

分类号: TP311.5

单位代码: 10335

密 级: 无

学号: 21860438

浙江大学



工程硕士专业学位论文

中文论文题目 : 基于区块链的电子病历共享系统
的设计与实现

英文论文题目: Design and Implementation of
Electronic Medical Record Sharing System Based
on Blockchain

申请人姓名: 李亚辉

指导教师: 黄启春

合作导师: 胡斌

专业学位类别: 工程硕士

专业学位领域: 计算机技术

所在学院: 工程师学院

论文提交日期 2021 年 9 月 14 日

基于区块链的电子病历共享系统的设计与实现



论文作者签名: 李五辉

指导教师签名: _____

论文评阅人 1: 隐名评阅 1

评阅人 2: 隐名评阅 2

评阅人 3: 隐名评阅 3

评阅人 4: 隐名评阅

评阅人 5: 隐名评阅

答辩委员会主席: 张微\教授级高工\浙江大学软件学院

委员 1: 张宣\副教授\浙江大学软件学院

委员 2: 贝毅君\副研究员\浙江大学软件学院

委员 3: 梁秀波\副研究员\浙江大学软件学院

委员 4: 包贤晨\高级工程师\宁波工业互联网研究院

委员 5: _____

答辩日期: 2021 年 9 月 5 日

Design and Implementation of Electronic Medical
Record Sharing System Based on Blockchain



Author's signature: _____

Supervisor's signature: _____

Thesis reviewer 1: _____

Thesis reviewer 2: _____

Thesis reviewer 3: _____

Thesis reviewer 4: _____

Thesis reviewer 5: _____

Chair: _____

(Committee of oral defence)

Committeeman 1: _____

Committeeman 2: _____

Committeeman 3: _____

Committeeman 4: _____

Committeeman 5: _____

Date of oral defence: _____

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得浙江
大
学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：李五辉 签字日期：2021 年 9 月 14 日

学位论文版权使用授权书

本学位论文作者完全了解浙江
大
学有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权浙江
大
学可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

(保密的学位论文在解密后适用本授权书)

学位论文作者签名：李五辉 导师签名：
签字日期：2021 年 9 月 14 日 签字日期：2021 年 9 月 15 日

摘要

随着我国社会经济水平的持续发展，人们对于医疗服务水平的要求也在不断的提高。然而当下医疗机构之间信息系统不互通造成的数据孤岛现状严重阻碍了医疗服务水平和效率的进一步提升。于此同时，医疗数据因为其高价值，易于被攻击的特点而导致的数据泄露事件时有发生。

本文针对当下的现状，提出了一个基于区块链的电子病历共享系统，以期解决电子病历共享难和医疗数据泄露等问题。本系统的核心主要采用了 Hyperledger Fabric 联盟链框架并结合 IPFS (Inter Planetary File System) 星际存储文件系统，形成了“链上索引，链下存储”的存储模式。该模式在链上存储电子病历的索引信息并通过用户公钥加密防止泄露，同时确保了病历索引去中心化存储、不可篡改和可追溯等优势。另外，该模型又利用 IPFS 分布式存储技术，通过在链下存储病历密文，有效的降低了区块链系统的通信和存储压力。于此同此，用户通过代理重加密算法授予其他用户的解密权限，只有被授权的用户才可以解密病历密文，确保了病历记录安全可信的共享。

本文根据系统的具体实现给出系统的测试方案，并对系统进行了功能测试和系统测试，最后对测试结果和系统的安全性进行了分析，证实系统达到预期的效果。

关键词： 区块链，Hyperledger Fabric，代理重加密，电子病历共享，IPFS

Abstract

With the continuous development of our country's economic level, people are also pursuing better medical services. However, the data islands between medical institutions are an obstacle to the further development of medical services. At the same time, data leakage incidents occur from time to time because of the high value and vulnerability of medical data.

In view of the current situation, this thesis proposes a blockchain-based Medical data sharing system to solve the above problems. The core of this system mainly adopts the Hyperledger Fabric alliance chain framework and combines with the IPFS interplanetary storage file system to form a storage mode of "index on chain, storage off chain". This model stores the index information of electronic medical records on the blockchain and encrypts them with the user's public key. Because it is stored on the chain, the index has the characteristics of secure decentralized storage, tamper-proof and traceability, etc. In addition, the model uses IPFS distributed storage technology to effectively reduce the communication and storage pressure of the blockchain system by storing the ciphertext of the medical records under the chain. At the same time, in order to realize the secure sharing of electronic medical records, the user grants the decryption authority to other users through the proxy re-encryption algorithm, and only authorized users can decrypt the ciphertext of the medical record to ensure the safe and credible sharing of the medical record.

According to the concrete realization of the system, this thesis gives the system test plan, and carries on the function test and the system test to the system, and finally analyzes the test results and the safety of the system to confirm that the system achieves the expected effect.

Keyword: Blockchain, Hyperledger Fabric, Proxy re-encryption, IPFS, Medical Record Sharing

目录

摘要.....	i
Abstract.....	ii
第1章 绪论.....	1
1.1 研究背景及意义.....	1
1.2 国内外研究现状.....	2
1.2.1 国外研究现状.....	3
1.2.2 国内研究现状.....	4
1.3 本文主要研究内容.....	5
1.4 本文的章节安排.....	6
第二章 相关技术与理论介绍.....	8
2.1 区块链基础知识.....	8
2.1.1 区块链简介.....	8
2.1.2 区块链数据结构.....	9
2.2 Hyperledger Fabric 框架.....	10
2.2.1 Hyperledger Fabric 系统架构.....	11
2.2.2 Hyperledger Fabric 交易流程.....	12
2.3 IPFS 星际文件系统.....	13
2.4 密码学技术.....	14
2.4.1 哈希算法.....	14
2.4.2 对称与非对称加密算法.....	15
2.4.3 代理重加密.....	15
2.5 本章小结.....	16
第三章 系统的需求分析.....	17
3.1 系统总体需求概述.....	17
3.2 功能性需求分析.....	17
3.3 非功能需求分析.....	21
3.4 本章小结	21
第四章 系统的总体设计.....	23
4.1 系统病历共享流程设计.....	23
4.2 系统软件架构设计	24
4.3 系统功能模块概要设计	26
4.4 系统数据模型设计	27
4.3.1 系统数据概念模型设计.....	27
4.3.2 系统数据物理模型设计.....	28
4.5 本章小结.....	32
第五章 系统的详细设计与实现.....	33
5.1 系统功能模块的详细设计与实现	33
5.1.1 用户信息管理模块的详细设计与实现.....	33
5.1.2 医疗业务功能模块的详细设计与实现.....	36
5.1.3 系统管理模块的详细设计与实现.....	46
5.2 智能合约的详细设计与实现	49

5.3 区块链网络搭建	53
5.4 IPFS 私有化集群搭建	55
5.5 本章小结	57
第六章 系统测试.....	58
6.1 测试环境.....	58
6.2 功能测试.....	58
6.2.1 用户信息管理模块测试.....	59
6.2.2 医疗业务功能模块测试.....	61
6.2.3 系统管理功能测试测试.....	65
6.3 性能测试.....	67
6.4 安全性分析与方案对比.....	70
6.5 本章小结.....	72
第七章 总结与展望.....	73
7.1 总结.....	73
7.2 展望.....	74
参考文献.....	75
作者简历.....	80
致谢.....	81

图目录

图 2.1 区块链发展示意图.....	8
图 2.2 区块链的区块结构图.....	10
图 2.3 Hyperledger Fabric 系统架构图.....	11
图 2.4 Fabric 交易流程示意图.....	13
图 2.5 Hash 函数示意图.....	14
图 2.6 代理重加密示意图.....	16
图 3.1 患者用户用例图.....	18
图 3.2 医生用户用例图.....	19
图 3.3 系统管理员用例图.....	20
图 4.1 病历共享模型流程图.....	23
图 4.2 系统逻辑架构图.....	24
图 4.3 系统物理部署架构图.....	25
图 4.4 系统功能模块图.....	26
图 4.5 系统 E-R 图.....	27
图 5.1 系统登录注册流程图.....	43
图 5.2 用户注册时序图.....	34
图 5.3 个人信息查看与修改流程图.....	35
图 5.4 系统用户模块核心类图.....	36
图 5.5 挂号就诊功能流程图.....	37
图 5.6 用户挂号就诊核心类图.....	38
图 5.7 病历添加与授权流程图.....	38
图 5.8 医生添加病历时序图.....	39
图 5.9 医生解密病历时序图.....	40
图 5.10 患者授权与取消流程图.....	40
图 5.11 患者授权时序图.....	41
图 5.12 病历共享核心类图.....	42
图 5.13 重加密加密算法图.....	42
图 5.14 重加密密钥生成算法图.....	43
图 5.15 重加密密钥加密算法图.....	43
图 5.16 重加密解密密钥生成算法图.....	44
图 5.17 修改药品信息流程图.....	45
图 5.18 修改药品时序图.....	45
图 5.19 用户状态管理时序图.....	46
图 5.20 用户状态管理时序图.....	47
图 5.21 更新通道配置流程图.....	48
图 5.22 更新通道配置时序图.....	48
图 5.23 智能合约设计类图.....	49
图 5.24 添加病历索引代码示意图.....	50
图 5.25 查询病历索引代码示意图.....	51
图 5.26 执行代理重加密代码示意图.....	51

图 5.27 查询代理重加密结果示意图.....	52
图 5.28 链接区块链网络伪代码示意图.....	53
图 5.29 生成证书配置文件图.....	54
图 5.30 通道配置模板和配置文件示意图.....	54
图 5.31 区块链网络启动示意图.....	55
图 5.32 IPFS 节点集群示意图.....	55
图 5.33 swarm.key 示意图.....	56
图 5.34 IPFS 节点启动示意图.....	57
图 6.1 系统登录示意图.....	60
图 6.2 系统注册页面示意图.....	60
图 6.3 查看个人信息示意图.....	61
图 6.4 系统挂号示意图.....	62
图 6.5 医生申请查看病历示意图.....	63
图 6.6 患者授权医生示意图.....	63
图 6.7 医生添加病历示意图.....	64
图 6.8 管理病历授权示意图.....	64
图 6.9 查看申请病历示意图.....	65
图 6.10 查询药品信息示意图.....	65
图 6.11 管理员管理系统账户示意图.....	67
图 6.12 管理员更改通道配置示意图.....	67
图 6.13 链码查询性能结果图.....	68
图 6.14 链码更新性能结果图.....	69
图 6.15 代理重加密示例结果图.....	71

表目录

表 2.1 加密算法分类表.....	15
表 4.1 系统账户信息表.....	28
表 4.2 患者个人信息表.....	29
表 4.3 医生信息字段设计表.....	29
表 4.4 病历信息字段设计表.....	30
表 4.5 病历索引表.....	31
表 4.6 药品信息字段设计表.....	31
表 6.1 系统测试环境表.....	58
表 6.2 用户管理模块测试用例表.....	59
表 6.3 医疗业务功能用例测试表.....	61
表 6.4 系统管理模块测试用例表.....	66
表 6.5 更新和查询类型对应函数说明表.....	68
表 6.6 更新资源使用情况说明表.....	70
表 6.7 查询测试资源使用情况说明表.....	70
表 6.8 存储方案对比表.....	72

第一章 绪论

本章在开篇阐述了本文工作的研究背景和意义，之后对涉及到本文相关领域研究内容的现状做了介绍。最后，本章在这个基础之上总结和归纳了本文主要的研究工作以及本文的章节脉络结构。

1.1 研究背景及意义

伴随着我国的经济水平不断提升，人们的生活水平也得到了改善。于此同时，人们对医疗资源服务的需求也在不断的增长，但是我国目前医疗资源总体来说还是比较的匮乏并且分布也不平衡^[1]。因此，患者进行异地就诊并在不同医疗机构进行问诊的情况较为常见^[2]。但是由于在信息化发展的初期，不同的医疗机构使用的是各自的医疗信息管理系统，这些系统之间往往存在异构性，导致了病人的医疗病历数据只能在同一个医疗机构内部进行共享和流转。这样的现状造成了患者医疗电子病历数据的分散存储在不同医疗机构中，客观上形成了医疗机构信息系统之间的“数据孤岛”现象^[3]。在当前的就医情境下，患者能掌握的只有经过复印的纸质材料以及手写的病历本，不仅不易携带而且相对容易发生磨损和丢失。如果患者进行就医时无法提供精准的以往的诊治信息，那么当前进行诊断的医生往往无法快速而准确的把握患者的病情及其当下的进展，进而导致难以对患者进行快速精准的治疗。除此之外，这种情况还可能导致患者做重复的检查^[4]，这不仅对患者的精力和金钱是一种浪费，而且也挤占了医院原本就为数不多的医疗资源，造成了患者和医疗机构双输的局面。

不仅如此，医疗信息本身具有极其特殊的敏感性和重要性，近些年来已经发生了多起医疗数据泄露的事件。例如：挪威的一家医疗机构的计算机信息管理系统遭到了黑客的攻击，据事后统计分析结果表示超过一半的公民数据遭到泄露^[5]；美国的一家名为 HealthNow Networks 的公司由于软件开发人员的失误泄露了超过 90 万份的医疗健康数据^[6]。以上这些信息的泄露极易导致患者遭遇各种医疗保健信息的骚扰以及诈骗等行为。因此，面对医疗数据中存在巨大经济价值，我

们在保证医疗数据共享的同时也要着重保护医疗数据的隐私，解决数据共享和隐私保护的两难困境。

区块链技术是当下热度比较高的计算机技术之一，在国务院已经发布的关于十三五规划文件中已经明确指出要深入挖掘区块链技术在社会领域的应用，引领有关行业的进一步发展，并将其作为国家信息技术发展的重点^[7]。区块链技术的特点是通过加密技术与共识机制对区块中的数据进行集体维护，因此确保了区块中的数据具有一致性、难篡改等特性^[8]。另外，智能合约扩展了区块链技术的应用领域^[9]。因此，区块链技术优势使得为解决医疗领域中的数据共享问题提供了新的思路。本文基于区块链技术，提出了一个基于区块链的电子病历共享系统。该系统通过多个医疗机构组成联盟来共同维护区块链系统，通过这样的方式来确保数据在不同机构中的互联互通。与此同时，为了减轻区块链系统的通信和存储负担，本文采用星际文件系统（Inter Planetary File System）作为补充，形成了“链上索引，链下存储”的数据存储模式。最后在整个过程中采用代理重加密技术实现了加密数据在密文状态下的解密密钥转换，保护了用户医疗数据隐私，实现了安全可信的电子病历共享。

综上所述，采取区块链技术和密码学技术的结合来解决医疗数据目前存在的数据孤岛问题与医疗数据隐私保护的问题，可以有效提高当下的医疗机构的服务水平和医疗资源利用率并且保护患者的医疗数据隐私，因此具有一定的研究意义和应用价值。

1.2 国内外研究现状

区块链技术作为一个创新性的技术具有广阔的发展空间令全世界的诸多国家十分的重视区块链技术^[10]。目前很多国家都希望通过推动区块链技术的发展为当下的诸多领域提供创新的动力并带动相关产业的进一步发展。而应用区块链解决医疗领域面对的数据孤岛难题和隐私保护问题是目前国内外学者的研究热点之一。

1.2.1 国外研究现状

为了保护医疗数据的安全和隐私问题，国外的诸多学者在该领域结合区块链提出了多种保护医疗数据安全和实现医疗数据共享的方案。例如：文献^[11]提出了一种基于区块链的医疗保健数据的管理模型，主要用于管理和共享癌症患者的电子病历数据。该模型采用链上存元数据，链下存储真实数据来确保医疗数据的完整性同时给与患者权限来指定访问对象，而执行主体交由链码强制执行，实现细粒度访问控制的可能。该方案的特点是模型采用了拜占庭共识算法，在提高用户数量方面有着出色的表现。文献^[12]为了保护电子病历的隐私安全，基于区块链提出了一个轻量级的隐私保护机制，在该机制中主要采用了交错编码器来加密原始的电子病历数据，隐藏电子病历的敏感信息，然后采用了轻量级的阈值数据共享方案将电子病历映射到不同的区块链份额上去。这些存储数据份额生成索引连接在一起形成区块链，授权用户可以根据这些份额来恢复原始数据。该方案的主要优势在于充分利用了区块链的存储空间。

文献^[13]为了保护在公有链上的数据隐私，在以太坊的基础上设计了一个隐私保护的方案。该方案使用对称加密技术和以太坊提供的智能合约在数据提供者和数据消费者之间建立了一个访问控制列表。只有通过这个访问列表的授权用户，才能向数据提供者请求安全密钥访问其共享数据，该方案最后在以太坊的测试网络上测试了 gas 消耗和性能指标并有着不错的表现，基本可以满足日常使用。文献^[14]针对医疗数据的隐私性和敏感性，为了保护电子健康档案的安全与隐私，在数据共享协议上采取了无证书的基于身份和类型的加密方式，同时使用 DPOS 共识公式算法选取代理节点，以此来保证整个系统可以抵抗身份伪装或者重放攻击。文章在最后证明了该方案的安全与高效。文献^[15]深入调查当前医院中健康信息的交换流程之后，基于区块链建立了保护患者个人健康数据安全和隐私的数据共享模型。该模型提供以患者为中心的健康信息数据交换并通过智能合约建立临床医生访问列表来实现权限控制。因此该模型有着较好的访问控制粒度，经过大量的模拟测试证明了该方案具有较好的稳定性以及安全性。文献^[16]主要通过建立基于以太坊的私有区块链系统，该系统设计了医疗 IOT 设备与区块链系统的通信协议，解决了患者监视系统中的安全问题和漏洞。该方案直接和 IOT 设备进行通信确保了系统数据源头上的真实性。

除此以外，为了提升区块链系统中的数据安全和系统性能，文献^[17]引入了分布式私有计算方案，主要通过零知识证明和递归证明组合隐藏了所有关于离线运算的信息，任何人都可以在恒定的时间内验证交易且生成的可证明交易时间也较短。该方案极大的保护了用户资产数据的隐私。文献^[18]利用了比特币节点上的端到端之间的延迟为上限，提出了一种新的区块确认机制。该机制特别适合于低延迟且带许可的区块链系统，极大的提高了区块链系统的吞吐量。文献^[19]为了提高区块链的共识效率，通过在仅保证消息直接相对位置的一致性，提出了一个高性能的区块链系统，该系统的吞吐量可达 43kTPS，明显高于目前的以太坊等区块链系统。文献^[20]将公有链和私有链结合，通过一个去中心化的网关架构实现了公有链和私有链的互操作，打破了区块链数据流通的边界，并证明了该方案的可行性。最后通过实现证实了该方案具有较好的性能。文献^[21]提出一个可有凭证办法机构颁发选择性披露凭证的方案，该方案支持公有和私有属性的随机化和不可链接的属性披露，有效的保证了数据的安全型。文献^[22]提出了一种基于隐私保护权益证明的区块链协议，并对该协议进行安全建模和分析。该文献通过 SNARK 和密钥的私有向前安全加密技术证明了该协议对于自适应的攻击是安全的。

1.2.2 国内研究现状

近些年，我国政府也高度关注医疗领域数据共享和隐私问题，并下发了有关的指导文件^[23]。

因此，我国研究人员在利用区块链技术保护我国公民医疗记录隐私方面也做出许多的贡献。文献^[24]针对医疗领域中的跨域记录分享与溯源中存在的问题，在以太坊上设计了对应的智能合约，并结合安全性高、多方协作简单的非对称加密技术提升了身份认证的效率并过滤非法用户。该方案的优点在于降低了身份验证时的开销。文献^[25]根据分布式密钥生成技术和身份代理重加密技术设计一个数据安全共享协议。选取执行代理重加密节点时采用 DPoS 算法，达成数据共享目标，最后采用了区块链网络和分布式数据库技术共同存储加密数据和其相关的访问策略。该方案去中心化程度高，可以有效的抵抗身份伪装和重放攻击。

在文献^[26]中，薛腾飞等人设计了一个医疗数据的共享模型，该模型通过区块链的数据结构保证医疗数据的完整性，并依赖医疗机构之间的服务器集群使用代

理重加密的方式保证数据的安全性，最后提供给进行代理中加密的机构服务信用积分激励。在文献^[27]中，周正强等人通过设计数据存储的事务结构，将数据通过密文属性加密技术加密保护数据的安全和隐私，最终提出了一种基于联盟链的医疗数据共享模型。该模型通过云存储加密后的数据，联盟链存储数据的元数据保证数据的完整性不易篡改，最后通过事务中的时间设置信息来保证了在时间维度上的访问控制。文献^[28]提出了一种在联盟链环境下的医疗数据共享方案，首先在该系统的共识层采用了改进的拜占庭协议，提升了共识效率。其次，在该协议中选取重点医院或者其他级别较高受信赖的医疗机构作为初始的记账节点，并在应用层使用访问者签名作为身份属性的唯一表示，最后通过签名找到存储数据的块，实现医疗数据的共享。文献^[29]提出了一个双链结构医疗记录存储和共享方案。该方案中利用存储链中存储相关的元数据和摘要信息，并于其他医疗机构共同维护该链，医疗记录数据由用户公钥加密后上传值分布式的云服务器中进行保管。在医疗记录共享链中主要通过智能合约来实现医疗数据的访问请求和访问授权等功能。在医疗记录存储链中存储共享链上的操作记录，形成日志账本便于将来追溯查询。

1.3 本文主要研究内容

本文以医疗行业中各个医疗机构之间医疗数据难以共享和医疗资源隐私保护的两难困境为背景，结合了 Hyperledger Fabric 联盟链技术、IPFS 技术以及代理重加密技术，设计和实现了一个医疗电子病历共享系统。该系统主要解决了患者电子病历在不同医疗机构之间数据共享的难题和电子医疗数据隐私保护问题，不仅有效的提升了医疗服务水平，而且也提高了医疗资源的利用效率。

本文的研究内容主要包括以下内容：

1. 对医疗领域目前存在的数据共享难题和隐私保护问题进行调查，分析医疗数据共享的业务需求以及相关的业务流程，同时研究区块链技术与联盟链 Hyperledger Fabric 项目、星际文件存储系统以及代理重加密技术在医疗领域中的应用。并此基础上总结出系统的总体需求，之后结合系统的不同角色的用例图分析本文系统的功能性需求，并在最后从安全性、可用性等方面对系统提出了非功能性需求。

2. 根据前面的需求分析选取相应技术设计系统的总结架构，并对系统架构进行了进一步解释说明。然后划分系统功能模块，明确各个功能模块的职责。最后对系统进行数据建模分析并给出了数据模型设计的核心内容并对其进行了解释说明。

3. 根据本文设计好的系统架构和功能模块，对每个模块的内容进行了详细的设计与实现。本系统的模块主要包括用户信息管理模块，医疗业务功能模块以及系统管理模块等模块的业务逻辑实现和智能合约的编写和使用等工作。并叙述了如何搭建了区块链网络环境和 IPFS 私有化节点集群。

4. 完成了对基于联盟链的本文系统的各个业务模块的功能测试，设计相应功能用例检验了该系统的可用性，并给出了系统的运行截图作为展示。之后针对系统 Fabric 网络进行性能测试，并分析测试结果。最后，对系统的测试结果和系统的安全性进行了分析。

本文系统的创新点在于：

1. 提出一种区块链和 IPFS 技术的结合的方案，在区块链中存储数据加密索引，而在 IPFS 中存储医疗病历数据，即采用“链上索引，链下存储”的存储模式，该模式减轻了区块链系统中的存储压力和通信开销，提升了系统的存储能力。

2. 采用代理重加密技术，在联盟链环境下通过智能合约执行代理重加密过程，实现了数据加密索引在加密状态下的解密密钥转换，既保护了用户的医疗数据隐私，又完成了用户之间安全可信医疗病历数据共享。

