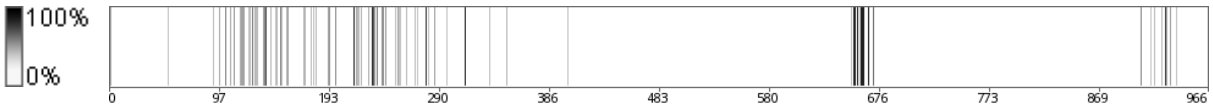
PaperPass旗舰版检测报告
简明打印版

比对结果(相似度):

总　体： 9% (总体相似度是指本地库、互联网的综合对比结果)
本地库： 7% (本地库相似度是指论文与学术期刊、学位论文、会议论文、图书数据库的对比结果)
期刊库： 5% (期刊库相似度是指论文与学术期刊库的比对结果)
学位库： 3% (学位库相似度是指论文与学位论文库的比对结果)
会议库： 0% (会议库相似度是指论文与会议论文库的比对结果)
图书库： 1% (图书库相似度是指论文与图书库的比对结果)
互联网： 2% (互联网相似度是指论文与互联网资源的比对结果)

报告编号：5CC5EDC55924BZMD7
检测版本：旗舰版
论文题目：基于患者E-Health诊治数据的安全存储与管理
论文作者：黎炜烨
论文字数：26026字符(不计空格)
段落个数：566
句子个数：966 句
提交时间：2019-4-29 2:15:33
比对范围：学术期刊、学位论文、会议论文、书籍数据 、互联网资源
查询真伪：http://www.paperpass.com/check

句子相似度分布图:



本地库相似资源列表(学术期刊、学位论文、会议论文、书籍数据):

1.相似度： 1% 篇名：《区块链技术:架构及进展》
来源：学术期刊 《计算机学报》 2018年5期

互联网相似资源列表：

1.相似度： 1% 标题：《Swift-关于Swift编程语言 - 码出境界...》
https://www.cnblogs.com/cchHers/p/8912029.html
2.相似度： 1% 标题：《关于Swift - 简书》
https://www.jianshu.com/p/92cbac8c24ef
3.相似度： 1% 标题：《关于 Swift - Swift 编程语言》
https://www.cnswift.org/about-swift/
4.相似度： 1% 标题：《2. Solidity - 简书》
https://www.jianshu.com/p/065c3086db8b

全文简明报告:

基于患者E-Health诊治数据的安全存储与管理

摘要

{47%：着眼于传统电子医疗信息系统中存在的医疗数据的安全存储和授权共享的问题，本文提出了一个利用区块链特性设计的电子病历安全存储方案，} 还提出了一个利用智能合约实现的授权共享算法，该方案和算法有效地解决了安全存储和医疗数据“信息孤岛”的问题。此外，编码设计了一个基于 iOS平台和以太坊平台的医疗助手 App，该系统前端通过iOS平台实现了问诊和就诊的流程， 后台通过以太坊平台来存储患者的电子病历信息和管理电子病历的访问控制权限。 经过编码实践，该系统能有效地实现本文提出的安全存储解决方案和授权共享算法。

关键词： 安全存储； 授权共享； 以太坊； 电子病历；

Secure storage and management based on patient E-Health diagnosis and treatment data

Abstract

Focusing on the problem of secure storage and authorization sharing of medical data in traditional electronic medical information systems, this paper proposes an electronic medical record secure storage scheme based on blockchain characteristics, and proposes an authorization sharing algorithm using smart contracts. The program and algorithm effectively solve the problem of ”information islands” for secure storage and medical data. In addition, the code design has a medical assistant App based on the iOS platform and the Ethereum platform. The front end of the system realizes the process of consultation and treatment through the iOS platform, and the backstage stores the patient’ s electronic medical record information and manages the electronic medical record through the Ethereum platform . Access control permissions. After coding practice, the system can effectively implement the secure storage solution and authorization sharing algorithm proposed in this paper.

Keywords: Secure storage; authorized sharing; Ethereum; electronic medical record;

目录

1.绪论

1.1研究背景、目的与意义

医疗健康问题与每一个人都息息相关，患者的医疗记录（包括病历、化验单、 CT图等）都是患者个人健康状况的有效证明， 对于患者了解自身健康状态和调养十分重要[1]。 然而我国的医疗发展起步晚，医疗设施落后，医疗信息化和数字化进程缓慢，对于病人的医疗隐私数据也缺乏安全有效的管理。 医疗信息数字化是未来的发展趋势，利用现代先进的计算机技术，可以减少传统医疗系统中重复的人力操作，使得诊治过程规范化。 其中医疗数据（处方、病历记录、身份信息、文档信息）的安全存储和管理更是重中之重，如何安全有效地解决医疗数据的存储和管理是本文的重点。

此外，当前时代是大数据时代，数据的作用被极大的释放，医疗数据更是数据分析的宝贵来源。 在传统的医疗信息系统中，由于信息缺乏安全存储，信息容易被不法分子利用，进行贩卖。 同时，由于各种医疗机构之间的不信任，医疗数据的共享更是困难，逐渐的形成了”数据孤岛”。 数据的不流通导致了协同诊治平台的推进困难，患者往返于各种医院和

医疗机构，无法综合各个机构的诊治结果形成完善的就诊报告。

综上所述，设计一套可以独立于第三方医疗机构，对患者个人的医疗数据进行安全存储，同时可以由患者授权给第三方读取，{41%：方便不同的医院和医疗机构进行协同诊治的系统十分必要。}

1.2国内外研究现状

21世纪是信息化时代，个人数据被大量收集和利用，医疗数据也不例外。对医疗数据的挖掘和分析，有利于医疗机构进行对于疾病的研究和防控，但同时也会带来隐私问题。隐私保护是针对公开发布的数据采取的一系列措施，防止第三方通过数据挖掘等不法手段获取病人的敏感信息。{48%：在数据发布领域，学者提出了多种隐私保护模型，主要包括：}t-closeness、I-多样性、k-匿名模型等。这些模型的原理都是通过隐藏公开的隐私数据和具体个人之间的关系，但是保证发布数据的信息公开可用[2]。

