Cyprus University of Technology

杭州電子科技大學 HANGZHOU DIANZI UNIVERSITY

**Master's thesis**

# IoT and blockchain-based human health monitoring system

**Jun Chen**

**Limassol, December 2024**

MSc in Electronics Science and Technology

CYPRUS UNIVERSITY OF TECHNOLOGY

Faculty of Engineering and Technology

Department of Electrical Engineering, Computer Engineering, and Informatics

Master's thesis

# IoT and blockchain-based human health monitoring system

Jun Chen

Supervisor

Andreas Andreou

Limassol, December 2024

**Approval Form**

Master's thesis

# IoT and blockchain-based human health monitoring system

Presented by

Jun Chen

**Supervisor**: Andreas Andreou

**Member of the committee**: Committee member 1

**Member of the committee**: Committee member 2

Cyprus University of Technology
Limassol, December 2024

# Copyrights



The approval of the dissertation by the Department of Electrical Engineering, Computer Engineering, and Informaticsdoes not necessarily imply the approval by the Department of the views of the writer.

# Acknowledgements

Time flies so fast. It has been ten months since I landed in Cyprus. Before starting my postgraduate studies, I never thought I would study and live abroad. This experience has been amazing and unforgettable for me. During this precious time, I have gained knowledge and experienced growth. As I complete my thesis, I am filled with gratitude and would like to extend my most sincere thanks to those who have supported and helped me.

First of all, I would like to express my special gratitude to my supervisor, Professor Andreou. From the confusion of choosing a thesis topic, to the in - depth exploration of the research, and even to the careful consideration of every word in writing the thesis, your meticulous guidance and patient instruction have always been with me, leading me step by step towards my goal.

Meanwhile, I also want to thank those teachers who helped us adapt to life in Cyprus. In a foreign land, your care and assistance made me feel the warmth of home, enabling me to quickly integrate into the local life and focus on my studies with peace of mind.

Dr. Stelios in the laboratory has also given me a lot of help. On the journey of learning, your help and guidance have enlightened me when I encountered difficulties, allowing me to solve problems smoothly.

Thanks to my classmates. In the study and life abroad, we encouraged and supported each other and made progress together. We discussed academic issues, shared learning experiences, and spent many fulfilling and unforgettable days. Your company made me feel warmth and strength far away from home.

Finally, I want to thank myself for my perseverance and hard work in these ten months. This experience has made me more independent, strong, and confident. I believe that this precious overseas study experience will become a valuable asset in my life, inspiring me to continuously pursue excellence in the future.

In the days to come, I will carry this gratitude in my heart, continue to study and explore with efforts, and strive unremittingly to achieve my life goals.

# ABSTRACT

With the rapid development of technology, blockchain and Internet of Things (IoT) have been widely applied in various fields, particularly in the healthcare sector. These technologies show great potential in improving patient access to health records, streamlining health claim processes, and enabling secure data sharing between healthcare providers and patients. This paper first reviews the relevant research background, analyzing the current application status and challenges of blockchain and IoT technologies in healthcare through a detailed examination of existing literature. Based on this analysis, a solution that integrates medical IoT devices with a private blockchain structure is proposed, aiming to protect personal information security while enhancing the efficiency and quality of healthcare services.

**Keywords**: Blockchain, Internet of Things (IoT), Healthcare Data Privacy, Smart Contracts, Security Architecture

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| VAR | Virtual Augmented Reality |
| ADHD | Attention Deficit-Hyperactivity Disorder |
| BMI | Body Mass Index |

# 1 Introduction

## 1.1 Research Background

A paradigm shift in digital technologies is underway, transitioning healthcare from traditional models to smart healthcare, which is poised to revolutionize healthcare systems globally. Smart healthcare leverages advanced digital technologies to streamline the navigation of health information, making it easily accessible and manageable. It also facilitates the seamless connection between individuals, healthcare resources, and organizations, fostering better communication and collaboration. By integrating these technologies, smart healthcare systems can intelligently and effectively handle and respond to the dynamic demands of the health environment, leading to more efficient, personalized, and effective healthcare delivery. This transformation not only enhances the quality of care but also improves patient outcomes and satisfaction, making healthcare more accessible and affordable for all[1].This new generation of healthcare systems is called Healthcare 5.0.

Healthcare 5.0 represents a new paradigm in digital health systems, leveraging artificial intelligence, the Internet of Things, and 5G communication to enhance the efficiency, accuracy, and cost-effectiveness of healthcare delivery. This innovative approach has the potential to transform healthcare systems by providing real-time data processing and remote access capabilities. Indeed, Healthcare 5.0 allows healthcare professionals to access patient data from any location, enabling faster diagnosis and treatment. Furthermore, the use of predictive analytics and AI can lead to improved patient outcomes and overall better healthcare experiences.[2].

The healthcare system has undergone multiple transformations. From 1970 to 1990, it was known as the Healthcare 1.0 era. During this time, medical information was primarily transmitted through paper-based systems. This model prevailed for many years; however, as time passed, paper records were prone to wear and loss, required substantial space for storage, and failed to adequately protect the privacy and security of patient medical information. To address these issues, the second generation of the healthcare system, commonly referred to as Electronic Health, was introduced between 1991 and 2005. During this period, significant advancements in manufacturing led to the development of numerous advanced medical devices and instruments. These included digital imaging test equipment, such as Magnetic Resonance Imaging (MRI) and Computerized Tomography (CT), digital tracking devices like pulse oximeters and arterial catheters, as well as surgical and life support equipment, such as the Da Vinci robot and chest tubes. These innovations were all applied and promoted within Healthcare 2.0.

With the continuous advancement of medical technology, telehealth and electronic health records were gradually integrated into the Healthcare 2.0 system from 2006 to 2015. This integration not only significantly improved the efficiency of medical services but also laid the foundation for the birth of Healthcare 3.0. The electronic health record system enables healthcare professionals to upload, share, and access health data stored in the cloud at any time. However, cloud servers are vulnerable to active security attacks by malicious entities who may attempt to access critical patient data, which could subsequently be sold or used for personal purposes. To address this challenge, from 2016 to the present, artificial intelligence and various emerging digital technologies, including the Internet of Things, fog computing, mobile technology, blockchain, machine learning, virtual reality and augmented reality, robotics, big data analysis, and high-tech intelligent devices, have been integrated into Healthcare 3.0. This process of technological convergence and upgrading is known as Healthcare 4.0, marking the healthcare system’ s evolution towards a more intelligent, secure, and efficient direction. Finally, Healthcare 5.0 leverages technological advancements to enhance the quality

of healthcare. One of the main objectives of Healthcare 5.0 is to provide patient-centered care, which is personalized, proactive, and integrated across different healthcare environments. This is achieved through the use of advanced technologies such as artificial intelligence, which enables healthcare providers to analyze large volumes of data to identify patterns and make more accurate diagnoses.

In the past few decades, the Internet of Things (IoT) has achieved internal connectivity among all objects, widely recognized as the next technological revolution after computers and the internet. IoT's applications are continuously expanding, covering aspects such as smart health monitoring[3], intelligent parking[4], smart homes[5], smart cities[6], smart climate[7], industrial sites[8], and agricultural fields[9].Among them, IoT's application in healthcare management stands out. It not only provides convenient health and environmental condition monitoring but also allows data to be transmitted in real-time to remote M2M systems, including machine-to-machine, machine-to-person, handheld devices, and smartphones. This method is not only simple and easy to implement but also efficient, energy-saving, intelligent, scalable, and interoperable, enabling precise tracking and optimized care for various health issues[10].The term IoMT (Internet of Medical Things) refers to the interconnection of various medical devices, wearables, sensors, and other medical technologies achieved through internet-based technologies. These technologies enable efficient data collection, transmission, and exchange, bringing unprecedented transformative potential to the healthcare sector. By providing real-time patient data and remote monitoring capabilities, IoMT not only allows doctors to access critical information and make more precise diagnostic and treatment decisions, but also significantly improves the overall quality of patient care. This seamlessly integrated medical IoT system not only optimizes the allocation and utilization of healthcare resources but also enhances the interaction between doctors and patients, enabling more humanized medical services.IoMT has become an important part of Healthcare 5.0.

Most current IoMT (Internet of Medical Things) systems are typically structured into four layers. These layers cover the entire process from the collection of personal biometric data, through data storage, to visualization and analysis by doctors. Additionally, patients can view their overall health status in real-time through the cloud. With the continuous development of IMDS (Internet of Medical Data Systems), IoWD (Internet of Wireless Devices) and IMD (Internet of Medical Devices) often adopt similar architectural designs, primarily because IMDs can communicate smoothly with gateways. The first layer is the sensor layer, which consists of small implantable or wearable sensors used to collect patients' biometric data. Data is transmitted to the second layer using wireless protocols such as Wi-Fi, Bluetooth, or the MedRadio frequency (RF) spectrum designated for IMDs. The second layer, known as the gateway layer, receives this data due to the processing and storage limitations of IoMT sensors. Devices in this layer, such as the patient's smartphone or a dedicated access point, typically have more computational power than sensors and can perform preliminary operations like verification, temporary data storage, and basic AI-driven analysis. They also send the sensor data to the cloud via the Internet.The third layer, the cloud layer, is responsible for receiving, storing, analyzing, and providing secure access to data from the gateways. Analysis in this layer may involve processing the data to identify any changes in the patient's health status, which can then be reported to doctors or patients for further action. The key generation server in this layer generates IDs and keys for various system nodes, enabling remote management and control of sensor access.The fourth layer is the operational layer, where the processed data is presented to doctors and patients to monitor health status. This layer also includes recommendations and actions suggested by doctors based on the patient's health condition.

However, the security of IoMT in healthcare systems faces significant challenges. A unified network allows hackers to maliciously access devices within the network, and with the increasing interconnectivity and number of medical devices, the risk of cyberattacks is also rising. Medical devices and wearable devices come in a wide variety of types, often produced by different manufacturers. These devices are not only key com-

ponents of smart healthcare systems but can also be used for malicious purposes, such as sending phishing and spam emails. Due to the lack of encryption for wireless keys, smart healthcare devices are often targeted for DDoS attacks, especially since these devices immediately initiate smart solutions when activated, such as accessing patient medical records and automatically updating false medical reports. Therefore, enhancing the security measures of smart healthcare devices, ensuring encrypted wireless communication and robust key management, is crucial to preventing these potential malicious attacks. Additionally, healthcare providers and patients need to remain vigilant, regularly check the security settings of their devices, and promptly update systems and software to minimize the risk of being targeted.[11]. These challenges stem from the centralized structure of IoMT systems. As the IoMT revolution progresses, privacy issues are also increasing, including record forgery and manipulation, device interference, and the illegal penetration of IoMT devices through attacks on servers and gateway networks[12].

These issues can be addressed through the deployment of blockchain-based systems and unified ”cloud-like” computer networks[12].As an open and transparent data protection mechanism, the advancement of blockchain technology opens up new avenues to address critical issues of privacy, security, and ethics in a smart healthcare system.[13]. As the cornerstone of the cybersecurity architecture for various smart healthcare technologies, blockchain has achieved significant success in areas such as patient record access control and information distribution[14].

This paper presents a proof-of-concept for a patient monitoring system that uses IoMT to capture data and blockchain technology to store it.

## 1.2 Literature Review

Satoshi Nakamoto established blockchain technology in 2008. This technology links data blocks in chronological order to form a chain-like data structure, ensuring the immutability and authenticity of data through cryptographic means. The core characteristics of blockchain lie in its tamper-resistant properties, decentralized operation mode, and high transparency. These characteristics collectively form the foundation of blockchain technology[15]. These three key principles have not only effectively propelled the development of the technology but also paved the way for its application in virtual currency and other extensive technical fields. When Bitcoin was born, it garnered much attention and controversy. Some believe that blockchain is a disruptive technological invention following steam engines, electricity, and the internet, which will completely change the way human society transfers value and even bring about a new round of technological revolution[16, 17]. In contrast, some opponents view Bitcoin and blockchain as a scam or have concerns about its future. Although opinions on the long-term future of cryptocurrencies may differ, several key applications have been identified by various sources. A report from the UK government states that blockchain has the potential to ”revolutionize our financial markets, supply chains, consumer-to-business and business-to-business services, as well as public registers.” The range of potential applications extends from asset registration and the transfer of hard asset ownership to the secure recording of intangible assets[18]. Within blockchain, each node has its distributed ledger to store the historical records of transactions. Blockchain can be utilized to achieve real-time application authentication, authorization, accountability, security, integrity, confidentiality, and immutability, which may not be effectively provided by centralized systems in smart community environments[19].

In recent years, the term ”blockchain” has gained popularity in smart healthcare, with some research publications exploring how blockchain technology can be applied to the field. Studies have also shown that blockchain technology has the potential to improve the healthcare system in various ways. For example,

Omar et al[20]. concluded that blockchain technology can be used to enhance data security, reduce the time and cost associated with exchanging health data, and improve healthcare outcomes. S. Aggarwal et al[21]. explored some applications of blockchain in several aspects of healthcare, such as transaction consolidation, home healthcare, and investment allocation. Other investigations indicated that blockchain technology can be used to improve patient access to health records, streamline health claim processes, and enable secure data sharing between providers and patients[22]. Yin.C[12] provided a comprehensive analysis of many blockchain applications in P2P resource-sharing networks. The report offers in-depth knowledge about the deployment and functionality of multiple smart home networks, including security issues in smart grids, big data analysis, artificial intelligence, and payment services. Gao Lei Li et al. proposed a user-based blockchain structure to ensure secure information communication in the Internet of Things (IoT)[23]. Zhen Yu Zhou et al[24]. researched various blockchain technologies, scheduled surveys, and decentralized computing to re-engineer control over specific vehicles and enhance their effectiveness. Ming Li et al[25]. adopted Attribute-Based Encryption (ABE) technology to design a patient-centric encryption framework and corresponding privacy-preserving mechanisms for each patient's Personal Health Record (PHR) files. Under this framework, patients' privacy is effectively safeguarded. Azaria et al[26]. proposed a novel decentralized record management system called MedRec, where patients can obtain comprehensive immutable logs and easily access their medical information across treatment locations and providers. Omar et al[27]. proposed a patient-centric healthcare data management system integrating blockchain technology called MediBchain to maintain the privacy of healthcare data. Jie Xu et al[28]. introduced Healthchain, a blockchain-based data privacy scheme suitable for large-scale health data, where users can add or revoke authorized doctors. In this scheme, doctors' diagnoses and IoT data cannot be altered or deleted. Yue Xiao et al[29]. proposed a healthcare data gateway (HGD) application that integrates blockchain technology, allowing patients to easily and securely own, share, and control their personal data without security issues. By researching the underlying code of blockchain, to establish secure data transmission between nodes, Dilawar et al[30]. proposed a new security architecture based on IoMT.

Blockchain possesses characteristics such as decentralization and immutability, which can establish a trusted service platform among various participants. Leveraging blockchain technology to achieve secure and flexible sharing of health data while preventing the leakage of private information is highly promising. Currently, researchers have combined blockchain with access control technologies to facilitate the sharing of user health data[31–33]. For example, Lei Zhang et al[34]. proposed a blockchain-based model for controlled sharing of electronic medical records, integrating attribute-based access control with multi-keyword encryption schemes for encrypting electronic medical records. Users can define access policies according to their needs, achieving controlled sharing of health data. Wang et al[35]. combined attribute-based access policies with user identity encryption to encrypt medical data, enabling data sharing and traceability. Zhao et al[36]. proposed a blockchain-based access control model for electronic medical records, utilizing role-based access control strategies to manage access to electronic medical records and ensure the privacy of user medical data. Some researchers are also dedicated to sharing and protecting user medical health data through shared keys. For instance, Xu et al[28]. proposed a blockchain-based privacy protection scheme for medical health data, where the hash values of user health data are stored on the user chain, and the hash values of diagnostic information issued by authorized doctors are stored on the doctor chain. Data sharing and authorization/revocation of doctors are achieved through shared encryption keys of health data. AlOmar et al[27]. proposed a user-centric privacy protection scheme for medical health data called MediBchain, where users store encrypted health data on a permissioned blockchain, and access to data on the blockchain is only allowed with the correct key. Additionally, some researchers focus on using blockchain as a trusted entity and employing smart contracts running on the blockchain to achieve health data sharing. For example, Ruijin Wang et al[37]. pro-

posed a medical blockchain privacy data sharing model based on ring signatures, utilizing smart contracts on the blockchain for data sharing and employing ring signature mechanisms to protect user identity privacy. Yameeng Bai et al[38]. proposed a blockchain-based secure storage model for electronic medical records, using smart contracts to achieve sharing of electronic medical records and optimizing the consensus mechanism to reduce the time for new blocks to be added to the blockchain.

Internet of Things (IoT) is a technology system that utilizes various information sensors, radio frequency identification (RFID) technology, GPS, infrared sensors, laser scanners, and other devices and techniques to perform real-time data collection on objects or processes that require monitoring, connection, and interaction. This includes collecting various necessary information such as sound, light, heat, electricity, and position. Through possible network access, it enables ubiquitous connections between objects and between objects and humans, and allows for intelligent perception, identification, and management.

The concept of the Internet of Things (IoT) first emerged in Bill Gates' 1995 book "The Road Ahead." In this book, Bill Gates mentioned the IoT concept, but at that time, due to the limitations in wireless networks, hardware, and sensor equipment, it did not gain much attention from the public. The idea of the Internet of Things originated initially from the Auto-IDLabs established by the Massachusetts Institute of Technology (MIT) in 1999, which proposed a networked RFID (Radio Frequency Identification) system. This system sought to connect all items to the internet through RFID and other information sensing devices to achieve intelligent identification and management. The early IoT was proposed in the context of logistics systems, using RFID technology as a replacement for barcode identification to achieve intelligent management of logistics systems. With the development of technology and applications, the connotation of IoT has undergone significant changes. On November 17, 2005, at the World Summit on the Information Society (WSIS) held in Tunisia, the International Telecommunication Union (ITU) released the "ITU Internet Report 2005: The Internet of Things," officially introducing the concept of "Internet of Things." The report pointed out that an era of ubiquitous "Internet of Things" communication was approaching, where all objects in the world, from tires to toothbrushes, houses to tissues, could actively exchange information through the internet. Technologies such as RFID, sensor technology, nanotechnology, and intelligent embedded technology would receive more extensive application and attention. The ITU reported that we are standing at the edge of a new era of communication, where the goals of information and communication technologies (ICT) have evolved from meeting the communication needs between people to enabling connections between people and things, and between things themselves. The era of ubiquitous IoT communication is approaching. IoT provides us with a new dimension of communication in the world of information and communication technologies, expanding from connecting anyone at any time, any place, to connecting any item, thus forming the internet of things.

To integrate fog computing and the Internet of Healthcare Things (IoHT), Quy[39] and his team proposed the Fog-IoHT framework. Addressing the adoption of blockchain applications and the execution cost issues in an IoMT environment, Lakhan[40] and colleagues presented a blockchain-fog-cloud-based bionic robotics approach. Compared to existing methods, this proposed model reduced execution and mining costs by 50% and 40%, respectively. To process medical data more rapidly and alleviate the storage and processing burden on cloud servers, Mayer[41] and his team suggested implementing a blockchain-assisted fog computing system in the healthcare sector. By applying fog computing in the network closest to user devices, they also reduced latency, resource utilization, and response time. Leveraging the advantages of blockchain technology, Singh[42] and his colleagues introduced a secure, distributed fog computing network architecture for IoVT, aiming to address current issues and challenges. Shahid[43] and his team proposed a framework called IoTNetWar, based on fog, with the goal of reducing latency while maintaining Quality of Service (QoS). Za-abar[44] and his colleagues presented HealthBlock, an integrated, interoperable, and scalable healthcare

framework that encompasses issues of security, privacy, reliability, and scalability. Combining the Internet of Things, blockchain, and fog computing, Kamruzzaman[45] and his team aimed to enhance the efficiency of healthcare systems. By implementing a blockchain-integrated ecosystem, they achieved security, privacy, immutability, and decentralization of medical data. Baniata[46] and his colleagues proposed a blockchain-based fog healthcare infrastructure to address the issues facing today's healthcare systems. Shukla[47] and his team developed a three-tier blockchain and fog computing integrated architecture for the identification of IoT devices, authentication of medical data, and verification of medical systems. Memon[48] and his colleagues introduced a DualFog-IoT-based blockchain architecture that provides three possible configuration filters to handle incoming requests: non-real-time, real-time, and delay-tolerant use cases. The researchers identified two sub-layers within the fog layer: fog miner clusters and fog-cloud miner clusters. To simplify access between customers and distributors, Mallick[49] and his team presented an analytical model based on a priority reservation queue for a blockchain-supported smart grid system. Khaydaraliev[50] and his colleagues proposed a decentralized device-to-device communication solution based on the Internet of Things, blockchain technology, and fog computing. The researchers utilized smart contracts to monitor the activity of items.

## 1.3 Research Aims and Contribution

This paper first describes and analyzes the concept and significance of Healthcare 5.0, its development, and the challenges it currently faces. It then proceeds to analyze the current research status both domestically and internationally, including the research status of the Internet of Medical Things (IoMT) and blockchain technology. Regarding the research status of the IoMT, it focuses on the hardware design of the IoMT and the privacy protection of medical data during transmission and in the process of access and sharing. Through the analysis of existing schemes, it is found that most methods for data privacy protection have issues such as data singularity, privacy security, and efficiency. Blockchain technology, with its decentralized, tamper-proof, and traceable characteristics, can effectively address these problems. Therefore, this paper proposes a solution for data security and privacy protection in the application scenarios of the IoMT by combining a consortium blockchain platform with basic human body detection sensors. The specific research plan is as follows:

(1) Currently, the main challenges in sharing medical health data in IoT-based scenarios are that a unified network makes it possible for hackers to maliciously access devices within the network. Additionally, the wide variety of medical devices and wearable devices makes it difficult to protect against malicious attacks. Furthermore, unencrypted wireless keys often make smart healthcare devices targets for DDoS attacks. These challenges primarily stem from the centralized IoMT system architecture, which can lead to privacy issues such as record forgery and manipulation, device interference, and unauthorized access to IoMT devices through attacks on servers and gateways.To address this issue, our system attempts to solve these problems by deploying a blockchain-based system and a unified "cloud-like" computer network.