1.4 本文的章节安排

第一章 绪论。本章节首先抛出本论文的研究背景和意义，然后接着是分析了区块链技术各个领域应用的国内外研究现状，之后点出了本文所做的工作的具体内容和创新点，最后是本论文章节安排信息。

第二章 相关技术介绍。本章节介绍系统关键技术。首先是区块链里面的基本原理和核心技术进行了介绍和讨论，为之后系统使用这些知识打下了基础。接着讨论了所开发的系统采用的 Hyperledger Fabric 框架中所涉及到基本知识。例如该框架的系统架构、提交交易的流程、交易流程所涉及到的组件以及账本的组成

等知识。最后介绍了 IPFS（星际文件系统）和相关密码学技术，为之后使用这些技术开发做了铺垫。

第三章 系统的需求分析。本章首先强调了系统的总体需求，确立了系统的总体目标。然后主要通过用户的用例图详细描述了系统内用户应该拥有的功能用例，并对其功能点进行了解释和说明，完成系统的功能性需求分析。最后，在文中对系统的非功能性需求进行了分析。

第四章 系统的总体设计。本章首先叙述了系统的病历共享方案流程设计，然后介绍了系统的架构设计，在架构设计中介绍了各个层级之间包含的功能和各层之间的调用方式。接着对系统内的功能进行了逻辑划分，并展示了各个功能模块的功能。最后设计了系统的数据模型，主要通过概念模型展示数据之间的关系和物理模型展示数据具体的字段设计。

第五章 系统的详细设计与实现。本章叙述系统各个功能模块的详细设计流程和实现细节。首先针对系统的各个模块进行了详细的分析和设计，并分别叙述了每个模块内各个功能的流程图，并通过类图和时序图说明每个功能涉及到的类和方法之间的调用关系。除此之外，还叙述了区块链网络的搭建和 IPFS 私有化节点的启动流程信息。

第六章 系统测试。本章首先明确测试环境，然后对设计的各个模块进行功能测试，验证之前设计的正确性，并展示系统运行图。在性能测试部分主要使用 Caliper 工具还对本文的区块链系统的性能做了测试，并分析了测试的结果。在本章的最后对系统的安全性进行了分析说明。

第七章 结论和展望。本章总结了本论文所作的工作，对本文实现的系统做一个全面的评价，并且根据现有的系统对未来的工作和改进方法进行了展望。

第二章 相关技术与理论介绍

本章介绍的相关理论和技术，主要包括区块链基础知识、Hyperledger Fabric 框架、星际文件系统以及相关密码学技术。

2.1 区块链基础知识

本小节主要介绍区块链和 Hyperledger Fabric 框架的基础知识。为之后业务逻辑如何利用区块链技术和为什么选择使用 Hyperledger Fabric 框架作为区块链技术支撑打下了基础。

2.1.1 区块链简介

2008 年末，一篇文章^[30]横空出世，将区块链技术带入了人们的视野范围之内。在这篇文章中一个化名为中本聪的人创造了一种名为比特币的虚拟货币，区块链进入了 1.0 阶段。随着智能合约的兴起，区块链进入包括 DAPP，容器和可编程金融等特点的 2.0 阶段，在金融领域发挥着重要的作用。最后，随着区块链技术被其他领域不断应用，区块链技术涉及的领域越加广泛，概念边界不断扩展到社会的其他领域，进入到区块链 3.0 阶段^[31]。综上所述，区块链发展阶段如图 2.1 所示

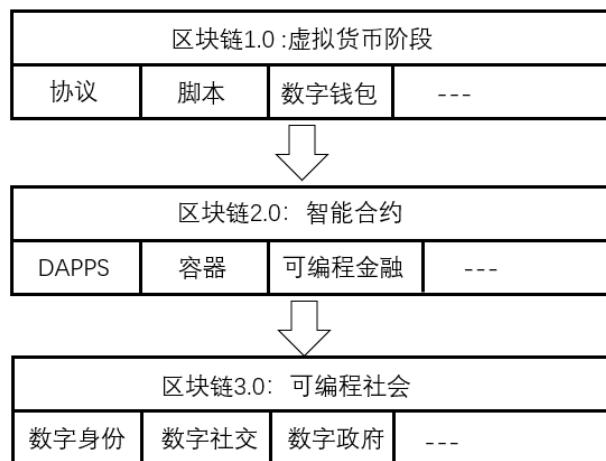


图 2.1 区块链发展示意图

区块链又可以根据准入权限的不同，将其划分成为三种类型^[32]。

第一类是公有链，这种模式的区块链中的任何节点都可以在区块链上进行读写以及数据的验证操作，并且可以根据其共识过程的贡献发放相应的奖励，也就是所谓的激励层。这个区块链模型是完全的去中心化的，监管难度较大。

第二类是联盟链，联盟链主要应用于预定义的多个实体构成的组织之间的成员才有权限对区块链进行操作，需要提供一系列的成员服务。它的共识过程只在这些联盟成员之内形成，特点是部分去中心化的。

第三类是私有链。私有链则是将操作的权限完全收束到一个组织之内，已经失去了中心化的特点，成为了一个中心化的区块链。这种模式的区块链通常只用于一个组织之内的部门之间的数据进行管理和操纵的追溯。

根据上述的三种类型的描述可以得知：公有链是完全去中心化的，但是管理难度也是最高的。私有链完全失去了去中心化的特性，因此应用的场景也不多。联盟链将去中心化特性缩小到了加入联盟的组织之间，但是管理的难度也降低了，在目前的社会环境下应用环境更为广泛。本文即采用联盟链的著名代表Hyperledger Fabric 框架来实现本系统。

2.1.2 区块链数据结构

区块链的数据结构设计是区块链技术诸多特性的基础，也是区块链设计的核心思想体现。如图 2.2 所示为区块结构的一般示意图。

从图中可以看出，对于区块头包含的信息可以划分成为三类。第一类是连接前后区块相关的信息，比如前一个区块的父哈希值。第二类是涉及到挖矿行为，例如：挖矿的难度、区块生成的时间戳和有矿工找到的随机值。最后一类是有关验证数据有效性的信息，目前普遍采用默克尔树根^[33](Merkle Root)，该结构可以有效快速的验证交易信息。

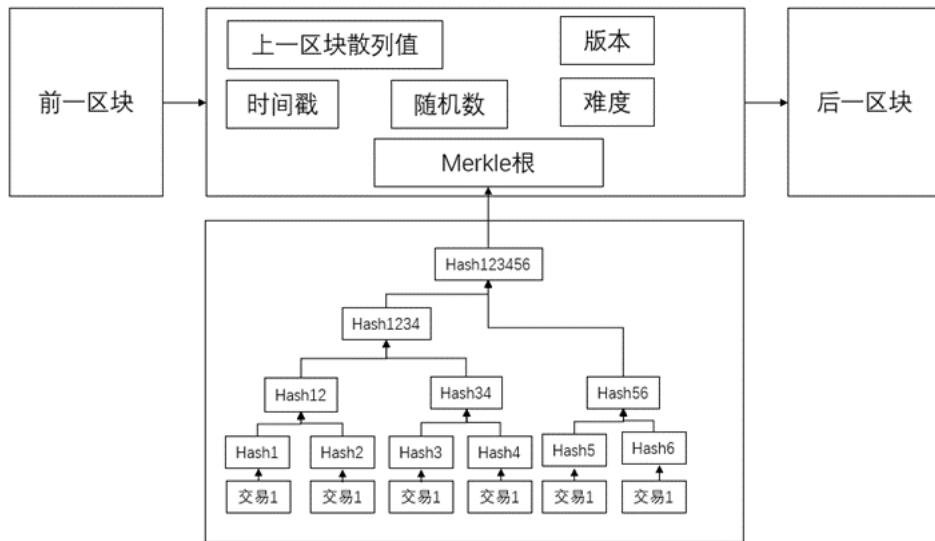


图 2.2 区块链的区块结构图

一个一个区块之间相连就会构成一条链，这条由一个一个区块连接起来的就是本文提到的区块链。随着新的区块的不断产生，节点验证该区块并将其连接到区块链上，不断的扩展这条链，最终实现区块链的增长。如果其中一个数据会导致后面的所有区块数据连接不上，实现了数据的防篡改。

2.2 Hyperledger Fabric 框架

超级账本项目（Hyperledger）是一个开源的协作成果，这个项目旨在促进区块链的跨行业应用，主要面对的企业级的区块链需求场景。Hyperledger 是一个联合项目，其中 Fabric 是其中最重要的子项目，最早由 IBM, Intel 等公司带头开发。它基于 GO 语言实现，是一个典型的联盟链开发平台^[34]。

Fabric 的特点有以下几个方面，首先 Fabric 一个重要的特点就是参与者的有条件的的部分信任，参与者可能是竞争者的关系。其次，它具有高度模块化以及可拔插的设计。这种设计使得 Fabric 具有可配置的架构，支持不同场景下的企业需求，扩大了 Fabric 的应用场景。例如 Fabric 支持对共识协议模块的替换，可以根据不同的场景选取适当的共识协议，适应特定的用例和信任模型。最后 Fabric 还提供使用常见的编程语言来进行智能合约的编写，目前为止，Fabric 支持 Go，

Java, Node.js 这三种常用的编程语言。使用这三种常用的语言意味着大多数企业不需要额外的成本来编写智能合约。

2.2.1 Hyperledger Fabric 系统架构

Fabric 是一个采用联盟链思想实现的分布式账本平台，可以支持企业间建立分布式账本的需求。如图 2.3 所示为 Fabric 的系统架构图，其主要分为四大模块，分别是成员管理服务、区块链服务、智能合约和事件流^[35]。

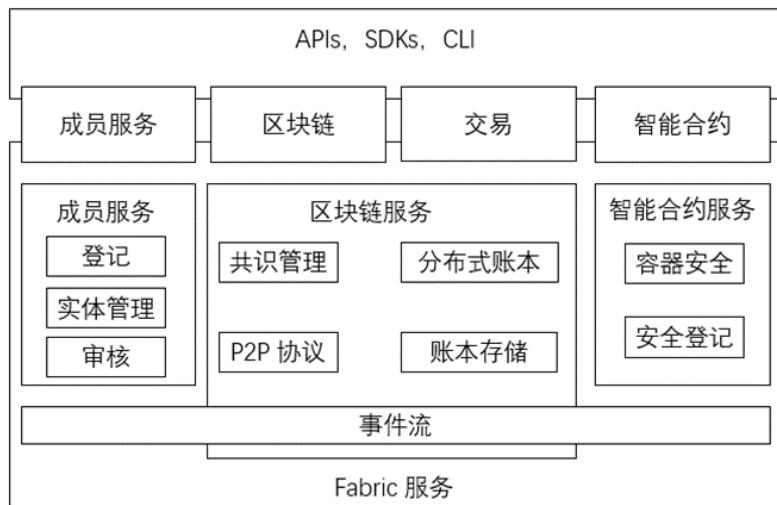


图 2.3 Hyperledger Fabric 系统架构图

1. 成员管理服务

成员服务主要保证 Fabric 平台访问的身份机制，主要提供成员的注册，管理以及审核等功能。

2. 区块链服务

该服务是框架的核心功能，其主要包括负责节点通讯的 P2P 协议、节点的分布式账本、账本存储数据库以及维护账本的共识机制管理。在 Fabric 中节点之间的 P2P 协议主要是使用 gRPC^[36]来提供节点之间以及客户端的远程服务调用。网络中的节点使用了 Gossip^[37]协议实现状态的同步和分发。在 Fabric 中，分布式账本可以分为两个部分，一个是世界状态，另一个是记录的状态操作日志的区块链账本。状态数据库具体可以选择使用 LevelDB 或者 CouchDB^[38]。为了保证

这些数据状态的一致性，Fabric 使用共识机制维护账本数据，在 Fabric 中可以自主选择框架封装好的共识协议实现，也可以对其进行替换。目前 Fabric 内部主要支持 SOLO（开发的时候单节点使用），Kafka^[39]，Raft^[40]三种共识机制。

3. 智能合约

智能合约服务主要是提供包括安全的链码容器环境和安全的注册仓库等功能。链码是一个可以对区块链账本进行操作的组件，通常情况下需要开发人员按照相应的需求场景编写对应的逻辑。之后，这些链码被部署在一个安全的容器中去，客户端通过调用链码中的函数实现相应的业务逻辑操作。链码可以部署多个在背书节点上，客户端可以指定访问的背书节点，做一个负载均衡有利于提升交易的吞吐量^[41]。

4. 事件流

事件流主要为 Fabric 中的各个组件提供异步通信的功能，增加了 Fabric 的灵活性。

2. 2. 2 Hyperledger Fabric 交易流程

Fabric 的交易流程是在开发软件的时候必须要掌握的知识，这些都将为之后理解和开发相应的软件打下坚实的基础。在 Fabric 服务是由不同的服务节点提供的，按照不同的角色的划分，节点可以大致分为三种类型。第一种是背书节点，它主要负责的是对客户端提交参数执行进行计算、检查其他背书节点执行结果、并对检查通过的结果签名背书等功能。第二种是确认节点，它主要在交易结果提交前再次检验合法性，如果合法则记入账本中去。第三种是排序节点，这个节点是为了保证所有节点所获得的账本具有一致性，对交易按照配置中的约定进行排序，并分发给其他节点。

在 Fabric 中完成一次交易的流程如图 2.4 所示。客户端或者应用程序需要在 CA 节点或者使用 Fabric 官方提供的工具生成公私钥证书以此来证明自己的身份。之后客户端需要构造交易的请求。背书节点上部署的智能合约负责执行有相关操作并对执行结果进行签名背书，并将参数通过 P2P 协议传递调用参数，通知其他背书节点进行计算背书。接下来，客户端收集结果并提交到排序节点。排序节点对其进行打包处理，并分发给网络中的提交节点。最终提交节点对交易进行合法性

检测（交易的背书签名是否符合策略，交易的结构是否合法等），检查通过后写入账本。值得一提的是，Fabric 采取了于传统区块链平台主动复制模型^[42]（先排序，后执行）不同的方式，它采用了先执行，后排序，最后验证的系统结构，一般称之为结合被动复制模型^[43]。这种模型和传统的模型相比可以在数据写入前就识别出数据的非确定性。

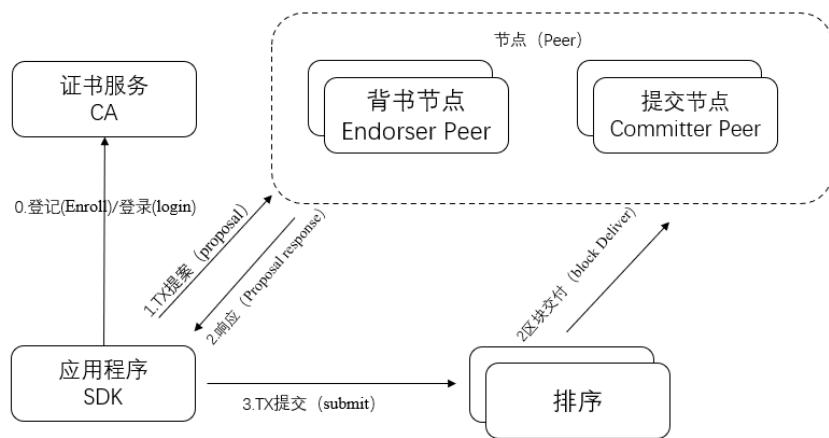


图 2.4 Fabric 交易流程示意图

2.3 IPFS 星际文件系统

IPFS (Inter Planetary File System) 是集成了诸多技术于一体分布式存储系统，主要包括分布式哈希表、比特流、自认证文件系统以及 Git 等技术，并充分利用了这些技术的优势^[44]。因此，IPFS 技术在未来可能会颠覆当下 HTTP 协议在互联网中的地位。它与建立在 HTTP 协议最大的不同之处在于建立在 IPFS 之上的系统的数据是分布式存储的，而 HTTP 协议则是将数据放置在服务端，对服务器造成了巨大压力。IPFS 寻址的原理是通过内容寻址，当将数据存入 IPFS 系统中后，系统返回基于该数据得出的哈希值^[45]。根据哈希函数的原理，即使只对信息做轻微修改，也会得到完全不同的哈希值。这个特性将保证在 IPFS 上的数据不会被篡改。当向 IPFS 文件系统请求数据的时候，它会使用分布式哈希表找到请求数据所在的节点，并在该节点中请求数据返回。IPFS 的这些特性使得在应用层面区块链技术完美结合，为传统的区块链程序提供分布式存储方案。

2.4 密码学技术

在本节主要介绍一些基本的密码学基础知识，为后文使用这些技术打下基础。

2.4.1 哈希算法

哈希函数是一种数学函数，也被称为散列函数或者消息摘要函数。对于任意长度的数据返回固定长度的输出，且该过程是不可逆的，即无法从输出信息反推出任何与原数据相关的内容。从图 2.5 中即可看出，即使相似的信息的输入会导致输出结果的大相径庭并且很难找到两个不同的输入值产生相同的输出^[46]。

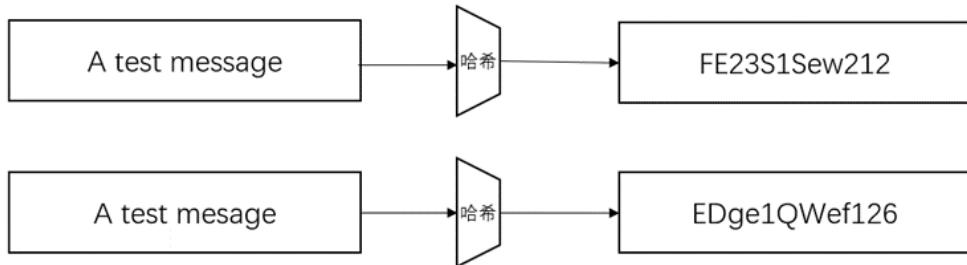


图 2.5 Hash 函数示意图

目前常见的哈希函数算法主要包括 MD 和 SHA 系列算法以及我国国家密码局认定的 SM 系列算法^[47]。MD 系列中包括 MD4 和 MD5 两种哈希算法，目前比较常用的是 MD5 算法。SHA 算法系列可以分为 SHA-1，SHA-2 以及 SHA-3 系列，其中 SHA1 被认为不够安全目前已经弃用，SHA2 算法包括 SHA-224, SHA-256, SHA-384, SHA-512 等算法。比特币底层的区块链网络中使用的就是 SHA-256 算法。为了应对日益复杂的国际情势，保障我国各个领域的安全，我国自主研发了 SM 系列国密算法，例如：SM2 算法是采用了椭圆曲线加密原理的非对称加密算法^[48]；SM3 算法是我国自主设计的摘要算法^[49]。

2.4.2 对称与非对称加密算法

加密和解密技术作为密码学的核心技术，根据加密解密类型的不同，可以把加解密技术分为两类。其中，加密和解密使用同一密钥的算法称为对称加密算法，加密和解密使用不同密钥的算法称为非对称加密算法。具体情况如表 2.1 所示。

表 2.1 加密算法分类表

算法类型	特点	优点	缺点	典型代表
对称加密算法	加密和解密阶段使用同一密钥	效率高，加解密占用资源少	密钥容易泄漏	DES, 3DES, AES 等
非对称加密算法	加密和解密阶段使用不同的密钥	安全强度高，密钥不容易卸扣	加密解密耗费资源多	RSA, ECC 等

2.4.3 代理重加密

代理重加密是建立在非对称加密算法上的一种特殊类型的加密技术，它最早在文献^[50]中由 Blaze 等人提出，主要包括加密钥生成算法、加密算法，代理重加密密钥生成算法、重加密算法以及解密算法等五个算法。这门技术主要是为了解决如何在不暴露自己私钥的情况下，将只能由自己私钥解密的密文，利用代理方使用重加密密钥将密文转换指定身份私钥可解密新密文。于此同时，不暴露双方的私钥，代理者也无法获得明文信息，确保数据安全隐私。该技术的流程模型示意图如图 2.6 所示，主要涉及数据所有者（Alice），代理加密方（Proxy）以及数据请求者（Bob）三个角色，其具体过程如下：

1. 数据所有者（Alice）和数据请求者（Bob）利用密钥生成算法得到各自的公私钥对。然后数据所有者（Alice）使用公钥（pk_A）加密明文 M 为密文 C_A。

2. 数据所有者 Alice 调用代理密钥生成算法，生成重加密密钥（ $rk_{A \rightarrow B}$ ）交由代理加密方（Proxy）。
3. 代理加密方（Proxy）可以提供计算环境的任意实体，例如：云服务器或者区块链上的智能合约。该实体使用重加密密钥（ $rk_{A \rightarrow B}$ ），对密文 C_{M1} 进行加密生成密文 C_B 。
4. 数据请求者（Bob）请求重加密过后的密文 C_B 后，即可利用自己的私钥（ sk_B ）调用解密算法获取原始明文数据 M 。

在上述过程中代理者并没有直接接触明文数据，代理者只需要忠实的执行代理重加密过程，只有数据共享双方可以获取明文数据且无需交换交换私钥，提升了数据的安全性。

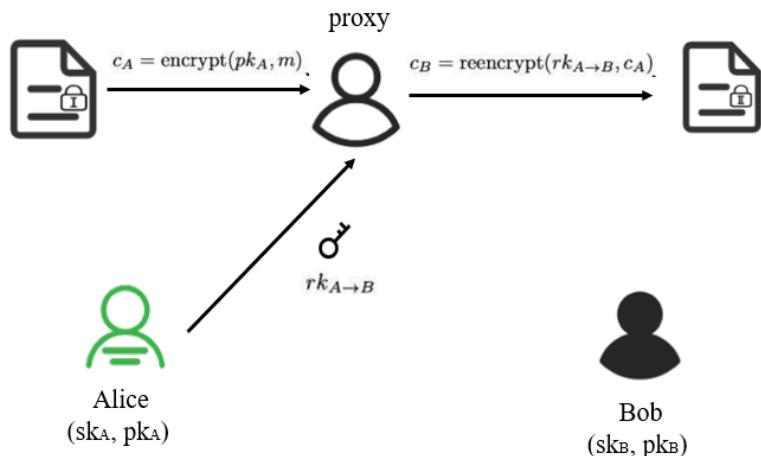


图 2.6 代理重加密示意图

2.5 本章小结

本章对系统关键技术理论及进行介绍，首先叙述了区块链的起源分类和区块数据结构。然后介绍了 Fabric 框架的相关知识和其架构和交易流程的分析。最后介绍了星际文件系统（IPFS）和密码学技术，并重点讲述了代理重加密算法示例流程，为之后的使用这些技术做了铺垫。

第三章 系统的需求分析

本章主要讲述系统的需求分析。通过总分的方式，首先叙述了系统的总体需求。然后根据用户用例图分析系统的功能性需求。最后对系统的非功能需求进行了分析。

3.1 系统总体需求概述

电子医疗病历管理系统在医疗机构中已经十分的普及，但是这些医疗病历管理系统往往并不互通，使得电子医疗病历数据的在不同机构中流转和共享颇为困难。而我国目前医疗资源总体上并不丰富，分布也失衡，导致了患者在不同的医疗机构之间进行就诊的现象十分普遍。以上两种矛盾的现状导致了在不同医疗机构之间电子医疗病历数据共享的需求普遍存在。于此同时，因为医疗数据具有极高的利用价值，极易引来黑客攻击导致数据泄露，危害患者的权益。因此，在构建该系统的同时不能忽略对患者医疗数据的安全和隐私性的保护，避免泄露用户隐私数据。

因此，本文的系统总体需求如下：

1. 打破目前医疗电子病历系统中存在的数据孤岛，建立一个可以实现在不同医疗机构中进行数据共享的电子病历共享系统，使得患者医疗信息可以跨机构共享，提升患者的就医体验并节省医疗资源，达到患者和医疗机构双赢的目的。
2. 在实现医疗数据在不同医疗机构之间互联互通的基础上的同时确保患者的医疗数据隐私安全，将医疗数据共享的权限掌握在患者自己的手中，除非得到患者授权其他人并不能获取患者的医疗数据。另外，系统需确保达到保护患者隐私的目的，解决医疗数据共享和隐私保护的两难困境。