1.3论文内容与组织

本文提出了一种基于区块链实现的电子病历存储方案，还提出了一个基于以太坊智能合约实现的授权共享算法。{56%：经过编码实现，验证了上述方案和算法的有效性和安全性。}

{61%：本文的主要内容共分为六章。}

第一章，绪论，介绍了医疗数据在当今社会的重要性，分析了医疗隐私数据存储和共享的难点，对比了国内外关于隐私数据存储算法的发展和趋势。

第二章，介绍本文提出的方案和算法中涉及到的核心技术相关的理论基础，主要包括：{48%：区块链技术、比特币技术、以太坊平台等。}

第三章，提出基于以太坊平台的电子病历存储和授权共享方案，详细讨论了系统的整体框架和所用技术，经过编程验证得出实验结果。

第四章，详细描述了系统的验证实现，对系统中相关核心源码进行解释，最后展示最终的系统界面和功能实现。

{54%：第五章，主要讨论了系统设计开发过程中遇到的困难和解决办法。}

第六章，总结和展望。

2.本论文相关理论

2.1区块链技术

2.1.1定义

{46%：本文提出的方案和设计的系统以近年来火热的区块链技术为核心。}{51%：区块链技术的由来源自2008年，中本聪在《Bitcoin：}A peer-to-peer Electronic Cash System》一文中首次提出了“区块链”的概念。{51%：但是该文着重描写比特币系统，对于区块链技术并没有给出具体的定义。}{51%：在中本聪的论文中，区块和链被描述为一种用于记录比特币交易过程中交易历史的数据结构。}维基百科对于区块链的定义是：区块链（英语：blockchain或block chain）是一串连接起来的加密过的交易记录，每一个记录被称为一个区块。{41%：每一个区块包含了前一个区块的hash值、对应时间的时间

戳以及交易记录，上述特点使得区块内容具不可篡改性。} {44%：本文认为区块链既是一种数据结构，又可以认为是分布式数据库，从广义上看它也代指一系列分布式记账技术，包括智能合约、共识机制等。}

2.1.2区块链体系架构

{63%：区块链平台自底向上可以分为数据层、网络层、共识层、合约层、应用层五个层次[3]。} 数据层采用默克尔树、交易结构、签名结构等数据结构与数据库对区块、交易进行存储；{46%：网络层采用P2P协议完成各个节点间的数据传输；} {43%：共识层通过对应的算法和激励机制，解决了拜占庭容错和分布式一致性问题；} {43%：合约层通过编写智能合约，让开发者再合约上可以进行交易；} 应用层提供公开的API接口，允许用户创建和运行智能合约。

/

图1 区块链体系架构

2.1.3区块链的应用场景

根据区块链的设计技术，区块链系统具有分布式高冗余存储、不可篡改、去中心化、自动执行的智能合约等特点。 {51%：区块链不仅可以应用在金融领域，在医疗、社会系统中也有着巨大的应用潜力。} {59%：区块链目前的应用场景主要有以下几方面：} {77%：数字货币、数据存储、数据鉴证、金融交易、资产管理和投票选举六大场景[7]。}

2.1.4工作原理

传统的金融交易系统，都需要可信赖的第三方金融机构作为担保，实践证明这种机制一直运转良好，但是这类系统天生存在着以信任作为基石的的弱点。 传统的交易系统并不能实现完全不可逆的交易，因为现实中存在的第三方机构不能保证不会发生分歧。 {45%：另一方面，第三方金融机构的存在，一定程度上增加了交易的复杂性。} 所以，人们需要一种新型的电子交易系统，基于密码学技术而以信用为基础。 买家和卖家达成共识后可以直接进行交易，交易过程不需要第三方金融机构的参与，并且产生的交易是不可逆的。

/

图2 基于数字签名的电子货币

{42%：区块链或者说比特币的出现解决了上述难题，中本聪的论文中提出了一种完全通过peer-to-peer技术实现的电子现金交易系统。} {45%：一般的电子交易系统无法解决”双重支付”的难题。} 所谓”双重支付”指的是同一枚电子货币被支付两次，为了解决”双重支付”的问题我们又要引入第三方机构来作为公证， 这无疑不符合去中心化的特点。{61%：因此中本聪提出了一种解决方案，该系统通过随机散列对全部交易记录加上时间戳，}{41%：将他们加入到一个不断延伸的基于 hash值的工作量证明的链条作为交易记录，}除非重新完成所有的工作量证明，否则这条链条被认为是不可更改的。 只要互联网中大多数的计算能力没有合起来对系统发起攻击，那么可以认为这条链条是安全的[5]。

2.1.5共识机制

{46%：共识机制常见于区块链领域中，即多个节点如何达成共识的机制。} {52%：由于区块链去中心化的特点，区块链中的各个节点是分布式的。} 要维护系统的运作顺序和公平

性，必须设计一套算法，来统一调度和提供激励机制和惩罚机制。 这样的制度，必须依赖某种方式来证明，由哪个节点获得记账权，并且可以获取记录这一个区块的奖励； 或者惩罚攻击诚实节点的攻击者。 这就是共识机制。 常见的共识机制有以下几种：

1.工作量证明（Proof-of-Work，PoW），典型案例： 比特币网络

2.权益证明（Proof-of-Stake，PoS，又译持有量证明），典型案例： 以太坊

3.股份授权证明（Delegated-Proof-of-Stake，DPoS），典型案例： EOS

4.容量证明（Proof-of-space，PoSpace，又称 Proof-of-Capacity，PoC）

本文主要讨论工作量证明机制：

{46%：工作量证明（Proof-of-work）是比特币和以太坊等货币中最常用的算法，其核心依赖于密码学。} {46%：在分析工作量证明算法前先介绍散列函数（哈希函数）的概念。}散列函数是一种函数映射关系，可以将任何长度的数据映射到同一大小的数据。 以MD5算法举例如下：