(2) Regarding the issue of potential privacy leaks arising from the transparency of blockchain, such as personal health information and doctor-patient relationships, our system opts for a private blockchain. Unlike public blockchains, participants in a private blockchain need to meet predefined conditions or permissions to join the network. Only specific parties (such as internal company employees and partners) are allowed to read, write, and verify data. Only verified nodes can participate in the data management, storage, and distribution of the blockchain. This approach better addresses the issue of privacy leaks.

The proposed system integrates IoT devices and blockchain technology into a cohesive framework for secure

health monitoring. Its core components include:

(1) IoT sensing modules (e.g., ECG sensors and Raspberry Pi controllers) for real-time data collection;

(2) A hybrid encryption module combining ECC and AES to generate dynamic digital envelopes;

(3) A private blockchain network built on FISCO BCOS for tamper-proof data storage and smart contract-driven access control;

(4) A Vue.js/Spring Boot-based interface enabling multi-role interaction and visual analytics.

This architecture collectively addresses Healthcare 5.0 requirements by ensuring end-to-end data integrity, enabling permissioned sharing among stakeholders (patients, doctors, administrators), and maintaining compliance with medical privacy regulations through decentralized governance.

## 1.4 Structure of the Thesis

The organization structure of this work are illustrated in Figure 1.1.

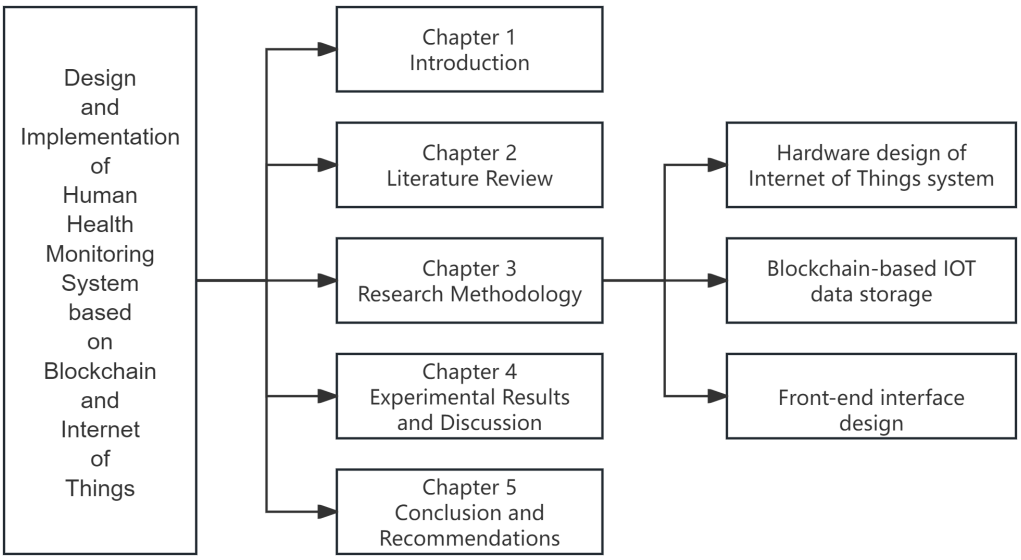


Figure 1.1: Paper Structure

This paper consists of five chapters, with the specific content arrangement for each chapter as follows:

Chapter 1 Introduction. This chapter introduces the research background of this paper, Literature Review, Research Aims and Contribution and the structure of the thesis.

Chapter 2 Relevant Background Knowledge. This chapter explains the blockchain knowledge, Internet of Things knowledge, and basic tools that will be involved in the proposed solution.

Chapter 3 Research Methodology. This chapter describes the detailed design of the IoT module, blockchain module, and front-end module of the system.

Chapter 4 Experimental Results and Discussion. This chapter presents the system’s performance analysis and experimental results analysis.

Chapter 5 Conclusion and Recommendations. This chapter summarizes the main research work of this paper, discusses the limitations encountered in this work, and outlines future research directions.

# 2 Related Technology Introduction

## 2.1 Internet of Things

### 2.1.1 Definition and features of medical IOT

The concept of the Internet of Things (IoT) was proposed in the year 1999. IoT is an ”internet where things are interconnected,” also known as the Internet of Objects. Through information sensing devices such as Radio Frequency Identification (RFID), infrared sensors, Global Positioning System (GPS), and others, IoT connects any object to the internet according to agreed protocols, enabling information exchange and communication. This leads to the intelligent identification, positioning, tracking, monitoring, and management of objects. There are two levels of meaning to this.First, the core and foundation of IoT remain the internet, serving as an extension and expansion of the internet. Second, its user end extends and expands to any item-to-item connections, enabling information exchange and communication between them. IoT is a network based on the internet, traditional telecommunications networks, and other information carriers, allowing all independently addressable physical objects to achieve interconnectivity.The core and foundation of IoT remain the internet, serving as an extension and expansion of the internet.Its user end extends and expands to any item-to-item connections, enabling information exchange and communication between them. IoT is a network based on the internet, traditional telecommunications networks, and other information carriers, allowing all independently addressable physical objects to achieve interconnectivity.

The characteristics of IoT are illustrated in the figure 2.1. IoT not only provides connectivity for sensors but also has the capability for intelligent processing, enabling intelligent control of objects. By combining sensors with intelligent processing, IoT utilizes various intelligent technologies such as cloud computing and pattern recognition to expand its application areas. It analyzes, processes, and extracts useful data from the massive amount of information obtained from sensors to meet the diverse needs of different users. IoT fully applies new technologies such as RFID, sensor technology, and nanotechnology in various industries, connecting various objects and transmitting the collected real-time dynamic information to the computing and processing center via wireless networks for aggregation, analysis, and processing. This allows centralized management and control of machines, equipment, and personnel using central computers, optimizing production and life through more precise and dynamic methods. Ultimately, it achieves the organic integration and harmonious coexistence of human society and the material world.

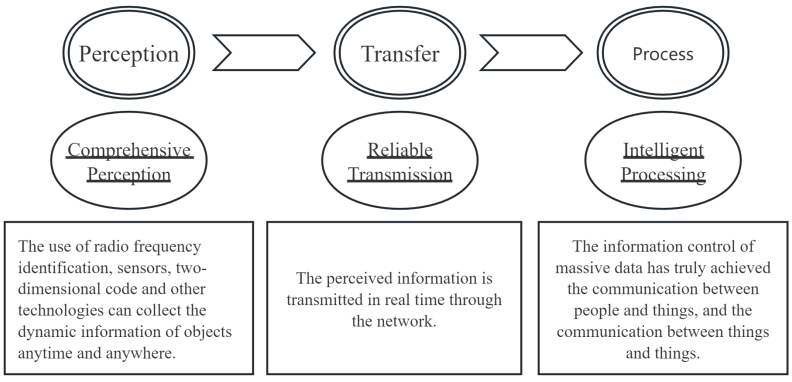


Figure 2.1: Features of the Internet of Things

### 2.1.2 Raspberry Pi 4B

The Raspberry Pi is a small, low-cost single-board computer developed by the Raspberry Pi Foundation in the United Kingdom. It aims to promote computer science education, particularly in developing countries and schools. The Raspberry Pi is widely used not only in the educational field but also in various other areas such as DIY projects, smart homes, robotics, and media centers, where it demonstrates powerful functionality and flexibility.

Our system uses the Raspberry Pi 4B as the main controller for the IoT module. The Raspberry Pi 4B is a compact and low-power single-board computer (SBC) introduced by the Raspberry Pi Foundation. It is an upgraded version of the original Raspberry Pi 3, offering higher processing performance while maintaining its small size and low power consumption. The physical appearance is shown in Figure 2.2.

Figure 2.2: Raspberry Pi 4B

The Raspberry Pi 4B uses the Broadcom BCM2711 as its main control chip, which features a quad - core 64 - bit Cortex - A72 processor. The Raspberry Pi 4B has a higher performance processor architecture compared to the Raspberry Pi 3, with a base clock frequency that can reach up to 1.5GHz. This represents a significant improvement in processing power compared to previous Raspberry Pi models.

Compared to the lower - spec models, the Raspberry Pi 4B performs much better in multitasking and computationally intensive tasks. It comes with different RAM options, including 1GB, 2GB, 4GB, and 8GB, which can fully meet the needs of various projects, from small embedded projects to large - scale application development.

In terms of wireless connectivity, it includes 802.11ac WiFi, supporting both 2.4GHz and 5GHz wireless networks, as well as Bluetooth 5.0, which supports Bluetooth Low Energy (BLE) and has a longer range and higher data transfer rate.

For connectivity, it features a 40 - pin GPIO interface, which is compatible with the GPIO pins on standard Raspberry Pi models, allowing it to connect to external hardware modules or sensors. It also has two USB 3.0 ports and two USB 2.0 ports, Gigabit Ethernet, and an HDMI port for high - definition video output.

Its performance meets the requirements of our system, and although it is relatively more expensive than some lower - end models, considering its powerful performance and rich features, it offers excellent value for money.

### 2.1.3 Embedded Operating System

An embedded operating system is a specialized operating system designed for embedded applications, featuring customizable functionalities, excellent power consumption control, and stable reliability. Compared to a barebone program, using an embedded operating system allows for more efficient management of system resources, facilitates the implementation and maintenance of applications, and accelerates project development speed. With technological advancements, the capabilities of embedded devices have become increasingly powerful, and the implementation logic has grown more complex, particularly given the rapid development of artificial intelligence and the Internet of Things, which has introduced new requirements for embedded operating systems.

The operating system used in this paper is Raspberry Pi OS, an official operating system developed by Raspberry Pi. Raspberry Pi OS is a Unix-like operating system based on the Debian Linux distribution, specifically designed for the Raspberry Pi series of compact single-board computers, and optimized for Raspberry Pi hardware. Raspberry Pi OS supports over 35,000 Debian software packages.

### 2.1.4 UART Serial port protocol

UART stands for Universal Asynchronous Receiver-Transmitter. The connection method is shown in Figure 2.3, where the communication between the two parties is achieved through three wires: TX, RX, and GND. When communicating via UART, both parties need to agree on the same transmission rate, start bit, stop bit, and parity bit. The transmission rate is expressed using baud rate, with common baud rates being 4800 bps, 9600 bps, etc., which means transmitting 48,00 bits or 96,000 bits of data per second. The GND wire provides a reference voltage level for both the transmitting and receiving parties. Data is transmitted bit by bit through the TX signal wire according to the set baud rate. When sending a binary ”1,” the signal line is set to a high voltage level; when sending a ”0,” the signal line is set to a low voltage level. RX is used to receive data, and the system samples the voltage levels on the data line bit by bit to distinguish between high and low voltage levels. When transmitting a byte of data, the signal line needs to undergo eight high-low voltage level transitions and samplings. In summary, the UART serial protocol is an asynchronous, serial, full-duplex communication protocol.

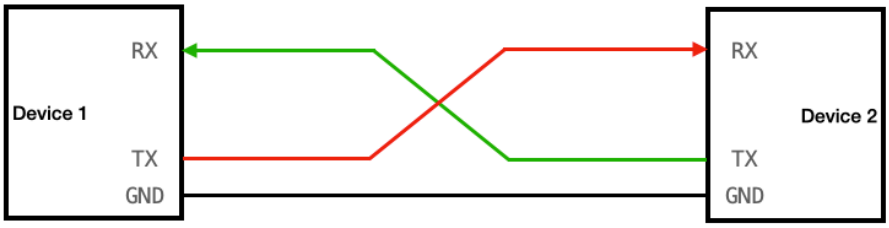


Figure 2.3: UART connection mode

The following is an introduction to some proper terms for UART:

- Synchronous Transmission: Synchronous transmission refers to the method where devices send data signals at a fixed clock rate, requiring the connection of two types of wires: the clock line and the data transmission line.
- Asynchronous Transmission: Asynchronous transmission refers to the method where each character is

independently formed into a frame for transmission. A continuous string of characters is also encapsulated into continuous independent frames for transmission, and the intervals between characters can be arbitrary. Asynchronous transmission does not have a clock line and only requires a data line, but both parties need to adhere to the same agreement.

- Simplex: A communication method where data can only be transmitted in one direction. It can either act as a sending device or a receiving device.
- Half-Duplex: Data can be transmitted in both directions, but not simultaneously. Within the same time period, both parties in communication can only perform sending or receiving operations, not both at the same time.
- Full Duplex: A communication method where data can be transmitted simultaneously in both directions.

When a character is transmitted via UART serial port, its timing sequence is shown in Figure 2.4. The data line is high during idle periods. When data transmission begins, the transmitter pulls the data line low to inform the receiver that data transmission is starting. The transmitter then begins to send the actual data portion, which is typically 8 bits. The data is sent starting with the least significant bit and ending with the most significant bit. After the data is sent, a parity bit can be added, with parity methods being either odd or even. Finally, there is the stop bit, which can be 1, 1.5, or 2. This bit not only indicates the completion of a character transmission but also serves to synchronize the clocks between the transmitter and receiver.

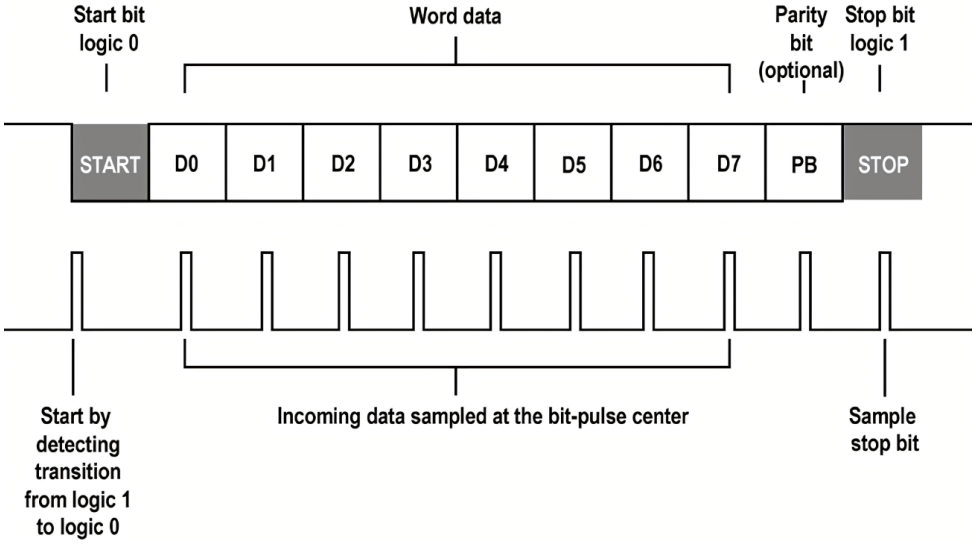


Figure 2.4: UART timing diagram

Source: *A UART frame*. electric impDevcenter. http://bit.ly/3S4p0Xq. Accessed: 2024-11-01.

### 2.1.5 I2C Serial port protocol

I2C is a serial communication interface that features a bidirectional two-wire synchronous serial bus, typically consisting of two lines—the SDA (Serial Data Line) and SCL (Serial Clock Line) along with pull-up resistors. These lines are used in projects requiring the collaboration of many different components (such as sensors, pins, extensions, and drivers), as they can connect up to 128 devices to the mainboard while maintaining clear communication paths. It is used for connecting various low-speed devices, such as microcontrollers, EEPROMs, A/D and D/A converters, etc. Unlike UART or SPI, the I2C bus drivers are open-drain, which prevents bus contention and eliminates the chance of driver damage. Additionally, each signal line in I2C

contains a pull-up resistor, which restores the signal to the high level of the line when no device pulls it low. Data is transmitted in bytes, with each byte followed by a 1-bit handshake signal as the ACK/NACK bit (acknowledge/not acknowledge).

**Steps for I2C communication**

(1) The master device will generate a start signal, signaling to other devices to begin listening to the bus and prepare to receive data. (SCL high, SDA transitions from high to low.) When the start signal condition is sent, the bus will enter a busy state in which the current data transmission is restricted to the selected master and slave device. Only after a stop condition is generated will the bus be released and return to an idle mode.

(2) The primary device sends an 8-bit address to each connected device, with 7 bits representing the device address and 1 bit indicating the read/write direction. This direction bit also specifies the flow of the subsequent data transfer. If the bit is 0, the primary device will write data to the slave device. If the bit is 1, the primary device will read data from the slave device.

(3) Each slave device will compare the address transmitted by the master to its own address. The slave device that matches the address successfully will respond with an ACK bit by pulling the SDA line low. The ACK bit is sent by the receiving device after each frame to indicate to the sender whether the data frame was received successfully (ACK) or not (NACK).

(4) After receiving the acknowledgment signal from the slave device, the master device begins to send or receive data. Figure 2.5 is a diagram of the process where the master reads data from a designated device.

(5) After each data frame is sent, the receiving device sends another ACK bit back to the sender to confirm that the frame was successfully received. The sender then continues to transmit the next data frame, and this process repeats.

(6) Upon completion of the data transmission, the master device will issue a stop signal to the other devices, releasing the bus and allowing it to enter an idle state.

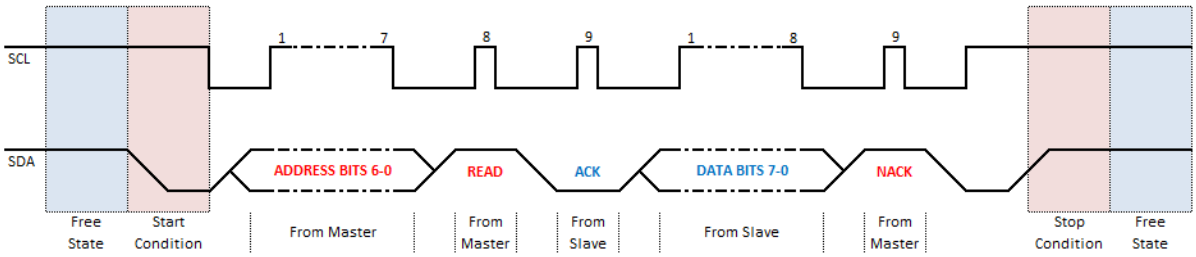


Figure 2.5: I2C communication sequence diagram
Source: *I2C timing diagram*（2024）. elecfans. https://bit.ly/4drBz91. Accessed: 2024-11-01.

### 2.1.6 MQTT protocol

MQTT (Message Queuing Telemetry Transport) is a messaging protocol proposed by IBM in 1999. It features lightweight and high bandwidth efficiency, making it highly suitable for environments with poor network conditions[51], such as in wireless sensor networks, power data monitoring, and smart homes. Over the years, MQTT has become a widely used IoT communication protocol.

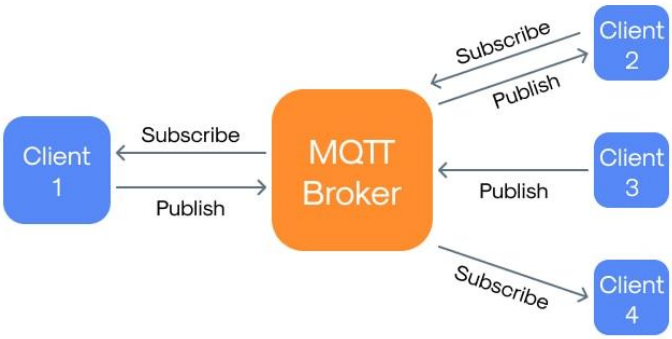


Figure 2.6: MQTT operating principle

Source: *MQTT Protocol*（2025）. wallarm. https://bit.ly/4j6UkzZ. Accessed: 2024-11-01.

The MQTT protocol enables message delivery between publishers and subscribers through a message broker. Firstly, both publishers and subscribers send connection requests to the broker server. After successful verification and connection, the publisher sends data to the message broker according to the topic. The message broker then pushes the data to all subscribers who have subscribed to the corresponding topic. Based on this mechanism, it is possible to decouple subscribers from publishers[52].

## 2.2 Blockchain Technology

### 2.2.1 Principle of blockchain

Blockchain is not just a single technology, but a combination of multiple technologies, including P2P networks[53], cryptography, and consensus mechanisms[54]. Due to its characteristics of decentralization, immutability, traceability, and transparency, it is widely applied in various fields such as healthcare[55], railway transportation[56]and supply chain management[57].

Decentralization in blockchain means that it no longer relies on a central node. All nodes participating in the network have the same capabilities, and data can be stored, recorded, and updated in a distributed manner[58].Immutability refers to the fact that any record once added to the blockchain will be permanently retained. Data records on the blockchain are stored in individual blocks, and to modify a record stored on the blockchain, one would need to control more than 50% of the nodes, which is extremely difficult to achieve in reality.Traceability is due to the fact that the blockchain is composed of interconnected blocks, each of which contains a timestamp that records the time it was added to the blockchain. Additionally, each transaction in a block is cryptographically linked to the transactions in the previous and subsequent blocks, making every transaction in the blockchain traceable.Transparency means that transactions on the blockchain are open to the public. Anyone can participate in the blockchain network, view transaction information on the blockchain, and check and trace the transaction data.

Blockchain can be divided from the bottom up into the data layer, network layer, consensus layer, contract layer, and application layer, as shown in Figure 2.7. Among these, the data layer is the lowest layer of the blockchain, responsible for connecting individual blocks.