3.2 功能性需求分析

本文系统的用户主要包括三种类型，分别是患者、医生以及医疗机构的管理人员。其中患者是自己电子病历的所有者，通过使用本系统应当可以实现挂号就诊、医疗信息查看管理授权、个人信息管理等操作。医生应当可以管理自己病人

挂号信息、为患者增加电子病历等医疗信息并向患者请求授权访问患者的历史病历等服务。医疗机构中的管理人员主要负责整个系统的正常运转，可以管理用户的账户信息、医院内药品的信息以及底层的区块链系统通道配置维护等功能。

通过前文对系统的功能需求分析，患者的功能用例图如图 3.1 所示

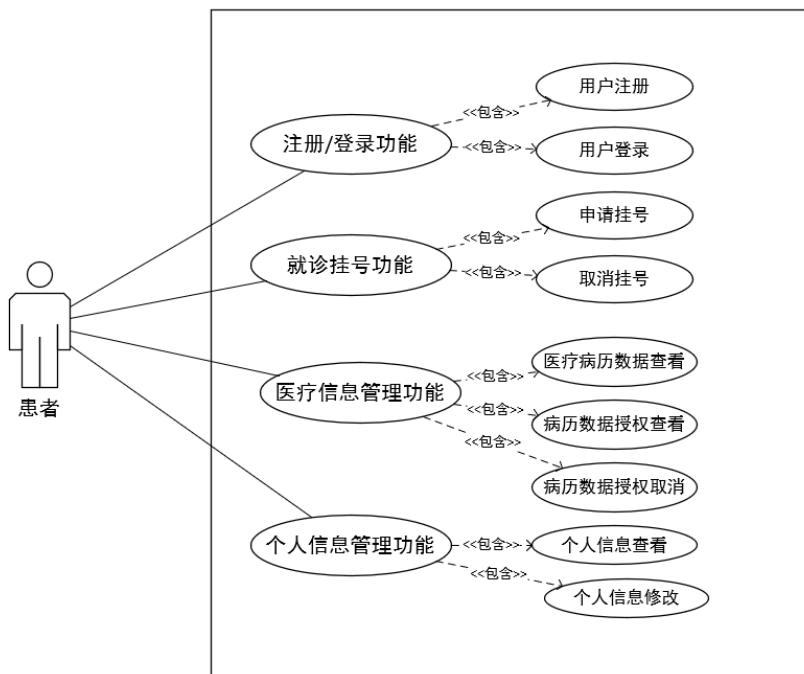


图 3.1 患者用户用例图

在本文的系统中患者所需要的功能用例主要由以下几个部分组成：

1. 登录注册功能。用户在注册页面上填写相关信息进行注册。成功注册之后，用户即可通过这个账户信息登录本系统进一步的使用系统的功能。
2. 就诊挂号功能。患者在进行就诊之前需要根据自己的需要进行查询相关信息，并选择医生进行挂号操作。挂号成功之后就会出现在医生病人列表中，且在挂号成功之后用户还可以取消之前的挂号操作。
3. 医疗信息管理功能。在该模块主要包括病人可以查看系统内关于自己的病历信息、授权给相应的医生进行查看自己病历信息以及在授权列表中取消给某个医生的授权等功能。患者在这个功能实现了对于自身医疗数据的管理。
4. 个人信息管理功能。在该模块病人可以查看自己注册相应个人信息，并对一些个人信息进行修改。

医生角色是系统的关键角色之一，在本文的系统中医生角色主要负责对挂号的用户进行诊断、创建电子病历并加密存储以及申请查看以往的患者病历数据，医生角色的用例图如图 3.2 所示。

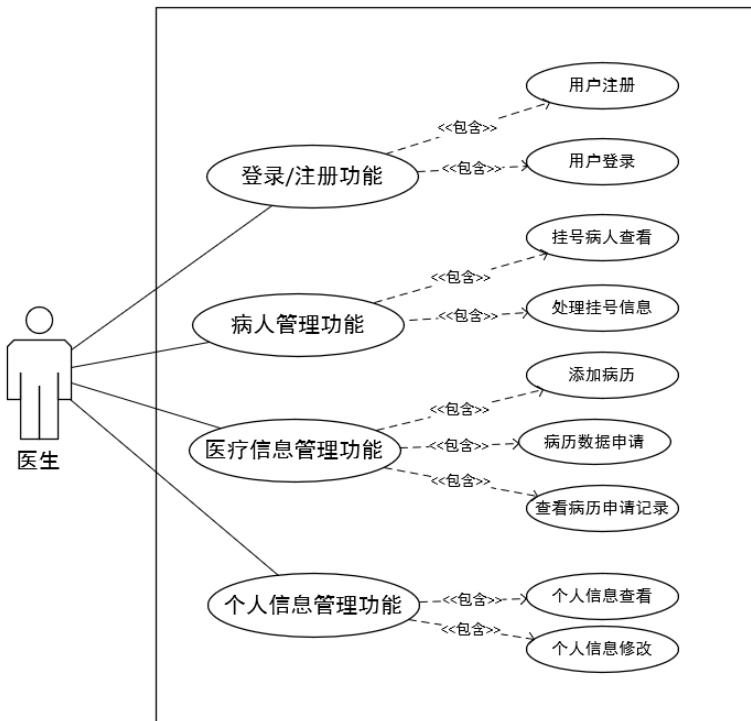


图 3.2 医生用户用例图

在医生的用例图中主要包括以下几个功能。

1. 登录注册功能：在医生的登录注册功能中和患者的功能基本一致，系统提供相应的注册页面给医生使用，方便医生进行填写相应的信息，注册成功之后即可登录系统到主页查看个人注册信息。
2. 病人管理功能：在这个功能中，医生可以查看到相应的已经挂号在他名下的患者的信息，并能够对挂号信息进行相应的处理，例如添加病历信息或者跳过当前排队的挂号患者。
3. 医疗信息管理功能：此模块是系统的核心功能模块，首先添加医疗病历信息，在该功能中，医生可以为已挂号的病人进行创建病历记录并加密后存储 IPFS 节点中，实现医疗病历的可信存储。另外，在该模块医生可以向病人申请病人以往的医疗病历记录以及在得到病人授权调用代理重加密中的解密算法查看明文数据，实现医疗病历的安全共享。

4. 个人信息管理功能：在该模块中医生的功能和患者的基本一致，主要包括医生查看自己注册的个人信息以及对相应的个人数据进行修改等功能。

在本文的系统的中还应存在着一个系统管理员进行系统的日常维护和医疗机构中药品信息管理的功能，同时由于本文底层采用了 Fabric 联盟链作为底层支撑，该角色还应承担联盟链的系统通道配置功能。因此，管理员用例图如图 3.3 所示。

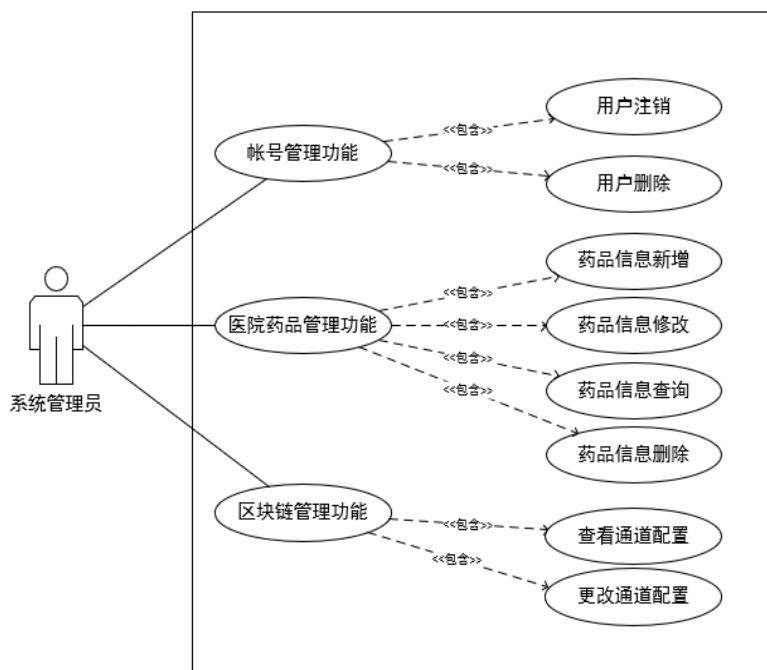


图 3.3 系统管理员用例图

在医疗机构系统管理员的用例图中主要包括了如下几个功能。

1. 系统账户管理：在这个功能模块中主要包括了对系统内部账户的注销和删除功能，被注销或者删除的账户信息将不能登录到本系统提供的页面上进行操作。系统管理员账户信息由系统分配，无需进行注册。

2. 医院药品信息管理：该模块对医疗机构加强用药指引，规范医生合理用药，方便日后扩展对药品信息和价格监管和审计，所有药品通过医院管理员审核之后才能添加进入药品库，提供给患者使用。该功能主要包括医院里面的药品信息的

添加、查询、修改以及删除等操作。这些信息在系统的相关页面进行展示，其他用户可以查询到相关药品的信息。

3. 区块链管理模块：在该模块中，系统管理员还承担着区块链底层系统的维护工作，主要包括查看和更改通道配置两个功能。这些功能将帮助系统更好的管理区块链系统，增加区块链系统的可扩展性。

3.3 非功能需求分析

在功能性需求之外，非功能性需求也是系统需求不可缺少的一环，对系统的正常使用来说十分重要。

1. 安全性：本系统要求做到保护用户的医疗数据隐私，做到用户医疗数据不被篡改，并且不泄露用户的医疗数据，保护用户医疗数据的安全性和完整性。

2. 可用性：从用户的角度出发，系统应当是易于学习的，不应有过于复杂的流程，对所有使用本系统的用户来说，快速上手是十分有必要的。从系统本身的角度出发，系统也必须是高效的，不能够在用户使用的过程中出现过长的等待事件导致用户失去耐心。

3. 可扩展性：系统的功能模块设计应遵守高内聚低耦合原则。各个模块之间通过对应的接口来相互通信，不仅方便在本系统的上进行下一步的功能扩展，丰富系统的功能，也会为将来运行维护等工作的开展提供了便利。

4. 页面需求：系统的页面要简洁和易于使用，应当合理化页面布局以及遵循常规的设计原则，最终做到适配如今市面常用的各种浏览器类型，避免出现一些低级的请求错误，致力于给用户带来的良好的体验。

3.4 本章小结

本章首先对目前医疗数据中存在的数据孤岛现象和隐私保护问题之间的两难困境问题做了系统的总体需求分析，确定了系统总体需求。接着将系统的需求划分为功能需求和非功能需求两类。在功能需求部分主要通过系统角色用例图来进一步分析系统应该具有的功能点，并对每一个功能点进行了解释说明。在非功

能需求部分主要对系统的安全性、可用性、可扩展性以及页面布局的易用性需求等方面做出了相应的规定。

第四章 系统的总体设计

本章针对前面给出的系统需求分析内容，叙述基于区块链的电子病历共享系统的总体设计方案，主要包括系统病历共享流程设计、系统架构设计、功能模块设计和数据模型设计等内容。

4.1 病历共享流程设计

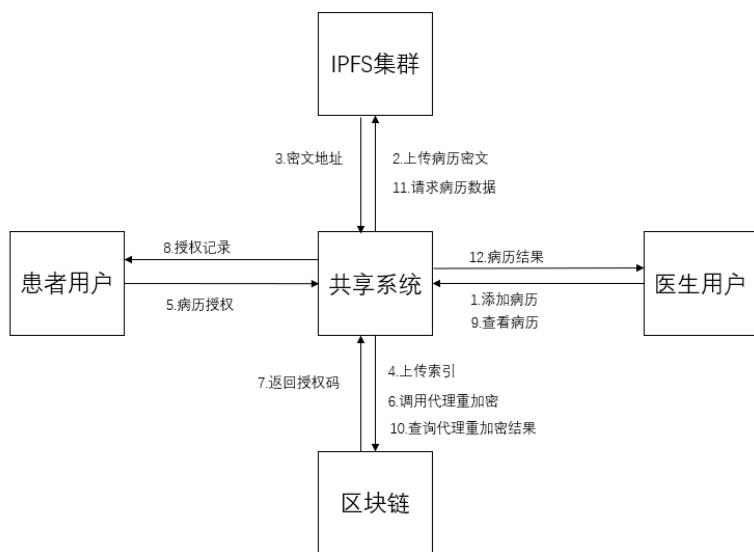


图 4.1 病历共享模型流程图

如图 4.1 所示为系统病历共享流程设计图。在本系统中病历共享流程一共可划分为三个阶段。步骤 1 至步骤 4 为添加病历阶段，在该阶段中医生用户为患者添加病历记录。系统将加密过后的病历数据添加到 IPFS 系统中去，IPFS 系统返回数据地址。系统将病历地址和加密密钥以及用户其他的信息组成病历索引并添加到区块链中。步骤 5 至步骤 8 为病历授权阶段，患者选定需要共享的医生之后，根据相关信息生成代理重加密密钥并将其传入智能合约中执行代理重加密，然后生成一个授权码关联代理重加密结果，作为键值对存入区块链系统中。在该阶段的最后系统生成一条授权记录并返回授权码。步骤 9 至步骤 12 为病历查看阶段，在该阶段医生通过输入授权码，在区块链中查询其关联的代理重加密结果，

利用该结果解密索引密文，即可通过索引信息请求 IPFS 系统拿到病历记录密文并解密查看。

4.2 系统软件架构设计

本文系统的整体逻辑架构图如图 4.2 所示，主要包括展示层、业务逻辑层、以及存储层三层组成，下面对每一层进行详细的介绍。

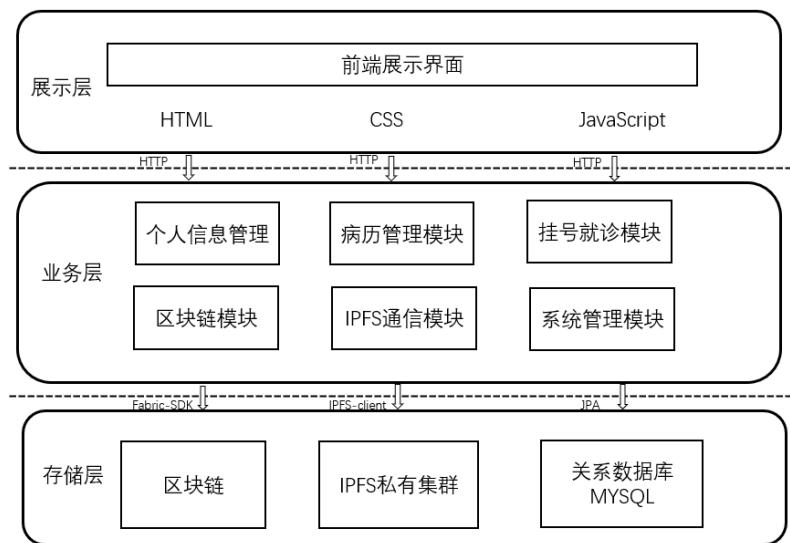


图 4.2 系统逻辑架构图

展示层：该层的主要功能是提供一个易用的界面，方便用户使用系统的功能，使得用户拥有一个良好的使用体验。在该层主要采用了 HTML、CSS 以及 JavaScript 等技术，并采用 Vue.js 框架简化了前端代码的书写，提高了开发的效率。本系统采用前后端分离的方式，前端通过 HTTP 请求的方式与业务端进行通信。

业务层：该层的主要作用是提供系统功能的具体业务逻辑实现，并披露 Restful 风格的接口供前端接口调用。在本文的系统中的功能主要划分为以下几个模块。在个人信息管理模块，医生和患者可以注册和登录该系统和管理个人信息。在病历管理系统，主要包括患者与医生之间的电子病历数据共享等功能。在挂号就诊模块提供患者选择医生就诊和医生处理挂号病人的功能实现。在系统管理模块主要包括系统内部账户状态的管理以及区块链系统通道的管理操作。智能

合约虽然是运行在区块链网络上的但是其功能与系统业务层逻辑密不可分，所以仍将其视作业务层的区块链模块。该模块主要负责管理存在区块链中的加密索引等数据的操作以及使用代理重加密密钥执行代理重加密过程。最后，本系统还需要包括与 IPFS 通信模块，该模块利用 Java-IPFS-HTTP-Client 框架与 IPFS 节点进行通信。

存储层：该层主要存储整个系统的数据，在本文的系统中主要包括三个部分：第一部分是区块链系统部分，主要负责存储电子病历的索引信息和代理重加密算法解密需要的椭圆曲线点位信息。本文区块链系统由 Hyperledger Fabric 框架进行搭建，使用 Fabric 官方提供的 SDK 对区块链系统进行通信；第二部分是 IPFS 文件系统，在该系统下存储加密之后的患者电子病历。这样做的目的是减轻区块链中的通信负担，提升系统的性能。以上两个部分形成了“链上索引，链下存储”的模型。最后一部分是传统的关系型数据库，负责存储一些其他业务信息辅助整个系统的运转。在传统的关系型数据库 MYSQL 中采用通过 Spring-Data-JPA 框架与其交互。

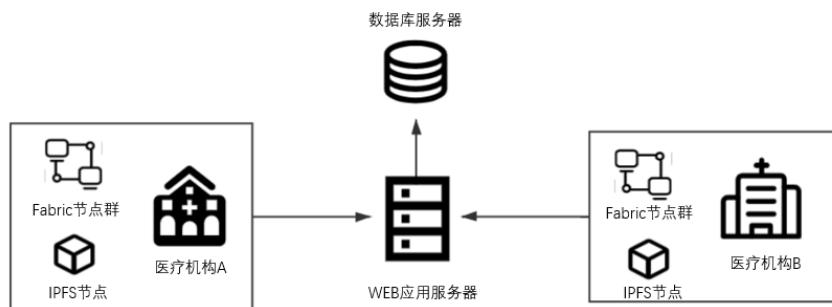


图 4.3 系统物理部署架构图

如图 4.3 所示是本文系统的物理部署架构图，该图中主要包括 Fabric 节点群、IPFS 节点、部署系统应用的 WEB 服务器和关系数据库服务器。医疗机构一般都维护着 Fabric 节点和 IPFS 节点两种节点类型。Fabric 节点可以根据逻辑划分为不同的角色，例如：负责运行智能合约并对智能合约执行的结果进行背书和存储数据的背书节点；负责搜集数据将满足背书策略的交易打包成区块分发给不同的机构之间节点的排序节点。在本文的系统中医疗机构中的一般都维护这两种 Fabric 节点类型，并在背书节点处存储病历加密过后索引数据。医疗机构之间运

行 IPFS 节点组成的 IPFS 私有化集群储加密过后的患者医疗数据，减轻区块链系统中的通信和存储负担，通过这样的方式形成“链上索引，链下存储”的模型。最后，本系统部署在 WEB 应用服务上负责和 Fabric 节点、IPFS 节点以及业务数据库服务器的通信交互，WEB 应用服务器和数据库服务器可由第三方可信机构来提供。通过这样的方式将不同的医疗机构连通起来，打破数据孤岛，既实现电子病历在不同医疗机构之间的共享，达到医疗数据的互联互通的效果，又有效的保护了用户数据的隐私安全。

4.3 系统功能模块概要设计

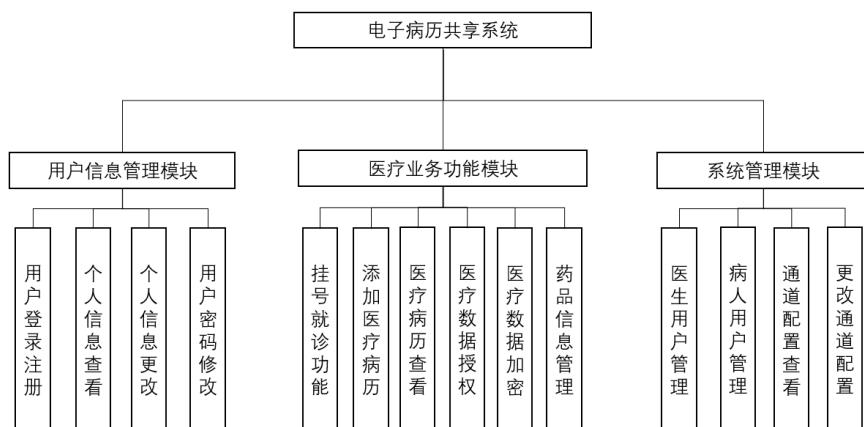


图 4.4 系统功能模块图

如图 4.4 所示，本文系统主要由个人信息管理模块、医疗业务功能模块以及系统管理三个功能模块组成。首先，在个人信息管理模块，主要包括的功能是用户的注册登录、用户个人信息的查看与修改以及用户密码修改等功能。该模块是系统其他功能模块的基础，只有在该模块中注册并登录的用户才可以使用系统的进一步功能。其次，在医疗业务管理模块是系统的核心模块，主要包括了患者挂号就诊功能、医生为患者添加和申请查看以往的电子病历并将新增电子病历加密存储的功能、患者对医生进行授权查看电子病历的功能以及管理员对于医疗机构内药品信息管理的功能。这些功能组成了患者在医疗机构中就诊和在就诊过程中

产生的医疗电子病历的安全存储和共享的完整流程。最后，在系统管理模块，管理员对用户帐号状态进行管理、查看区块链系统内的通道信息以及修改通道的配置。这些功能可以帮助系统管理员更好的维护整个系统运行并提升系统的可扩展性。

4.4 系统数据模型设计

4.4.1 系统数据概念模型设计

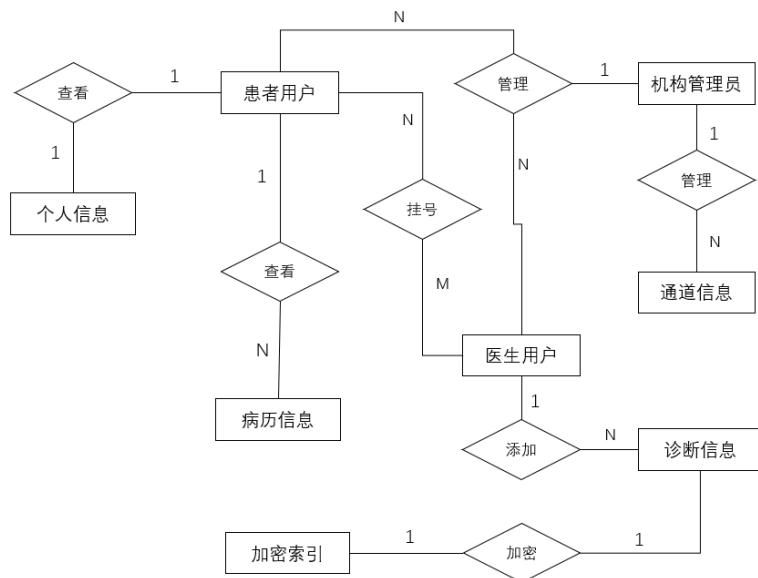


图 4.5 系统 E-R 图

如图 4.5 所示为本文系统核心 E-R 图的具体设计。首先关注和患者实体相关的对应关系：首先找到个人信息实体和病历信息实体。患者用户可以查看个人信息数据，也可以查看既往历史的病历信息数据。另外，依据生活的常识判断，同一个病人拥有的个人信息有且只有一条，因此可以得出患者用户和自己的个人信息是一对一的关系。而在病历信息实体中，对于患者在医院进行多次就诊十分常见的，因此产生多个病历信息也是一种合理的现象，由此得出患者和病历信息实体之间呈现一对多关系。患者用户通过系统向相应的医生用户实体进行挂号操作，因为在一般情况下同一个病人可以向不同的医生发起挂号操作行为，并且同