MD5(“Hello World”) = B10A8DB164E0754105B7A99BE72E3FE5

MD5(“Hello World! ”) = RNG76287532E96365E841E92BFC50EDG

{43%：由上述结果可见，原文细小的改变都会导致随机散列值巨大的变化。}

工作量证明算法中，区块链通过共识算法来推选一位Leader来记录下一个区块的内容。{42%：Leader还要负责把该区块广播到外界网络，方便其他节点检验该区块的有效性。} 对该问题简单叙述如下： {44%：给定数据X，找到一个数n，使得X+n得到的哈希值是一个小于Y的数。} 举例如下：

Y = 10，X = “Hello World! ”

hash(X) = hash(“Hello World”) = 0x0f = 15 > 10

hash(x+1) = hash(“Hello World1”) = 0xff = 255 > 10

hash(x+2) = hash(“Hello World2”) = 0x09 = 9 < 10

因此，找到n为2满足要求。 由于MD5函数式加密安全的，所以我们通过暴力求解来尝试所有的组合。 我们把最快计算出上述结果的成员称为矿工，每一个区块被开发的时候，矿工会获得一部分奖励。

另一方面，工作量证明解决了确定最长的链条作为可信任的区块链的问题，因为最长的链条被认为是安全的。 {50%：如果有攻击者想要对现在的区块进行篡改，那么攻击者需要完成当前区块之前的所有工作量，并最终超越现有的区块的工作量。} {54%：经过证明，攻击者想要追赶上现有的区块，其成功概率呈指数化递减。} 但是，这不意味着区块链系统是绝对安全的。 根据摩尔定律，硬件的运算速度在高速增加，而且诚实节点参与的网络状况也会有所变化。 为了解决这种不安全的因素，工作量证明将难度与区块的生成速度的平均值相关联，如果区块生成的速度过快，那么难度将会提高。

2.1.6区块链的类型

{58%：区块链技术根据实际应用场景和需求具有公有链、联盟链和私有链三种应用模式[4]。}

他们的特点如下：

1.公有链。 公有链允许任何人自由进出，共识机制采用 pow/ pos/ Dpos，需要所有人参与记账，需要激励机制， 完全去中心化，每秒可以承载最多100笔交易，典型应用场景是加密货币，代表项目为： 比特币、以太坊和EOS。

2.联盟链。 联盟链只允许联盟成员参与，共识机制采用分布式一致性算法，记账人由联盟成员协商确定， 激励机制可选，弱中心化，每秒可承载最多10万笔交易，典型应用场景包括供应链金融、银行、物流、电商等， 代表项目有 R3、 Hyperledger。

3.私有链。 私有链只允许链的所有者，共识机制为 solo/ pbft等，记账人为链的所有者， 不需要激励机制，强中心化，承载能力视配置而定，典型场景包括大型组织、机构。

/

图3区块链的三种类型

2.1.7 区块链的优点

{89%：区块链是一种多方共同维护的分布式数据库[9]，与传统数据库系统相比，其主要优势如下:}

去中心化。 {47%：传统的数据库系统包括关系型数据库、NoSQL等非关系型都需要由某个机构单独进行管理。} {44%：区块链去中心化的设计是由多节点管理，每个节点都存储了一份完整的数据。}

不可篡改。 {43%：区块链利用Merkle树这种数据结构实现了区块链上数据的不可篡改的特性。}

可追溯。 {45%：区块链系统中的每一个节点都存储了自系统运行以来所有的交易数据，由于数据的不可篡改性，可以方便地还原出所有的交易信息。}

高可信。 区块链的参与者不需要互相信用，每一笔交易需要发送者通过数字签名验证身份，并且需要大多数的节点达成共识才可以写到链上， 交易一旦写入，不允许被篡改，除非攻击者完成之前的所有工作量。

高可用。 传统数据库通过配置主数据库和从数据库并且保证主备一致来达到高可用，但是这种方式代价高昂。 {48%：区块链中每一个节点都是主数据库，某一个节点故障不会影响整个网络的运行。}

2.2智能合约

智能合约(smart contract)是一种特殊协议，在区块链内制定合约时使用，当中内含了函数(Function)， {85%：也能与其他合约进行互动、做决策、存储资料以及传送以太币等功能。} {72%：智能合约主要提供验证及执行合约内所订立的条件。}

{49%：智能合约的概念最早于1994年由从事智能合约和数字货币研究的尼克萨博(Nick

Szabo)博士提出，是与互联网同一时期的产物。} {41%：当时因为智能合约没有可信的执行环境，智能合约只是一种理论，并没有被技术落地。} /

{63%：图4 区块链上的智能合约模型结构}

2.3以太坊

本文设计的电子医疗系统利用了以太坊平台，在已有的平台基础上开发设计了一套智能合约。 以太坊（英语： {56%：Ethereum）最初由 Vitalik Buterin 在2013年提出，是一个开源的具有智能合约功能的公共区块链平台。} {57%：以太坊通过其专用加密货币以太币(英语：} Ether)提供去中心化的虚拟机来处理点对点合约。 {54%：以太坊创建的目的是创建一个可自我执行、抗审查和可以自我维护的去中心化的世界级计算机。} 它延伸了比特币的区块链概念： 在分布式的计算机上存储、交易和验证数据。 以太坊（Ethereum）在比特币的概念上作出了更加巨大的创新，使在被互联网连接的多个计算机上运行代码成为现实。 可以简单的理解： {41%：比特币的基础区块链上面存储的是交易记录，而以太坊的基础区块链上存储的是一段可运行的程序代码。} 以太坊的目的是让用户可以通过支付以太币来使用大量的节点进行计算和存储的功能。

2.3.1 以太坊运作机制

以太坊和区块链一样需要大量的节点在各自的计算机上运行，来保证系统的运作。{42%：以太坊虚拟机（ EVM）需要被安装在每一个节点上，由每一个节点运行，} EVM在某种程度上 EVM相当于操作系统，它可以解释 Solidity语言，将 Solidity语言转换成字节码， {50%：让 Solidity语言编写的智能合约代码可以运行在各个节点上。} {41%：和区块链相同，在以太坊上执行智能合约需要支付以太币。} {41%：以太坊上执行的智能合约记录了在区块上执行的交易的信息，以太坊区块链上的交易如下：}

