Design and Implementation of a Credible Blockchain-based E-health Records Platform

By:

Lingyu Xu (x_lingyu@sina.com)

A project submitted in fulfillment of the requirements

for the degree of

Master of Science

Department of Computer Science,

University of the Western Cape

Supervisors:

Prof. Antoine Bagula

Dr. Omowunmi Isafiade

May 2020

Acknowledgements

I would like to express my gratitude to all those who helped me during the writing of this thesis.

My deepest gratitude goes first and foremost to Professor Antoine Bagula, my supervisor, for his constant encouragement and guidance. He has walked me through all the stages of the writing of this thesis. Without his consistent and illuminating instruction, this thesis could not have reached its present form.

Second, I would like to express my heartfelt gratitude to Dr Omowunmi Isafiade for her support, and Post-Doctoral fellow: Dr Olasupo Ajayi, who led me into the world of translation. English is not my first language, I had some challenges with writing in English and they helped with structure and grammar construct.

Last, my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years. I also owe my sincere gratitude to my friends and my fellow classmates who gave me their help and time in listening to me and helping me work out my problems during the difficult course of the thesis.

Abstract

With the development of information and network technologies, Electronic Health Records (EHRs) management system has gained wide spread application in managing medical records. One of the major challenges of EHRs is the independent nature of medical institutions. This non-collaborative nature puts a significant barrier between patients, doctors, medical researchers and medical data. Moreover, unlike the unique and strong anti-tampering nature of traditional paper-based records, electronic health records stored in centralization database are vulnerable to risks from network attacks, forgery and tampering. In view of the data sharing difficulties and information security problems commonly found in existing EHRs, this dissertation designs and develops a credible Blockchain-based electronic health records (CB-EHRs) management system. To improve security, the proposed system combines digital signature (using MD5 and RSA) with Role-Based Access Control (RBAC). The advantages of these are strong anti-tampering, high stability, high security, low cost, and easy implementation. To test the efficacy of the system, implementation was done using Java web programming technology. Tests were carried out to determine the efficiency of the Delegated Byzantine Fault Tolerance (dBFT) consensus algorithm, functionality of the RBAC mechanism and the various system modules. Results obtained show that the system can manage and share EHRs safely and effectively. The expectation of the author is that the output of this research would foster the development and adaptation of EHRs management system.

Keywords - Electronic Health Records, Blockchain, Data Sharing, Privacy Protection, CB-EHRs

Table of Contents

List of Figures

List of Tables

Glossary

Apache is the most widely used web server software.

ASP.Net is an open source web framework, created by Microsoft, for building modern web apps and services that run on macOS, Linux, Windows, and Docker.

Byzantine General problem is term etched from the computer science description of a situation where involved parties must agree on a single strategy in order to avoid complete failure

C ++ is a computer programming language created in 1983 and designed to serve as an enhanced version of the C language.

Consensus mechanism refer to the process of attaining a unified agreement (consensus) on the state (status) of the network in a decentralized way.

Cascading Style Sheet(CSS) is a way to design a website, or a group of websites.

Decentralized system is a public, decentralized system does not rely on any single authority and is self-regulated.

Distributed System is a collection of computers that act, work, and appear as one large computer.

Ethereum is a peer-to-peer network of virtual machines that any developer can use to run distributed applications.

Hash function is a method of computer error checking and data organization.

HyperText Markup Language (HTML) is a computer language devised to allow website creation.

HyperText Transfer Protocol (HTTP) is the underlying protocol used by the World Wide Web to define how messages are formatted and transmitted.

J2EE is a platform-independent, Java-centric environment from Sun for developing, building and deploying Web-based enterprise applications online.

Java is a high-level object-oriented programming language.

JavaScript (JS) is a scripting languages, primarily used on the Web.

JSON (JavaScript Object Notation) is a lightweight data-interchange format.

Linux is an open-source operating system modelled on Unix.

MD5 is a type of algorithm that is known as a cryptographic hash algorithm.

Merkle tree is a specific type of data construct in which each non-leaf node of the tree contains hash values of its own child nodes.

Model View Controller (MVC) is a technique used to isolate different elements of the software process.

MySQL is a database management system.

MyEclipse is a Java Integrated Development Environment (IDE) for the enterprise with the latest tools and technologies.

Node.js is an event-driven server-side JavaScript environment for building network apps, including web apps.

Smart contract is a contract created and maintained on a blockchain platform.

Timestamp is a time registered to a file, log, or notification that records when data is added, removed, modified, or transmitted.

Virtual currency is any currency that people can use in virtual environments like gaming and social networking sites.

Wide Area Network (WAN) is a collection of computers and devices connected by a communications network over a wide geographic area.

eXtensible Markup Language (XML) is an open, text based markup language that provides structural and semantic information to data. Acronyms and

Abbreviations

B / S: Browser/Server

CB-EHRs: Credible Blockchain-based E-health Records

CSS: Cascading Style Sheet

CPU: Central Processing Unit

dBFT: Delegated Byzantine Fault Tolerance

DDOS: Distributed denial-of-service attack

DHCP: Decentralized Hospital Communication Program

DNS: Domain Name System

DOS: Denial-of-Service:

DPoS: Delegated Proof of Stake

EHRs: Electronic Health Records

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

IP: Internet Protocol

JS: JavaScript

JSON: JavaScript Object Notation

LAN: Local Area Network

MVC: Model View Controller

NIST: National Institute of Standards and Technology

P2P: Peer-to-Peer

PBFT: Practical Byzantine Fault Tolerance

PHRs: Paper Health Records

PoS: Proof of Stake

PoW: Proof of Work

R&D: Research and Development

RAM: Random Access Memory

RBAC: Role-Based Access Control

REST: Representation State Transfer

RPCA: Ripple Protocol of Consensus Algorithm

SHA: Secure Hash Algorithm

TPS: Transaction Per Second

UNL: Unique Node List

WAN: Wide Area Network

WEF: World Economic Forum

XML: eXtensible Markup Language

1. Introduction

1.1. Background and Motivation

The development of information technology has a huge impact on people's lifestyles. This is further enhanced by its potentials for data dissemination and information sharing, which are particularly prominent. “We Are Social and Hootsuite”[1] recently published their 2018 Global digital suite of reports, visualizing data about Internet usage around the world. The global overview and in-depth breakdowns of Internet usage on a country-by-country basis were detailed in the reports, including data on South Africa. Citing the International Telecommunications Union, it was reported that South Africa added 2 million Internet users in 2017, growing to 30.81 million, and placing our Internet penetration at 54% [1]. In South Africa, users spend an average of 46% of their time working, learning, and playing on the Internet. The Internet has become the main source of information for most people. With respect to medicine, information technology has seen application in medical diagnosis as well as medical information exchange. Information Technology not only supports information dissemination but allows for support and interaction between medical institutions.

Mobile apps for healthcare and disease management are also on the rise. A number of start-up companies and entrepreneurs now focus on building apps for Internet or Cloud hospitals via mobile platforms [2]. A major objective is to build "Internet + Medical" service platform with versatile apps that enable a closed-loop business model for telemedicine services or chronic disease management[3]. As an example, presently, the storage of health records generally adopts two methods, which are:(i) paper file storage; and (ii) digital data storage. Traditional paper health records (PHRs) are saved discretely. PHRs are difficult to classify effectively and are easily susceptible to loss. This has caused great inconvenience to academic researchers and medical staff[4]. Digital data storage or EHRs on the other hand do not suffer from the shortcomings of

PHR. Asides that, they have other advantages such as being comprehensive, in standardized format, easy retrieval and fast transmission speed – a feature which enables remote consultation and rural data gathering[5].

According to the 2015 annual statistical report from IMS Heal - which is the leading information provider in the global medical and health field; in the past two years, the number of medical and health applications in the Apple App Store has doubled in terms of statistics[6, 7]. As the number of applications grows, so does the variety of the medical and health applications. At the same time, the survey also pointed out that in the past two years, doctors' interest in e-medical applications has increased. More than one-third of doctors will recommend their own electronic medical software to patients. Due to the increase in the number of e-health applications, the variety and the increased adoption by medical doctors, the economic value potential of e-health market has grown affirmatively.

Investors also believe that e-health applications benefits are considerable. A well-known medical application research institute called research2guidance has investigated and predicted that the market profit of e-health applications will increase from the current 6 billion to 26 billion US dollars in 2017[8]. As e-health has been greatly promoted, the market for its related applications is booming. With this boom, a potential problem arises, which is that of security issue of e-health applications.

As an important part of e-health, EHRs provide patients with convenient and fast medical services, while at the same time facing the huge risk of leaking patient privacy information[9]. However, compared with business, financial and social services, the contribution of the Internet and information technologies to e-health is still in its infancy[10]. This therefore raises the question of how to demonstrate the enormous impact of information technology on the development of e-health; as well as how to utilize information technology to promote comprehensive changes in e-health. These are the problems that experts and scholars in the electronic medicine field need to solve. In the field of electronic medicine, the development of a centralized treatment system is hampered by public trust issues[11]. Information security has become a major concern for patients to choose online consultation[12]. There is therefore an urgent need for a decentralized information management system based on high-end Internet technology, which relies on distributed technology and makes users become part of system management. Such system would rely on secure distributed mechanisms to ensure that stored information are tamper-proof, credible and authentic[13]. To achieve these, Blockchain technology comes into play.

Blockchain technology as a distributed public ledger is revolutionizing the concepts and models of current global financial, business, public management, and e-health development[14]. Little wonder that major banks, stock exchanges, governments and medical institutions globally have invested heavily in the development and application of Blockchain technology. According to the 2015 report of the World Economic Forum, 10% of global GDP will be stored in Blockchain or Blockchain-related technologies by 2023, and Blockchain technology will be widely applied by government agencies by 2023[15].

The information stored in the Blockchain is secure and transparent[16]. Asides virtual currency transactions, Blockchain is gradually being taken seriously by the government and medical institutions in the field of electronic medicine[17]. Technology and model innovation are needed in all areas of society, thus, building a block-based trust-based information management system has become the focus of research in many countries. In 2016, Blockchain applications began to emerge and get the attention of global audience. The focus of investors with capital is only current events and economic returns. Investors have not used scientific and prudent attitudes to investigate the feasibility and usability of emerging technologies. This kind of imprecise and irrational investment has brought a market bubble to the actual development and application environment of the Blockchain. The revolutionary technology that originally promoted the development of the Internet has become a negative example of media exposure. Taking the development of Bitcoin[18] in some developed countries as an example, the market value of Bitcoin has risen more than 400% in 2017, while that of Ethereum rose by almost 13 folds[19]. Though some quarters argued that Blockchain investment is heading in the wrong direction due to the frenetic market, its real value comes from the application prospects of the technology itself[20]. This technology itself is the focus of this research work and one we consider as the correct propaganda and application of the Blockchain technology.

In stark contrast to the fiery heat of the financial industry, the process of exploring Bitcoin technology in other industries is more pragmatic. Australia, North America, and Europe are the pioneers to combine other apply Bitcoin technology in other industries. As examples – Blockchain has been applied in the energy sector, wherein energy processing facilities act as Blockchain. This enables electricity to be traded through the use of virtual currency within the Blockchain system. Electricity produced by renewable energy generation equipment can also be certified and traded in Blockchain[21]. In terms of government policy information, projects which integrate Blockchain technology to ensure transparency and integrity in projects that affect people's livelihood. Expenditure spend on projects such as agricultural production, urban and rural construction, poverty alleviation can be tracked through the Blockchain thus reducing or eliminating corruption and possibly eradicate poverty[22].

In the medical field, Blockchain can protect patient privacy by using features such as anonymity and decentralization. EHRs, DNA wallets, and drug anti-counterfeiting are all possible applications of Blockchain technology. In a report by IBM on healthcare and Blockchain in 2017, Blockchain technology was shown to be of great value in clinical trial records, regulatory compliance, and medical/health monitoring records[23]. Blockchain technology can play an important role in health management, medical device data recording, drug treatment, billing and claims, adverse event safety, medical asset management, and medical contract management.

1.2. Research Trends and Developments of Blockchain

Technology

The first well-established underlying technical support literature on the Blockchain is "Bitcoin: a peer-to-peer electronic cash system" published by Satoshi Nakamoto[24]. This literature elaborates on the method of constructing a whole new set of concepts for decentralized and trust-free peer-to-peer virtual currency circulation system. The feasibility of this method has been proved on the virtual currency represented by Bitcoin, which has been running since 2009. The representative advantage of Blockchain technology lies in the decentralized structure. By using encrypted transmission technology, timestamp, hash tree, consensus algorithm, etc., Blockchain technology enables the system to implement decentralized peer-to-peer transactions in a networked environment that users trust[25]. This kind of transaction method effectively solves a series of hidden dangers such as low stability, low user information security, high storage cost, and low storage and extraction efficiency under the centralized management rules. Blockchain technology is considered by other technology practitioners to be another computer science singularity following large/personal computers, Internet technologies, and mobile internet[26].

As the market value of Bitcoin has gradually increased globally, the Blockchain technology relied on by similar virtual currencies has also received extensive attention from investors, developers and scholars. Compared with the early concept of "Satoshi Nakamoto", the consensus mechanism of the Blockchain has been greatly improved in terms of performance and storage logic. A variety of technologies, such as peer-to-peer transactions, distributed ledgers, and contract-only codes, are constantly being optimized[27]. Therefore, the transmission of information gradually changes to the transmission of value. Based on the technical characteristics of the Blockchain and the current development of Internet technology, Swan[28] summarizes the development of Blockchain technology into three phases. These phases are as follows: phase 1.0 wherein virtual currency such as Bitcoin is popularized, phase 2.0 - the new concept of smart contract is applied in real business and in phase 3.0, Blockchain technology realizes Internet of Things management. Eventually, the global network nodes will be merged into the Blockchain system to realize true interconnection and interoperability in a safe and efficient manner.

The “Satoshi Nakamoto” Blockchain technology solved the “Byzantine General” problem[29] and built a special distributed network system. The system is a consensus system that interacts without trusting a particular node. Researchers believe that the structure of the Blockchain system is different from that of the general WAN. The structure of the Blockchain system is an organizational structure that is closer to nature and humanity. Many financial giants and R&D (Research and Development) institutions have begun to recognize the technological revolution caused by Blockchain technology and have participated in Blockchain research[30].

1.2.1. Financial Sector and Blockchain

With the continuous improvement of technology, the Blockchain has been widely used and valued in the financial field, virtual currency, and e-commerce. It’s typical representatives are: Bitcoin, Ethereum, etc. become the main futures of the bulk market[31]. The application of Blockchain technology in the virtual currency field continues to be upgraded and improved. The investment and research and development related to it gradually moved to the financial industry. At present, dozens of financial institutions including the National Association of Securities Dealers in the United States, the New York Stock Exchange, and Citigroup have launched business exploration for Blockchain[32].

In 2015, the National Association of Securities Dealers in the United States completed its securities business in a Blockchain-based trading system[33]. This system uses distributed ledgers to verify the validity of stock trading. Intermediaries and clearing organizations are no longer the necessities for bond transactions. Past securities transactions need to be done through a large number of unofficial systems and paper credentials, which means that the transaction process is not transparent. Using Blockchain technology makes financial assets and transactions completely open. Transparent information increases transaction security and improves agency-led information asymmetry.

In 2016, R3CEV, a Blockchain financial institution, released a distributed ledger based on the Ethereum and Azure Blockchain structure[34]. R3CEV also opened a shared service with financial institutions such as the Commonwealth Bank of Australia and RBS[35]. These financial institutions abandoned the traditional centralized service system and used the token assets in the decentralized ledger to simulate trading behavior. The British government invited professors of information technology from the Royal Academy of Sciences to conduct research on Blockchain technology. In the spring of 2016, professors from these academies published a British government science report entitled "Distributed Ledger Technology: Beyond Blockchain." [36]. This report provides strategic support at the government level for the application of Blockchain technology.

At the 45th Annual Economic Forum Annual Meeting held in early 2016, Blockchain, AI, quantum communication, and autopilot technologies were summarized as human strategic cooperation issues[37]. The overseas publication, reprinted by Hong Kong's "Apple Daily", said that the Blockchain is one of the top ten cutting-edge technologies with the most exploration potential in 2016. In January 2016, the People's Bank of China launched a digital currency exchange meeting to explore the feasibility and value of digital currency[38]. The Blockchain has since received wide attention from Chinese industry insiders.

Due to the profit-making of the capital market, Blockchain technology is at the forefront. However, when the Blockchain business shifts from value business to information storage, the market attention is relatively calm[39]. This is also the reason why all industries, including medical care, are very cautious about Blockchain investment and research. Even so, agencies and research teams in some countries still see the advantages and potential application value of the Blockchain. In addition to applications in the financial industry, they have conducted research on Blockchain in other areas and achieved initial results.

1.2.2. Energy Applications and Blockchain

In smart energy applications, Grid Singularity has developed a power payment/transaction system with the help of university research projects[40]. Users can transfer traditional cash to the prepaid meter of the Bitcoin system. This process does not have to rely on agencies or community groups and ensures full transparency of user payments.

In the US, L03, Konsen and Siemens operate the "Energy Transfer" project. This project utilizes Blockchain technology to enable power service system allows surplus electricity generated by household solar generators be sold to other users[41]. Germany companies RWE and Slock have developed a clean energy vehicle payment system based on Blockchain technology[42]. Buyers pay bills online instead of signing a car contract with a car supplier.

A Blockchain system has been used in the Brooklyn Smart Microgrid Agency in the United States[43]. The system connects new energy suppliers and users in the northeastern United States directly without complex manual power service operations. Filament Inc. of the United States expects to use the "Internet of Things & Blockchain" technology in Australia in the near future[44]. Through the sensing device installed on the cable, the company broadcasts real-time information to users and maintenance personnel through the Blockchain network, so that the hidden dangers of the power grid can be discovered in the first time. Using the Internet of Things, Filament company allows wireless sensing devices to interconnect and allow messages to be transmitted independently between each device. Moreover, these sensing devices can also store virtual currency information, including virtual currency protocol data, transmission information, transaction information and other information can be transmitted. Transactions between these sensors are controlled through smart contracts, so their interactions allow the transaction to execute autonomously.

The US IDEO company also practices the "Blockchain & Internet of Things" program[45]. IDEO has developed a light energy charging board that uses Blockchain technology. The charging board is connected to the US stock exchange system using the filament's consensus interface to automatically generate solar power currency while recording the generated electricity. This virtual certificate reward mechanism can drive the development of the clean energy market while ensuring transparency in the clean energy currency/subsidy market.

If the above application models are applied to the medical field, they will provide new hardware and institutional funding management thinking for large medical institutions such as hospitals. The management and maintenance of facilities of medical institutions and the material support of the society to the hospital can be realized by using the Blockchain. At the same time, problems such as redundant management of personnel and opaque flow of funds can be solved.

1.2.3. E-government and Blockchain

In January 2018, The World Economic Forum (WEF) was working with the Canadian and Dutch governments on a Blockchain-based digital identity pilot event called the “Known Traveller Digital Identity”[46]. The Known Traveller Digital Identity utilises biometrics, cryptography and distributed ledger technology to give travellers control over, and the ability to share their information with authorities in advance of travel for expedited clearance, while building trust between public authorities to improve risk and threat detection.

In May 2018, the US state of West Virginia announced that it was testing the Blockchain voting system and used same to provide simpler voting program for the Senate elections in November of the same year[47]. The program also addresses the issue of absentee voting rights in the military service and provided a more secure and anonymous method for absentees to cast their votes. Furthermore, the West Virginia House of Representatives is reported to be in the process of forming a special study group to research ways of adopting Blockchain technology across different government services[48].

In addition to the above examples, there are many other government agencies that have tried projects related to the Blockchain. This means that innovation is key to enhancing global competitiveness and productivity.

1.2.4. Education and Blockchain

In 2016, the attention, investment, and research and development of Blockchain technology in the social and academic circles reached an unprecedented level. Educational practitioners and scholars began to think about how to introduce Blockchain technology into the education industry. This would help improve and reform the educational information infrastructure, as well as to solve the problem of over reliance of education management systems on centralized management.

A number of institutions and research teams around the world have made preliminary explorations and innovative practices on the combination of Blockchain and education. The Digital Media Lab at the Massachusetts Institute of Technology has used a Bitcoin-based underlying Blockchain technology to award degrees. The college can generate a degree certificate interface in a timely manner through mobile device applications and can also view and print relevant degree information[49]. The Hobart Software Engineering Institute in the United States awarded course-oriented certification data through Blockchain technology in 2017, and in the same year began sharing academic-related information on the public chain[50].

1.3. Problems and Research Status of EHRs System

1.3.1. Development Status and Existing Problems of EHRs System

Reviewing the historical development of the EHRs system, the United States has been building EHRs system for many years before other countries. In 1960, the

Massachusetts General Hospital was a pioneer in the development of outpatient EHRs. A landmark project in the development of EHRs was the Decentralized Hospital Communication Program (DHCP) developed by the US government for the Department of Veterans Affairs in the mid-1980s[51]. DHCP enables all veteran hospitals to share medical information. In September 1993, the first international conference on health care systems was held in Marseille, France, to study the application and development of the system[51]. In 1995, the Ministry of Health and Welfare of Japan established the Electronic Health Record Development Committee, and invested 290 million yen in the same year to develop EHRs[51]. Subsequently, many other countries have spent a lot of resources to develop and/or improve the functions of EHRs.

At present, the application scope of EHRs has already covered clinical decision-making, medical education, scientific research literature retrieval, patient services, hospital information system construction and planning, medical insurance, and remote consultation in developed countries such as Europe and the United States. By studying some existing EHRs systems, the following summarizes the problems of existing EHR systems into two main areas – technical and practical.

1.3.1.1. Technical Field

There are many ways to implement EHR management. Some are based on ASP.Net and XML technologies, MVC and J2EE technologies, while others are based on more advanced technologies. The technologies used in these systems are quite different, but the structures adopted by these systems are traditional Browser/Server structure and WEB technologies.

The Browser/Server structure is a widely used network structure model after the rise of WEB technology. This traditional mode unifies the client and centralizes the implementation core of the system functions to the server[52]. It simplifies the process and operation of system development and maintenance. Though convenient, the centralized nature suffers from some inevitably challenges. Some of which include:

A. Single point of failure

The centralized structure combines all the core functions together, which leads to an increased chance of a single point of failure in the system. The failure of the central node can easily lead to the collapse of the entire system. The centralized system thus has lower system reliability and security. Hence, in order to improve the reliability of the central node, it is necessary to maintain and update the central node. However, as the size of the system expands, these costs will increase dramatically.

B. Distributed denial-of-service attack (DDOS)

Denial-of-service (DOS) attack is a common cyber-attack. It is a method of attack that denies the target node in a network access to system resources so that the service is temporarily interrupted or stopped[53]. Therefore, normal users cannot access the system being attacked.

When an attacker uses two or more controlled nodes to perform a denial of service attack on the target node, it is called DDOS. DDOS attacks can be classified into two types. One is bandwidth-consuming attack, and the other is resource-consuming attack[54]. All of the above attack methods first create a large number of legitimate or forged requests. Then these requests cause a large amount of network bandwidth or system resources of the nodes to be occupied. Eventually, the network reaching the target node becomes unavailable or the node's resources are exhausted and the system crashes.

The central node is the core of all the functions in a centralized system. Due to the existence of the central node, the success rate of denial of service attacks is higher. Because the attacker only needs to attack the central node and the entire system can be paralyzed.

1.3.1.2.Practical Application Field

The Browser/Server structure is a centralized structure. In this structure, the central node has a resource advantage and bound to become the focus of each node's contention. When a node gains the status of a central node, its abuse of resource becomes inevitable. The use of a centralized structure makes the system administrator's rights unlimited. The status of the administrator will become the focus of the competition, attracting malicious attacks. At the same time, due to the infinite power of administrators, the phenomenon of abuse of power would also appear frequently[55]. In this digital age, assets are gradually being digitized. Most of the time our assets, such as bank deposits, are just a bunch of records in the database and a number displayed on a page. We choose to trust the central node of the bank and give our assets to them for preservation. But no one can guarantee that our property is completely safe. The malicious modification of the data by the administrator can instantly turn our assets into nothing. Moreover, in today's global big data environment, people can now obtain information on all aspects of society through data sharing mechanisms. For medical experts and medical research institutes, the data and information of medical institutions are important research resources and assets. Therefore, the "information island" problem of the existing centralized medical system has gradually been receiving attention by experts in the medical field.

Reviewing the existing EHR management system, patient information is individually interacted and stored between a medical institution and its parent organization. The non-disclosure of the patient's EHR information causes the medical institution and the superior medical institution to have absolute control over the patient's EHR information. Just like for all other third-party organizations, patients choose to trust them. These authorities are generally credible, but we cannot rule out the possibility of errors. Moreover, EHRs often contain important medical privacy information, such as the patient's and doctor's name, date of birth, contact number, e-mail address, appointment information, medical record number etc. While ensuring that EHRs are not tampered with, medical institutions also need to ensure that the patient's legal EHRs information cannot be maliciously modified or leaked.

In the modernized technology, the improvement of the mobile portability of smart devices has brought about the problem of medical privacy leakage. As an example, a loss of phone, though extremely common, if timely and effective actions are not taken could result in the leakage of sensitive medical information. More severely, in 2003, an EHR computer with information of about 500,000 soldiers was reportedly stolen from the US Defense Department stores[56]. Similarly, in 2009, a Los Angeles hospital was punished for leaking patients' EHRs[56]. Security and privacy problems have always been a concern of all countries in the world. To this end, all countries in the world are seeking solutions in terms of technology, laws and systems. Studying a system that can effectively protect patient privacy and apply to electronic medical record management systems is one of the feasible solutions.

1.3.2. Blockchain based EHRs Management System

At present, Blockchain-based EHRs management systems generally integrate queries for EHRs. For example, SONY announced a partnership based block chain of remote digital medical items[57]. The system is divided into three modules: Issuer, Viewer, and Schema. The three code libraries which make up their remote digital medical system architecture are described below.

- Schema: A data specification that describes a number. A digital certificate is essentially a JSON file that contains some necessary fields, and the cert-issuer code places these fields on the Blockchain.
- Issuer: The issuer holds a JSON certificate. It creates a hash of this certificate (a string of characters that can be used as a unique identifier for a large data file) and embeds this hash value in the OP_RETURN field. This field includes the Bitcoin address of the publisher organization and the recipient Bitcoin address. When a broadcast of a Bitcoin transaction is sent

from the publisher to the recipient, a certificate is issued.

- Viewer: Used to display and verify a certificate that has already been published. The viewer also provides the user with the ability to request certificate verification and can generate a new Bitcoin ID (address).

A similar project is the block diploma-based digital diploma system project created by MIT. After a comprehensive analysis of the existing Blockchain-based EHRs management system, we found that they are all built on the system of Bitcoin.

Bitcoin uses the POW (Proof of Work) consensus mechanism. Although POW is highly secure, its consensus cycle is long and computationally expensive. At the same time, Bitcoin adopts the form of public chain, and the miners have strong mobility, but the stability is poor[58]. This means that a good miners group cannot be maintained without profit. These are all existing problems in the existing Blockchain-based EHRs management system.

Faced with these problems in the existing EHRs management system, this dissertation proposes a decentralized EHRs management system based on Blockchain. Using the decentralization and de-trusting features of Blockchain technology, we removed the central nodes in the traditional EHRs system. Establish a shared EHRs information sharing network that makes all EHRs release and recall operations open and fair. The EHRs management system adopts the alliance chain method and appropriately modifies the consensus mechanism to shorten the time period for reaching consensus. Using the non-tamperable nature of the Blockchain, the patient's EHRs information is written in the block to ensure that the patient's EHRs are not forged or maliciously modified.

1.4. Research Steps

Based on the decentralization of Blockchain, the focus of this dissertation is to implement a generalized P2P Blockchain EHRs system. The socket system is built using the socket.io component to implement point-to-point links between network nodes. The socket system is then embedded into the express framework of Node.js to build static servers and socket parallel servers, so as to separate the request of the static resource from the processing of the node communication. In-depth study of Blockchain verification and storage mechanisms is necessary in order to build a system that meets the security standards of confidence, tamper resistance, anti-intervention, and anti-hijacking. In order to achieve the above objectives, the research steps in this dissertation include:

1) The use of traditional EHRs in the medical field has the disadvantages of low efficiency, high cost and management difficulties. How a B/S-based system architecture can be designed to enable EHRs to quickly retrieve information in a cost efficient manner are the main considerations in the system construction process. To achieve this, related research works on Blockchain in medical application are compared. Afterwards an easy-to-use and easy-to-extract Blockchain data storage structure for medical records is designed.
2) By studying the development status of Blockchain technology, this dissertation analyzes the generation mechanism and structural characteristics of Blockchain in the other sectors. Combined with the content of this study and system functions, this dissertation improves the way in which blocks store information. Finally, data structures suitable for this study were designed.
3) The dissertation conducts theoretical research on Blockchain technology and medical applications of Blockchain technology, including research on existing Blockchain medical applications, system architecture analysis, and medical applications. An analysis of contradictions of the current Blockchain application in the medical field is done, in order to avoid artificially controllable development problems.
4) Combining computer science research methods, this dissertation

analyzes the requirements of the EHRs system of this study. It then designs the function module and system framework according to the requirements analysis. Research is done on the key technologies needed to implement the system, including cryptography principles of asymmetric encryption and hashing algorithm, which lay technical foundation for this research area.

5) Using computer programming principles for system design, code compilation and project implementation, the author considers the advantages and limitations of programming languages to overcome the difficulties of development. In the implementation process, we optimize system performance and simplify the operation process. The system is ultimately guaranteed to be safe, reliable and easy to use. Finally, in this dissertation, the developed system is tested; its usability analyzed using related performance metrics. The success of this study was demonstrated based on the test results.

1.5. Research Significance

At present, both public and private medical institutions have proprietary right to medical record and certifications. This often implies that patients do not have the right to edit the diagnosis and treatment process. In the same vein, though most online medical institutions have relevant records for the diagnosis and treatment process, the society usually does not recognize these data sorted on their centralized management systems[59]. This is because these institutions monopolize the management authority of the medical record data. Therefore, this situation hindered the researchers on the study of chronic diseases. Based on the characteristics of Blockchain, this dissertation designs a new EHRs management system and applies the consensus mechanism to solve the problem of EHRs issued by the current institutions not being recognized. The main significances of this study are as follow:

1) In the era of network globalization, traditional medical institutions lack

awareness of reform, and their technical capabilities and practical strategies are relatively backward in the certification and recording of medical behaviors for patients. The Blockchain technology, which is characterized by decentralization and trust, will effectively solve the above problems. Some researchers have proposed the specific application scope and mode of Blockchain technology in medical care. In Africa, people usually only seek and pay attention to Blockchain technology from the financial point of view. Although the Blockchain has already had some practical applications in the global medical field, the actual medical applications and technology development using the Blockchain have not received attention in Africa in general and more precisely in South Africa. This dissertation runs through the actual requirements, to the system framework design, model design, communication network design, code implementation and other computer science research methods, to achieve the EHRs management system based on Blockchain.

2) The contradiction between online diagnosis and centralized management mechanism has become increasingly prominent. There are many shortcomings in the practical use of the Blockchain decentralized platform. On one hand, this dissertation will focus on the development of Blockchain platform, combining theory with practice, to provide development cases and practical ideas for the application of Blockchain in the e-health field. While on the other hand, it focuses on the process of building the system. It explores new human-computer interaction mode and system architecture. The proposed model is implemented as a browser based software. It is a novel solution, which combines EHRs management, user friendly front-end technology, and medical software development.

2. Key Technologies

2.1. Blockchain Technology

Blockchain is a special data structure that combines data blocks in a logical order. It is a forged decentralized ledger that uses encryption algorithm to ensure that internal transaction information is not tampered with[60]. Since the Blockchain logically presents a sequential connection, it has a chain structure. First, the Blockchain discards the normal key-value access method to access the data, but it uses a chain structure and checks the block data by calculation. Second, it generates blocks and loads new blocks through consensus and voting mechanisms. Furthermore, the Blockchain uses encryption algorithms to improve the reliability of the message transmission process. The intelligent contract formed by the digitized code is a distributed control architecture that can manage data. A Blockchain can completely store virtual currency transaction records or other interactive data, while the relevant data in the database cannot be forged and altered.

Since the Blockchain can automate the execution of the contract code, there is no need for an authoritative centralization review institution to intervene. Transaction messages recorded in a Blockchain can not only be virtual currency such as Ethereum[61], but also multi-modal virtual assets such as stocks and cultural interests. Blockchain technology solves the problem of Byzantine failures, greatly reduces the trust cost and accounting cost of the real economy, and redefines the property rights system in the Internet era.

2.1.1. Blockchain

There are many definitions of Blockchain technology. In Wikipedia, Blockchain is defined as: “An intelligent peer-to-peer network that uses distributed databases to identify, propagate, and record information”[62]. It is based on Bitcoin, and is a series of data blocks generated by cryptography; with each data block containing several multiples of Bitcoin network transaction information, which are used to verify the validity of its information (anti-counterfeiting) and generate the next block. There are differences in the definition of Blockchain in countries around the world. For instance, in the UK[27], Blockchain is a database that uses blocks to store records. In Japan[39], Blockchain is defined as a digital asset trading technique. For some developer communities, Blockchain is a peer-to-peer distributed ledger technology.

As an emerging technology, Blockchain technology essentially uses a decentralized, fully distributed database approach to record data as compared to the traditional centralized accounting methods.

2.1.2. Blockchain Structure

In the case of Bitcoin, a client with computing privileges throughout the network is called a node. The Bitcoin node is always performing mathematical calculations, and the calculation method is to hash the previous block header information. Prior to this, the system will generate a random number to adjust the incoming information. The node needs to push back the random number through hash calculation. The node that successfully produces the result will receive Bitcoin as a calculation bonus and obtain the record permission of the current block. All nodes in the network will simultaneously acquire the Bitcoin transaction records during the current time period, using them as part of the block and storing them in the block.

Transactions cannot be stored as actual content during the storage process. In order to save storage space, all transaction information needs to be hashed by hash calculation. The stored structure is stored according to the Merkle tree[63], that is, all transaction information is used as a leaf node. In blockchain technology, Merkle tree is a tree in which every leaf node is labelled with the hash of a data block, and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes. In the process, hash of adjacent nodes is merged and the hash calculation of the same rule is performed again to obtain the merged hash. This merged hash is used as the parent of both nodes. By analogy, the node finally gets the value of the root node of the Merkle tree. After the complete block is generated, this value is saved on the Blockchain (stored in all local nodes that have permission to record the block). The chain structure of Blockchain is shown in the figure 2-1.

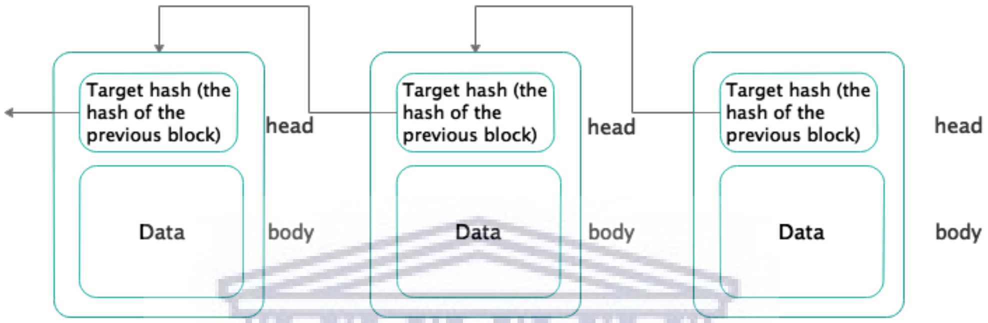

Target hash (the hash of the previous block)
head
Data
body
Target hash (the hash of the previous block)
head
Data
body
Target hash (the hash of the previous block)
head
Data
body

Figure 2-1: Structure of Blockchain

The structure of the Blockchain is divided into a block header and a block body. The data in the block is the transaction record stored in the Merkle tree. The construction of the Merkle tree is a process of calling recursively computed hash values, usually calculated as SHA256 or MD5[64]. Since the Merkle tree has the characteristics of a binary tree, it leads to its excellent scalability. Merkle tree can record a lot of information in several layers.

The block header includes the root value of the Merkle tree, the hash value of the previous block, the timestamp, the random number, and the version number[65]. The hash value is an important attribute in the block header. The hash value of the previous block is obtained by hashing to ensure that the block is stored in the chain. At the same time, this hash value also ensures that the block information cannot be falsified. Since the calculation process needs to combine the characteristics of the current block, if some node wants to modify the information in a certain block, the node needs to modify the information of all the blocks after the objective block. Such a process requires a great cost, and the hash algorithm thus ensures the security of the system.

Time stamps are another important component of the block structure. Since there is no central server in the Blockchain network, there may be a delay between system-time of each node. In order to avoid information inequality, the system needs to detect internal nodes. Node blocks with long time and full transaction storage are designated as priority blocks. Other nodes discard their own blocks and use the priority blocks instead. The timestamp stores the Blockchain in time series to facilitate block sorting and block query. Combined with the hash value, the system can achieve the separation of the generated block order and system time, and can achieve the purpose of distributed storage[65].

In this dissertation, the store information function of the Blockchain is used to generate, verify and store the patient's diagnosis and treatment information. In theory, Blockchain technology in the medical field does not need to translate computing power and energy consumption into capital, in contrast to the financial value of money. As the random number difficulty of the Bitcoin block under the PoW mechanism usually consumes ten minutes of commercial graphics card power. The computational cost of a computer or mobile phone needs to be significantly reduced when a medical institution conducts medical treatment. Thence, this dissertation will select the appropriate consensus mechanism for the EHRs management system.

2.1.3. Blockchain Features

Blockchain technology solves many practical problems in distributed storage:

Blockchain technology no longer relies on the authority of the central organization to make EHRs credible. It no longer trusts a single node, but performs mutual verification between nodes using a consensus mechanism[66]. Blockchain technology uses weighted accounting and node credibility for weighted verification. This ensures the consistency of the information, and also ensures the irreversible writing of the data in the calculation process.

Blockchain technology adopts peer-to-peer network (P2P) network in the process of information interaction to realize the transmission and recording of node data. Node accounting is random and redundant. Due to the dual role of incentives and consensus mechanisms, the system encourages nodes to perform mining calculations and participate in storage. This random event guarantees relative independence between system nodes. The number of participating nodes ensures that there is enough data in the network to form a complete Blockchain, which guarantees the integrity of the data. The interaction authority between nodes come from the degree of participation and management frequency of nodes in the system. If a node does not participate in the accounting, the node authority is insufficient to support the verification process of the participating Blockchain[66].

Block chain has the following characteristics:

1) Security: Distributed storage of Blockchain ensures decentralization of data interactions and avoids human intervention in data interaction. For an internal node to tamper with the data in the Blockchain, control of a certain number of nodes is required. This number varies according to different consensus mechanisms. At the data transmission level, for an attacker other than the nodes to hijack and tamper with the data using the P2P network, the attacker needs to crack the private key in the encrypted transmission, which is a very difficult process. At the information integrity level, in order to modify one block information, attacker needs to modify the information in all remaining blocks of all accounting nodes. The block information is obtained by strict encryption of the hash algorithm. Since the hash algorithm is irreversible, the security of the system is guaranteed.
2) Transparency: Blockchain have features such as ledgers that can be queried, transactions can be traced, sorted by time, and so on. The

completed transaction information will be fully recorded and can be accessed. Any node in the Blockchain can query the information in any block in the system, which greatly improves the transparency of the interaction[67].

3) Participation: Blockchain generation and storage are dependent on accounting nodes. If the number of nodes is too small, or the participation of node is not high, the integrity of the Blockchain cannot be ensured. When querying or calling a Blockchain, if the selected node stores less than expected data, the block is lost in the process of generating a complete Blockchain[67]. In addition, as the new block continues to be generated during the Blockchain storage process, the Blockchain needs to constantly store new data. This puts some pressure on each node involved in the accounting. Although, participation ensures the independence of node interaction, it also brings potential overhead and security risks to the system itself[66].

2.2. Technology to Support Blockchain

2.2.1. Peer-to-peer Network

The role of a P2P network in a Blockchain is to connect all nodes so that the Blockchain allows communication between any pair of nodes without relying on third parties[68]. The data information is transmitted in the form of a broadcast during the normal operation of the P2P network. The following are two key concepts of P2P network.

1) Broadcast mechanism: Nodes in a P2P network interact using a flat topology structure. In theory, there are no centralized special nodes and hierarchical structures in the network. Each node is responsible for network routing, verifying and disseminating information, discovering new nodes, and more. The method of publishing information by Blockchain and the transaction information generated in the system is broadcast to all nodes. During the broadcast process, node verifies

whether the message is legal, thus, determines whether to broadcast to the neighboring node. As long as the transaction information is received by more than 51% of the nodes, it can be assumed that the transaction is passed, and can be recorded in a new block. If the node confirms that the transaction information is incorrect, it will discard the information and terminate the broadcast operation[69].

In addition to verifying the transaction information, the P2P broadcast mechanism is also used to confirm the node's accounting right. A node that can derive a dynamic random number by calculation and has the most complete record, broadcasts new block data to the entire network. Other nodes discard their own blocks and receive the block data obtained by broadcasting. After verifying the accounting right, the system will save the new block into the Blockchain. The broadcast mechanism implemented in this dissertation will be realized through the network process broadcast mechanism, namely the socket management framework. The framework is able to propagate block information and medical records to nodes other than themselves.