一个医生也可以对不同的患者用户进行诊断操作，因此，患者用户实体和医生用户实体之间的关系是多对多的。

医生用户实体的对应关系解释如下：在医生用户可以编辑和诊断相应的病历信息，根据生活常识可以判断，一个医生用户对不同的患者进行不同的就诊操作，可以产生多个就诊信息，因此医生用户和就诊信息的关系是一对多的。而加密索引则关联医生添加的就诊信息，一个索引信息对应一条诊断实体。因此，诊断信息的实体和加密索引之间的关系是一对一的关系。

最后是系统管理员实体关系的分析，系统管理员是在系统初始化分配的。管理员用户管理其他用户，对应关系为一对多。最后，管理员负责区块链系统内的通道配置信息的管理，在区块链系统中管理员是可以创建多个通道的，因此一个管理员可以管理多个通道的配置信息，它们的实体关系是一对多的。

4.4.2 系统数据物理模型设计

由于本文系统的功能模块较多，数据库表数量也较为繁杂，为节省论文的篇幅，在本小节将只介绍相对来说比较重要的数据库表格。在下面的几个表格，本小节将详细描述这些核心的数据库表的内容，并对设计的思路进行一定的解释说明。

表 4.1 系统账户信息表

字段名称	数据类型与 长度	是否可 为空	约束条件	解释说明
userID	Varchar (24)	否	PK(主键)	用户的 ID 主键
UserName	Varchar(24)	否	无	用户名称
PasswordHash	Varchar (50)	否	无	用户密码 Hash 值
Cert	Blob	否	无	用户证书信息
UserType	Int (4)	否	无	用户类型
StatusCode	Int (4)	否	无	用户状态
LastLogTime	Datetime	否	无	上次登录时间
CreateTime	Datetime	否	无	创建时间
LogInIP	Varchar (32)	是	无	登录的 IP 地址

表 4.1 所示，患者和医生在登录系统之间需要输入账户名称（userName）以及用户密码（UserPassword）等信息，并选择正确的用户类型（userType）才能合法的登录系统使用系统的功能。系统提供了修改密码的功能，且管理员可以改变用户的状态（StatusCode），只有处于正常状态下的帐号才可以进行业务操作，管理员吊销其账户的时候患者将不能进行操作。另外该表中还记录了每个帐号的最后登录时间以及 IP 地址，以此来记录用户的登录行为。

表 4.2 患者个人信息表

字段名称	数据类型与长度	是否可为空	约束条件	解释说明
PatientID	Varchar (24)	否	主键 (PK)	病人信息 Id
UserID	Varchar (20)	否	无	用户 ID 信息
PatientName	Varchar (10)	否	无	用户名
Birthday	TimeStamp	否	无	出生日期
Nation	Varchar (18)	否	无	民族
Telephone	Varchar (18)	否	无	电话
Gender	Int (5)	否	无	用户性别
Email	Varchar (30)	否	无	电子邮件
PatientIDNum	Varchar (20)	否	无	身份证号

如表 4.2 所示，患者用户在合法登录到本系统中的时候会进入到个人信息页面并展示病人用户的基本信息，主要包括的就是患者的姓名（PatientName）、性别（Gender）、年龄（age）、电话（Telephone）等信息。患者用户还可以通过本系统更改这些数据。

表 4.3 医生信息字段设计表

字段名称	字段类型与长度	是否可为空	约束条件	解释说明
DocID	Varchar (24)	否	主键 (PK)	医生信息 ID
DocName	Varchar (10)	否	无	医生姓名

(续表 4.3)

字段名称	字段类型与 长度	是否可控	约束条件	解释说明
DocBirth	TimeStamp	否	无	医生生日
DocQualification	Varchar (10)	否	无	医生学历
DocDepartment	Varchar (10)	否	无	科室名称
DocTitle	Varchar (10)	否	无	医生职称
GoodAt	Varchar (255)	否	无	医生擅长信 息
Status	Int (4)	否	无	医生状态
Gender	Varchar (4)	否	无	医生性别
HospitalName	Varchar(20)	否	无	医院名称

如表 4.3 所示为医生个人信息表，主要包括医生的一些基本信息记录。该表的部分信息还提供给挂号患者查看，方便患者选择恰当的医生进行挂号操作。系统可以根据医生目前所处的状态进行判断挂号是否成功。

表 4.4 病历信息字段设计表

字段名称	字段类型与 长度	是否可为空	约束条件	解释说明
Id	Varchar (24)	否	主键	病历主键
DepartmentID	Varchar (20)	否	无	就诊科室 ID
SeekTime	Datetime	否	无	就诊日期
Description	Varchar (255)	否	无	患者自述
AllergicRecord	Varchar (255)	是	无	患者过敏记 录
CurrentSituation	Varchar (255)	否	无	患者现状
DocID	Varchar (20)	否	无	就诊医生 ID
Diagnosis	Varchar (255)	否	无	诊断结果
Prescription	Varchar (255)	否	无	处方信息

(续表 4.4)

字段名称	字段类型与长度	是否可为空	约束条件	解释说明
Remark	Varchar (255)	否	无	注意事项
PaitentID	Varchar (20)	否	无	患者 ID

如表 4.4 所示是病历信息表，在医生选择患者进行治疗的时候可以为患者新增这样一条记录，主要填写患者的患者基本情况信息以及医生对患者下的诊断结论信息。在本系统中该表信息会采取加密手段保存到 IPFS 私有集群中去通过在数据库中存储索引信息来检索相关数据。

表 4.5 病历索引表

字段名称	字段类型与长度	可否为空	约束条件	解释说明
ID	Varchar (24)	否	主键	主键
DataAddr	Varchar (40)	否	无	数据加密地址
AESKey	Varchar (40)	否	无	加密密钥密文
PaitentID	Varchar (20)	否	无	患者 ID
DocID	Varchar (20)	否	无	医生 ID
CreateTime	DateTime	否	无	创建日期

如表 4.5 所示为病历数据存储索引表，主要存储病历和其他医疗数据的在 IPFS 文件系统中的加密索引和加密密钥密文信息以及创建和使用这些数据的医生和患者的 ID 信息。本文将这些信息序列化为 Json 格式的数据并存储到区块链的 CouchDB 中去。这些信息是实现医疗数据的安全可信共享的关键信息，可以通过 CouchDB 提供的富查询功能来查出需要使用的数据信息。

表 4.6 药品信息字段设计表

字段名称	字段类型与长度	可否为空	约束条件	解释说明
Id	Varchar (24)	否	主键 (PK)	主键

(续表 4.6)

字段类型	字段类型与长度	是否为空	约束条件	解释说明
MedName	Varchar (20)	否	无	药品名称
MedPrice	Double (10)	否	无	药品价格
MedType	Int(4)	否	无	药品类型
MedSpecs	Varchar (12)	否	无	药品规格
Manufacturer	Varchar (24)	否	无	生产厂家
CreateTime	DatETIME	否	无	创建时间
Status	Int (4)	否	无	可用状态
Detail	Varchar (255)	否	无	详细信息

如表 4.6 所示为系统内的药品信息字段设计，医疗机构的管理员登录系统之后可以在药品展示页面对系统内的药品名称、药品描述、药品价格等信息进行对应的增加、删除、修改、查询以及对药品状态的修改等操作。

4.5 本章小结

本章主要讲解了系统的总体设计方案。首先介绍了系统的共享流程设计，然后系统的总体架构开始，从逻辑架构角度讲解了系统的架构模型设计思路。接着对系统的功能模块进行了概要设计，主要依据前文的需求分析，将系统的功能划分为不同的模块，并对各个功能模块所包含的功能点进行了说明。最后对系统的数据模型进行了设计，主要通过 E-R 图说明了系统数据的概念模型设计和详细的字段设计表说明了系统数据物理模型的设计。

第五章 系统的详细设计与实现

本章将根据前面章节对系统的需求分析和概要设计，对系统涉及到的所有模块进行说明，主要包括用户信息管理模块、医疗业务功能模块以及系统管理模块等功能模块。同时还单独对智能合约模块详细的设计与实现进行了说明。最后叙述了搭建 Fabric 区块链网络和 IPFS 私有化节点的过程。

5.1 系统功能模块的详细设计与实现

本节通过前文的功能模块概要设计内容中划分的功能模块，首先通过流程图示意系统功能如何使用，然后通过时序图讲解系统内类之间调用关系，最后通过类图展示了相关功能的核心类图设计细节。

5.1.1 用户信息管理模块的详细设计与实现

用户信息管理功能模块是系统的基础模块，只有在这个模块中注册并登录的用户才能使用系统的其他功能。在该模块中主要包括的是用户的登录注册、查看个人基本信息、修改个人信息以及修改账户密码等四个业务功能。

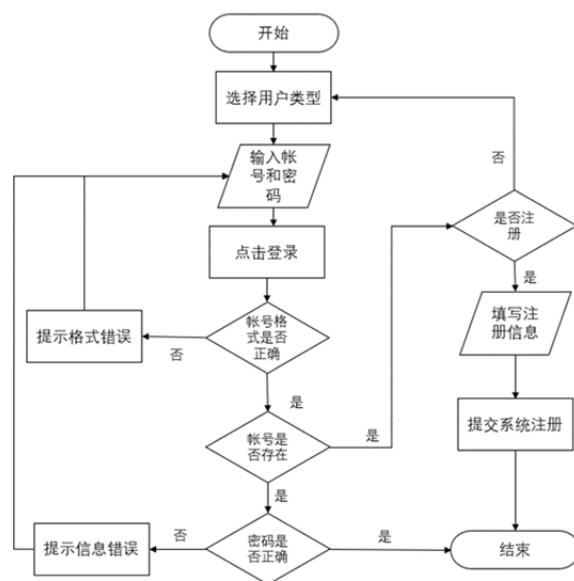


图 5.1 系统登录注册流程图

登录注册功能流程图如图 5.1 所示。首先选择账户类型，然后输入相关信息，执行登录操作。系统内判断账户的格式是否正确，如果不正确则提示用户相关信息。之后系统将检查账户是否存在，如果账户存在则校验账户密码，当两者都通过系统的检测之后即可登录成功。如果发现账户不存在则提示相关信息，并询问是否注册。如果同意注册则跳转到注册流程，在注册页面填写相关的信息并提交系统注册。如果放弃注册，可以返回登录首页。最后，如果信息有误，那么系统提示相关信息，并跳回登录信息页面。

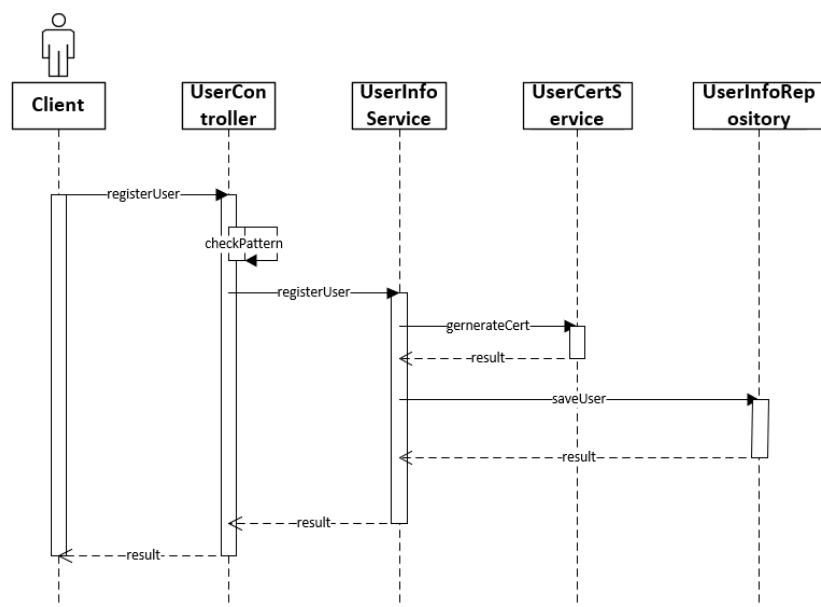


图 5.2 用户注册时序图

如图 5.2 所示为注册功能时序图，首先，前端调用 `UserController` 类中的 `registerUser` 接口，在该接口中调用 `checkPattern` 方法检查用户填写的相应信息是否符合格式规范，如果满足的数据符合规范再进行下一步的操作。`registerUser` 方法进行下一步的操作主要就是调用 `UserCertService` 类中的方法生成当前用户的证书和私钥信息返回，该证书和私钥将用于加密和解密医疗病历数据，其中证书存入系统业务数据库中，私钥将其放在用户使用的客户端进行保存。然后将用户填写的其他信息通过调用 `UserInfoRepository` 类中的方法存储到数据库当中，最后将注册结果返回。

在该模块个人信息管理与修改密码的功能流程图如图 5.3 所示。在个人信息查看的功能里，用户可以点击查看个人信息按钮进入相关的展示页面，并且通过点击修改功能按钮修改信息。修改密码功能主要流程为先校验旧密码，通过即可更新账户密码。

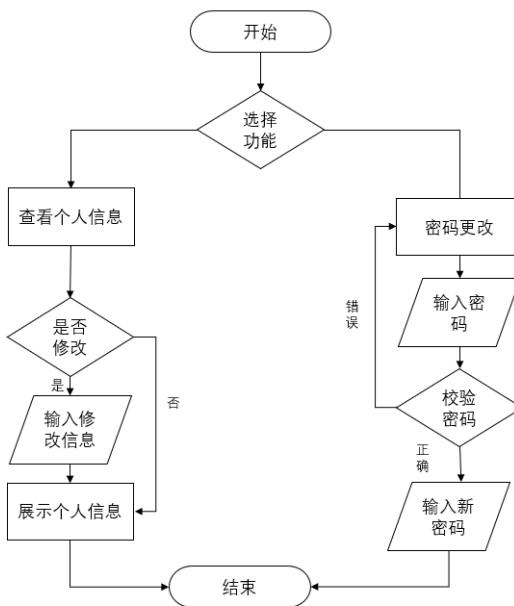


图 5.3 个人信息查看与修改流程图

以上几个功能涉及到的主要类图如图 5.4 所示，`CommonController` 类包含了一些日志记录和验证字符格式等常见的通用方法。`UserController` 类除了继承父类 `CommonController` 中提供的基本功能之外，此类主要负责查询和更新用户信息，并对外提供了 Restful 风格的接口供前端调用。`UserInfoService` 负责具体的业务逻辑实现，例如：注册用户方法、通过用户类型查询用户信息方法、更新用户信息方法等各种实现业务功能的方法。`UserInfo` 类是用户信息实体类。`UserCertService` 类主要负责生成用户的证书和私钥信息，用于系统内数据的加解密过程。`UserInfoRespository` 类中使用 Spring-Data-JPA 的规范添加方法名称和相应的注解来拼接对应的 SQL 语句的方式，实现了数据库中用户数据查询和更新等功能。`JPRepository<UserInfo, Integer>` 接口为 JPA 规范下数据操作类必须实现的接口。

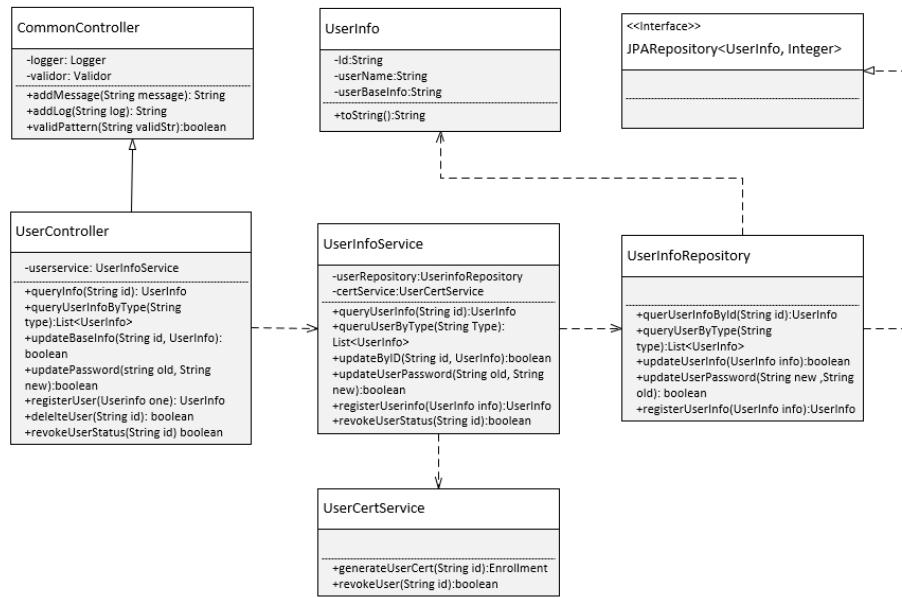


图 5.4 系统用户模块核心类图

5.1.2 医疗业务功能模块的详细设计与实现

医疗业务功能模块是系统的核心业务模块，在本模块实现了和电子病历共享的相关功能。首先，从患者的角度来说主要包括选择医生挂号就诊、授权医生病历查看权限、查看病历授权记录以及取消授权等功能。从医生的角度来说主要包括处理挂号病人、为患者添加病历信息、病历查看申请等功能。

如图 5.5 所示为患者就诊挂号和医生查看挂号病人的流程图。首先，在患者用户就诊挂号功能中，患者可以在挂号页面根据自己的病情需要选择相应的科室医生进行挂号操作，系统会判断当前的医生当前的状态判断是否挂号成功。如果没有挂号成功则系统出现提示信息挂号失败并返回挂号页面。如果挂号成功，则系统提示成功挂号。其次，在医生查看挂号病人的功能中，医生用户可以查看挂号在其名下的患者信息，系统查询相应的数据反馈给对应的医生并展示当前挂号患者信息，医生可以进行下一步的诊断操作。

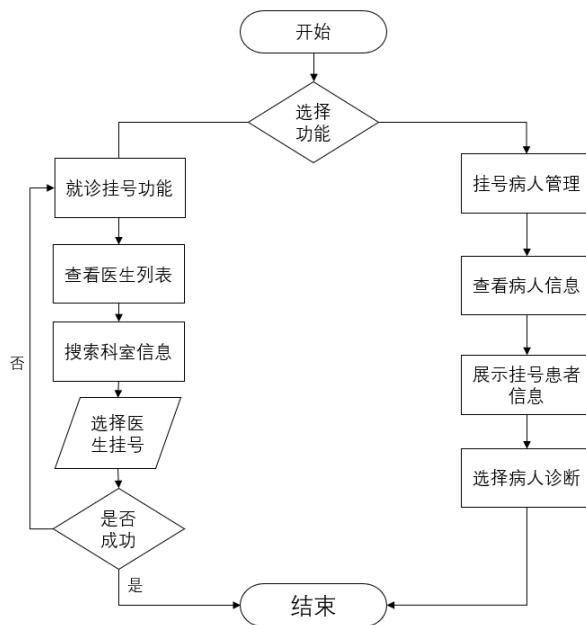


图 5.5 挂号就诊功能流程图

如图 5.6 所示为患者就诊挂号和医生查看挂号病人列表两个功能实现的涉及到的类图设计。MedicalRegisterController 类主要负责提供前端负责请求的接口，主要包括根据不同的条件查询挂号的数据记录以及对挂号数据的状态的操作。例如：用户调用 queryRegisterInfoByUserId 方法通过自己的 id 查询自己的挂号数据并进行对应的操作；医生调用 queryRegisterInfoByDocId 方法，通过自己的 Id 查询目前所有挂在自己 ID 下的挂号列表，并通过 updateRegisterInfo 方法处理这条挂号数据的状态。同样依据 WEB 开发中的 MVC 模式规范的要求，业务方法具体的逻辑实现在 MedicalRegisterService 类中提供对挂号记录的各种条件查询、添加挂号记录数据、修改以及删除挂号记录等操作功能。最后通过在 MedicalRegisterRepository 类中编写相应的 SQL 语句操作数据库中的挂号记录数据内容。

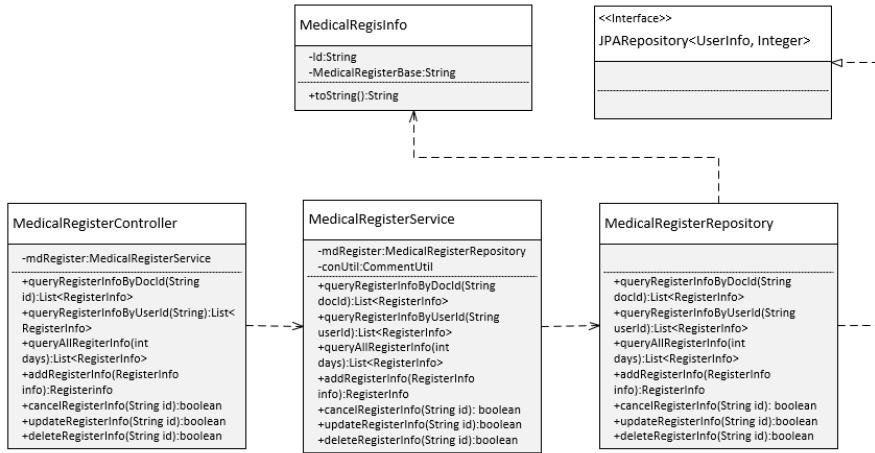


图 5.6 用户挂号就诊核心类图

如图 5.7 所示为医生添加病历和申请查看病历记录的流程图。首先，在添加病历的功能里面医生可以在挂号列表中的患者添加病历信息，并使用 AES 加密算法将其存储在 IPFS 私有集群中。然后将 IPFS 返回的地址信息、AES 加密钥、用户 Id 以及时间戳等信息来组成密文索引数据结构。之后再使用患者用户的公钥加密 IPFS 地址和 AES 密钥之后存储到区块链的数据库中，实现用户医疗电子病历的安全存储。其次，在申请查看病历记录的功能里，医生可以根据挂号病人 ID 信息查询到该病人下的病历索引记录列表，并输入授权码之后系统即可通过授权码查询到该索引代理重加密结果，并利用该结果解密索引信息，通过索引获取并解密病历明文。

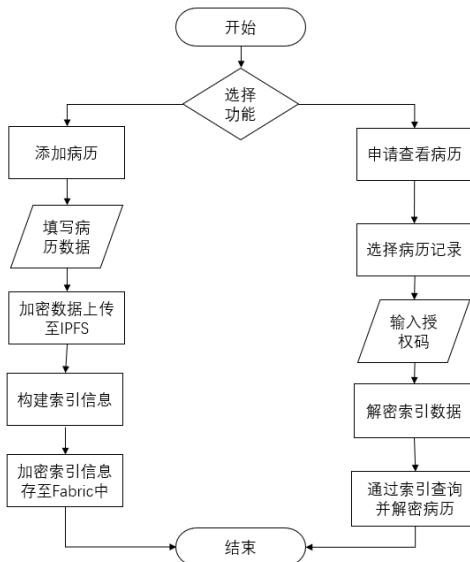


图 5.7 病历添加与授权流程图

如图 5.8 所示为医生用户为患者添加电子病历功能的时序图。用户首先调用 MedicalRecordController 类中调用添加数据的接口，在 MedicalRecordService 类中首先调用 EncryptService 加密类中使用 AES 加密算法加密数据明文数据。然后调用 IPFSService 类中封装的与 IPFS 节点通信的上传数据接口，实现将加密后的记录上传至 IPFS 系统，并取回地址信息。下一步是构建记录索引，该类将使用 IPFS 地址、AES 加密密钥以及用户 ID 等信息构建一个索引结构，并使用用户证书中的公钥将地址和 AES 加密密钥加密。最后通过 FabricChaincodeService 类调用相应的智能合约将其存储到区块链中并返回执行结果完成电子病历的加密安全存储。

