**基于区块链的献血管理新体系的研究**

**对于血液的获取、存储及供应的管理是在血液管理中的重要方面，目前的管理体系相对老旧，使得其所存在的问题越来越突出，其主要导致血液获取、存储及供给管理的不够精细，造成血液的浪费。血液的获取方也就是血站等需要了解所采集血液的血型、品种及数量需求的类型，并且需要了解目前存储的血液的血型、品种及数量及其随时间的变化情况，血液的使用方也就是医疗机构需要持续的了解血液的血型、品种及数量的真实数据，医疗管理机构需要了解血液的真实的类型和数量的变化。这就提出了血液管理系统需要数据不可更改、并且持续在线服务，否则无法保证严格的数据真实和持续了解数据。**

**一：背景与现状**

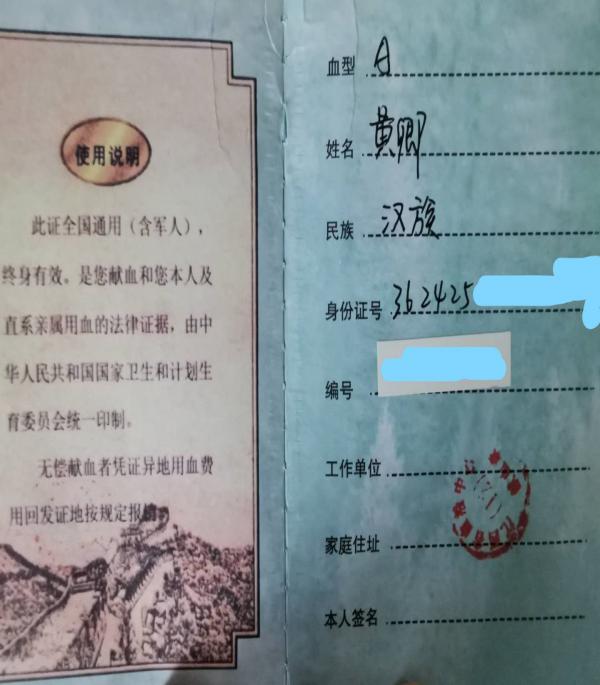
**1.1献血的发展历程**

**1958年我国第一个初具规模的现代化血站在天津的中国医学科学院输血及血液学研究所诞生（该血站是现在位于成都的中国医学科学院输血研究所的前身）。随着我国经济建设和医疗卫生事业的发展，临床用血量逐年攀升，于上世纪六十年代和七十年代，血源组织发生混乱，输血事故不断发生，在此情况下，卫生部专门组织了一个调查组，在中国医学科学院输血研究所肖星甫所长率领下到中南、西南和东北地区调查输血工作现状，发现不少地区献血管理混乱、卖血成风，有的医院不按规定采血，而是随便降低体检标准，甚至不体检，不做血液化验。有些地区血霸横行，对献血者敲诈勒索。当时，使用的输血技术和输血器具也落后于国外，操作规程执行不严，病人输血反应多，甚至因血液被污染造成死亡。1978年11月24日《国务院批转卫生部关于加强输血工作的请示报告的通知》颁发（国发[1978]242号），通知指出“实行公民义务献血制度，是改变我国输血工作落后面貌，解决医疗和战备储备用血的一项根本办法”。**

**1.2我国献血所存在的问题**

**1998年10月1日起，《中华人民共和国献血法》开始实施。我国的临床输血的执法能力；医院实行专家审核审批责任制度更加完善，输血的不合理性、不科学性现象也在减少，我国更多的是将血站的操作的规范和发展进行了分析和发展，但是对血液和献血者本身的管理就稍微落后了。**

**1.2.1 我国的献血证**

****

**图1-1**

**从图中我们可以看出，我们可以看到献血证上那句话“此证是您献血和本人其直系亲属用血的法律依据”。那么这个“法律依据”那可就有几个致命的缺陷了。**

**第一：极易丢失和损坏，我国献血证依旧是由几张纸和线组合而成，其在日常生活中很容易丢失和损坏；第二：信息及其不安全且及其没有可信任度，很多时候献血车的工作人员仅仅根据献血证上的信息来确定一个人的血型和信息。但是仅根据那手写的信息就来确定一个人的身份是及其不安全的。其它各地献血证与上图有些不同，但是献血证不安全的这一关键性缺点却并无不同。**