2) Consensus mechanism: Consensus refers to the process by which network nodes adhere to a common rule and achieves consistent results for certain problems through asynchronous interaction[58]. The consensus mechanism in the Blockchain is mainly used to make the network nodes agree on block generation and benefit distribution. The biggest difference between different Blockchain networks comes from the difference in consensus mechanisms. Therefore, the characteristics and performance of different Blockchain networks are also different. There is no centralized management system with centralized permission in the Blockchain network, and all the nodes that participate in accounting are the key participants. How to make nodes interact to form a simple, easy to use, low-cost, group-managed storage system is a problem to be solved by Blockchain technology. Ensuring data

consistency and data availability in a data sharing network system are two factors to consider when building a system.

Currently, POW (Proof of Work)[70] is used in the Bitcoin network; Ethereum currently also uses PoW but plans to will gradually replace PoW with POS (Proof of Stake)[71]. Ripple uses RPCA (Ripple Protocol of Consensus Algorithm)[72], while Fabric uses PBFT (Practical Byzantine Fault Tolerance)[73]. There are two popular consensus mechanisms, namely DPoS (Delegated Proof of Stake)[74] and dBFT (Delegated Byzantine Fault Tolerance)[75]. Our CB-EHRs system uses the dBFT consensus mechanism. This consensus mechanism was proposed by NEO[76] which is an open source Blockchain project driven by the community. This dissertation will discuss dBFT in Chapter 3.

2.2.2. Encryption Algorithm

The Blockchain technology based on cryptography can realize communication between any two nodes, which solves the channel credit problem. In the traditional e-health field, a secure authentication mechanism is required to establish a secure communication between a medical institution and a user. Third-party certification bodies need to establish a secure channel to ensure that the security of communications and monitor the behavior of both parties, and to prevent illegal operation of patients. Such certification bodies are usually legal persons who issue EHRs and provide electronic signatures, and are responsible for keeping public keys and issuing electronic signatures. One of the characteristics of the certification body is to receive service remuneration and assume legal responsibility[77]. Therefore, this traditional three-party certification requires a large capital cost and bears legal risks. In contrast, the point-to-point data transmission method of Blockchain solves the problem of third-party cost. Its data transmission range is only between nodes, with trust and authentication between unfamiliar nodes implemented by an encryption algorithm.

The asymmetric encryption algorithms used in the Blockchain are elliptic curve encryption algorithm and RSA signature algorithm[78]. Asymmetric encryption is used to prevent key compromise and to facilitate identity authentication during communication. Especially in the P2P network, if symmetric encryption is used, a system with n nodes requires n(n-l)/2 keys, making the management and delivery of keys expensive. Asymmetric encryption solves the problem of key distribution in symmetric encryption systems. The RSA algorithm can be used to ensure that information is complete and non-repudiation. At present, RSA has been applied in various protocols including https. The security of RSA comes from the difficulty of factorization of large integers. The number of bits RSA key is usually 1024[79].

In the blockchain encryption process, when the user registers the Blockchain account, the system generates a key pair for the user. The private key is reserved by the user. As long as the private key is guaranteed not to be leaked, it is difficult to reverse reasoning through the information in the Blockchain. In the general sense, Blockchain transparency refers to the process by which a user can verify the transaction by exposing his or her own information, rather than reverse reasoning the user information by trading hash values[80]. Blockchain transparency is a feature that ensures the system is manageable, traceable, and controllable.

The design of the communication logic needs to simulate the user's operation from the user's point of view. The process designs system rules in the communication process based on interaction logic[81]. The design of communication logic in this dissertation belongs to the logic analysis and design of the underlying system, which is an important part of the Blockchain system.

After the public key cryptosystem was proposed, the mathematically based asymmetric cryptographic algorithm made a breakthrough. The biggest feature of this mechanism is the separation of encryption and decryption capabilities using two keys. The public key is used as the encryption key, and the private key is used as the decryption key. The public key cryptosystem enables secure communication between two parties without the need to exchange keys in advance[82]. It is impossible for an algorithm to analyze a private key from a public key or cipher text with limited computing resources and limited storage space. If the private key is used as the encryption key and the public key is used as the decryption key, it is possible to realize that the message encrypted by one user can be decrypted by a plurality of users, thereby realizing a digital signature instead of the handwritten signature.

The communication encryption mechanism of the CB-EHRs system uses the RSA encryption algorithm[79] to ensure the communication security of nodes in the Blockchain network. In the block generation process, since the length of the content delivered by the RSA cannot exceed the key length, the segment signature of the block needs to be performed by using the RSA signature algorithm. Firstly, the system stores the block in the JS0N file, then turns it into a string and splits it. The data is encrypted and sequentially issued along with the generated public key using the RSA encryption algorithm. Other nodes hash the data in the decrypted block body and compare it with the hash in the block header. If the two are the same, then the data is complete and has not been tampered with. This means that the block communication security check is completed, the record is valid, and the block is available. Finally, the node stores this block at the end of the Blockchain data, and the Blockchain recording process is completed. The above encryption process is designed as shown in Figure 2-2.

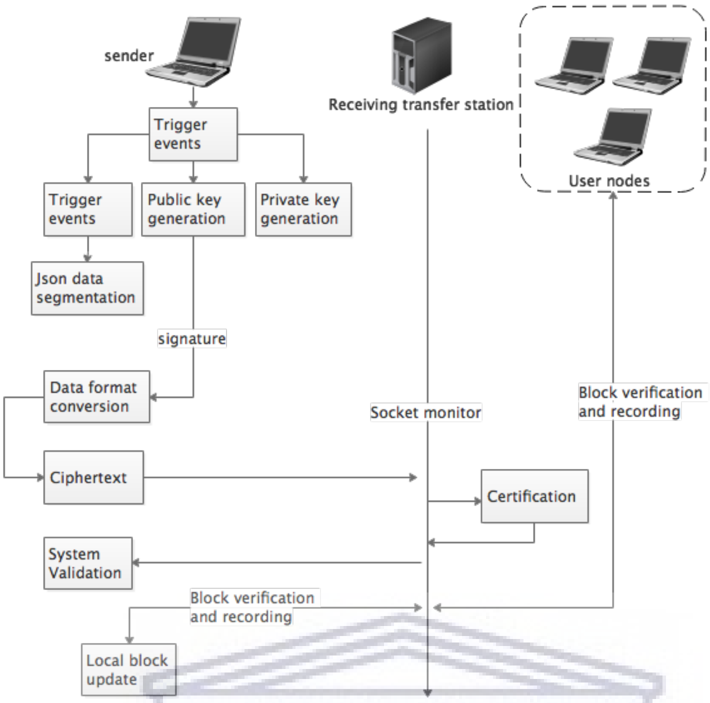


Figure 2-2: An Overview of the Blockchain Encryption Process

The above RSA signature process design has the following characteristics:

(1) The purpose of the signature is to verify the identity of the user node. Suppose the attacker uses the plaintext to make a request to the server, and the data of the invaded dirty node will still be verified by signature. However, since the plaintext cannot be decrypted and verified, the node ignores the request. The system thus enables users to trust each other.

(2) The RSA encryption process consumes a large amount of routing resources if it is completed by routing. When the number of parallel network nodes is large, this process will cause channel congestion, communication anomalies, and socket disconnection. If the system spreads the amount of computation to the user node, the network can be made smooth.

2.2.3. Hash Algorithm

Secure Hash Algorithm (SHA) is a commonly used data encryption algorithm. It was published by the National Institute of Standards and Technology (NIST) as a federal information processing standard in 1993 (the first generation SHA algorithm SHA-0)[83]. In 1995, its improved version of SHA-1 was also officially published. The SHA algorithm is currently the most commonly used secure hash algorithm and the most advanced encryption technology. The general idea of the hash algorithm is to receive a plaintext and then convert it into a short, fixed-numbered output sequence, which is a hash value (called a message digest), in an irreversible manner. The algorithm outputs a 160-bit message digest for the messages which length does not exceed 264 bits[84].

The hash algorithm operates on three 32-bit words using multiple consecutive logic functions $f_t (0 \leq t \leq 80)$ and produces a corresponding 32-bit output structure. In the hash algorithm, three 32-bit words are set to A, B, and C, and the related functions are defined as:

Equation 1:

$$f_t(A, B, C) = \begin{cases} (A \wedge B) \oplus (\sim A \wedge C), & 0 \leq t \leq 19 \\ A \oplus B \oplus C, & 20 \leq t \leq 39 \\ (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C), & 40 \leq t \leq 59 \\ A \oplus B \oplus C, & 60 \leq t \leq 79 \end{cases}$$

In terms of constant selection, the hash algorithm requires a total of 80 32-bit constants K_t. These constants are given in hexadecimal form. The specific parameters are as follows:

Equation 2:

$$K_t = \begin{cases} 0 \times 5A827999, & 0 \leq t \leq 19 \\ 0 \times 6ED9EBA1, & 20 \leq t \leq 39 \\ 0 \times 8F1BBCDC, & 40 \leq t \leq 59 \\ 0 \times CA62C1D6, & 60 \leq t \leq 79 \end{cases}$$

Before the hash calculation of the character, it is necessary to do the split filling of the message and the setting of the initial value of the hash. First, we set the original message to M and the length to L. Then we add "1" to the end of the binary code of the message and add k "0"s. $L + 1 + k$ should satisfy the modulo 448 after modulo, and it gets a 512-bit padding message after adding 64 bits of M binary data. Next, we use the patched message to calculate the message digest. The calculation requires two buffers, each consisting of five 32-bit, and also requires another buffer of 80 32-bit. The first buffer is identified as A, B, C, D, and E. The second buffer is identified as H_0, H_1, H_2, H_3,and H_4. In addition, the 80 words buffer is identified as W_0, $W_{1, \ldots,}W_{79}$. Finally, we also need to prepare a TEMP buffer for each word. The filled message is split into M_1, $M_{2,}$ . . . , M_n which are processed sequentially. The process of processing each data block M_i consists of 80 steps. The buffer $\{H_i\}$ is initialized to the following value (hexadecimal) before all blocks are processed.

$$H_0 = 0\times67452301$$

$$H_1 = 0\times\mathrm{EFCDAB89}$$

$$\mathrm{H}_2 = 0\times98\mathrm{BADCFE}$$

$$H_3 = 0\times10325476$$

$$\mathrm{H}_4 = 0\times\mathrm{C3D2E1F0}$$

Next processing M_l, M_2, . . . , M_n. In order to process M_i, the steps required to perform the hash calculation are as follows.

1. Divide M_i into 16 byte W_0, W_1, . . . , W_{15}, W_0 is the leftmost byte.
2. If t=16~79, Set $\mathrm{W[t]} = \mathrm{S1(W[t-3]} \oplus W[t-8] \oplus W[t-14] \oplus W[t-16])$
3. Set A=H_0, B=H_1, C=H_2, D=H_3, E=H_4
4. For t=0 to 79, execute the following loop:

TEMP=S5 (A) +Ft (B, C, D) +E + Wt + Kt ;

E=D ; D=C ; C=S30 (B) ; B=A ; A=TEMP.

5. Set $H_0=H_0+A$, $H_1=H_1+B$, $H_2=H_2+C$, $H_3=H_3+D$, $H_4=H_4+E$. After M_n has been processed, the message digest is a 160-bit string tagged with the sequence identifiers H0, H1, H2, H3, H4.

2.3. Classification of Blockchain

The Blockchain is divided into the following three categories according to different network node selection methods:

a. Public Blockchain: is a Blockchain that has a consensus process that anyone in the world can access, send, and participate in. Its consensus process determines which blocks can be added to the Blockchain and clarifies the state of the current block[85]. As an alternative to centralized or quasi-centralized trust, the security of the public Blockchain is maintained by the "encrypted digital economy." The “encrypted digital economy” combines economic rewards with encrypted digital verification by means of a PoW or a PoS mechanism, and follows the general principles; The economic rewards that each node can derive from it are directly proportional to its contribution to the consensus process[85]. These Blockchain are often considered to be "completely decentralized".

b. Community Blockchain (also known as Alliance chain): refers to the process by which consensus pre-selected node control block chain[85]. For example, a community of multiple financial institutions. Each of its organizations runs a node. In order for each block to be effective, the financial community needs to obtain confirmation from more than half of the institutions. The Blockchain is either readable by everyone, or restricted to participants, or a hybrid route. This can be implemented by exposing the root hash of the block in the system and API (application programming interface). At the same time, the system's API allows users outside the organization to make a limited number of queries and get information about the status of the Blockchain[85]. This Blockchain can be thought of as "partial decentralization."

c. Fully private Blockchain: is a Blockchain whose write access is only in the hands of an organization. Its read access is either open to the outside or limited to a certain extent[86]. In many cases, this type of Blockchain is not necessarily readable for public users.

2.4. RBAC Model and REST Framework

This dissertation uses a REST (Representation State Transfer)[87]-based RBAC (Role-Based Access Control) model[88].

In computer systems security, RBAC or role-based security is an approach that restricts system access to authorized users. RBAC is a policy-neutral access-control mechanism defined around roles and privileges[88]. The components of RBAC such as role-permissions, user-role and role-role relationships make it simple to perform user assignments.

The most basic RBAC model includes three main categories; entity user, role and permission set. Their relationship is shown in Figure 2-3.

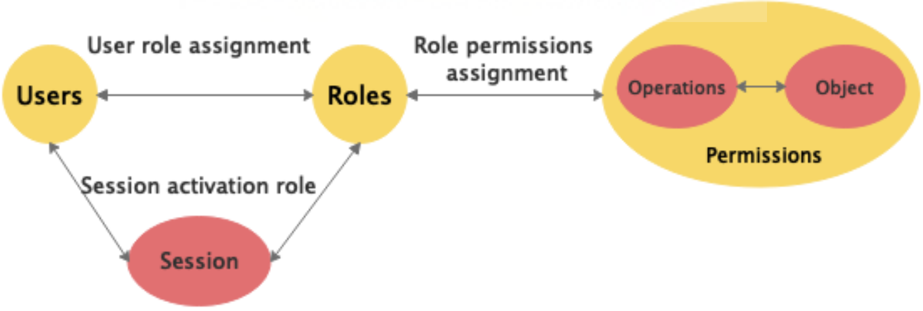


Figure 2-3: The RBAC model

Users and roles are many-to-many relationships, meaning that a user can have different roles in different scenarios. Roles and permissions are many-to-many relationships, indicating that a role can have multiple permissions. In this structure, the same right can be granted to multiple roles. What must be mentioned is that the separation of users and permissions make the authorization of the authority more flexible.

REST (Representation State Transfer) describes the architectural style of a network system. It first appeared in the PhD thesis of Roy Fielding in 2000[87]. Roy Fielding is one of the main authors of the HTTP specification.

REST is a set of constraints and principles for system architecture implementation. A system that satisfies REST constraints and principles can be called RESTful. In a RESTful system, the process of a client initiating a request is shown in Figure 2-4.

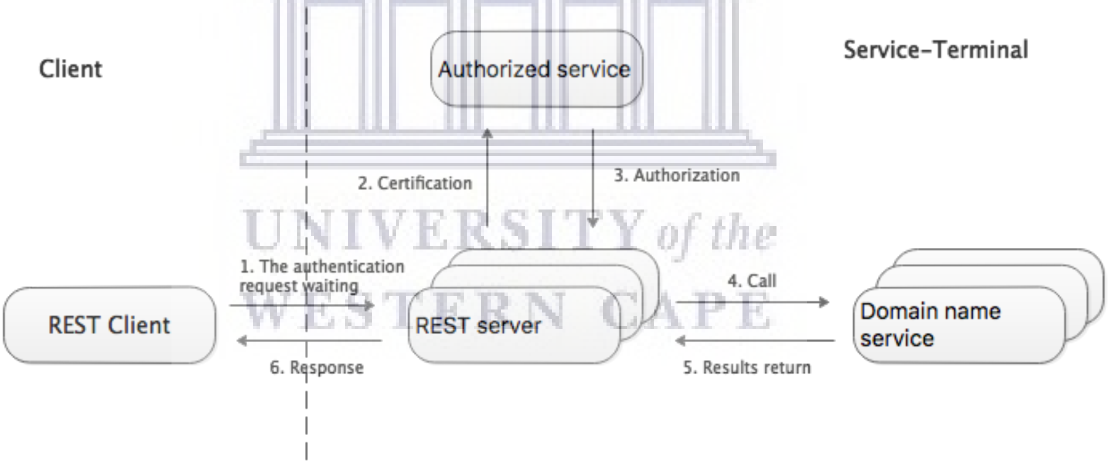


Figure 2-4: REST System

The most important principle of REST is that the interaction between the client and the server is stateless. Each time a client initiates a request, the content must include all the information needed for this request. This stateless request from the client can be answered by any server.

Another important principle of REST is the layering of the system. The functional components of the system can only access components that interact with it, and cannot access other components. This approach makes the various levels independent of each other and simplifies the implementation of the client and server.

2.5. Consensus Mechanism

2.5.1. The Byzantine Generals Problem

In 1982, scientists such as Leslie Lamport proposed the famous Byzantine failures problem, which discusses the issue of achieving consensus in a scenario that allows a few malicious nodes. A brief description of the problem of Byzantium generals: The vast Byzantine empire has its own troops deployed throughout the country for defensive purposes. Because of the distance, the generals had to rely on messengers to pass on messages. When the war broke out, the generals of the Byzantine Empire must decide unanimously whether to attack an enemy army in order to win the battle. However, the generals could not determine whether there were traitors among them. A traitor may arbitrarily change the attack intention or attack time to destroy the consistency of the attack and cause the operation to fail. In this state, what kind of remote communication methods should the Byzantine generals agree on to achieve consensus?

The paper points out that in the Byzantine problem, if the total number of nodes is N and the number of rebel generals is F, then the problem will be solved when $N \geq 3F + 1$. The entire process is guaranteed by BFT (Byzantine Fault Tolerance Algorithm). Here is a brief description of the demonstration process. We assume that the total number of nodes is N, the total number of malicious nodes is F, the number of valid good nodes is L1, and the number of invalid (failure) good nodes is L2. In order to reach a safe agreement, the system must meet two points: the number of valid good nodes exceeds the number of bad nodes, and it must also exceed the number of good nodes that fail. The above description is transformed into a mathematical formula as follows: $L1 \geq F + 1$, $L1 \geq L2 + 1$ ($L2 = F$), and $L1 = N-L2-F$, that is, $N-F-F \geq F + 1$, and $N \geq 3F + 1$. Therefore, when the mutineers do not exceed 1/3, there is an effective BFT algorithm. However, BFT always has the problem of too high complexity, and it has not really landed in the actual scene.

Until 1999, Miguel Castro and Barbara Liskov proposed PBFT (Practical Byzantine Fault Tolerance) to solve the problem of the inefficiency of the original BFT algorithm. PBFT greatly reduces the complexity of the algorithm, making BFT feasible in practical applications. In 2008, the design of Bitcoin's blockchain network creatively improved the problem of multiple proposals and eventual consistency confirmation in PBFT. In the end, the Bitcoin blockchain proposed and adopted the PoW (Proof of Work) algorithm. Although the PoW mechanism is very robust, it is costly. Some blockchain users have criticized Bitcoin for wasting a lot of computing power, electricity, and computing equipment. Therefore, scholars around the world have developed many consensus mechanisms to replace PoW.

2.5.2. Common Consensus Mechanisms

At present, there are many common consensus mechanisms, such as PoW (Proof of Work), PoS (Proof of Stake), DPoS (Delegate Proof of Stake), RPCA (Ripple Consensus Algorithm), PBFT (Practical Byzantine Fault Tolerance) and dBFT (delegated Byzantine Fault Tolerance). A brief introduction to the above consensus algorithms is given in the succeeding paragraphs.

The PoW mechanism is based on the sacrifice of computing power to consensus. In order to prevent sybil attack[89], this mechanism requires nodes in the network to prove their true identity through a certain amount of work. The job in PoW is to solve a cryptographic problem. In order to find a random number less than the difficulty target value, the network node continuously calculates the block header hash value of the new block by changing the random number in the block header. This process is called mining. The smaller the target difficulty value in the calculation process, the more difficult the mining is, and the higher the requirement for the node calculation. The advantage of PoW is complete decentralization and nodes are free to enter and exit the network. Its disadvantage is that a large part of the computing power in the world has been absorbed into the Bitcoin system. Other Blockchain-based applications that use the PoW consensus will be more difficult if they want to get the same amount of computing power to keep themselves safe. The process of mining will also result in waste of resources. This consensus is not suitable for commercial applications due to the long consensus cycle.

PoS uses proof of equity instead of proof of workload as in the case of PoW. The node with the highest equity performs new block generation and acquisition rewards in a Blockchain network using PoS consensus. The stakes of these nodes represent the ownership of a node to a certain number of digital currencies. Stake's measure is based on the age of the coin, which is the product of the number of currencies owned by the node and the time the node holds for those currencies[71]. The digital currency holding time refers to the length of time since the last transaction. The difficulty of competition in the PoS mechanism is inversely proportional to the age of the currency spent in the transaction. That is, the more the number of coins that the node is willing to consume, the easier it becomes to become a billing node to obtain block billing rights and rewards. PoS makes any malicious node in the network have to cost more than the main chain. Since no calculation of power is required, PoS can shorten the period of consensus reached within a certain range. However, it still needs mining, and it does not essentially solve the weakness of commercial applications.

DPoS is an evolution of PoS. The network node of DPoS has the corresponding voting power according to the stakes it owns. These nodes vote to select a certain number of trustees to maintain the Blockchain system. The trustee is responsible for generating blocks in turn in the prescribed order and equally dividing 10% of the total transaction costs obtained as incentives. The trust nodes of the Blockchain system in DPoS are changed from all nodes to all trustees. When a node initiates a transaction, the system only need to wait for the trustee to confirm. Therefore, DPoS can greatly shorten the transaction time of the Blockchain system. All trustees in DPoS are required to pay a certain margin before they can compete for trusteeship to ensure they are not malicious. This consensus mechanism ensures the legal identity of all trustees. DPoS can reduce the number of nodes participating in the verification, thus reducing the consensus time. However, the entire Blockchain system still needs to rely on token operation. so it is unlikely that it will be used commercially.

RPCA is the consensus algorithm used by the Ripple system. The Ripple currency system have a P2P clearing network designed for currency clearing[72]. A UNL (Unique Node List) is introduced in the network to verify the nodes and vote for consensus. In the RPCA consensus process, each verification node collects the resolution results of other verification nodes on the transaction by querying the local UNL. The verification node compares the local set of resolutions with the resolution results from other nodes. If the comparison is consistent, the transaction set gets a vote. A plurality of rounds of voting are used to select the final set of eligible transactions and generate new blocks. The RPCA mechanism operated by the Ripple system does not require mining. It can achieve second-level consensus verification. However, its network maintenance costs are high and it is more vulnerable to attacks because nodes can be manually maintained.

PBFT is a message-based consistency algorithm[73]. The Fabric system uses the PBFT consensus. In the PBFT consensus process, the network node first selects the primary node of the current round of consensus. Then, the master node performs pre-prepare, prepare, and commit steps with other backup nodes to complete the consensus on the transaction. Finally, the master node generates and publishes blocks. In addition, a member management mechanism is provided in the fabric system to which PBFT is applied. This mechanism can manage the node's access to the network and handle transactions, thus making up for the low fault tolerance of Byzantine fault-tolerant protocols compared with other Blockchain networks. There are no economic incentives needed in PBFT.

The security of the system is guaranteed by the parties or regulators of the business. However, PBFT's communication process is too long. During the commit phase, the system designed complex judgment statements and communication flows to ensure that all nodes reached the prepared state. Therefore, the PBFT consensus algorithm is difficult to support large-scale network nodes.

dBFT comes from the NEO Blockchain[90]. The NEO Blockchain is a distributed intelligent contract platform. The dBFT proposed by NEO is an improved Byzantine fault-tolerant consensus algorithm based on PBFT. At the same time dBFT consults the characteristics of PoS (NEO holders need to vote for consensus nodes) to use the least resources to protect the network from Byzantine failures. However, dBFT solved some problems with PoS. dBFT provides fault tolerance for a consensus system consisting of n consensus nodes. This fault tolerance includes both security and availability, and applies to any network environment. The nodes in the NEO Blockchain network are divided into two types of nodes. One type is the consensus point, responsible for consensus communication with other consensus points to generate new blocks. The other type is a normal node that does not participate in the consensus but is able to verify and accept new blocks. The consensus node is generated by voting carried out by all nodes in the entire network. The main idea of dBFT proposed by NEO is that PBFT algorithm can solve the consensus problem of distributed nodes very well; however, the greater the number of nodes participating in the PBFT consensus, the lower the performance. NEO thus uses voting to select a relatively small number of consensus nodes. It then performs a PBFT consensus between these nodes to generate a new block. Finally, the new block will be released to the whole network to reach the consensus of the whole network.

The comparison of these main consensus mechanism characteristics is shown in Table 2-1[91-93]. It should be noted that the Blockchain technology is in the stage of rapid development. The following data represent only the real-time data of the dissertation in the research and analysis stage. The comparative results shown in the tablecome from a similar network and hardware environment. They will be used for basic analysis and does not represent the final performance of each consensus mechanism.

Table 2-1: Comparison of Consensus Mechanisms

Consensus mechanism	PoW	PoS	DPoS	RPCA	PBFT	dBFT
Scenes	Public chain	Public chain Alliance chain	Public chain Alliance chain	Public chain	Alliance chain	Alliance chain
Accounting nodes	All nodes	All nodes	Select representative nodes	All nodes	Static selection	Dynamic selection
Response time	About 10 minutes	1 minute	<1 minute	<1 minute	<1 minute	<1 minute
Ideal state of Transaction Per Second (TPS)	7 TPS	300 TPS	500 TPS	>10 thousand TPS	1000 TPS	1000 TPS
Fault tolerance	50%	50%	50%	20%	33%	33%

3. System Analysis and Overall Architecture

3.1. Overall Project Planning

The main function of the CB-EHRs system based on Blockchain technology studied in this dissertation is to secure the upload and storage of medical data. The system will store the EHRs from the medical institution into pre-processing record tables, the data stored is the variable users' EHRs. The EHRs data when authorized by the patient will be transmitted to the Blockchain database. The data used for verification at the same time will be stored in a log file. The Blockchain database and the log file are separated from each other and associated with each other to implement a medical data sharing platform.

In the decentralized data sharing system discussed in this dissertation, all nodes are equal, and they all can participate in the authentication and storage of user identity and user uploaded data. The status of all nodes in the CB-EHRs system is kept as consistent as possible. This enables contents uploaded by the user be sent to all nodes in the system and identified in a timely manner. Then the entire network will be notified to update the Blockchain. Every time a user uploads a health record, the system will verify the validity of its signature, afterwhich the verified data will be added to the Blockchain. During the operation of the system, all participating P2P network nodes will periodically detect the Blockchain data in order to prevent the EHRs from being tampered with. The tamper-proof features of EHRs files are the most important functions in this system.

In addition to the user login and data upload functions, the system provides users with other auxiliary functions to operate in the interface. The mail function has been added to facilitate communication between medical personnel and patients. At the same time, the conditional query function is also added, which is convenient for users to access the data based on different situations. More important significance is that this function can facilitate the work of medical researchers. For example, tracking and studying a chronic disease.

3.2. Requirements Analysis

The purpose of requirements analysis is to understand the relevant requirements and objectives of system development[94]. The requirements analysis plays an important role in the subsequent design and implementation of the CB-EHRs system. This dissertation focuses on the application of Blockchain technology in the field of EHRs. The design goal is to implement a safe, stable, tamper-proof and traceable EHRs management system based on Blockchain technology. This dissertation analyzes the system's functional requirements, feasibility, security and stability. According to the analysis results, we designed the overall architecture and develop specific functional modules for the Blockchain-based CB-EHRs management system.

3.2.1. Functional Requirements Analysis

After analyzing the overall planning of the project, we can divide the system modules into three functional modules according to its function. They are the user registration login function module, the user data authorization function module, and the broadcast upload function module. The user uses the registration login module to join the system and become an ordinary patient with private key. The patient user then uses the data authorization module to add descriptive information to the EHRs. The broadcast upload function module will verify their digital signature information after receiving the EHRs. The verified data will be updated to the Blockchain by the entire network node. The overall analysis structure of the system is shown in Figure 3-1.

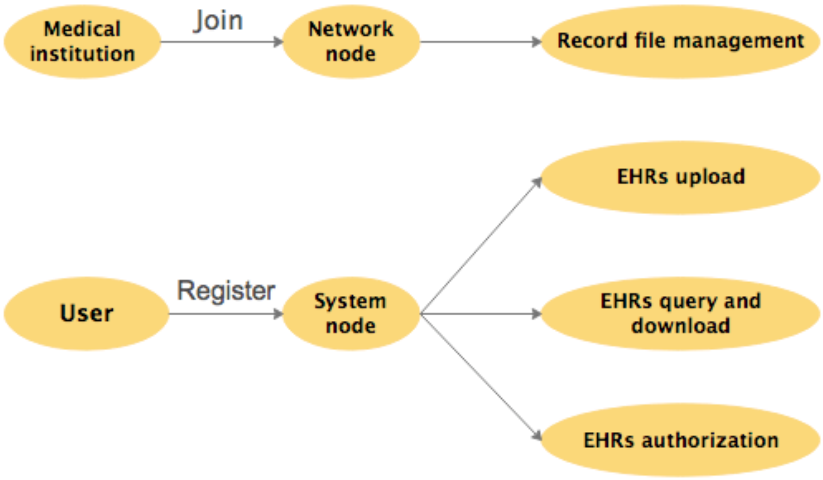


Figure 3-1: System Structure Overview

User registration and login functions mainly implement user registration and login operations. Users who want to join the system must be first be registered. Upon successful registration, the user's data is stored and the patient role is enabled for the user within the EHRs system. Subsequent times, the user simply needs to log in to access the system. The process structure diagram of this function module is shown in Figure 3-2.

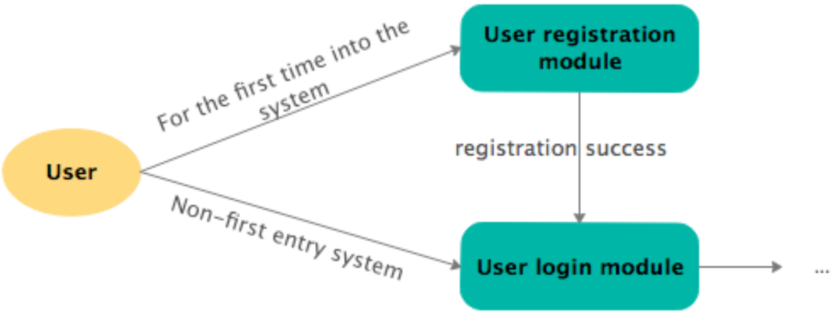


Figure 3-2: User Registration and Log in

1) Requirements analysis of user registration and login module

- User registration function: The user fills personal information such as email, username and password in accordance with the registration requirements set by the system. Only user data that meets all the set requirements can be accepted by the system. The basic information of the registered user will be stored for verification when it is logged in to the module. The system generates a unique private key and corresponding public key for the user while the user is registered.
- User login function: After the user successfully registers, the email and password are saved on the account table. Users need to accurately fill in their email and password when logging into the system.. The user can successfully implement the login operation only by entering the email and password that match the data in the system. When the user logs in, the system recognizes the user's role permission and displays the corresponding interface to the current user through the RBAC mechanism.

2) Requirements analysis of data authorization module

This module is mainly responsible for authorizing and conditional query. Based on the expected results of the CB-EHRs system proposed in this dissertation, the process of user authorization must conform to the characteristics of decentralization, trust and security in order to make the design of this system a feasible technical solution. The sub-modules of this function module are mainly divided into a data upload function module and data conditional query module. Its corresponding functional structure diagram is shown in Figure 3-3.

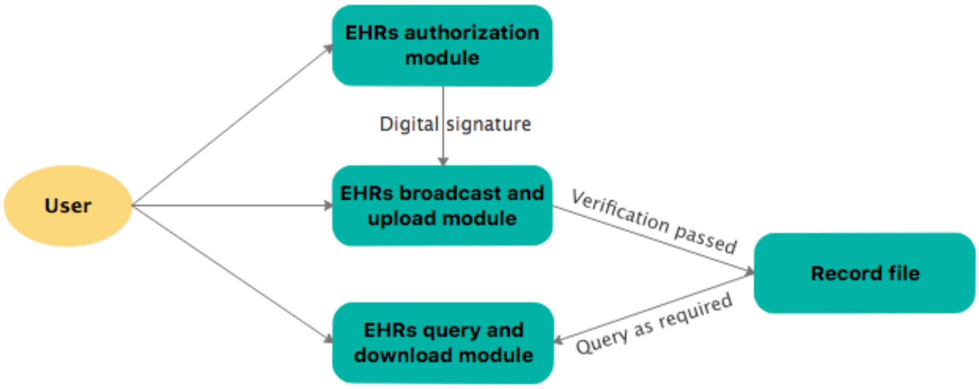

EHRs authorization module
Digital signature
User
EHRs broadcast and upload module
Verification passed
Record file
EHRs query and download module
Query as required

Figure 3-3: Data Authorization Analysis

- Data Authorization module: Firstly, the doctor uploads the patient's EHRs to the pre-processing table. The patient then selects the record that he/she wants to authorize for upload and clicks "Confirm Authorization" to add a description to the record. Next, the system produces an MD5 value corresponding to the selected EHR, and combines the time stamp, user information, MD5 value and description information as the original input information. Finally, the system digitally signs the input information using the private key and sends the signature along with the original input through the broadcast upload module. The transmission message must include the original input information, the digital signature, and the public key corresponding to the signature private key, so that the CB-EHRs system can perform the verification operation.
- Data download module: This module avails users the ability to search for information of interest to them. The method is to display relevant content by searching for keywords. For example, after entering the main interface, the doctor can find a record of a specific patient or a specific disease through the query interface. This method of data sharing facilitates the study of specific cases by medical personnel. Another use case of this module, is that pharmacist can confirm the

patient's allergen through the patient's history.

3) Requirements analysis of broadcast upload module

The function module is mainly responsible for verifying the EHRs authorized by the user and broadcasting the qualified EHRs to the network nodes of the Blockchain. This feature involves a consensus mechanism. The functional structure of the broadcast upload module is shown in Figure 3-4.

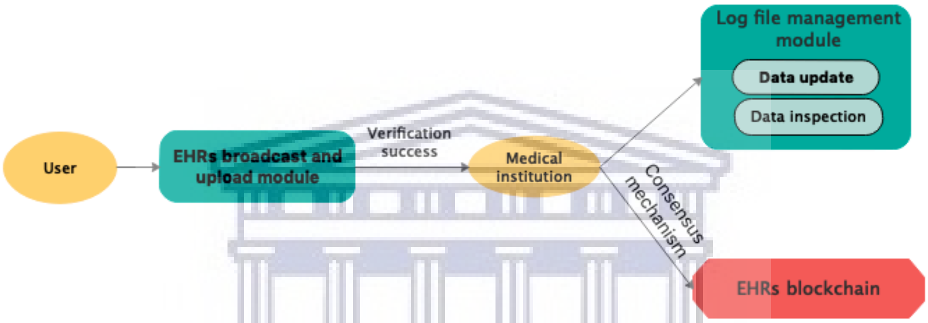


Figure 3-4: Broadcast Module

When a node in the network receives a message transmitted by the broadcast upload module, the system verifies the signature of the message by using the local corresponding public key. The verification signature is to confirm that the message does not have a message intercepted or modified during transmission. The process of secure transmission reflects the characteristics of system security. When the signature verification is successful, the system will notify the consensus mechanism for subsequent processing. The message is then updated to the Blockchain database. If the verification fails, it will not be processed later.

3.2.2. Non-functional Requirements Analysis

Non-functional requirements are an overall requirement of the CB-EHRs system.

It has a major impact on the long-term operation and maintenance of the system[94]. The non-functional requirements analysis of the Blockchain-based EHRs management system designed in this dissertation are as follows:

(1) System efficiency

The storage of data in the Blockchain must wait for the relevant transaction record to be encapsulated into a new block before it can be completed. This is an inefficient process. There is therefore the need to find a way to choose the appropriate consensus mechanism for the system. This is an important issue that needs to be solved in this dissertation.

(2) System security

The CB-EHRs system stores a large amount of EHRs information from medical institutions. The disclosure and tampering of data will bring huge losses to individuals and institutions. The purpose of this dissertation is to design a EHRs management system to protect EHRs data. Therefore, how to protect EHRs information by combining data encryption, hashing and Blockchain technology is the subject to be tackled in this dissertation.

(3) System scalability

After the development of the CB-EHRs system is completed, there is usually work to maintain the system and add new features. Therefore, in the system development process, the maintainability and scalability of the system should be fully considered to facilitate the follow-up work of the developer.

3.3. Feasibility Analysis

3.3.1. Technical Feasibility

The project being developed is called the Credible Blockchain-based E-Health Records System (CB-EHRs). It runs on a MySQL database[95] and was developed using Eclipse IDE[96].

The first practical application of Blockchain technology is in Bitcoin [18]. Since the birth of Bitcoin, Blockchain technology has been able to support the stable development of Bitcoin in such a large global computing environment, which has continuously attracted people's attention. Therefore, from the research and development of the Bitcoin system, it is feasible from a technical point of view to use the underlying key technologies of the Blockchain to integrate into the data sharing component. In particular, cryptography, hashing algorithms, digital signature algorithms, and broadcast communication modes are mature and stable technologies. Using them for data processing when sharing is a very viable solution.

The relevant data store in development uses the MySQL database, which not only can store and process large amounts of data, but also maintain the stability of the data, and it is easy for ordinary developers to learn and use the MySQL database. From a technical point of view, it is feasible to use the MySQL data source for storing user data and uploading EHRs file information in the system.

3.3.2. Economic feasibility

Economic feasibility refers to whether the system development and maintenance personnel have the ability to bear the system development costs and maintain the stable development of the system[97]. Through the preliminary investigation of data sharing and asymmetric encryption technology, it can be concluded that the development cost of the application combining asymmetric encryption technology and data sharing is acceptable for system development and maintenance personnel.

Why don't we implement our project directly using an existing blockchain development platform? For example, NEO and Ethereum. As far as the current situation is concerned, the CB-EHRs system project does not require equipment with superior computing power, nor does it require a very wide range of nodes to participate. As long as the medical institution nodes in the system work together to maintain the system, the system can develop steadily. If we use the existing blockchain platform directly, it will greatly increase the development cost. The CB-EHRs system designed and implemented in this dissertation refers to the NEO blockchain using the dBFT consensus mechanism. So here, we will explore the costs that may be incurred after the EHRs system is built on the NEO platform.

1. What is GAS?

   NEOGAS / GAS is the fuel cost required to use smart contracts. It represents the right to use the NEO blockchain. GAS is generated with each new block generation. It follows the established slowly decaying release rate and experiences a total process from 0 to 100 million. The total amount reached 100 million in about 22 years.

2. The charging rules of the NEO platform.

   Users need to pay a certain fee when using the NEO network. The total fee includes the system fee and network fee, and the unit of charge is GAS. Fees distribution rule is shown in Figure 3-5.

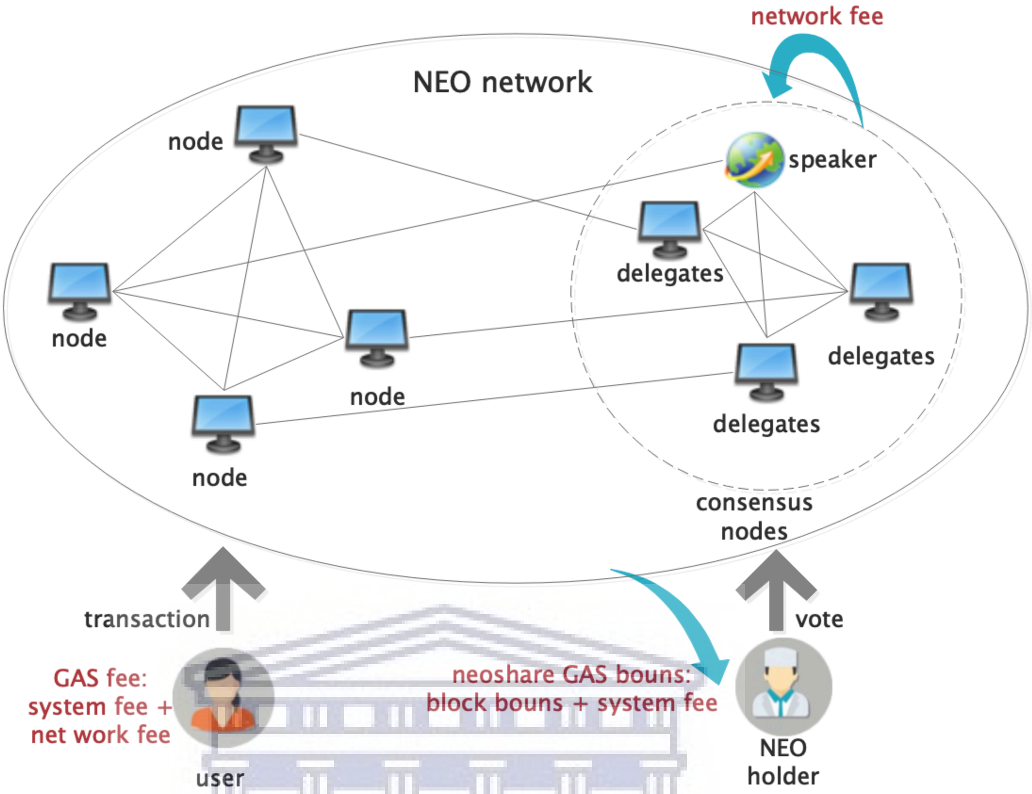


Figure 3-5: NEO Charging Model

- System fee: When a developer deploys a smart contract to the NEO blockchain, the developer needs to pay a system fee of 500 GAS to the NEO system. Every time a smart contract instruction is executed, a certain execution fee must be paid to the NEO system. Each smart contract has a 10 GAS free quota during each execution, whether it is a developer deployment or a user call. Therefore, if the single execution cost of a smart contract is less than 10 GAS, there is no need to pay a handling fee. If it exceeds 10 GAS, the 10 GAS fee will be waived.
- Network fees: Network fees are fees that users pay when submitting transactions to the NEO network. Users can set their own network fees. Theoretically, the higher the network fee per unit byte, the easier it is to be packaged and confirmed.