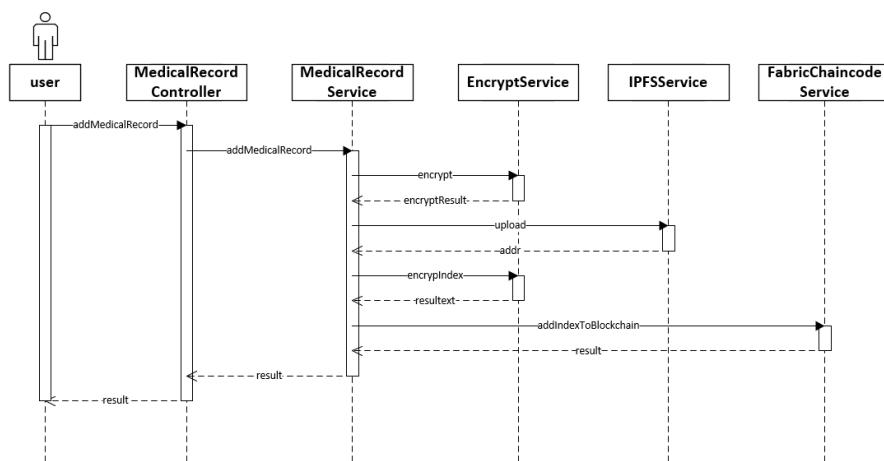


图 5.8 医生添加病历时序图

医生通过授权码解密病历功能的时序图如图 5.9 所示。在图中可以看出用户首先通过请求 MedicalIndexController 类中的 queryMedicalIndexInfo 方法向系统相应的病历索引记录列表。该方法之后调用在 MedicalRecordIndexService 类中通过 FabricChaincode 类调用智能合约，智能合约通过富查询语句查询患者的电子病历索引信息。然后通过 FabricChaincodeService 类调用对应的智能合约函数，使用 AuthCode 信息查询代理重加密的结果。根据该结果调用 EncryptService 类中的 decryptIndex 方法获取索引明文。接下来通过拿到索引中的 IPFS 地址信息和加密数据的对称加密密钥，然后即可以使用解密的 IPFS 地址利用 IPFSService 封装的 queryMedicalData 函数向 IPFS 节点请求数据密文。最后再次调用 EncryptService 中的 decryptMedicalData 方法解密病历密文并返回。

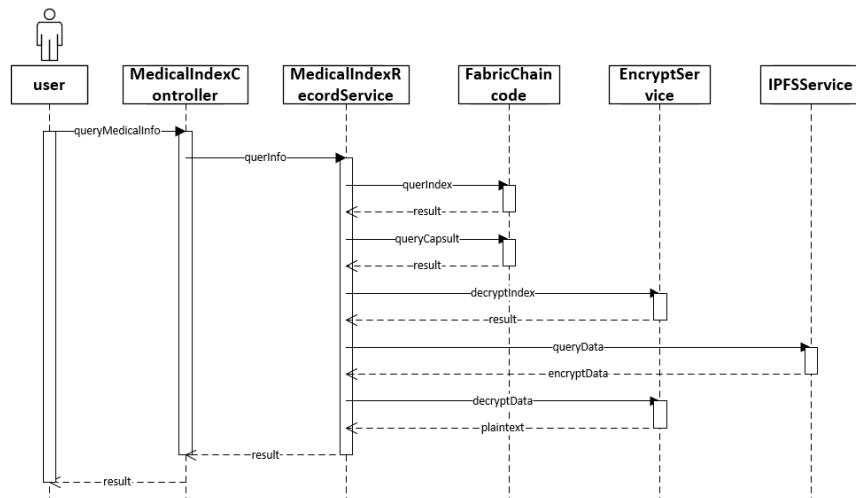


图 5.9 医生解密病历时序图

如图 5.10 所示为患者用户的在本功能中的病历授权和授权取消功能流程图。在病历授权功能中，患者可以查看自身系统中的病历索引记录列表，并选择特定的病历记录并输入医生编码，查询医生信息并生成代理重加密密钥，并将其作为参数传入智能合约中执行代理重加密。然后将生成授权码和代理重加密后的结果关联，最后返回一条授权记录。在授权取消的功能中，患者可查看以往的授权记录，并选择特定的记录将其删除，系统将调用智能合约删除该授权的代理重加密结果。医生将不能通过之前授权码查询到代理重加密的结果，达成了取消授权的结果。

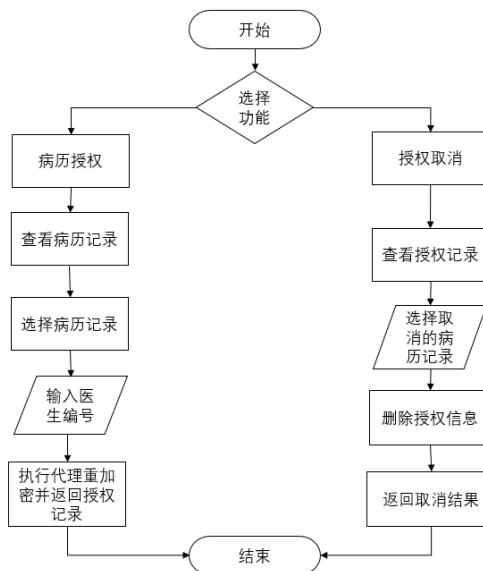


图 5.10 患者授权与取消流程图

如图 5.11 所示是用户进行医疗数据授权操作的时序图。当用户同意数据授权的时候，通过调用 `MedicalRecordController` 类中的授权接口，来实现医疗数据的授权访问操作。该类的授权操作通过 `MedicalRecordService` 类中的 `authProxy` 函数进行实现。首先，调用 `UserInfoService` 通过申请者的 Id 信息和编号等信息查询数据申请用户的证书信息，并通过该证书中包含的公钥等信息调用 `EncryptService` 类中的 `generateReEncryptKey` 函数来生成指定的重加密密钥 `ReEncryptKey`。最后通过 `FabricChaincodeService` 类调用智能合约并传入重加密密钥执行代理重加密，之后智能合约将返回 `AuthCode` 信息作为索引，关联其代理重加密结果。最后，调用 `MedicalAuthListService` 类结合 `Authcode` 以及其他信息生成授权记录 `authInfo` 返回。

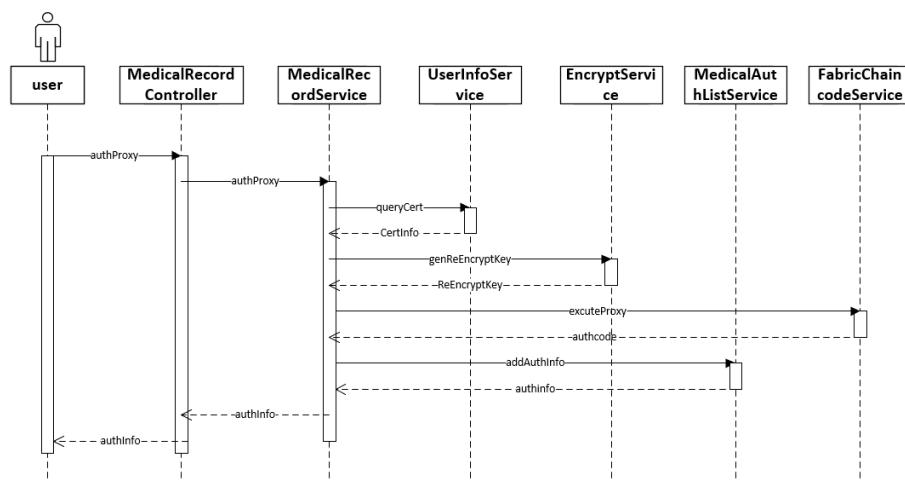


图 5.11 患者授权时序图

患者取消授权记录操作具体的具体流程为患者在授权记录表中选择特定的记录，然后通过系统调用智能合约中封装删除操作，即可删除掉代理重加密过后的密文结果，达成取消授权的效果。该过程于前文的流程基本相似，因此不再展示其类间调用关系的时序图。

综上所述，为系统用户的医疗电子病历数据从添加存储到申请授权查看的类之间实现的全过程。上述几个功能涉及到了核心类图设计如图 5.12 所示。

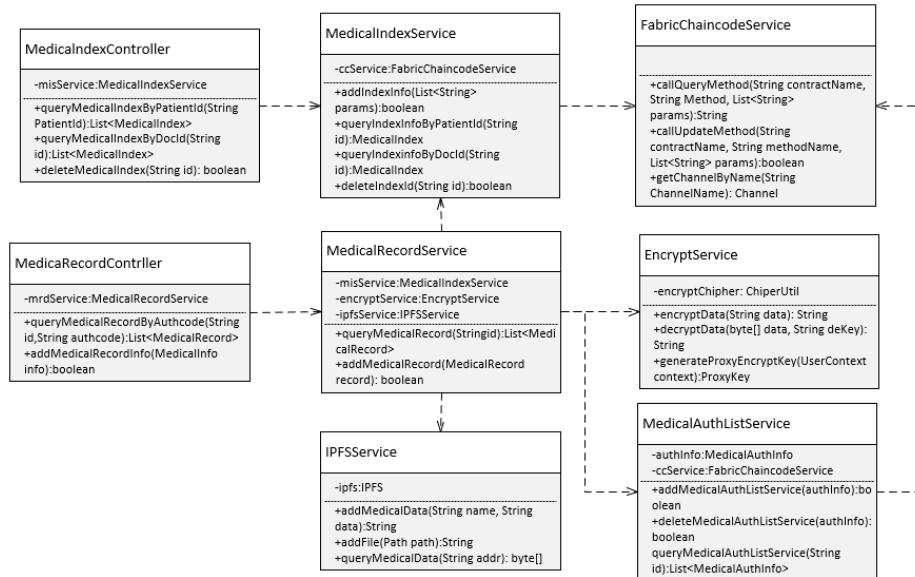


图 5.12 病历共享核心类图

在 `MedicalRecordController` 类中提供添加和查询电子病历记录的接口，供前端调用。在 `MedicalRecordService` 类是对系统病历记录的具体业务实现，主要包括查询和添加功能。为了实现对病历的操作功能，该类中依赖一些服务类来实现管理医疗电子病历功能的具体操作。例如：`EncryptService` 类就是负责提供数据的加密和解密的方法以及代理重加密密钥的生成等代理重加密算法的类。其中加密算法的核心密钥生成代码图如图 5.13 所示。首先随机生成两个椭圆曲线加密钥对，然后根据传入的 `publicKey` 信息加上注释的公式进行计算椭圆曲线上的点位信息 `point` 并依赖此来计算加密密钥，最后封装为一个 `EncryptWarpper` 类返回。

```

private EncryptWarpper encryptKeyGen(PublicKey publicKey) {
    // 生成E,V 加密钥对
    KeyPair keyPairE = curve.GenrateKeys();
    KeyPair keyPairV = curve.GenrateKeys();
    // 计算 H2(E || V)
    BigInteger h = ProxyUtil.HashToCurve(ProxyUtil.ConcatBytes(
        curve.PointToBytes(keyPairE.getPublicKey()),
        curve.PointToBytes(keyPairV.getPublicKey())));
    // 计算 s = v + e * H2(E || V)
    BigInteger s = MathUtil.BigIntAdd(keyPairV.getPrivateKey().getD(),
        MathUtil.BigIntMul(keyPairE.getPrivateKey().getD(), h));
    // 计算 (pk_A)^{e+v}
    Point point = curve.PointScalarMul(publicKey,
        MathUtil.BigIntAdd(keyPairE.getPrivateKey().getD(),
            keyPairV.getPrivateKey().getD()));
    // 取其点位信息计算 aes key
    byte[] keyByte = Util.Sha3Hash(curve.PointToBytes(point));
    Capsule capsule = new Capsule(keyPairE.getPublicKey(), keyPairV.getPublicKey(), s);
    EncryptWarpper encryptWarpper = new EncryptWarpper(capsule, keyByte);
    return encryptWarpper;
}

```

图 5.13 重加密加密算法图

代理重加密的代理密钥生成算法如图 5.14 所示，在该算法中首先生成一个椭圆曲线密钥对 keyPairX，然后根据传入的 publicKey 和 keyPairX 中私钥的信息计算出一个椭圆的点位 point，最后根据公式 $rk = sk_A * d^{-1}$ 计算重加密密钥 rk，并将其和曲线上的 N 点进行封装返回。

```
public ReKeyGenWarpper ReKeyGen(PrivateKey privateKey, PublicKey publicKey){
    //生成加密钥匙对X
    KeyPair keyPairX = curve.GenrateKeys();
    //获取 d = H3(X_A || pk_B || pk_B^{x_A})
    Point point = curve.PointScalarMul(publicKey, keyPairX.getPrivateKey().getD());
    BigInteger d = ProxyUtil.HashToCurve(ProxyUtil.ConcatBytes(
        curve.PointToBytes(keyPairX.getPublicKey()),
        ProxyUtil.ConcatBytes(curve.PointToBytes(publicKey),
        curve.PointToBytes(point)
    )));
    //计算重加密密钥: rk = sk_A * d^{-1}
    BigInteger rk = MathUtil.BigIntMul(privateKey.getD(), MathUtil.getInvert(d));
    rk.mod(curve.getN());
    //将结果封装返回
    ReKeyGenWarpper reKeyGenWarpper = new ReKeyGenWarpper(rk, curve.getN());
    return reKeyGenWarpper;
}
```

图 5.14 重加密密钥生成算法图

代理重加密的重加密算法核心代码图如图 5.15 所示，该算法一共分为两个步骤，首先根据传入 capsule 和重加密密钥 rk 判断下传入的参数是否正确，通过之后再次椭圆曲线上的乘方运算加密所需要新的点位信息并封装成一个新的 deCapsult 信息返回即可。

```
public Capsule ReEncryption(PrivateKey rk, Capsule capsule){
    //检查 g^s == V * E ^ {H2(E || V)} (公式一)
    List<BigInteger> listXY = curve.ScalarBaseMult(capsule);
    BigInteger X = listXY.get(0);
    BigInteger Y = listXY.get(1);
    List<BigInteger> tmpList = curve.ScalarMult(X, Y,
        ProxyUtil.ConcatBytes(curve.PointToBytes(capsule.getE()), curve.PointToBytes(capsule.getV())));
    List<BigInteger> addListXY = curve.add(capsule.getE(), capsule.getV(), tmpList.get(0), tmpList.get(1));
    BigInteger nX = addListXY.get(0);
    BigInteger nY = addListXY.get(1);
    //如果不相等直接返回
    if(!nX.equals(Y) && !nY.equals(Y)){
        return null;
    }
    //计算 E' = E^{rk}, V' = V^{rk} (公式二)
    Capsule deCapsule = new Capsule(curve.PointScalarMul(capsule.getE(), priKey),
        curve.PointScalarMul(capsule.getV(), priKey))
    return deCapsule;
}
```

图 5.15 重加密密钥加密算法图

代理重加密生成解密密钥的方法如图 5.16 所示，该算法首先计算出通过椭圆曲线上的标量乘法计算出点位 S，然后根据传入的解密密钥等信息进行拼接，并在曲线上再次进行点位的标量乘法进行计算，从而得出解密私钥，最后根据这个私钥解密密文即可。

```
public byte[] decryptKeyGen(PrivateKey bPrivate, Capsule capsule, PublicKey publicKey){  
    // 计算 S = X_A^sk_B  
    Point S = curve.PointScalarMul(publicKey, bPrivate.getD());  
    // 重新生成 d = H3(X_A || pk_B || S)  
    BigInteger d = ProxyUtil.HashToCurve(  
        ProxyUtil.ConcatBytes(  
            curve.PointToBytes(publicKey),  
            curve.PointToBytes(bPrivate.getCurve().getN()),  
            curve.PointToBytes(S)  
        )  
    );  
    Point point = curve.PointScalarMul(  
        curve.PointScalarAdd(capsule.getE(), capsule.getV()), d  
    );  
    byte[] keyBytes = Util.Sha3Hash(curve.PointToBytes(point));  
    return keyBytes;  
}
```

图 5.16 重加密解密密钥生成算法图

IPFSService 类的功能主要负责和 IPFS 节点进行通信，主要包括根据地址查询函数和存储加密过后的电子病历记录函数。MedicalIndexController 类和 MedicalIndexService 类主要负责查询和存储医疗电子病历的索引信息。FabricChaincodeService 类调用相应的智能合约将索引信息进行上链存储和查询等功能。MedicalAuthListService 类负责管理患者授权记录主要包括查询，添加和删除授权记录功能，并依赖 FabricChaincodeService 类实现调用智能合约的操作。

最后该功能模块还包括医疗药品信息管理功能，其功能流程图如图 5.17 所示。医疗机构的管理员负责该机构的药品管理。该功能可以加强用药规范，为医生提供精准的药品信息。以更改药品信息操作为例，管理员在药品管理页面查看医疗机构中的药品信息，然后通过更改功能填写更改的部分提交系统进行更改。系统进行权限审核之后进行实际的信息更改操作，最后将更改结果返回给用户查看。

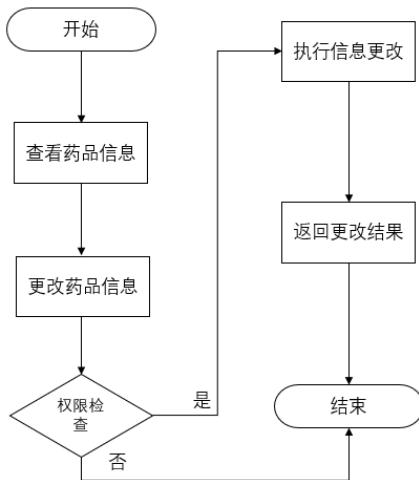


图 5.17 修改药品信息流程图

如图 5.18 所示为管理员调用药品信息更改功能涉及到的时序图。首先客户端调用 DrugController 里面的 updateDrugInfo 方法进行药品信息的更改，updateDrugInfo 方法中传入了药品的主键信息，通过这个主键信息调用 DrugService 方法中的 queryDrugInfo 方法。该方法使用 DrugRepository 类从数据库中找出对应的药品信息返回。DrugService 在修改药品信息之前调用了 UserAuthService 检查了权限，权限验证通过之后就将修改的信息覆盖查询到的信息。最后再次调用 DrugRepository 类将更新过后的信息存入数据库当中。

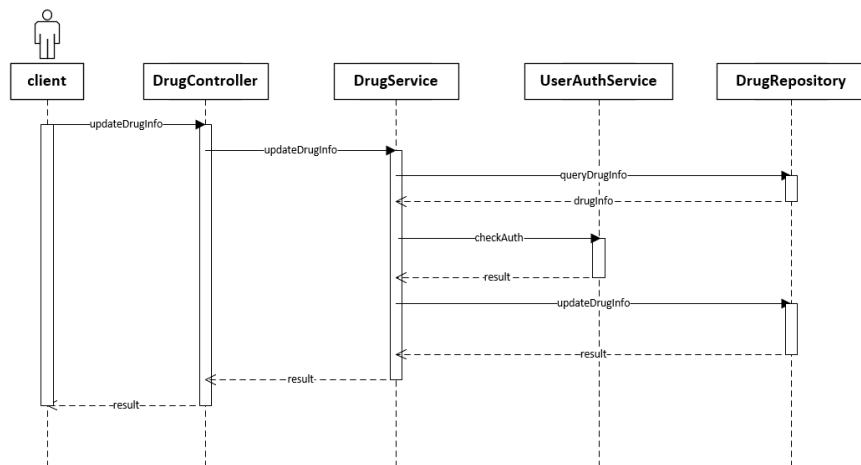
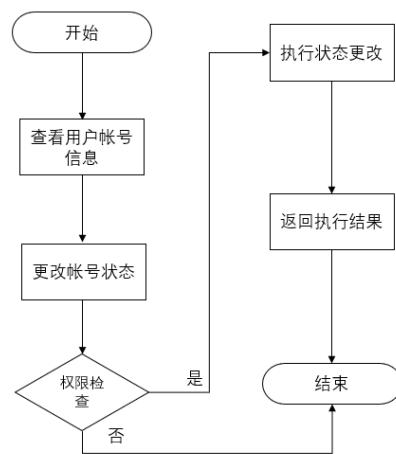


图 5.18 修改药品时序图

5.1.3 系统管理模块的详细设计与实现

在系统管理功能模块主要包括用户账户状态管理和区块链通道配置查看和修改等功能。这些功能都有有助于系统管理员维护整个系统的正常运行。

用户账户管理的功能由系统管理员进行操作，系统管理员可以查看注册在本系统的用户账户信息，并对其状态进行修改。其流程图如图 5.19 所示，管理员在账户管理页面查看注册在系统中的账户信息和状态，然后选择需要更改的账户进行更改操作，系统会执行相关操作并返回执行结果。



上述流程图中注销用户状态涉及到的时序图如图 5.20 所示。首先管理员调用 UserController 类中的 revokeUserStatus 方法执行更改用户状态的操作。在 UserController 类中调用服务层的 UserInfoService 方法执行具体的操作，在更改用户状态之前在 UserInfoService 类中还需要调用 checkAuth 方法来检验该用户的类型是否是管理员。只有通过了权限检查才能进行下一步的用户状态更改，即调用 UserInfoRepository 类中的方法更改用户信息状态，从而完成整个用户状态更改流程的操作。

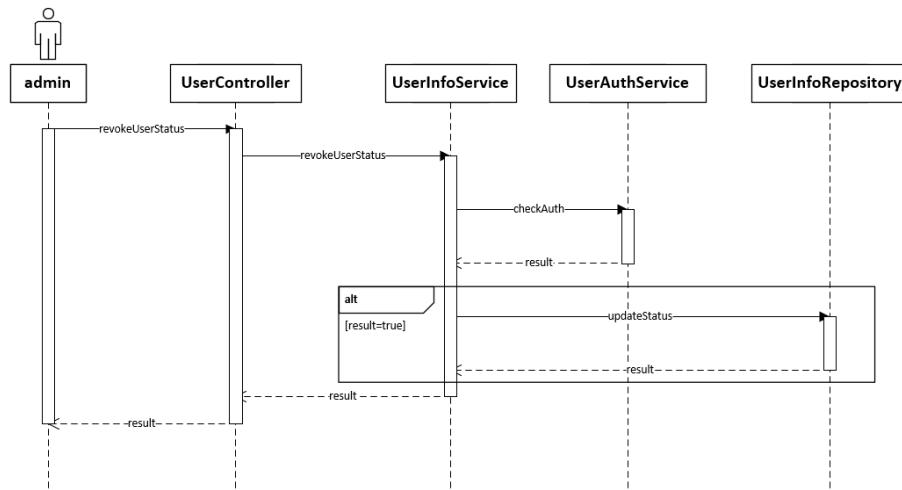


图 5.20 用户状态管理时序图

系统管理功能模块还包括区块链系统配置信息的管理，主要包括查看区块链系统内通道信息以及更改通道配置等功能。这些功能将帮助管理员更好的维护区块链网络的运行。

管理员在通道信息查看页面可以查看当前通道配置信息，其中主要包括打包区块时间，打包区块的最大信息数量等参数，这些参数会影响到区块链的吞吐性能。管理员可以在这个功能模块中修改此参数，当提交修改后的参数的时候系统会检查用户权限，只有管理员权限的账户可以更改这些配置并通过 Fabric-SDK-Java 的 client 客户端与区块链系统交互完整配置修改，并返回更改结果。其具体的流程图如图 5.21 所示。

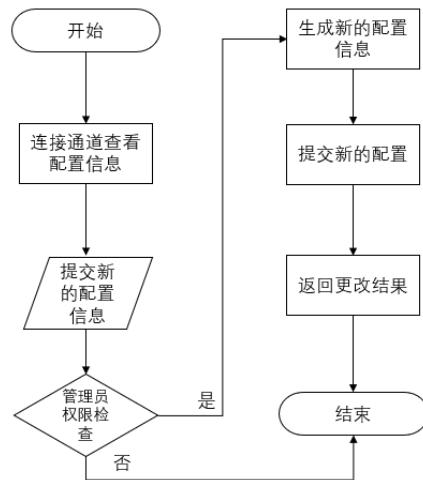


图 5.21 更新通道配置流程图

以更新通道配置为例，更新通道配置之间的序列图如图 5.22 所示，前端接口请求 FabricChannelController 类的更新通道配置接口，该接口主要向 FabricChannelService 类请求 updateChannelConfig 方法。该方法通过 HFClient 对象获得 Channel 对象，利用 Channel 对象向 FabricServer 请求通道配置。FabricServer 接到请求之后将会返回包含通道配置信息的二进制字节数组。FabricChannelService 会请求 configtxlator 服务解析这个二进制数组返回配置字符串。修改这个字符串的内容之后再次请求 configtxlator 服务将修改过后的字符串重新生成二进制配置数组。FabricChannelService 类再向 FabricServer 请求更新通道配置，完成整个过程。