**同时献血证上的信息也应该得到完善，我们都知道献血证可以用来给直系亲属免费用血，但是由于人们对政策的不了解，人们在日后需要用血的时候会发生误解产生不满的情绪。**

**1.2.2 献血者的困惑**

**将自己的血液奉献给社会的公益事业的这一行为，已经得到很多人的理解和支持，但是一些民众对于这种献血这一个概念还是很陌生，他们有些人和我们有着同样的困惑：我们的血去哪里了？**



**图1-2**

**当献血者从献血车下来之后血液的去向就变得不可追溯，人们不知道它去了哪里，何时何地用在了什么地方。**

**1.2.3 如何证明这是你的血**

**我们都知道献血证有以下作用：**

**1.无偿献血的公民，本人有享受免费用血的权利。无偿献血者（不包括自身储血）累计献血量在800毫升以下的，按献血量的三倍免费用血**

**2.累计献血量在800毫升（含800毫升）以上的，终身免费用血。无偿献血者的配偶和直系亲属（指父母、子女）需要临床用血的，可以按献血者实际献血量等量免费用血。**

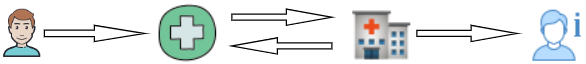
**理论上只要病人有献血证，医院就应该按照规定减免，但是有时候医院并不信任献血证，再加之如果献血证丢失，你该如何证明你献过血，你是什么时候献的血？在用献血证获得报销或者减免的过程中，难免少不了官僚主义的现象的发生，并且需要病人加开各种材料的证明，这些材料往往还可以需要在不同机构之间踢皮球，这些情况往往让一些病人放弃使用献血证。**

**二：区块链与新型献血管理的实用性分析**

**区块链的去中心化同步记账，身份认证，数据加密和不可篡改等特征，如果将其引入到献血管理体系中，这将可以确保血库中血液信息变得可追溯并且提高其信任度，在新的管理体系上这将使得各地得献血点，血液中心和医院共同建造、共同维护、共同监督，从而满足献血者和被献血者以及社会公众的知情权、监督权。增强民众对血站和医院的信任度，提高献血者日后使用献血证报销等一些列办事效率。其区块链与新的献血管理具体分析如下**

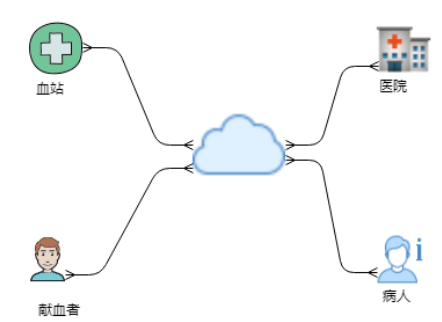
**（1）信息更加安全可靠：将区块链技术引入献血管理之后，我们的献血证将迎来历史性改变，我们将会拥有一个电子献血证。我们都知道区块链技术是一种新型分布式数据组织方法及运算方式，通过去中心化来集体维护一个可靠数据库的技术。在数据共享过程中可明确数据 的来源、所有权和使用权，达到数据在存储上不可篡改、在流通上路径可追溯、在数据管理上可审计的目的，保证数据在存储、共享、审计等环节中的安全，实现真正意义上的数据全流程管理。介时我们可以在献血证上面储存更多的信息，比如直系亲属等等，这些信息被很好的保护起来，不会有被篡改和遗失的风险。**

**（2）打破传统信息交流模式：区块链所采用的分布式数据结构模式，不需要构建多级数据管理中心，同时不需要跨部门，跨地区的数据集中，并且不会改变现有的医院，血站的业务系统。不仅如此，区块链所构建的新型部门协作模式，促进献血者、血站、血库、医院之间的可信信息流动，提高其协同的效应。与传统的数据集中共享模式比较起来，这种模式的效率和实行性更高。**



**图2-1**

**传统的信息交流流动仅限于血站和医院之间，献血者无法从医院和血站获取到自己的血液信息，病人也无法获取自己所得到的血液信息**



**图2-2**

**采用这种信息交流模式，这将极大的促进血液信息在各个角色之间的流通，使得信息可溯源，增加可信度，消除群众的疑惑。**

**（3）可信的存在证明：鲜血点通过对电子献血证的文件签名，将献血者的信息保存在区块链上面，可证明献血的某次献血行为的合法存在，杜绝了伪造献血证的现象，当你的血液信息上链之后，其他地方的血站也可以查询到你信息，介时你在外地鲜血的时候，可以证明你是否处于安全间隔期。这促进了社会互信，降低了公众信任成本提高了工作效率。**