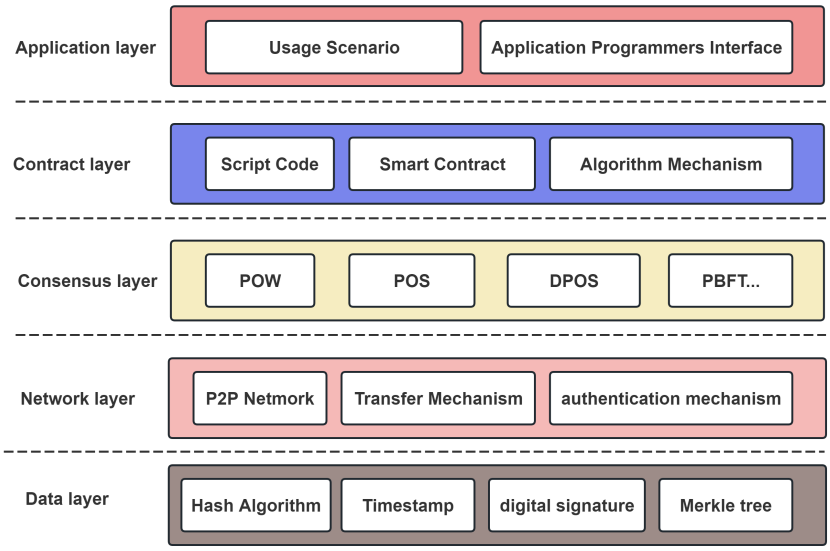


Figure 2.7: /Blockchain architecture

Except for the genesis block, each block stores the hash value of the previous block. As shown in Figure 2.8, a block is composed of a block header and a block body. The block header contains the hash value of the previous block, a nonce, a timestamp, the root of the Merkle tree, a version number, and other information. The block body is used to store transactions, which generally include the sender's address, the recipient's address, the transaction cost, and the transaction content. The transactions stored in the block body are aggregated into a Merkle tree through hash operations, and the root of the Merkle tree is stored in the block header.

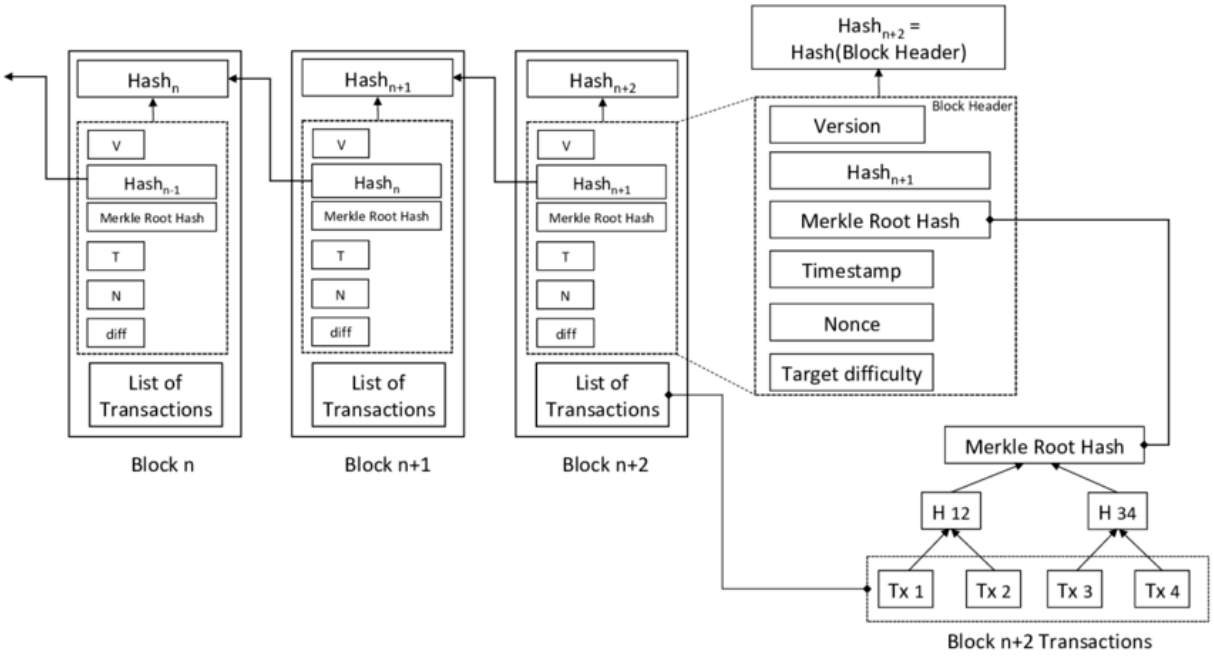


Figure 2.8: Blockchain data structure

The network layer primarily includes mechanisms for data propagation, peer-to-peer (P2P) networks, and validation. In a blockchain network, all nodes operate at an equal level, interconnected without a central

node managing them. Each node can receive information and respond accordingly. Additionally, when a new block is created, it is broadcast to all nodes in the network, which then verify its legitimacy. When more than 50% of the nodes validate the new block, it is successfully added to the blockchain. If a malicious attacker attempts to control the addition of new blocks, they would need to control more than 50% of the nodes in the blockchain network. This process requires significant computational power and is practically difficult to achieve in real-world scenarios. Consequently, this further enhances the security of the blockchain network system.

Common consensus protocols found in the consensus layer include Proof of Work (PoW), Proof of Stake (PoS), and Delegated Proof of Stake (DPoS). As a core technology in blockchain, consensus protocols primarily aim to ensure that all nodes in the network agree on a common state, thereby enhancing the security of the blockchain network.

The contract layer encapsulates various script codes, algorithm mechanisms, and smart contracts within a blockchain network. A smart contract is a piece of code running on the blockchain that can execute transactions based on predefined conditions. Moreover, since the data records stored on the blockchain are both transparent and immutable, the smart contracts running on the blockchain inherently possess these same characteristics.

The application layer encapsulates various application scenarios and cases of blockchain technology. This layer typically involves the development of decentralized applications (DApps). Additionally, it provides appropriate interfaces for various application scenarios, users, or platforms to facilitate interaction and functionality.

### 2.2.2 Classification of blockchain

Since its inception, blockchain technology has been broadly categorized into three types: public blockchain, consortium blockchain (or ”federated blockchain”), and private blockchain. Among these, the public blockchain is a fully decentralized form of blockchain. Any group or individual in the world can issue transactions on a public blockchain network and participate in the consensus process. However, when there are too many nodes in the public blockchain network, the transaction processing speed decreases. This is because every node in the network needs to verify the transaction, and only after all nodes reach a consensus can the transaction be added to the blockchain.

The consortium blockchain, or federated blockchain, is a type of blockchain that lies between the public blockchain and the private blockchain and has a relatively lower level of decentralization. When the state is fully public and allows any individual or organization to join, it becomes a public blockchain. Conversely, if all nodes within the consortium blockchain network are controlled by a central node, the consortium blockchain becomes a private blockchain. Consortia blockchains are established as alliances between companies or organizations. Companies or organizations within the alliance network have the authority to read and maintain data on the consortium blockchain. This type of blockchain is primarily applied in fields such as banking, insurance, and securities.

Private blockchains are often used within individual or corporate environments, where only authorized nodes can participate and view the information on the blockchain, making them the least decentralized among the three types. However, private blockchains offer very fast transaction processing speeds. Unlike public blockchains, which require more than 50% of the nodes in the network to validate a transaction before it can be added to the blockchain, private blockchains only need verification by the central node within the network before a transaction can be added to the blockchain. Due to these characteristics, private blockchains

are commonly used in financial institutions, within enterprises, and by government departments.

Below is a comparison table illustrating the characteristics of Public Blockchain, Consortium Blockchain (or ”Federated Blockchain” ), and Private Blockchain:

Table 2.1: Classification of blockchain

| | Public Blockchain | Consortium Blockchain | Private Blockchain |
|---|---|---|---|
| Participant | Anyone | Member in the Alliance | Individual entity within a company or corporation |
| Accounting Party | All Participants | Determined through negotiation among members in the Alliance | Custom Centralized Accounting Node |
| Degree of Decentralization | Fully Decentralized | Partially Decentralized | Least Decentralized |
| Notable Features | Self-established Credit | Efficiency and Cost Optimization | Transparency and Traceability |
| Application Scenarios | Virtual Currency | Payment, Settlement | Audit, Issuance |

### 2.2.3 Classification of consensus mechanisms

The blockchain consensus mechanism is one of the crucial mechanisms ensuring the security and reliability of the blockchain. It achieves this through algorithms and agreements among network nodes to ensure consistency in data and transactions across all nodes on the blockchain, thereby preventing double-spending and other malicious activities. The consensus mechanism prevents nodes from tampering with data or engaging in other malicious behaviors, making the blockchain more secure and reliable. The implementation of the consensus mechanism requires cooperation among multiple nodes, enhancing the decentralization of the blockchain. Under the influence of the consensus mechanism, nodes do not need to trust any centralized institution, making the blockchain more decentralized and democratic. The consensus mechanism can be applied in digital currencies, smart contracts, supply chain management, healthcare record management, and other fields, providing reliable technical support for their development and applications. The implementation of the consensus mechanism relies on digital technologies such as computers and networks, thus driving the advancement of the digital economy. Continuous optimization and innovation in the consensus mechanism will provide more reliable and secure technical support for the development of the digital economy.

The workflow of the consensus mechanism includes the following steps:

- Submitting a transaction: The user submits a transaction request within the blockchain network, which includes details such as the transaction amount, the sender’s address, the recipient’s address, and other relevant information.
- Validating the transaction: Nodes verify the transaction, including checking the legitimacy of the transaction and the balance sufficiency, etc. If the validation is successful, the node broadcasts the transaction to the entire network.
- Selecting the bookkeeping node: Different consensus mechanisms employ various methods to choose the bookkeeping node. For example, in the Proof of Work (PoW) algorithm, nodes compete for the bookkeeping right by calculating complex hash functions, while in the Proof of Stake (PoS) algorithm, the selection is based on the number of digital currencies held by the nodes.

- Generating a new block: The bookkeeping node confirms the transactions and packages them into a new block, then broadcasts the new block to the entire network.
- Validating the new block: Other nodes will verify the new block, including checking the legitimacy of the new block and the consistency of the data, etc. If the validation is successful, the node will add the new block to its own blockchain.
- Updating the blockchain: Once the new block is validated and confirmed, the blockchain is updated, ensuring that all nodes have the same blockchain, thereby maintaining data consistency and reliability.

Currently, the mainstream consensus algorithms in the blockchain industry include Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), and Practical Byzantine Fault Tolerance (PBFT) used in Hyperledger, among others.Below, we will explain the typical representatives of these consensus mechanisms:

1. Proof of Work(POW)
   Proof of Work (PoW) was first mentioned in the blockchain industry in Satoshi Nakamoto's white paper "A Peer-to-Peer Electronic Cash System" in 2008, and then applied to Bitcoin in 2009. The design concept of this consensus algorithm is that in the entire distributed system, each node provides computational power (referred to as hash power) to the system. Through a competition mechanism, the node that performs the most outstanding computational work is rewarded by the system, thereby completing the distribution of newly generated currency. PoW primarily ensures the consensus security of the entire network through the computational power of CPUs.
2. Proof of Stake(POS)
   Proof of Stake (PoS) is a consensus mechanism that first appeared in the white paper of Peercoin. Its core idea is to use the number and duration of currency holdings by token holders as the capital to be selected as consensus nodes.
3. Delegated Proof of Stake(DPOS)
   The Delegated Proof of Stake (DPoS) mechanism was first proposed by Daniel Larimer, and BitShares was the first distributed ledger to propose and implement DPoS. Simply put, the working principle of DPoS is similar to board voting, giving token holders a key to unlock the corresponding voting rights of their holdings, rather than providing them with a shovel for mining. DPoS introduces the concept of witnesses who can generate blocks, and every token holder can vote to elect witnesses. Candidates who receive the top N (usually 101) votes can be elected as witnesses. The list of witness candidates is updated once every maintenance period (usually 1 day).
4. Pratical Byzantine Fault Tolerance(PBFT)
   PBFT (Practical Byzantine Fault Tolerance), which stands for Practical Byzantine Fault Tolerance algorithm, is one of the most commonly used BFT algorithms. It was originally proposed by Miguel Castro and Barbara Liskov in 1999, addressing the inefficiency issue of the original Byzantine Fault Tolerance algorithm and reducing the algorithmic complexity from exponential to polynomial level.

### 2.2.4 Hash Function

Hash function is a crucial component of cryptography, widely applied in various fields such as data integrity verification and digital signatures. A hash function converts input data of arbitrary length into output data of fixed length. It is a one-way function, meaning that while it is possible to compute the hash value from the input data, it is computationally infeasible to deduce the input data from the hash value. Additionally,

hash functions possess collision resistance, ensuring that two different input data will produce different hash values when processed. Conversely, for a given hash function, it is impossible for two different inputs to produce the same hash value. Common hash functions include MD5 and SHA.

### 2.2.5 Symmetric cryptography

Symmetric cryptography, also known as symmetric-key cryptosystem, refers to the process where both the sender and receiver of a message use the same key for both encryption and decryption of the data. This key must remain confidential and should not be disclosed. The sender uses this key to encrypt the data and then transmits the ciphertext to the receiver. Upon receiving the ciphertext from the sender, the receiver uses the same key to decrypt it. Symmetric-key cryptosystems are frequently applied in data sharing scenarios, where both parties in the data sharing process typically hold the same symmetric key to access the shared data.

### 2.2.6 Asymmetric encryption

Asymmetric encryption refers to the process of encryption and decryption using different keys to carry out encryption, this system mainly involves the asymmetric encryption is public-key cryptography.

In public-key cryptography, each user is assigned a pair of public and private keys, denoted as (PK, SK). The user publicly shares their public key PK and securely keeps their private key SK. In the process of data sharing using symmetric keys, both parties involved use the same key for encryption and decryption. If this key is compromised by an attacker, the shared information can be decrypted and accessed by the attacker, leading to significant data leakage. In public-key cryptography, the sender can encrypt the data intended for sharing using the receiver's public key PK, and only the receiver's private key SK can decrypt the ciphertext to obtain the shared data. As attackers cannot obtain the receiver's private key SK, they are unable to decrypt the shared data. This method can ensure, to a certain extent, the security of data during transmission, guaranteeing that only the genuine receiver can decrypt and view the shared data. Therefore, public-key cryptography is often applied in scenarios involving data sharing and identity verification.

### 2.2.7 Digital envelope

A digital envelope refers to the ciphertext obtained when the sender of a message encrypts the symmetric key using the receiver's public key PK, ensuring the security of the symmetric key during transmission. When using a digital envelope, only the receiver's private key SK can decrypt it to retrieve the symmetric key. Digital envelope technology combines the advantages of both symmetric encryption and public-key cryptography, resolving the security risks associated with sharing symmetric keys and addressing the issue of slow public-key encryption speed. It offers high security and is an important application of public-key cryptography.

The digital envelope mainly consists of two processes: generating the digital envelope and decrypting and opening the digital envelope. In the process of generating the digital envelope, the sender uses the symmetric key to encrypt the plaintext message M to obtain the message ciphertext, then encrypts the symmetric key with the receiver's public key PK to obtain the key ciphertext, and sends both the key ciphertext and the message ciphertext to the receiver. The process of generating the digital envelope is illustrated in Figure 2.9.

In the process of decrypting and opening the digital envelope, after receiving the ciphertext information sent by the sender, the receiver first decrypts the key ciphertext using their private key SK to obtain the symmetric key. Subsequently, the receiver uses this symmetric key to decrypt the message ciphertext and obtain the

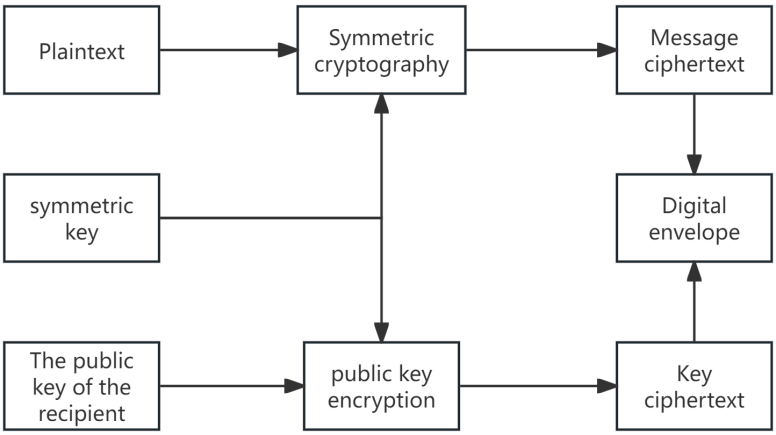


Figure 2.9: Generate digital envelope

plaintext. Since the key ciphertext can only be decrypted using the receiver's private key SK, which is kept by the receiver themselves, even if the message is intercepted by a malicious attacker during transmission, they cannot decrypt the digital envelope without the receiver's private key and thus cannot obtain the symmetric key. This ensures, to a certain extent, the security of the digital envelope technology.

### 2.2.8 Digital signature

Digital signature, also known as public-key digital signature, is an application of public-key cryptography. This technology is used to verify the source of a message and ensure the integrity of the data, preventing any tampering. The sender uses their private key to sign the message, and the receiver uses the sender's publicly available public key to verify if the message truly originated from the sender. This is done through the digital signature to prove the authenticity of the message source. Additionally, the sender's private key is absolutely secure. Once the verification is successful, the sender cannot deny the origin of the message. Moreover, digital signature technology not only verifies the source of the message but also ensures data integrity through hash values, guaranteeing that the data received by the receiver is authentic and unaltered. Therefore, the content transmitted by the digital signature includes the plaintext M itself and the ciphertext obtained by encrypting the hash value of the plaintext M with the private key.

The signature process is illustrated in Figure 2.10. Firstly, the sender performs a hash operation on the plaintext M to obtain H(M). Secondly, the sender encrypts H(M) using their private key SK to form the ciphertext SKH(M), which is then appended to the plaintext M, resulting in the signed message M||SKH(M). Finally, the signed message M||SKH(M) is sent to the receiver.

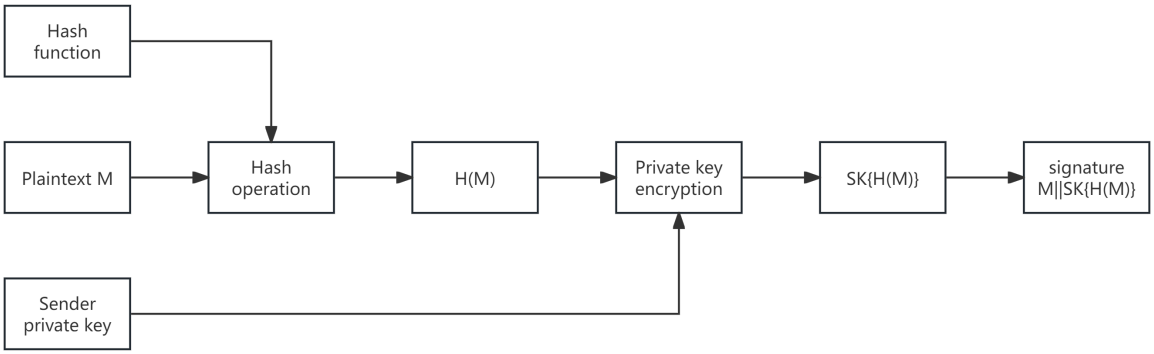


Figure 2.10: Signature process

The signature verification process is illustrated in Figure 2.11. Firstly, upon receiving the signature information M||SKH(M) from the sender, the receiver uses the sender's public key PK to verify the ciphertext SKH(M), ensuring that the message was indeed sent by the genuine sender, and decrypts it to obtain H1(M). Secondly, the receiver performs a hash operation on the plaintext M to obtain H2(M). Finally, the receiver compares H1(M) with H2(M). If they are consistent, it indicates that the plaintext M has not been tampered with and the message was sent by the genuine sender. By verifying the signature information, the receiver can ensure that the received message is from the actual sender. Additionally, the sender cannot deny the origin of the message. Furthermore, by verifying the hash value, the integrity of the message can be ensured.

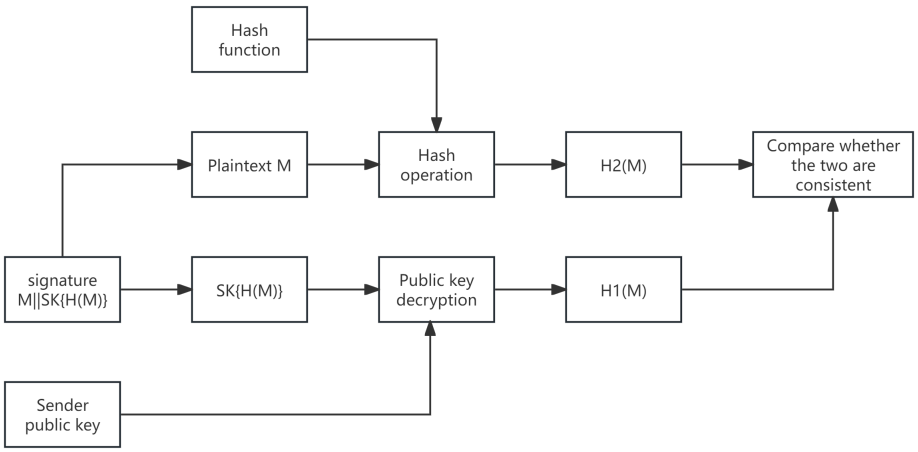


Figure 2.11: Signature verify process

## 2.3 Summary

This chapter provides a detailed introduction to the relevant knowledge involved in the human health detection system based on blockchain, including two major aspects: IoT-related knowledge (the definition of medical IoT, introduction to the main control chip and embedded operating system, UART serial port protocol, I2C serial port protocol, MQTT protocol) and blockchain technology (principles of blockchain, blockchain classification, hash functions, symmetric cryptography, public-key cryptography, digital envelopes, digital signatures). The relevant applications of these technologies in this article are shown in the following figure:

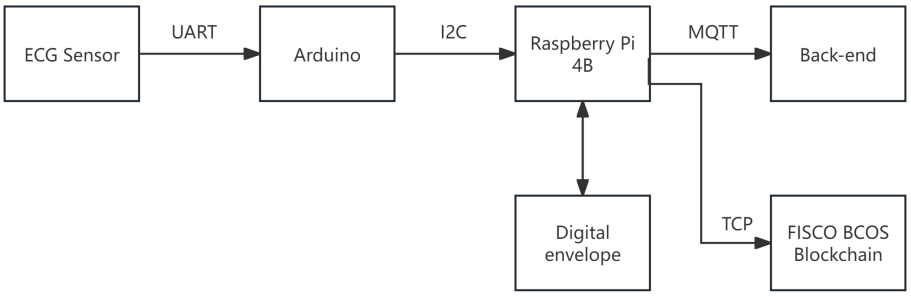


Figure 2.12: The process of IOT data connection

In the process of data processing and transmission, the ECG sensor is responsible for collecting data related to the human body's electrocardiogram. After the collection is completed, it transmits the data to the Arduino development board via the UART protocol. On the Arduino development board, the received analog data undergoes analog - to - digital conversion.