/

图5以太坊区块链上的交易

Timestamp指的是每一个区块的时间戳， Sender代表这个区块的发起者， Recipient代表交易的收货方， Amount代表交易的数额， Data用于创建记录和执行智能合约。

{43%：以太坊区块链可以包含以下三种交易方式：}

人与人之间的以太币转账。 类似于比特币交易，“Data”为空。

无接收方的以太币转账。 进行无接收方的交易时，数据项里面存储的是新建的智能合约代码。

用户与智能合约之间的以太币转账。 Recipient设置为智能合约的地址，执行的指令存储在Data项中。

2.3.2以太坊账户

{43%：以太坊的账户主要包括计数器、以太币余额和智能合约代码。}

两个账户之间的交易信息和信息的状态转换构成了以太坊系统的”状态”。 {40%：以太币(Ether)是以太坊里面的加密用度，主要用来支出交易用度。} 以太坊有两种类型账户：合约账户CA（Contracts Accounts）和外部账户EOA（Externally Owned Accounts）。

合约账户是智能合约代码用的账户，外部账户是人用的账户； 所以合约账户可以存储并执行智能合约代码，它的”智能性”被外部账户激活。 合约账户不存储和存储私钥，合约账户还可以调用其他合约。

外部账户是由使用以太坊的人直接使用的账户，可以存储以太币，可以发送交易到合约账户，触发既定的代码逻辑。 外部账户由公钥标识，由对应的私钥控制。

{84%：当合约账户被调用时，存储在其中的智能合约可以在矿工处的虚拟机中自动执行，}并消耗 Gas，如果 Gas不足则会触发” Out of Gas”异常，被终止执行。{41%：无论是合约账户还是外部账户，在以太坊内部都被看做状态对象。} 其中合约账户存储了余额和代码，外部账户存储了以太币的余额。

3.基于区块链的电子病历安全存储和授权共享方案

3.1研究目标和研究内容

本系统是对原慢病移动监护协同诊治平台的子系统进行升级。 {52%：“慢病移动监护协同诊治平台”(简称：} “慢病家居监护平台”), 该开发平台通过 APP端的手机移动模式，提供直观、方便、快捷的家居移动医疗监护数据的实时采集、上传云服务器， 原有平台的功能组成图如下所示：

/

图6平台功能组成图

本系统主要对原医疗协同平台系统中的电子病历管理和权限管理两个模块进行改造升级。

/

图7原平台模块调用关系图

原有平台的用户数据、医疗信息数据都存储在医院的关系型数据库中，由医院统一进行管理。 现代医疗的一大难题就是数据的获取，高昂的数据成本和患者对于隐私的谨慎使得获得医疗数据十分困难。 {41%：在传统的医疗信息系统中，患者的医疗信息都由医院统一管理，患者的访问权限十分有限。}

3.1.1 基于区块链的电子病历存储系统

针对上述问题，本文提出了一种基于区块链的电子病历存储方案，该方案采用以太坊为跨医疗机构的医疗数据创建去中心化的病历管理系统。 该系统可以为用户提供一个全新的、去中心化的病历管理方案，使用区块链来保存管理电子病历。 所有存储在这个系统的记录有以下特点：

1.患者可以自行管理自己的病历不需要依赖于第三方机构。

2.患者可以将自己病历的权限赋予医院，医院的医生获得权限后可以查看和编辑患者的病历。

3.利用独特的区块链属性，以及内含的加密模块，能够在处理敏感信息时为用户提供强大的保密技术。

3.1.2 基于智能合约的授权共享算法

在传统的医疗系统中，存在医疗记录授权流程繁琐、记录分享效率低下和身份验证困难等问题[10]。 区块链技术与密码学技术相结合，可以简化传统的身份验证过程，同时保证较高的安全性。 本文提出了一种利用以太坊智能合约实现的身份验证算法，通过 Solidity 编程语言本身的特性可以很好的实现鉴权的过程， 同时利用以太坊区块链不可篡改的特性保证了用户医疗信息隐私。

在以太坊中，智能合约本质上是一段计算机程序，它可以自动执行，通常用Solidity编程语言来编码实现，运行在EVM虚拟机上[8]。 智能合约也可以被看做是一组函数，每一个函数对应着一系列的字节码，它可以像函数一样接受参数，并对接收到的参数进行回应。

在Solidity语言中，modifier是很重要的一个关键字。 {92%：通过modifier 可以更便捷的校验函数的入参。} 如果能够合理的利用modifier去检验发送者的地址，那么就可以实现权限控制的功能。 授权的功能可以通过增加一个 checker对象，在合约初始化时，判断发送者的地址是否是合约所属者或者是添加的 checker对象， 如果是则可以通过合约拿到病历数据，反之返回错误。 算法的Solidity伪代码如下：

```
modifier check() {
require(
msg.sender == owner msg.sender == checker,
”Only ath-person can do this ops.”
);
_;
}
```

3.2拟解决的关键科学问题

设计一套模拟患者就诊和医生问诊的在线医疗系统，该系统应该具有以下功能：

患者/医生的登录/注册功能

患者预约挂号

患者查看自己的过往病历

医生问诊（包括： 查询患者过往病历、撰写诊断结果）

此外，还需要搭建本地私有链环境。 使用以太坊作为区块链平台，需要解决以太坊的配置，还需要编写一套智能合约，用来接收患者和医生的请求和实现数据的存储与读取。

{42%：4.基于以太坊的电子病历安全存储和授权共享的验证实现}

该部分针对上一章节提出的基于区块链的存储方案和授权共享算法进行编码实现。 系统前端利用iOS平台开发医疗助手App，模拟了患者就诊和预约挂号的流程和医生问诊、查看患

者病历和撰写病历的流程。 系统后台基于NodeJS、MongoDB、Ganache搭建了一套利用以太坊区块链存储的后台框架，实现了数据库和区块链联合存储并对外暴露访问API的功能。 下文将从系统的框架设计、核心代码实现和界面演示三方面进行详细的讲解。