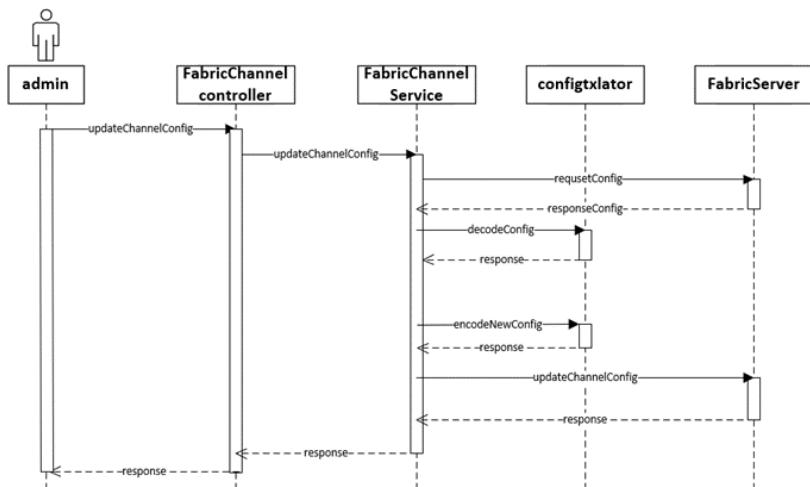


图 5.22 更新通道配置时序图

5.2 智能合约的详细设计与实现

在医疗业务功能模块调用智能合约功能是实现整个流程的不可缺少的一部分，在本系统中智能合约主要负责医疗病历索引的管理和使用重加密密钥执行代理重加密两个主要的功能。因为与系统功能模块不同，智能合约是运行在区块链系统之上的，故此在本小节中单独叙述智能合约的设计与实现以及如何在系统中与智能合约进行通信。

在 Hyperledger Fabric 项目中的智能合约里面可以对区块链中数据的管理和提供程序执行环境，当系统通过 SDK 传入响应参数的时候，则可以触发一些预先定义的操作。Fabric 目前提供了使用 Java、Go、JavaScript 等多种主流语言进行智能合约编写的功能。本文采用 Java 语言编写链码。因此，需要按照规定继承 Fabric 提供的 ChaincodeBase 类，才能被区块链节点识别为智能合约类，并且需要实现该类中的 init 方法和 invoke 方法。Init 方法主要用于在实例化链码的时候智能合约执行的操作，invoke 方法是执行调用智能合约的时候入口方法，所有其他的方法都需要在 invoke 中注册后才能够被调用。因此所有继承 ChaincodeBase 类的智能合约必须实现 init 方法和 invoke 方法，并将提供给外界调用的方法注册到 invoke 中。在本系统中设计了三个智能合约来管理在区块链中的数据。智能合约的类图设计如图 5.23 所示。

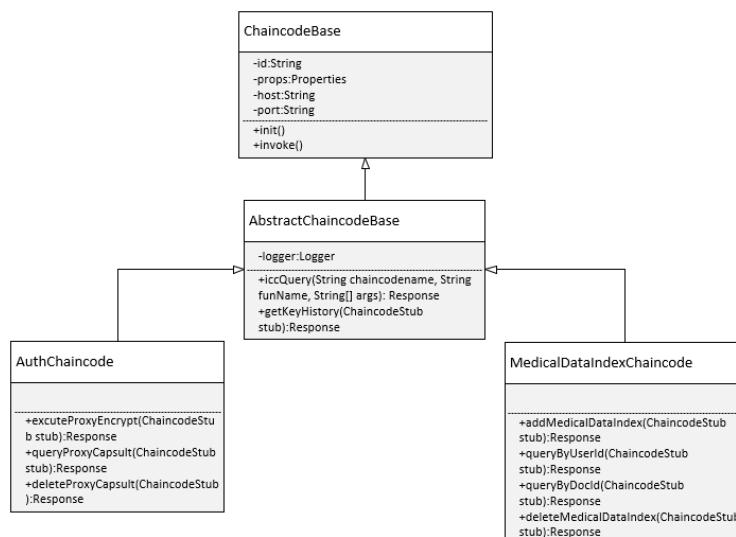


图 5.23 智能合约设计类图

由图中可以看出，本文的系统抽象除了一个 `AbstractChaincodeBase` 类作为一个抽象类按照编写链码的规定继承了 `ChaincodeBase` 接口，而其他智能则继承 `AbstractChaincodeBase` 智能合约类实现具体的系统医疗业务功能操作。其中 `AbstractChaincodeBase` 类主要负责几个常见的查询、查询键值的历史记录以及跨链码数据调用功能。这些功能与具体的业务逻辑无关，但是其他的负责业务逻辑的智能合约通过继承关系可以很好的减少通用代码的编写的工作，提升开发的效率。

在本系统中的承担业务逻辑的智能合约是继承 `AbstractChaincodeBase` 类的另外两个智能合约类。首先介绍 `MedicalDataIndexChaincode` 智能合约，因为区块链中的数据库存储容量有限，因此本文系统使用这个智能合约负责电子病历索引添加、查询和删除操作。添加操作的代码如图 5.24 所示，首先从对象 `stub` 中取出传入的参数并进行检查，如果参数个数错误则返回相应的提示，如果通过检查则取出传入的参数 `key` 和 `value` 值，最后调用 `stub` 的 `putStringState` 方法将数据存储区块链数据库中即可。

```
private Response addMedDataIndex(ChaincodeStub stub) {
    //参数检查
    List<String> parameters = stub.getParameters();
    if(parameters.size() != 2){
        return ResponseUtils.newErrorResponse(null, "parameters error".getBytes());
    }
    String key = parameters.get(0);
    String value = parameters.get(1);
    //存入区块链中
    stub.putStringState(key, value);
    return ResponseUtils.newSuccessResponse("success".getBytes());
}
```

图 5.24 添加病历索引代码示意图

如图 5.25 所示是医疗索引结构的查询代码实现过程展示，同样通过 `stub` 对象拿到传入的参数信息并进行参数检查，参数检查通过之后根据参数信息调用字符串的 `format` 方法将传入的参数信息拼接到相应的位置，组成完成的富查询语句，之后调用 `getQueryResult` 方法在区块链中查询出满足条件的数据，查询的结果是一个迭代器对象。最后通过返回的迭代器，利用循环语句将查询的结果进行拼接，然后调用 `ResponseUtils` 的 `newSuccessResponse` 方法将结果返回。

```

private Response queryDataIndexInfo(ChaincodeStub stub) {
    //参数检查
    List<String> parameters = stub.getParameters();
    if(parameters.size() != 1){
        return ResponseUtils.newErrorResponse("parameter Error".getBytes());
    }
    //拼接查询语句
    String queryStr = null;
    QueryResultsIterator<KeyValue> queryResult = null;
    String id = parameters.get(0);
    queryStr = "{ \"selector\": { \"owner\": \"%s\" } }";
    queryResult = stub.getQueryResult(String.format(queryStr, id));
    //拼接结果返回
    StringBuilder queryRes = new StringBuilder();
    Iterator<KeyValue> iterator = queryResult.iterator();
    while (iterator.hasNext()){
        queryRes.append(iterator.next() + "+");
    }
    return ResponseUtils.newSuccessResponse(queryRes.toString().getBytes());
}

```

图 5.25 查询病历索引代码示意图

另外一个智能合约就是病历授权智能合约，该智能合约的功能主要是根据传入的代理重加密密钥执行代理重加密。为了实现该功能同样需要引入前文提到的代理重加密算法开源库。该智能合约的 `excuteProxyReEncryption` 方法负责调用前文提到开源库中提供的对象接口执行代理重加密，其具体的代码示意如图 5.26 所示。从图中可以看出首先通过 `stub` 取出传入的参数，然后使用 `Proxy` 对象的 `ReEncryption` 方法利用传入的代理重加密的密钥(`ReEncryptKey`)生成 `DeCapsule` 解密信息的过程。其中 `DeCapsult` 封装的是椭圆曲线的上点位信息，被授权用户可以根据这些点位信息利用自己的私钥来解密索引密文。根据代理重加密的安全模型^[51]，没有被授权的用户无法通过这些点位信息攻击数据，从而有效的保护了用户的数据隐私。然后，智能合约调用 `getAuthCode` 函数按照一定的规则生成一个授权码作为 `key` 调用 `stub` 对象的 `putState` 方法存储到区块链中，最后并返回该授权码即可。

```

private Response proxyReEncryption(ChaincodeStub stub) {
    List<String> parameters = stub.getParameters();
    if(parameters.size() != 2){
        return ResponseUtils.newErrorResponse(null, "parameters error".getBytes());
    }
    String rk = parameters.get(0);
    String capsule = parameters.get(1);
    //执行代理重加密
    Object nCapsule = proxy.ReEncryption(rk, capsule);
    String authCode = getAuthCode();
    stub.putState(authCode, nCapsule);
    return ResponseUtils.newSuccessResponse(authCode.getBytes());
}

```

图 5.26 执行代理重加密代码示意图

被授权的用户可以通过这个授权码调用该智能合约中的 queryProxyDeCapsule 方法来找到所需解密的 DeCapsule 信息，并利用里面包含椭圆曲线点位信息，调用解密算法获取数据明文。这个过程本质上就是一个根据 key 查询 value 的过程，主要就是通过 stub 的 getState 方法通过 authcode 信息找到所需的 result。被授权的用户拿到这个信息使用自己的私钥即可解密密文数据，其代码如图 5.27 所示。在智能合约中还提供了删除点位信息的方法，利用 stub 对象的 deleteState 方法传入索引进行删除，其流程和代码示意图与前文类似，在此不在赘述。

```
private Response queryProxyDeCapsule(ChaincodeStub stub) {
    //检查参数
    List<String> parameters = stub.getParameters();
    if(parameters.size() != 1){
        return ResponseUtils.newErrorResponse(null, "parameters error".getBytes());
    }
    String authcode = parameters.get(0);
    //取出value
    String result = stub.getState(authcode);
    if(result == null){
        return ResponseUtils.newErrorResponse("no result".getBytes());
    }
    return ResponseUtils.newSuccessResponse(result.getBytes());
}
```

图 5.27 查询代理重加密结果示意图

综上所述，就是本系统中的智能合约主要的功能设计和实现描述。在区块链公开的数据库中索引数据全程保持密文状态，达到了数据安全可信共享的目标。

最后，在本文的系统中使用 Fabric 官方提供的 Fabric-Gateway-Java 项目调用智能合约与区块链网络进行通信的内容。智能合约实现完成之后，本系统需要在应用层上使用相应的 SDK 对智能合约进行调用。在本系统中使用 Hyperledger Fabric 官方提供的 Fabric-Gateway-Java 项目和 Fabric 网络上的智能合约进行交互，Gateway 项目相较于传统的 SDK 提供了更为简单的接口调用智能合约上的函数。如何连接 Gateway 对象的代码如图 5.28 所示，首先必须拥有一个 Identity 身份对象，Identity 对象是通过读取组织的证书信息，然后通过 Wallets 对象创建获得的，该对象本质上就是一个符合 X.509 标准的证书信息。其中 Identity 对象可以放在 Wallet 对象里面保存。连接 Gateway 还需要区块链网络的组织拓扑信息，主要就是填写区块链网络的组织和证书信息，便于创建 Gateway 对象的时候

读取。在配置了相关的信息之后调用相关 API 就可以获取一个封装好的 Gateway 对象了。通过 Gateway 对象可以获取 Fabric 网络中的通道和智能合约对象，通过合约对象可以选择调用该智能合约的那个方法，并且将参数一起提交到 Fabric 网络中去，智能合约会返回相应的结果。这样就实现了使用 Gateway 与 Fabric 交互的过程。

```
public Gateway getGateway() throws IOException {
    //创建Wallet对象存放连接网络的对象
    Wallet wallet = Wallet.createInMemoryWallet();
    .....
    //获取证书信息之后，生成连接网络的对象
    Wallet.Identity identity = Wallet.Identity.createIdentity(certInfo);
    wallet.put(identity);
    .....
    //使用建造者模式创建连接对象，并配置网络信息
    Gateway.Builder builder = Gateway.createBuilder();
    builder.identity(IdentityConfig);
    builder.networkConfig(connectConfig);
    //...省略其他配置
    Gateway gateway = builder.connect();
    .....
    //最后返回结果
    return gateway;
}
```

图 5.28 链接区块链网络伪代码示意图

5.3 区块链网络搭建

在本小节主要根据前面叙述的系统架构设计来搭建 Fabric 网络。首先，需要搭建起 Fabric 网络首先需要准备证书生成配置文件，本文使用 Fabric 官方提供的 cryptogen 工具为 Orderer 节点和 Peer 节点生成证书文件。这些证书文件拥有代表 Orderer 节点和 Peer 节点的身份、交易签名以及验签等功能。而生成证书文件需要依赖 crypto-config.yaml 文件，这个文件包含了网络的拓扑结构。cryptogen 工具可以根据这个文件上的网络拓扑信息为每个 Peer 节点和 Orderer 节点生成对应的证书文件。在本系统中其配置文件的具体内容结构如图 5.29 所示，图中的配置表示创建两个组织，每个组织维护两个 peer 节点，同时创建两个排序节点，组成区块链网络。

```

OrdererOrgs:
  - Name: Orderer
    Domain: example.com
    Specs:
      - Hostname: orderer
      - Hostname: orderer2

PeerOrgs:
  - Name: Org1
    Domain: org1.example.com
    EnableNodeOUs: true
    Template:
      Count: 2
    Users:
      Count: 1
  - Name: Org2
    Domain: org2.example.com
    EnableNodeOUs: true
    Template:
      Count: 2
    Users:
      Count: 1

```

图 5.29 生成证书配置文件图

在生成相关证书之后，本系统需要生成交易相关的文件。这些文件主要是通过使用 `configtxgen` 工具生成 Orderer 节点的 `genesis.block` 文件、组织之间的 `channel.tx` 文件以及组织内锚节点的 `anchor.tx` 文件。这些文件是 Fabric 网络启动时的初始化文件。在启动之前必须指定锚节点负责组织的对外通信。锚节点配置文件包含了对外通信相关的配置。生成这些相关文件需要依赖 `configtx.yaml` 文件，这个文件主要包含了上述信息的配置。例如：在 Fabric 提供的多种共识算法在这里指定、节点服务地址与端口号、通道内区块的批处理信息等。最后利用 `configtxgen` 工具指定设置的 `configtx.yaml` 的信息，即可生成前面提到的配置文件。如图 5.30 所示为 `configtx.yaml` 中通道配置的模板和相应生成的配置文件。

```

OrgsChannel1:
  Consortium: MedicalConsortium
  <<: *ChannelDefaults
  Application:
    <<: *ApplicationDefaults
    Organizations:
      - *Org1
      - *Org2
    Capabilities:
      <<: *ApplicationCapabilities

```

```

ubuntu@ubuntu:~/fabricNetwork/channelArtifact$ tree
.
├── channel.tx
├── genesis.block
└── Org2MSPanchors.tx
└── Org1MSPanchors.tx

```

图 5.30 通道配置模板和配置文件示意图

当组织证书和通道配置文件都准备完毕之后就可以通过 `docker` 启动相关的容器。为了提高启动容器效率，本文使用 `docker-compose` 工具来启动区块链网络上所有的需要启动的节点。该工具通过编辑容器编排配置文件 `docker-`

compose.yaml 文件配置节点网络端口映射关系和相关证书文件的挂载路径，然后通过启动命令启动相关镜像即可完成搭建节点网络工作。当区块链网络成功启动后可以使用 docker 的 ps 命令即可查看如图 5.31 所示的 Fabric 节点容器网络。

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED	STATUS
82eb9bf9be08	hyperledger/fabric-tools:latest	"/bin/bash"	cli	7 seconds ago	Up 5 seconds
4fa75fb5af95	hyperledger/fabric-peer:latest	"peer node start"	peer0.org1.example.com	9 seconds ago	Up 6 seconds
296c32f3771c	hyperledger/fabric-peer:latest	"peer node start"	peer0.org2.example.com	10 seconds ago	Up 8 seconds
8f36c738c387	hyperledger/fabric-peer:latest	"peer node start"	peer1.org1.example.com	11 seconds ago	Up 7 seconds
0adeec13184d	hyperledger/fabric-peer:latest	"peer node start"	peer1.org2.example.com	11 seconds ago	Up 8 seconds
1d71e08cce259	hyperledger/fabric-couchdb	"tini -- /docker-ent..."	couchdb3	13 seconds ago	Up 10 seconds
a188644f7c99	hyperledger/fabric-couchdb	"tini -- /docker-ent..."	couchdb0	13 seconds ago	Up 8 seconds
dcc66231540f	hyperledger/fabric-orderer:latest	"orderer"	orderer.example.com	13 seconds ago	Up 9 seconds
0226cbdbfa84	hyperledger/fabric-couchdb	"tini -- /docker-ent..."	couchdb2	14 seconds ago	Up 9 seconds
40ef0b0216f1	hyperledger/fabric-couchdb	"tini -- /docker-ent..."	couchdb1	14 seconds ago	Up 10 seconds

图 5.31 区块链网络启动示意图

5.4 IPFS 私有化集群搭建

为了解决区块链中不适宜存储大容量数据的特性和共识算法消耗资源大的问题，IPFS 作为一个分布式文件系统可以很好的和区块链相互补充。而在联盟链环境中，采取私有化的 IPFS 节点集群更为合适。私有化的 IPFS 集群只会连接拥有共同密钥(swarm.key)的其他 IPFS 对等节点，而不会响应其他网络环境下的节点的通信请求。如图 5.32 所示为 IPFS 私有化集群示意图。

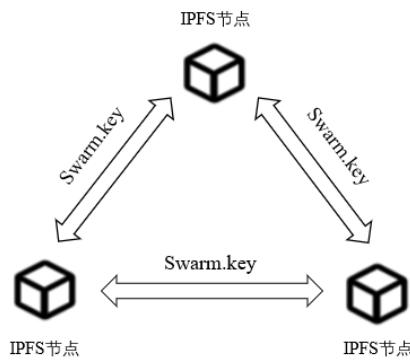


图 5.32 IPFS 节点集群示意图

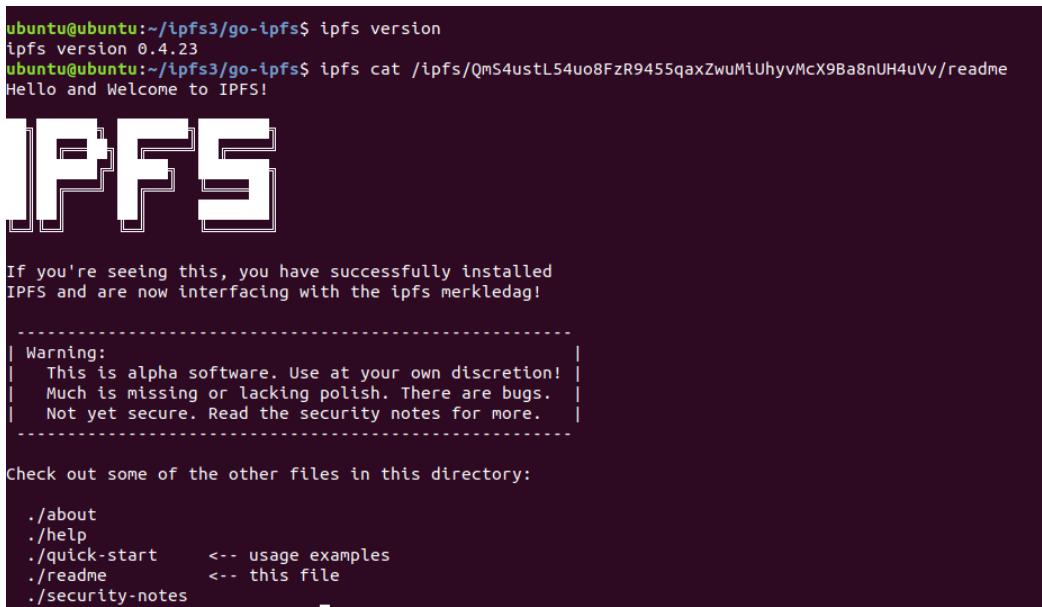
首先，需要安装 IPFS 节点。IPFS 官方提供了的 IPFS 多种安装方案，在本小节中，我们将采用 ipfs-update 的方式介绍如何搭建一个私有化的 IPFS 节点，存储系统中的数据。创建 IPFS 私有节点集群的方式并不复杂，首先需要安装 go 环境，之后就可以使用 go 的 get 命令安装 ipfs-update。当安装好 ipfs-update 之后就可以使用 ipfs install 命令来下载指定的 ipfs 版本。

完成了 IPFS 的下载之后，想要组建节点私有集群还需要下载并使用 IPFS 官方提供的 swarm-key 工具创建共享的密钥 swarm.key，并将这个共同密钥共享给其他的 IPFS 节点中的配置文件夹中，这样将保证在 IPFS 节点启动的时候只响应 swarm.key 相同的节点，换句话说就是拥有相同的 swarm.key 的 IPFS 节点之间组建了私有化集群，这个步骤也是搭建 IPFS 私有化集群的关键。如图 5.33 所示为创建的 swarm.key 示意图。

```
ubuntu@ubuntu:~/go/bin$ ls
ipfs-swarm-key-gen  swarm.key
ubuntu@ubuntu:~/go/bin$ cat swarm.key
/key/swarm/psk/1.0.0/
/base16/
47bf1d572486ada3d9196dc4fdc747ada902f4227bf4e6ec74c968d03e0cf2b
```

图 5.33 swarm.key 示意图

最后，需要使用 ipfs 的 rm 命令移除默认配置的公网节点。接着通过 systemctl 命令启动 IPFS 后台进程，完成 IPFS 私有化节点的搭建过程。如图 5.34 所示为 IPFS 节点搭建成功后看到的示意图。最后在不同的服务器上重复同样的过程，并通过 ipfs 提供的 bootstrap 命令，将第一个节点的 Id 设置为 bootstrap 节点即可完成 IPFS 私有化集群的搭建。



```
ubuntu@ubuntu:~/ipfs3/go-ipfs$ ipfs version
ipfs version 0.4.23
ubuntu@ubuntu:~/ipfs3/go-ipfs$ ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme
Hello and Welcome to IPFS!
IPFS
-----
If you're seeing this, you have successfully installed
IPFS and are now interfacing with the ipfs merkledag!
-----
| Warning:
| This is alpha software. Use at your own discretion!
| Much is missing or lacking polish. There are bugs.
| Not yet secure. Read the security notes for more.
-----
Check out some of the other files in this directory:
./about
./help
./quick-start    <-- usage examples
./readme        <-- this file
./security-notes
```

图 5.34 IPFS 节点启动示意图

5.5 本章小结

本章主要叙述了系统的各个功能模块的详细设计与实现，本章节也是论文的核心内容。在本章开头首先描述了系统电子病历共享模型，然后根据前面章节将系统功能划分的用户信息管理模块、医疗业务功能模块以及系统管理模块三个功能模块里面包含的功能点进行了叙述，具体细节主要利用了流程图、时序图以及类图等方式对系统功能逻辑的实现进行了详细的解释和说明。然后，本章单独叙述了运行在区块链上的智能合约的详细设计思路，并通过展示部分代码解释了具体实现的细节。最后叙述了区块链网络环境的搭建以及 IPFS 私有化节点集群的搭建的过程。综上所述，整个的区块链网络加上 IPFS 私有节点集群以及系统的业务功能共同组成了本文系统。