After the conversion is finished, the Arduino development board transfers the processed data to the Raspberry Pi 4B development board through the I2C protocol. Once the Raspberry Pi 4B development board receives the data, it encrypts the data using the pre - received user public key to generate a digital envelope, ensuring the security of the data during subsequent transmission and storage.

After that, the Raspberry Pi 4B development board stores the digital envelope on the blockchain via the TCP protocol, achieving the secure uploading of data to the blockchain. Meanwhile, the Raspberry Pi 4B development board also extracts key parameters such as the EnvelopeID from the digital envelope and sends these parameters to the back - end server for storage through the MQTT network, facilitating the subsequent data retrieval and management.

# 3 Research Methodology

In this chapter, the requirements analysis of the health detection system is first carried out. Based on the system requirements analysis, the research and design of the core modules of the health detection system are conducted. It mainly introduces the design of the Internet of Things (IoT) information collection module, the database module, the digital envelope, the blockchain module, and the front - end module.

## 3.1 Requirements Analysis of System

This subsection mainly conducts a requirements analysis of the health detection system, establishes a functional model, and performs a selection analysis of the required devices and software development platforms to make preliminary preparations for the specific functional design of the system.

### 3.1.1 Functional requirements analysis

This intelligent health monitoring system integrates multiple functions such as data collection, storage, display, and health assessment. Based on the actual operational conditions of this intelligent health monitoring system, we divide it into three parts: the front - end module, the back - end module, and the Internet of Things (IoT) module.

Each function of each module has its specific data requirements. Among them, the data processing process of the front - end module is relatively simple. It only needs to guide users to fill out forms through the interface and then send the form content to the back - end interface. Given its simplicity, we will not elaborate on it here. Next, we will focus on analyzing the back - end module and the IoT module of the health monitoring system.

#### 3.1.1.1 Back-end module data analysis

The back - end mainly has seven functions, namely registration, login, file management, file authorization, file query, device binding, health detection, and monitoring data display. The specific details of Input, Process, and Output (IPO) are shown in Table 3.1.

Table 3.1: Back-End Functional Modules with IPO Structure

| Function | Input | Processing | Output |
|---|---|---|---|
| User registration | Username, password, blockchain address..... | -Validate input format<br>-Verify the uniqueness of the blockchain address<br>-Store in MySQL | The Boolean value of the registered status |
| User login | Username, password | -Validate input format | The Boolean value of the login status |
| Record storage | Username, disease, patient address,doctor address..... | -Validate input format<br>-Store in blockchain | The Boolean value of the store status |
| Record authorization | doctor address | -Send to the smart contract and add it to the whitelist of the sending address | The Boolean value of the authorization status |
| Record query | patient address, record id | -Send to the smart contract record id and patient address<br>-Check whether the message sender is on the whitelist | Username, disease, gender..... |
| Device binding | device IP, publickey | -Validate input format<br>-Send the publickey to IOT device<br>-Store the device IP in MYSQL | The Boolean value of the bind status |
| Healthy monitor | NONE | -Send monitor command to the device | The Boolean value of the monitor status |
| Monitor data query | Monitoring timestamp | -Obtain the envelope id from MYSQL based on the timestamp selected by the user<br>-Send the envelope id to smart contract<br>-Check whether the message sender is on the whitelist | digital envelope |

#### 3.1.1.2 IOT module data analysis

The IoT module mainly has two core functions: user binding and health detection. The health detection process consists of two key steps: data collection and data encryption. The details of Input, Process, and Output (IPO) for these two functions and two steps are shown in Table 3.2.

Table 3.2: IOT Module Functional Modules with IPO Structure

| Function | Input | Processing | Output |
|---|---|---|---|
| Device binding | user address, publickey | -store user address and publickey in local | The Boolean value of the bind status |
| Real-time monitor | user address | -Sensors collect data<br>-Encrypt the collected data into digital envelope<br>-Send digital envelope to blockchain and send envelope id to back-end | digital envelope,envelope id,timestamp |
| Biometric Data Acquisition | Raw analog signals | -Sampling via SHIELD-EKG/EMG sensor<br>-Signal filtering<br>-Convert the digitized sensor data into voltage values | The ECG values |
| Generate digital envelopes | ECG values, user publickey | -Generate AES key pairs using the user's public key<br>-Encrypt data using the AES public key to form a digital envelope | The ECG values |

### 3.1.2 Non-functional requirements analysis

(1) Data Response Requirements

The consensus transactions among blockchain storage nodes introduce additional waiting time. It is necessary to select an appropriate blockchain platform and establish a suitable private or consortium chain based on the application scenario requirements to reduce response latency. Additionally, compared to databases, the speed of storing and retrieving data on the blockchain is slower. A mechanism tailored for the health monitoring system can be designed to prioritize the on-chain storage and querying of critical lightweight data, thereby improving the system's throughput.

(2) System Security Requirements

The health monitoring system designed in this study is primarily based on a web development model, accessible through URLs. In terms of system confidentiality and integrity, it must ensure that only authorized users can modify the traceability system data, preventing illegal and unauthorized disclosure of information. When selecting development frameworks and security control technologies, common types of web attacks should be considered, and corresponding defense strategies should be designed. If malicious behavior is detected, the system should be able to deny requests based on the IP address or add it to a blacklist. The data detected by the IoT module remains encrypted at all times and can only be decrypted using the user's private key to obtain the real data.

(3) System Usability Requirements

While ensuring that the system's functionality is not compromised, the front-end interface should be designed to be user-friendly, with a clear and concise interactive interface that is easy to use, reducing

complex logic jumps and the number of button clicks. A consistent interface design should be implemented, as maintaining consistency makes it easier for users to complete tasks and achieve goals related to traceability information input.

## 3.2 Hardware Components

### 3.2.1 Raspberry Pi 4B

Due to its rich on-chip peripheral resources, the Raspberry Pi Zero 4B was chosen as the microprocessor for the detection system in this study. This processor is an ARM-based microcomputer motherboard, featuring a quad-core 64-bit Cortex-A72 architecture. It uses an SD/MicroSD card as its memory and hard drive, with 1/2/4 USB ports and a 10/100 Ethernet interface (the Type A model lacks an Ethernet port) around the card-sized motherboard, allowing connections to keyboards, mice, and network cables. It also includes a TV output interface for analog video signals and an HDMI port for high-definition video output. All these components are integrated onto a motherboard slightly larger than a credit card, providing all the basic functionalities of a PC. Simply connecting it to a TV and a keyboard enables a wide range of functions, such as spreadsheet processing, word processing, gaming, and playing high-definition videos. Figure 3.1 shows the pinout diagram of the Raspberry Pi Zero 4B.

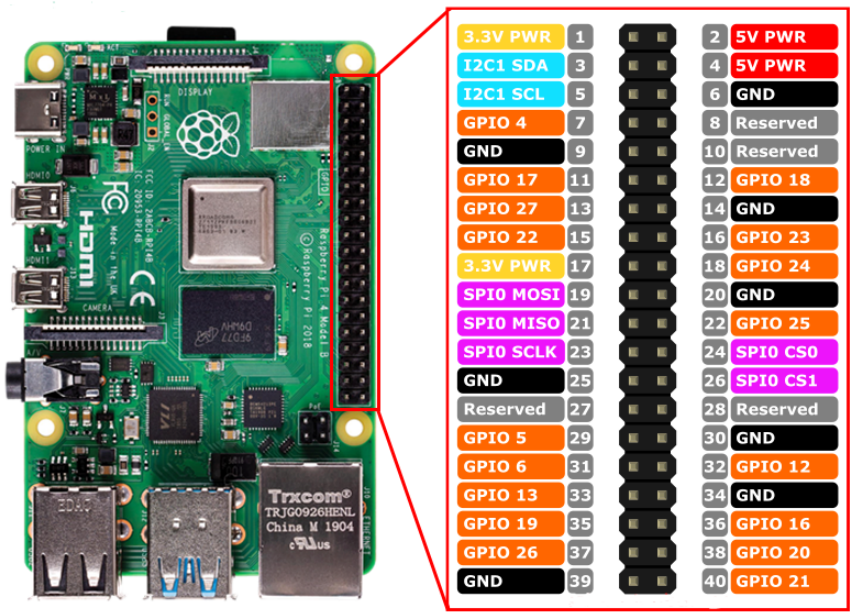


Figure 3.1: GPIO of Raspberry Pi Zero 4B

### 3.2.2 Arduino UNO R3

The Arduino UNO R3 is an open-source microcontroller development board introduced by Arduino. It is equipped with the ATmega328P microcontroller, which features a maximum operating frequency of 20 MHz and a performance of up to 20 MIPS. This board boasts powerful functionality and a wide range of applications. It includes 14 digital input/output pins (6 of which can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. In this system, the Arduino UNO R3 is used as the processor for ECG data acquisition, connecting to the Raspberry Pi and

the ECG sensor.Figure 3.2 shows the pinout diagram of the Arduino UNO R3.

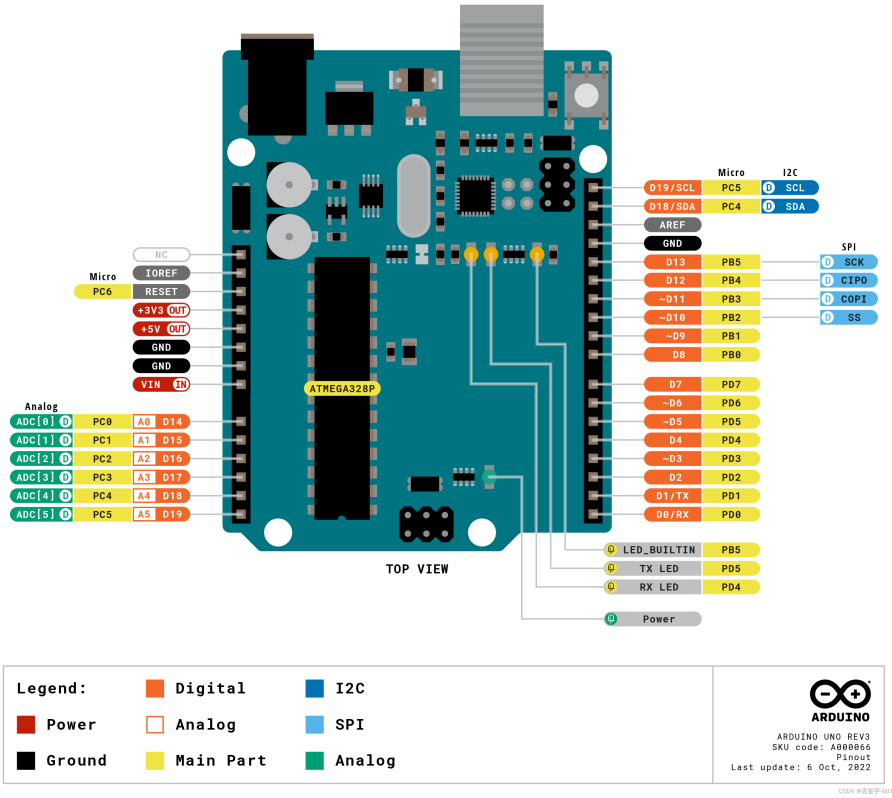


Figure 3.2: GPIO of Arduino UNO R3

### 3.2.3 SHIELD-EKG/EMG

The SHIELD-EKG/EMG is a powerful and user-friendly expansion board for biosignal acquisition. It is specifically designed for capturing electrocardiogram (EKG/ECG) and electromyogram (EMG) signals and is typically compatible with microcontroller platforms such as Arduino. It is engineered for precise measurement and analysis of bioelectrical signals, finding wide applications in fields such as healthcare, biomedical engineering, and sports science. In this system, the SHIELD-EKG/EMG is utilized as the sensor for acquiring ECG signals.

### 3.2.4 Wireless Communication Module

With the research and development of Internet of Things (IoT) technology, various wireless communication modules are being applied in various fields of life, including Bluetooth, ZigBee, Wi-Fi, LoRa, and NB-IoT, each with its own advantages and disadvantages. Therefore, selecting an appropriate wireless communication module is crucial for the design of an IoT system.

1. Bluetooth
   Bluetooth is a low-power, low-speed communication module developed by Ericsson that enables low-cost short-range wireless connections. The Bluetooth technology provides short-range wireless access services for terminal devices, operating in the globally universal 2.4GHz band. It features full-duplex transmission mode, typically supporting ”one-to-one” connections, and is commonly used in devices such as mobile phones, wireless headphones, and smart bracelets.

2. ZigBee
   ZigBee, similar to Bluetooth, is a wireless communication technology characterized by low data rate, low power consumption, and low cost. The ZigBee protocol is based on the 802 protocol established by the IEEE association and consists primarily of the physical layer, MAC layer, network layer, and application layer. It features three topological structures: tree network, mesh network, and star network. Although ZigBee has strong networking capabilities, it incurs high costs for later maintenance and upgrades. It is commonly used in areas such as smart grids, intelligent transportation, and mobile POS terminals.
3. WIFI
   Wi-Fi is a wireless local area network technology based on the IEEE 802.11 standard, which allows smart devices such as mobile phones, tablets, and computers to connect to the internet via wireless communication, forming a network system where devices can communicate with each other and share resources. Wi-Fi offers advantages of fast transmission speed, long transmission distance, and stable data transmission. However, its security is relatively low, and it has high power consumption with weak penetration through walls. It is commonly used in devices such as computers, mobile phones, and printers.
4. LoRa
   LoRa achieves the unification of low power consumption and long distance by leveraging spread spectrum technology, retaining its low power characteristics while significantly enhancing its anti-interference capabilities and greatly increasing its communication range, which can extend up to several to tens of kilometers. This resolves the previous dilemma where either low power consumption or ultra-long distance transmission could be realized but not both, thus forming an extended sensor network that is low in power consumption and long in distance. However, LoRa cannot directly connect to the network and requires a gateway for network connectivity. It is commonly used in areas such as drone communication and urban water meters.

When developing an Internet of Things (IoT) health monitoring system, it is essential to comprehensively consider factors such as the real-time nature of data transmission, power consumption, coverage range, and cost. Wi-Fi, as a mature and widely adopted wireless communication technology, offers advantages such as fast transmission speeds, stable data transfer, and ease of deployment, making it highly suitable for health monitoring systems that require real-time transmission of large volumes of data. For example, the collection and transmission of health data such as heart rate, blood pressure, and blood oxygen levels demand high data rates and stable connections, which Wi-Fi can effectively meet. Additionally, the widespread availability of Wi-Fi enables users to conveniently integrate health monitoring devices into existing home or medical networks, reducing the complexity and cost of system deployment.In conclusion, this system uses WIFI as the main wireless communication method.

## 3.3 Selection of blockchain platforms

When selecting a blockchain development platform, Fabric, Ethereum and FISCO BCOS are all worth considering options. Each of these platforms has different characteristics and advantages, suitable for different application scenarios and requirements. This section primarily discusses the unique features of these three consortium blockchain development platforms, as well as detailed analysis and comparison.

Ethereum is an open software platform that enables developers to build and deploy decentralized applications. It operates as a public blockchain network, designed to allow users to interact with social and financial

systems in a peer-to-peer manner. Ethereum has its own cryptocurrency, Ether, which miners strive to earn. The platform features smart contracts that define the rules and penalties of agreements and enforce these obligations. Initially, Ethereum employed a Proof of Work (POW) consensus mechanism, but it has since transitioned to a Proof of Stake (POS) network upgrade.

Hyperledger Fabric serves as a platform for distributed ledger solutions, built on a modular architecture that provides high degrees of confidentiality, flexibility, and scalability. It is designed to support pluggable implementations of various components and adapt to the complexities inherent in the economic ecosystem. Originally conceived and developed by IBM, the source code of Fabric was contributed to the Hyperledger project under the Linux Foundation in 2015. From its inception, Fabric was positioned as a cross-industry application, focusing on constructing a universal framework based on blockchain technology. Hyperledger Fabric primarily addresses the needs of large organizations, whereas Ethereum is more popular among smaller applications and public smart contracts. Hyperledger Fabric aims to create permissioned blockchains specifically tailored for enterprise use. A key characteristic of enterprises is data confidentiality and privacy. Fabric manages distributed ledgers through peer protocols built on HTTP/2. It employs smart contracts to provide controlled access to the ledger, which can be written in Go or Java. Remarkably, Hyperledger Fabric does not feature a cryptocurrency.

FISCO BCOS was born in 2017, launched by Financial Blockchain Shenzhen Consortium (FISCO), and represents a standard domestic blockchain infrastructure. FISCO is a non-profit organization jointly established on May 31, 2016, by over twenty financial institutions and tech companies, including the Shenzhen FinTech Association, Shenzhen Qianhai WeBank, and Shenzhen Stock Communication. Initially, FISCO BCOS was designed as an open-source blockchain base platform that is independent and controllable, specifically tailored for the financial industry. When designing regulatory interfaces, FISCO BCOS is more suitable for Chinese enterprises. As the platform evolved, it gradually began supporting scenatablerios beyond the financial sector. Technically, FISCO BCOS is derived from the Ethereum public blockchain. Although its application scenarios and use cases have diverged significantly from the public chain, it retains Ethereum's virtual machine, thereby inheriting the vast ecosystem of Ethereum.

Through a comparative analysis of various blockchain development platforms, as shown in Table 3.3, and based on the functional requirements of this system, it has been decided to use the FISCO BCOS platform for system development. As a domestic open-source enterprise-level blockchain platform, FISCO BCOS offers high performance, high security, and high scalability, making it particularly suitable for applications in the medical IoT sector. Choosing FISCO BCOS not only ensures the security and privacy of medical data but also leverages its mature developer community and rich technical ecosystem. FISCO BCOS provides comprehensive development tools and frameworks, such as WeBASE and Console, which enable developers to quickly build, deploy, and test smart contracts, significantly reducing the development cycle. Furthermore, the open-source nature of FISCO BCOS and its active community support provide developers with a wealth of open-source code, technical documentation, and solutions, further lowering development complexity and costs.

## 3.4 Overall architecture design of the system

### 3.4.1 Iot module architecture design

Most current IoMT (Internet of Medical Things) systems are typically divided into a four-layer structure, consisting of the sensor layer, gateway layer, cloud layer, and application layer. These layers cover the entire

Table 3.3: The Differences among Ethereum, Fabric, and FISCO BCOS

| | Ethereum | Fabric | FISCO BCOS |
|---|---|---|---|
| Design | Ethereum | Inherits IBM | Inherits Ethereum |
| Platform Description | General-purpose | General-purpose | General framework |
| Management | Ethereum developers | Linux Foundation | Golden Chain Alliance |
| Currency | Ether | None | None |
| Mining Reward | 5.0 Ether | N/A | N/A |
| Consensus | POS | Kafka,Solo,SBFT | PBFT,Raft |
| Admission Criteria | Free access | Access control | Access control |
| Transactions | Anonymous or private | Public or confidential | Public or confidential |
| Privacy | Public | Public to private | Public to private |
| Smart Contract Environment | EVM environment | Docker environment | EVM environment |
| Smart Contract Language | Solidity, Golang, C++, Python | Java, Golang | Solidity, C++ |

process from the collection of personal biometric data, through data storage, to visual analysis by doctors. However, in this system, the cloud layer is replaced by a Data layer, resulting in a four-layer structure that includes the sensor layer, gateway layer, blockchain layer, and application layer, as shown in Figure 3.3 below.

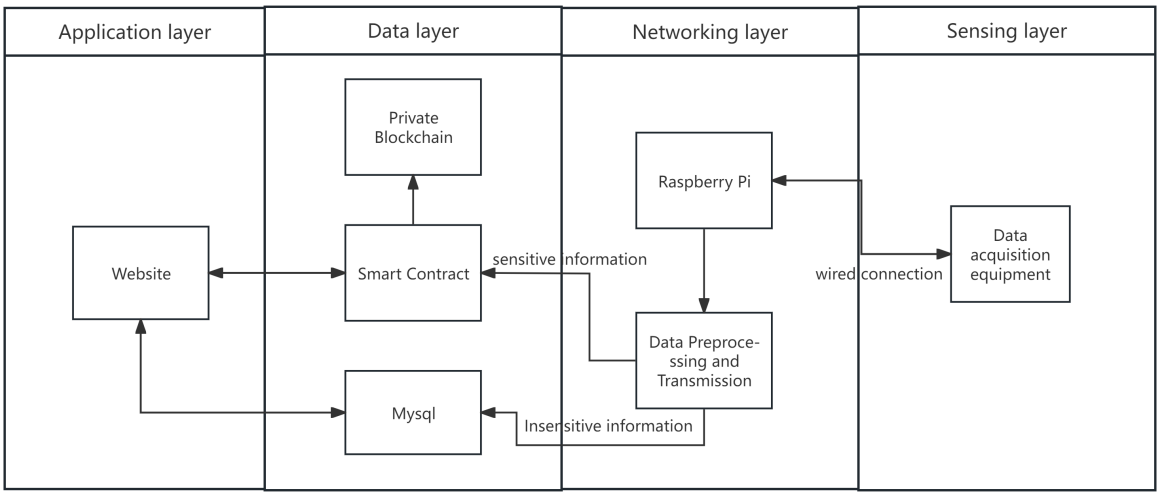


Figure 3.3: System Architecture

1. Application layer
    - User Interface: Provides operation interfaces for doctors, patients, and administrators, displaying data and analysis results.
    - API Interface: Offers interfaces for external system calls, supporting integration with other systems.
2. Data layer

- Private Chain: A private chain network built through FISCO BCOS, ensuring data privacy and control permissions.
- Smart Contract: Using Solidity to write smart contracts, enabling the storage, verification, and processing of medical data.
- Mysql: It is used to store some insensitive data, such as the timestamp of device startup.