If we use the NEO blockchain platform to build the EHRs system, when a medical institution joins the blockchain network and becomes a consensus node, corresponding costs will be incurred. The fee structure for this operation is shown in Table 3-1. (i. All charging standards in the table below are from the NEO white paper. ii. As of December 13, 2019, the exchange rate of GAS to Rand is: 1 GAS = 14.9436 ZAR)

Table 3-1: Handling Fee Table of Setting One Consensus Node

SysCall	Fees（GAS）	Cost in ZAR
Validator.Register	1000	14943.62
Account.SetVotes	1	14.94
Runtime.CheckWitness	0.2	2.99
Blockchain.GetHeader	0.1	1.49
Blockchain.GetValidators	0.2	2.99
Blockchain.GetContract	0.1	1.49
Total	991.6（1001.6-10）	14818.10

Summary: When we set up the EHRs system on the NEO platform, we first need to pay 500 GAS (about 7471.81 ZAR) to the NEO community. Assume that 100 medical institutions nationwide join the system and become consensus nodes. Then the initial establishment cost of the entire system will be as high as 1.49 million rand. But the CB-EHRs system in this dissertation is a completely independent and own EHRs system. We only need to pay some network hardware rental or purchase costs in the development of the project. This means that the cost of the developer's investment is very small, so it is economically feasible.

3.4. System Security Analysis

- Algorithm security

The security of the CB-EHRs system algorithm mainly comes from MD5 hash algorithm and MD5withRSA digital signature algorithm, as these two algorithms play vital roles in the data processing and message passing process of CB-EHRs system. So far, the MD5 hash algorithm and the MD5 with RSA digital signature algorithm have not been easily cracked. Decrypting it by means of brute force will incur a large cost. Therefore, from the perspective of the algorithm security of this system, it is very safe and feasible to apply cryptography technology to data sharing.

- Data tamper resistance

Nodes added to the system will inevitably have malicious nodes. For this system, the most important purpose of a malicious node to attack the system is to modify or delete the data in database. In theory, malicious node must have more than at least 33% of the computing power of the entire network of the system[98]. Though it is possible for a malicious node to make modified records acceptable to all nodes of the system; being able to control such a large computing power is a major hurdle. For this to happen, the malicious node must in itself be a very very powerful computer. The cost of such a computer system is too prohibitively expensive for the sole purpose of making data stored in the CB-EHRs system open to the public. Therefore, such malicious attacks cannot bring enough economic benefits to justify being done. In view of this consideration, the system is still very stable and secure.

3.5. Summary of Chapter 3

According to the overall project plan, this chapter elaborates on the requirements analysis of each sub-module of the system. When introducing the overall planning of the project, this chapter provided a more comprehensive analysis of the overall requirements of the CB-EHRs system, and then divided the overall requirements analysis according to different functions. Finally, this chapter analyzes the feasibility and security considerations of the system.

4. Design of the Credible Blockchain-based E-health Records System

The Blockchain serves as the underlying technology to support the smooth and reliable operation of the Bitcoin system. It timestamps Bitcoin transactions to ensure the orderliness of Bitcoin transactions[99]. The Blockchain consensus mechanism prevents the emergence of Bitcoin "double flower" problems. At the same time, the consensus mechanism verifies the validity of the transaction records and ensures that the transaction records are safe and cannot be tampered with. This dissertation draws on the Bitcoin and attempts to set up various hospitals or other medical institutions as nodes on the Blockchain. Such a system can convert traditional operations of EHRs into Bitcoin-like transactions and store transaction records on the constructed Blockchain-based system.

Firstly, according to the characteristics of the Blockchain, all "transactions" in the network (all operations on EHRs) will be recorded and broadcast to other nodes. The transparency of EHRs release operations effectively regulates the behavior of EHRs issuing agencies.

Furthermore, due to the structure of the Blockchain itself and the existence of consensus mechanism, the data on the Blockchain cannot be tampered with. This feature ensures security of information in EHRs.

Finally, in the Blockchain, the transaction data is ultimately stored in the form of a block file. These block files have a fixed format and their storage path format is also uniform. On the one hand, this dissertation uses the REST framework to build the CB-EHRs system to enable data to be shared more efficiently and to provide various possibilities for future expansion. On the other hand, this dissertation establishes the pre-processing of EHRs according to the actual situation, to ensure that invalid EHRs can be abolished according to a reasonable process before being authorized by the user.

A complete Blockchain network includes: nodes, consensus mechanisms, and transaction data. This chapter will start from these aspects, introducing the Blockchain-based trusted EHRs system designed being designed and its key underlining/enabling technologies.

4.1. Overall Design of System Architecture

The proposed architecture of CB-EHRs platform is designed as a layered architecture. The entire platform is divided into three layers, comprising of: (i) The user interface layer, which is at the top; (ii) the business logic layer, which is in the middle; and (iii) the data access layer, positioned at the bottom of the framework. Figure 4-1 presents an overview of the CB-EHRs architectural framework.

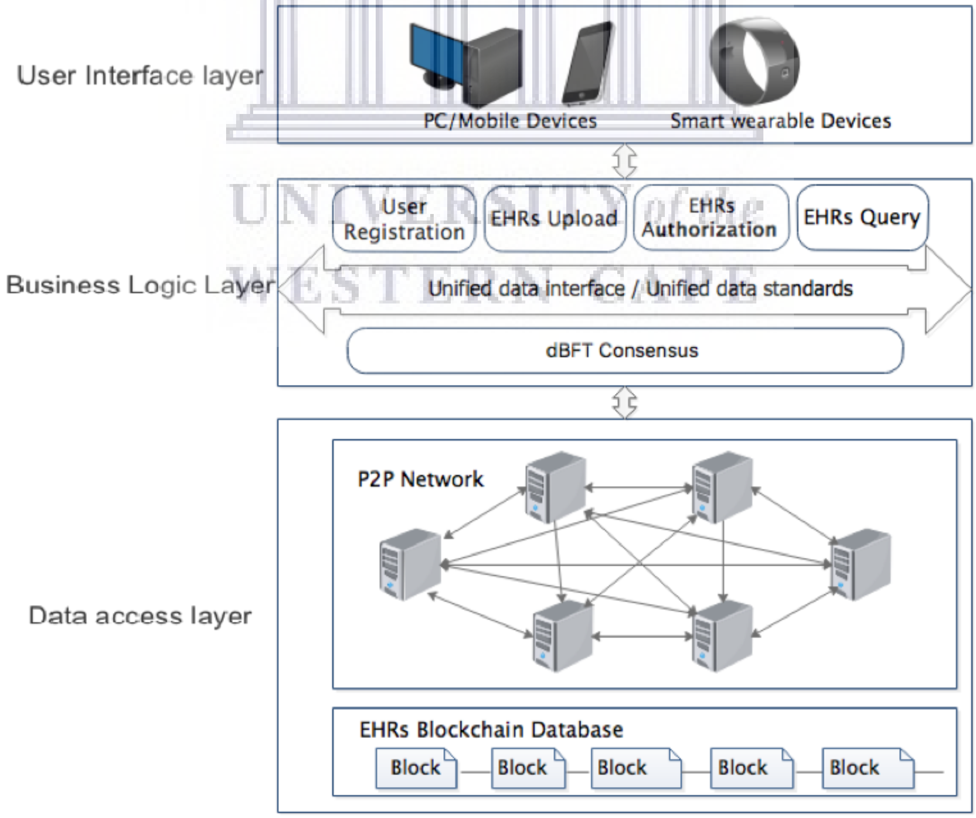


Figure 4-1: CB-EHRs System Architecture

- User interface layer： This is used to display data and receive user's input information. It provides users with interfaces that can interoperate with the entire Blockchain platform. The Blockchain-based EHRs platform proposed in this dissertation can be used on both mobile devices and PCs. On one hand, medical workers need to use PC to fill out electronic health records, and users are able to access their own electronic health records. On the other hand, users can also use their mobile phone to share electronic health records with doctors at any time and any place, when they need to conduct remote diagnosis. Hence this promotes ubiquitous access.
- The business logic layer: is used to provide data conversion between actual users and Blockchain-based EHRs platform. It provides users with a unified data interface and unified data standards. Business logic layer can encapsulate user data into virtual transactions and assets, then transfer these transactions and assets to nodes in the Blockchain network. In the end, the business logic layer will store new user data in the Blockchain database.

There are four interfaces, namely: (i)User registration; (ii) EHRs upload; (iii) EHRs authorization; and (iv) EHRs query. The registration interface implements the registration function for new users. The Record Upload and Record Authorization interface provide users with services for uploading and signing their own medical records. The Record query interface provides medical staff with a rich EHRs database that facilitates the research process for various diseases.

The business logic layer also contains a dBFT (delegated Byzantine Fault Tolerance) consensus module. This module acts on the P2P network and is responsible for running a set of nodes that implement the consensus mechanism. The dBFT Consensus Module works with these nodes to validate transactions and maintain distributed ledgers. Through the interaction of these modules with the previously mentioned interfaces, data exchange between users and the Blockchain database can be smoothly realized.

- The data access layer contains a unique Blockchain and a P2P network that maintains the Blockchain. The P2P network is used to maintain the operation of the CB-EHRs platform. It can receive transaction verification request, generate blocks and vote on new blocks. The verification nodes in the network agree on the transaction and transaction sequence by running the dBFT consensus mechanism. The platform can then generate blocks and update the local Blockchain database. Another part of the data access layer is the Blockchain. It is a unique Blockchain maintained by the entire network. This Blockchain holds all validated EHR in the system.

4.2. Selection of Consensus Mechanism

Given that the business characteristics and business processing logic of different Blockchain applications are different, they have different requirements for the transaction processing capability of the underlying Blockchain network. Therefore, we use the Blockchain data storage feature to implement the system and also need to consider the business characteristics of EHRs management. Based on the selection of the appropriate Blockchain framework, we should also choose the appropriate consensus mechanism.

The choice of consensus algorithm is closely related to the application scenario. Based on the analysis and comparison of Table 2-1, PoW and PoS are mainly applicable to the digital currency system. They all need to consume a enormous amount of resources and are not suitable for commercial application services. DPoS is suitable for Blockchain systems that rely on token operations, but has poor support for Blockchain systems that operate without tokens. RPCA is currently only available in the currency or electronic asset clearing area and has poor support for other applications. PBFT and dBFT are widely available for commercial applications because they do not have an underlying token mechanism. In order to further select a suitable consensus mechanism for the CB-EHRs system. This dissertation will focus on the PBFT and dBFT consensus mechanisms.

4.2.1. PBFT Consensus Mechanism

PBFT requires the system to maintain a state together, and the actions taken by all nodes are consistent. To achieve this, three types of basic protocols need to be run, including: consistency protocols, checkpoint protocols, and view replacement protocols. At the same time, the consistency protocol requires that requests from clients be executed in a certain order on each service node. This protocol divides nodes into two categories (master and slave). There is only one master node and is responsible for request sequencing, and the slave nodes process requests according to the ordering.

Figure 4-2 shows one round of PBFT consensus process (consistency agreement). Each client request requires 5 stages to complete. PBFT executes the client's request after the server reaches an agreement by using two-by-two interaction. Because the client cannot obtain any server operating status information from the server, the server can only be monitored by the server for errors in the master node in the PBFT protocol.

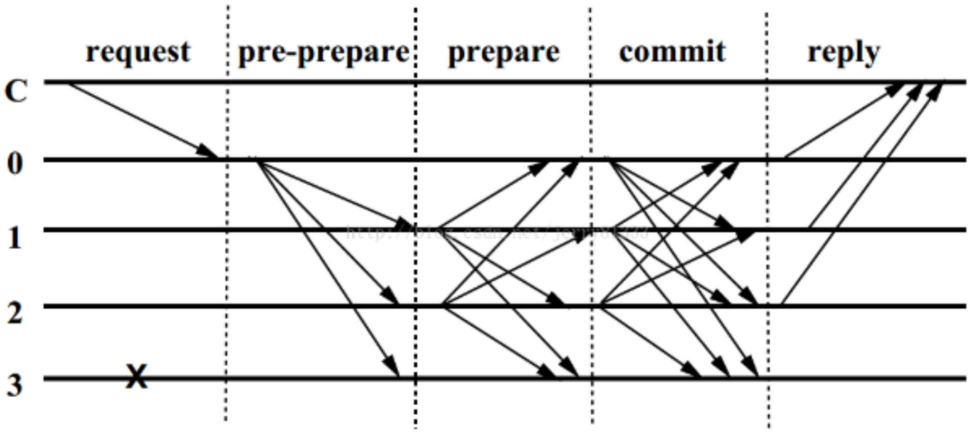


Figure 4-2: One Round of PBFT Consensus Process

Ensuring the correctness of PBFT is mainly divided into three stages: pre-preparation, preparation and confirmation.

- In the preparation phase, the master node sends a sequence number k, a message m and a signature digest d. If the signature is correct and the backup node i has not received the pre-preparation information under the same k value and view, then i will enter the pre-preparation phase.
- After entering the pre-preparation phase, the backup node will broadcast a preparation message across the entire network. For the same view, when the node receives no less than 2f of the preparation messages matching the local preparation phase, it indicates that the node is already in the preparation phase. Therefore, for a certain view, the non-faulty replica node has agreed on the overall order of the requests. Once $2f + 1$ non-faulty nodes are ready, the network can enter the confirmation phase.
- The replication node broadcasts a confirmation message at each confirmation stage. Once node i receives $2f + 1$ confirmation message, it indicates that the node is already in the committed-local phase. Eventually, even if the view changes, a system with a locally confirmed node will complete the confirmation.

The design idea adopted by PBFT is to put all the work on the server side, such as reaching consensus and monitoring the Byzantine master node. Therefore, its consistency protocol design is more complicated. There are two phases that require two-to-two interactions between servers, with a large amount of data processing and complicated calculations.

4.2.2. dBFT Consensus Mechanism (Selected in this dissertation)

NEO proposed the dBFT consensus algorithm based on the PBFT algorithm. The algorithm determines the nodes participating in the next round of consensus based on the real-time voting of the blockchain. Such a design effectively reduces the time consumption of the algorithm, thereby increasing the block production speed and reducing the transaction confirmation cycle.

A round of dBFT consensus has the following four steps:

1. The speaker initiates a consensus and broadcasts Prepare Request.
2. After receiving a Prepare Request, delegates broadcast Prepare Response.
3. After receiving enough Prepare Responses, the consensus node broadcasts a Commit.
4. After receiving enough Commits, the consensus node generates new blocks and broadcasts them.

dBFT is the same as DPOS in that the selected super nodes (accounting nodes) perform collaborative accounting. In the case of weak centralization, they all achieve high efficiency, but the difference lies in the specific algorithm and node selection.

The general principle of the DBFT algorithm is as follows. Participating in bookkeeping are super nodes. Ordinary nodes can access the consensus process and synchronize ledger information, but do not participate in bookkeeping. A total of n super nodes is divided into one speaker and n-1 delegates. The Speaker is elected in turn by all super nodes. Each time the account is booked, the speaker first initiates a block proposal. Once at least (2n + 1) / 3 bookkeeping nodes (speaker plus delegates) agree to this proposal, then this proposal becomes the final released block. The released blocks are irreversible, and all transactions in the blocks are 100% confirmed.

The network based on the PBFT mechanism assumes that "the message may not be delivered, delayed, duplicated, or delivered out of order," and public-key encryption is used to authenticate the replica. The same is true for NEO's dBFT algorithm. Because the algorithm does not rely on synchronization to ensure security, it must rely on synchronization to ensure activity.

The PBFT mechanism considers that the client directly interacts with the master node and broadcasts the message, while receiving an independent response from 2f + 1 nodes in order to perform the next operation. A similar situation exists with the dBFT mechanism. In the dBFT, information is transmitted through the point-to-point network, but the location of the consensus nodes is unknown (in order to prevent direct delay attacks and denial of service).

In PBFT, the client performs atomic commit operations at different timestamps, and these operations are processed and published independently. The dBFT mechanism is different. Consensus nodes must group transactions into batches. Each group is called a block. Due to different grouping methods, there may be thousands of valid blocks at the same block height. Therefore, in order to ensure the final certainty of the block (that is, there is only one block for a given block height), the dBFT mechanism takes into account that the "client" also fails. PBFT does not take this into account.

In terms of the choice of consensus mechanism, this dissertation refers to a part of the underlying Blockchain architecture of the Fabric Blockchain. The reason is that this framework provides a pluggable consensus interface that can set up consensus protocols based on specific requirements. However, the shortcoming is that the current Blockchain network provides only one PBFT consensus mechanism for multiple verification nodes. PBFT is able to cope with failed high-frequency consensus business and ensure that all transactions that have reached the prepared state can be directly assembled in the next round of consensus when the consensus fails. PBFT can be divided into three phases: pre-prepare, prepare, and commit. However, its process of verifying communication between nodes is too complicated and reduces the consensus efficiency of transactions. Moreover, the more nodes join PBFT consensus, the quicker the performance drops, as the time complexity of the PBFT is $O(n^2)$[73]. Therefore, the PBFT mechanism cannot support large-scale network node activities.

According to the actual situation involved in the project, the Blockchain proposed in this dissertation adopts the alliance chain. The nodes in the node network are composed of medical institutions, and the nodes are basically trusted. On the one hand, the main business logic in EHRs management is relatively simple and involves few transaction types. On the other hand, because of its shared enactment, the number of transactions that the system needs to process will grow on a large scale as the number of users increases. Therefore, the system is less likely to fail in the normal network environment. On the contrary, the system has higher requirements for the consensus efficiency of processing transactions. Based on this feature, this dissertation adopts the consensus-efficient dBFT consensus mechanism to better support the upper-layer application. dBFT is an improved Byzantine fault-tolerant protocol based on PBFT. It also combines the characteristics of DPoS. By voting on the Blockchain, Blockchain network authorize a few nodes as the accounting nodes. After the simplified PBFT consensus (dBFT consensus), the accounting nodes reach consensus and create new block. This consensus mechanism simplifies the process and improves efficiency compared to PBFT, but retains all the excellent performance of PBFT. Table 4-1 is a comparison of PBFT and dBFT.

Table 4-1: Comparison of PBFT and dBFT Consensus Mechanisms

Consensus Mechanism	PBFT	dBFT
Application range	It cannot be applied to applications with large node due to complex communication processes.	It simplifies the process compared to PBFT, and can be used in the current EHRs system.
Performance	The more nodes join PBFT consensus, the quicker the performance drops, as the time complexity of the PBFT is $O(n^2)$.	Select a few nodes to reach consensus by voting. The system is highly efficient.

4.3. P2P Network Design

4.3.1. The Overall Structure of the P2P Network

In the P2P network of the Blockchain, all nodes jointly maintain a network for transaction and block exchange. The network structure of the CB-EHRs system draws on the underlying core code of Bitcoin technology, the P2P network. According to the characteristics of the P2P network, this dissertation designs a special electronic medical P2P network. Due to the special structure of the designed P2P network, each node in the network has the complete EHRs data replication. According to the rules of the alliance chain, the nodes in the Blockchain network are composed of various medical institutions throughout the country. The user's device does not need to store a huge electronic health record. Each medical institution has equal status in the network, they have same rights and need to assume same obligations, as can be seen in the overview presented in Figure 4-3.

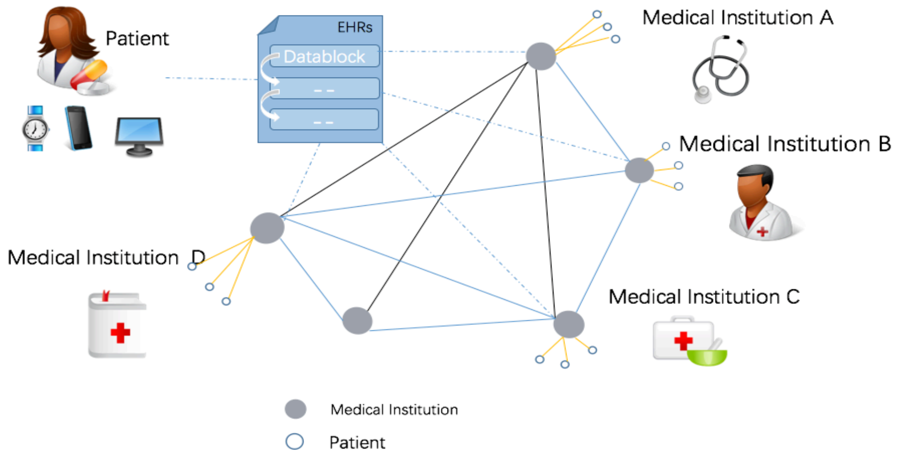


Figure 4-3: Blockchain EHRs Distributed in the Network

Compared with the traditional EHRs system, the CB-EHRs system developed in this dissertation has a new network structure applicable to the field of e-health. This structure is the key to solving the difficulties of data sharing between medical institutions and protected data security, as can be seen an overview in Figure 4-3. The traditional EHRs system uses centralized network structure, its electronic health records databases of medical institution are independent of each other and its institutions are all interconnected. This structure cannot share data between institutions and it completely depends on the central node. If the central node fails, the entire database will become inaccessible. Conversely, every institution in the Blockchain-based EHRs system has the completed electronic health records data. They can easily share data between themselves. This means that no matter which medical institution the user visits, the doctor can get the full electronic health records of the user, thus providing a huge convenience for patients' diagnosis. Furthermore, this CB-EHRs system provides an advantage in terms of database security such that even when one or more nodes fail, the database is still accessible, because the other nodes can still form a complete network. A comparison between the properties of traditional EHRs system and Blockchain-based EHRs system is shown in Figure 4-4 and summaried on Table 4-2.

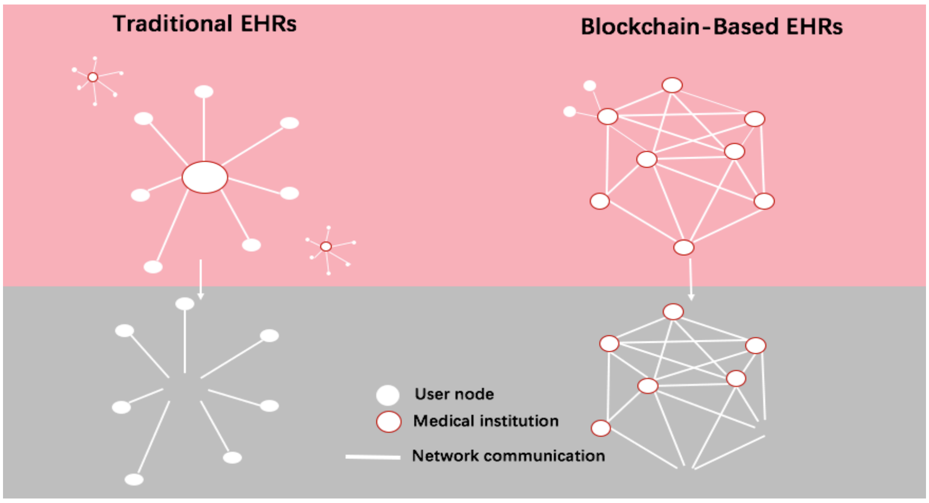


Figure 4-4: Comparison of Network Structure between Traditional EHRs System and Blockchain-Based EHRs System

Table 4-2: Comparison between Traditional EHRs System and Blockchain-Based EHRs System

Platform type	Network structure	Data sharing	Stability	Data tamper-proof
Traditional EHRs Platform	Centralized network	Data sharing is difficult.	If the central node is damaged, the entire platform is not accessible	No
Blockchain-Based EHRs Platform	Decentralized network	Easy to share data between medical institutions	Damage to one or more nodes will not affect the operation of the platform.	Yes

4.3.2. Initialization

When a new node accesses the network, it needs to know about the other nodes in the network and synchronize the Blockchain. In order for the newly joined node to recognize other nodes, there are multiple node IP addresses in the system code that can provide similar DNS services. These nodes are able to scan active nodes in the current network and record their IP addresses.

Once a new node attempts to join the Blockchain network, it randomly initiates a request to a DNS-like node in the system to obtain other active nodes in the network. Due to the strong activity of the nodes, some nodes may be offline. The new node will initiate multiple requests until the request is responded by a node and returns a list of IP addresses of the current network active node.

If however, the system relies solely on these types of DNS nodes to return data, the data is likely to be attacked or directed to a hosted forged network. Therefore, when a node comes online, other nodes connected to the node send the IP address and port number of other nodes in the network to it, and use a decentralized method to verify safety of the node.

4.3.3. Connection between Nodes

Once obtaining the IP address of other nodes, the new node will send its own version of information to the correspond node to try to establish a connection. This version information includes the system version of the node, the block that has been synchronized, the current system time of the node, etc.

Once received, the correspond node will return its own version information. When both parties get the version information of the other party, a confirmation message is sent. At this point the connection between the two nodes is successfully established.

The time synchronization process on the node is run by default, which ensures that the system time is synchronized with the standard time server. After receiving the version information sent by the correspond node, the two nodes trying to establish a connection check the time of the system and confirm that the system time of both parties is synchronized. If the system time of both parties is not synchronized, the connection cannot be established normally.

The connection between each pair of node needs to be maintained. Every 30 minutes, nodes send messages to correspond node. This period of time is called keepalive cycle. The previously mentioned information becomes keepalive information, which is used to tell the peer node that it is still active. If the peer's keepalive information is not received within 3 keepalive periods, the node will disconnect the link.

4.3.4. Synchronization of Blocks

When the first node joins the Blockchain network, there is only one block in the Blockchain in its local storage. This block is called the Genesis block. Different Genesis blocks identify different Blockchain. Before the node becomes a complete node that can confirm the transaction and generate the block, it needs to download all the block data on the longest Blockchain in the network. This process is called block initialization. When a new node is connected to the Blockchain network, this node randomly selects a node in the network for block synchronization. The selected node is called a synchronization node. The new node sends a Req_Headers message to the sync node. The content of the message is shown in Figure 4-5.

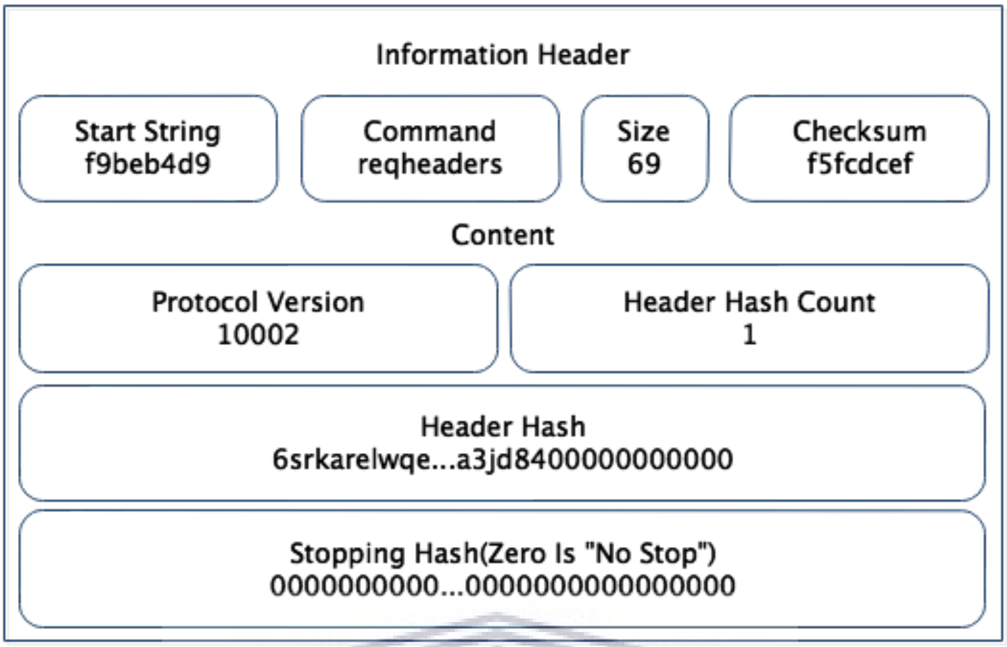
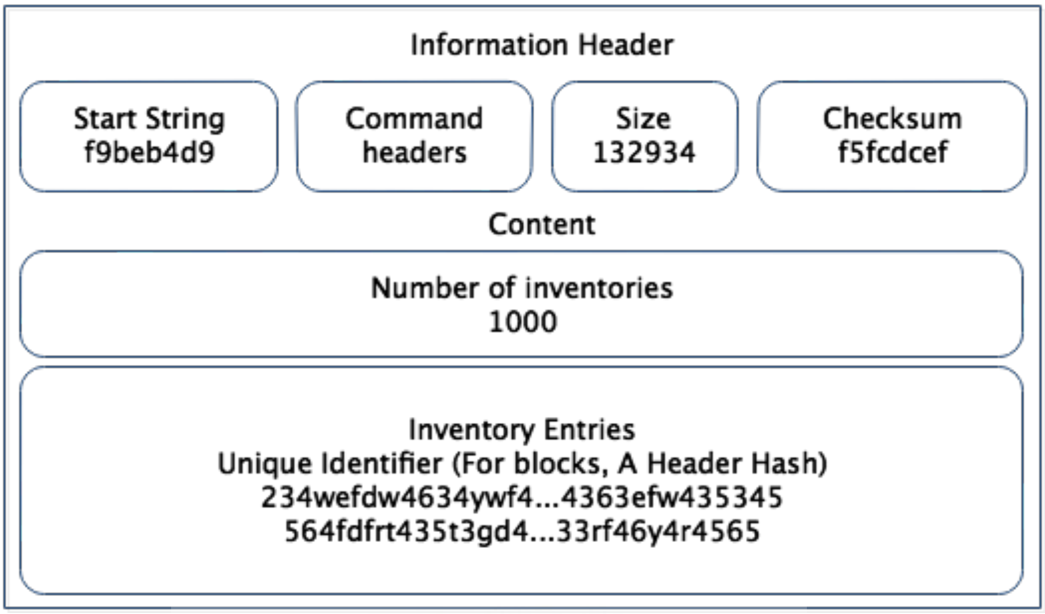


Figure 4-5: Req_Headers Message

In the header hash field of the Req_Headers message, the new node fills in the header hash of the block it owns, which is the head hash of the Genesis block. At the same time, the node will also use the 0 completion to stop hash value field, indicating that the maximum number of responses can be requested. This dissertation sets this maximum to 1000.

After receiving the Req_Headers information, the synchronization node returns a header information according to the request in the information. The specific content of the header information is shown in Figure 4-6.



Information Header

Start String
f9beb4d9 | Command
headers | Size
132934 | Checksum
f5fcdcef

Content

Number of inventories
1000

Inventory Entries
Unique Identifier (For blocks, A Header Hash)
234wefdw4634ywf4...4363efw435345
564fdfrt435t3gd4...33rf46y4r4565

Figure 4-6: Headers Message

This header information contains the header hash values of the 1000 blocks starting from block 1 on the Blockchain.

After receiving the header information returned by the synchronization node, the new node initially determines whether the header hash in the header information is correct according to the consensus mechanism and the target number. After that, the new node will then send the Req_Headers message again to request the subsequent 1000 block header hashes until it receives a header with less than 1000 hashes returned from the sync node.

Thereafter, the new node repeats the same block initialization process to the other nodes. If there is an untrusted sync node in the network, it will be discovered by comparing the block header data. After confirming that the acquired header information belongs to the optimal Blockchain in the current network, the new node sends the request data information to the complete node in the network to obtain the complete block information. The information for requesting block data is shown in Figure 4-7.

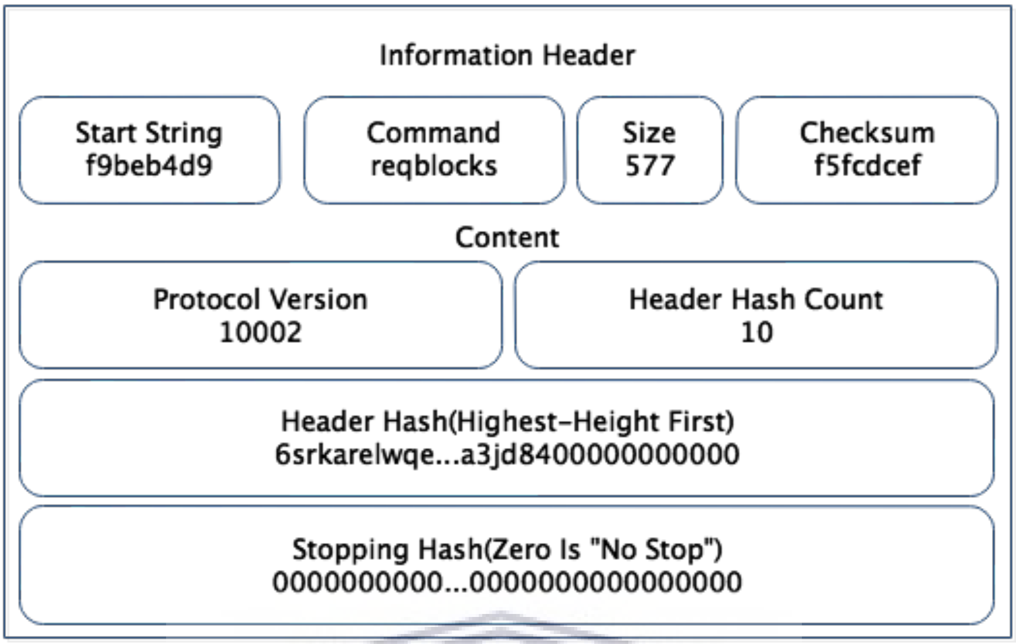


Figure 4-7: Req_Block Message

The new node randomly requests complete information from any of the blocks in the network. However, the new node will verify the correctness of the block according to the order in the Blockchain. This leads to the possibility that the next block which needs to be verified has not yet been obtained from the peer node. When this happens, the system waits for a while. If the block has not been received, the new node will disconnect from the node and request block information from other nodes.

A node performs block synchronization not only when it first joins the Blockchain network. When a node finds that a large number of blocks need to be synchronized, it can also synchronize the blocks. For example, after a node has been offline for a long time, it may need to perform block synchronization to get a large number of blocks generated during its offline. The flow of block synchronization for the new node is shown in Figure 4-8.

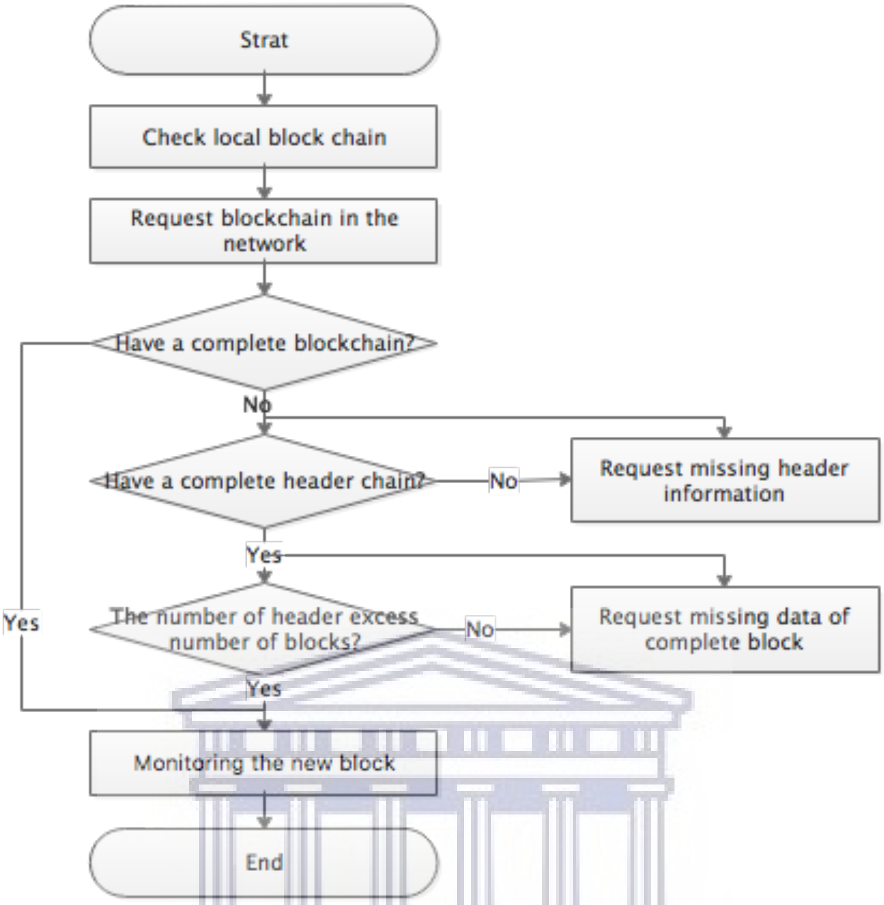

Strat
Check local block chain
Request blockchain in the network
Have a complete blockchain?
No
Have a complete header chain?
No
Request missing header information
Yes
The number of header excess number of blocks?
No
Request missing data of complete block
Yes
Yes
Monitoring the new block
End

Figure 4-8: Flow Chart of Block Synchronization

4.4. Role-based Access Control Structure Design

In order to better control user permissions, the EHRs system uses the Role-Based Access Control scheme. Role-Based Access Control (RBAC) means that user rights are not directly specified by the system, but are obtained by associating with the role and permission data tables, as shown in Figure 4-9.

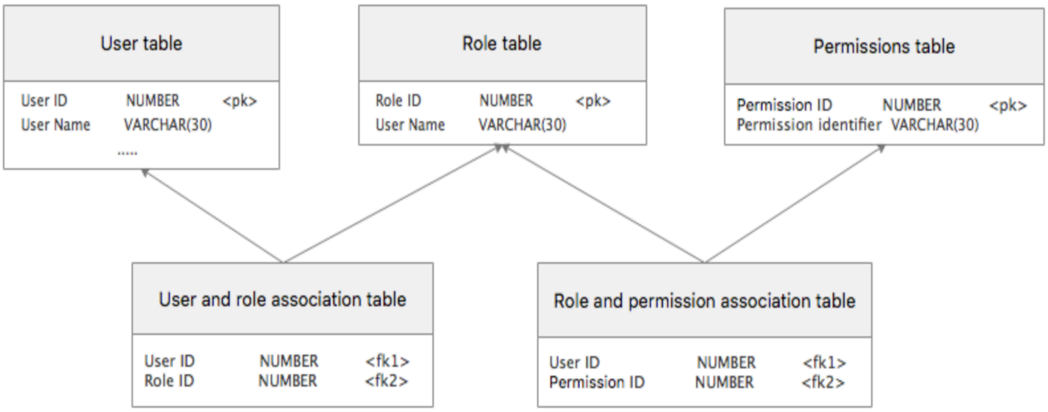


Figure 4-9: Association between Roles and Permission Data Tables

The default roles of the CB-EHRs system are divided into three categories: patient users, doctor users, and system administrators. (i) The patient can only perform inquiry, verification and mail operations of EHRs belonging to the currently logged in user. (ii) The doctor has the authority to add and inquiry EHRs and is able to access all EHRs in the database for analysis of the disease or condition of the patient. Doctors can also perform information release and sending and receiving operations. (iii) The system administrator has the highest authority of the system, can create new roles and users, change role permissions and user roles. The system administrator also has permission to change system settings. The specific RBAC structure of the CB-EHRs system is shown in Figure 4-10.

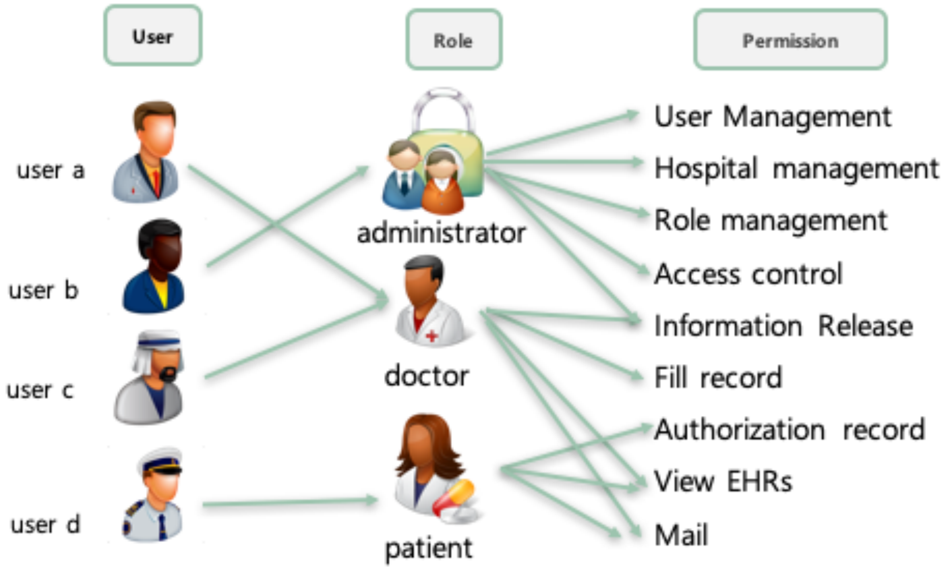


Figure 4-10: The Specific RBAC Structure of The CB-EHRs System

Each module of the EHRs system performs a strict permission check as shown in Figure 4-11. The specific process is as follows.