4.1系统框架设计

系统框架图如下：

/

图 9 系统整体框架图

系统流程图如下：

/

图8系统流程图

客户端采用了iOS平台，设计了一款医疗助手App。 患者和医生都可以通过 App注册和登录到系统，其中患者登录后可以查看自己的历史病历信息还可以预约挂号， 患者预约某个医院相当于将自己的病历的访问控制权限授权给该医院的医生。 {45%：该医院的医生拿到患者的授权后，可以查看到患者的历史病历信息。} 医生诊治后，可以在App上给患者新建病历，录入诊治内容，新建的病历会写入到以太坊的区块链中。

4.2 核心代码实现

4.2.1 智能合约Solidity源码

```
pragma solidity ^0.5.0;
contract Ehealth {
//
address payable public patient;
string name;
string record;
string gender;
address myaddress;
address checker;
uint age;
constructor(string memory _name, string memory _gender, uint _age) public {
name = _name;
```

```
gender = _gender;
age = _age;
myaddress = msg.sender;
checker = msg.sender;
}
modifier check() {
require(
msg.sender == myaddress msg.sender == checker,
”Only ath-person can do this ops.”
);
_;
}
modifier check_doc() {
require(
msg.sender == checker,
”Only doc can do this ops.”
);
_;
}
event makeChange(address _a);
function authens(address _checker) public{
checker = _checker;
}
function unAuth() public check{
checker = myaddress;
}
function getName() public view returns(string memory){
```

```
return name;
}
function getRec() public view check returns(string memory, string memory, uint, string memory){
return (name, gender, age, record);
}
function setRec(string memory _rec) public check{
record = _rec;
emit makeChange(msg.sender); // who changed it.
}
}
```

{41%：智能合约是运行在以太坊上的一段代码，用户与智能合约提供的函数交互从而达到读取电子病历和存储电子病历的功能。} address payable public patient; 存储了患者的账户，代表这份智能合约属于哪一位患者。 合约中存储了患者的信息，包括姓名、年龄等。 其中最为重要的是record和check两个变量，其中record存储的是病历的内容。前端App将病历的信息转换为json格式的字符串，存储在record中，每一份病历通过一定的标志进行分离。 checker存储的是被授权的机构的地址，可以看做医院的凭证，其中checker对应的医院的医生有读取和写入record的权限。 其中权限控制的代码在 modifier check()函数中， msg. sender== myaddress msg. sender== checker， 这行代码保证了只有当发起人为患者或者是被授权的医院时， 才可以访问智能合约中的数据，否则拒绝访问。

iOS客户端核心源码

```
import Foundation
import Alamofire
import SwiftyJSON
class NetworkService {
static let shared = NetworkService()
// 用户注册
func register(user: User) {
DispatchQueue.global().async { [weak self] in
AF.request(”http: //localhost: 5000/user/register”，
```

```
    method: . post,

    parameters: self? .encodeUser(user: user)).responseJSON { response
in

    if let data = response.data, let utf8Text = String(data:
data, encoding: . utf8) {

    print(" Data: \(utf8Text)") // original server data as UTF8
string

    }

    }

    }

    }

    // 用户登录

    func login(userName: String, password: String) {

    DispatchQueue.global().async { [weak self] in

    AF.request(" http: //localhost: 5000/user/login",

    method: . post,

    parameters: self? .encodeLoginMsg(userName: userName, password:
password)).responseJSON { response in

    if let data = response.data {

    do {

    let res = try JSONSerialization.jsonObject(with: data, options:
[]) as! [String: Any]

    if let error = res[" errors"] {

    UserService.shared.isLogin = false

    print(error)

    } else {

    UserService.shared.isLogin = true

    // 保存用户的信息

    let user = User(name: res[" name"] as! String,
```

```
    phone:  ””,
    userName:  ””,
    password:  ””,
    userType:  res[”userType”]  as!  Int,
    isCreated:  ””)
    UserService.shared.setLoginUser(user:  user)
    }
    }  catch  let  error  as  NSError  {
    print(error)
    }
    }
    }
    }
    }
    //  获取医院信息
    func  getHospitalMsg(completionHandler:  @escaping  (_  resArray:
[[String:  Any]])  -]  Void)  {
    DispatchQueue.global().async  {
    AF.request(”http:  //localhost:  5000/home/hospitals”,
    method:  .  get).responseJSON  {  response  in
    if  let  data  =  response.data  {
    do  {
    if  let  hospitalDictionaryArray  =  try
JSONSerialization.jsonObject(with:  data,  options:  [])  as?  [[String:
Any]]  {
    //  把数据传输给VC
    completionHandler(hospitalDictionaryArray)
    }
```

```
    }   catch   let   error   as   NSError   {

    print(error)

    }

    }

    }

    }

    }

    //  预约挂号

    func   authorization(hospitalName:   String,   completionHandler:
@escaping   (_   status:   Int)   -]   Void)   {

    DispatchQueue.global().async   {

    let   url   =   URL(string:   ”http:   //localhost:
5000/home/hospitals”)?
.appendingPathComponent(hospitalName).appendingPathComponent(”reservation”)

    AF.request(url!   .absoluteString,

    method:   .   post).responseJSON   {   response   in

    if   let   data   =   response.data,   let   utf8Text   =   String(data:
data,   encoding:   .   utf8)   {

    do   {

    print(utf8Text)

    if   utf8Text   ==   ”1”   {

    completionHandler(1)

    }   else   {

    completionHandler(0)

    }

    }

    }

    }
```

```
    }

    }

    //  获取病历

    func  getRecords(name:  String,  completionHandler:  @escaping  (_
res:  [[String:  Any]])  -]  Void)  {

    DispatchQueue.global().async  {

    let  url  =  URL(string:  ”http:  //localhost:
5000/home/checkinfo”)?  .appendingPathComponent(name)

    AF.request(url!  .absoluteString,

    method:  .  get).responseJSON  {  response  in

    if  let  data  =  response.data  {

    do  {

    if  let  res  =  try  JSONSerialization.jsonObject(with:  data,
options:  [])  as?  [[String:  Any]]  {

    //  把数据传输给VC

    completionHandler(res)

    }

    }  catch  let  error  as  NSError  {

    print(error)

    }

    }

    }

    }

    }

    //  新增病历

    func  createNewRecords(name:  String,  record:  Record,
completionHandler:  @escaping  (_  res:  [String:  Any])  -]  Void)  {

    DispatchQueue.global().async  {
```

```
let url = URL(string: ”http: //localhost: 5000/home/edit”)?
.appendingPathComponent(name)

AF.request(url! .absoluteString,

method: . post,

parameters: self.encodeRecord(patientRecord: record)).responseJSON {
response in

if let data = response.data {

do {

let res = try JSONSerialization.jsonObject(with: data, options:
[]) as! [String: Any]

// 把数据传输给VC

completionHandler(res)

} catch let error as NSError {

print(error)

}

}

}

}

}

}
```