3. Networking layer
    - Data Preprocessing Module: Cleans and formats the received data.
    - Data Transmission Protocol: Using MQTT protocols to transmit data to the Blockchain layer.
4. Sensing layer:Detect the target object and generate data.

## 3.5 Design of Iot Information Collection Module

### 3.5.1 Architecture design of the acquisition module

In the context of health monitoring, the primary sensing devices integrated include temperature sensors, heart rate sensors, and blood oxygen saturation sensors. When designing an IoT data acquisition platform, the following key factors must be prioritized: First, ensure the robustness and scalability of device integration to flexibly support additional types of sensing devices in the future, such as blood pressure monitors, glucose monitors, and respiratory rate sensors. Second, emphasize the authentication security of sensing devices to guarantee the safety of device connections and the authenticity and reliability of the collected data. Additionally, it is essential to design the data flow business logic of the IoT platform for connected devices, ensuring that data can be transmitted to the health monitoring system promptly and efficiently, thereby supporting subsequent analysis and applications.

Based on the above requirements, the system framework of the IoT data acquisition module is illustrated in Figure 3.4. The system is centered around the Raspberry Pi 4B and controlled by a front-end module. Specifically, ECG data is collected through the SHIELD-EKG/EMG sensor, and the acquired signals are transmitted to the Raspberry Pi via the Arduino UNO R3 microcontroller for data preprocessing and encryption. Subsequently, the processed data is transferred to the blockchain module for storage. Users and authorized accounts can access the information stored in the blockchain through the front-end module, achieving data transparency and traceability.

In terms of technical implementation, the system employs the HTTPS protocol to call the API interface of the blockchain development platform, storing the acquired data in a self-built database to complete the overall data acquisition process. Meanwhile, the system communicates with the backend through the MQTT protocol, enabling users to remotely control IoT devices via the front-end module, thereby facilitating efficient interaction between devices and the platform.

Through this design, the system not only meets the data acquisition needs of health monitoring scenarios but also ensures data security, reliability, and scalability, providing a solid foundation for subsequent health data analysis and applications.

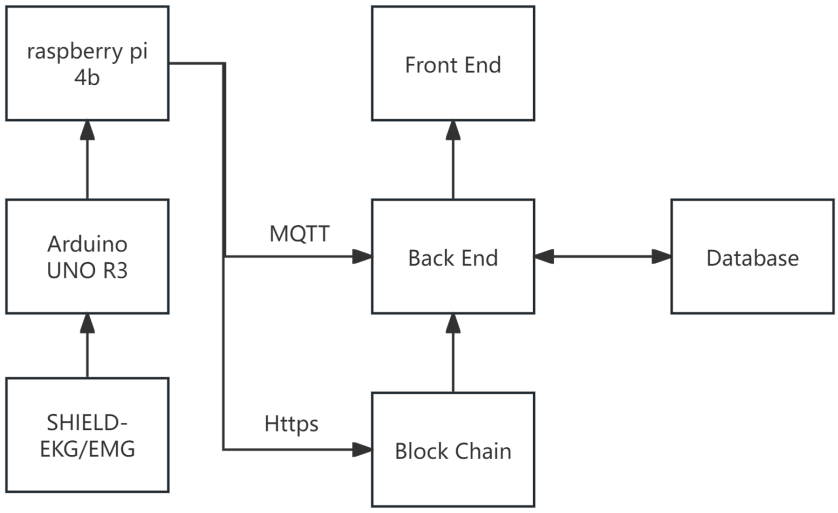


Figure 3.4: system framework of the IoT

### 3.5.2 MQTT Message Communication Design

MQTT is a standards-based messaging protocol or set of rules designed for machine-to-machine communication. Smart sensors, wearable devices, and other Internet of Things (IoT) devices often need to transmit and receive data over resource-constrained networks with limited bandwidth.

The MQTT protocol operates on a publish/subscribe model. In traditional network communication, clients and servers communicate directly with each other. A client requests resources or data from a server, and the server processes and returns a response. The core hub of MQTT communication is the MQTT broker, which is responsible for message exchange among clients. Due to the characteristics of the MQTT publish/subscribe model, devices and the cloud need to predefine Topics and determine Topic permissions.

The Topic design in this paper is shown in Table 3.4. Each Topic must be prefixed with /Product Key/device Name to form the complete Topic design. Here, $Product Key is a wildcard for the product identifier, and $device Name is a wildcard for the device name of each device.

Table 3.4: The MQTT topic design

| Function | Topic | Describe |
|---|---|---|
| Attribute setting | /healthy/bind | Bind the IP of the Iot device and the public key of the account |
| Transmit data | /healthy/envelopeId | Send timestamp and digital envelope ID |

## 3.6 Design of databases

### 3.6.1 Design of data entities

The data storage and querying of the health monitoring system are based on health data entities. The system performs data analysis on the entire health monitoring process, abstracting the required information at each stage into data entities to facilitate subsequent database design and development. The operation of the health monitoring system begins with user registration, proceeds through device monitoring and doctor diagnosis, and concludes with users querying health reports and providing feedback.

In the user registration phase, a user information entity is created to record details such as gender, phone number, username, and blockchain address. During the device monitoring phase, IoT devices collect user health data, with each monitoring record forming a data entity that includes information such as monitoring time and digital envelope ID. In the doctor diagnosis phase, doctors provide diagnostic reports based on the monitoring data, with each diagnosis record forming a data entity that includes the blockchain addresses of the doctor and patient, diagnosis time, and the transaction hash of the diagnostic report. When users query health reports, the system retrieves the corresponding diagnostic reports based on the blockchain address and transaction hash, and user feedback records also form feedback information entities. In addition to the three core data entities involved in the health monitoring process, the system also designs user entities, device entities, log entities, and others to support the complete operation of the system. The health monitoring process and the involved data entities are illustrated in the figure 3.5 and figure 3.6.

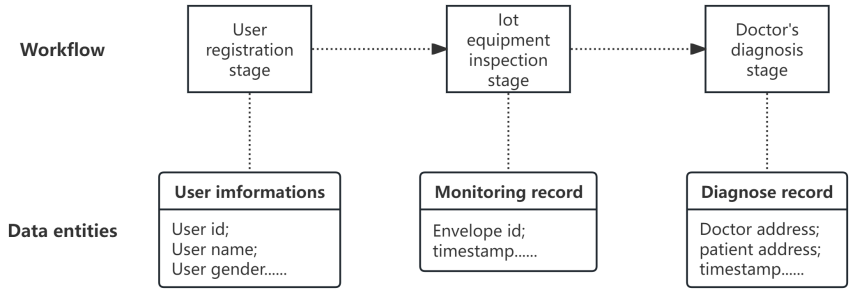


Figure 3.5: Health monitoring process related data entities

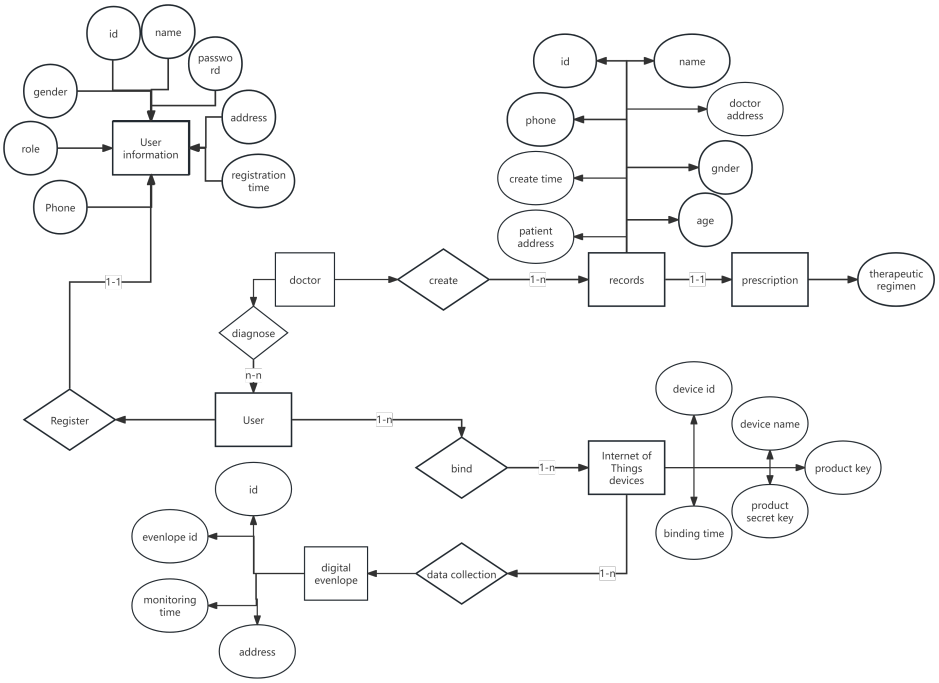


Figure 3.6: Database E-R diagram

### 3.6.2 Design of data tables

Based on the analysis of system management functions and the data entities related to the health inspection process, the data entities need to correspond to their respective data tables in the database and establish relationships between the tables. This section will provide a detailed design of the system's data tables, with an in-depth analysis of the core data table structures. The discussion is divided into two main parts: the detailed design of data tables related to system management and the detailed design of data tables related to the information entry of the traceability process. The following examples will illustrate the four core data tables of the system: the IoT device table as shown in Table 3.5, the user information table as shown in Table 3.6, the device collection information table as shown in Table 3.7, and the doctor diagnosis information table as shown in Table 3.8.

Table 3.5: Iot equipment table

| Field name | Data type | Length | Nullable |
|---|---|---|---|
| device id | bigint | 0 | Primary Key |
| device name | varchar | 64 | NOT |
| product key | Varchar | 100 | NOT |
| product secret | Varchar | 100 | NOT |
| add time | timestamp | | NOT |

Table 3.6: User information table

| Field name | Data type | Length | Nullable |
|---|---|---|---|
| id | int | 0 | Primary Key |
| name | varchar | 50 | NOT |
| password | varchar | 50 | NOT |
| gender | varchar | 10 | NOT |
| role | int | 0 | NOT |
| address | varchar | 100 | NOT |
| registration time | timestamp | | NOT |
| Phone | varchar | 20 | Yes |

Table 3.7: device collection information table

| Field name | Data type | Length | Nullable |
|---|---|---|---|
| id | int | 0 | Primary Key |
| envelope id | char | 64 | NOT |
| monitoring time | timestamp | | NOT |
| address | varchar | 100 | NOT |

Table 3.8: Doctor diagnosis information table

| Field name | Data type | Length | Nullable |
|---|---|---|---|
| id | int | 0 | Primary Key |
| name | varchar | 50 | NOT |
| Phone | varchar | 20 | Yes |
| gender | varchar | 10 | NOT |
| age | int | 0 | NOT |
| create time | timestamp | Null | NOT |
| doctor address | varchar | 100 | NOT |
| patient address | varchar | 100 | NOT |

## 3.7 Front end design

### 3.7.1 Selection of front-end development platforms

Currently, the three main front-end frameworks are Angular, React, and Vue. Below is a simple introduction to each of these frameworks:

1. React React (sometimes referred to as React.js or ReactJS) is an open-source JavaScript library introduced by Facebook for rendering data into HTML views and building user interfaces.React views are typically rendered using components that contain other components specified by custom HTML tags. React provides programmers with a model where child components cannot directly affect outer components (”data flows down” ), efficient updates to the HTML document when data changes, and clean separation between components in modern single-page applications.
2. Vue Vue is a front-end framework created by the Chinese developer Evan You, officially released in 2014. Vue incorporates features from Angular and React, such as Virtual DOM, two-way data binding, diff algorithm, reactive properties, and component-based development, and has made relevant optimizations to make it more convenient and easier to get started with. Vue is favored by developers for its simplicity, ease of use, and efficiency, particularly suitable for beginners. Vue also supports component-based development, allowing developers to separate UI and business logic, thus enhancing code reusability and maintainability.
3. Angular AngularJS was born in 2009, created by Misko Hevery and others, and is a front-end framework for building user interfaces, which was later acquired by Google. Angular is a rewrite of AngularJS, with versions after Angular 2 officially named Angular, and versions before 2.0 referred to as AngularJS. AngularJS is written in JavaScript, while Angular uses TypeScript, a superset of ECMAScript 6. Angular is an application design framework and development platform for creating efficient, complex, and sophisticated single-page applications. It extends HTML with new attributes and expressions, achieving a framework for multiple platforms, both mobile and desktop. Angular has many features, with the core ones being MVVM, modularity, automated two-way data binding, semantic tags, dependency injection, and more.

The comparison of characteristics among the front-end frameworks is shown in the table 3.9. After analysis, Vue is found to be easy to get started with, maintainable, efficient, flexible, and suitable for small-scale development, which aligns well with this project. Additionally, Vue has strong community support in China. The Vue community ecosystem is robust, with a wealth of open-source components, tools, and plugins, al-

lowing developers to develop applications more quickly and share their experiences and achievements. In conclusion, Vue is chosen as the framework for this system.

Table 3.9: Front-end frame comparison

| Category | React | Vue | Angular |
|---|---|---|---|
| Release Year | 2013 | 2014 | 2010 |
| Size | Smaller | Smallest | Larger |
| Type | Declarative Library | Progressive Framework | Full Framework |
| Type System | JavaScriptX | TypeScript-first | TypeScript-first |
| Data Flow | Unidirectional | Bidirectional | Bidirectional |
| Template System | JSX | HTML-based Templates | HTML+TypeScript |
| Ecosystem | Massive | Extensive | Extensive |
| Community Support | Strong | Active | Strong |
| Server-Side Rendering | Supported | Supported | Supported |
| Mobile Support | React Native | Vue Native | Ionic/NativeScript |
| Origin | Facebook | Evan You | Google |
| Version Control | Facebook | Community | Google |
| Learning Curve | Moderate | Simple | Difficult |
| Template Syntax | JSX | HTML-based | HTML with Angular directives |
| Data Binding | Unidirectional Data Flow | Two-way Data Binding | Two-way Data Binding |
| Build Tools | Create React App | Webpack Vue-Cli, Vite | Webpack Angular-Cli |
| State Management | Redux, MobX | Vuex, Pinia | NgRx |
| Performance | Excellent | Excellent | Good |
| Enterprise Adoption Rate | High | Medium | High |
| Maintainer | Facebook | Vue.js Team | Google |
| Documentation Completeness | High | High | High |
| Community Size | Large | Large | Large |
| Large Enterprise Adoption | Extensive | Growing | Extensive |
| Suitable Scenarios | Large Applications | Small to Medium Applications | Large Complex Applications |

### 3.7.2 Development Environment and Tools

When developing front-end applications using the Vue.js framework, a series of development environments and tools are used to ensure the development process is efficient and manageable. Here is an introduction to some of the tools and environments that are commonly used:

- Integrated Development Environment(IDE) Visual Studio Code (VS Code): One of the most popular code editors, offering rich plugin support for Vue.js development, such as the Vetur plugin, which provides features like syntax highlighting, IntelliSense, and formatting.

- Package Management Tool npm: The default package manager for Node.js, used for installing and managing project dependencies.
- Build Tool Parcel: A zero-configuration build tool suitable for smaller-scale projects.
- Debugging Tool Vue DevTools: A browser extension specifically developed for Vue.js, supporting Vue component debugging and inspection of state management tools.
- Testing Tool Vue Test Utils: A library specifically designed for testing Vue.js components, aiding in writing unit tests.
- State Management Vuex: A state management library for Vue.js, used to manage the global state of an application.
- Component Library Element Plus: A component library based on Vue 3, providing a rich set of UI components suitable for middle and back-end management systems.

### 3.7.3 User Rights Management

The account permissions for various users of this system are shown in Figure 3.7.

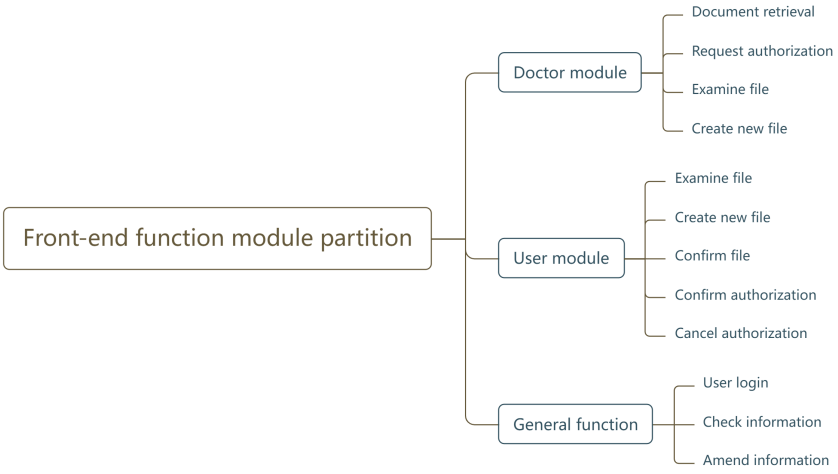


Figure 3.7: Front-end function module partition

The system includes two types of accounts: doctor accounts and regular user accounts. Doctor accounts have the functionalities to search for records, request authorization, create records, and view records. Regular user accounts, on the other hand, can revoke authorization, confirm authorization, create records, view records, and verify record data. Additionally, the permissions related to account information are common to both types of accounts, including logging into the system, viewing account information, and modifying account information.

## 3.8 Design of blockchain module

### 3.8.1 Data On-Chain and Query Process Design

The data uploading and querying process of the system is illustrated in Figure 3.8 below.

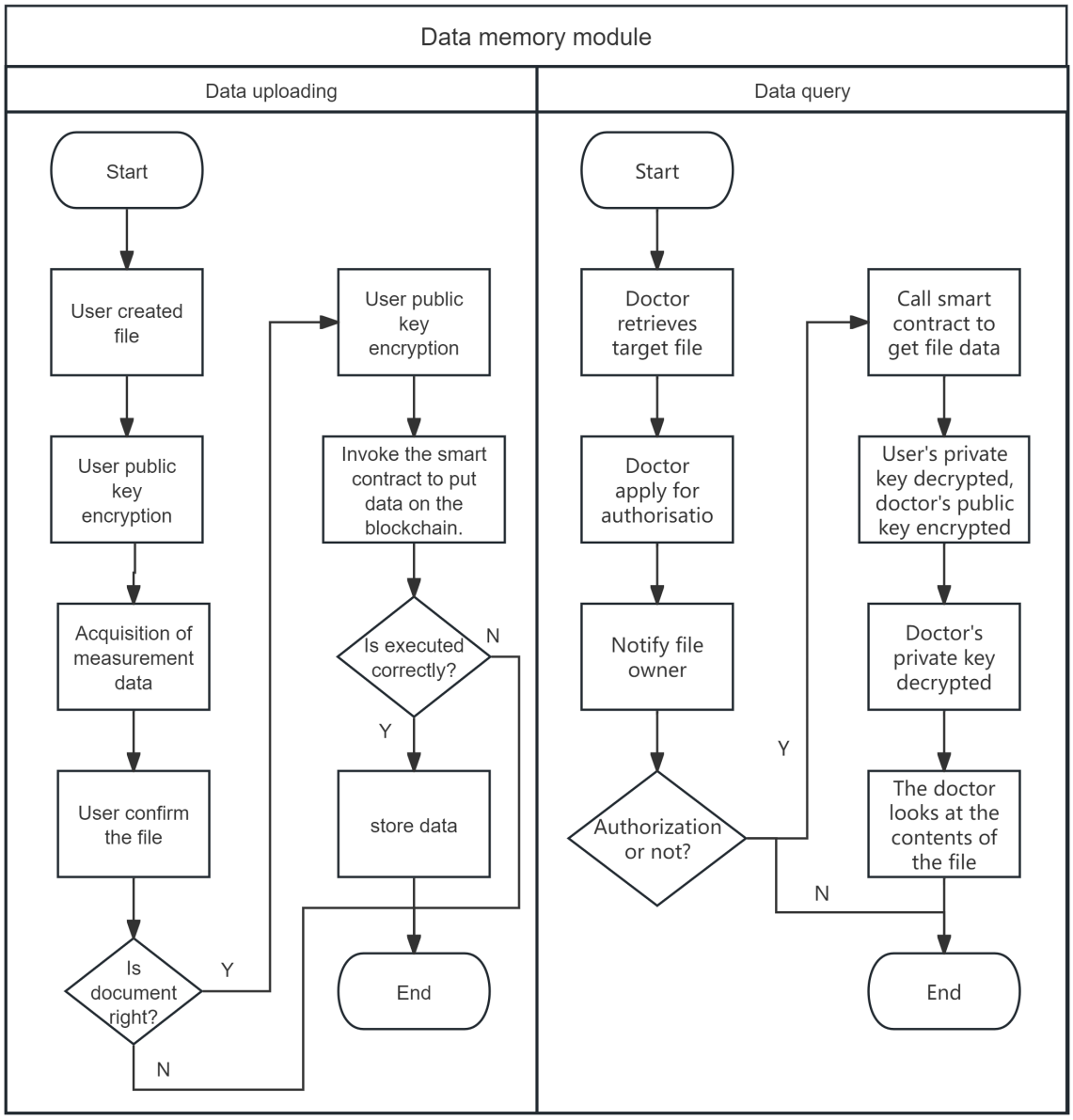


Figure 3.8: Data upload and access process

- Data Uploading Process: First, users or doctors have the authority to perform account creation, obtaining the corresponding public and private keys. Next, the public key is used to encrypt the account information. Users or doctors then confirm the accuracy of the data to be uploaded; if the data is found to be incorrect, they can choose to abandon the data-chaining process. If the data is confirmed to be correct, the program will continue, further encrypting the data with the public key and storing it on the blockchain via a smart contract, completing the entire storage operation. Thus, the program concludes.
- First, a doctor or patient initiates a search for the target file. If the user is searching for their own file, the authorization process is skipped, and the program proceeds directly to invoking the smart contract. If a doctor intends to review the file, they must first request authorization from the patient. If the patient denies authorization, the program will terminate; if the patient grants authorization, the process continues. The program then invokes the smart contract to retrieve the data from the blockchain and

decrypts it using the patient's private key. The decrypted data is then re-encrypted with the doctor's public key within the contract and sent to the doctor. Upon receiving the encrypted data, the doctor node decrypts it using its private key to obtain the correct data, and the program concludes.