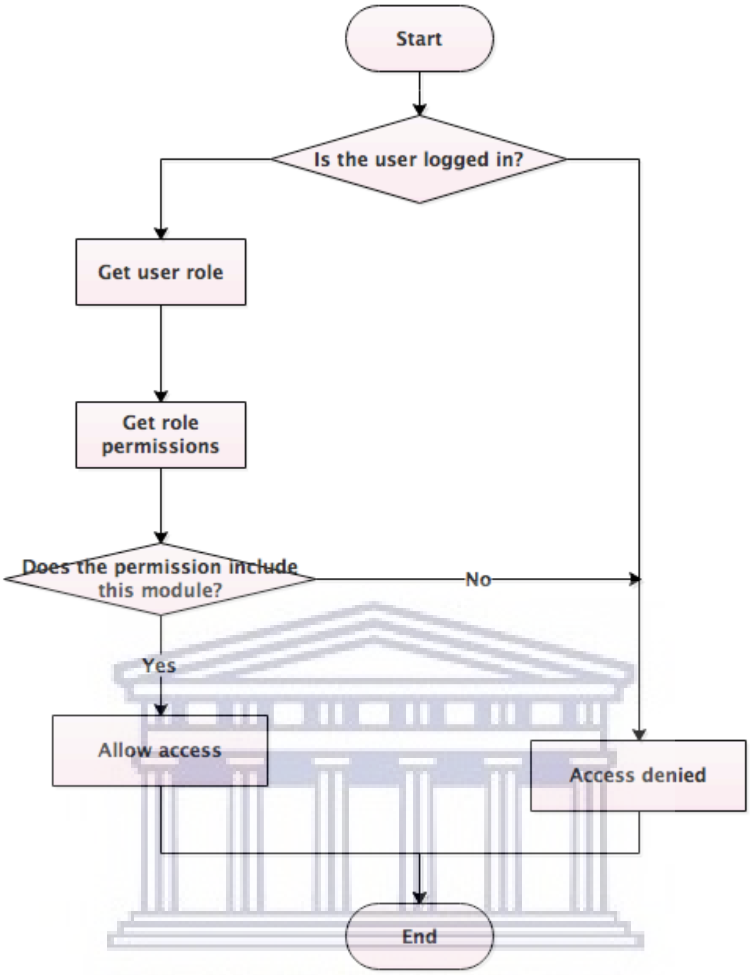


Figure 4-11: Permission Check Process

a) First, the system checks the login status of the currently accessed user. If the user is not logged in, the system prompts the user to log in to the system and end the permission checking process of the module. If the user has logged in, the system obtains the user ID and continues the permission check of the module.

b) Using the user ID obtained in step a, the system obtains the role ID from the user table. Then according to the ID of the role, the system further obtains the permission ID from the role table.

c) The system obtains a specific permission list from the permission table by using the permission ID obtained in step b.

d) Finally, the system determines whether the permission list includes the operation rights of the module. If the permission list includes the operation permission of the module, the user is allowed to continue access. if it does not include the operation permission of the module, the user is denied access and the service page is hidden.

4.5. Process Structure and Detailed Design of The Module

4.5.1. Detailed Design of User Registration and Login Module

According to the overall requirements and design of the system, the functional design of the user registration login module is divided:

(1) User registration module

The flow chart for new user registration is shown in Figure 4-12:

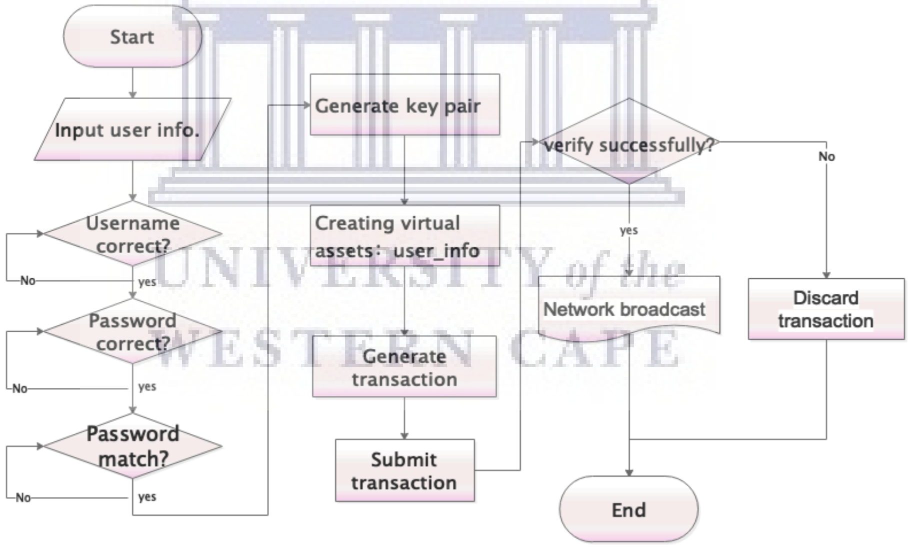


Figure 4-12: Process of User Registration and Identity Information Validation

The system registration module is the entrance for new users to enter the system. To be part of the CB-EHRs system node and share data in the system, new users must be registered as a system user. The system designed in this dissertation simplifies the user registration operation. The new user only needs to fill in the user name, e-mail, set the password, and confirm the password in the password to successfully register. However, the e-mail entered by the new user cannot have the same data in the account table. The new user registration operation requires three steps:

1. The first step is to fill in the username and e-mail to facilitate user identification and system management. The system sets the e-mail to a mailbox that must be 6 to 16 digits or letters;
2. The second step is to set the password. In order to protect the security of the user account, the password must be set to 6 to 16 digits. If the password is too short, the security is lacking. On the contrary, the password is too long to remember;
3. The third step is to confirm the password. In order to ensure that the password entered by the user is set in the mind of the user, it is necessary to confirm the password to ensure consistency.

When a new user registers, if any of the above three steps does not meet the requirements, the registration request cannot be verified by the system. Account information will also not be added to the database. Only the user information that meets the requirements will be successfully stored in the database for verification when logging in.

In our proposed CB-EHRs system, medical institution is like a special type of transaction body. User's key pairs and unique identifier will be generated when patient join in CB-EHRs platform as an user. According to the setting of the CB-EHRs framework in this dissertation, a transaction is first initiated by the user, then the user transfers the virtual asset USER_INFO with his or her own identity information (public key and unique identifier) to the medical institution. In this way, the medical institution can obtain the user's public key and unique identifier to verify subsequent actions that may come from this user.

(2) User login module

The flowchart for user login is shown in Figure 4-13.

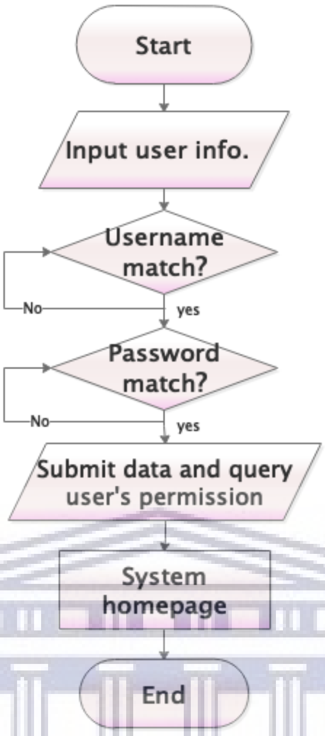


Figure 4-13: Process of User Login

Successfully registered users only need to log in to access the system home page. In the design of this system, the user's login operation is not complicated. Users only need to enter an e-mail address and password to achieve. User login operations are divided into three steps:

1. In the first step, the user enters the e-mail address used in the registration in the e-mail input box. Since the e-mail address is restricted during the registration operation, the input condition should also be set during the login operation, which is convenient for reminding the user to input the correct e-mail;
2. The second step is to input the password corresponding to the account. Similarly, when the password is input, a restriction condition should be set to remind the user to input the correct password corresponding to the

account;

3. The third step, click on the login button, the form will submit the e-mail and password to the background for processing.

When the form is transferred to the background of the system, the back-end code is looked up in the database account table via the e-mail address. If the entered e-mail does not exist in the table, or the found password does not match the input password, the user login fails. If there is a corresponding e-mail address and password, the user logs in successfully. Finally, the system will query the database for the user's role and role rights based on the user's unique e-mail address. According to the query result, the user system will jump to the corresponding page.

4.5.2. Detailed Design of Data Authorization Module

According to the overall design of the project, this module is an important part of the project. It is a key part of combining hash encryption algorithm technology with a data sharing platform. This module mainly implements user authorization and conditional query of EHRs. The process structure of this part are shown in Figure 4-14.

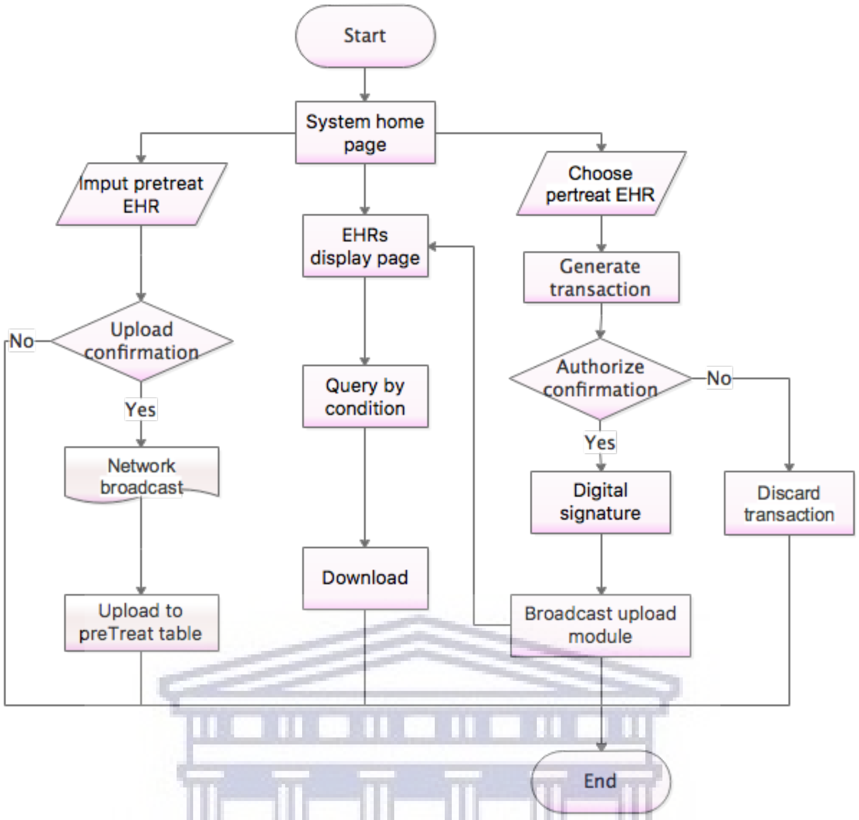


Figure 4-14: Process of User Authorization

According to the design of the module part, this dissertation divides the module by function:

(1) Pre-treatment EHRs upload function

Doctors are able to upload pretreat EHRs on the pretreat page. Considering that doctors sometimes upload more than one EHR for the same file description, the front page can freely add or delete added files, and the back end only needs to process EHR one by one. After the completion of the pretreat EHR by the doctor, the pretreat EHR confirmed to be uploaded will be transmitted to the network node in the form of a broadcast. It is then passed to the pretreat table in the database.

(2) Data authorization function

In order to better combine the hash encryption algorithm and data sharing storage, in the design and implementation of this module, this dissertation also designs the underlying related technology in detail.

Hash algorithm design: The system will generate MD5 values for the EHR selected by the user. Therefore, this dissertation needs to obtain the MD5 value of the authorized EHR through the relevant algorithm and code design. In the Java development package, the java.security.MessageDigest class implements the mainstream hash hashing algorithm, which implements the initialization of the MD5 algorithm in this class through MessageDigest.getInstance ("MD5"). Since the system produces MD5 values for EHR files, the java.nio.MappedByteBuffer class and java.nio.channels.FileChannel class are also needed to process the input files. When the program generates the MD5 value, it also initializes the processed file data through the update() function in the MessageDigest class. Afterward, the program calls the digest() function to get the MD5 value represented by the byte data. Finally, the program only needs to use the byte array to string processing function to get the MD5 value in the form of a string.

Design of digital signature algorithms: The design of the digital signature algorithm in this dissertation considers the characteristics of the algorithm. If the system directly digitally signs the original EHRs information, the length of each digital signature output is different due to the inconsistent size of the EHRs. The size of the signature input data also affects the time of the signature process. Based on the consideration of this feature, it is necessary to produce an MD5 value for the EHR file before signing. The program then uses the splicing of the EHR content, upload user, upload time, MD5 value, and file description information as input to the digital signature, which greatly reduces the signature time and verification time.

The digital signature algorithm used in this dissertation is MD5withRSA. In the Java development kit[100], the java.security.KeyPairGenerator class provides function for instantiating and producing key pairs. This function needs to be combined with KeyPair, RSAPrivateKey and RSAPublicKey to implement a public and private key for signing and verifying data. The MD5withRSA algorithm is instantiated by KeyPairGenerator. getInstance("MD5withRSA"), and the key pair is instantiated by calling the generateKeyPair() function.The digital signature function is provided in the java.security.Signature class during the digital signature process. Before signing, the program needs to call the initSign() function and the update() function to initialize the signature private key and update the data. Then the program can get the byte array digital signature by the sign() function. Finally, after processing the array, the system can obtain the signature information in the form of a string.

(3) Conditional query unction design

After entering the system home page, users can see the latest authorized EHRs on the EHR library page of the interface. In order to facilitate users to quickly select the EHRs files of their interest. Users can perform conditional queries by entering keywords in the query input box. After clicking on the query, the front-end code passes the keyword to the background. The backend code can filter EHRs with this keyword and return the results to the front page for display. Through the implementation of this function, doctor's work can be made more convenient. Medical researchers can access all EHRs, as well as access specific users or EHRs for specific diseases.

4.5.3. Detailed Design of Broadcast Upload EHRs Module

The broadcast upload module mainly works on the P2P network. After the EHRs authorized by the user are passed into the module, the broadcast upload module transmits them to other nodes in the network. After processing by the consensus mechanism, the accounting nodes will verify these EHRs. The final legal EHRs will be processed into blocks and saved to the database.

Design of P2P distributed network: P2P is an infrastructure in which various system nodes in the network can communicate with each other[101]. Since the nodes in the network are equal, the design of the P2P network module uses multicast and combines the overall design of the system with the basic communication mode. In the Java development kit, the java.net.MulticastSocket class can be used to send and receive IP multicast packets, and the MulticastSocket class can be used to send a single message to multiple clients. By defining a special set of network addresses as multicast group addresses, hosts on the network can join the multicast address by calling the join() function. When nodes joining the multicast network send messages, they need to serialize the data. These nodes then encapsulate the serialized data using the DatagramPacket packet. Finally, it is sent out via the send () function in the MulticastSocket class. In the above process, the data that needs to be broadcasted is composed of the input data during the signature, the digital signature and the signature public key. This combination facilitates the next data verification work.

Data verification process: After the user selects the EHRs data that needs to be authorized and uploaded, the system will encapsulate and sign the EHR and transmit it through multicast. The nodes in the network need to verify the encapsulated data after receiving it. Verifying the integrity of the data or verifying that the data has been tampered with is necessary for the system to run. Any selected accounting nodenode in the multicast can receive the data information through the receive () function in the MulticastSocket class. At the same time, the node can deserialize the received data to obtain relevant data information, and then the node needs to verify the data. In the process of implementing data verification, the java.security.Signature class provides a digital signature verification verify() function. When the node verifies the signature, it also needs to call the initSign () function and the update () function to initialize the signature public key and update the data. Finally, the program will verify by the verify () function and return the Boolean type validation result.

4.6. EHRs Blockchain Verification and Database Design

4.6.1. Verifying the EHRs Blockchain Process

(1) Update of log files

Based on the overall requirements analysis of the system, the log file is designed to be updated once an hour. Each log file is recorded with a data transfer record for one hour. Starting at the hour of the hour, the node generates a new log every hour. The new log file is named using the string of the hour when it was generated. The beginning of the file of the new log file records the Merkle Tree root of the previous log file and the timestamp string when the file was generated. The system will then continue to record the user-uploaded information that was verified during the hour after the new log file was generated.

In the process of verifying the EHRs Blockchain, the verified data transfer messages are recorded into the file in succession using the format of the recorded text design. When the new log file is generated, the system needs to extract the Merkle Tree root of the previous log file. All the log files in the system record the data of the previous file one by one from the back to the front. These files are concatenated to form a one-way chain that fully reflects the tamper-proof and traceability of Blockchain data.

(2) Verification of log files

In order to ensure the uniformity of the EHRs stored in the system and to realize the feature that the nodes jointly maintain a reliable database, all nodes in the system will periodically communicate and verify the relevant information of the latest recorded files. In the design of the module, the system will communicate every 5 minutes. All nodes send the Merkle Tree root value in their latest log file to the network, while receiving the Merkle Tree root value sent by other nodes.If the node's own Merkle Tree root value is consistent with more than half of the Merkle Tree root value from other nodes, it means that the local log file is consistent with the overall system. If your own value is different from more than half of the value, there is a data loss in the local record file. Therefore, the node needs to delete the latest local log file and then request other nodes in the network to download the latest correct log file data.

4.6.2. Log File Design

Asymmetric encryption technology is an important part of this project. This dissertation combines hash algorithm technology and data sharing to demonstrate the use of hash encryption algorithm in daily life. The log file is an important credential for historical transfer messages in this system. In the log file update function, the system will write the file multiple times. This dissertation fully considers the efficiency of file read and write operations and how to improve the readability of the file. Therefore, this dissertation defines the message record format in the log file text as follows.

Email_author_time_md5_description

Email is the email address of the user; author is the username of the uploaded EHRs; time is the time when the file is uploaded; md5 is the MD5 value of the EHR content; description is the description of the EHR file when the EHR file is uploaded.

In the log file, each log file records all successful upload messages for the entire network in one hour. A new log file is generated every hour of the system starting. The beginning of the new log file records the Merkle Tree root value of the previous log file and the timestamp when the current log file was generated. The new log file should also record the entire network message for the next hour, as shown above. This design greatly improves the efficiency of the system to update the log file. The new log file is named with the current time so that the system can quickly determine the scope of the query by reading the log file name when performing the query operation. The system no longer needs to open all log files to query EHRs. Therefore, log files increase the efficiency of the system.

4.6.3. Database Design

When the user performs a registration login operation, the system needs to manage the user data. After the new user enters the user name, email address and password through the registration page, the system submits the above information to the user table for database update. If the data is not saved successfully because the same email address exists, the user may have registered an account. If the data is updated successfully, the new user can complete the registration. This dissertation considers the above factors and designs the structure of the user table as shown in Table 4-3 in the MySQL data.

Table 4-3: The Structure of the User Table

Field Name	Datatype	Len	PK?	Not Null?	Note
id	biaint	20	PK	√	User's identifier
user_name	varchar	40		√	Username
email	varchar	30		√	Email address
password	varchar	100		√	Account password
birth	date	0			User's birthday
phone	varchar	20			Phone number
sex	smallint	6			User's gender

When the user performs an EHR authorized operation, the system needs to record the data authorized by the user. In theory, this part of the data can be stored in various forms. However, considering the overall requirements and system design of this dissertation, the EHRs uploaded by the user and the description information of the EHRs are independent of each other, but they are related to each other logically. Therefore, the system stores the original data and block separately and stores the related description information such as the MD5 value of the EHRs in the form of a database table. This method facilitates the management of data information and the query of EHRs. Based on the above factors, the structure design of the EHRs information table in MySQL data is shown in Table 4-4.

Table 4-4: The Structure Design of the EHRs Information Table

Field Name	Datatype	Len	PK?	Not Null?	Note
id	biaint	20	PK	√	EHR unique identifier
user	varchar	40		√	Owner
doctor	varchar	40		√	Diagnostic doctor
time	datetime	0		√	Upload time
hash	varchar	100		√	MD5 value of EHR
description	varchar	200		√	Description of EHR

4.7. Summary of Chapter 4

This chapter mainly determines the basic architecture of the CB-EHRs system from two aspects. First of all, this chapter introduces the overall architectural design of the system. Then give a detailed design of each module of the system. In the overall architecture design of the system, this chapter presents a structure of the system architecture. This chapter divides the system into three sub-modules successively, and then analyzes the specific design of each module. This chapter also introduces the design of the consensus module and RBAC mechanism involved in this project. Finally, this chapter expounds the format design of the text content of the log file and the design of the database table.

5. Implementation of CB-EHRs System

Through the system framework design, data model analysis, functional module design, this chapter will compile and implement these functions. The following is an introduction to the code module and main files.

5.1. Development Environment

The CB-EHRs system is mainly written in Java language[102] and HTML language[103]. The Java language is an object-oriented programming language with very powerful features and is easy to use. It can be used to write cross-platform applications and is widely used in the fields of PC, supercomputer, mobile phone and internet[102]. It includes all the advantages of the C++ language but does not inherit C++'s shortcomings such as pointers, operator overloading, and multiple inheritance. Java is object-oriented, simple, distributed, multi-threaded, and dynamic. The HTML language refers to the hypertext markup language, which allowsweb developers to create complex web pages that combine text and images based on different design needs. These pages can be viewed by anyone as long as they are posted online, and they can be read by any type of computer or browser.

From the aspect of system optimization, high concurrency, security, and fault tolerance are the most important contents faced by current systems including P2P network applications. Many websites have taken corresponding measures against this phenomenon. Large websites can alleviate the pressure of user concurrency by increasing the number of intermediate servers or increasing server hardware metrics. Building a high-performance web system can improve application pressure from HTTP service performance, database performance, caching, and more. So after considering the actual factors, this dissertation chooses Apache[104] as the HTTP server. Because of its power and reliability, it is widely used in various application scenarios.

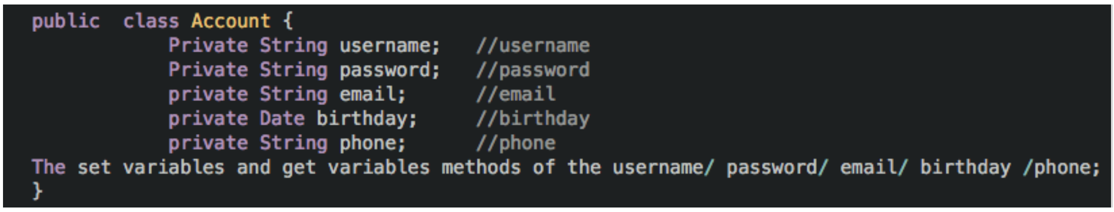
Because Apache supports blocking 1/0, it is not efficient in high concurrency situations. Every connection to Apache will generate a new thread. When the thread reaches the upper limit of the connection pool, Apache can respond to customers in a timely manner. Apache's performance bottlenecks can be mitigated by adding servers, load balancing, and static server separation. Static server separation is to put some basically unchanged code on many independent servers, such as CSS/JS/pictures. This approach reduces the pressure on servers that provide dynamic page access and ensures that the system does not crash due to image issues.

5.2. User Registration and Login Module Implementation

5.2.1. User Registration Module Implementation

The most important class involved in the user registration module is RegisterServlet, which is mainly used to process account data from the front-end registration page. Any data in the front-end page that meets the basic requirements will be submitted to the RegisterServlet class for processing. This class encapsulates user data and submits them to a database table for data updates. The updated result will be returned to the RegisterServlet class, and the class will output the corresponding result to the front page display.

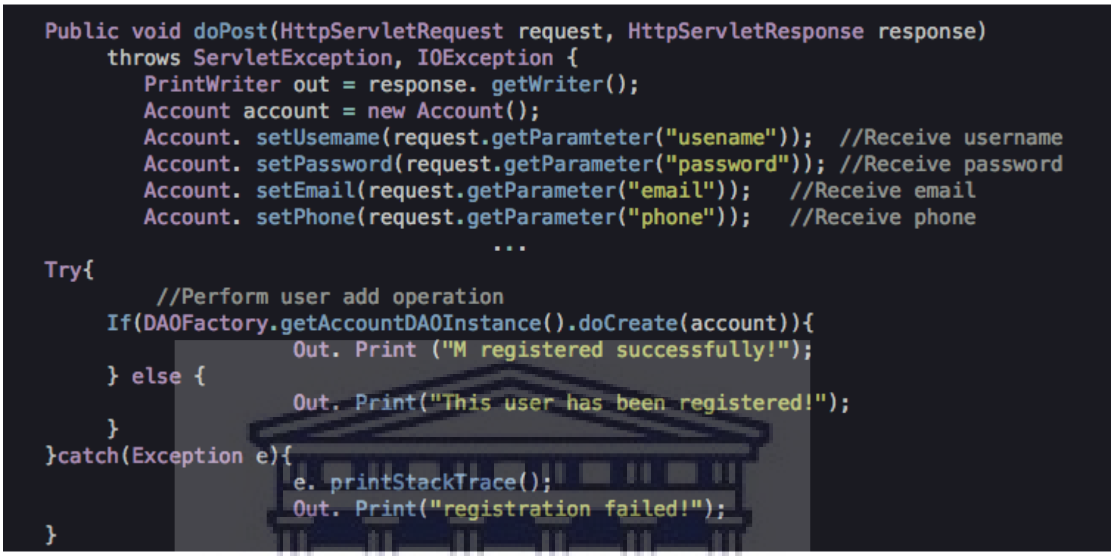
The Account class is a class that encapsulates the user's basic data. It defines the user information variables to receive the username, password, email address, birthday and phone number by the user. The main part of the Account class code is shown in Figure 5-1.



```
public  class Account {
          Private String username;   //username
          Private String password;   //password
          private String email;      //email
          private Date birthday;     //birthday
          private String phone;      //phone
The set variables and get variables methods of the username/ password/ email/ birthday /phone;
}
```

Figure 5-1: Account Class Snippet

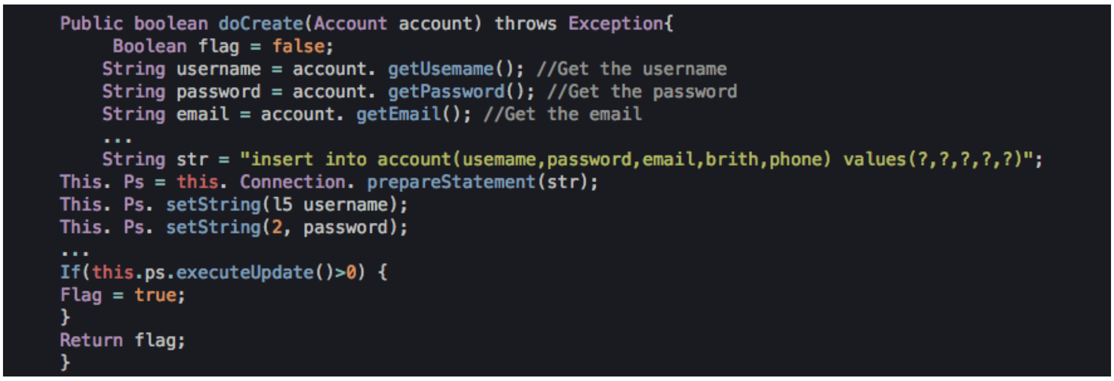
The RegisterServlet class needs to use the Account class to encapsulate the user data sent from the front page. Then, the RegisterServlet class submits the user data to the database table for updating. The main part of the RegisterServlet class is shown in Figure 5-2.



```
Public void doPost(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException {
        PrintWriter out = response. getWriter();
        Account account = new Account();
        Account. setUsemame(request.getParamteter("usename"));  //Receive username
        Account. setPassword(request.getParameter("password")); //Receive password
        Account. setEmail(request.getParameter("email"));   //Receive email
        Account. setPhone(request.getParameter("phone"));   //Receive phone
                                    ...
Try{
         //Perform user add operation
     If(DAOFactory.getAccountDAOInstance().doCreate(account)){
                    Out. Print ("M registered successfully!");
     } else {
                    Out. Print("This user has been registered!");
     }
}catch(Exception e){
                    e. printStackTrace();
                    Out. Print("registration failed!");
}
```

Figure 5-2: DoPost Code Snippet

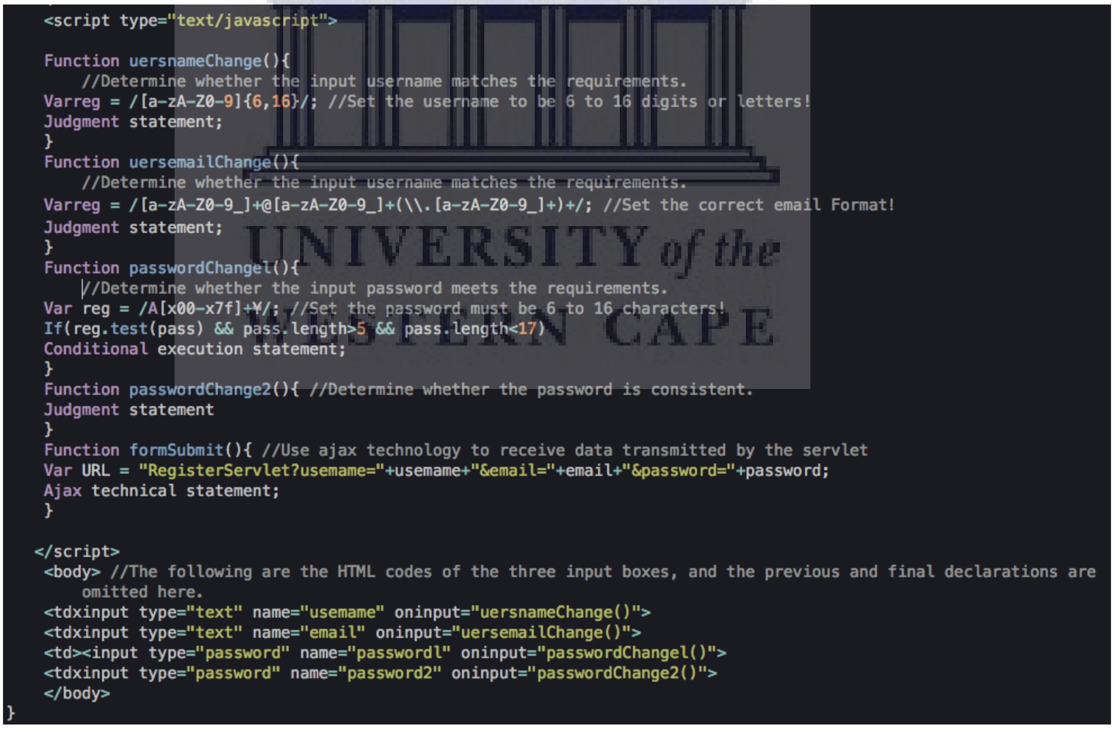
During the execution of the user data addition operation, the system used the factory class DAOFactory. The getAccountDAOInstance() method in this class is used to generate the AccountDAOProxy class object and initialize the connection database through the constructor. The AccountDAOProxy class inherits from the AccountDAOImpl class, and the key code of doCreate(account) is shown in Figure 5-3.



```
    Public boolean doCreate(Account account) throws Exception{
         Boolean flag = false;
        String username = account. getUsemame(); //Get the username
        String password = account. getPassword(); //Get the password
        String email = account. getEmail(); //Get the email
        ...
        String str = "insert into account(usemame,password,email,brith,phone) values(?,?,?,?,?)";
    This. Ps = this. Connection. prepareStatement(str);
    This. Ps. setString(l5 username);
    This. Ps. setString(2, password);
    ...
    If(this.ps.executeUpdate()>0) {
    Flag = true;
    }
    Return flag;
    }
```

Figure 5-3: The CreateAccount Code Snippet

We used HTML and JavaScript when writing front-end user registration pages (rgister.jsp). The jsp file sets a limit on the username, email address and password entered by the user in order to facilitate the management of user data. The key part of the register.jsp code is shown in Figure 5-4.

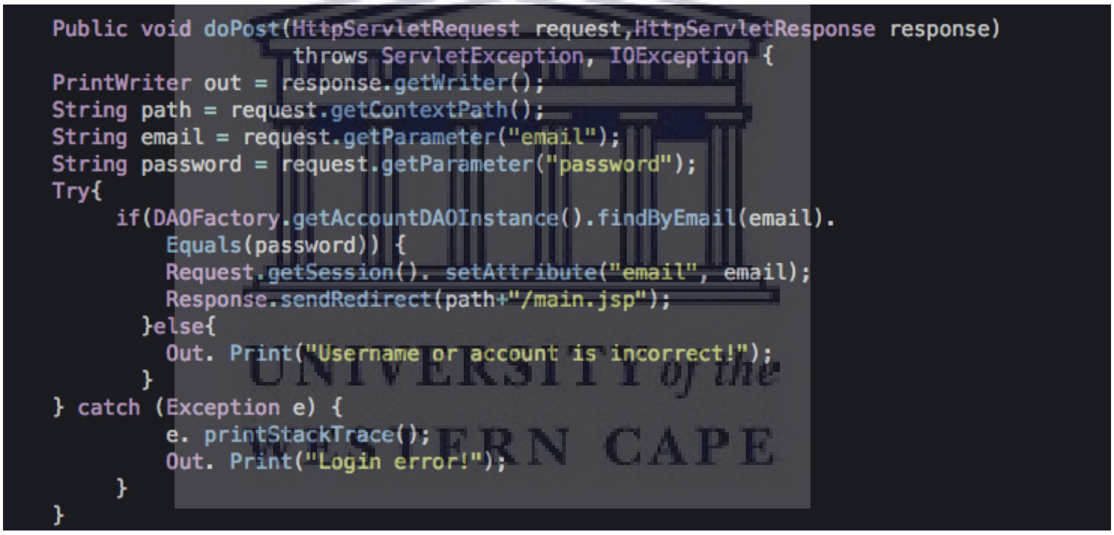


```
   <script type="text/javascript">

   Function uersnameChange(){
       //Determine whether the input username matches the requirements.
   Varreg = /[a-zA-Z0-9]{6,16}/; //Set the username to be 6 to 16 digits or letters!
   Judgment statement;
   }
   Function uersemailChange(){
       //Determine whether the input username matches the requirements.
   Varreg = /[a-zA-Z0-9_]+@[a-zA-Z0-9_]+(\\.[a-zA-Z0-9_]+)+/; //Set the correct email Format!
   Judgment statement;
   }
   Function passwordChangel(){
       //Determine whether the input password meets the requirements.
   Var reg = /A[x00-x7f]+¥/; //Set the password must be 6 to 16 characters!
   If(reg.test(pass) && pass.length>5 && pass.length<17)
   Conditional execution statement;
   }
   Function passwordChange2(){ //Determine whether the password is consistent.
   Judgment statement
   }
   Function formSubmit(){ //Use ajax technology to receive data transmitted by the servlet
   Var URL = "RegisterServlet?usemame="+usemame+"&email="+email+"&password="+password;
   Ajax technical statement;
   }

  </script>
   <body> //The following are the HTML codes of the three input boxes, and the previous and final declarations are
       omitted here.
   <tdxinput type="text" name="usemame" oninput="uersnameChange()">
   <tdxinput type="text" name="email" oninput="uersemailChange()">
   <td><input type="password" name="passwordl" oninput="passwordChangel()">
   <tdxinput type="password" name="password2" oninput="passwordChange2()">
   </body>
}
```

Figure 5-4: Registration Code Snippet

5.2.2. User Login Module Implementation

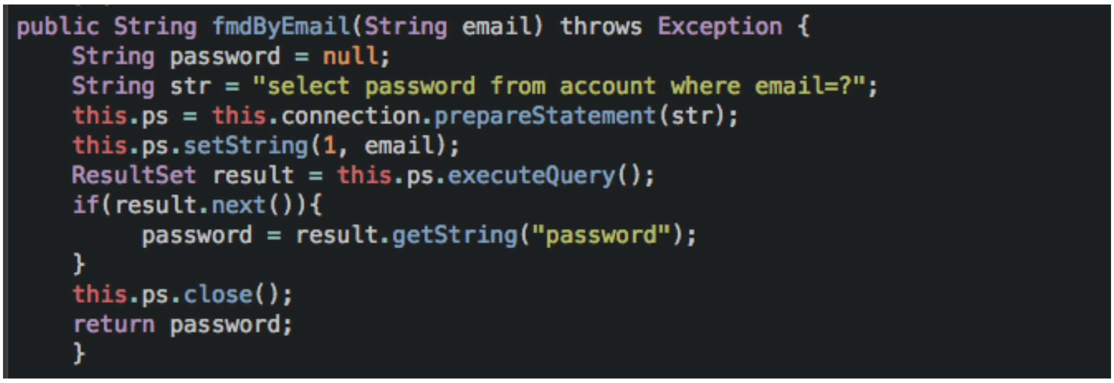
The most important class involved in implementing the user login module is the LoginServlet class. The principle of this class is the same as that of RegisterServlet. The LoginServlet class is mainly used to process account data from the front-end login page. As long as the data provided by the front-end login page meets the requirements, the data will be submitted to the LoginServlet class for processing. This class also uses the Account class to encapsulate user data and submit the encapsulated data to a database table for query operations. The query results are then returned to the LoginServlet class by the database. Finally, the class executes the output statement or performs a page jump based on the returned result. The main part of the LoginServlet class code is shown in Figure 5-5.



```
Public void doPost(HttpServletRequest request,HttpServletResponse response)
                   throws ServletException, IOException {
PrintWriter out = response.getWriter();
String path = request.getContextPath();
String email = request.getParameter("email");
String password = request.getParameter("password");
Try{
     if(DAOFactory.getAccountDAOInstance().findByEmail(email).
         Equals(password)) {
         Request.getSession(). setAttribute("email", email);
         Response.sendRedirect(path+"/main.jsp");
       }else{
         Out. Print("Username or account is incorrect!");
       }
} catch (Exception e) {
         e. printStackTrace();
         Out. Print("Login error!");
     }
}
```

Figure 5-5: The LoginServlet Class Code Snippet

Similarly, the getAccountDAOInstance() method in the factory class DAOFactory is used to generate the AccountDAOProxy class object and initialize the connection database through the constructor. The AccountDAOProxy class inherits from the AccountDAOImpl class, and the key code for the findByEmail(email) method is shown in Figure 5-6.



```
public String fmdByEmail(String email) throws Exception {
    String password = null;
    String str = "select password from account where email=?";
    this.ps = this.connection.prepareStatement(str);
    this.ps.setString(1, email);
    ResultSet result = this.ps.executeQuery();
    if(result.next()){
        password = result.getString("password");
    }
    this.ps.close();
    return password;
    }
```

Figure 5-6: The findByEmail Method Code Snippet

We used HTML and JavaScript when writing front-end user login pages (login.jsp). In the jsp file, the same restrictions are placed on the user's email and password input in order to remind the user to enter the correct email address and password. The key part of the login.jsp code is similar with the code of register.jsp in Figure 5-4.

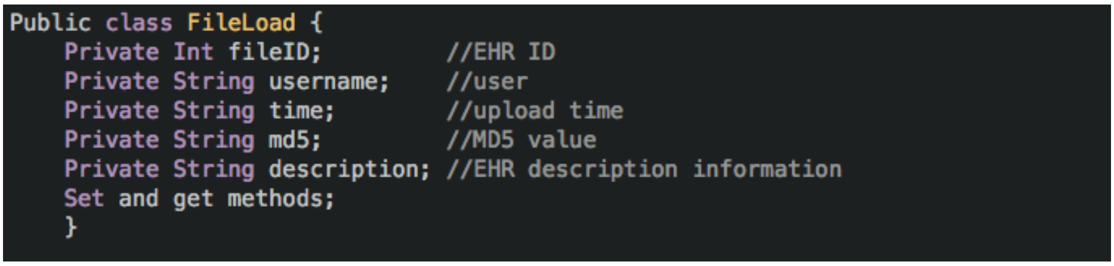
5.3. Data Authorization Module Implementation

5.3.1. Data Authorization Module Implementation

The Java classes involved in the implementation of the data authorization module are mainly the authoriFileServlet class and the MulticastController class. The authoriFileServlet class is mainly responsible for processing the data passed by the main page of the system front end, and obtaining the user name of the current authorization file obtained from the system session. The parameters passed include the EHR file data selected by the user and the description information of the file. The system wraps these file informations through the FileLoad class.

The MulticastController class is mainly responsible for encrypting and passing out the encapsulated FileLoad object. The result of this step will be passed to the next module for processing.