上述是iOS客户端中一个核心的NetworkService类，主要用于网络通信，负责与系统后台交互。 通过注释可以看出，主要对用户注册、用户登录、获取医院信息、预约挂号、获取病历、新增病历等后台接口进行了二次封装。 上述代码使用了 Alamofire框架，简化了iOS中网络请求和结果回调的难度，在每一个接口内部都通过异步队列进行网络请求， 防止阻塞主线程，保证 App的性能。

4.3运行环境

/

图9 系统运行环境

本系统的运行环境如下：

操作系统： macOS Mojave 版本10.14.4

处理器： 2.2 GHz Intel Core i7

内存： 16GB 2400 MHz DDR4

数据库： MongoDB

以太坊客户端： Ganache

4.4开发技术

系统前端基于iOS平台，主要开发语言为Swift，主要使用了SnapKit、SwityJson、Alamofire等框架。 系统后台基于 Node. js框架，通过Ganache搭建本地私有以太坊区块链，利用 Solidity语言开发本系统的智能合约， 通过Web3. js开发库发布和运行智能合约，患者的非敏感信息存储在 MongoDB数据库。{41%：后台暴露API给iOS客户端调用，客户端和后台通过HTTP协议进行通信，数据格式为Json。}

4.4.1Solidity

{100%：Solidity是一种面向对象的高级语言，用于实现智能合约。} {100%： 智能合约是管理以太坊内账户行为的程序。} Solidity受C++，Python和JavaScript的影响，旨在针对以太坊虚拟机（EVM）。 {100%：Solidity是静态类型的，支持继承，库和复杂的用户定义类型以及其他功能。} {87%：通过Solidity，可以创建智能合约，例如投票，众筹，盲目拍卖和多重签名钱包。}

4.4.2Swift

{40%：Swift是2014年苹果公司在WWDC上发布的一中支持多编程范式、闭包、函数式编程的现代编程语言。} {92%：Swift是一门集现代语言之大成，集结了苹果的工程师文化精髓以及开源社区多样化于一身的编程语言。} {100%：编译器为专为性能所调优，语言专为开发所优化，二者绝不互相妥协。} {95%：Swift代码为大部分现代硬件编译和优化，语法和基本库都基于指导原则设计。} Swift最大的优点就是在做到安全的同时又保证了速度，摒弃了 Objective- C动态性的缺点， 增加了许多静态的类型判断，保证了代码安全，是一款适合用于工业开发的编程语言。

4.4.3Ganache

{98%：Ganache是以太坊开发的个人区块链，可用于部署合同，开发应用程序和运行测试。}它既可用作桌面应用程序，也可用作命令行工具（以前称为TestRPC）。 Ganache适用于Windows，Mac和Linux。

4.4.4Web3.js

{65%：web3.js是一个库集合，允许开发者使用HTTP，WebSocket或IPC连接与本地或远程以太坊节点进行交互。}

4.5 数据库表结构

系统后台涉及到两张数据库表，分别是患者信息表和病历表，表结构如下：

表1患者信息表（patientInformation）

列名（字段）

解释

数据类型

id

患者ID

String

userame

患者登陆名

String

name

患者名字

String

gender

患者性别

String

表2病历表（medicalRecord）

列名（字段）

解释

数据类型

record_id

病历ID

String

patient_id

患者ID

String

patient_name

患者名字

String

patient_gender

患者性别

String

patient_record

病历内容

String

4.6系统演示

本系统后台通过Node.js搭建，其中以太坊采用本地Ganache客户端，数据库采用Mongodb，利用Web3.js与智能合约进行交互。

本系统前端采用iOS平台开发，命名为医疗助手App。 主要的系统界面包括： 登录/注册界面、患者个人主页、医生个人主页、患者查看个人病历、患者预约挂号、医生问诊主页、医生撰写病历页面等。

4.6.1 启动Node.js服务器

/

图10 启动Node.js服务器

通过npm start命令启动Node.js服务器。 Node.js会启动一个监听本地5000端口的Web服务器。

4.6.2 启动Mongodb数据库

1）通过本地配置在终端前台启动Mongodb

mongod --config /usr/local/etc/mongod.conf

2）将Mongodb数据库设置为守护进程

brew services start mongodb-community@4.0

本文采用第二种方式，启动后输入mongo出现以下界面表示Mongodb成功启动：

/

图12 启动Mongodb数据库

4.6.3 启动Ganache客户端

Ganache是一个用来快速启动个人区块链的本地客户端，方便程序员用来本地测试智能合约的代码。 打开Ganache客户端：

/

图13 Ganache客户端

选择已经搭建好的spiffy-ornament环境，启动后如下：

/

图14 以太坊账户信息

Ganache通过图形化的界面为我们显示现在已有的以太坊账户、已经创建的区块链、已经存在的交易信息和创建好的智能合约等信息。

/

图15 以太坊默认账户

Ganache默认为我们创建好若干个账户，每个账户拥有1000个虚拟以太币。

/

图16 Ganache初始区块

Ganache已经事先创建好若干个初始区块，开发者编写的智能合约的数据都会存储在这条区块链上。

/

图17 Ganache查看历史交易信息

Ganache中也可以查看已经产生的历史交易信息，可以查看到每次交易的hash值、发送者的地址、接受者的地址以及每一次交易消耗的gas总额。 后台配置到此全部开启成功，下面演示医疗助手App。