第六章 系统测试

本章对基于区块链的数据共享交换平台进行一系列的测试。测试内容主要包括功能测试和性能测试。其中功能测试主要用来保障开发平台的正常运行。测试中的性能测试使用 Hyperledger Caliper 工具来测试本平台系统的性能指标，并对测试结果进行了分析。

6.1 测试环境

本文的系统部署在云服务器上，区块链网络节点采用的是单机的伪分布式部署方式。IPFS 节点部署在相同的配置的另一台服务器上。服务器的配置如表 6.1 所示。

表 6.1 系统测试环境表

项目	配置
CPU	2 核
内存	4GB
操作系统	Ubuntu 18.04
软件环境	Hyperledger Fabric 1.4.4 Golang 13.1, Java 1.8 Docker 19.03, Docker-compose 19.03, IPFS 0.4.23

6.2 功能测试

系统的功能测试是系统开发过程中的极为重要的一个环节，它主要用来检测系统的可用性、完整性。为了检测本系统的功能的可用性以及完整性，本文采用

黑盒测试的方式通过编写相应的测试用例表来对系统的主要功能进行测试，并通过相应的系统的运行页面展示测试的结果。

6.2.1 用户信息管理模块测试

本功能模块主要测试系统的登录注册功能和个人信息展示功能。如表 6.2 所示为本功能模块的测试用例表。

表 6.2 用户管理模块测试用例表

测试编号	测试目的	测试内容	预期结果	测试结果
A-1	测试用户能否注册	按照用例输入符合规范用户的参数信息	新用户注册成功	通过
A-2	测试用户能否注册	按照用例要求输入错误的信息	新用户创建失败，提醒参数错误	通过
A-3	测试注册用户能否登录	按照用例输入注册成功的用户信息	成功登入系统	通过
A-4	测试能否查看个人信息	登录系统之后进入个人信息界面	展示注册个人信息	通过
A-5	测试能否修改个人信息	在个人信息页面更改个人信息数据	成功更新个人基本信息数据	通过

如图 6.1 所示为系统的登录页面，点击注册按钮即可跳转系统注册页面如图 6.2 所示。在该页面中填写相应注册信息，即可向系统发送注册请求，当系统校验注册信息符合规范时则成功为系统添加一名用户，并登录并使用系统功能。



图 6.1 系统登录示意图



图 6.2 系统注册页面示意图

在登录系统之后，用户可以在本模块中查看和修改个人信息展示，如图 6.3 所示为系统的个人信息查看页面，在该页面用户可以查看本用户的个人信息，主要包括工号、职位、科室、擅长诊治领域和一些基本的个人与联系信息，点击下方的修改按钮可以修改相应的信息项。



图 6.3 查看个人信息示意图

6.2.2 医疗业务功能模块测试

本功能模块主要测试患者登录使用的就诊挂号功能、病历查看和授权以及查看授权记录等功能和医生登录使用的查看挂号病人列表、添加病历、申请病历查看以及查看申请记录等功能。本功能模块的测试用例表如表 6.3 所示。

表 6.3 医疗业务功能用例测试表

测试编号	测试目的	测试内容	预期结果	测试结果
B-1	测试患者能否选择医生进行挂号	在挂号界面选择医生进行挂号处理	成功执行挂号操作	通过
B-2	测试医生能否查看挂号病人	在挂号界面查看成功挂号的病人	成功查看挂号的病人	通过
B-3	测试医生为患者添加病历数据	在查看病人界面点击添加病历填入相关信息	成功为患者添加病历数据	通过

表 6.3 (续)

测试编号	测试目的	测试内容	预期结果	测试结果
B-4	测试患者授权医生查看以往的病历数据	在病历管理页面选择病历信息点击授权按钮	成功返回解密所需授权码	通过
B-5	测试医生输入授权码之后，查看解密的医疗数据	在病历查看界面输入授权码之后，查看解密后的病历	成功查看解密的病历数据	通过
B-6	测试系统能否查看药品信息	在药品信息查看界面，查询药品信息	成功查询到药品数据	通过
B-7	测试系统能否更新药品数据	在药品信息查询页面，点击更改按钮，更改药品信息	成功修改药品数据	通过

如图 6.4 所示为用户的挂号列表展示图。患者可以在该页面选择相应的科室医生点击挂号按钮进行挂号操作，成功挂号之后患者信息就会出现在医生的挂号列表中，等待医生进行下一步的处理。

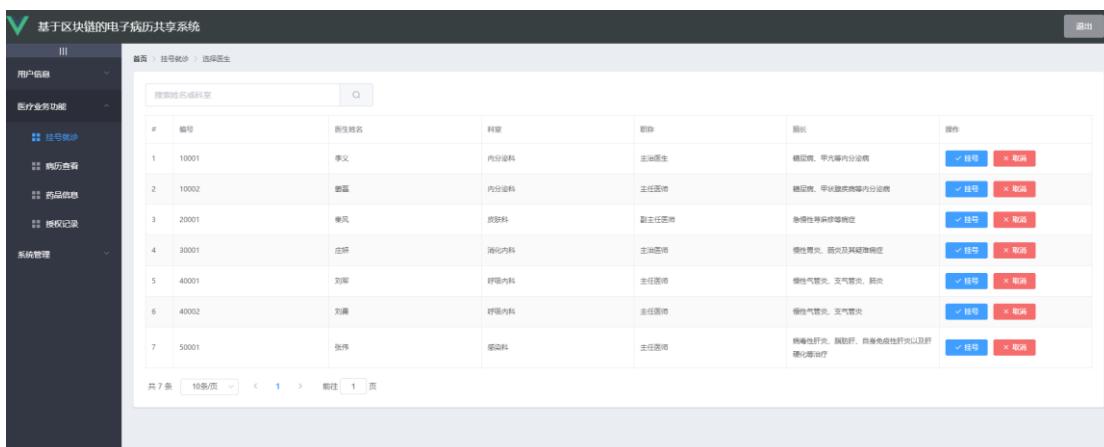


图 6.4 挂号示意图

医生在挂号列表页面中处理则跳转至病历列表页面，在该页面可以医生可以看到当前患者以往的历史电子病历的非加密介绍信息。点击右侧的申请按钮会弹出对话框要求输入授权码，当输入正确的授权码之后，就可以查询到患者授权的病历重加密结果，系统将其解密后进行展示。如图 6.5 所示为医生输入授权码的页面展示。

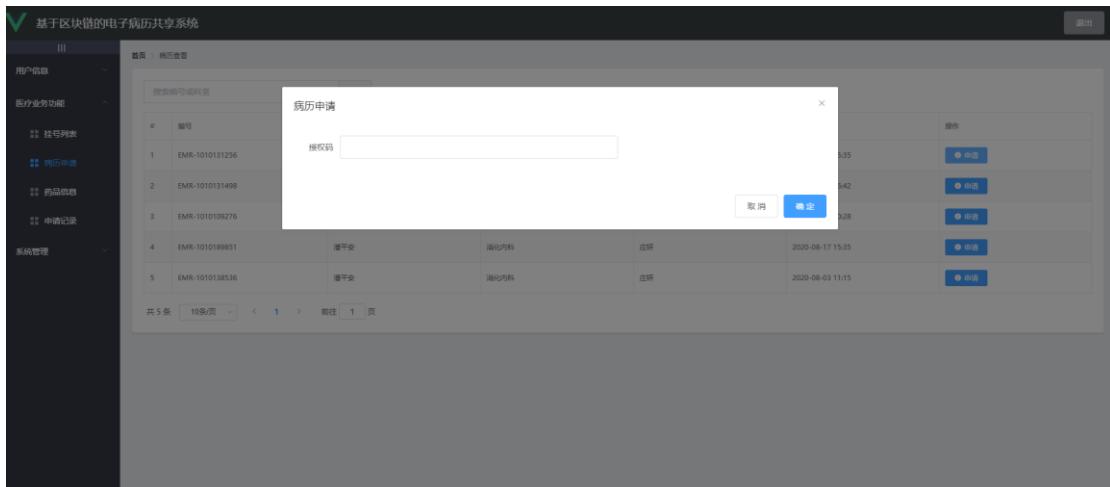


图 6.5 医生申请查看病历示意图

如图 6.6 所示，患者可以在病历查看页面查看当前系统内关于自己的病历列表。在该页面患者可以选择某条病历信息点击旁边的授权按钮，并在弹窗中输入医生编号。然后系统调用相关功能对医生进行授权，并返回一个授权码和授权记录。如上文所述该授权码可以提供给医生查询代理重加密的结果，并解密病历密文。

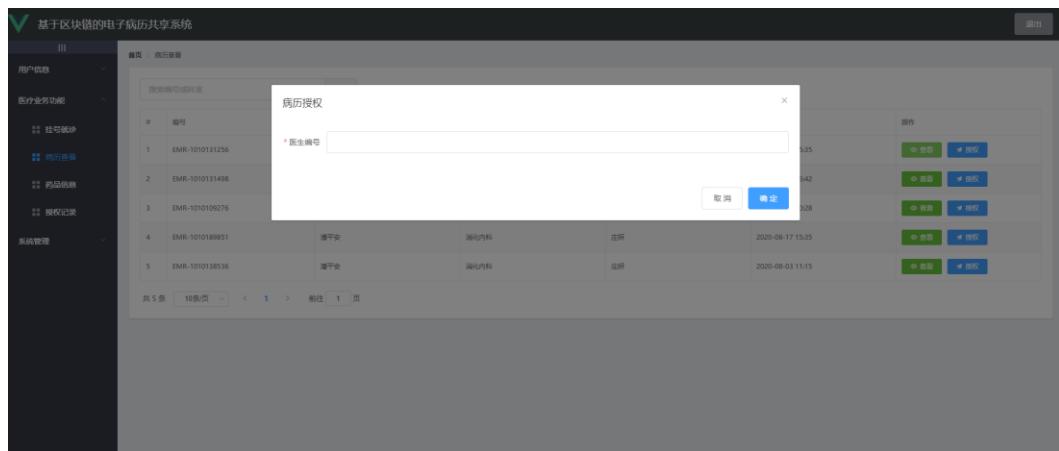


图 6.6 患者授权医生示意图

医生还可以在挂号列表中选择对应的患者，然后点击添加病历按钮则会弹出添加病历的窗口，医生在该界面对输入对患者的本次就诊的针对信息，主要包括主诉信息、初步诊断、处理意见、注意事项等内容。当完成相应信息的输入之后点击保存将该数据保存到系统中。如图 6.7 所示为医生为患者添加病历的展示图。

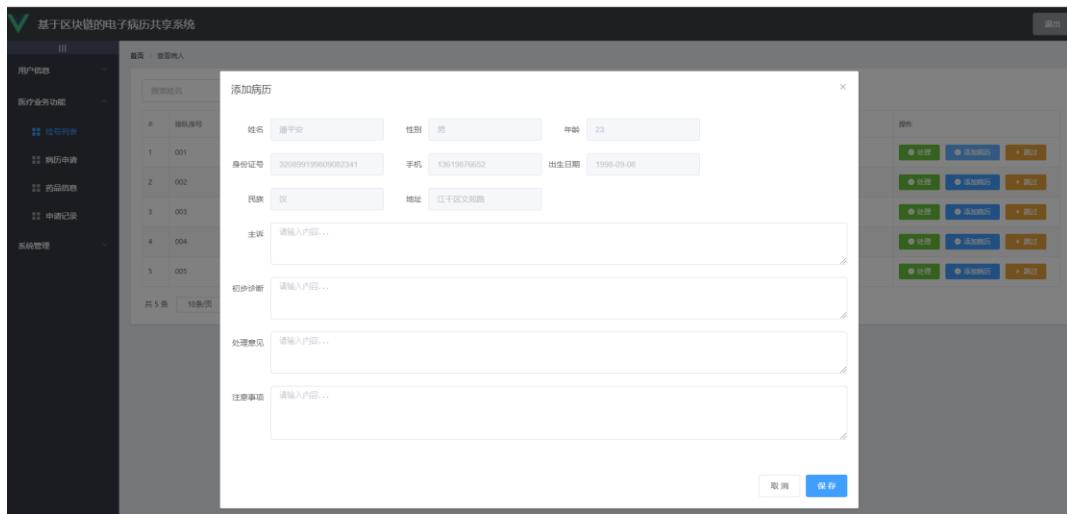


图 6.7 医生添加病历示意图

如图 6.8 所示为患者的授权记录页面展示。患者可以在授权记录页面查看自己所有授权的病历信息，并可以点击列表旁的删除按钮，删除某条授权记录信息。如果删除某条记录信息，医生将不能通过该授权码查询到重加密的结果，将无法解密数据。

#	病历编号	就诊姓名	科室	门诊	授权时间	授权码	操作
1	EMR-1010131256	李义	内分泌科	主治医师	2020-10-10 10:45	E1010090809	<button>删除</button> <button>修改</button>
2	EMR-1010131498	张丽	内分泌科	主治医师	2020-10-10 10:45	E1010099867	<button>删除</button> <button>修改</button>
3	EMR-1010109276	李义	内分泌科	主治医师	2020-10-10 10:45	E1010090980	<button>删除</button> <button>修改</button>
4	EMR-1010188851	李义	内分泌科	主治医师	2020-10-10 10:45	E1010093651	<button>删除</button> <button>修改</button>
5	EMR-1010138536	李义	内分泌科	主治医师	2020-10-10 10:45	E1010098914	<button>删除</button> <button>修改</button>

图 6.8 管理病历授权示意图

医生可以在申请记录页面查看自己以往申请的病历记录，点击列表右侧的查看按钮即可再次查看其病历内容，无需再次申请，方便医生随时查看申请过的病历信息，但是无法更改其内容。如图 6.9 所示为医生查看申请的病历记录示意图。

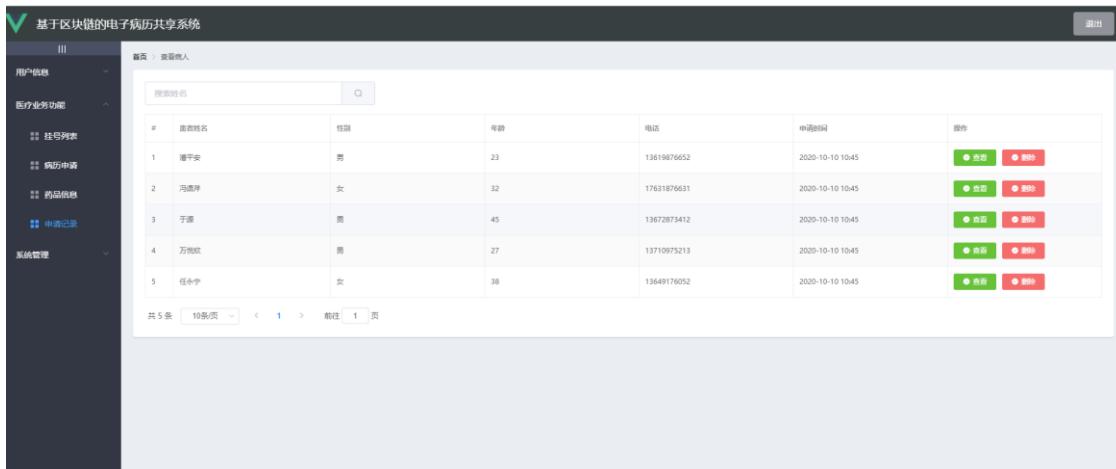


图 6.9 查看申请病历示意图

如图 6.10 所示为药品信息页面展示，为规范医疗用药和给医生提供准确的用药参考，系统提供精准的药品信息查询，点击列表的详情可以更为详细的药品规格信息，医生可以通过该页面搜索对应的药品信息并给患者对症下药。

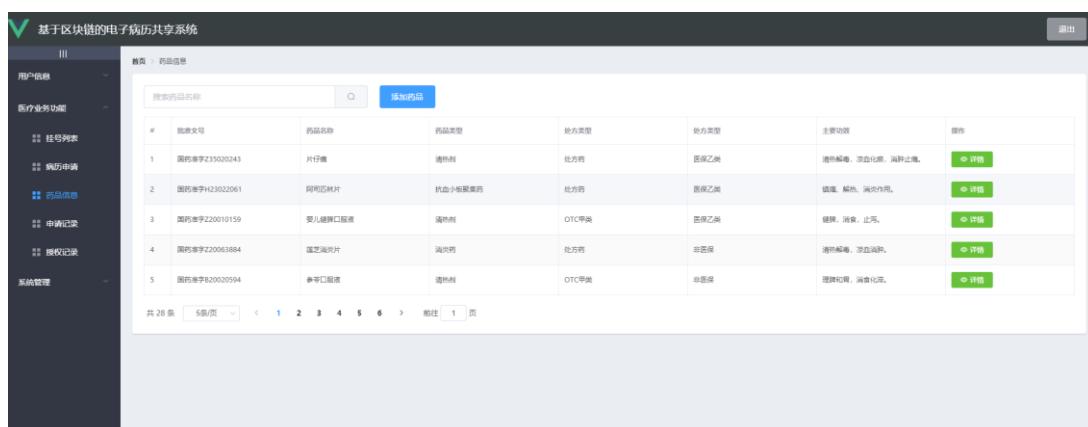


图 6.10 查询药品信息示意图

6.2.3 系统管理功能测试

本功能模块主要测试系统管理对系统的维护功能，主要包括管理员对系统内注册用户的管理和对区块链系统通道的配置的查看和修改。本功能模块的测试用例表如表 6.4 所示。

表 6.4 系统管理模块测试用例表

测试编号	测试目的	测试内容	预期结果	测试结果
C-1	测试管理员对系统内账户的注销功能	在用户管理界面选择用户进行点击注销	成功更改账户状态，但仍存在该账户在列表	通过
C-2	测试管理员对系统内账户的删除功能	在用户管理界面选择用户点击删除	成功删除账户，账户列表不再显示	通过
C-3	测试管理员能否查看区块链通道配置信息	在通道查看页面当前区块链中通道配置信息	成功查看系统内通道配置信息	通过
C-4	测试管理员能否更改通道配置	在通道查看页面点击修改，更改超时时间选项为 2s	成功修改通道配置	通过

如图 6.11 所示为系统用户管理页面，系统管理员可以在该页面查看已经注册到本系统的所有用户信息，点击注销即可更改其状态信息。除此之外，在通道管理页面可查看当前通道配置信息，并点击修改按钮进行修改通道配置的界面如图 6.12 所示。

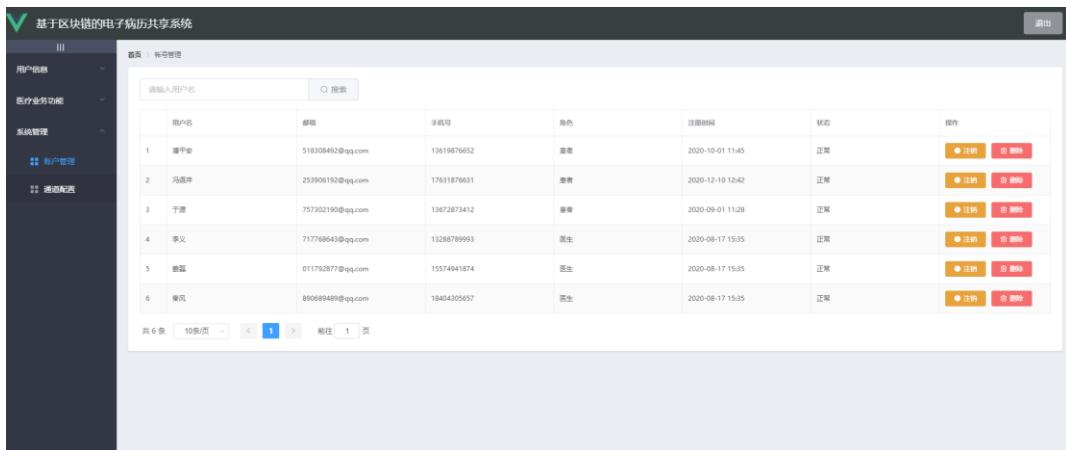


图 6.11 管理员管理系统账户示意图

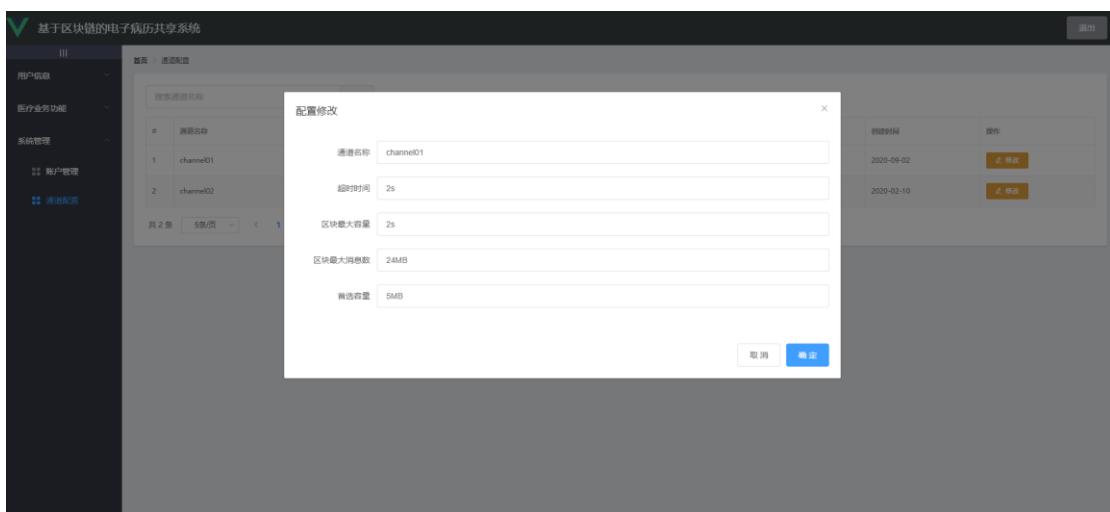


图 6.12 管理员更改通道配置示意图

6.3 性能测试

在基于 Hyperledger Fabric 框架的系统中，其智能合约的查询和更新是制约系统性能的关键因素，因此在本文主要使用 Hyperledger 项目官方提供的 Caliper^[52] 工具对系统内的智能合约这样两项指标进行测试。在本系统中涉及到更新操作指的是添加病历索引和存储代理重加密过程中的 capsule 等功能，查询操作主要包括查询病历索引和代理重加密的结果等功能，其对应的表格如表 6.5 所示。

表 6.5 更新和查询类型对应函数说明表

更新	查询
ExecteProxyEncrypt 方法	QueryProxyEncrypt 方法
AddMedicalDataIndex 方法	QueryByUseId 方法
DeleteProxyCapsule 方法	QueryByDocId 方法

在本小节中我们通过 Caliper 固定发送交易数 6000 次，通过改变发送的交易频率 (Send Rate)，进行多轮测试后得到系统的平均时延以及吞吐量两个测试的结果，并借此来分析系统的性能。其中查询涉及到的函数平均值的性能结果折线表如图 6.13 所示。

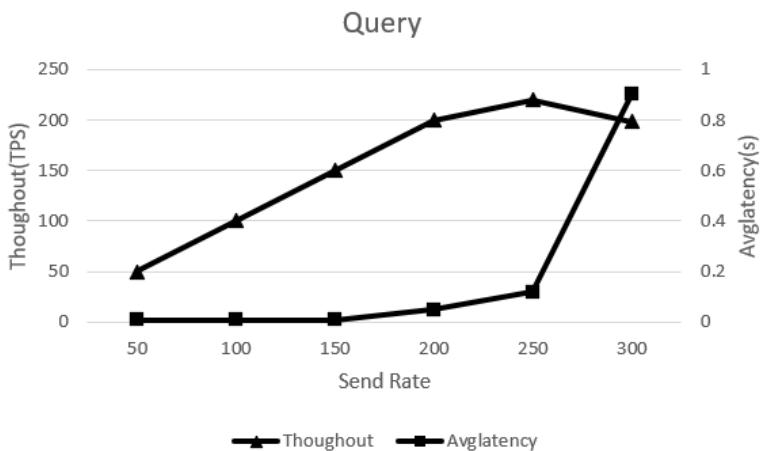


图 6.13 链码查询性能结果图

从图中可以看出随着 SendRate 数值的增加，吞吐量和平均时延都有所增加。其中当发送频率达到 250TPS 时，系统吞吐量达到了最大约 220TPS，而系统平均时延则是随着发送频率的增大而变大，且在超过 250TPS 的时候延迟上升极为明显，说明在超过 250TPS 的时候超过了系统可以处理的极限导致交易阻塞情况大为增加。