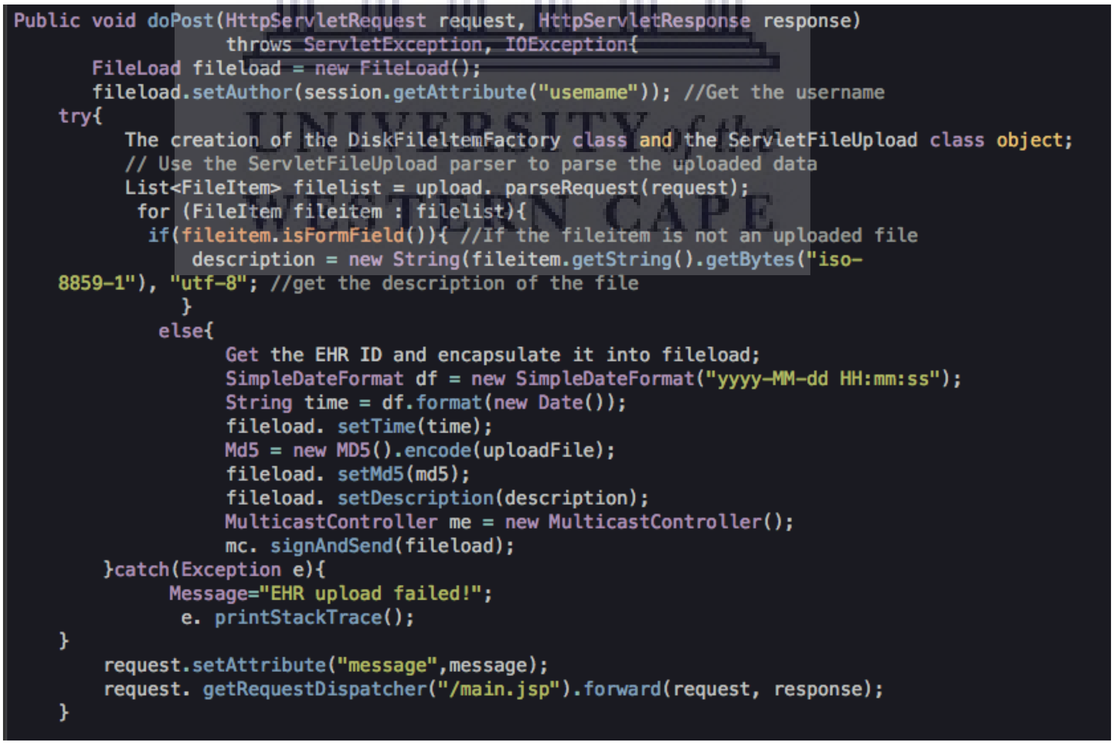
The FileLoad class is a class that encapsulates authorized EHR data. The object of this class encapsulates the five parts of EHR content, user name, upload time, MD5 value and description information. The key code of the FileLoad class is shown in Figure 5-7.



```
Public class FileLoad {
    Private Int fileID;          //EHR ID
    Private String username;     //user
    Private String time;         //upload time
    Private String md5;          //MD5 value
    Private String description; //EHR description information
    Set and get methods;
    }
```

Figure 5-7: The FileLoad Class Code Snippet

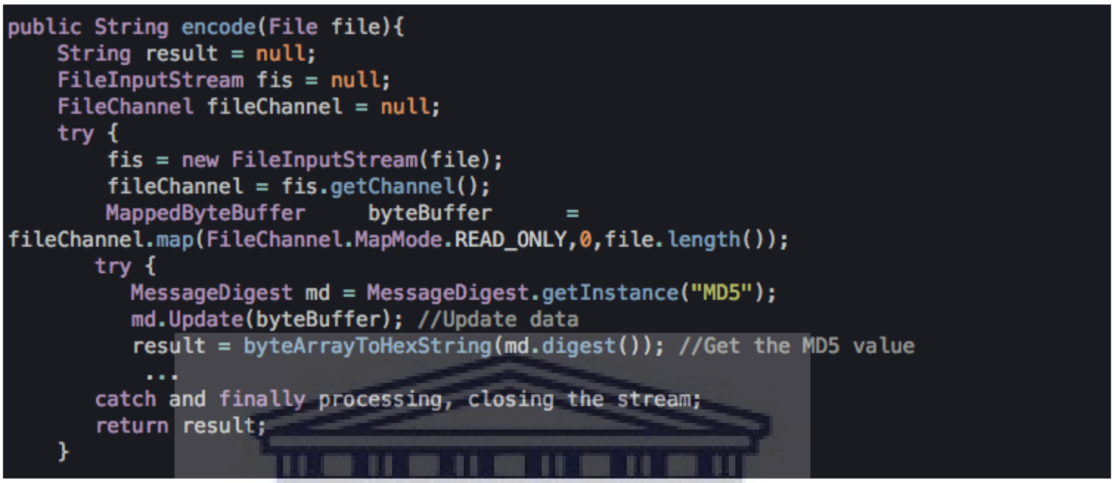
The authoriFileServlet class encapsulates the EHR data selected by the front-end main page user into the FileLoad object. In this class, the upload time is accurate to the second and the format is "yyyy-MM-dd HH:mm:ss". The key part of the authoriFileServlet class code is shown in Figure 5-8.



```
Public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException{
        FileLoad fileload = new FileLoad();
        fileload.setAuthor(session.getAttribute("usemame")); //Get the username
    try{
            The creation of the DiskFileltemFactory class and the ServletFileUpload class object;
            // Use the ServletFileUpload parser to parse the uploaded data
            List<FileItem> filelist = upload. parseRequest(request);
             for (FileItem fileitem : filelist){
              if(fileitem.isFormField()){ //If the fileitem is not an uploaded file
                 description = new String(fileitem.getString().getBytes("iso-
    8859-1"), "utf-8"; //get the description of the file
                 }
               else{
                    Get the EHR ID and encapsulate it into fileload;
                    SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                    String time = df.format(new Date());
                    fileload. setTime(time);
                    Md5 = new MD5().encode(uploadFile);
                    fileload. setMd5(md5);
                    fileload. setDescription(description);
                    MulticastController mc = new MulticastController();
                    mc. signAndSend(fileload);
       }catch(Exception e){
              Message="EHR upload failed!";
               e. printStackTrace();
    }
       request.setAttribute("message",message);
       request. getRequestDispatcher("/main.jsp").forward(request, response);
    }
```

Figure 5-8: The authoriFileServlet Class Code Snippet

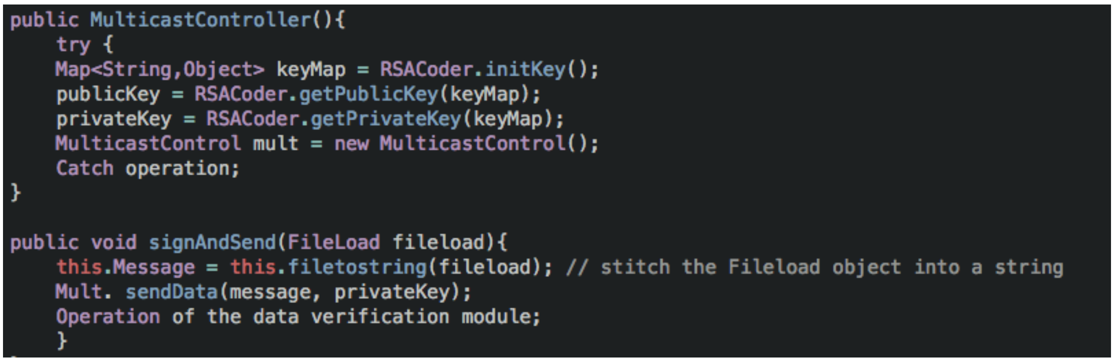
According to Figure 5-8, the system needs to hash the file after obtaining the file. The specific method is that the system uses the MD5 algorithm to obtain the MD5 value of the file, and the implementation of the MD5 algorithm is shown in Figure 5-9.



```
public String encode(File file){
    String result = null;
    FileInputStream fis = null;
    FileChannel fileChannel = null;
    try {
        fis = new FileInputStream(file);
        fileChannel = fis.getChannel();
        MappedByteBuffer     byteBuffer      =
fileChannel.map(FileChannel.MapMode.READ_ONLY,0,file.length());
       try {
         MessageDigest md = MessageDigest.getInstance("MD5");
         md.Update(byteBuffer); //Update data
         result = byteArrayToHexString(md.digest()); //Get the MD5 value
          ...
       catch and finally processing, closing the stream;
       return result;
    }
```

Figure 5-9: The Implementation of the MD5 Algorithm Code Snippet

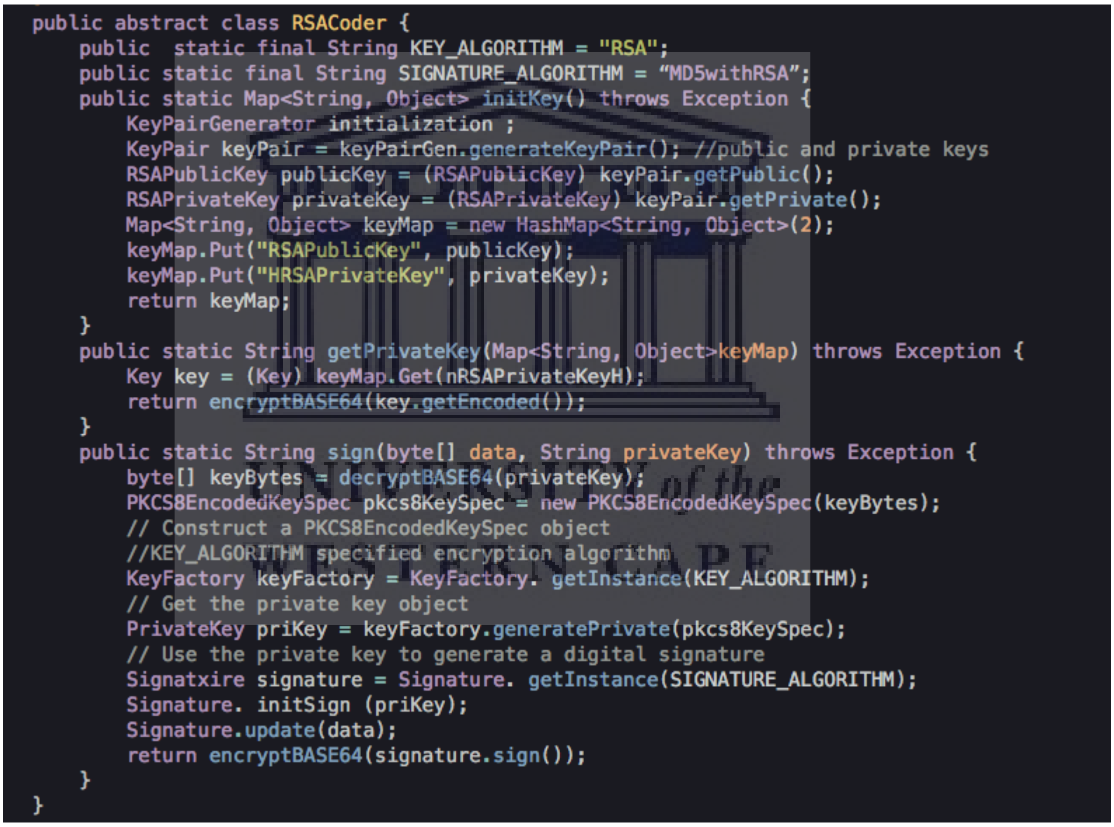
Firstly, the authoriFileServlet class encapsulates the EHR data into a FileLoad object. The class then passes the encapsulated data to the MulticastController class for processing and transmission to the multicast network for data validation. The code for the key part of the MulticastController class is shown in Figure 5-10.



```
public MulticastController(){
    try {
    Map<String,Object> keyMap = RSACoder.initKey();
    publicKey = RSACoder.getPublicKey(keyMap);
    privateKey = RSACoder.getPrivateKey(keyMap);
    MulticastControl mult = new MulticastControl();
    Catch operation;
}

public void signAndSend(FileLoad fileload){
    this.Message = this.filetostring(fileload); // stitch the Fileload object into a string
    Mult. sendData(message, privateKey);
    Operation of the data verification module;
    }
```

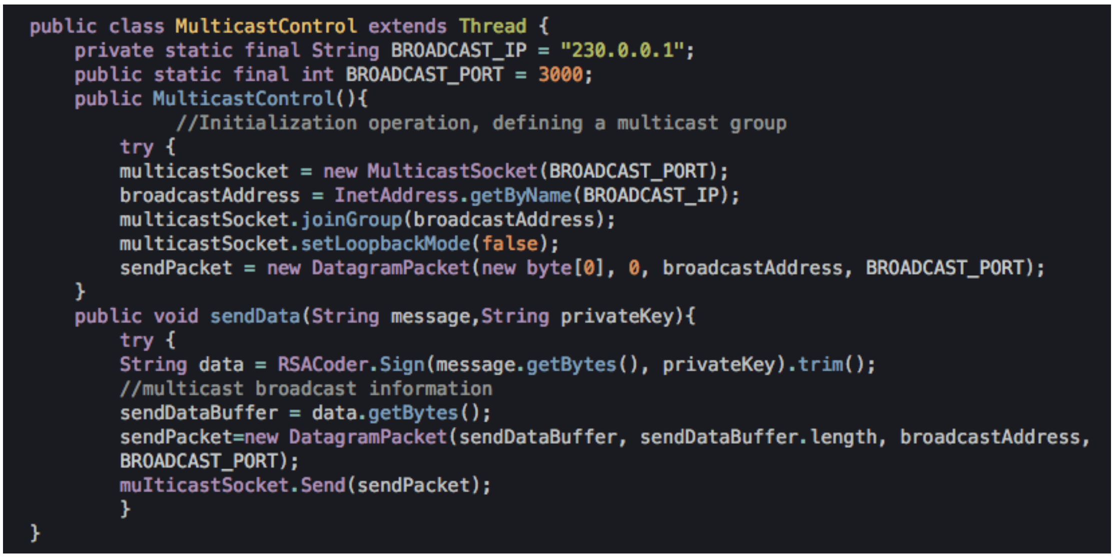
Figure 5-10: The MulticastController Class Code Snippet

As can be seen from Figure 5-10, when the MulticastControl class is initialized, it needs to obtain the public and private keys through the getPublicKey(keyMap) method and the getPrivateKey(keyMap) method in the RSACoder class. The acquired private key can digitally sign the EHR file information and only the corresponding public key can correctly verify the signature. Objects of the MulticastControl class are responsible for transmitting related information such as digital signatures through the multicast module. The key codes for obtaining the private key and digital signature in the RSACoder class are shown in Figure 5-11. The key code in the MulticastControl class is shown in Figure 5-12.



```
public abstract class RSACoder {
    public  static final String KEY_ALGORITHM = "RSA";
    public static final String SIGNATURE_ALGORITHM = “MD5withRSA”;
    public static Map<String, Object> initKey() throws Exception {
        KeyPairGenerator initialization ;
        KeyPair keyPair = keyPairGen.generateKeyPair(); //public and private keys
        RSAPublicKey publicKey = (RSAPublicKey) keyPair.getPublic();
        RSAPrivateKey privateKey = (RSAPrivateKey) keyPair.getPrivate();
        Map<String, Object> keyMap = new HashMap<String, Object>(2);
        keyMap.Put("RSAPublicKey", publicKey);
        keyMap.Put("HRSAPrivateKey", privateKey);
        return keyMap;
    }
    public static String getPrivateKey(Map<String, Object>keyMap) throws Exception {
        Key key = (Key) keyMap.Get(nRSAPrivateKeyH);
        return encryptBASE64(key.getEncoded());
    }
    public static String sign(byte[] data, String privateKey) throws Exception {
        byte[] keyBytes = decryptBASE64(privateKey);
        PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec(keyBytes);
        // Construct a PKCS8EncodedKeySpec object
        //KEY_ALGORITHM specified encryption algorithm
        KeyFactory keyFactory = KeyFactory. getInstance(KEY_ALGORITHM);
        // Get the private key object
        PrivateKey priKey = keyFactory.generatePrivate(pkcs8KeySpec);
        // Use the private key to generate a digital signature
        Signatxire signature = Signature. getInstance(SIGNATURE_ALGORITHM);
        Signature. initSign (priKey);
        Signature.update(data);
        return encryptBASE64(signature.sign());
    }
}
```

Figure 5-11: The RSACoder Class Code Snippet

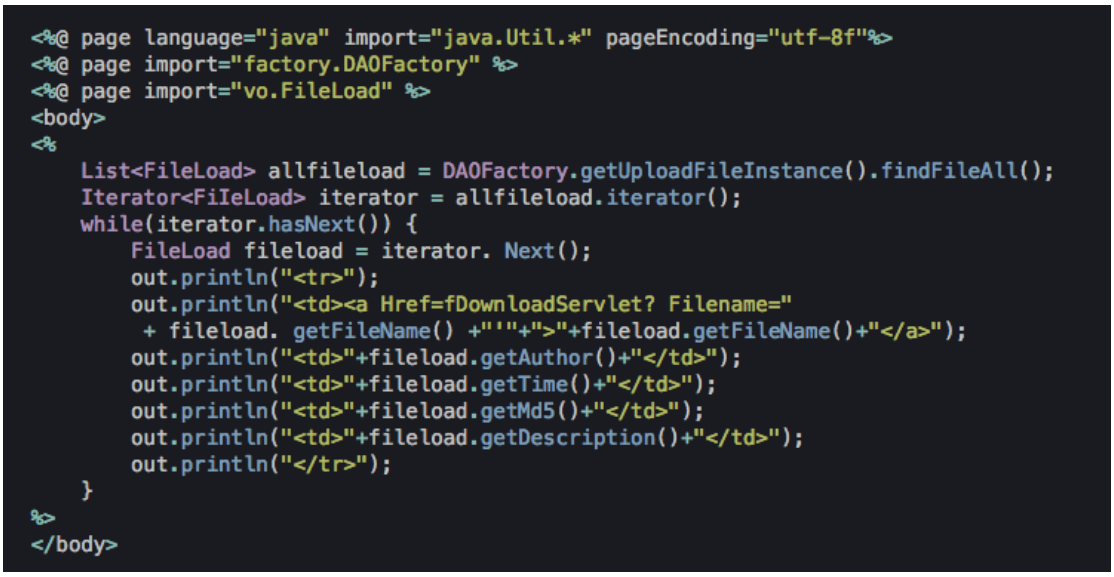


```
public class MulticastControl extends Thread {
    private static final String BROADCAST_IP = "230.0.0.1";
    public static final int BROADCAST_PORT = 3000;
    public MulticastControl(){
             //Initialization operation, defining a multicast group
        try {
        multicastSocket = new MulticastSocket(BROADCAST_PORT);
        broadcastAddress = InetAddress.getByName(BROADCAST_IP);
        multicastSocket.joinGroup(broadcastAddress);
        multicastSocket.setLoopbackMode(false);
        sendPacket = new DatagramPacket(new byte[0], 0, broadcastAddress, BROADCAST_PORT);
    }
    public void sendData(String message,String privateKey){
        try {
        String data = RSACoder.Sign(message.getBytes(), privateKey).trim();
        //multicast broadcast information
        sendDataBuffer = data.getBytes();
        sendPacket=new DatagramPacket(sendDataBuffer, sendDataBuffer.length, broadcastAddress,
        BROADCAST_PORT);
        muIticastSocket.Send(sendPacket);
        }
}
```

Figure 5-12: The MulticastControl Class Code Snippet

5.3.2. Conditional Query Module Implementation

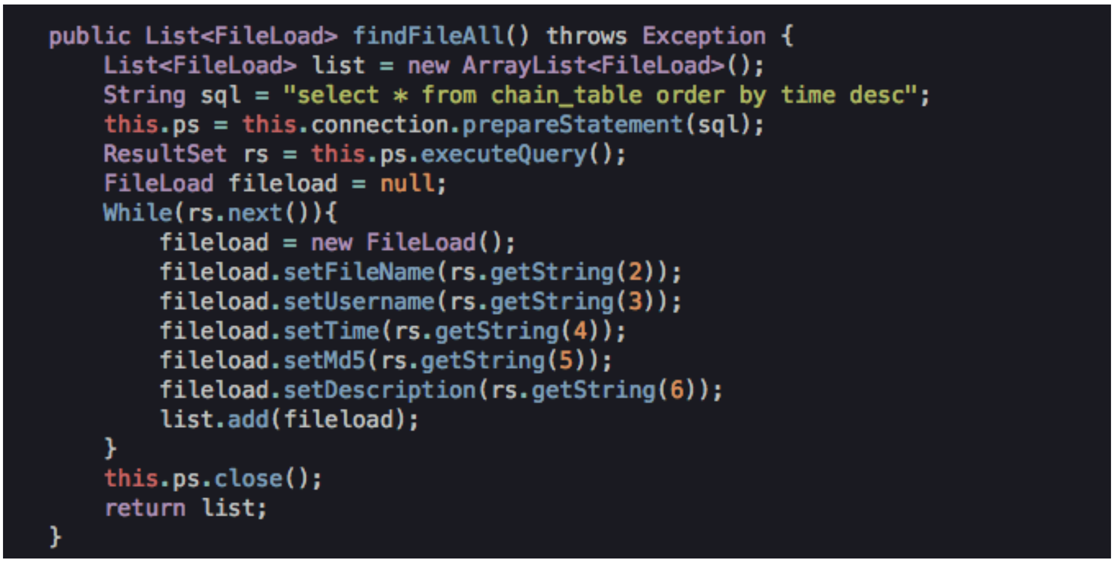
The Conditional query module mainly implements the query operation of the user to the EHRs. In the front page main.jsp, the system sets the conditional query butten at the botten of the form. Users can use the pop-up text box to find the content they are interested in. Afterward system displayed in detail in the front end page. So in the front-end page researcher need to write the code interpreted by the server side, that is, through the Java code to achieve data query and display. The key code for dynamically displaying the most recently uploaded EHRs in main.jsp is shown in Figure 5-13.



```
<%@ page language="java" import="java.Util.*" pageEncoding="utf-8f"%>
<%@ page import="factory.DAOFactory" %>
<%@ page import="vo.FileLoad" %>
<body>
<%
    List<FileLoad> allfileload = DAOFactory.getUploadFileInstance().findFileAll();
    Iterator<FileLoad> iterator = allfileload.iterator();
    while(iterator.hasNext()) {
        FileLoad fileload = iterator. Next();
        out.println("<tr>");
        out.println("<td><a Href=fDownloadServlet? Filename="
         + fileload. getFileName() +"'"+">"+fileload.getFileName()+"</a>");
        out.println("<td>"+fileload.getAuthor()+"</td>");
        out.println("<td>"+fileload.getTime()+"</td>");
        out.println("<td>"+fileload.getMd5()+"</td>");
        out.println("<td>"+fileload.getDescription()+"</td>");
        out.println("</tr>");
    }
%>
</body>
```

Figure 5-13: The Dynamically Displaying Code Snippet

As seen in Figure 5-13, the <%%> contains the Java statement DAOFactory.getAuthoriFileInstance().fmdFileAll(). This method is to implement the search function of EHRs data. DAOFactory. getAuthoriFileInstance() generates the AuthoriFileDAOProxy class object through the factory class DAOFactory and initializes the connection database. In the above process, the findFileAlia method is to implement data lookup. The parent class of AuthoriFileDAOProxy is AuthoriFileDAOImpl. The key part of the fmdFileAll() method in AuthoriFileDAOImpl is shown in Figure 5-14.



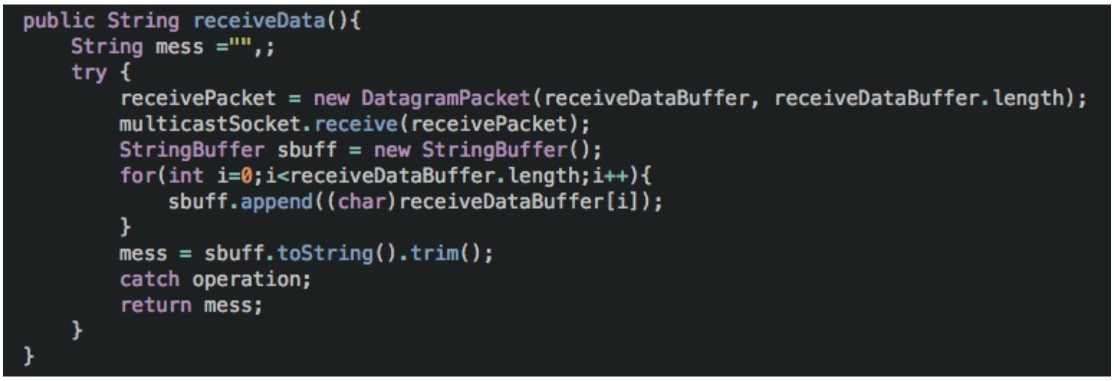
```
public List<FileLoad> findFileAll() throws Exception {
    List<FileLoad> list = new ArrayList<FileLoad>();
    String sql = "select * from chain_table order by time desc";
    this.ps = this.connection.prepareStatement(sql);
    ResultSet rs = this.ps.executeQuery();
    FileLoad fileload = null;
    While(rs.next()){
        fileload = new FileLoad();
        fileload.setFileName(rs.getString(2));
        fileload.setUsername(rs.getString(3));
        fileload.setTime(rs.getString(4));
        fileload.setMd5(rs.getString(5));
        fileload.setDescription(rs.getString(6));
        list.add(fileload);
    }
    this.ps.close();
    return list;
}
```

Figure 5-14: The fmdFileAll Method Code Snippet

5.4. Broadcast Upload Module Implementation

The EHRs data authorization module is only responsible for processing and encapsulating the EHRs files selected by the user and sending these EHRs to the multicast network. Prior to this, the CB-EHRs system will perform a consensus operation to select the representative node. Since this part of the content has been specifically stated, it is not introduced here. To truly share the system, other representative nodes in the multicast network are required to verify the data. Therefore, this part of the function is implemented by the broadcast upload module. The main Java classes involved in this module are the MulticastController class and the MulticastControl class. The MulticastControl class is mainly responsible for receiving data sent by nodes in the network. The received data is then checked by the MulticastController class for the EHRs data.

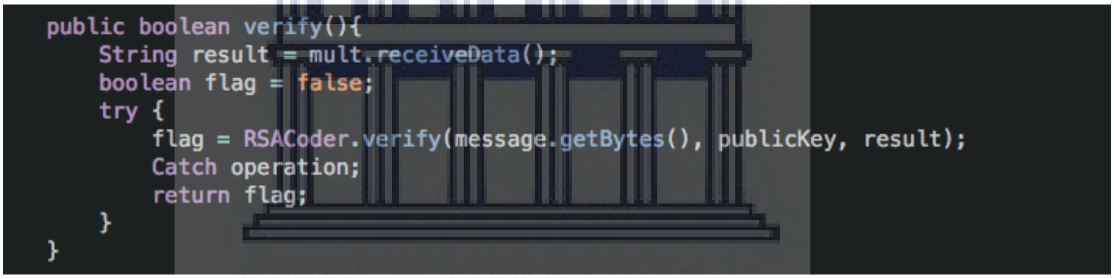
The pseudo code for receiving EHRs information in the MulticastControl class is shown in Figure 5-15.



```java
public String receiveData(){
    String mess ="",;
    try {
        receivePacket = new DatagramPacket(receiveDataBuffer, receiveDataBuffer.length);
        multicastSocket.receive(receivePacket);
        StringBuffer sbuff = new StringBuffer();
        for(int i=0;i<receiveDataBuffer.length;i++){
            sbuff.append((char)receiveDataBuffer[i]);
        }
        mess = sbuff.toString().trim();
        catch operation;
        return mess;
    }
}
```

Figure 5-15: The Receiving EHRs Information Code Snippet

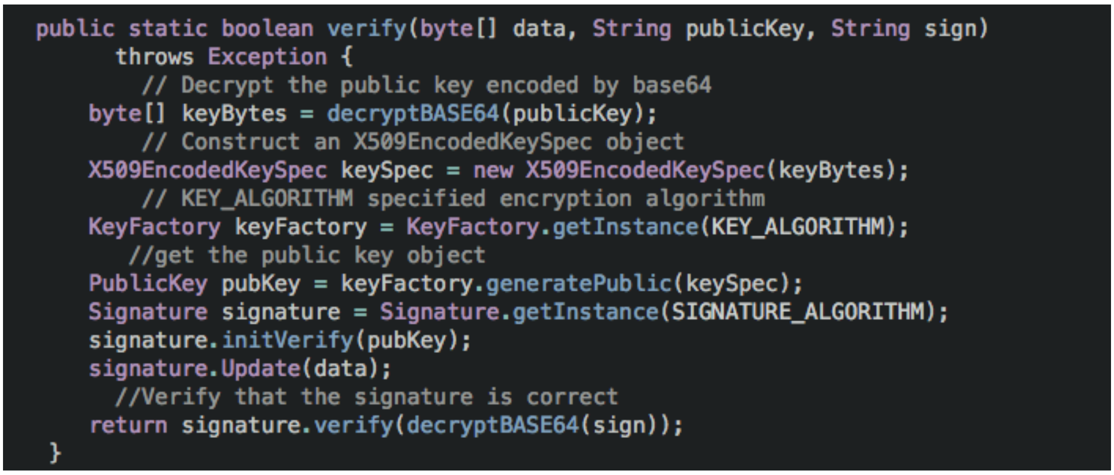
The system returns the received data to the MulticastController class through the MulticastControl class for verification processing. The data validation method (verify()) in the MulticastController class is shown in Figure 5-16.



```java
public boolean verify(){
    String result = mult.receiveData();
    boolean flag = false;
    try {
        flag = RSACoder.verify(message.getBytes(), publicKey, result);
        Catch operation;
        return flag;
    }
}
```

Figure 5-16: The Data Validation I Code Snippet

According to Figure 5-15, the system calls the verify () method of the RSACoder class during data validation. The verification principle of the method is that only when the original data before and after the signature is not changed and the public key corresponding to the signature private key is used for verification, the correct verification result can be obtained. The code for the verify () method of the RSACoder class is shown in Figure 5-17.



```
public static boolean verify(byte[] data, String publicKey, String sign)
      throws Exception {
        // Decrypt the public key encoded by base64
    byte[] keyBytes = decryptBASE64(publicKey);
        // Construct an X509EncodedKeySpec object
    X509EncodedKeySpec keySpec = new X509EncodedKeySpec(keyBytes);
        // KEY_ALGORITHM specified encryption algorithm
    KeyFactory keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);
       //get the public key object
    PublicKey pubKey = keyFactory.generatePublic(keySpec);
    Signature signature = Signature.getInstance(SIGNATURE_ALGORITHM);
    signature.initVerify(pubKey);
    signature.Update(data);
      //Verify that the signature is correct
    return signature.verify(decryptBASE64(sign));
 }
```

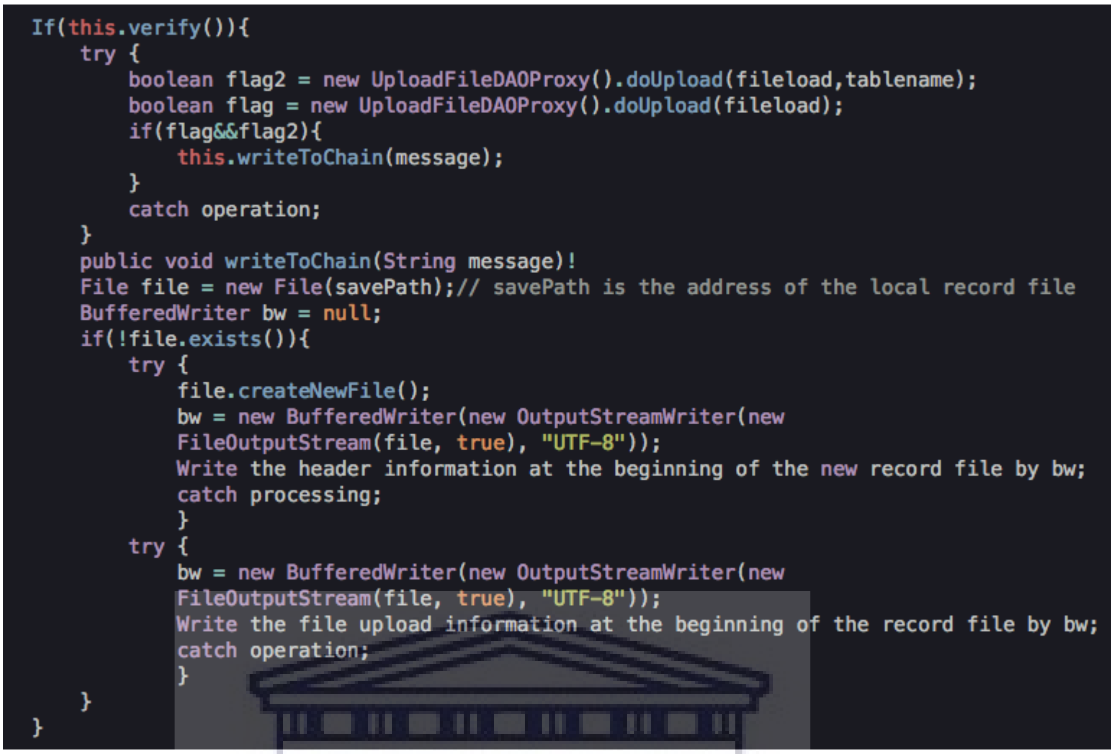
Figure 5-17: The Data Validation II Code Snippet

The MulticastController class performs subsequent processing based on the verification result of the digital signature. When the signature verification is correct, it adds the basic information of the passed EHRs to the database table, and calls the code of the inspection Blockchain part to update the record file.

5.5. The Implementation of Detecting EHRs Blockchain

5.5.1. Log File Update Implementation

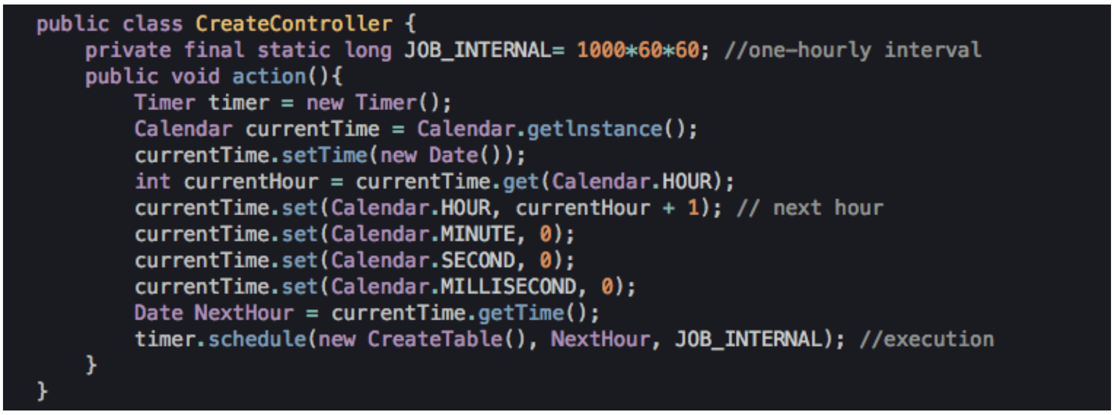
In the broadcast upload module, the EHRs information data verified by the nodes in the multicast network enters the Blockchain verification section for the next operation. According to the content format of the log file given in Chapter 4, the Blockchain verification part records the broadcasted EHRs information in the latest log file and also stores them in the database table. The most important Java class that the system depends on during the implementation of this function is the MulticastController class. The writeToChain() method in the MulticastController class implements the update operation of the log file. Its key part of the code is shown in Figure 5-18.



```
If(this.verify()){
    try {
        boolean flag2 = new UploadFileDAOProxy().doUpload(fileload,tablename);
        boolean flag = new UploadFileDAOProxy().doUpload(fileload);
        if(flag&&flag2){
            this.writeToChain(message);
        }
        catch operation;
    }
    public void writeToChain(String message)!
    File file = new File(savePath);// savePath is the address of the local record file
    BufferedWriter bw = null;
    if(!file.exists()){
        try {
            file.createNewFile();
            bw = new BufferedWriter(new OutputStreamWriter(new
            FileOutputStream(file, true), "UTF-8"));
            Write the header information at the beginning of the new record file by bw;
            catch processing;
            }
        try {
            bw = new BufferedWriter(new OutputStreamWriter(new
            FileOutputStream(file, true), "UTF-8"));
            Write the file upload information at the beginning of the record file by bw;
            catch operation;
            }
    }
}
```

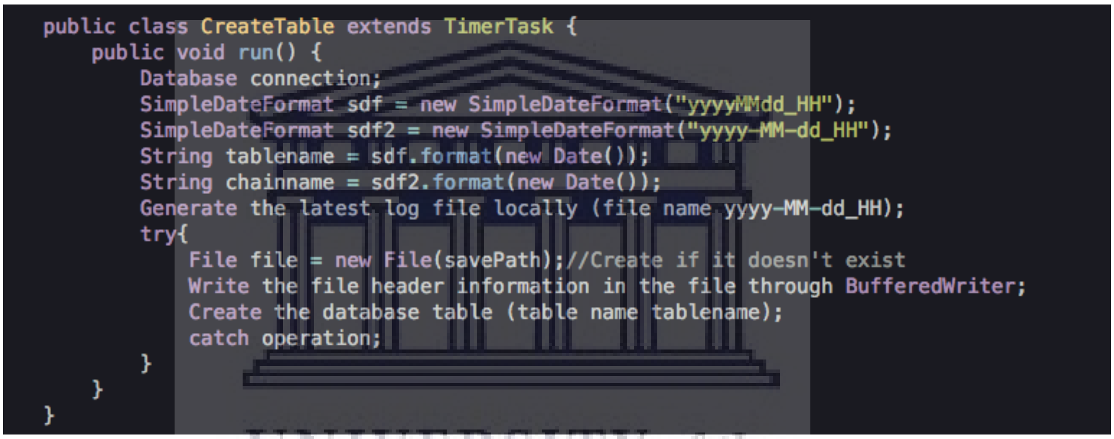
Figure 5-18: The Update Operation of the Log File Code Snippet

According to the system design in Chapter 4, the log file generates a new log file at intervals of one hour. The new log file is created starting at the integral poin of the system time. All uploaded EHRs in the system will be recorded in the new log file within the next hour after the log file is created, until the next new log file is generated. The main Java classes that depend on the implementation of this part are the CreateController class and the CreateTable class. The CreateTable class is used to implement specific operations, that is, to generate new log files and database tables. The naming format of the new record file is "yyyy-MM-dd_HH", and the naming format of the database table is "yyyyMMdd_HH". The function of the CreateController class is equivalent to the time timer. It will perform the operations in the CreateTable class at each hour. The key code of the scheduled task of the CreateController class is shown in Figure 5-19. The key code of the specific operation in the CreateTable class is shown in Figure 5-20.



```
public class CreateController {
    private final static long JOB_INTERNAL= 1000*60*60; //one-hourly interval
    public void action(){
        Timer timer = new Timer();
        Calendar currentTime = Calendar.getInstance();
        currentTime.setTime(new Date());
        int currentHour = currentTime.get(Calendar.HOUR);
        currentTime.set(Calendar.HOUR, currentHour + 1); // next hour
        currentTime.set(Calendar.MINUTE, 0);
        currentTime.set(Calendar.SECOND, 0);
        currentTime.set(Calendar.MILLISECOND, 0);
        Date NextHour = currentTime.getTime();
        timer.schedule(new CreateTable(), NextHour, JOB_INTERNAL); //execution
    }
}
```

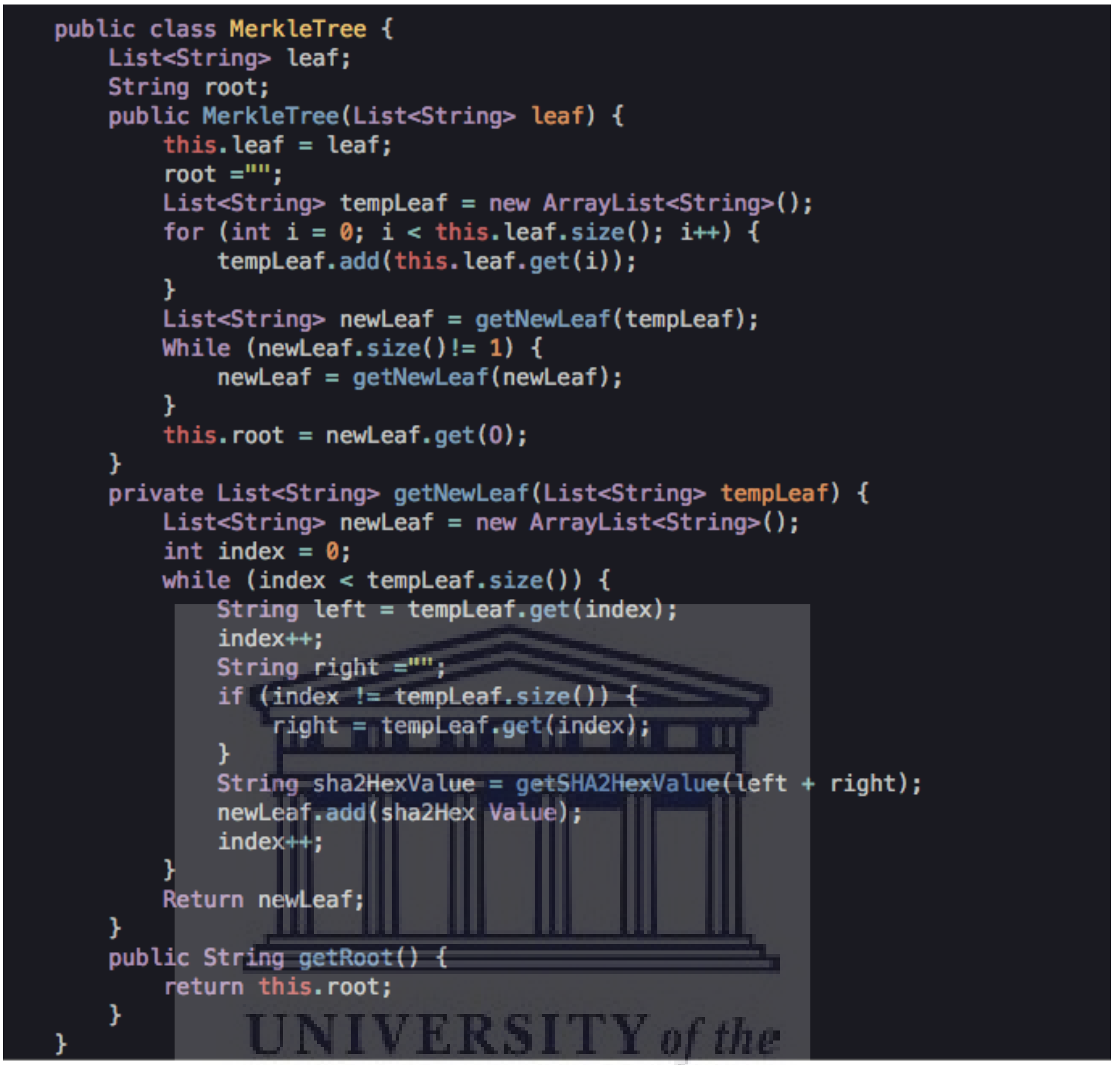
Figure 5-19: The Scheduled Task Code Snippet



```
public class CreateTable extends TimerTask {
    public void run() {
        Database connection;
        SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd_HH");
        SimpleDateFormat sdf2 = new SimpleDateFormat("yyyy-MM-dd_HH");
        String tablename = sdf.format(new Date());
        String chainname = sdf2.format(new Date());
        Generate the latest log file locally (file name yyyy-MM-dd_HH);
        try{
            File file = new File(savePath);//Create if it doesn't exist
            Write the file header information in the file through BufferedWriter;
            Create the database table (table name tablename);
            catch operation;
        }
    }
}
```

Figure 5-20: The Specific Operation Code Snippet

According to Figure 5-20, the header information is written at the beginning of the file whenever a new log file is generated. This information includes the Merkle Tree root value of the previous log file and the current system time. Therefore, the system needs to obtain the MD5 value of all uploaded EHRs in the system within the last hour as the leaf node of MerkleTree. Thereafter, the value of the root node is obtained through the Merkle Tree root node algorithm. The most important class that the system relies on during the implementation of this function is the MerkleTree class. The key part of the MerkleTree class is shown in Figure 5-21.



```
public class MerkleTree {
    List<String> leaf;
    String root;
    public MerkleTree(List<String> leaf) {
        this.leaf = leaf;
        root ="";
        List<String> tempLeaf = new ArrayList<String>();
        for (int i = 0; i < this.leaf.size(); i++) {
            tempLeaf.add(this.leaf.get(i));
        }
        List<String> newLeaf = getNewLeaf(tempLeaf);
        While (newLeaf.size()!= 1) {
            newLeaf = getNewLeaf(newLeaf);
        }
        this.root = newLeaf.get(0);
    }
    private List<String> getNewLeaf(List<String> tempLeaf) {
        List<String> newLeaf = new ArrayList<String>();
        int index = 0;
        while (index < tempLeaf.size()) {
            String left = tempLeaf.get(index);
            index++;
            String right ="";
            if (index != tempLeaf.size()) {
                right = tempLeaf.get(index);
            }
            String sha2HexValue = getSHA2HexValue(left + right);
            newLeaf.add(sha2Hex Value);
            index++;
        }
        Return newLeaf;
    }
    public String getRoot() {
        return this.root;
    }
}
```

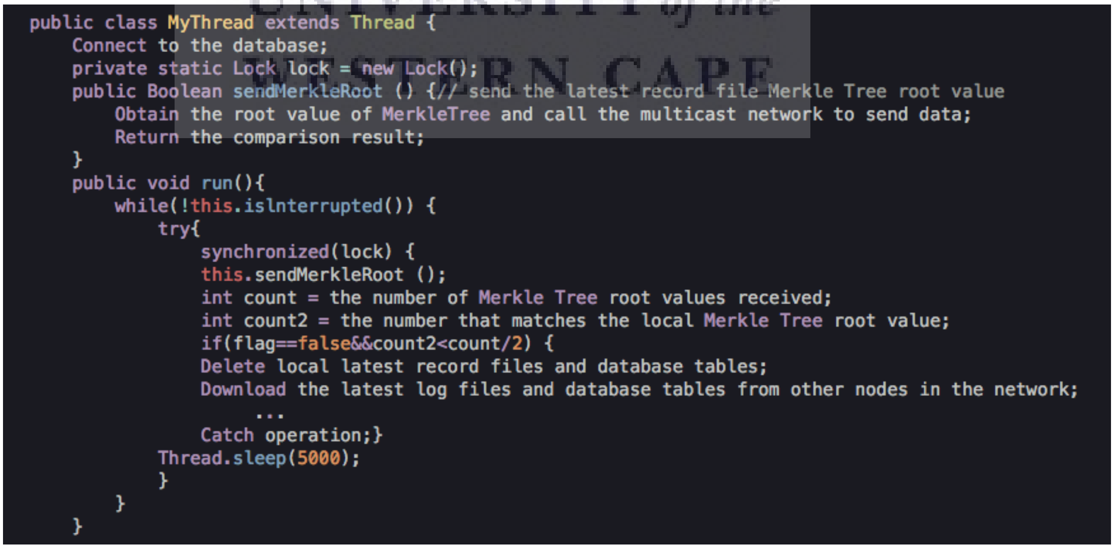
Figure 5-21: The MerkleTree Class Code Snippet

As can be seen from Figure 5-21, the system uses the MerkleTree class's constructor MerkleTree (List<String>leaf) to calculate the root value of the MerkleTree with the leaf as the leaf node. In this process, the system can obtain the root by getRoot(). Before this, the system needs to query the MD5 value of all successful upload EHRs in the last hour to be stored in the collection leaf. The implementation of the query operation is similar to the code shown in 5-14, and will not be described here.