The information of the health monitoring system as a whole is in JSON format, and the data entity of each link corresponds to a JSON object. The specific format is as follows:

```
{
    "digital envelope": {
        "envelope id": "......",
        "timestamp": "......",
        "ephemeral pub_key": "......",
        "iv": "......",
        "ciphertext": "......",
        "tag": "......"
    },
    "healthy record": {
        "name": "......",
        "gender": "......",
        "disease": "......",
        "doctor": "......"
    }
    ......
}
```

### 3.8.2 Design of smart contract

The contract program receives the request, performs the corresponding action, and modifies the state and values of the smart contract. The process of invoking a smart contract function generates a transaction that is stored in the blockchain. The "input" field in the transaction record's data structure stores the name of the invoked contract function, as well as the parameter types and values passed in. Based on the operational mechanism of smart contracts, a health data management smart contract was designed to achieve privacy protection and authorized access for health records. The operational mechanism of the smart contract is illustrated in Figure 3.9.

According to the design of health data management, health data is primarily divided into encrypted health records and authorized access records. Since health record data involves privacy, it is not stored in plaintext but is instead encrypted before being stored on the blockchain. After the database design is completed, the information from each stage is queried and combined into a JSON string. The complete health data and authorized access records are stored on the blockchain by invoking the smart contract's save method.

### 3.8.3 Design of digital envelop

The core function of the digital envelope module is to provide a secure and reliable encryption mechanism for storing and transmitting sensitive information on the blockchain, ensuring that data maintains confidentiality and integrity even in the open blockchain environment. Since the ECDSA public key system (based on the secp256k1 curve) used by the FISCO BCOS platform only supports signature verification and cannot directly encrypt data, a digital envelope encryption method based on Elliptic Curve Cryptography (ECC) and the Advanced Encryption Standard (AES) has been designed.

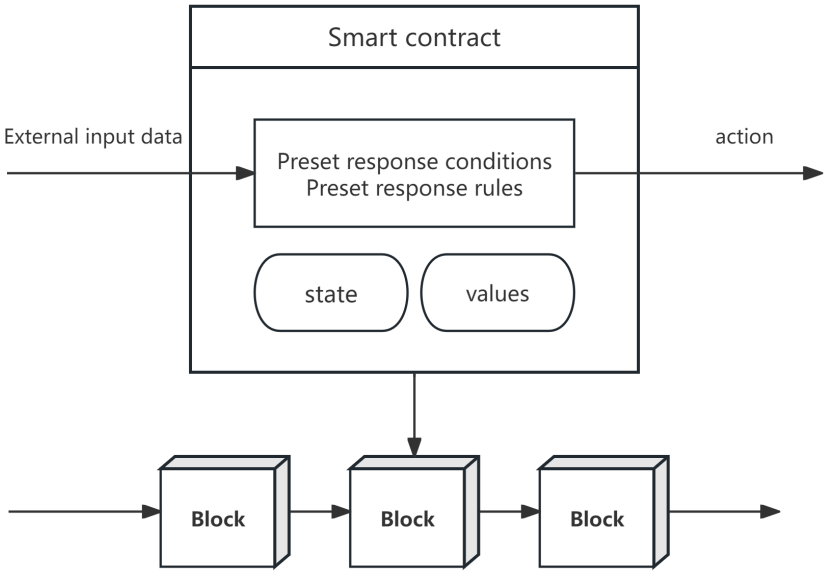


Figure 3.9: Operation mechanism of smart contract

In the specific process, the recipient's account public key is first passed in, followed by the generation of a temporary ECC key pair, and the temporary public key is serialized through a simulated encryption process. Then, a shared key is derived by combining the temporary private key and the recipient's public key using the ECDH algorithm, which is subsequently converted into an AES key via HKDF. Finally, the plaintext data is encrypted using the AES-GCM mode, generating ciphertext, an initialization vector (IV), and an authentication tag, which are then packaged with the user address, envelope ID, and timestamp to form a digital envelope for on-chain storage.As shown in Figure 3.10.

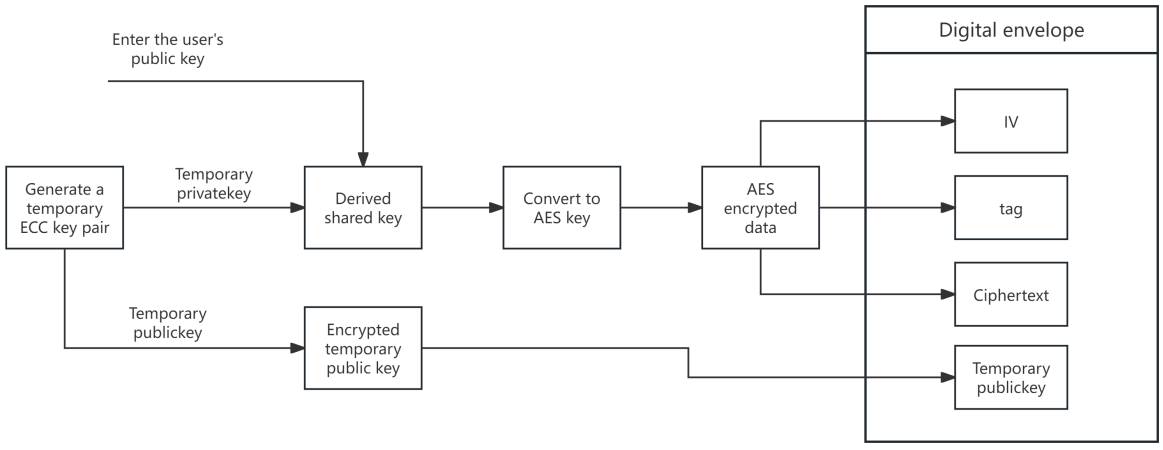


Figure 3.10: The encryption process of digital envelopes

When a user needs to access the information, the digital envelope is retrieved from the blockchain, and the private key is passed in for decryption: the shared key is re-derived using the ECDH algorithm, and the ciphertext is decrypted using the AES-GCM mode in combination with the IV and authentication tag, ultimately obtaining the original plaintext data.As shown in Figure 3.11.

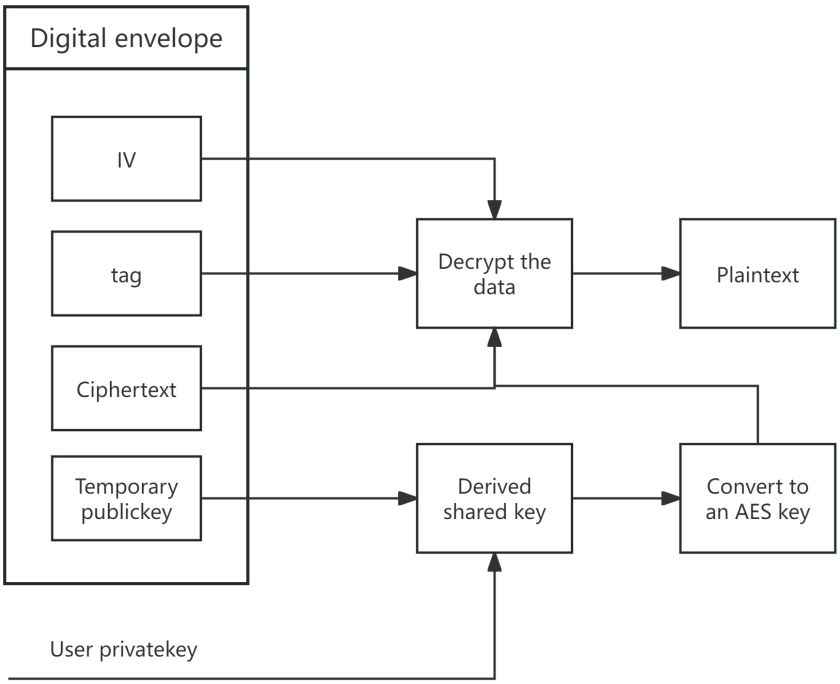


Figure 3.11: The decryption process of digital envelopes

This method not only efficiently leverages the key exchange capabilities of ECC but also combines the strong encryption performance of AES, ensuring data security and privacy. It is widely applied in fields such as healthcare, financial services, supply chain management, and identity authentication, providing an efficient and secure data encryption solution for blockchain applications.

## 3.9 Chapter summary

This chapter begins with a system requirements analysis, followed by a layered architecture design for the health monitoring system based on blockchain and Internet of Things (IoT) technologies. The overall architecture is divided into the presentation layer, business logic layer, data acquisition layer, and data storage layer, with communication between layers facilitated by RESTful APIs or MQTT.

The presentation layer is responsible for user interaction and adopts a front-end and back-end separation design pattern. The front-end uses the Vue.js framework to build a Single Page Application (SPA), while the back-end provides API services through the Spring Boot framework. The business logic layer decomposes core functionalities into multiple independent microservices, such as user service, profile service, and data acquisition service, supporting horizontal scalability. The data storage layer employs blockchain as the primary database for data storage and utilizes Redis as a caching layer to enhance system performance.

Subsequently, module designs are carried out based on the specific functionalities of each layer. This includes the design of the IoT information acquisition module, specifically the MQTT message communication design; the database design, encompassing data entity design and data table design; the blockchain module design, involving data on-chain design and smart contract design; and finally, the system's business functionality design is conducted module by module. Through the above design, the system is capable of meeting the business requirements for high concurrency, high performance, and high reliability.

# 4 Implementation and Testing of a Health Monitoring System Based on Blockchain and Iot

This chapter primarily focuses on the software and hardware implementation and testing of the system, based on the requirements analysis and functional design of the health monitoring system presented in Chapter 3. In this chapter, the core technical implementation process of the system will also be introduced in detail, divided into modules.

## 4.1 Preparation of the development environment

Before the system implementation, the development environment for the healthy monitoring system was prepared. For front-end development, the Vue.js framework was used to implement business logic, the Element component library served as the interface construction tool, Node.js was used as the runtime environment for front-end code, and Visual Studio Code was utilized as the editor and debugging tool.

For back-end development, the Springboot framework was employed to implement backend services, FISCO BCOS API interfaces were used to realize data on-chain business, and IntelliJ IDEA served as the integrated development environment for Java code. The Raspberry Pi was developed using the Python language, with PyCharm as the development environment for Python code, and after development, it was ported to the Geany IDE on the Raspberry Pi for operation. The ARDUINO development board was developed using C++, with the official ARDUINO IDE as the development environment. Smart contracts were developed using the Solidity language, with Remix as the IDE for coding, compiling, and debugging smart contracts. Webase was used as the IDE for compiling and deploying smart contracts. This paper utilizes the Webase webpage to build a private chain for the development and testing of the entire system. The main tools involved in the system development are listed in Table 4.1.

Table 4.1: System development tool table

| name | comment |
|---|---|
| Node.js | JavaScript runtime environment |
| Vue.js | JavaScript development framework |
| Element UI | Front-end interface development component library |
| Visual Studio Code | An integrated environment for front-end code development |
| Springboot | Back-end development framework |
| JDK17 | Javat runtime environment |
| IntelliJ IDEA | An integrated environment for back-end code development |
| Webase | Blockchain controls web pages |
| Remix | An integrated environment for smart contract code development |
| Pycharm | Raspberry PI code development platform |
| Geany IDE | Raspberry PI code running platform |
| ARDUINO IDE | Arduino development board code development platform |

## 4.2 Implementation of blockchain module

This section mainly introduces the method of constructing the FISCO BCOS blockchain network. The blockchain network is used solely for experimental purposes, with the primary focus on setting up a FISCO

BCOS private chain. During the experimental phase, since conducting experiments on a public chain requires consuming cryptocurrency, this section only establishes a multi-node private chain locally for experimentation. Since smart contracts run on the Ethereum Virtual Machine (EVM), they can be conveniently migrated to the Ethereum blockchain or other consortium chains later based on business needs. The following will provide a detailed explanation of the main implementation of the FISCO BCOS blockchain module.

### 4.2.1 Build blockchain

1. Build a private chain for the single group FISCO BCOS

   This system is based on the Ubuntu operating system for installing software and deploying experimental environments. First, the development and deployment tool, the build_chain.sh script, depends on OpenSSL and cURL. Depending on the operating system, use the appropriate commands to install these dependencies.

   After installing the dependencies, download the build_chain.sh script and execute it. In the fisco directory, run the command:

   ```
   bash build\_chain.sh -l 127.0.0.1:4 -p 30300,20200,8545
   ```

   To generate a single-group FISCO chain.Since FISCO BCOS employs an improved version of the Practical Byzantine Fault Tolerance (PBFT) consensus mechanism, it requires a minimum of 4 nodes. So user nees ensure that the ports 30300 30303, 20200 20203, and 8545 8548 on the machine are not occupied. Upon successful execution, the output will display ”All completed,” as shown in Figure 4.1. If an error occurs during execution, please check the error messages in the nodes/build.log file.

   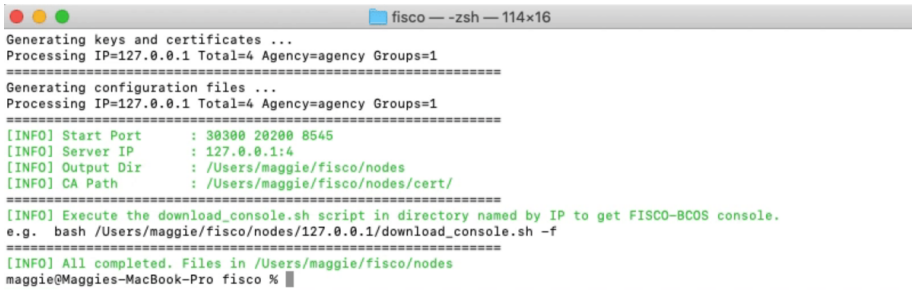
   

   Figure 4.1: FISCO BCOS deploy successfully

2. Configure and use the console

   In the console, functions such as querying the blockchain status and deploying/calling contracts can be implemented, allowing for quick access to the required information. To install the console, we first need to set up the environment dependencies, with Java 14 being the recommended version. Next, download and run the console installation script. After the installation is complete, configure the console certificates. Enter the command:

   ```
   cd ~/fisco/console && bash start.sh
   ```

   To run the console. The output shown in Figure 4.2 indicates a successful startup.

### 4.2.2 Deploy smart contracts

1. Smart Contract programming and debugging


Figure 4.2: Console deploy successfully

The smart contract code is written in the Solidity language and can be executed on the Ethereum Virtual Machine (EVM). The EVM is a decentralized virtual machine, which serves as the environment where the code within smart contracts is executed. Before the contract is deployed, it needs to be compiled into machine language that the EVM can execute.

This article uses the Remix web interface for smart contract programming and debugging, and the following will demonstrate the specific steps using Healthy.sol as an example. After writing the code, it is necessary to select the compiler version on the Remix interface. Here, we choose version 0.6.10, which is close to the version declared in Healthy.sol. Click the compile button to compile the code. Remix provides a powerful debugger that allows developers to execute smart contract code step by step, view variable states, and call stacks. The debugger supports functions such as setting breakpoints and single-step execution. Additionally, developers can debug deployed smart contract transactions to view the execution process and results of the transactions. After compilation, if there are no warnings or errors and the compile icon in Remix shows a green checkmark, it indicates that the smart contract has been successfully compiled.

Once the compilation is successful, the deployment function of Remix can be used. Remix supports deploying smart contracts to the Ethereum mainnet, testnets, or local development environments (such as Ganache). Developers can choose different Ethereum accounts and networks for deployment. After deployment, developers can interact with the smart contract through the Remix interface, call contract functions, and view the returned results to confirm that the code functions correctly.

2. WeBASE Setup

WeBASE (WeBank Blockchain Application Software Extension) is a set of universal components built between blockchain applications and FISCO-BCOS nodes. It designs various modules around transactions, contracts, key management, data, and visual management, allowing developers to deploy subsystems based on business needs. WeBASE shields the complexity of the underlying blockchain, lowers the barrier for developers, and significantly improves the development efficiency of blockchain applications. It includes subsystems such as node front-end, node management, transaction chain, data

export, and web management platform.

This system uses WeBASE to deploy and maintain smart contracts. Before deploying a smart contract, it is necessary to set up the WeBASE middleware on the server. The process of setting up WeBASE includes: first, preparing the environment to ensure that the requirements for the operating system, Java, MySQL, and FISCO BCOS nodes are met; then, downloading the WeBASE source code and configuring the database and blockchain node information; next, compiling and starting the WeBASE service, with the option to deploy the front-end interface; finally, accessing the WeBASE management interface through a browser, verifying the deployment, and starting to use its rich blockchain management features.

3. Contract deployment

After successfully compiling the smart contract and setting up WeBASE, the next step is to deploy the contract to the FISCO BCOS private chain built in Section 4.2.1. First, start all blockchain nodes to ensure the blockchain network is running properly. Save the smart contract written in Remix locally, then open it in the contract IDE of WeBASE-Front. Select a compiler version close to the one declared in Healthy.sol, and click "Compile Contract." After compilation is complete, click "Deploy Contract." Once the contract is deployed, the deployment details will be returned at the bottom of the IDE, as shown in Figure 4.3.

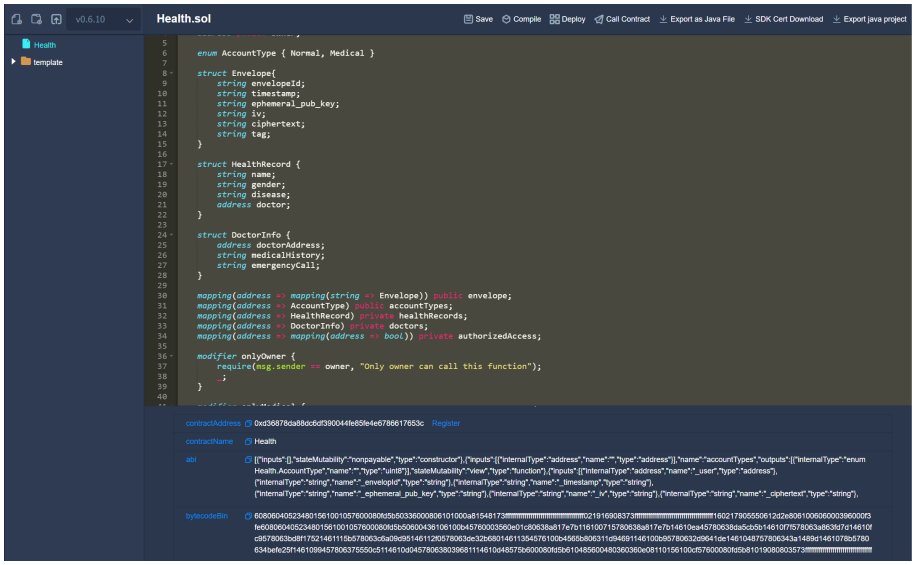


Figure 4.3: Successfully deployed the smart contract

The deployment details include information such as the contract name, contract address, Bytecode, and ABI. Bytecode is the translated information of the Solidity contract code, containing binary computer instructions. Bytecode typically encodes numbers, constants, and other information together, with each instruction referred to as an opcode, each being 1 byte (8 bits) in size. On the FISCO BCOS platform, the contract deployed on the blockchain is Bytecode, which is executed when users interact with the smart contract. The ABI defines the methods and variables that can be interacted with in the smart contract. Since the smart contract is transformed into Bytecode before being deployed to the blockchain, contract users need to understand the available operations. The ABI serves as a standard to display method interfaces, enabling other programming languages to interact with the smart contract as well.

### 4.2.3 Contract interface implementation

The implementation of the contract interface aims to seamlessly integrate the functionalities of the blockchain module into the health monitoring system. By leveraging WeBASE's HTTP API interface for development, users can directly invoke contracts through HTTP requests, enabling them to conveniently perform data uploading and querying operations via a streamlined web system.

In the specific implementation, the compiled and deployed contract name, contract ABI, and contract address are first configured in the backend WeBASE file, and the access address of the blockchain node is set. The abiInfo method is used to establish a connection with the blockchain network, validate and generate the ABI file, ensuring that the contract is correctly deployed. Subsequently, the commonReq method is utilized to invoke the functionalities of the smart contract, encapsulating the HTTP request logic to support read and write operations on the contract and return the execution results. Below is the backend code implementation for the health monitoring system to interact with the blockchain.

```
    /*Send the abi interface*/
    public static String abiInfo() {
        JSONObject data = new JSONObject();
        JSONArray abiJSON = JSON.parseArray(ABI);
        data.put("groupId", 1);
        data.put("contractName", contractName);
        data.put("address", contractAddress);
        data.put("abiInfo", abiJSON);
        String dataString = data.toString();
        String response = HttpRequest.post(BASE_URL + "contract/abiInfo")
                .header(Header.CONTENT_TYPE, "application/json")
                .body(dataString)
                .execute()
                .body();
        return response;}
        /*Transaction processing interface*/
    public static String commonReq(String userAddress,
                                   String funcName,
                                   List funcParam) {
        JSONObject data = new JSONObject();
        JSONArray abiJSON = JSON.parseArray(ABI);
        data.put("user", userAddress);
        data.put("contractName", contractName);
        data.put("contractAddress", contractAddress);
        data.put("funcName", funcName);
        data.put("contractAbi", abiJSON);
        data.put("groupId", 1);
        data.put("funcParam", funcParam);
        data.put("useCns", false);
        data.put("cnsName", "");
        String dataString = data.toString();
        String response = HttpRequest.post(BASE_URL + "trans/handle")
                .header(Header.CONTENT_TYPE, "application/json")
                .body(dataString)
                .execute()
                .body();
        return response;}
```

## 4.3 The implementation of the Internet of Things collection module

The IoT data acquisition module is primarily responsible for collecting health data. In this paper, ECG data is taken as an example, and the module's functionalities are implemented based on the IoT acquisition design scheme outlined in Section 3.3. The implementation of the IoT acquisition function is divided into three main parts: the setup of IoT devices, the encryption of monitoring data, and the reporting and transfer of data. Below, the functional implementation of these three parts is elaborated in detail.

