperledger fabric as a blockchain solution for healthcare applications," *Blockchain: Research and Applications*, vol. 2, no. 1, p. 100012, 2021.

[31] A. Rizzardi, S. Sicari, A. Coen-Porisini *et al.*, "Iot-driven blockchain to manage the healthcare supply chain and protect medical records," *Future Generation Computer Systems*, vol. 161, pp. 415–431, 2024.

[32] F. Pelekoudas-Oikonomou, J. C. Ribeiro, G. Mantas, G. Sakellari, and J. Gonzalez, "Prototyping a hyperledger fabric-based security architecture for iomt-based health monitoring systems," *Future Internet*, vol. 15, no. 9, p. 308, 2023.