5.5.2. Implementation of EHRs Blockchain Detection Function

According to the corresponding design part of Chapter 4, the most important Java class involved in the EHRs Blockchain detection function is the MyThread class. The MyThread class is a thread class. In the MyThread class, a method of periodically detecting a local log file, sending a log file Merkle Tree root, and processing a received message is set. Its thread is set to communicate with the system every 5 minutes.

The following is the process of data exchange: the node detects whether the MD5 value in the local log file corresponds to the value in the database table. The node then sends the Merkle Tree root of its own local record file and receives the Merkle Tree root from other nodes in the network. Next, the node compares the local value with the received data one by one and calculates the number of matches. If the number of matches exceeds half of the total, the local log file is consistent with the system. Otherwise, the system deletes the latest local log file and requests other nodes in the system to download the latest log file. During the local log file detection process, the system needs to add a lock on the thread to ensure that no other threads interfere with the function. The pseudo-code of the key part of the MyThread class is shown in Figure 5-22.



```
public class MyThread extends Thread {
    Connect to the database;
    private static Lock lock = new Lock();
    public Boolean sendMerkleRoot () {// send the latest record file Merkle Tree root value
        Obtain the root value of MerkleTree and call the multicast network to send data;
        Return the comparison result;
    }
    public void run(){
        while(!this.islnterrupted()) {
            try{
                synchronized(lock) {
                this.sendMerkleRoot ();
                int count = the number of Merkle Tree root values received;
                int count2 = the number that matches the local Merkle Tree root value;
                if(flag==false&&count2<count/2) {
                Delete local latest record files and database tables;
                Download the latest log files and database tables from other nodes in the network;
                    ...
                Catch operation;}
            Thread.sleep(5000);
            }
        }
    }
```

Figure 5-22: The MyThread Class Code Snippet

5.6. Summary of Chapter 5

This chapter gives an introduction to the specific implementation details of each module based on the system requirements in Chapter 3 and the detailed design of the system described in Chapter 4. This chapter also shows the key code for the detailed implementation of each module and further explains the code.

6. Testing of the Credible Blockchain-based E-health Records System

This chapter tests the consensus mechanism optimization scheme proposed in Chapter 4 and the key functions of EHRs based on the Blockchain. First, this chapter confirms the feasibility of the optimization scheme through the correct configuration of the test environment. Then, the performance tests of dBFT and the original PBFT consensus algorithm were performed on the CB-EHRs alliance chain, and the test results were compared. Finally, according to the business logic of the EHRs system, the implementation results and functional test conclusions of the application are presented.

6.1. System Performance Test

6.1.1. Test Environment Construction

This dissertation uses Docker[105] virtualization technology to build a Blockchain network for node access testing. Docker is written in Go language. It is a high-level control tool based on kernel container technology. Unlike previous development tools, Docker is very easy to understand and use. Most developers can get started and deploy the development environment in the short term. This high ease of use makes Docker quickly catch the attention of developers. It also accelerates the update of its technology itself. One of the biggest features of Docker is the small overhead of resources other than the hypervisor. It can share operating system resources with the underlying layer to reduce the load on the system itself and enable more applications to run in the same operating system[106]. In addition, Docker allows developers to package their own deployment programs with the development runtime environment. It can publish packaged data to any popular Linux system. This feature facilitates cross-platform and cross-host usage of application code, and enables application code to be written once and run anywhere.

This technology has presented many advantages such as lightweight resource occupation, portability, and predictability in a short period of time[106]. It is therefore being accepted and used by more and more developers, testers, and application deployers. To facilitate testing, we simulated the CB-EHRs system in Fabric, and put both the PBFT consensus and the dBFT consensus into the simulated system. During the testing of this dissertation, the software environment in which the code runs is Docker with version number 17.09. The test server hardware environment is a 4-core CPU, 8GB RAM, and a 40GB hard drive. The system environment is macOS HighSierra 10.13.6. During the node deployment process, We first prepare the tar package of related mirrors such as peer, memsrvc, and fabric-ccenv. We use the "docker load-ipeer.tar" command to import the package into the local docker mirror library. Then we go to the directory where docker-compose.yaml is located to execute the "docker-compose up -d" command. After the above process, we can start the peer node and the memsrvc container one by one.

This dissertation builds a Blockchain test environment as shown in Figure 6-1 based on the system module diagram. There are five nodes in the basic network, which are identified as memsrvc and vp0, vp1, vp2, and vp3 nodes. Memsrvc is a member management node, which provides services such as registration, identity certificate issuance, and transaction certificate issuance when the node joins the federation chain. are the verification nodes in the Blockchain network, which together perform the functions of communication, consensus, block generation and verification in the block generation process. In the figure, vp4 represents a new node that applies to join the network.

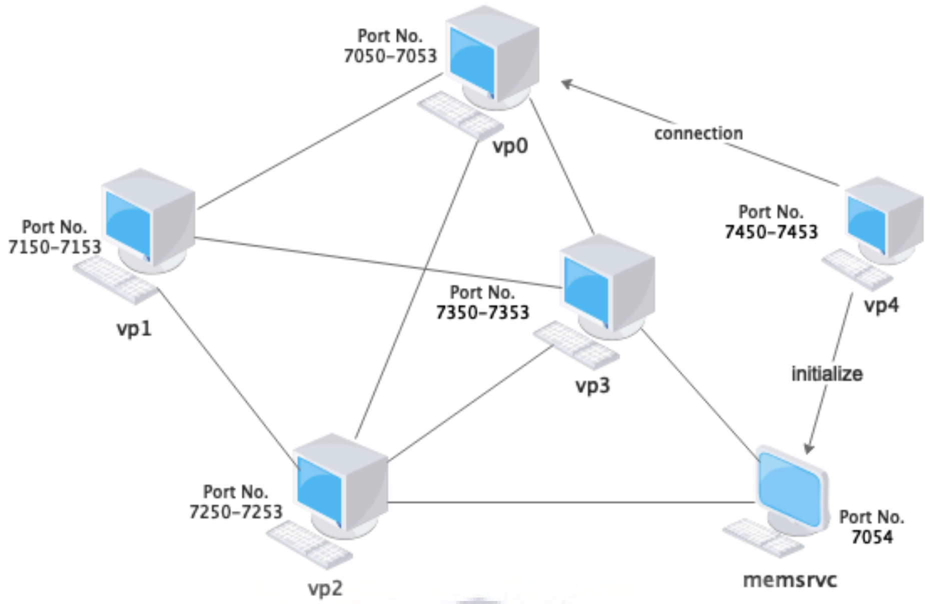


Figure 6-1: The Nest Network Topology

In the test environment, the mapping ports corresponding to each node in Docker are:

Memsrvc: 7054→7054;

Vp0: 7050-7053 →7050-7053;

Vpl: 7050-7053 →7150-7153;

Vp2: 7050-7053 →7250-7253;

Vp3: 7050-7053 →7350-7353;

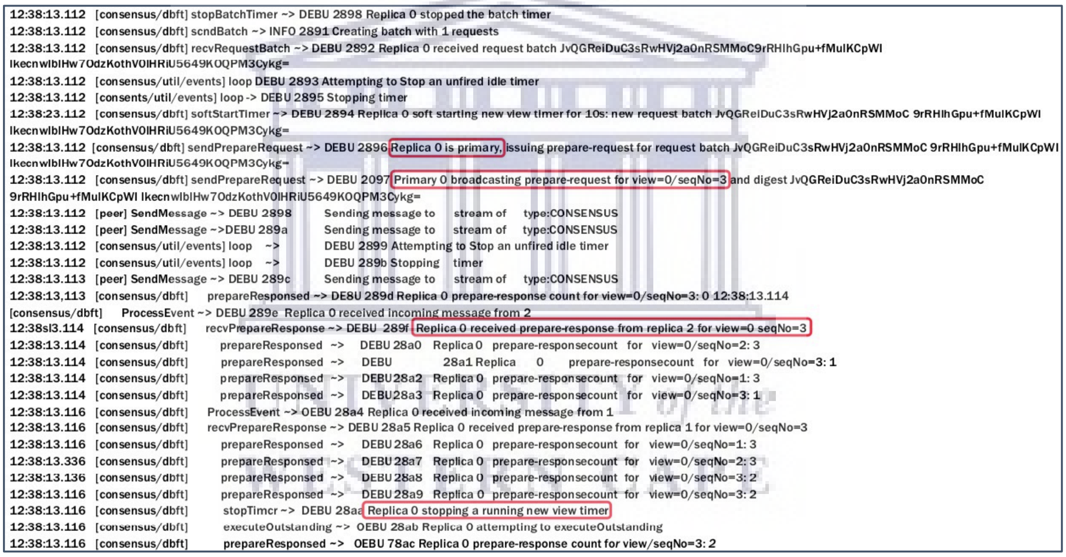
Vp4: 7050-7053 →7450-7453;

7050 port are used for information query; 7051 port are used to receive communication messages; 7052 port are temporarily discarded inside the alliance chain; 7053 port are used for listening events; and 7054 port are used for memsrvc node to process various requests.

6.1.2. Testing of the dBFT Consensus Mechanism

6.1.2.1. Testing of TPS

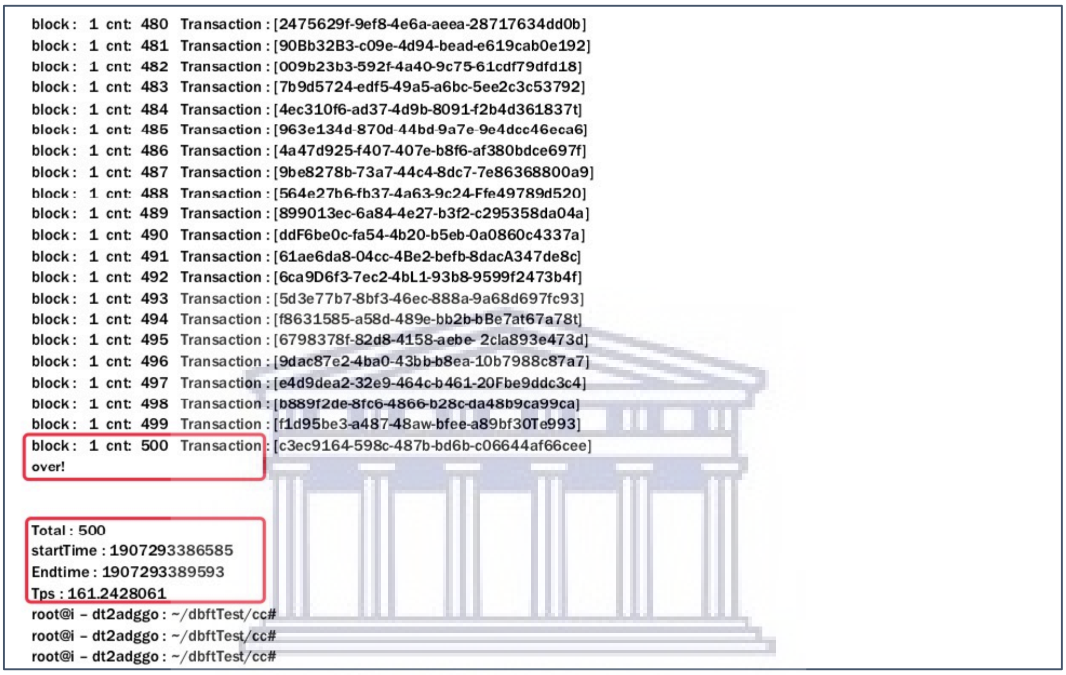
In the Blockchain network composed of vp0-vp3 (as shown in Figure 6-1), the dBFT consensus mechanism is tested in the CB-EHRs alliance chain. Figure 6-2 shows a round of consensus process for dBFT displayed on vp0. Vp0 in the network acts as the master node of the round of consensus and broadcasts prepare-request messages to other authentication nodes in the network. It then receives the prepare-response messages from vp1 and vp2, and reached the consensus before the timer expired. Finally, vp0 executes the transaction and generates a block.



```
12:38:13.112  [consensus/dbft] stopBatchTimer ~> DEBU 2898 Replica 0 stopped the batch timer
12:38:13.112  [consensus/dbft] scndBatch ~> INFO 2891 Creating batch with 1 requests
12:38:13.112  [consensus/dbft] recvRequestBatch ~> DEBU 2892 Replica 0 received request batch JvQGReiDuC3sRwHVj2a0nRSMMoC9rRHIhGpu+fMulKCpWI
IkecnwlblHw7OdzKothV0IHRiU5649KOQPM3Cykg=
12:38:13.112  [consensus/util/events] loop DEBU 2893 Attempting to Stop an unfired idle timer
12:38:13.112  [consents/util/events] loop -> DEBU 2895 Stopping timer
12:38:23.112  [consensus/dbft] softStartTimer ~> DEBU 2894 Replica 0 soft starting new view timer for 10s: new request batch JvQGReIDuC3sRwHVj2a0nRSMMoC 9rRHIhGpu+fMulKCpWI
IkecnwlblHw7OdzKothV0IHRiU5649KOQPM3Cykg=
12:38:13.112 [consensus/dbft] sendPrepareRequest ~> DEBU 2896 Replica 0 is primary, issuing prepare-request for request batch JvQGReiDuC3sRwHVj2a0nRSMMoC 9rRHIhGpu+fMulKCpWI
IkecnwlblHw7OdzKothV0IHRiU5649KOQPM3Cykg=
12:38:13.112  [consensus/dbft] sendPrepareRequest ~> DEBU 2097 Primary 0 broadcasting prepare-request for view=0/seqNo=3 and digest JvQGReiDuC3sRwHVj2a0nRSMMoC
9rRHIhGpu+fMulKCpWI IkecnwlblHw7OdzKothV0IHRiU5649KOQPM3Cykg=
12:38:13.112  [peer] SendMessage ~> DEBU 2898       Sending message to    stream of    type:CONSENSUS
12:38:13.112  [peer] SendMessage ~>DEBU 289a        Sending message to    stream of    type:CONSENSUS
12:38:13.112  [consensus/util/events] loop   ~>     DEBU 2899 Attempting to Stop an unfired idle timer
12:38:13.112  [consensus/util/events] loop   ~>     DEBU 289b Stopping   timer
12:38:13.113  [peer] SendMessage ~> DEBU 289c       Sending message to    stream of    type:CONSENSUS
12:38:13,113  [consensus/dbft]    prepareResponsed ~> DE8U 289d Replica 0 prepare-response count for view=0/seqNo=3: 0 12:38:13.114
[consensus/dbft]    ProcessEvent ~> DEBU 289e  Replica 0 received incoming message from 2
12:38sl3.114  [consensus/dbft]    recvPrepareResponse ~> DEBU  289f Replica 0 received prepare-response from replica 2 for view=0 seqNo=3
12:38:13.114  [consensus/dbft]      prepareResponsed  ~>    DEBU 28a0   Replica 0  prepare-responsecount  for  view=0/seqNo=2: 3
12:38:13.114  [consensus/dbft]      prepareResponsed  ~>    DEBU         28a1 Replica     0      prepare-responsecount  for  view=0/seqNo=3: 1
12:38:13.114  [consensus/dbft]      prepareResponsed  ~>    DEBU 28a2   Replica 0  prepare-responsecount  for  view=0/seqNo=1: 3
12:38:13.114  [consensus/dbft]      prepareResponsed  ~>    DEBU 28a3   Replica 0  prepare-responsecount  for  view=0/seqNo=3: 1
12:38:13.116  [consensus/dbft]    ProcessEvent ~> OEBU 28a4 Replica 0 received incoming message from 1
12:38:13.116  [consensus/dbft]    recvPrepareResponse ~> DEBU 28a5 Replica 0 received prepare-response from replica 1 for view=0/seqNo=3
12:38:13.116  [consensus/dbft]      prepareResponsed  ~>    DEBU 28a6   Replica 0  prepare-responsecount  for  view=0/seqNo=1: 3
12:38:13.336  [consensus/dbft]      prepareResponsed  ~>    DEBU 28a7   Replica 0  prepare-responsecount  for  view=0/seqNo=2: 3
12:38:13.136  [consensus/dbft]      prepareResponsed  ~>    DEBU 28a8   Replica 0  prepare-responsecount  for  view=0/seqNo=3: 2
12:38:13.116  [consensus/dbft]      prepareResponsed  ~>    DEBU 28a9   Replica 0  prepare-responsecount  for  view=0/seqNo=3: 2
12:38:13.116  [consensus/dbft]       stopTimcr ~>  DEBU 28aa Replica 0 stopping a running new view timer
12:38:13.116  [consensus/dbft]       executeOutstanding ~>  OEBU 28ab Replica 0 attempting to executeOutstanding
12:38:13.116  [consensus/dbft]       prepareResponsed ~>  OEBU 78ac Replica 0 prepare-response count for view/seqNo=3: 2
```

Figure 6-2: The dBFT Execution on vp0

While achieving the dBFT consensus, the dissertation also compared the consensus performance of dBFT and PBFT under the same network conditions. The test process is as follows. First, we enable dBFT in the CB-EHRs system. Then, we send 500, 1000, and 2000 transactions to the Blockchain network node in turn and wait for the Blockchain network to consensus, execute, and eventually generate the blocks. Afterward, Docker will calculate for us the TPS (Transaction Per Second) of the system. Finally, each of the above transactions is repeated 50 times to obtain a relatively stable processing capability of the network. Figure 6-3 shows one execution process and results of 500 transactions for the CB-EHRs system with dBFT enabled. Similarly, the PBFT consensus module is enabled in the CB-EHRs system, and the transaction processing capability test is performed in the same manner.



```
block :  1 cnt:  480  Transaction : [2475629f-9ef8-4e6a-aeea-28717634dd0b]
block :  1 cnt:  481  Transaction : [90Bb32B3-c09e-4d94-bead-e619cab0e192]
block :  1 cnt:  482  Transaction : [009b23b3-592f-4a40-9c75-61cdf79dfd18]
block :  1 cnt:  483  Transaction : [7b9d5724-edf5-49a5-a6bc-5ee2c3c53792]
block :  1 cnt:  484  Transaction : [4ec310f6-ad37-4d9b-8091-f2b4d361837t]
block :  1 cnt:  485  Transaction : [963e134d-870d-44bd-9a7e-9e4dcc46eca6]
block :  1 cnt:  486  Transaction : [4a47d925-f407-407e-b8f6-af380bdce697f]
block :  1 cnt:  487  Transaction : [9be8278b-73a7-44c4-8dc7-7e86368800a9]
block :  1 cnt:  488  Transaction : [564e27b6-fb37-4a63-9c24-Ffe49789d520]
block :  1 cnt:  489  Transaction : [899013ec-6a84-4e27-b3f2-c295358da04a]
block :  1 cnt:  490  Transaction : [ddF6be0c-fa54-4b20-b5eb-0a0860c4337a]
block :  1 cnt:  491  Transaction : [61ae6da8-04cc-4Be2-befb-8dacA347de8c]
block :  1 cnt:  492  Transaction : [6ca9D6f3-7ec2-4bL1-93b8-9599f2473b4f]
block :  1 cnt:  493  Transaction : [5d3e77b7-8bf3-46ec-888a-9a68d697fc93]
block :  1 cnt:  494  Transaction : [f8631585-a58d-489e-bb2b-bBe7at67a78t]
block :  1 cnt:  495  Transaction : [6798378f-82d8-4158-aebe- 2cla893e473d]
block :  1 cnt:  496  Transaction : [9dac87e2-4ba0-43bb-b8ea-10b7988c87a7]
block :  1 cnt:  497  Transaction : [e4d9dea2-32e9-464c-b461-20Fbe9ddc3c4]
block :  1 cnt:  498  Transaction : [b889f2de-8fc6-4866-b28c-da48b9ca99ca]
block :  1 cnt:  499  Transaction : [f1d95be3-a487-48aw-bfee-a89bf30Te993]
block :  1 cnt:  500  Transaction : [c3ec9164-598c-487b-bd6b-c06644af66cee]
over!


Total : 500
startTime : 1907293386585
Endtime : 1907293389593
Tps : 161.2428061
root@i – dt2adggo : ~/dbftTest/cc#
root@i – dt2adggo : ~/dbftTest/cc#
root@i – dt2adggo : ~/dbftTest/cc#
```

Figure 6-3: The Result of Sending 500 Transactions to the Blockchain Network

After collecting the test data, the dissertation used the line chart to compare the three sets of test results separately. The comparison results are shown in Figures 6-4, 6-5 and 6-6.

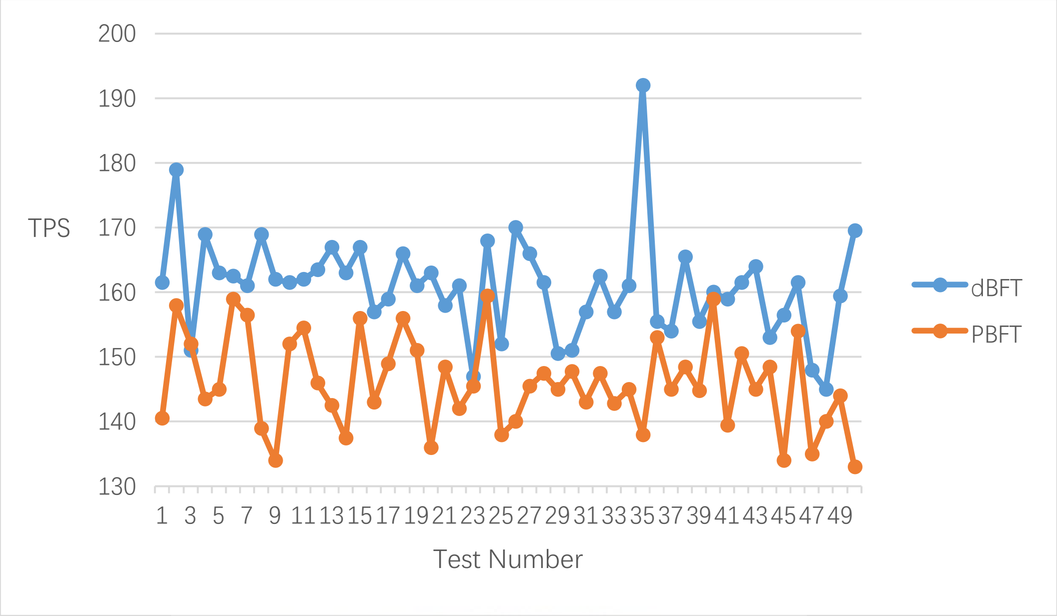


Figure 6-4: Comparison Result of 500 Transactions between dBFT and PBFT

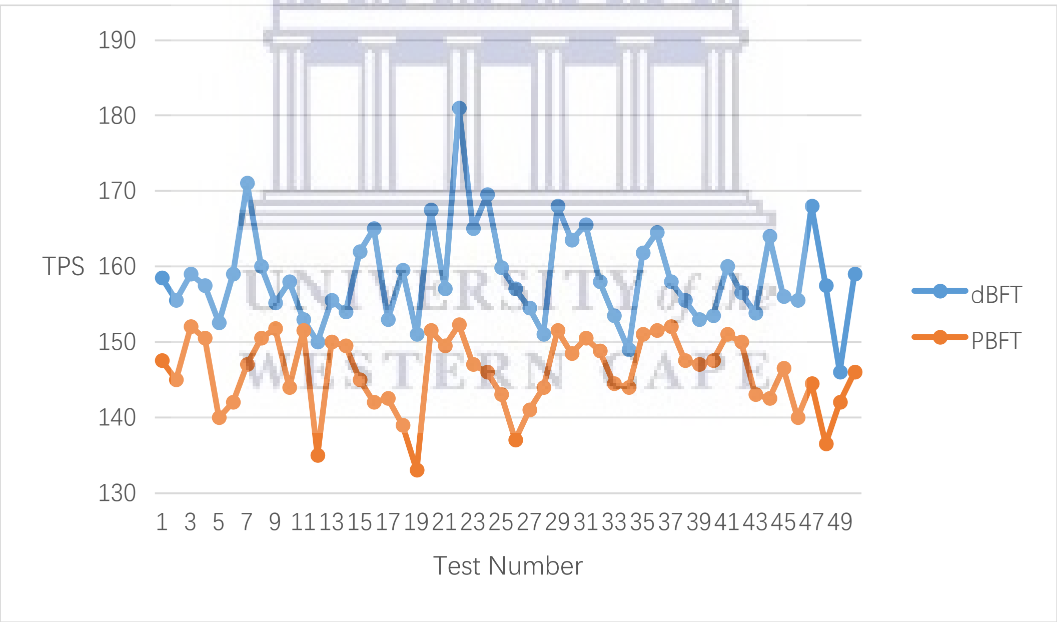


Figure 6-5: Comparison Result of 1000 Transactions between dBFT and PBFT

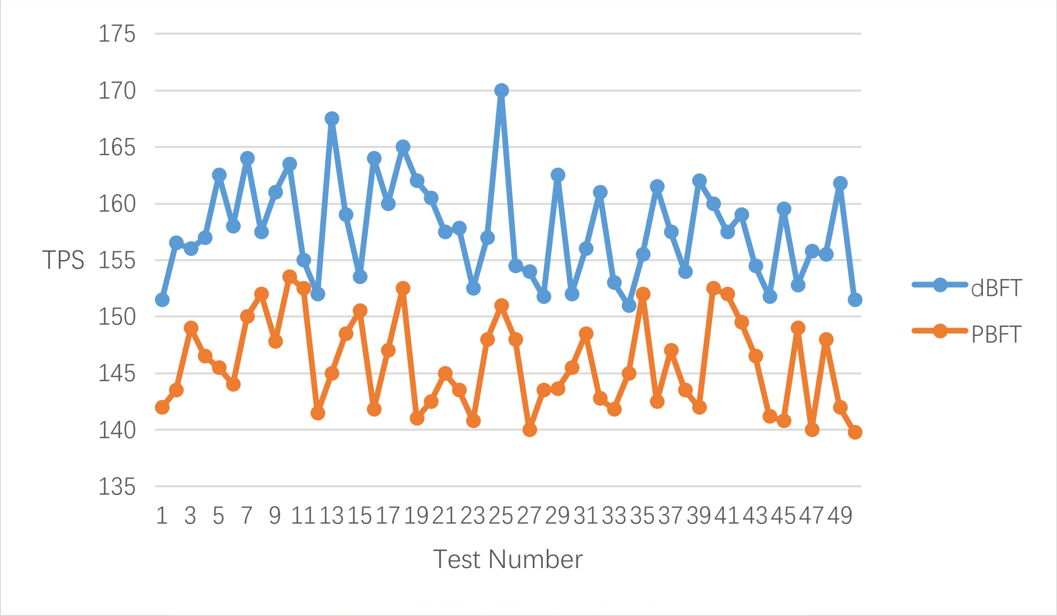


Figure 6-6: Comparison Result of 2000 Transactions between dBFT and PBFT

The dBFT algorithm used in this dissertation is an improvement of the PBFT algorithm. Excluding a few groups with large difference in result values caused by network fluctuations and transaction randomness, we can find from the above experimental diagrams that the transaction processing performance of dBFT is higher than PBFT in the three sets of experiments.

As mentioned in the fourth chapter of this dissertation, the time complexity of the PBFT algorithm is O(N^2)[73]. This means that when more nodes enter the network, the time at which a block is acknowledged grows uncontrollably. This situation will cause network congestion. At the same time, redundant nodes that exceed fault tolerance will reduce network efficiency. In order to have a clearer understanding of this characteristic, we conducted more tests (increasing the number of nodes in the above test conditions to 6 and 8).

In this dissertation, the TPS values in each of the above experimental data are averaged to obtain the network consensus performance stability value. The results of our summary are shown in Table 6-1. The three rows in the table represent the transaction processing capabilities of the two consensus mechanisms of dBFT and PBFT in networks with different consensus nodes. The vertical nine columns represent the data corresponding to their TPS when the volume is 500, 1000 and 2000 respectively.

Table 6-1: Performance Comparison between dBFT and PBFT

Number of nodes	dBFT / TPS			PBFT / TPS		
	500	1000	2000	500	1000	2000
4 Nodes	161.242806 1	158.7006085	157.701429	146.0908167	145.9727725	145.7940789
6 Nodes	152.387263 7	149.0378482	148.5736732	138.6473983	138.0394872	137.2839271
8 Nodes	134.536284 3	133.3748567	131.4758392	122.3748292	121.7384793	121.1923744

By analyzing Figures 6-4 to 6-6 and Table 6-1, on the one hand, we conclude that the transaction processing efficiency of dBFT is about 9% higher than that of PBFT under the same network conditions. On the other hand, as the number of nodes increases, the performance of the two consensus mechanisms decreases significantly. However, the dBFT consensus used in this dissertation is a way of using delegate voting to solve this problem. The core idea of this approach is that only the representative nodes selected by voting can obtain the accounting rights and participate in the consensus. Therefore, even if there are more network nodes, dBFT will select a few nodes to participate in the consensus to ensure the performance of the consensus mechanism. In summary, the CB-EHRs system designed in this dissertation can improve the performance of the consensus mechanism to a certain extent by using dBFT, which is consistent with the theoretical analysis results.

6.1.2.2.Testing of Latency and CPU Usage

At present, the requirement of PBFT algorithm for network bandwidth has been reduced to the level of polynomial. However, the consensus nodes in the network still cannot dynamically increase or decrease. As the number of nodes increases, the bandwidth requirements of the PBFT algorithm become higher. When the node's bandwidth is not high, it will sacrifice the throughput, latency and other performance of the consensus network. So in order to further study the advantages of the dBFT algorithm, we continue to install the Caliper, a blockchain test framework, in the previous Docker test environment for more tests.

Caliper is a blockchain performance benchmarking framework that allows users to test different blockchain solutions using predefined use cases to obtain a set of performance test results. The structure of the Caliper framework is shown in Figure 6-7.

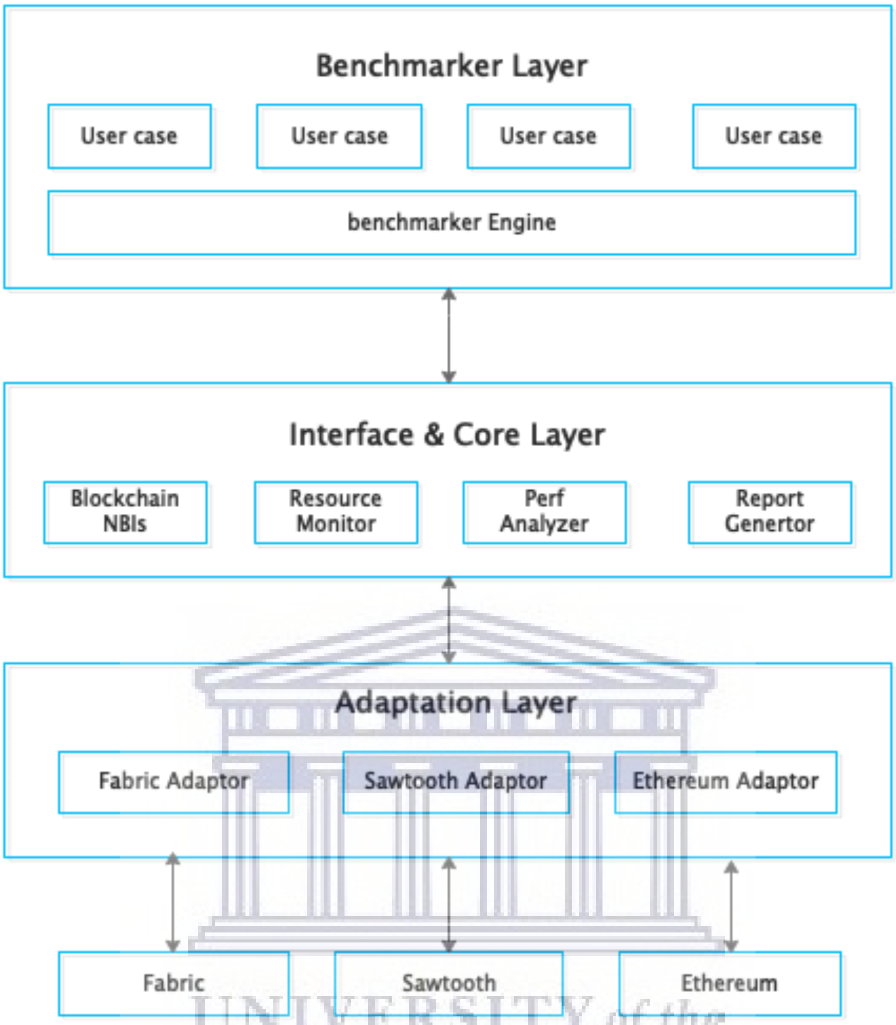
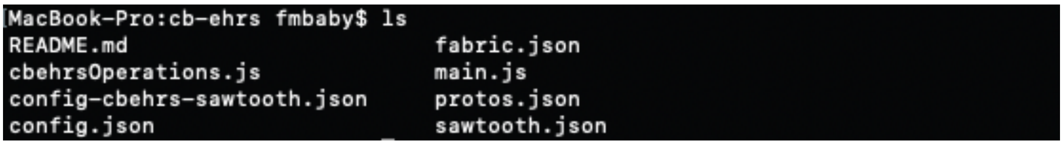


Figure 6-7: The structure of the Caliper framework

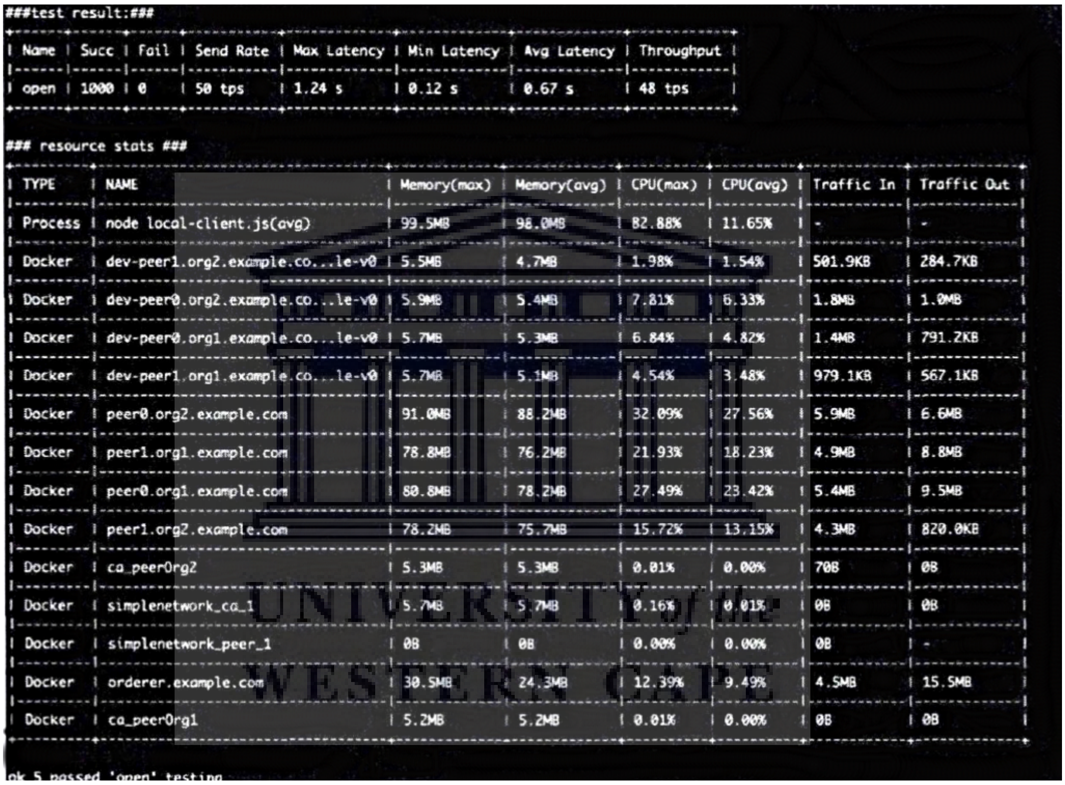
As can be seen from the above figure 6-7, Caliper can not only test fabric, but also test popular Etherum and Sawouth. Before our testing, we need to deploy the SDK of this project in the caliper directory. Figure 6-8 shows the test folder after deployment.



```
MacBook-Pro:cb-ehrs fmbaby$ ls
README.md                       fabric.json
cbehrsOperations.js             main.js
config-cbehrs-sawtooth.json     protos.json
config.json                     sawtooth.json
```