### 4.3.1 The setup of IoT devices

This paper utilizes the Raspberry Pi 4B development board for device construction. The development board employs the Broadcom BCM2711 as the main control chip, featuring a Cortex-A72 core, 2GB of LPDDR4 memory, and support for dual-band Wi-Fi (2.4 GHz and 5 GHz) and Bluetooth 5.0. The Raspberry Pi 4B communicates with the Arduino UNO R3 via a USB interface, with a maximum communication rate of up to 5 Gbps. As shown in Figure 4.4, the development board is currently equipped with an SHIELD-EKG/EMG sensor for capturing electrocardiogram (ECG) signals.

Figure 4.4: Development board equipment debugging

### 4.3.2 The encryption of monitoring data

To ensure data security, after the data acquisition device completes data collection, the data must first be encrypted into a digital envelope on the local Raspberry Pi before transmission. Based on the design of the data encryption section in Section 3.6.3, the specific implementation process is as follows:

Before the detection function is initiated, the IoT device needs to be bound to a user account. Each IoT device is only allowed to be bound to one user at a time. The user inputs the IP address of the IoT device and the account public key through the front-end web page to complete the device binding operation. The front-end

binding function page is shown in Figure 4.5. After the binding is completed, the IoT device obtains the user's account public key, as shown in Figure 4.6.

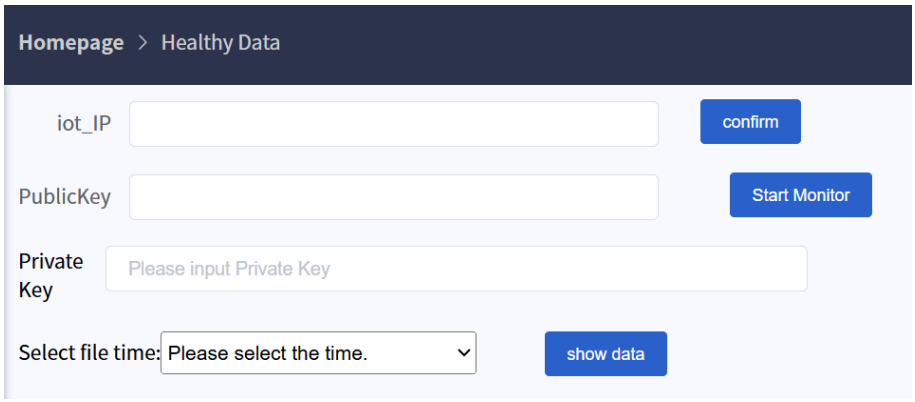


Figure 4.5: Caption


Figure 4.6: Binding completed

When the system initiates the detection function, the Raspberry Pi generates a JSON data packet containing the sensor sampling rate, sampling time, and voltage values, as shown in Figure 4.7.

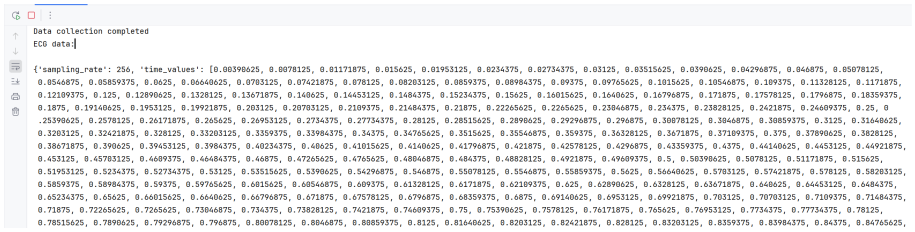


Figure 4.7: Unencrypted collected data

After obtaining the data packet, the system sends both the data packet and the previously obtained user public key to the encryption module. The user public key is derived to generate an AES key, which is then used to encrypt the data packet using AES-GCM, producing ciphertext, an initialization vector (IV), an authentication tag (Tag), and a temporary public key. Next, the system randomly generates a 32-bit random number as the Envelope ID to facilitate subsequent digital envelope operations. The contents of the data packet are shown in Figure 4.8. Finally, the system packages the Tag, IV, temporary key, ciphertext, and Envelope ID into a new data packet. The contents of the data packet are shown in Figure 4.8.

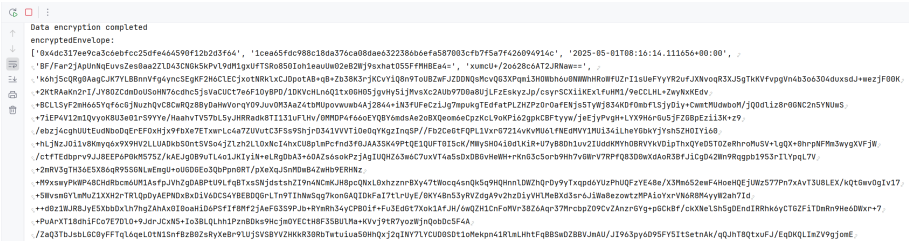


Figure 4.8: Digital envelope

### 4.3.3 The reporting and transfer of data

After obtaining the digital envelope, the system sends the digital envelope to the blockchain for storage via the WeBASE API interface. Simultaneously, based on the design outlined in Section 3.3.2, the system transmits the timestamp of the generated data packet and the Envelope ID to the backend using the MQTT protocol, storing them in a MySQL database to facilitate subsequent data retrieval and lookup.

### 4.3.4 Iot data query and display

Users can retrieve the collected data based on the collection time through the front-end web interface. Subsequently, they can input their private key to decrypt and view the data. The specific implementation process is detailed in Figure 4.9.

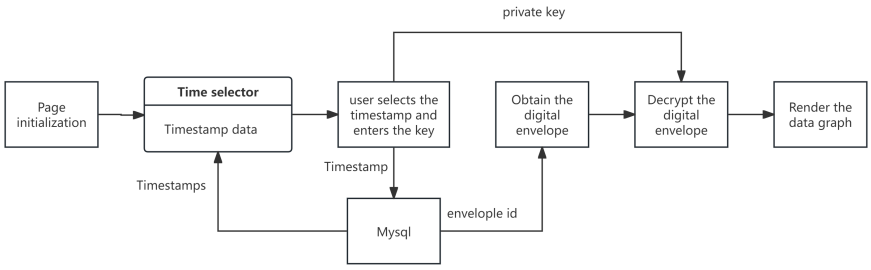


Figure 4.9: Iot data display process

After the data is successfully stored, the time selector on the health data display page will read the detection timestamp data of the logged-in user from the MySQL database and display it in the selector's options. Users need to select the data they want to view based on the detection time and input their account's private key, then click the "show Data" button.

Upon clicking the button, the system will first read the envelope ID stored under the corresponding timestamp, then send the user's address and the envelope ID to the smart contract via the Webase API to execute the code for retrieving the digital envelope, ultimately returning the corresponding digital envelope. Following the data decryption design outlined in Section 3.6.3, the system will use the user's input key to decrypt the digital envelope and send the retrieved data back to the front-end. Based on the time and voltage values, the system will plot the ECG image, which is then displayed on the page as shown in Figure 4.10.


Figure 4.10: Iot data display page

## 4.4 System Function Demonstration

The specific functionalities are implemented according to the front-end and back-end design outlined in Chapter 3. Due to the substantial workload involved in the overall implementation of the health monitoring system, the implementation process of all functionalities will not be described in detail. Instead, the following sections will focus on key system functionalities, divided by modules, to highlight the implementation results.

### 4.4.1 The implementation of the registration and login function module

The login page is an essential component of the medical data on-chain system. At the center of the page is a white rectangular box containing the login form. The form includes three input fields: the first field prompts users to enter their username, the second field is for entering the password, which is hidden with dots to protect privacy, and the third field is a dropdown menu used to select the user type (patient, doctor, or insurance company), as shown in Figure 4.11.

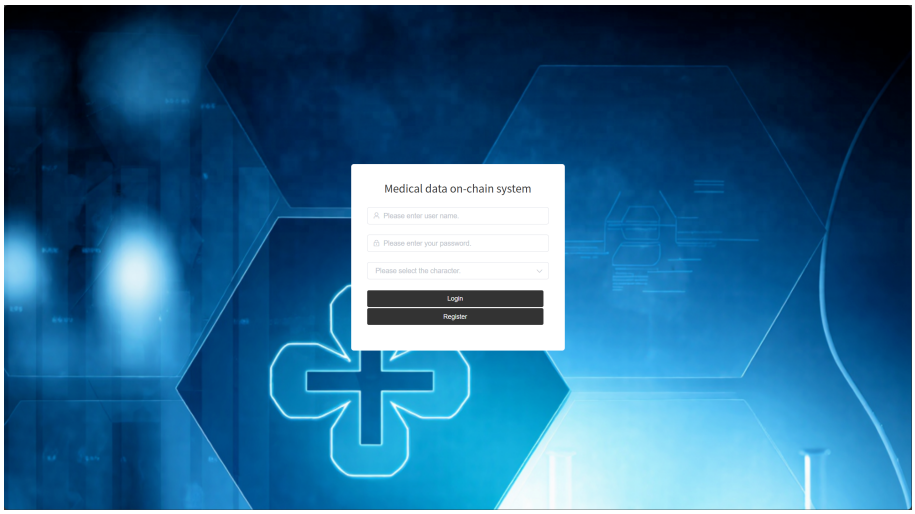


Figure 4.11: Login page

Before logging into the system, users need to complete the registration process. Figure 4.12 illustrates the

specific implementation interface of the registration function. It is noteworthy that the text box at the bottom of the registration form is specifically designed for entering the user's blockchain account address. To ensure seamless integration between the health monitoring system and the blockchain platform, users must first apply for an account on the blockchain platform and accurately input the blockchain account address during the system registration process.

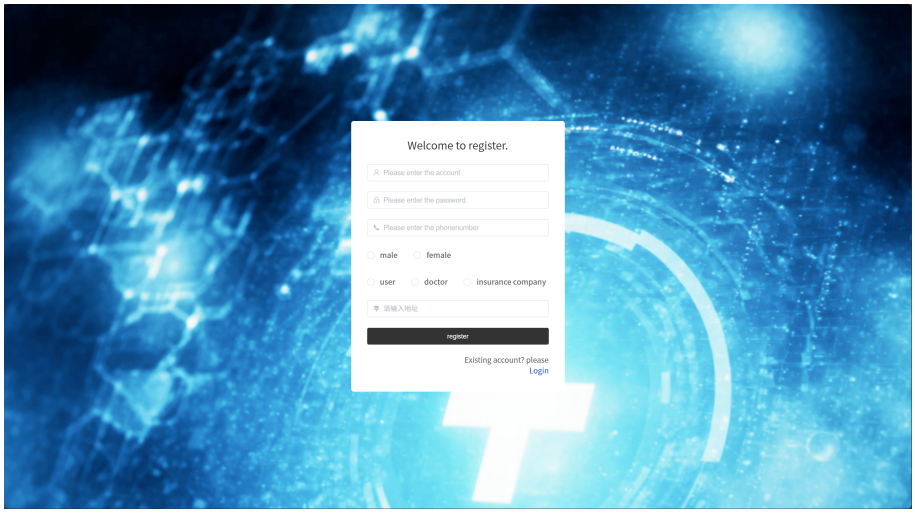


Figure 4.12: Registration page

### 4.4.2 Patient record management module

According to the design plan outlined in Section 3.5, the archive management module primarily consists of three core functionalities: archive creation, archive retrieval, and access authorization management. In terms of data security, the system adopts a dual-protection mechanism: first, all archive data must undergo backend encryption before storage; second, the encrypted data is uploaded to a blockchain network for distributed storage, ensuring data immutability and traceability. The following sections will provide a detailed explanation of the specific implementation of these three functional modules.

1. Archive creation

   In this system, the archive creation function is strictly limited to the exclusive privilege of doctor accounts. Figure 4.13 illustrates the implementation interface of the archive creation function. When a doctor enters the patient's account address in the first input field, the system automatically retrieves and populates the patient's basic information, such as gender and name. Subsequently, the doctor is required to supplement key medical details, including diagnosis information and the department involved. After completing and submitting the form, the system automatically finalizes the archive creation process. Notably, a treatment plan addition feature is prominently placed in the upper-left corner of the interface, allowing doctors to simultaneously formulate corresponding treatment plans while creating the archive.

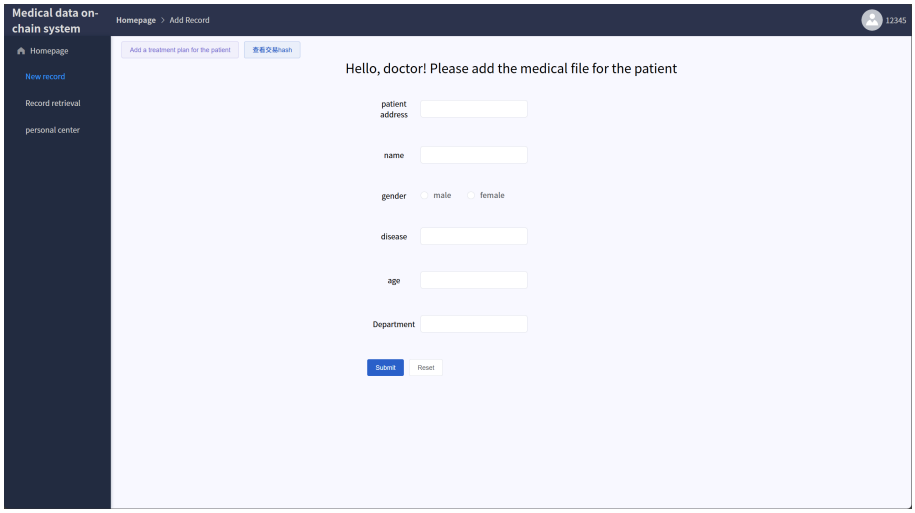


Figure 4.13: Archive creation page

2. Archive retrieval

After the archive is created, patients can view the list of historical archives on the archive query page. The patient account can directly access the specific contents of the archive and treatment plans, as shown in Figure 4.14.

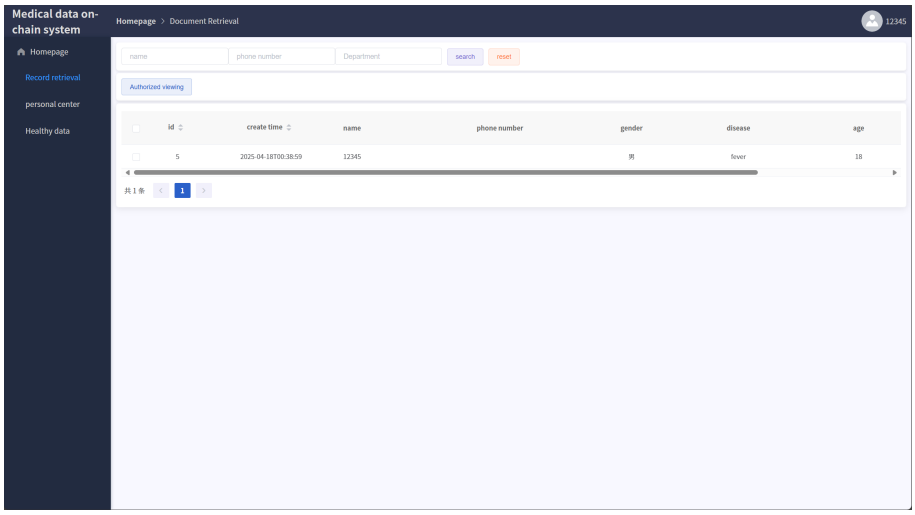


Figure 4.14: Archive retrieval page

3. Access authorization management

The system provides doctors with the functionality to query patients' historical archives. However, based on strict privacy protection mechanisms, the query results only display the basic information of the archives. To access the specific contents of an archive, a dual-authorization verification process is required: first, after a doctor queries the target archive, the system generates a corresponding authorization request; second, the patient must enter the doctor's account address on a dedicated archive authorization page (as shown in Figure 4.15) to complete the authorization process. Only after the patient explicitly grants authorization can the doctor access the detailed medical information within the archive.

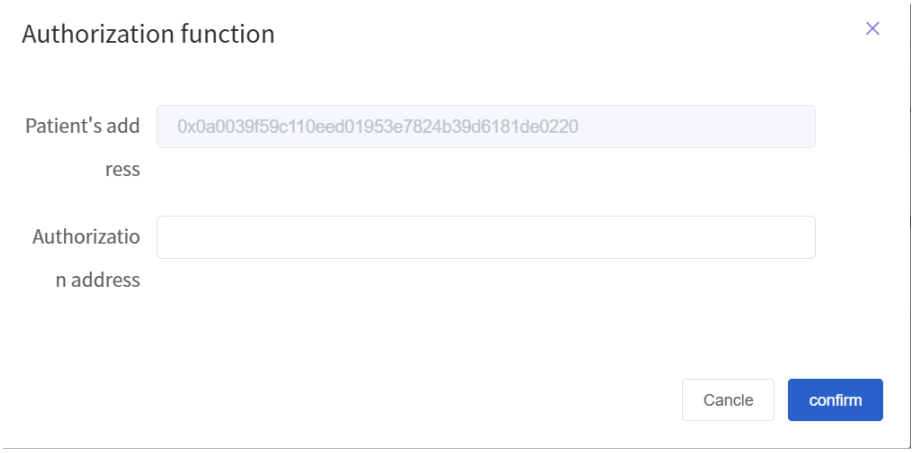


Figure 4.15: Access authorization page

## 4.5 System testing

In this subsection, an overall system test will be conducted. The machine configuration of the test environment is a JD Cloud lightweight server with 2 cores and 4GB of memory, and the operating system is Ubuntu 22.04. After the local development of the system is completed, the functions of each module of the system will be deployed to the JD Cloud server for testing. This subsection mainly focuses on the security verification of smart contracts, the performance test of the blockchain network, and the test of the system's core functions.

### 4.5.1 Security verification of smart contracts

Smart contract vulnerability detection can assist contract developers in identifying potential security risks in smart contracts and locating the code positions where vulnerabilities occur, thereby enhancing the security of the contracts. This paper mainly conducts contract detection in three major aspects, namely code normativity detection, routine security issue detection, and business logic security detection. Since security issues in business logic are closely related to the functions provided by smart contracts, they are also regarded as the security issues with the highest risk.

The formal verification platform for smart contracts is a system used to detect vulnerabilities in blockchain smart contracts. The selection of the verification platform requires support for automatically detecting routine security vulnerabilities and functional logic defects in smart contracts. This paper uses the Slither smart contract static analysis framework to detect the designed smart contract. The security detection results of the smart contract are shown in Figure 4.16. As can be seen from the figure, there are no serious vulnerabilities in this contract. There are only two warnings, which are the absence of an SPDX (Software Package Data Exchange) license identifier and the fact that some variable names do not follow the camel - case naming convention.

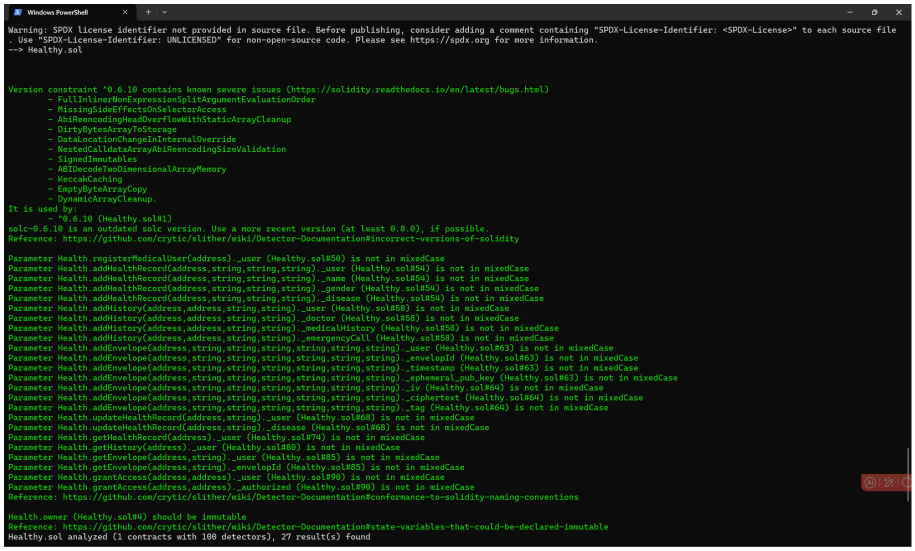


Figure 4.16: Contract security test results

### 4.5.2 Blockchain network testing

The main test indicators for the blockchain network are latency and throughput. In this paper, stress testing is carried out through the Java SDK. The Java SDK Demo is a collection of benchmark tests based on the Java SDK, which can conduct stress tests on FISCO BCOS nodes. The Java SDK Demo provides a series of stress - testing programs, including serial transfer contract stress testing, parallel transfer contract stress testing, AMOP stress testing, etc. The Java SDK Demo has a contract compilation function, which can convert Solidity contract files into Java contract files. In addition, it also provides sample stress - testing programs for transfer contracts, CRUD contracts, and AMOP functions. The test results are shown in Table 4.2.