4.6.3登录/注册

登录/注册界面可以选择以患者身份登录还是以医生身份登录：

/

图18登录/注册页面

4.6.4个人首页

/

图19患者个人首页

/

图20医生个人首页

4.6.5患者查看历史病历

患者点击查看病历后可以看到自己的历史病历信息，包括就诊的医院名称、医生名称和就诊的时间：

/

图21患者个人病历

4.6.6病历详情

查看某一次的病历详情：

/

图22 病历详情

病历详情中可以查看到这张病历存放的智能合约的地址，还有就诊的医院的名称、医生名称、病历的创建时间和医生的诊断结果。

4.6.7预约挂号

患者点击预约挂号后， App会显示目前已经注册了的医院的信息，患者可以根据自己的需要授权某家医院， 医院拿到患者的授权后，医院下注册的医生就拥有了查看患者病历信息和给患者编写病历的权限。

/

图23预约挂号页面

给汕头附一医院授权，若授权成功，App会提示授权成功，反之提示授权失败：

/

图24患者预约挂号

4.6.8医生问诊

医生点击“我要问诊”进入问诊页面：

/

图25医生问诊页面

App上方有一个搜索框，通过输入患者的名称，医生可以查看到患者的历史病历信息：

/

图26查看患者病历

医生点击某一行的病历信息可以查看到患者以前的就诊信息，包括之前就诊的医院、就诊医生的姓名、就诊时间和以前医生开的处方等信息。

4.6.9新建病历

医生对患者问诊结束后，点击新增按钮可以给患者新建一份病历，写入此次诊断证明：

/

图27新建病历页面

医生需要填写自己的姓名、所在医院及科室、诊断治疗的时间以及诊断结果和开设的处方信息。　医生写入新的病历后，病历会写入到患者拥有的以太坊智能合约内，再次刷新App就可以看到刚才写入的新的病历信息。

/

图28　查看新建病历

4.6.10写入验证

医生新增一份电子病历，病历会实时写入到以太坊的区块链中，查看以太坊的回调信息：

[

{

”to” : ”0xbb98aa07736d72ba94381f53c96ebc345f09caa9”，

”s” : ”0 x50 ca5 c645 d5 ce9 c6 a53 e284 bc3028 bc4 d73571 ab0208319 a6 c9 d7375994 f239 f” ，

”logsBloom” : ”0x0000000000000000000000000000000000000000000000080000000002000000000000000000000000000004000000000000000000000000001000000008000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000”，

”v” : ”0x1b”，

”blockHash” : ”0 x5 f6 a71548 c464293051 eaca8 d167229538 b36286 a 6 e06 b7 d39 f2 ac02 c596 fc6 a” ，

”gasUsed” : 407109，

”blockNumber” : 28，

”from” : ”0xa0ba4385d1020f6c5869fa51e2435f368ae8eefe”，

”r” : ”0xc6164677f781f14e4dff217bd139961040cdf9fcb78b2ff5e9dcb167a353e8cc”，

”transactionIndex” : 0,

”events” : {

”makeChange” : {

”transactionIndex” : 0,

”event” : ”makeChange”,

”signature” : ”0 x1 c5 c01 d25862 c15354 d25454 df5 e17517 eb4 b9328 fc0 d52846 b007332 ef38 cc8 ” ,

”blockNumber” : 28,

”transactionHash” : ”0x3cb590390544fc25f20a224ba9351b0b66fc032f603efa18f79453661fbfb4cb”,

”blockHash” : ”0 x5 f6 a71548 c464293051 eaca8 d167229538 b36286 a 6 e06 b7 d39 f2 ac02 c596 fc6 a” ,

”type” : ”mined”,

”returnValues” : {

”_a” : ”0xa0BA4385d1020f6c5869fA51E2435F368ae8EeFe”,

”0” : ”0xa0BA4385d1020f6c5869fA51E2435F368ae8EeFe”

},

”logIndex” : 0,

”raw” : {

”data” : ”0x000000000000000000000000a0ba4385d1020f6c5869fa51e2435f368ae8eefe”,

”topics” : [

”0 x1 c5 c01 d25862 c15354 d25454 df5 e17517 eb4 b9328 fc0 d52846 b007332 ef38 cc8 ”

]

},

”id” : ”log_0x2562de00048c49c4ebbfbe2fb87618a9dfee8d79cd9c4bca83730dae12526fd3”,

”address” : ”0xbB98AA07736d72ba94381F53C96EBc345f09CAa9”

}

},

”transactionHash” : ”0x3cb590390544fc25f20a224ba9351b0b66fc032f603efa18f79453661fbfb4cb”,

”status” : true,

”contractAddress” : null,

”cumulativeGasUsed” : 407109

}

]

从上述回调中可以看出，新建的电子病历写入到了地址为0 xbb98 aa07736 d72 ba94381 f53 c96 ebc345 f09 caa9的合约账户中，

写入的区块链的Hash值为：

0 x5 f6 a71548 c464293051 eaca8 d167229538 b36286 a6 e06 b7 d39 f2 ac02 c596 fc6 a

本次写入用了407109个Gas，区块链的序列号为28。 综上，实验结果符合预期。

5.问题与解决办法

5.1遇到的问题

首先，区块链技术是比较新颖的一门技术，在对其开发前我们需要先了解其基本原理、工作机制。 刚开始学习区块链时遇到的比较大的困难对共识机制的了解，共识机制指的是在分布式的环境下多个节点如何达成共识， 在比特币中主要应用了工作量证明(Proof-of- Work)的方法。 除此之外，对于时间戳服务器、双重支付的问题的理解也十分关键。