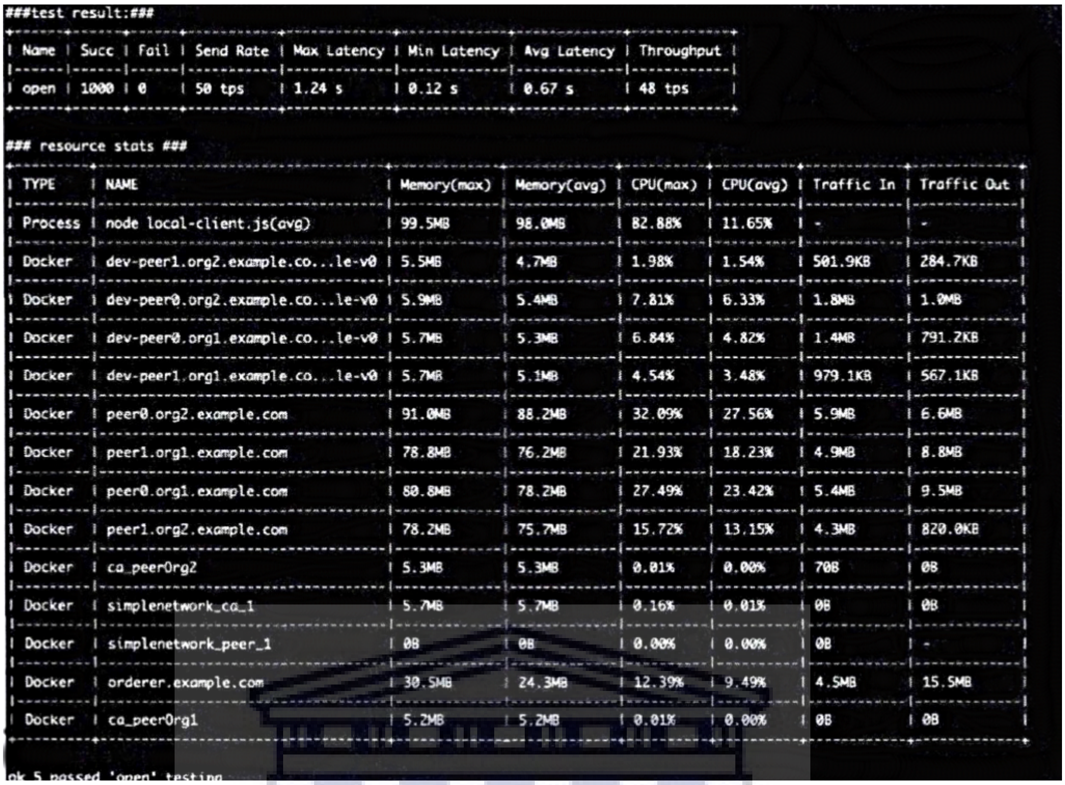
Figure 6-8: benchmark/cb-ehrs folder

We enable dBFT and PBFT in the CB-EHRs system, and then run the main.js file for testing. Caliper tests CB-EHRs based on relevant configuration information. Taking the account opening request as an example, we set different request rates to obtain test reports that meet the requirements. And each group of tests will be repeated 20 times to get a stable average. Finally, we collated the test results and observed the service latency and system resource occupation of the two consensus algorithms. Some reports are shown in Figures 6-9 and 6-10.



```
###test result:###
+------+------+------+-----------+-------------+-------------+-------------+------------+
| Name | Succ | Fail | Send Rate | Max Latency | Min Latency | Avg Latency | Throughput |
|------|------|------|-----------|-------------|-------------|-------------|------------|
| open | 1000 | 0    | 50 tps    | 1.24 s      | 0.12 s      | 0.67 s      | 48 tps     |
+------+------+------+-----------+-------------+-------------+-------------+------------+

### resource stats ###
+---------+------------------------------------+-------------+-------------+----------+----------+------------+-------------+
| TYPE    | NAME                               | Memory(max) | Memory(avg) | CPU(max) | CPU(avg) | Traffic In | Traffic Out |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Process | node local-client.js(avg)          | 99.5MB      | 98.0MB      | 82.88%   | 11.65%   | -          | -           |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | dev-peer1.org2.example.co...le-v0  | 5.5MB       | 4.7MB       | 1.98%    | 1.54%    | 501.9KB    | 284.7KB     |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | dev-peer0.org2.example.co...le-v0  | 5.9MB       | 5.4MB       | 7.81%    | 6.33%    | 1.8MB      | 1.0MB       |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | dev-peer0.org1.example.co...le-v0  | 5.7MB       | 5.3MB       | 6.84%    | 4.82%    | 1.4MB      | 791.2KB     |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | dev-peer1.org1.example.co...le-v0  | 5.7MB       | 5.1MB       | 4.54%    | 3.48%    | 979.1KB    | 567.1KB     |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | peer0.org2.example.com             | 91.0MB      | 88.2MB      | 32.09%   | 27.56%   | 5.9MB      | 6.6MB       |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | peer1.org1.example.com             | 78.8MB      | 76.2MB      | 21.93%   | 18.23%   | 4.9MB      | 8.8MB       |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | peer0.org1.example.com             | 80.8MB      | 78.2MB      | 27.49%   | 23.42%   | 5.4MB      | 9.5MB       |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | peer1.org2.example.com             | 78.2MB      | 75.7MB      | 15.72%   | 13.15%   | 4.3MB      | 820.0KB     |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | ca_peerOrg2                        | 5.3MB       | 5.3MB       | 0.01%    | 0.00%    | 70B        | 0B          |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | simplenetwork_ca_1                 | 5.7MB       | 5.7MB       | 0.16%    | 0.01%    | 0B         | 0B          |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | simplenetwork_peer_1               | 0B          | 0B          | 0.00%    | 0.00%    | 0B         | -           |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | orderer.example.com                | 30.5MB      | 24.3MB      | 12.39%   | 9.49%    | 4.5MB      | 15.5MB      |
|---------|------------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | ca_peerOrg1                        | 5.2MB       | 5.2MB       | 0.01%    | 0.00%    | 0B         | 0B          |
+---------+------------------------------------+-------------+-------------+----------+----------+------------+-------------+

ok 5 passed 'open' testing
```

Figure 6-9: Test report of PBFT when the request rate is 50tps



```
###test result:###
+------+------+------+-----------+------------+------------+------------+------------+
| Name | Succ | Fail | Send Rate | Max Latency | Min Latency | Avg Latency | Throughput |
|------|------|------|-----------|------------|------------|------------|------------|
| open | 1000 | 0    | 50 tps    | 1.24 s      | 0.12 s      | 0.67 s      | 48 tps     |
+------+------+------+-----------+------------+------------+------------+------------+

### resource stats ###
+---------+-----------------------------------+-------------+-------------+----------+----------+------------+-------------+
| TYPE    | NAME                              | Memory(max) | Memory(avg) | CPU(max) | CPU(avg) | Traffic In | Traffic Out |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Process | node local-client.js(avg)         | 99.5MB      | 98.0MB      | 82.88%   | 11.65%   | -          | -           |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | dev-peer1.org2.example.co...le-v0 | 5.5MB       | 4.7MB       | 1.98%    | 1.54%    | 501.9KB    | 284.7KB     |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | dev-peer0.org2.example.co...le-v0 | 5.9MB       | 5.4MB       | 7.81%    | 6.33%    | 1.8MB      | 1.0MB       |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | dev-peer0.org1.example.co...le-v0 | 5.7MB       | 5.3MB       | 6.84%    | 4.82%    | 1.4MB      | 791.2KB     |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | dev-peer1.org1.example.co...le-v0 | 5.7MB       | 5.1MB       | 4.54%    | 3.48%    | 979.1KB    | 567.1KB     |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | peer0.org2.example.com            | 91.0MB      | 88.2MB      | 32.09%   | 27.56%   | 5.9MB      | 6.6MB       |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | peer1.org1.example.com            | 78.8MB      | 76.2MB      | 21.93%   | 18.23%   | 4.9MB      | 8.8MB       |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | peer0.org1.example.com            | 80.8MB      | 78.2MB      | 27.49%   | 23.42%   | 5.4MB      | 9.5MB       |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | peer1.org2.example.com            | 78.2MB      | 75.7MB      | 15.72%   | 13.15%   | 4.3MB      | 820.0KB     |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | ca_peerOrg2                       | 5.3MB       | 5.3MB       | 0.01%    | 0.00%    | 70B        | 0B          |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | simplenetwork_ca_1                | 5.7MB       | 5.7MB       | 0.16%    | 0.01%    | 0B         | 0B          |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | simplenetwork_peer_1              | 0B          | 0B          | 0.00%    | 0.00%    | 0B         | -           |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | orderer.example.com               | 30.5MB      | 24.3MB      | 12.39%   | 9.49%    | 4.5MB      | 15.5MB      |
|---------|-----------------------------------|-------------|-------------|----------|----------|------------|-------------|
| Docker  | ca_peerOrg1                       | 5.2MB       | 5.2MB       | 0.01%    | 0.00%    | 0B         | 0B          |
+---------+-----------------------------------+-------------+-------------+----------+----------+------------+-------------+

ok 5 passed 'open' testing
```

Figure 6-10: Test report of PBFT when the request rate is 100tps

Figures 6-11 and 6-12 show the relationship between system latency and request rate.

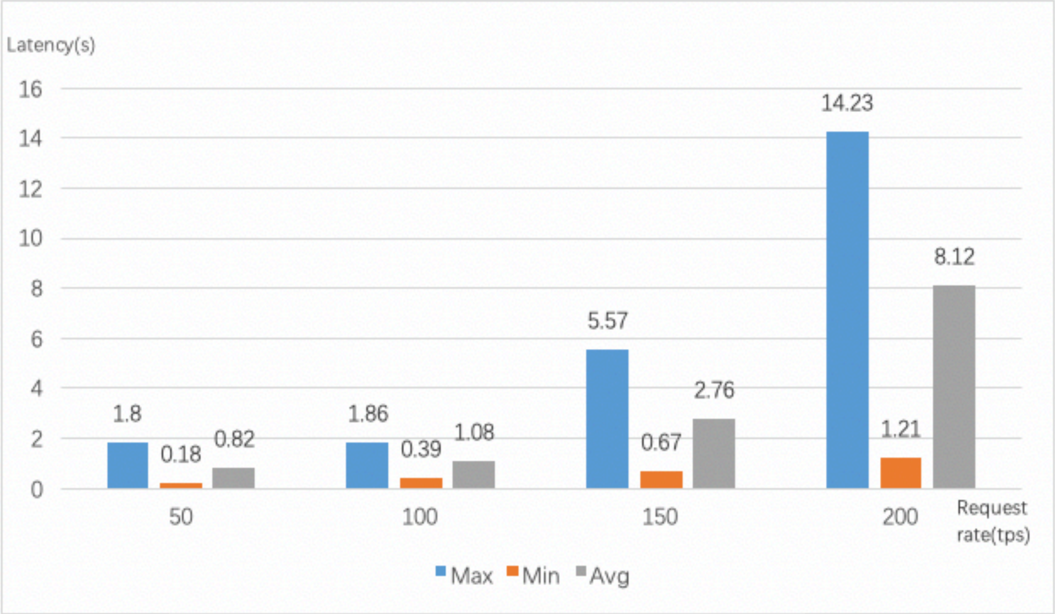


Figure 6-11: latency of the PBFT algorithm at different request rates

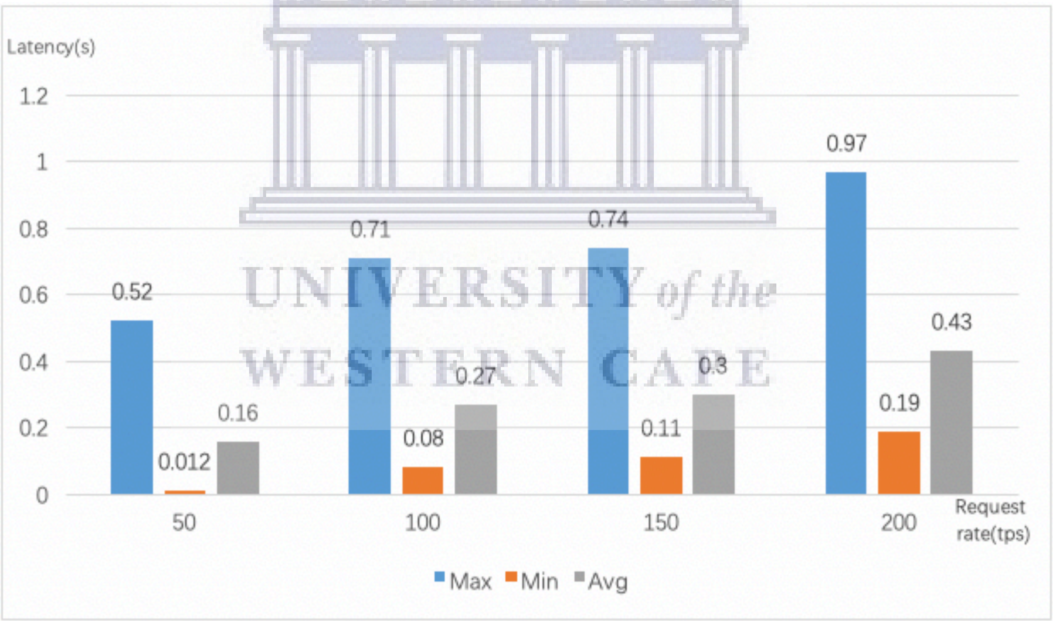


Figure 6-12: latency of the dBFT algorithm at different request rates

According to the two charts above（Figure 6-11 and Figure 6-12），when the two algorithms perform account opening transactions, the maximum latency, minimum latency and average latency of the system increase with the increase in the request sending rate. On the one hand, when the rate of request transmission reaches 200 TPS, the system latency of the PBFT algorithm will increase dramatically. This indicates that the request was sent too fast and it was difficult for the system to process it quickly. Therefore, the request is in the queue for too long, which causes the system latency to be too high. On the other hand, the latency increases of the system using dBFT is slow and can be maintained within 1 s.

Figure 6-13 shows the comparison of CPU usage rates of the two systems at different request rates.

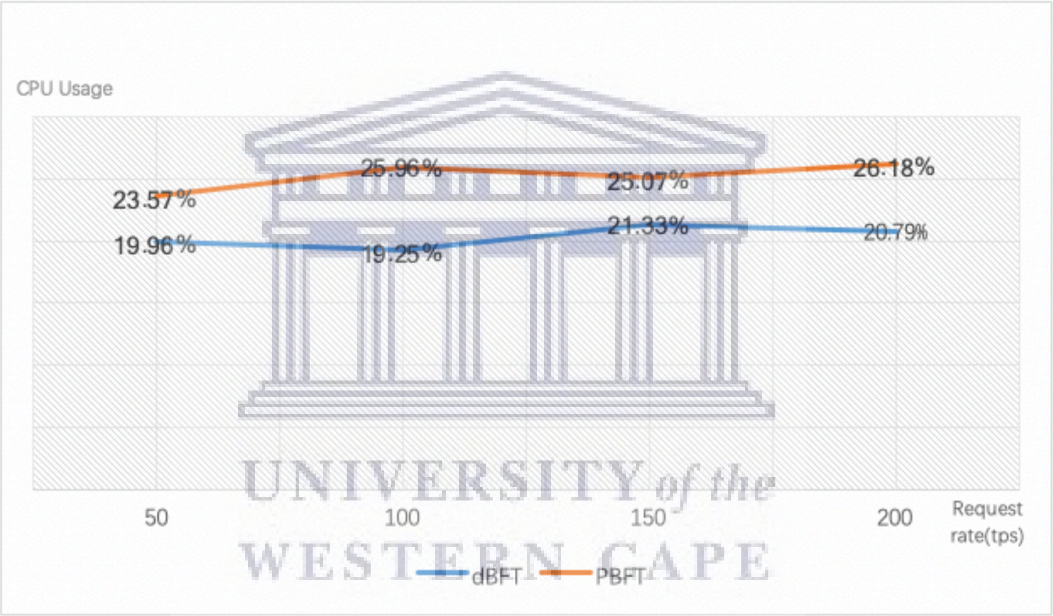


Figure 6-13: Comparison chart of average CPU usage

As a whole, the resource consumption rate of the dBFT algorithm is slightly lower than that of the PBFT algorithm.

Finally, we integrate all the test data into Table 6-2.

Table 6-2: System Latency and CPU Usage Test Result Table

Request rate (TPS)	Average Latency (s)	Average CPU Usage

	dBFT	PBFT	dBFT	PBFT
50	0.16	0.82	19.96%	23.57%
100	0.27	1.08	19.25%	25.96%
150	0.3	2.76	21.33%	25.07%
200	0.43	8.12	20.79%	26.18%

According to the above table, the dBFT algorithm has a greater improvement in system latency than the PBFT algorithm. When the account creation transaction is performed, the latency of the PBFT algorithm may even reach a few seconds or even more than 10 seconds. For systems with high latency requirements, the PBFT algorithm is difficult to meet their requirements. We can see that the latency of the consensus process of the dBFT algorithm stays within 1 second. As long as the requested TPS does not exceed the processing capacity of the node, the system latency can be kept within 1s without blocking.

The reason for the above phenomenon is that the consensus in the PBFT algorithm needs to be generated between all nodes, but the consensus in the dBFT algorithm is only performed between the elected representative nodes. As the number of nodes increases, the consensus time of the PBFT algorithm increases exponentially with the number of nodes. Also, the view change caused by the communication failure and the Byzantine uncertainty of the master node will cause the consensus latency to increase sharply. The dBFT algorithm avoids this, which makes the latency performance very good.

In terms of CPU usage, the CPU usage value of the dBFT algorithm remains around 20%, while the CPU usage value of the PBFT algorithm remains between 25% and 30%. Both can meet the current requirements of the blockchain system. Although the CPU usage of the two is not much different, we observe from Figure 6-13 that the average CPU usage of the dBFT algorithm is lower compared to the average CPU usage of the PBFT algorithm.

From the above analysis, it can be seen that compared to the PBFT algorithm, the dBFT algorithm has been improved in terms of latency performance and CPU usage. As a result, the blockchain using the dBFT algorithm can better meet the actual production needs, especially in alliance chains such as the CB-EHRs system.

6.2. System Function Test

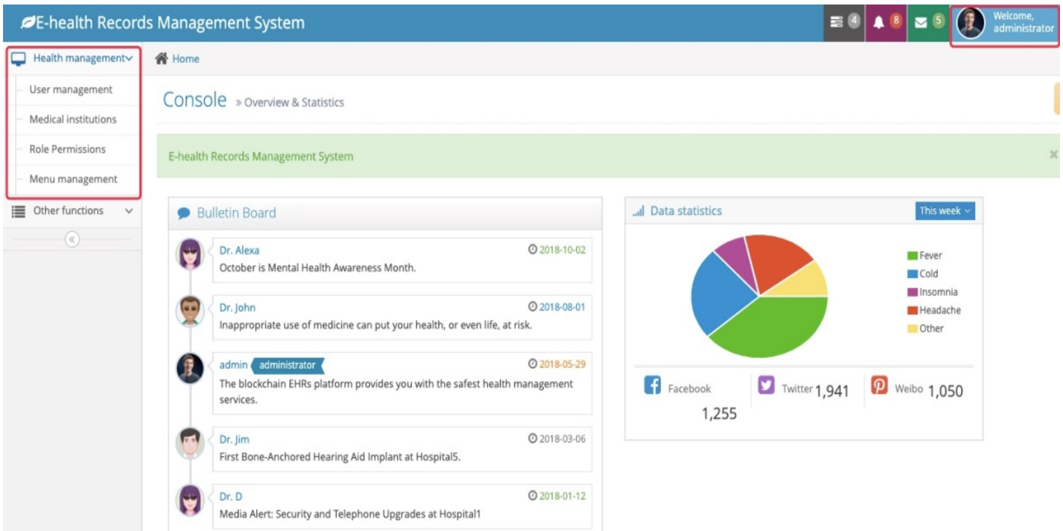
6.2.1. RBAC Privilege Control Test

In order to improve the security of the system and facilitate the operation of the user, the CB-EHRs system designed and implemented in this dissertation adopts the RBAC permission control mechanism. The system has three roles for administrators, doctors, and patients. The administrators have the highest authority. They are able to manage various settings of the system and permissions for other roles. Doctors and patients have only partial permissions that correspond to their identities. In order to clarify the permissions and privileges of each role, all the roles and corresponding permissions are summarized in Table 6-3.

Table 6-3: The Roles and the Permissions

Roles	Fill EHRs	Authorize EHRs	View EHRs(Owner)	Visit the EHRs Library	System Management	User Management
Doctor	√			√		
Patient		√	√			
Administrator					√	√

First, we create a default administrator A(administrator) for the system when the system is established. To test the system's permission control mechanism, we created two additional accounts B (Kun M) and C (Ling) whose default role is the patient role. Administrator A then changes the role of User B to the doctor role. Finally, we log in to ABC's previous account on the test PC. Through the processing of the permission control mechanism, the system showed them different operation menus. As shown in Figure 6-14 to 6-16.



Dr. Alexa 2018-10-02
October is Mental Health Awareness Month.

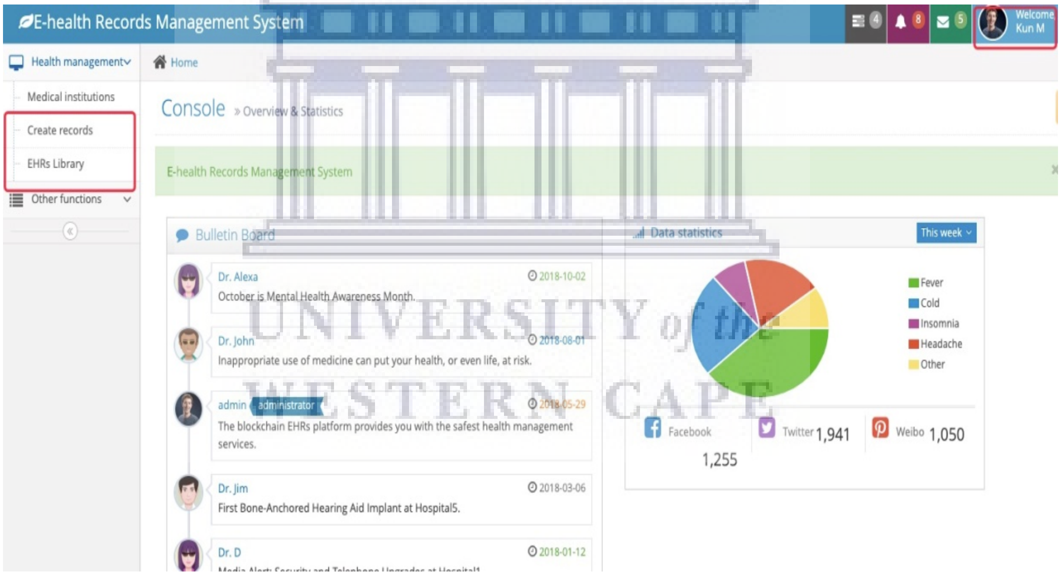
Dr. John 2018-08-01
Inappropriate use of medicine can put your health, or even life, at risk.

admin administrator 2018-05-29
The blockchain EHRs platform provides you with the safest health management services.

Dr. Jim 2018-03-06
First Bone-Anchored Hearing Aid Implant at Hospital5.

Dr. D 2018-01-12
Media Alert: Security and Telephone Upgrades at Hospital1

Figure 6-14: The Home Page Presented to User A



Dr. Alexa 2018-10-02
October is Mental Health Awareness Month.

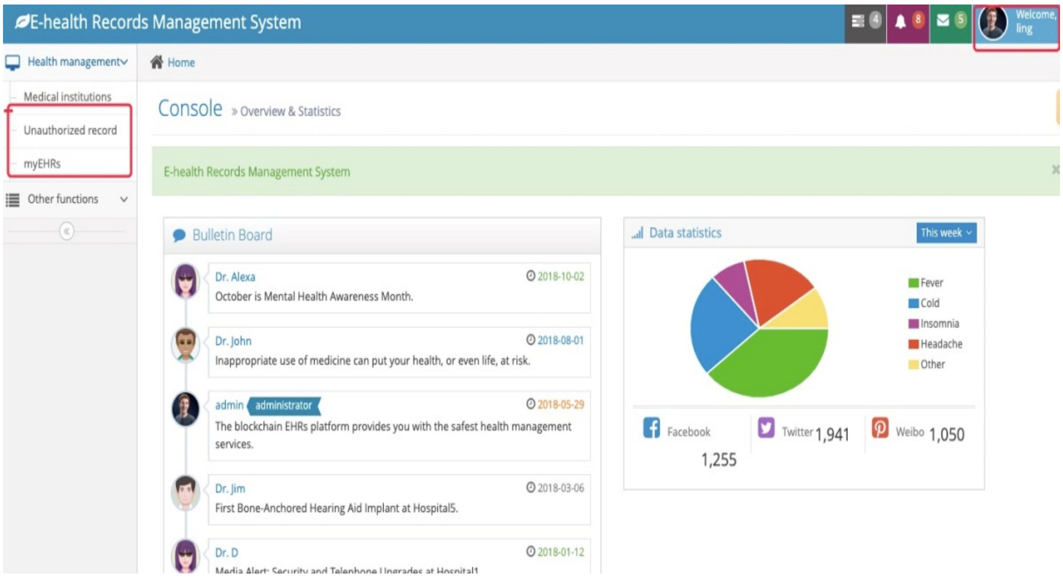
Dr. John 2018-08-01
Inappropriate use of medicine can put your health, or even life, at risk.

admin administrator 2018-05-29
The blockchain EHRs platform provides you with the safest health management services.

Dr. Jim 2018-03-06
First Bone-Anchored Hearing Aid Implant at Hospital5.

Dr. D 2018-01-12

Figure 6-15: The Home Page Presented to User B



Dr. Alexa 2018-10-02
October is Mental Health Awareness Month.

Dr. John 2018-08-01
Inappropriate use of medicine can put your health, or even life, at risk.

admin administrator 2018-05-29
The blockchain EHRs platform provides you with the safest health management services.

Dr. Jim 2018-03-06
First Bone-Anchored Hearing Aid Implant at Hospital5.

Dr. D 2018-01-12
Media Alert: Security and Telephone Upgrades at Hospital1

Figure 6-16: The Home Page Presented to User C

According to Figures 6-14,6-15 and 6-16, roles with different permissions get the pages corresponding to their role permissions. Therefore, the CB-EHRs system successfully implements the RBAC permission control mechanism, thereby improving the security of the system and facilitating the operation of the user.

6.2.2. Testing of User Registration Login Module

The front-end and back-end data interaction of the CB-EHRs system is based on servlet technology. Therefore, after writing the backend Java class (RegisterServlet) and front-end code (register.jsp), we also need to configure the RegisterServlet in the configuration file xml. This step allows the RegisterServlet class to receive requests from front-end pages. After configuring the program, we tested the user registration module in the integrated development environment myEclipse. The front-end registration page displayed by the register.jsp code is shown in Figure 6-17.

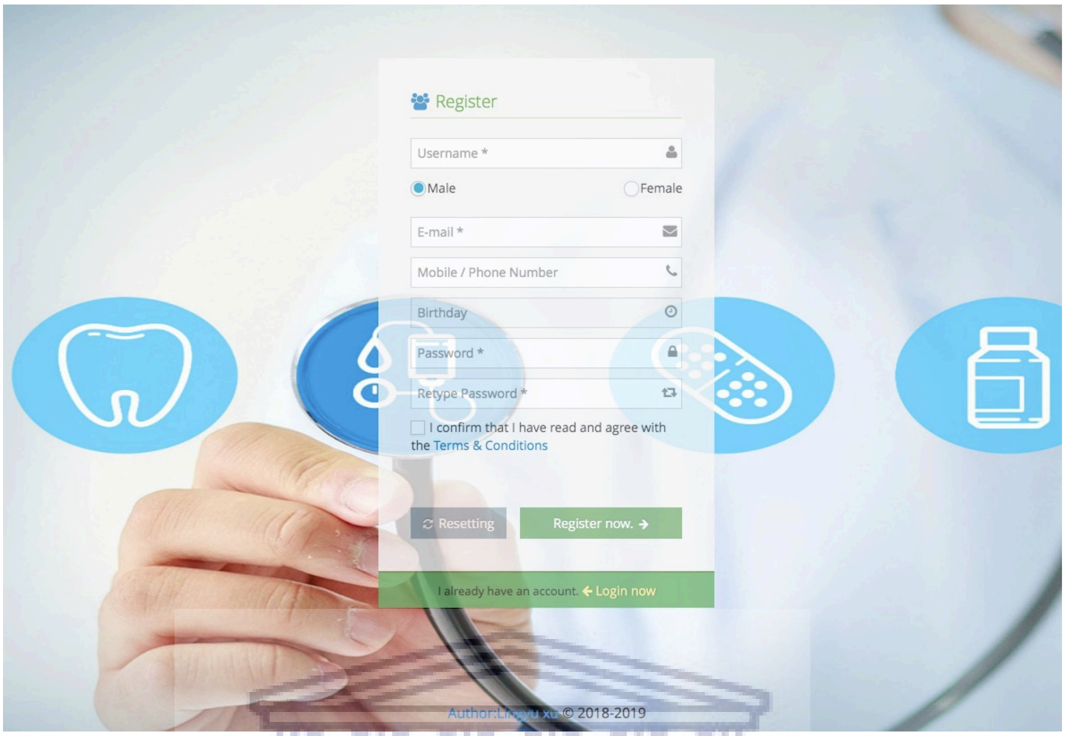

Register
Username *
● Male
○ Female
E-mail *
Mobile / Phone Number
Birthday
Password *
Retype Password *
□ I confirm that I have read and agree with the Terms & Conditions
Resetting
Register now. →
I already have an account. ← Login now
Author:Lingyu xu © 2018-2019

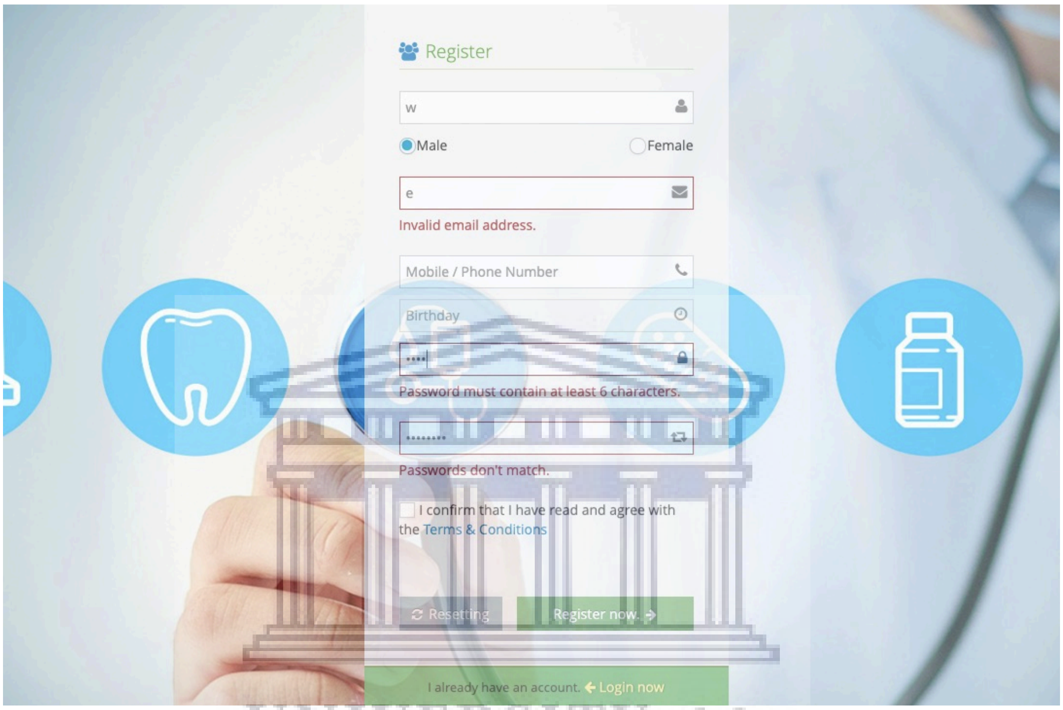
Figure 6-17: Registration Page

According to Figure 6-17, if new users want to register, they must enter the username, email address and password for the registered account on the registration page. They can also choose to fill in their gender, phone number and birthday information. According to the design and implementation scheme of the user registration module, the tests designed for this module are shown in Table 6-4.

Table 6-4: User Registration Module Test Table

Test Case	Test Results
Enter an invalid email address	System prompt: “Invalid email address.”
Enter an invalid password	System prompt: “Password must contain at least 6 characters.”
Enter a different confirmation password	System prompt: “Password don’t match.

According to the above test cases, this dissertation tests the e-mail, password and confirmation password input boxes separately, and displays the test results of illegal input as shown in Figure 6-18.



Register

w

● Male ○ Female

e

Invalid email address.

Mobile / Phone Number

Birthday

••••

Password must contain at least 6 characters.

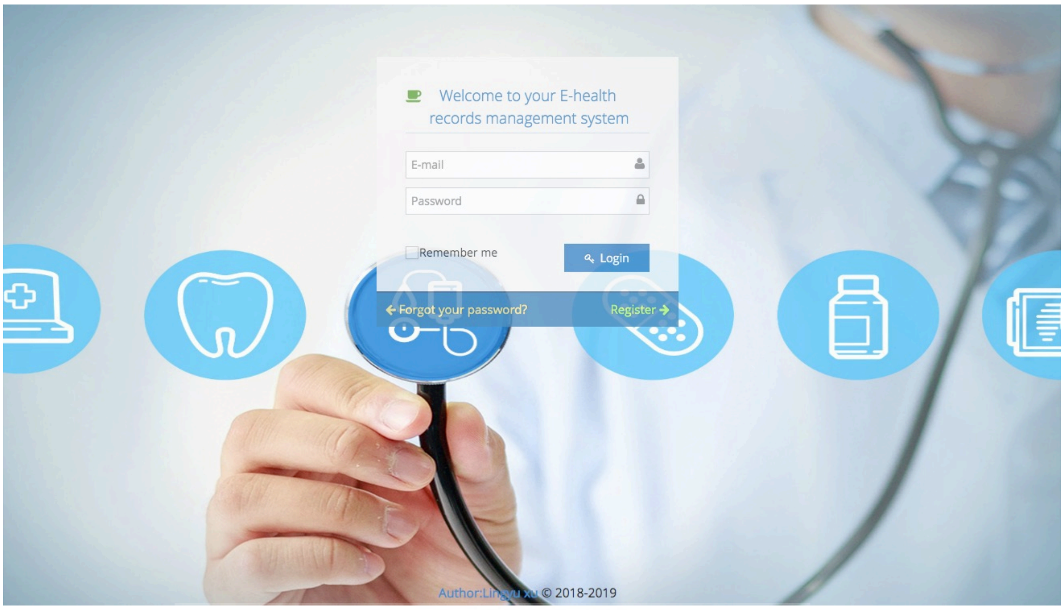
••••••••

Passwords don't match.

□ I confirm that I have read and agree with the Terms & Conditions

Figure 6-18: The Test Results of Illegal Input on Register Page

Similarly, when writing the backend Java class (LoginServlet) and the front-end code (login.jsp), we should configure the LoginServlet in the configuration file xml to implement the LoginServlet class to receive the request function from the login page. The front-end page displayed by the login.jsp code is shown in Figure 6-19.



Welcome to your E-health records management system

Author:Lingyu xu © 2018-2019

Figure 6-19: Login Page

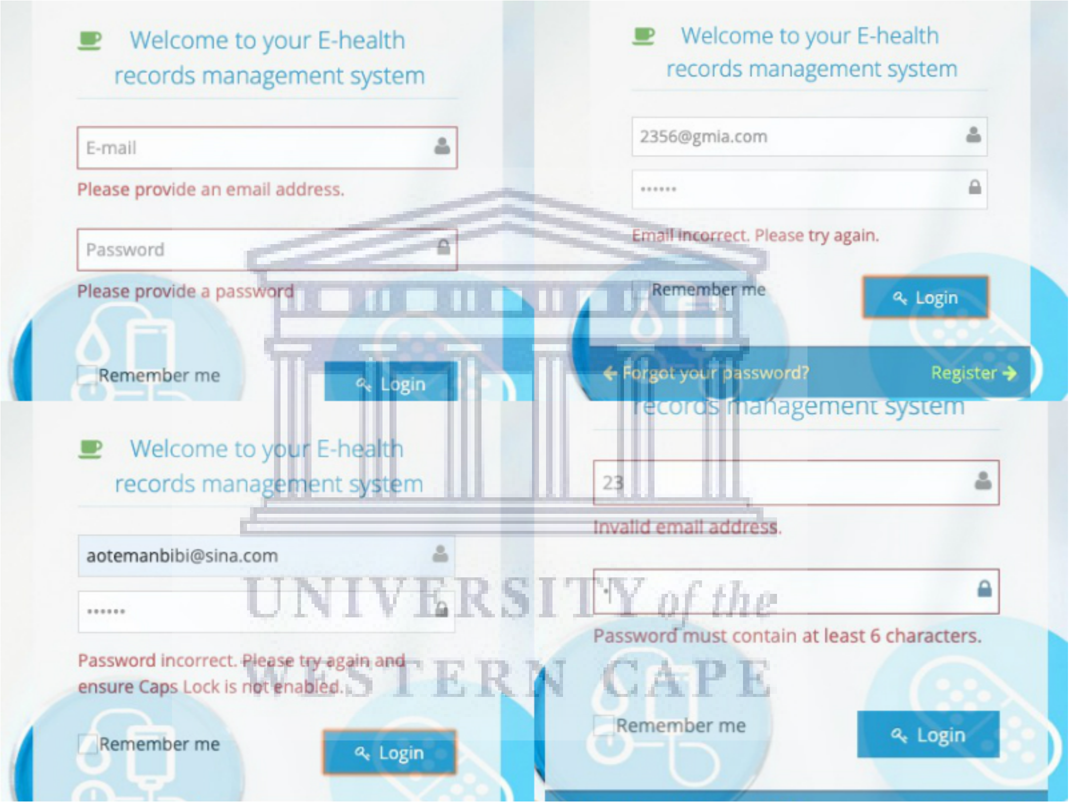
According to Figure 6-19, the registered user can enter the email address and password on the page, and click the login button to implement the login operation. According to the design and implementation scheme of the user login module, this dissertation designs the test cases shown in the following table 6-5 for the login module, and gives the corresponding test results.

Table 6-5: User Login Module Test Table

Test Case	Test Results
No email address entered	System prompt: “Please provide an email address.”
No password entered	System prompt: “Please provide a password.”
Enter an email address that has not been registered	System prompt: “Email incorrect. Please try again.
Enter a correct email address and password that does not match this account	System prompt: “Password incorrect. Please try again and ensure Caps Lock is not enabled.

Enter an invalid email address	System prompt: “Invalid email address.”
Enter an invalid password	System prompt: “Password must contain at least 6 characters.”

According to the content of the above test cases, the e-mail and password were tested separately. The input test result of the module is shown in Figure 6-20.


Welcome to your E-health records management system
E-mail
Password
Remember me
Login

Please provide an email address.

Please provide a password

Welcome to your E-health records management system
2356@gmia.com
••••••
Remember me
Login
Forgot your password?
Register

Email incorrect. Please try again.

Welcome to your E-health records management system
aotemanbibi@sina.com
••••••
Remember me
Login

Password incorrect. Please try again and ensure Caps Lock is not enabled.

records management system
23
Remember me
Login

Invalid email address.

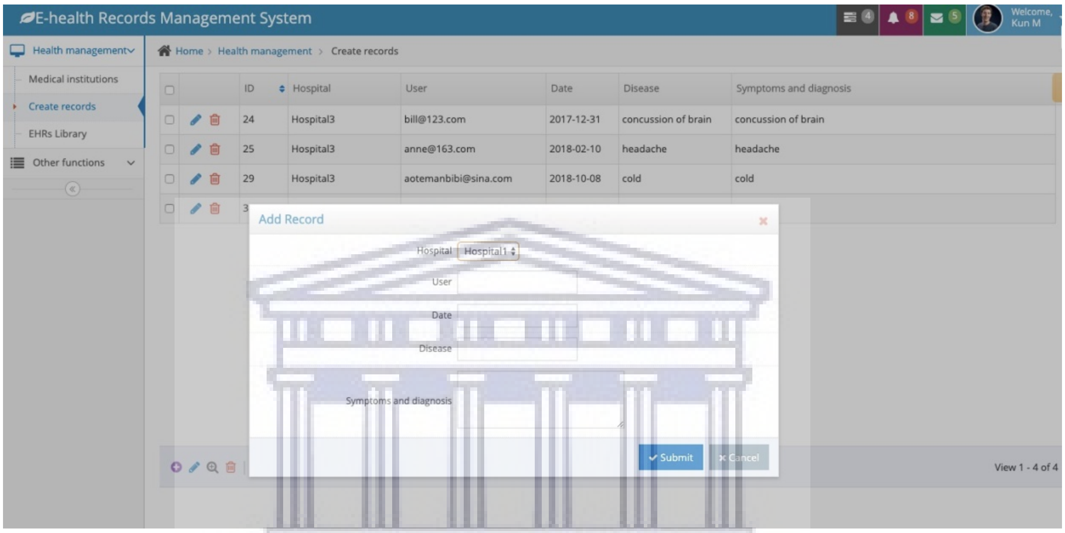
Password must contain at least 6 characters.

Figure 6-20: The Test Results of Illegal Input on Login Page

If the user has not performed the registration operation or the email address entered by the user does not correspond to the password, the login operation of the user will fail. When the login fails, the page prompts that the username or password is incorrect. Only users who have already registered, and the email and password they input are consistent with the data in the data table, will successfully log in to the system. When the user successfully logs in, the login page will automatically migrate to the system home page.