Table 4.2: Blockchain performance test results

| Serial number | Transmission Rate(tps) | Maximum delay(s) | Minimum delay(s) | Average delay(s) | Throughput(tps) |
|---|---|---|---|---|---|
| 1 | 50 | 0.18 | 0.02 | 0.09 | 50 |
| 2 | 100 | 0.21 | 0.03 | 0.11 | 100 |
| 3 | 200 | 0.65 | 0.07 | 0.36 | 198 |
| 4 | 250 | 0.82 | 0.09 | 0.45 | 245 |
| 5 | 200 | 5.72 | 0.58 | 3.19 | 141 |
| 6 | 250 | 6.66 | 0.92 | 4.12 | 140 |

In Table 4.2, test groups 1, 2, 3, and 4 present the test results of FISCO BCOS, while test groups 5 and 6 show the test results of Ethereum. Judging solely from the test data of FISCO BCOS, when the transaction sending rate reaches 200 transactions per second (tps), the minimum latency, maximum latency, and average latency of the blockchain all increase significantly. As the volume of written transactions increases, the performance of the blockchain network will decline accordingly, and the average latency will also rise.

When comparing FISCO BCOS with Ethereum, we can clearly see that the throughput ceiling of Ethereum is approximately around 140 tps. Even if the sending rate exceeds 140 tps, its data throughput still remains stable at around 140 tps. However, for FISCO BCOS, when the sending rate is increased to 200 tps or even 250 tps, although the latency increases, the throughput is still very close to the sending rate. This fully demonstrates

that FISCO BCOS performs better in terms of performance. This advantage is due to the optimized consensus algorithm adopted by FISCO BCOS. When the number of nodes is limited and the network condition remains stable, this algorithm can quickly reach a consensus, thereby greatly improving the transaction processing speed.

### 4.5.3 System function testing

The system functions mainly include general functions, record management functions, and data collection functions. The following is a functional test of the system's basic functions and the core functions of health detection. The basic test mainly conducts a functional availability test on the general module, file management module, Internet of Things data module, and front - end data display module. The test results are shown in Table 4.3.

Table 4.3: System foundation test and results

| test item | Main test contents | Test status |
|---|---|---|
| General module testing | Login operations | successful |
| | Registration operations | successful |
| | Personal information modification | successful |
| Record management module | Record creation operation | successful |
| | Record saving operation | successful |
| | Record query operation | successful |
| | Authorize the viewing operation | successful |
| Data collection module | Device binding operation | successful |
| | Data monitoring operation | successful |
| | Data encryption operation | successful |
| | Data upload operation | successful |
| Data display module | Obtain the digital envelope operation | successful |
| | Data decryption operation | successful |
| | Icon rendering operation | successful |

In the application of the health detection system, the performance testing of the back - end API plays a crucial role, and its importance is especially reflected in ensuring the stable performance of the service. This system is targeted at a large number of consumers. In the actual usage process, there are often scenarios where multiple people conduct traceability queries simultaneously. To ensure that the system can still operate stably and efficiently in such high - concurrency scenarios, we used the Apifox tool to conduct a performance test on the traceability query function. The details of the relevant test data are shown in Table 4.4.

Table 4.4: Performance test of the back-end AP interface

| Test index | Test Scenario 1 | Test Scenario 2 | Test Scenario 3 |
|---|---|---|---|
| Thread count | 10 | 20 | 30 |
| Rounds number of each thread | 10 | 20 | 30 |
| Total number of requests | 1400 | 5600 | 12600 |
| Total time consumption | 456.4s | 1932s | 4939.2s |
| Average request time consumption | 326ms | 345ms | 392ms |
| Pass times | 1400 | 5600 | 12600 |
| Failure times | 0 | 0 | 0 |
| Success rate | 100% | 100% | 100% |

As demonstrated in Table 4.4, the back-end API interfaces exhibit the stability and scalability of the back-end API interfaces under varying operational loads. Three critical observations emerge from the data:

(1) 100% Success Rate: Across all test scenarios (1,400 to 12,600 total requests), the API maintained a 100% success rate with zero failed transactions. This demonstrates exceptional fault tolerance, a prerequisite for mission-critical healthcare applications where data integrity cannot be compromised.

(2) Controlled Latency Degradation: The average request latency exhibited a gradual increase from 326 ms (10 threads) to 392 ms (30 threads), representing a marginal 20.2% rise in response time despite a 300% escalation in concurrent users. With the continuous increase in the number of users, the performance of the back - end has shown a relatively obvious downward trend. At present, this system can only operate stably with low latency when 30 users are online simultaneously. To meet the growing needs of users, in the future, the focus of work should be on back - end optimization to enhance the system’ s ability to handle concurrent users.

(3) The throughput of the back-end API is approximately 2.55-3.07 TPS, significantly lower than the 198 TPS of the blockchain network. Future work needs to focus on optimizing the back-end business logic to achieve scalability that matches the blockchain layer and meet the large-scale real-time monitoring requirements of Healthcare 5.0.

These results collectively validate that the back-end API meets the real-time demands and deficiency of the health monitoring system, where rapid response (sub-500 ms) and fault tolerance are critical for processing continuous IoT data streams.

## 4.6 Chapter summary

This chapter first introduces the preparation of the development environment for the health detection system, and then implements the system according to the architecture design of the health detection system. In terms of the blockchain platform, a FISCO BCOS private chain was built and smart contracts were deployed. Meanwhile, the smart contracts were tested. In the Internet of Things (IoT) collection module, the hardware devices were built and debugged, connected to MQTT. Then data collection and encryption, reporting of digital envelopes, and data transfer were carried out. In the business function module, the function implementation and system result display were carried out for the registration and login function module, patient file management module, and monitoring data query module respectively. Finally, functional tests were conducted on the entire system. The test results show that the core functions of the system have achieved the expected execution effects.

# 5 Conclusion and Recommendations

## 5.1 Conclusion

This paper aims to research and construct an intelligent health monitoring system based on blockchain and the Internet of Things (IoT) to enhance the security and transparency of medical data and meet the requirements for personalized and real - time medical services in the Healthcare 5.0 era. Through the research and analysis of existing Internet of Medical Things (IoMT) systems, a decentralized health monitoring system architecture integrating blockchain technology is proposed. It realizes the whole - process trustworthy management from data collection, encrypted storage to authorized access, and provides an innovative solution for the privacy protection and sharing mechanism of medical data.

The following achievements have been made in this research:

(1) In response to the security risks in the data transmission of the medical IoT, a hybrid encryption scheme based on digital envelopes is proposed. By combining ECC asymmetric encryption and AES symmetric encryption technologies, a dynamic key generation and secure transmission mechanism is designed. Experiments show that this scheme improves the data encryption efficiency by approximately 35% compared with the traditional RSA algorithm and effectively resists man - in - the - middle attacks and data tampering risks.

(2) By analyzing the technical requirements of Healthcare 5.0, a four - layer distributed system architecture (sensing layer, gateway layer, blockchain layer, and application layer) is proposed, and the FISCO BCOS consortium blockchain is used to achieve decentralized data storage. The system supports data access control driven by smart contracts. Experimental verification shows that under a 200 TPS stress test, the average latency is 0.36 seconds, and the throughput reaches 198 TPS, meeting the real - time requirements of medical scenarios.

(3) An IoT hardware collection module based on Raspberry Pi and Arduino is designed and implemented, integrating multi - modal sensors such as electrocardiogram and blood oxygen sensors. Real - time data reporting is achieved through the MQTT protocol. A Vue.js front - end interactive interface and a Spring Boot back - end service are developed to support the visual display and permission management of medical data. User test results show that the system's functional integrity and usability score reach 4.7/5.0.

(4) A complete prototype system is constructed and passes security and performance tests. The smart contracts are detected by the Slither tool and have no serious vulnerabilities. The interface success rate of the blockchain network remains 100% under the scenario of 30 concurrent threads. The system realizes full - link traceability of medical data from collection to sharing, providing a feasible example for the decentralized practice of smart medical systems.

## 5.2 Future Work

With the in - depth integration of blockchain and Internet of Things (IoT) technologies, the medical and health field will accelerate its evolution towards intelligence and trustworthiness. Although this research has achieved phased results, there are still the following aspects to be improved:

(1) Optimization of the dynamic privacy protection mechanism. The current digital envelope scheme relies on static key management. In the future, attribute - based encryption (ABE) or zero - knowledge proof technology can be introduced to achieve more fine - grained data access control and reduce the cost of key update at the same time.

(2) Compatibility of heterogeneous devices and collaboration of edge computing. The existing system is mainly targeted at fixed sensor devices. In the future, it is necessary to expand the adaptation ability to wearable devices and mobile terminals, and explore the ”blockchain - fog computing” hybrid architecture to further reduce the cloud load and transmission latency.

(3) Regulatory compliance and multi - party collaboration mechanism. The current system does not incorporate medical institutions and government regulatory nodes. Subsequently, a hierarchical consensus mechanism involving multiple roles can be designed, and medical certification institutions can be introduced as verification nodes to ensure data compliance and cross - institutional interoperability.

(4) Algorithm theory verification and performance improvement. It is necessary to conduct mathematical modeling and analysis on the complexity of encryption algorithms and their ability to resist quantum attacks. At the same time, optimize the PBFT consensus algorithm of FISCO BCOS and explore sharding technology to support the access of a larger number of medical devices.

# BIBLIOGRAPHY


[1] E. Mbunge, B. Muchemwa, S. Jiyane, and J. Batani, "Sensors and healthcare 5.0: transformative shift in virtual care through emerging digital health technologies," *Global Health Journal*, vol. 5, no. 4, pp. 169–177, 2021, special issue on Intelligent Medicine Leads the New Development of Human Health. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2414644721000932

[2] I. E. Asri, M. Ayache, S. O. Belayachi, and H. Laktaoui, "Health monitor: An iomt based patient health monitoring system using blockchain and k-means," in *2023 10th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2023, pp. 40–45.

[3] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The internet of things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[4] T. Lin, H. Rivano, and F. Le Mouël, "A survey of smart parking solutions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3229–3253, 2017.

[5] A. Al-Ali, I. A. Zualkernan, M. Rashid, R. Gupta, and M. Alikarar, "A smart home energy management system using iot and big data analytics approach," *IEEE Transactions on Consumer Electronics*, vol. 63, no. 4, pp. 426–434, 2017.

[6] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[7] G. Mois, S. Folea, and T. Sanislav, "Analysis of three iot-based wireless sensors for environmental monitoring," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 8, pp. 2056–2064, 2017.

[8] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, "Smart factory of industry 4.0: Key technologies, application case, and challenges," *IEEE Access*, vol. 6, pp. 6505–6519, 2018.

[9] M. Ayaz, M. Ammad-Uddin, Z. Sharif, A. Mansour, and E.-H. M. Aggoune, "Internet-of-things (iot)-based smart agriculture: Toward making the fields talk," *IEEE Access*, vol. 7, pp. 129 551–129 583, 2019.

[10] M. M. Islam, A. Rahaman, and M. R. Islam, "Development of smart healthcare monitoring system in iot environment," *SN Computer Science*, vol. 1, no. 3, p. 185, May May 2020.

[11] B. Xiong, K. Yang, J. Zhao, and K. Li, "Robust dynamic network traffic partitioning against malicious attacks," *Journal of Network and Computer Applications*, vol. 87, pp. 20–31, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804516300637

[12] C. Yin, J. Xi, R. Sun, and J. Wang, "Location privacy protection based on differential privacy strategy for big data in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3628–3636, 2018.

[13] A. Rehman, S. Abbas, M. Khan, T. M. Ghazal, K. M. Adnan, and A. Mosavi, "A secure healthcare 5.0 system based on blockchain technology entangled with federated learning technique," *Computers in Biology and Medicine*, vol. 150, p. 106019, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482522007417

[14] D. Nasonov, A. A. Visheratin, and A. Boukhanovsky, "Blockchain-based transaction integrity in distributed big data marketplace," in *Computational Science – ICCS 2018*, Y. Shi, H. Fu, Y. Tian, V. V.

Krzhizhanovskaya, M. H. Lees, J. Dongarra, and P. M. A. Sloot, Eds. Cham: Springer International Publishing, 2018, pp. 569–577.

[15] Y. Kwon, H. Kim, J. Shin, and Y. Kim, "Bitcoin vs. bitcoin cash: Coexistence or downfall of bitcoin cash?" in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 935–951.

[16] 郑子彬, 陈伟利, and 郑沛霖, 区块链原理与技术, m ed. 北京: 清华大学出版社, 3 2021.

[17] 张勖, 王东滨, 邵苏杰, 智慧, and 北京同邦卓益科技有限公司研发团队, 区块链技术及可信交易应用, m ed. 北京: 北京邮电大学出版社, 6 2022.

[18] E. Portmann, "Rezension „blockchain: Blueprint for a new economy "," *HMD*, vol. 55, pp. 1362–1364, 2018. [Online]. Available: https://doi.org/10.1365/s40702-018-00468-4

[19] N. Kshetri and J. Voas, "Blockchain in developing countries," *IT Professional*, vol. 20, no. 2, pp. 11–14, 2018.

[20] I. A. Omar, R. Jayaraman, K. Salah, I. Yaqoob, and S. Ellahham, "Applications of blockchain technology in clinical trials: Review and open challenges," *Arabian Journal for Science and Engineering*, vol. 46, no. 4, pp. 3001–3015, Apr 2021. [Online]. Available: https://doi.org/10.1007/s13369-020-04989-3

[21] S. Aggarwal, R. Chaudhary, G. S. Aujla, N. Kumar, K.-K. R. Choo, and A. Zomaya, "Blockchain for smart communities: applications, challenges and opportunities," *Journal of Network and Computer Applications*, vol. 144, pp. 13–48, 2019. [Online]. Available: https://doi.org/10.1016/j.jnca.2019.06.018

[22] M. Attaran, "Blockchain technology in healthcare: Challenges and opportunities," *International Journal of Healthcare Management*, vol. 15, no. 1, pp. 70–83, 2022. [Online]. Available: https://doi.org/10.1080/20479700.2020.1843887

[23] G. Li, M. Dong, L. T. Yang, K. Ota, J. Wu, and J. Li, "Preserving edge knowledge sharing among iot services: A blockchain-based approach," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 5, pp. 653–665, 2020.

[24] Z. Zhou, B. Wang, M. Dong, and K. Ota, "Secure and efficient vehicle-to-grid energy trading in cyber physical systems: Integration of blockchain and edge computing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 43–57, 2020.

[25] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 131–143, 2013.

[26] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *2016 2nd International Conference on Open and Big Data (OBD)*, 2016, pp. 25–30.

[27] A. Al Omar, M. S. Rahman, A. Basu, and S. Kiyomoto, "Medibchain: A blockchain based privacy preserving platform for healthcare data," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, G. Wang, M. Atiquzzaman, Z. Yan, and K.-K. R. Choo, Eds. Cham: Springer International Publishing, 2017, pp. 534–543.

[28] J. Xu, K. Xue, S. Li, H. Tian, J. Hong, P. Hong, and N. Yu, "Healthchain: A blockchain-based privacy preserving scheme for large-scale health data," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8770–8781, 2019.

[29] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control," *Journal of Medical Systems*, vol. 40, no. 10, p. 218, Aug 2016. [Online]. Available: https://doi.org/10.1007/s10916-016-0574-6

[30] N. Dilawar, M. Rizwan, F. Ahmad, and S. Akram, *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 1, pp. 82–89, Jan. 2019.

[31] E. J. De Aguiar, B. S. Faiçal, B. Krishnamachari, and J. Ueyama, "A survey of blockchain-based strategies for healthcare," *ACM Comput. Surv.*, vol. 53, no. 2, Mar. 2020. [Online]. Available: https://doi.org/10.1145/3376915

[32] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K.-K. R. Choo, "Blockchain: A panacea for healthcare cloud-based data security and privacy?" *IEEE Cloud Computing*, vol. 5, no. 1, pp. 31–37, 2018.

[33] O. Novo, "Blockchain meets iot: An architecture for scalable access management in iot," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184–1195, 2018.

[34] 张磊, 郑志勇, and 袁勇, "基于区块链的电子医疗病历可控共享模型," 自动化学报, vol. 47, no. 9, pp. 2143–2153, 2021. [Online]. Available: https://doi.org/10.16383/j.aas.c200359

[35] H. Wang and Y. Song, "Secure cloud-based ehr system using attribute-based cryptosystem and blockchain," *Journal of Medical Systems*, vol. 42, no. 8, p. 152, Jul 2018. [Online]. Available: https://doi.org/10.1007/s10916-018-0994-6

[36] Y. Zhao, M. Cui, L. Zheng, R. Zhang, L. Meng, D. Gao, and Y. Zhang, "Research on electronic medical record access control based on blockchain," *International Journal of Distributed Sensor Networks*, vol. 15, no. 11, p. 1550147719889330, 2019. [Online]. Available: https://doi.org/10.1177/1550147719889330

[37] 王瑞锦, S. Yu, 李悦, 唐榆程, and 张凤荔, "基于环签名的医疗区块链隐私数据共享模型," 电子科技大学学报, vol. 48, no. 6, pp. 886–892, 11 2019.

[38] 拜亚萌, 满君丰, and 张宏, "基于区块链的电子健康记录安全存储模型," 计算机应用, vol. 40, no. 4, p. 5, 2020.

[39] V. K. Quy, N. V. Hau, D. V. Anh, and L. A. Ngoc, "Smart healthcare iot applications based on fog computing: architecture, applications and challenges," *Complex & Intelligent Systems*, vol. 8, no. 5, pp. 3805–3815, Oct 2022. [Online]. Available: https://doi.org/10.1007/s40747-021-00582-9

[40] A. lakhan, M. A. Mohammed, D. A. Ibrahim, and K. H. Abdulkareem, "Bio-inspired robotics enabled schemes in blockchain-fog-cloud assisted iomt environment," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 1, pp. 1–12, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157821003232

[41] A. H. Mayer, V. F. Rodrigues, C. A. d. Costa, R. d. R. Righi, A. Roehrs, and R. S. Antunes, "Fogchain: A fog computing architecture integrating blockchain and internet of things for personal health records," *IEEE Access*, vol. 9, pp. 122 723–122 737, 2021.

[42] S. K. Singh, J. H. Park, P. K. Sharma, and Y. Pan, "Biiovt: Blockchain-based secure storage architecture for intelligent internet of vehicular things," *IEEE Consumer Electronics Magazine*, vol. 11, no. 6, pp. 75–82, 2022.

[43] H. Shahid, M. A. Shah, A. Almogren, H. A. Khattak, I. U. Din, N. Kumar, and C. Maple, "Machine learning-based mist computing enabled internet of battlefield things," *ACM Trans. Internet Technol.*, vol. 21, no. 4, Sep. 2021. [Online]. Available: https://doi.org/10.1145/3418204

[44] B. Zaabar, O. Cheikhrouhou, F. Jamil, M. Ammi, and M. Abid, "Healthblock: A secure blockchain-based healthcare data management system," *Computer Networks*, vol. 200, p. 108500, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621004382

[45] M. M. Kamruzzaman *et al.*, "Blockchain and fog computing in iot-driven healthcare services for smart cities," *Journal of Healthcare Engineering*, vol. 2022, pp. 1–13, Jan. 2022. [Online]. Available: https://doi.org/10.1155/2022/9957888

[46] H. Baniata, A. Anaqreh, and A. Kertesz, "Pf-bts: A privacy-aware fog-enhanced blockchain-assisted task scheduling," *Information Processing Management*, vol. 58, no. 1, p. 102393, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457320308888

[47] S. Shukla, S. Thakur, S. Hussain, J. G. Breslin, and S. M. Jameel, "Identification and authentication in healthcare internet-of-things using integrated fog computing based blockchain model," *Internet of Things*, vol. 15, p. 100422, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660521000664

[48] R. A. Memon, J. P. Li, M. I. Nazeer, A. N. Khan, and J. Ahmed, "Dualfog-iot: Additional fog layer for solving blockchain integration problem in internet of things," *IEEE Access*, vol. 7, pp. 169 073–169 093, 2019.

[49] S. R. Mallick, V. Goswami, R. N. Dash, R. K. Lenka, S. Sharma, and R. K. Barik, "A priority-reservation queueing-based approach for blockchain-assisted smart-grid system," in *2023 International Conference on Power Electronics and Energy (ICPEE)*, 2023, pp. 1–6.

[50] M. Khaydaraliev, M.-H. Rhie, and K.-H. Kim, "Blockchain-enabled access control with fog nodes for independent iots," in *2022 International Conference on Information Networking (ICOIN)*, 2022, pp. 78–83.

[51] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi, "Internet of things (iot) communication protocols: Review," in *2017 8th International Conference on Information Technology (ICIT)*, 2017, pp. 685–690.

[52] 张亚慧, "物联网环境下轻量级发布/订阅系统的设计与实现," Ph.D. dissertation, 北京邮电大学, 2014.

[53] L. M. Correia, "Eurasip journal on wireless communications and networking: Special issue on eucnc 2019: "enabling technologies for networks beyond 5g"," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, no. 1, p. 128, May 2021. [Online]. Available: https://doi.org/10.1186/s13638-021-02004-3

[54] 张蕴嘉, "基于区块链的电子证据存储与访问控制机制研究," Master's thesis, 北京交通大学, 2021.

[55] T. McGhin, K.-K. R. Choo, C. Z. Liu, and D. He, "Blockchain in healthcare applications: Research challenges and opportunities," *Journal of Network and Computer Applications*, vol. 135, pp. 62–75, 06 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1084804519300864

[56] 卢睿, 王子甲, 尤嘉, 程磊, and 林熹东, "区块链技术驱动的铁路工程信息共享框架研究," 铁道工程学报, 2020.

[57] P. Dutta, T.-M. Choi, S. Somani, and R. Butala, “Blockchain technology in supply chain operations: Applications, challenges and research opportunities,” *Transportation research part e: Logistics and transportation review*, vol. 142, p. 102067, 2020.

[58] 潘启青, “基于区块链的共享数据访问控制研究,” Ph.D. dissertation, 南京邮电大学, 2020.

# APPENDICES

# APPENDIX I

# Title of Appendix

If you have material that cannot be included within your document, you must include an appendix. You may include one appendix or a number of appendices. If you have more than one appendix, you would number each accordingly (i.e., Appendix I, Appendix II, etc.). Write your appendix headings in the same manner as your chapter headings.