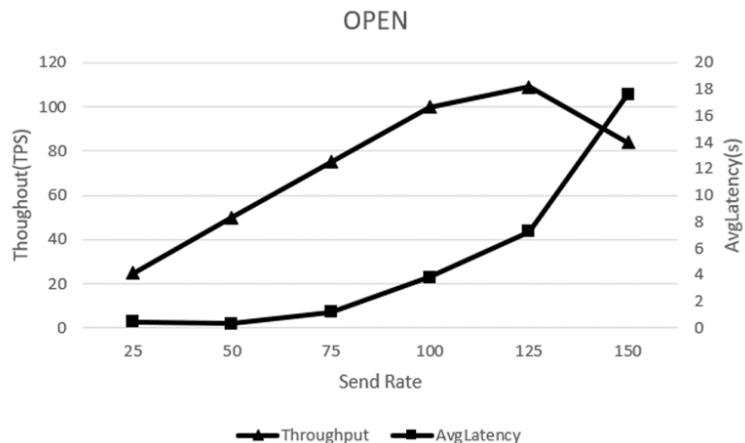


图 6.14 链码更新性能结果图

如图 6.14 所示是系统的更新性能的折线图，从图中可以看出更新函数平均值性能和发送频率的关系。在这个图中在发送频率在 25TPS 的时候系统平均延时要比发送频率 50TPS 略小，可能是因为系统等待签名相应的时间在发送频率比较小的时候占用了更多的时间。之后随着发送频率的不断增大，当发送频率达到 125TPS 的时候系统的吞吐量达到了最大约 103TPS。当发送频率进一步增大的时候系统吞吐量反而减小，并且系统平均时延进一步增大，这种情况说明 Fabric 节点处理能力在发送频率在 100TPS 左右达到了最大。

Caliper 的测试结果还提供了节点使用资源的情况，本文选取吞吐量达到最大的查询和更新的测试结果进行分析。其中查询性能测试的是数据的查询性能，更新世界状态性能测试的是添加数据的增添性能。

如表 6.6 所示是节点组件在更新世界状态时候占用资源的测试结果表。在本次测试过程中，Peer 类型的节点占用内存大约在 200MB 左右，Orderer 类型节点占用的内存资源稍小，大约在 100MB 以下。Peer 类型的 CPU 平均使用率大约在 60% 左右，Orderer 类型节点平均 CPU 使用率比 Peer 类型低，大约在 15% 左右。除此之外，Peer 类型的磁盘写入量大概在 50MB 左右，Orderer 类型的节点写入量大约在 35MB 左右。

表 6.6 更新资源使用情况说明表

节点组件	内存平均占用 (MB)	CPU 平均使用率 (%)	磁盘写入量 (MB)
Peer1 节点	172. 3	59. 19	50. 3
Peer2 节点	225. 4	60. 44	50. 3
Peer3 节点	183. 1	59. 46	50. 3
Peer4 节点	187. 9	59. 27	50. 3
Orderer 节点	64. 1	15. 38	35. 8

如表 6.7 所示是查询性能的结果表，查询性能的结果，更新能结果大部分一致。但是，依据 Fabric 框架的交易流程，查询是直接查询连接的节点的信息，没有共识过程，所以 Orderer 节点基本不占用 CPU。最后查询也不存在磁盘写入量。

表 6.7 查询测试资源使用情况说明表

组件	内存平均占用 (MB)	CPU 平均占用 (%)	磁盘写入量 (MB)
Peer1 节点	194. 0	59. 55	0
Peer2 节点	248. 4	59. 23	0
Peer3 节点	196. 3	59. 26	0
Peer4 节点	198.2	59.31	0
Orderer 节点	64. 1	0. 01	0

6.4 安全性分析与方案对比

在本文的系统中，患者的病历记录在使用 AES 算法加密后存储至医疗机构之间维护的 IPFS 私有化集群之中的，没有加密的 AES 密钥和在 IPFS 系统中的地址是无法获取加密后的病历信息。而加密密钥和密文数据地址组成的索引信息则是使用用户的证书中的公钥加密存储至区块链系统中的，没有用户私钥也无法解密索引密文。并且根据区块链数据自身的防篡改特性，极大的提高了加密索引的安全性。具体而言，由于区块链是一个分布式的存储系统，它本身包含由不同

机构之间维护的节点。一旦数据被写入区块链，那么每个节点都会备份该数据，所以链上的数据难篡改。这保证了链上数据的真实性和避免了中心化存储带来的单点故障问题。

当需要病历共享的时候，患者用户通过调用代理重加密技术生成代理重加密密钥并调用相应的智能合约执行代理重加密。医生用户通过请求相关智能合约函数拿到代理重加密过后的结果，并通过这些信息根据自己的私钥解密密文数据得到明文。在这个过程中，通过代理重加密实现了在数据在密文状态下的解密密钥转换，达到了医疗病历数据的共享的效果。代理重加密技术在随机预言模型下是基于迪菲赫尔曼假设（Diffie–Hellman Problem）的^[53]，攻击者是无法通过区块链中公开的信息攻击密文数据，因此可以保护用户的数据安全和隐私。

代理重加密测试示例程序运行结果如图 6.15 所示，在测试程序选取一段测试的文本数据作为展示。第一步：生成 Alice 和 Bob 的公私钥对，并展示要加密的明文数据 Origin Message。第二步：为在程序中使用 Alice 公钥加密明文数据 EncryptMessage 并展示。第三步：Alice 通过自身私钥和 Bob 公钥生成代理重加密密钥 ProxyReEncryptionKey，并利用重加密密钥将密文在此加密过后得到 DeCapsule 椭圆曲线点位信息进行展示。第四步：程序使用 Bob 的私钥和 DeCapsule 信息进行了解密并得到了和原来一样的数据明文 PlantText。

```
origin message: Proxy Re-Encryption Test Message
Use Alice pk encrypt Message
old key: d111dc7d7c27149722e254e19b3724818efbf901125f6bf9af483b5572d649d
Encrypted Message : YipTC9S1Mp/rmlh3DFT6w68WsJGTfeAZ0R64WuLrt76gammnKSo0N0oMfNC61ecwG
Generate ReEncryptKey
proxy-ReEncryptKey: 79659951277307291367779274706444527546119946618277347265181163831068776751134
Execute ReEncrypt Process
DeCapsule: &{0xc0000058e0 0xc000005940 20926851720206986076203105223929513326216664598111575354043065342803529755048}
use Bob sk decrypt
plainText: Proxy Re-Encryption Test Message
```

图 6.15 代理重加密示例结果图

如表 6.8 所示，本文根据区块链类型、隐私保护、访问控制以及数据存储等四个方面将本文方案和现有其他方案进行对比的结果如下。从表中可以看出第一个方案将数据存储在云服务器中并通过加密技术提供隐私保护和访问控制，但是该方案并未采用区块链技术。因此，该方案在数据放篡改方面有所欠缺。第二个方案采用了公有链，在提交区块链交易的时候需要额外消耗资源。第三个方案采用了联盟链，但是该方案将数据存储到了私有链上，系统存储性能并不高。本文

方案采用了联盟链，同时采用加密技术提供数据隐私保护和访问控制，最后通过 IPFS 进行数据存储，提升了系统的存储能力。因此在以上四个方面比其他方案具有一定优势。

表 6.8 存储方案对比表

	区块链类型	隐私保护	访问控制	数据存储
Niu ^[54]	无	是	是	云
Wang ^[55]	公有链	是	是	云
Zhai ^[56]	联盟链	是	是	链上
本方案	联盟链	是	是	IPFS

6.5 本章小结

本章主要叙述了对基于区块链的医疗电子病历共享系统的测试过程及结果。首先明确测试的环境，之后对系统进行功能测试和性能测试。在功能测试中，设计了相应的测试用例来对系统业务进行测试来检验系统的功能逻辑实现是否正确，能否满足正常的功能使用。在性能测试过程中，主要借助 Caliper 测试工具，固定发送次数，通过改变不同发送频率下对区块链系统进行查询和更新两种类型请求，获取相应的测试的结果。然后将结果绘制成为折线图进行分析并选取了在性能测试过程中节点使用的资源情况进行了解释说明。最后，对系统进行安全性分析，解释了系统是如何通过区块链技术和代理重加密技术来保护医疗电子病历安全和隐私。

第七章 总结与展望

本章主要对前面所叙述的内容进行总结，并对本系统的还存在的问题和不足做出展望。

7.1 总结

伴随着我国经济水平的不断提升，我国公民对于医疗质量的要求也不断提升，电子病历的出现虽然方便了患者的诊治过程。但是由于我国的医疗资源并不丰富并且分布也不均衡的现状，再加上不同医疗机构之间系统的异构性造成了数据孤岛现状的出现，不仅严重影响了患者的就医体验，而且也间接造成了医疗资源的浪费。于此同时，由于医疗数据具有的高价值特点，近些年来屡屡出现医疗数据泄露的事件。因此，本文针对医疗领域中存在的电子病历共享和隐私保护的两难困境，结合开源框架 Hyperledger Fabric 联盟链项目、IPFS 星际文件存储系统以及代理重加密技术设计了一个基于区块链的电子病历共享系统，解决了电子病历在不同医疗机构之间互联互通和共享流转以及隐私保护问题。

本文为了实现基于区块链的电子医疗病历共享系统，达成医疗数据的安全共享的目标，主要完成了以下工作：

1. 分析了我国目前医疗电子病历使用的现状，研究了目前电子病历共享和病历数据隐私保护中存在的问题和难点，并叙述了国内外关于医疗领域的电子病历数据共享的研究现状。
2. 针对这些问题系统研究了区块链技术和 Hyperledger Fabric 框架的各个组成部分和模块的功能，选用 Hyperledger Fabric 技术为系统提供了一个坚实的联盟链网络。同时为了减轻区块链系统的通信和数据存储压力结合了星际文件存储系统（IPFS）作为区块链系统的有效补充，形成了链上索引，链下存储的存储模式，提升了病历数据存储和查询能力。最后，采用代理重加密技术实现了病历索引在密文状态下的解密密钥转换，保证了数据安全隐私不被泄露。

3. 概述系统的总体需求，并针对系统的用户角色，通过用例图发掘出不同用户类型的功能点，提炼出系统的功能模块设计，并提出相应的系统的整体架构设计和数据模型的设计。然后在完成系统功能模块设计基础上，对各个模块进行了详细的解释说明，主要利用流程图和类图展示了核心功能的详细设计内容，并通过时序图展示了实现细节。
4. 对整个系统进行功能测试与性能测试，并展示了功能测试的系统运行结果。接着对性能测试结果和系统的安全性做了分析，保障该系统可以正常的使用，达成实现电子病历安全可信共享的系统目标。

7.2 展望

本文设计的基于区块链的数据共享交换系统初步取得了良好的运行效果，但是限于时间与能力的不足，仍然有很多可以改进和提升的地方。

1. 系统虽然解决了医疗电子病历数据共享和隐私保护的问题，但是对于这些数据中存在的价值挖掘不够，后续应该考虑结合隐私计算等技术在不侵犯数据隐私的前提下充分利用这些医疗数据的价值。
2. 本系统还应该提升对区块链网络信息可视化的能力，提升系统对整个联盟网络账本信息的监控和展示，即对联盟链内部的区块结构数据做一个具体的分析和展示。
3. 目前的系统只实现了一对一的电子病历数据共享，对于更为灵活的一对多的情况没有好的支持。因此可以考虑将代理重加密技术和属性加密技术结合来实现更为灵活的电子病历数据共享方式。

参考文献

- [1] 陈霜. 我国医疗卫生资源配置现状与政策建议 [J]. 中国总会计师, 2018(10): 118-119.
- [2] 胡燕平, 李乐乐, 卢清君, 周晓峰, 方方. 基于远程医疗的异地转诊就医一体化实践探索 [J]. 中国医院管理, 2017, 37(08): 76-77.
- [3] 胡芬. 大数据在医疗行业的应用 [J]. 大众标准化, 2020(07): 61-62.
- [4] 姚贱苟, 何英. 医疗资源浪费中的政府责任探析 [J]. 桂林师范高等专科学校学报, 2019, 33(03): 80-84.
- [5] 杨秀梅. 国外数据泄露事件与警示 [J]. 保密工作, 2019(01): 60-61.
- [6] 尚靖伟, 姜茸, 胡潇涵, 施明月. 医疗大数据及隐私泄露 [J]. 计算机与现代化, 2019(07): 111-115.
- [7] 陈纯. 联盟区块链关键技术与区块链的监管挑战 [J]. 中国工业和信息化, 2020, No. 29(11): 56-60.
- [8] 袁勇, 王飞跃. 区块链技术发展现状与展望 [J]. 自动化学报, 2016, 42(04): 481-494.
- [9] 贺海武, 延安, 陈泽华. 基于区块链的智能合约技术与应用综述 [J]. 计算机研究与发展, 2018, 55(11): 2452-2466.
- [10] 高奇琦. 主权区块链与全球区块链研究 [J]. 世界经济与政治, 2020, No. 482(10): 52-73+159-160.
- [11] Dubovitskaya Alevtina, Xu Zhigang, Ryu Samuel, Schumacher Michael, Wang Fusheng. Secure and Trustable Electronic Medical Records Sharing using Blockchain. [J]. AMIA . . Annual Symposium proceedings. AMIA Symposium, 2017.
- [12] Fu Junsong, Wang Na, Cai Yuanyuan. Privacy-Preserving in Healthcare Blockchain Systems Based on Lightweight Message Sharing. [J]. Sensors (Basel, Switzerland), 2020, 20(7).

- [13]Riaz Ahmad Ziar, Syed Irfanullah, Wajid Ullah Khan, Abdus Salam. Privacy Preservation for On-Chain Data in the Permission less Blockchain using Symmetric Key Encryption and Smart Contract[J]. Mehran University Research Journal of Engineering and Technology, 2021, 40(2).
- [14]Xiaodong Y, Ting L I, Rui L I U, et al. Blockchain-based secure and searchable EHR sharing scheme[C]//2019 4th International Conference on Mechanical, Control and Computer Engineering (ICMCCE). IEEE, 2019: 822–8223.
- [15]Zhuang Yan, Sheets Lincoln R, Chen Yin-Wu, Shae Zon-Yin, Tsai Jeffrey J P, Shyu Chi-Ren. A Patient-Centric Health Information Exchange Framework Using Blockchain Technology. [J]. IEEE journal of biomedical and health informatics, 2020, 24(8).
- [16]Kristen N. Griggs, Olya Ossipova, Christopher P. Kohlios, Alessandro N. Baccarini, Emily A. Howson, Thaier Hayajneh. Healthcare Blockchain System Using Smart Contracts for Secure Automated Remote Patient Monitoring[J]. Journal of Medical Systems, 2018, 42(7).
- [17]Bowe S , Chiesa A , Green M , et al. ZEXE: Enabling Decentralized Private Computation[C]// 2020 IEEE Symposium on Security and Privacy (SP). IEEE, 2020.
- [18]Hari A , Kodialam M , Lakshman T V . ACCEL: Accelerating the Bitcoin Blockchain for High-throughput, Low-latency Applications[C]// IEEE Conference on Computer Communications. 0.
- [19]Wang Q , Li R . A Weak Consensus Algorithm and Its Application to High-Performance Blockchain[C]// INFOCOM 2021. 2021.
- [20]Ghosh B C , Bhartia T , Addya S K , et al. Leveraging Public-Private Blockchain Interoperability for Closed Consortium Interfacing. 2021.
- [21]A Sonnino*†, M Al-Bassam*†, S Bano*†, et al. Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers[J]. 2018.

- [22]Kerber T , Kiayias A , Kohlweiss M , et al. Ouroboros Crypsinous: Privacy-Preserving Proof-of-Stake[C]// 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019.
- [23]吴银燕. 国务院办公厅关于促进和规范健康医疗大数据应用发展的指导意见 [J]. 华东科技, 2016(08) : 16–17.
- [24]徐健, 陈志德, 龚平, 王可可. 基于区块链网络的医疗记录安全储存访问方案 [J]. 计算机应用, 2019, 39(05) : 1500–1506.
- [25]罗文俊, 闻胜莲, 程雨. 基于区块链的电子医疗病历共享方案 [J]. 计算机应用, 2020, 40(01) : 157–161.
- [26]薛腾飞, 傅群超, 王枫, 王新宴. 基于区块链的医疗数据共享模型研究 [J]. 自动化学报, 2017, 43(09) : 1555–1562.
- [27]周正强, 陈玉玲, 李涛, 任晓军, 卿欣艺. 基于联盟链的医疗数据安全共享方案 [J]. 应用科学学报, 2021, 39(01) : 123–134.
- [28]王辉, 周明明. 基于区块链的医疗信息安全存储模型 [J]. 计算机科学, 2019, 46(12) : 174–179.
- [29]张利华, 蓝凡, 姜攀攀, 蒋腾飞. 基于双区块链的医疗记录安全存储与共享方案 [J]. 计算机工程与科学, 2019, 41(09) : 1581–1587.
- [30]Bitcoin: a peer-to-peer electronic cash system[Online]. Nakamoto S. <https://bitcoin.org/bitcoin.pdf> . 2009
- [31]王元地, 李粒, 胡谍. 区块链研究综述 [J]. 中国矿业大学学报(社会科学版), 2018, 20(03) : 74–86.
- [32]赵磊. 区块链类型化的法理解读与规制思路 [J]. 法商研究, 2020, 37(04) : 46–58.
- [33]Liwei Tian, Yu Sun. Research Summary of Blockchain Fragmentation Propagation Mechanism Based on Merkle Tree[J]. Journal of Physics: Conference Series, 2021, 1914(1).
- [34]IBM; IBM Launches Industry's Most Secure Enterprise-Ready Blockchain Services for Hyperledger Fabric v 1.0 on IBM Cloud[J]. Technology & Business Journal, 2017.

- [35] 邵奇峰, 张召, 朱燕超, 周傲英. 企业级区块链技术综述 [J]. 软件学报, 2019, 30(09) : 2571–2592.
- [36] I Laktineh. Construction of a technological semi-digital hadronic calorimeter using GRPC[J]. Journal of Physics: Conference Series, 2011, 293(1).
- [37] Da-Kuo Li, Chong-Xiao Shi, Guang-Hong Yang. Gossip-based distributed hierarchical algorithm for multi-cluster constrained optimisation[J]. IET Control Theory & Applications, 2019, 13(15).
- [38] Anderson JC, Lehnardt J, Slater N. CouchDB: The Definitive Guide. Sebastopol: O’ Reilly Media, Inc., 2010.
- [39] Narkhede N, Shapira G, Palino T. Kafka: The Definitive Guide. Sebastopol: O’ Reilly Media, Inc., 2017.
- [40] 王江, 章明星, 武永卫, 陈康, 郑纬民. 类 Paxos 共识算法研究进展 [J]. 计算机研究与发展, 2019, 56(04) : 692–707.
- [41] Foschini L , Gavagna A , Martuscelli G , et al. Hyperledger Fabric Blockchain: Chaincode Performance Analysis[C]// ICC 2020 – 2020 IEEE International Conference on Communications (ICC). IEEE, 2020.
- [42] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: a tutorial[J]. ACM Computing Surveys (CSUR), 1990, 22(4).
- [43] Budhiraja N , Marzullo K , Schneider F B , et al. The Primary-Backup Approach[J]. distributed computing systems, 2002.
- [44] 张国潮, 唐华云, 陈建海, 沈睿, 何钦铭, 黄步添. 基于区块链的数字音乐版权管理系统 [J]. 计算机应用, 2021, 41(04) : 945–955.
- [45] Franklin John, Suji Gopinath, Elizabeth Sherly. A decentralised framework for efficient storage and processing of big data using HDFS and IPFS[J]. International Journal of Humanitarian Technology, 2021, 1(2).
- [46] Alessandro Cilardo, Nicola Mazzocca. Exploiting Vulnerabilities in Cryptographic Hash Functions Based on Reconfigurable Hardware. [J]. IEEE Trans. Information Forensics and Security, 2013, 8(5).
- [47] 姚键. 国产商用密码算法研究及性能分析 [J]. 计算机应用与软

- 件, 2019, 36(06) : 327–333.
- [48] 国家密码管理局. SM2 椭圆曲线公钥密码算法 [EB/OL]. 2010[2018-9-23]
http://www.oscca.gov.cn/sca/xxgk/2010-12/17/content_1002386.shtml
- [49] 国家密码管理局. SM3 密码杂凑算法 [EB/OL]. 2010[2018-9-23].
http://www.oscca.gov.cn/sca/xxgk/2010-12/17/content_1002389.shtml
- [50] Blaze M, Bleumer G, Strauss M. Divertible protocols and atomic proxy cryptography[C]//International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin: Heidelberg, 1998: 127–144.
- [51] PanJun Sun. Security and privacy protection in cloud computing: Discussions and challenges[J]. Journal of Network and Computer Applications, 2020, 160.
- [52] Sukhwani H, Wang N, Trivedi K S, et al. Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network) [C]//2018 IEEE 17th International Symposium on Network Computing and Applications (NCA). IEEE, 2018: 1–8.
- [53] Dan B. The Decision Diffie–Hellman problem[J]. Third International Symposium on Algorithmic Number Theory, 1998.
- [54] Niu Shufen, Liu Wenke, Han Song, Fang Lizhi. A data-sharing scheme that supports multi-keyword search for electronic medical records.[J]. PloS one, 2021, 16(1):
- [55] Hao Wang and Yujiao Song. Secure Cloud-Based EHR System Using Attribute-Based Cryptosystem and Blockchain[J]. Journal of Medical Systems, 2018, 42(8) : 1-9.
- [56] 翟社平, 汪一景, 陈思吉. 区块链技术在电子病历共享的应用研究[J]. 西安电子科技大学学报, 2020, 47(05) : 103–112.

作者简历

教育经历：

李亚辉，男，2014年9月至2018年6月本科毕业于宁夏大学新能源材料与器件专业。

致谢

时间转瞬即逝，最后的研究生生涯也即将结束。在这人生非常重要的三年之中，虽然我遇到了许多的挫折和困难，但我也遇到了许多的良师益友，同时也学习到了专业的技能知识，为以后踏入社会打下了坚实的基础。这里我要对所有帮助过我的人说一声感谢。首先着重要感谢的是我的导师黄启春老师，在生活和科研中都给予了我关怀和帮助。同时还要感谢陈奇老师在实习过程中，给与我的帮助。当我遇到困难的时候是老师和同学鼓励让我坚持下去，这里再次向两位老师致以最高的敬意和感谢。于此同时我也要感谢在实习过程中，项目组中所有帮助和支持过我的人，没有他们，我也不能良好完成整个的实习工作。当然也不能忘了背后默默支持我的家人们，如果不是他们支持，我也不会有现在的境遇。感谢他们的陪伴与关怀，在我遇到困难和挫折的时候，及时给与我的安慰和鼓励，让我能够勇敢的面对以后所遇到的挑战和险阻，迈向人生新的未来。

最后，我也要感谢我的母校，感谢浙江大学提供了我研究生的生涯里良好的学习与生活环境，培养了我百折不挠和持之以恒的精神。我将以最好的面貌，开启人生新的篇章。