系统设计采用以太坊作为开发平台，以太坊是一个生态圈，含有许多的开发工具和开发框架。 其中最重要的是Solidity语言，Solidity语言用来开发以太坊智能合约的语言，掌握其语法与特性十分重要。 其次遇到的困难是学习Truffle框架，Truffle框架是一款以太坊的开发框架，帮助开发者优化了智能合约的编译、发布、交易等流程。 Truffle框架还包含了一系列的子框架，包括： Truffle-Contract等。 除此之外，对于 Ganache等以太坊客户端的学习也十分困难，开发者要访问以太坊区块链必须通过相关客户端提供的 API，有官方提供的 Go语言客户端还有方便本地使用的 Ganache客户端。 其次，比较困难的是对于Web3.js库的学习，Web3.js库是一个封装了访问区块链相关API的JavaScrtpt库。 简单来说，Web3.js是前端界面与区块链智能合约的桥梁，对于项目的搭建十分重要。

5.2解决办法

接触一个新的领域，或者接触一门新的技术，都需要花大量的时间去熟悉才能掌握。特别是对于Solidity语言、Truffle框架、Web3.js库，国内非常少中文资源，最好的学习资料是官方文档。 通过查阅官方文档，学习官方文档里的Demo，可以快速的入门。 除了官方

文档，Github上有非常优秀的开源项目，通过学习前人的代码，可以掌握高质量的开发技巧。

5.3开发技巧

本系统的开发主要包括两方面： 前端界面的开发和后台以太坊区块链的开发。 合理的利用Truffle框架可以帮助开发者一键搭建项目，包括智能合约、前端项目、编译模块和服务器脚本。 前端开发要注意与以太坊的接口对接，常见的可以通过 Web3. js或者 Truffle- contract库进行读取智能合约的示例， {59%：从而获取智能合约的数据和调用智能合约的函数。} 以太坊的开发则要合理利用 Ganache工具，在本地搭建私有的区块链作为测试， 可以迅速的返回结果，不需要等待出块的时间，方便开发者进行测试和开发。

6.总结与展望

6.1总结

本文提出了一个利用区块链特性设计的电子病历安全存储方案，还提出了一个利用智能合约实现的授权共享算法。 此外，编码设计了一个基于 iOS平台和以太坊平台的医疗助手App，该系统前端通过 iOS平台实现了问诊和就诊的流程， 后台通过以太坊平台来存储患者的电子病历信息和管理电子病历的访问控制权限。

6.2展望

{40%：本文介绍的基于以太坊的电子病历管理和共享系统还存在许多的不足和缺陷。} 一方面由于本文通过本地 Ganache客户端搭建的本地私有链是免费的，但是如果大规模商业化， 需要去以太坊平台注册相关密钥和信息，而以太坊需要收费，因为成本是非常大的一个考量。 {43%：其次，以太坊是基于区块链设计的去中心化的分布式数据库，本身需要消耗大量的计算资源和电力资源，} 因此从资源利用的方面考虑，以太坊方案也有它天生的不足。 最后，区块链中共识机制的达成需要相关算法的支持，而在以太坊中采用的权益证明（ Proof- of- Stake， PoS，又译持有量证明）需要多个节点达成共识，实时性不高，如果将本系统大规模的应用在医疗系统中， 系统的响应速度和 Qps也许会成为最大的瓶颈。

综上所述，以太坊有着其天生的去中心化、不可篡改和匿名性的优点，但同时也具有收费高、资源消耗大、实时性低等缺点， 综合以太坊平台的优点和缺点，本文认为可以将患者比较重要的医疗隐私信息通过以太坊平台存储， {44%：而与医疗敏感数据无关的个人信息等数据存储在相关机构的数据库即可。}

6.3致谢

牛顿曾经说过： {82%：”我之所以能取得现在的成就，是因为我站在巨人的肩膀上”。} {46%：同理，没有老师、同学、师兄和亲人的支持和鼓励，本文也难以如此快的完成。} 首先，我要感谢我的导师朱诗生老师，朱师生老师从毕业设计的选题、毕业设计的设计、毕业设计的中期进展到最终毕业论文的修改， 都给予了本人偌大的帮助和指导。{50%：再次对朱老师对我的帮助和关怀表示诚挚的谢意！} 其次，还要感谢黄仁俊师兄，黄仁俊师兄利用自己的空余时间给我们讲解区块链的相关知识， 以及目前关于区块链的发展情况，还将自己平时收集的资源无私的奉献给我们， 对我论文的研究方向给出了许多意见和指导，对黄仁俊师兄表示诚挚的感谢！ 此外，我还要感谢我的搭档梁汉帮同学，梁汉帮同学对于以太坊平台和 Web3. js等框架有深刻的见解， 在 Node. js方面也有着深厚的编程功力，在他的协助写系统才能顺利的完成。 {41%：最后，我要感谢我的父母，是他们的支持让我能一直坚持学习、坚持自己的研究方向，再次衷心感谢他们！}

参考文献：

[1] 梅颖. 安全存储医疗记录的区块链方法研究[J]. 江西师范大学学报(自然科学版), 2017, 41(5): 484-490.

[2]曹敏姿, 张琳琳, 毕雪华, 赵楷.个性化(α□ 1)-多样性 k-匿名隐私保护模型[J].计算机科学, 2018，45(11).

[3] 张亮, 刘百祥, 张如意, 江斌鑫, 刘一江.区块链技术综述[J/OL].计算机工程: 1-15[2019-04-17].http://kns.cnki.net/kcms/detail/31.1289.TP.20190316.1352.002.html.

[4] 赵延红, 原宝华, 梁军.区块链技术在医疗领域中的应用探讨[J].中国医学教育技术, 2018, 32(01): 1-7.

[5] Nakamoto S. Bitcoin: a peer-to-peer electronic cash system [Online], available: http: //bitcoin.org/bitcoin.pdf, June 12, 2016

[6] 薛腾飞, 傅群超, 王枞, 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.

[7] 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.

[8] 邱欣欣, 马兆丰, 徐明昆.以太坊智能合约安全漏洞分析及对策[J].信息安全与通信保密, 2019(02): 44-53.

[9]邵奇峰, 金澈清, 张召, 钱卫宁, 周傲英.区块链技术: 架构及进展[J].计算机学报, 2018, 41(05): 969-988.

[10]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/OL].计算机应用: 1-8[2019-04-27].http://kns.cnki.net/kcms/detail/51.1307.TP.20190121.1013.058.html.