6.2.3. Testing of Data Authorization Module

First, we use a doctor's account to create an EHR for the test patient. The operation interface is shown in Figure 6-21. The doctor role needs to enter the correct patient's username to ensure that the subsequent tests go smoothly.



☐		ID	Hospital	User	Date	Disease	Symptoms and diagnosis
☐		24	Hospital3	bill@123.com	2017-12-31	concussion of brain	concussion of brain
☐		25	Hospital3	anne@163.com	2018-02-10	headache	headache
☐		29	Hospital3	aotemanbibi@sina.com	2018-10-08	cold	cold
☐		3					

Hospital: Hospital1

User:

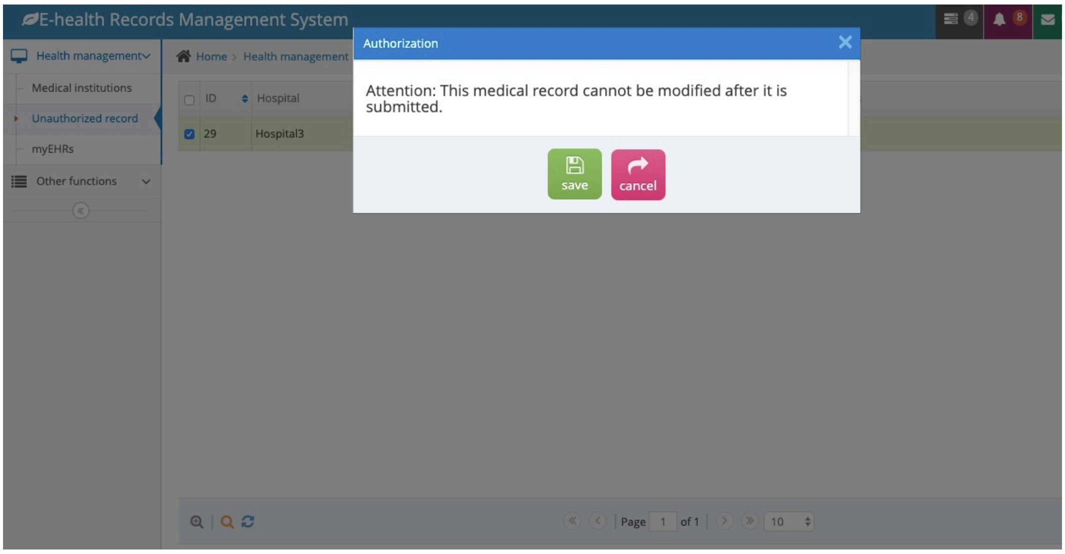
Date:

Disease:

Symptoms and diagnosis:

Figure 6-21: Add Record Page

If the test user has one electronic health record, we have tested the data authorization function. As with the previous module, we need to configure the authorizeFileServlet class in the xml file in order to implement the module's front-end data and background data interaction. The patient who successfully logged into the system selects the EHR that needs to be authorized in the generation authorization page, and clicks the confirmation authorization button to implement the authorization operation of the EHRs. The unauthorized page of the CB-EHRs system is shown in Figure 6-22.


E-health Records Management System

Health management

Medical institutions

Unauthorized record

myEHRs

Other functions

Home > Health management

☐	ID	Hospital
☑	29	Hospital3

Authorization

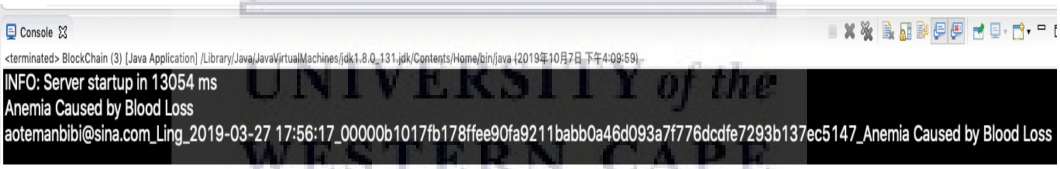
Attention: This medical record cannot be modified after it is submitted.

save cancel

Page 1 of 1   10

Figure 6-22: Data Authorization Page

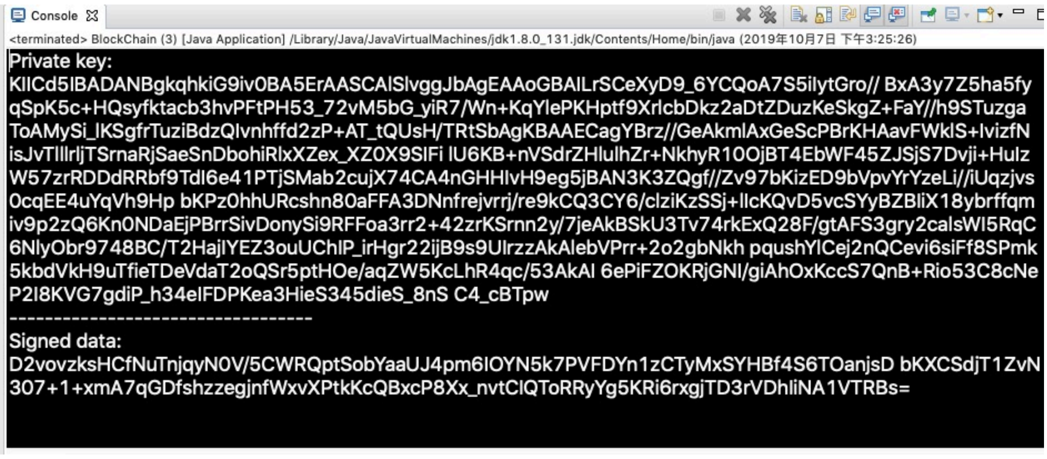
When we click the Confirm Authorization button, the authorizeFileServlet class and the MulticastController class will process the record selected by the user. In the background test, we stitched the record related information and printed it on the console, as shown in Figure 6-23.


Console

```
<terminated> BlockChain (3) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (2019年10月7日 下午4:09:59)
INFO: Server startup in 13054 ms
Anemia Caused by Blood Loss
aotemanbibi@sina.com_Ling_2019-03-27 17:56:17_00000b1017fb178ffee90fa9211babb0a46d093a7f776dcdfe7293b137ec5147_Anemia Caused by Blood Loss
```

Figure 6-23: Stitching the Record Information

According to Figure 6-23, the back-end Java code can correctly obtain the basic information of the authorization file. In the MulticastController class, the system will digitally sign the basic information of the EHRs using the user's private key before sending the data to the broadcast upload module. When debugging the test, we also printed the signed private key and the signed data in the console, as shown in Figure 6-24.



```
<terminated> BlockChain (3) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (2019年10月7日 下午3:25:26)
Private key:
KIICd5IBADANBgkqhkiG9iv0BA5ErAASCAISlvggJbAgEAAoGBAILrSCeXyD9_6YCQoA7S5ilytGro// BxA3y7Z5ha5fy
qSpK5c+HQsyfktacb3hvPFtPH53_72vM5bG_yiR7/Wn+KqYlePKHptf9XrlcbDkz2aDtZDuzKeSkgZ+FaY//h9STuzga
ToAMySi_lKSgfrTuziBdzQlvnhffd2zP+AT_tQUsH/TRtSbAgKBAAECagYBrz//GeAkmlAxGeScPBrKHAavFWklS+IvizfN
isJvTlllrljTSrnaRjSaeSnDbohiRlxXZex_XZ0X9SIFi lU6KB+nVSdrZHlulhZr+NkhyR10OjBT4EbWF45ZJSjS7Dvji+Hulz
W57zrRDDdRRbf9Tdl6e41PTjSMab2cujX74CA4nGHHIvH9eg5jBAN3K3ZQgf//Zv97bKizED9bVpvYrYzeLi//iUqzjvs
0cqEE4uYqVh9Hp bKPz0hhURcshn80aFFA3DNnfrejvrrj/re9kCQ3CY6/clziKzSSj+llcKQvD5vcSYyBZBliX18ybrffqm
iv9p2zQ6Kn0NDaEjPBrrSivDonySi9RFFoa3rr2+42zrKSrnn2y/7jeAkBSkU3Tv74rkExQ28F/gtAFS3gry2calsWI5RqC
6NlyObr9748BC/T2HajlYEZ3ouUChlP_irHgr22ijB9s9UlrzzAkAlebVPrr+2o2gbNkh pqushYlCej2nQCevi6siFf8SPmk
5kbdVkH9uTfieTDeVdaT2oQSr5ptHOe/aqZW5KcLhR4qc/53AkAI 6ePiFZOKRjGNI/giAhOxKccS7QnB+Rio53C8cNe
P2I8KVG7gdiP_h34eIFDPKea3HieS345dieS_8nS C4_cBTpw
---------------------------------
Signed data:
D2vovzksHCfNuTnjqyN0V/5CWRQptSobYaaUJ4pm6IOYN5k7PVFDYn1zCTyMxSYHBf4S6TOanjsD bKXCSdjT1ZvN
307+1+xmA7qGDfshzzegjnfWxvXPtkKcQBxcP8Xx_nvtClQToRRyYg5KRi6rxgjTD3rVDhliNA1VTRBs=
```

Figure 6-24: Private Key and Signed Data

6.2.4. Testing of Broadcast Upload Module

The data authorization module packages process the EHRs and transmit them to the P2P network. They are verified by nodes in the network using a consensus mechanism. The validated EHRs data is added to the database EHRs table.

The user registration login module is the system entrance. It enables the system to better manage uploaded data. Therefore, in order to test the synchronization of the system, we tested the system update in the LAN environment when multiple users simultaneously logged in and uploaded the EHR. Based on this, the dissertation designs the test flow shown in Figure 6-25 below.

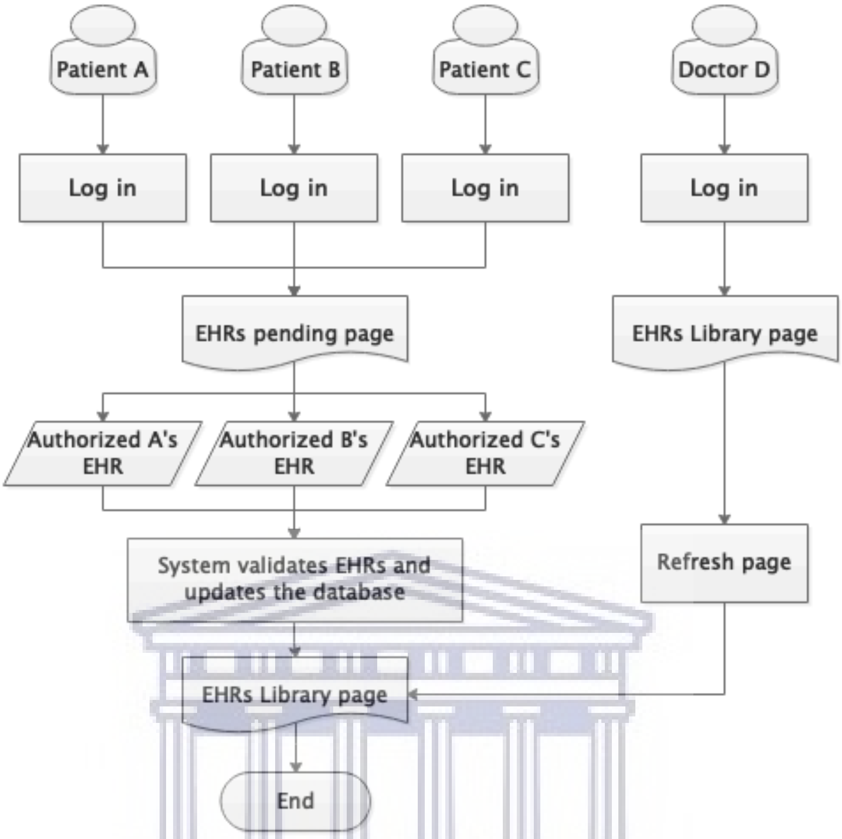
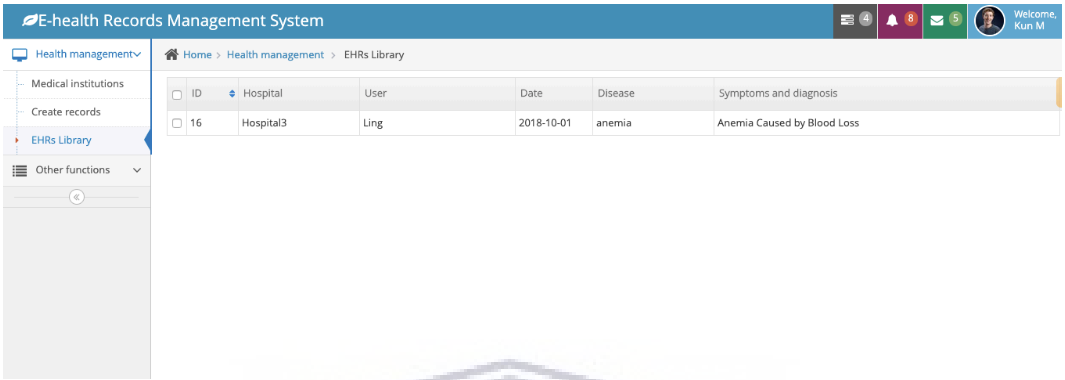


Figure 6-25: Multiple Users Login Test

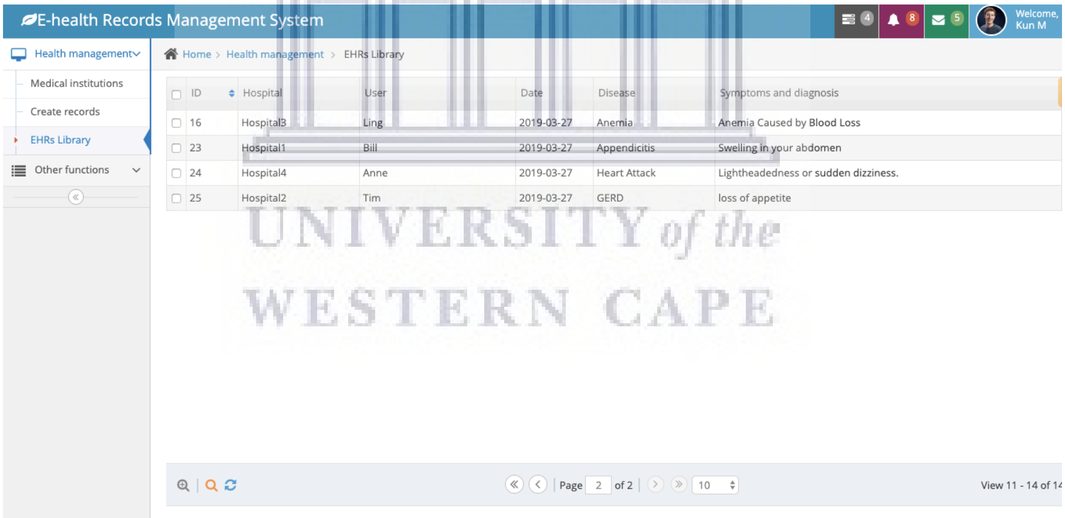
According to the flow of Figure 6-25, We connect computers A, B, C and D to the same router so that the four test computers are in the LAN environment. Patient A (Bill) logs in on computer 1, patient B (Anne) logs in on computer 2, patient C (Tim) logs in on computer 3, and doctor D (Kun M) logs in on computer 4. They all access system engineering through the IP address feature in the LAN environment (10.108.55.81:8080/project engineering). After entering the email address and password on the login interface, all four users can successfully log in and enter the main page of the system.

After the three patients log into the system, they each choose one electronic health record which belongs to them for authorization. When the three patients click on the “confirm authorization” after selecting the file, the doctor refreshes the EHRs library of the system and can find the EHRs authorized by the three patients to upload. Figure 6-26 shows the pre-test EHRs library page on the doctor's computer. Figure 6-27 shows the corresponding EHRs library page after testing.



□	ID	Hospital	User	Date	Disease	Symptoms and diagnosis
□	16	Hospital3	Ling	2018-10-01	anemia	Anemia Caused by Blood Loss

Figure 6-26: The Pre-test EHRs Library Page on the Doctor's Computer



□	ID	Hospital	User	Date	Disease	Symptoms and diagnosis
□	16	Hospital3	Ling	2019-03-27	Anemia	Anemia Caused by Blood Loss
□	23	Hospital1	Bill	2019-03-27	Appendicitis	Swelling in your abdomen
□	24	Hospital4	Anne	2019-03-27	Heart Attack	Lightheadedness or sudden dizziness.
□	25	Hospital2	Tim	2019-03-27	GERD	loss of appetite

Figure 6-27: The EHRs Library Page on the Doctor's Computer

The EHRs library page of doctor computer is updated and shows the EHR authorized by the three patients to upload. This means that the system node can consistently obtain all user upload information at the current time.

6.3. User Satisfaction Survey of CB-EHRs System

A study pointed out that any information system design needs to take into account hardware, software, users and other factors[107]. Health care workers and patients are direct users of the CB-EHRs system, and their satisfaction is an important indicator for evaluating the CB-EHRs system. Therefore, the author invited 15 (anonymous) users from Affiliated Hospital of Chengdu University of Traditional Chinese Medicine to conduct a 10-day CB-EHRs usage satisfaction survey. The details of the survey are as follows.

Purpose of the survey: (1) To investigate the status of users' satisfaction with CB-EHRs system and provide evidence for promoting users' usage of CB-EHRs system. (2) To collect the opinions given by the actual users and provide the basis for the system improvement.

Survey respondent: (1) Five surgeon. (2) Five nurses from nursing department. (3) Five patients who are being treated by these doctors.

Survey method: After the user used the CB-EHRs system for ten days, the author used the questionnaire to collect feedback from the users. The survey mainly includes the user's satisfaction with the system function, system interface and service response. The questionnaire was based on a rating scale (1 to 5), which represents very dissatisfied, dissatisfied, ordinary, satisfied and very satisfied.

The final CB-EHRs system satisfaction survey is shown in Table 6-6.

Table 6-6: User Satisfaction Survey

Users (Email)	System functions	System interface	Service response	Evaluation and recommendations
Surgeon 1 (gro**liu@sohu. com)	4	4	4	It's already good to be able to securely share data.
Surgeon 2 (kik**015@163. com)	3	5	4	If you can complete the functions of information release

				and statistics, as soon as possible, the system will become more perfect.
Surgeon 3 (cxa**520@hotmail.com)	3	4	4	Can support normal medical work, and hope to have more comprehensive system functions.
Surgeon 4 (wan**i88@126.com)	2	4	4	The function is not perfect. It is not efficient to communicate via mailbox, and I hope to change to a real-time communication method similar to WeChat.
Surgeon 5 (cai***_55@sina.com)	4	5	4	The system interface is very good. Data sharing can help me better understand the patient's condition.
Nurse 1 (lis***yin@126.com)	4	5	4	Standardize internal management and reduced workload.
Nurse 2 (wan***uan233@sohu.com)	3	5	3	If you can join the online payment function, the system will be better.
Nurse 3 (lih***g66@sina.com)	3	5	3	The system is sometimes not smooth enough. But the interface is beautiful and easy to operate. I hope this system has more features.
Nurse 4 (liw***eng_2012@126.com)	3	5	4	The system interface is beautiful but has fewer functions.
Nurse 5 (xia***ian@126.com)	3	4	5	I hope to classify the description of the condition. For example, data such as body temperature and blood pressure can be recorded separately.
Patient 1 (zen***009@163.com)	3	4	5	Communicating by email is not particularly convenient for me.
Patient 2 (wan***123@sina.com)	4	5	4	The system can clearly show me my medical treatment process.
Patient 3 (wb1***988@h	3	4	4	If you can join the online payment

otmail.com)				function, the system will be better.
Patient 4 (coc***345@163.com)	3	4	4	I hope the interface can bring more health warnings to patients.
Patient 5 (den***n00@sohu.com)	4	4	4	Being able to manage your own EHRs is a great design. The anonymized design guarantees the privacy of the patients is also very good.
The average score	3.27	4.47	4.00	\

Survey results: All users are satisfied with the user interface of the CB-EHRs system. In terms of service response, most users can use the system smoothly during the satisfaction survey. Only two users have responded abnormally during this time. In terms of system functions, a few users suggest that we add multiple functions (such as payment function). This part will be considered for future work.

Summary: The main topic of this dissertation is the application of blockchain technology in data sharing and privacy protection. Therefore, the author mainly realized the functions related to the blockchain. But a good application should be to make people's lives more convenient. Therefore, looking forward to the future of the CB-EHRs system, it needs to improve its functions for more user needs.

7. Summary and Outlook

7.1. Summary of Dissertation

Blockchain technology originated from the Bitcoin network proposed by Nakamoto[24]. It has gradually attracted the attention of all parties with the development of Bitcoin. However, a new technology still takes five to ten years before transition, from the attention of its researchers, to the scale of promotion and application. It is currently the developmental stage of the application of the Blockchain technology globally. Different from the existing network services and data storage methods, the decentralization and node-to-peer characteristics of Blockchain technology enable it to reduce credit risk and operating costs. Blockchain technology effectively prevents system failures and hacker attacks by using chained data books and cryptography.

With data sharing as the application background, this thesis mainly develops a CB-EHRs management and sharing system based on Blockchain technology, by exploring asymmetric encryption technology and key technologies related to Blockchain. Throughout the project, this dissertation follows the life cycle of the software development process. This work details the technical research for the implementation of the system, and documents the system requirements analysis, overall design and module design for the system. Furthermore, the implementation and testing of various modules of the system was conducted. This work explores a wide-range of relevant literature and theoretical principles in its analysis, development and implementation phases. The main work includes six aspects: technical analysis, requirements analysis, system design, system implementation, system testing, and documentation.

The CB-EHRs system is still in the R&D (Research and Development) testing phase and has not been officially put into use. The current implementation of this project provides a certain theoretical basis and practical experience for learning and developing projects based on Blockchain technology. In this CB-EHRs system, new users are registered as part of the system. The system uses the RBAC rights management mechanism to manage the access rights of users of different roles to the system page. Medical professionals can fill out EHRs for patients and access the full EHRs library. Patients can manage their own EHRs by selecting whether to upload their own EHRs through an authorization scheme. In the process of implementing the underlying technology of the system, this dissertation combines asymmetric encryption technology and data sharing. This dissertation also uses cryptography to process uploaded EHRs, so that the user-uploaded raw data is separated from the summary information and is also associated with the summary information. In the process of sharing EHRs, the MD5withRSA digital signature algorithm signs and verifies the summary information of EHRs. Finally, the system sends the signature to the P2P network through the multicast communication mode for other nodes to receive. According to the business characteristics of this CB-EHRs management system, this dissertation deliberately chooses the appropriate dBFT consensus algorithm as the consensus mechanism of the Blockchain system.

The CB-EHRs system is no longer the same as the traditional centralized data management sharing platform. It has the characteristics of decentralization, trust and collective maintenance. This design improves the security of the data. Users can develop trusted management and share data under the system platform without having to worry about data being deleted or modified after being maliciously attacked. As a new type of network service based on Blockchain, this system can be used as a reference for future Blockchain applications in the fields of distance education, culture and entertainment and public service.

7.2. Future Work

This system is a project based on JavaWeb development. In the process of development and testing, the system uses related theories and techniques such as hash algorithm, digital signature algorithm, P2P network and MerkleTree technology. These technologies effectively improve the viability of the implementation system and shorten the development cycle and cost. The reliability and security of the final system has been significantly improved. However, in the specific practice process, there are still some places in this work that need further improvement in the later stage. Some issues which could be the focus of further research work are:

1. Continue to optimize Blockchain technology. The research on Blockchain technology in this thesis mainly focuses on the application research level. However, Blockchain technology is still in the development stage. Because of the many limitations of the technology itself, the Blockchain cannot be applied on a large scale, such as restrictions on block capacity and data storage. Most of the information stored in the Blockchain is the summary information of the transaction. Transactions are also formatted in a standardized format, and non-structured data is currently not directly stored in blocks. In addition, the storage of the block has problems such as data redundancy. Therefore, researchers can continue to study the storage and management of block data in the Blockchain.
2. Improve the application. Although the CB-EHRs system proposed in this dissertation realizes the expansion of Blockchain technology in the field of electronic medical treatment, the business logic of the application itself is relatively simple. The system user interface also needs to be improved. The project did not achieve the commercial application goals of the EHRs system based on Blockchain technology. Therefore, the researcher can continue to improve the function and performance, and explore the commercial development direction of the system.

There are certain technical difficulties and challenges for researchers to complete the above two tasks in a short time. However, Blockchain technology has received more and more attention from scholars and industry professionals. Hence, it is still at a stage of development and exploration. The above problems and other difficulties encountered in the application process of the Blockchain will also be gradually addressed.

REFFERENCES

1. Social, W.A. *DIGITAL IN 2018: WORLD'S INTERNET USERS PASS THE 4 BILLION MARK*. 2018 [cited 30.01.2018; Available from: https://wearesocial.com/blog/2018/01/global-digital-report-2018. (Last viewed: March 2020)

2. Fern'ndez, G., I. De La Torre-díez, and J.J. Rodrigues. *Analysis of the cloud computing paradigm on mobile health records systems*. in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. 2012. IEEE.

3. Chu, X., et al. *An Empirical Study on the Intention to Use Online Medical Service*. in *2018 15th International Conference on Service Systems and Service Management (ICSSSM)*. 2018. IEEE.

4. Tadvi, S., et al. *Personal health records integrated using Android based health care system*. in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. 2017. IEEE.

5. Omotosho, A. and J. Emuoyibofarhe, *A criticism of the current security, privacy and accountability issues in electronic health records.* arXiv preprint arXiv:1501.07865, 2015.

6. West, J.H., et al., *There's an app for that: content analysis of paid health and fitness apps.* Journal of medical Internet research, 2012. **14**(3): p. e72.

7. Health, I. *2015 Annual Report*. 2015; Available from:

http://www.annualreports.com/Company/ims-health. (Last viewed: May 2020)

8. Alksnin, O., *Is e-health a disruptive innovation within a healthcare deliver?* 2016.

9. Ghani, M.A., et al., *Electronic health records approaches and challenges: a comparison between Malaysia and four East Asian countries.* International Journal of Electronic Healthcare, 2008. **4**(1): p. 78-104.

10. Leung, L. and C. Chen, *E-health/m-health adoption and lifestyle improvements: Exploring the roles of technology readiness, the expectation-confirmation model, and health-related information activities.* Telecommunications Policy, 2019. **43**(6): p. 563-575.

11. Ford, E., et al., *Our data, our society, our health: A vision for inclusive and transparent health data science in the United Kingdom and beyond.* Learning Health Systems, 2019: p. e10191.

12. Liu, F., Y. Li, and X. Ju, *Exploring Patients' Consultation Behaviors in the Online Health Community: The Role of Disease Risk.* Telemedicine and e-Health, 2019. **25**(3): p. 213-220.

13. Joseph, B.K., *Blockchain for Open Data–Exploring Conceptual Underpinnings and Practice*, in *Governance Models for Creating Public Value in Open Data Initiatives*. 2019, Springer. p. 161-175.

14. Li, Y., et al. *Blockchain technology in business organizations: A scoping review*. in *Proceedings of the 51st Hawaii International Conference on System Sciences*. 2018.

15. Rizzo, P., *World Economic Forum Survey Projects Blockchain 'Tipping Point' by 2023*. 2015.

16. Abeyratne, S.A. and R.P. Monfared, *Blockchain ready manufacturing supply chain using distributed ledger.* 2016.

17. Morabito, V., *The Future of Digital Business Innovation*. 2016: Springer.

18. Urquhart, A., *The inefficiency of Bitcoin.* Economics Letters, 2016. **148**: p. 80-82.

19. Bambrough, B., *Forget China—Is This The Real Reason Bitcoin, Ethereum, Litecoin, And Ripple's XRP Bounced?* 2019.

20. Mougayar, W., *The business blockchain: promise, practice, and application of the next Internet technology*. 2016: John Wiley & Sons.

21. Chitchyan, R. and J. Murkin, *Review of blockchain technology and its expectations: Case of the energy sector.* arXiv preprint arXiv:1803.03567, 2018.

22. Hughes, L., et al., *Blockchain research, practice and policy: Applications, benefits, limitations, emerging research themes and research agenda.* International Journal of Information Management, 2019. **49**: p. 114-129.

23. Sharma, U. *Blockchain in healthcare: Patient benefits and more*. 2017; Available from: https://www.ibm.com/blogs/blockchain/2017/10/blockchain-in-healthcare-patient-benefits-and-more/. (Last viewed: January 2020)

24. Nakamoto, S., *Bitcoin: A peer-to-peer electronic cash system.* 2008.

25. Miraz, M.H. and M. Ali, *Applications of blockchain technology beyond cryptocurrency.* arXiv preprint arXiv:1801.03528, 2018.

26. LaFever, M.G., T.N. Myerson, and S. Mason, *Systems and methods for enforcing centralized privacy controls in de-centralized systems*. 2018,

Google Patents.

27. Atzori, M., *Blockchain technology and decentralized governance: Is the state still necessary?* Available at SSRN 2709713, 2015.

28. Swan, M. and P. De Filippi, *Toward a philosophy of blockchain: A Symposium: Introduction.* Metaphilosophy, 2017. **48**(5): p. 603-619.

29. Lamport, L., R. Shostak, and M. Pease, *The Byzantine generals problem.* ACM Transactions on Programming Languages and Systems (TOPLAS), 1982. **4**(3): p. 382-401.

30. Hien, D.T.T., D.H. Hien, and V.-H. Pham. *A survey on opportunities and challenges of Blockchain technology adoption for revolutionary innovation.* in *Proceedings of the Ninth International Symposium on Information and Communication Technology*. 2018. ACM.

31. Casey, M.J. and P. Vigna, *The Truth Machine: The Blockchain and the Future of Everything*. 2018: St. Martin's Press.

32. Lennon, M.M. and D. Folkinshteyn, *From Bit Valley to Bitcoin: The NASDAQ Odyssey.* Global Journal of Business Research, 2017. **11**(1): p. 85-103.

33. Zhao, C. and X. Meng. *Research on Innovation and Development of Blockchain Technology in Financial Field.* in *2019 International Conference on Pedagogy, Communication and Sociology (ICPCS 2019)*. 2019. Atlantis Press.

34. Parkin, A. and R. Prescott, *Distributed ledger technology: beyond the hype.* Journal of Digital Banking, 2017. **2**(2): p. 102-109.

35. Kokina, J., R. Mancha, and D. Pachamanova, *Blockchain: Emergent industry adoption and implications for accounting.* Journal of Emerging

Technologies in Accounting, 2017. **14**(2): p. 91-100.

36. Crosby, M., et al., *Blockchain technology: Beyond bitcoin.* Applied Innovation, 2016. **2**(6-10): p. 71.

37. Fon Mathuros, H.o.M., World Economic Forum, *World Economic Forum Technology Governance Network Expands to more than 100 Organizations, Five G7 Nations*. 2019.

38. Raskin, M. and D. Yermack, *Digital currencies, decentralized ledgers and the future of central banking*, in *Research Handbook on Central Banking*. 2018, Edward Elgar Publishing.

39. Carson, B., et al., *Blockchain beyond the hype: What is the strategic business value.* McKinsey & Company, 2018.

40. Bergquist, J., et al. *On the design of communication and transaction anonymity in blockchain-based transactive microgrids*. in *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*. 2017. ACM.

41. DeCusatis, C. and K. Lotay. *Secure, decentralized energy resource management using the ethereum blockchain*. in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. 2018. IEEE.

42. Leonhard, R., *Developing renewable energy credits as cryptocurrency on ethereum's blockchain.* Available at SSRN 2885335, 2016.

43. Mengelkamp, E., et al., *A blockchain-based smart grid: towards sustainable local energy markets.* Computer Science-Research and Development, 2018. **33**(1-2): p. 207-214.

44. Kshetri, N., *Blockchain's roles in strengthening cybersecurity and protecting privacy.* Telecommunications Policy, 2017. **41**(10): p. 1027-1038.

45. Kushch, S. and F. Prieto Castrillo, *A review of the applications of the Block-chain technology in smart devices and dis-tributed renewable energy grids*. 2017.

46. Chedrawi, C. and P. Howayeck, *The role of Blockchain Technology in Military Strategy formulation, a resource based view on capabilities.*

47. Hsu, B., *The Role of Cybersecurity as a Guide in Protecting Election Integrity on Social Media Platforms*. 2019.

48. Chiang, C.-W., *Blockchain for Trustful Collaborations Between Immigrants, Citizens and Governments*. 2018, West Virginia University.

49. Wei, L., et al., *Enabling Distributed and Trusted IoT Systems with Blockchain Technology.* IEEE Blockchain Technical Briefs, 2019.

50. Maiti, A., et al., *Estimating Service Quality in Industrial Internet-of-Things Monitoring Applications with Blockchain.* IEEE Access, 2019.

51. Blobel, B. and P. Pharow, *Archetypes and the EHR.* Advanced Health Telematics and Telemedicine: the Magdeburg Expert Summit Textbook, 2003. **96**: p. 238.

52. Zhang, Y., et al., *Remote mobile health monitoring system based on smart phone and browser/server structure.* Journal of healthcare engineering, 2015. **6**(4): p. 717-738.

53. Asri, S. and B. Pranggono, *Impact of distributed denial-of-service attack on advanced metering infrastructure.* Wireless Personal

Communications, 2015. **83**(3): p. 2211-2223.

54. Ashraf, J. and S. Latif. *Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques*. in *2014 National Software Engineering Conference*. 2014. IEEE.

55. Yi, J. *The Network Security Analysis System Design Based on B/S Structure: An Approach Research*. in *4th International Conference on Computer, Mechatronics, Control and Electronic Engineering*. 2015. Atlantis Press.

56. Hughbanks, A.M.F., *Cybersecurity Within the Healthcare Industry and ElectronicHealth Records*. 2018, Utica College.

57. Aras, S.T. and V. Kulkarni, *Blockchain and Its Applications–A Detailed Survey.* International Journal of Computer Applications, 2017. **180**(3): p. 29-35.

58. Baliga, A., *Understanding blockchain consensus models*, in *Persistent*. 2017.

59. Barrows Jr, R.C. and P.D. Clayton, *Privacy, confidentiality, and electronic medical records.* Journal of the American medical informatics association, 1996. **3**(2): p. 139-148.

60. Yan, Q., *Exploration of Block Chain Technology in Data Security Field.*

61. Wood, G., *Ethereum: A secure decentralised generalised transaction ledger.* Ethereum project yellow paper, 2014. **151**(2014): p. 1-32.

62. Wikipedia. *Blockchain*. 2019; Available from: https://en.wikipedia.org/wiki/Blockchain. (Last viewed: February 2020)

63. Szydlo, M. *Merkle tree traversal in log space and time*. in *International*

Conference on the Theory and Applications of Cryptographic Techniques. 2004. Springer.

64. Roshdy, R., M. Fouad, and M. Aboul-Dahab, *Design And Implementation A New Security Hash Algorithm Based On Md5 And Sha-256.* International Journal of Engineering Sciences & Emerging Technologies, 2013. **6**(1): p. 29-36.

65. Zheng, Z., et al. *An overview of blockchain technology: Architecture, consensus, and future trends*. in *2017 IEEE International Congress on Big Data (BigData Congress)*. 2017. IEEE.

66. Yli-Huumo, J., et al., *Where is current research on blockchain technology?—a systematic review.* PloS one, 2016. **11**(10): p. e0163477.

67. Pilkington, M., *11 Blockchain technology: principles and applications.* Research handbook on digital transformations, 2016. **225**.

68. Fukumitsu, M., et al. *A proposal of a secure P2P-type storage scheme by using the secret sharing and the blockchain*. in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*. 2017. IEEE.

69. Arimilli, L.B., R.K. Arimilli, and R.S. Blackmore, *User level message broadcast mechanism in distributed computing environment*. 2012, Google Patents.

70. Gervais, A., et al. *On the security and performance of proof of work blockchains*. in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016. ACM.

71. Vasin, P., *Blackcoin's proof-of-stake protocol v2.* URL: https://blackcoin. co/blackcoin-pos-protocol-v2-whitepaper. pdf, 2014. **71**.

72. Schwartz, D., N. Youngs, and A. Britto, *The ripple protocol consensus algorithm.* Ripple Labs Inc White Paper, 2014. **5**: p. 8.

73. Castro, M. and B. Liskov. *Practical Byzantine fault tolerance*. in *OSDI*. 1999.

74. Larimer, D., *Delegated proof-of-stake (dpos).* Bitshare whitepaper, 2014.

75. Comben, C. *Delegated Byzantine Fault Tolerance (dBFT) explained*. 2019; Available from: https://coinrivet.com/delegated-byzantine-fault-tolerance-dbft-explained/. (Last viewed: February 2020)

76. NEO. *An Open Network for the Smart Economy*. 2019; Available from: https://neo.org/. (Last viewed: April 2020)

77. Turner, L., *"Medical tourism" and the global marketplace in health services: US patients, international hospitals, and the search for affordable health care.* International Journal of Health Services, 2010. **40**(3): p. 443-467.

78. ShenTu, Q. and J. Yu, *A blind-mixing scheme for Bitcoin based on an elliptic curve cryptography blind digital signature algorithm.* arXiv preprint arXiv:1510.05833, 2015.

79. Williams, H., *A modification of the RSA public-key encryption procedure (Corresp.).* IEEE Transactions on Information Theory, 1980. **26**(6): p. 726-729.

80. Francisco, K. and D. Swanson, *The supply chain has no clothes: Technology adoption of blockchain for supply chain transparency.* Logistics, 2018. **2**(1): p. 2.

81. Norman, D.A. and S.W. Draper, *User centered system design: New*

perspectives on human-computer interaction. 1986: CRC Press.

82. O'Neill, A., C. Peikert, and B. Waters. *Bi-deniable public-key encryption*. in *Annual Cryptology Conference*. 2011. Springer.

83. Burrows, J.H., *Secure hash standard*. 1995, DEPARTMENT OF COMMERCE WASHINGTON DC.

84. Eastlake, D. and P. Jones, *US secure hash algorithm 1 (SHA1)*. 2001, RFC 3174, September.

85. Lin, I.-C. and T.-C. Liao, *A Survey of Blockchain Security Issues and Challenges.* IJ Network Security, 2017. **19**(5): p. 653-659.

86. Pongnumkul, S., C. Siripanpornchana, and S. Thajchayapong. *Performance analysis of private blockchain platforms in varying workloads*. in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. 2017. IEEE.

87. Dhoolia, P., et al., *Process management using representation state transfer architecture*. 2015, Google Patents.

88. Sandhu, R.S., et al., *Role-based access control models.* Computer, 1996. **29**(2): p. 38-47.

89. Douceur, J.R. *The sybil attack*. in *International workshop on peer-to-peer systems*. 2002. Springer.

90. Elrom, E., *NEO Blockchain and Smart Contracts*, in *The Blockchain Developer*. 2019, Springer. p. 257-298.

91. Wang, W., et al., *A survey on consensus mechanisms and mining management in blockchain networks.* arXiv preprint arXiv:1805.02707, 2018: p. 1-33.

92. Xu, X., et al. *A taxonomy of blockchain-based systems for architecture design*. in *2017 IEEE International Conference on Software Architecture (ICSA)*. 2017. IEEE.

93. Li, X., et al., *A survey on the security of blockchain systems.* Future Generation Computer Systems, 2017.

94. Kendall, K.E., et al., *Systems analysis and design*. Vol. 4. 1992: Prentice Hall Englewood Cliffs, NJ.

95. Widenius, M., D. Axmark, and K. Arno, *MySQL reference manual: documentation from the source*. 2002: " O'Reilly Media, Inc.".

96. Burnette, E., *Eclipse IDE Pocket Guide: Using the Full-Featured IDE*. 2005: " O'Reilly Media, Inc.".

97. Xie, M., Y.-S. Dai, and K.-L. Poh, *Computing system reliability: models and analysis*. 2004: Springer Science & Business Media.

98. Ichikawa, D., M. Kashiyama, and T. Ueno, *Tamper-resistant mobile health using blockchain technology.* JMIR mHealth and uHealth, 2017. **5**(7): p. e111.

99. Malomo, O.O., *Cybersecurity through a Blockchain Enabled Federated Cloud Framework*. 2018, Howard University.

100. Gong, L., et al. *Going beyond the sandbox: An overview of the new security architecture in the Java development kit 1.2*. in *USENIX Symposium on Internet Technologies and Systems*. 1997.

101. Shteyn, Y., *Distributed storage on a P2P network architecture*. 2002, Google Patents.

102. Jeffrey, A. and J. Rathke. *Java Jr: Fully abstract trace semantics for a*

core Java language. in *European symposium on programming*. 2005. Springer.

103. Aronson, L., *HTML 3 manual of style*. 1995: Ziff-Davis Publishing Co.

104. Knaus, W.A., et al., *APACHE II: a severity of disease classification system.* Critical care medicine, 1985. **13**(10): p. 818-829.

105. Anderson, C., *Docker [software engineering]*. IEEE Software, 2015. **32**(3): p. 102-c3.

106. Bui, T., *Analysis of docker security.* arXiv preprint arXiv:1501.02967, 2015.

107. Debrabander, B. and A. Edström, *Successful information system development projects.* Management Science, 1977. **24**(2): p. 191-199.

Publication

Lingyu Xu; Antoine Bagula; Omowunmi Isafiade; Kun Ma; Tapiwa Chiwewe. Design of a Credible Blockchain-based E-health Records (CB-EHRs) Platform has been published by ITU K-2019.