**（4）可审计性高：电子献血证经过哈希运算备份保存在区块链上。由于加盖了时间戳使得每献血证上的信息可追溯、可证明。为不同地方的血站、医院提供了更加可信的数据支撑，避免了重复认证等不必要的措施，减少了资源的浪费。**

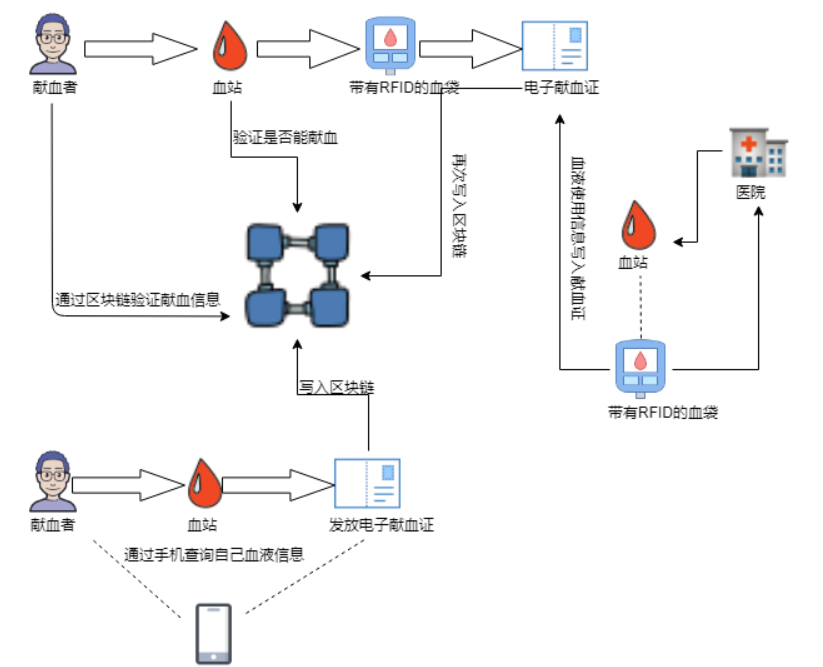
1. **信息互信互通：主链进行身份认证，侧链认证血液、血液流通的信息，两条链双向锚定，实现证件照自由流动，逐步提高服务质量，助理诚信社会的建设**

**三：基于区块链的献血管理研究方案设计**

**3.1.1 业务设计**

**基于区块链的可信电子献血证用于部门给公民发放电子献血证。区块链共识算法驱动区块链上的每个节点都参与到交易的验证过程中，保证区块链上交易都是经过确认可信的，区块链上每个节点都维护一个公共的账本，任何一个节点对自己所维护的账本的修改都将不被其他节点所承认，从而保证公共账本不可被伪造和篡改，并且通过区块链保存的不可篡改的献血、用血的记录，使得血站、医院共同建造，共同维护；献血者、血站、医院、病人来共同监督，从而满足公众的知情权、监督权。以保持人们对于献血的积极性。**

**从业务功能上来说，新型的管理模式增加了一些模式，如下图3-1所示献血者首先向献血点申请一份专属于自己的电子献血证；献血环节血站通过区块链查询到献血者的信息，并查询他是否处于安全间隔期，并将血液装入一个带有RFID的血袋，RFID血袋将记录各种信息，再将RFID上传至电子献血证并且再次上链；医院从血站那里使用了带有RFID的血袋，使用信息将记录再电子献血证里面并再次上链。**



**图3-1**

**1.申请**

**献血者首次献血的时候应该向血站提出申请，填写个人信息，以获得电子献血证**

**2.发证**

**为了实现电子献血证的发放，公民申请献血后，由血液管理中心发放。该证件可以通过客户端，电脑客户端等集中渠道进行发放，申请的个人献血证通过客户端于后台进行统一管理。具体流程图如下：**

1. **申请者通过手机客户端或者在献血点提供个人信息，申请一个数字身份。**
2. **血液管理中心审核持证人的信息，生成个人数字身份；血液管理中心用其私钥对持证人的数字身份签名后，发送给持证人，以及向电网信 息中心报审/备案。**
3. **持证人使用自己的公钥对接收到的有血液管理中心签名的数字身 份进行加密，并提交数字身份区块链。**
4. **电子献血证区块链完成验证和共识过程，电子献血证发放成功**

**3.查验**

**每当献血者献血的时候，献血点应对献血者的身份进行识别，并且查询献血者是否处于安全间隔期。献血点的工作人员可以通过手机客户端进行查验工作。以确保流程的安全具体流程如下**

1. **持证人出示电子献血证上的身份码，供工作人员扫描；**
2. **献血点工作人员利用公匙验证献血证的合法性**

**（3）工作人员查询电子献血证上身份信息和上次献血的间隔期**

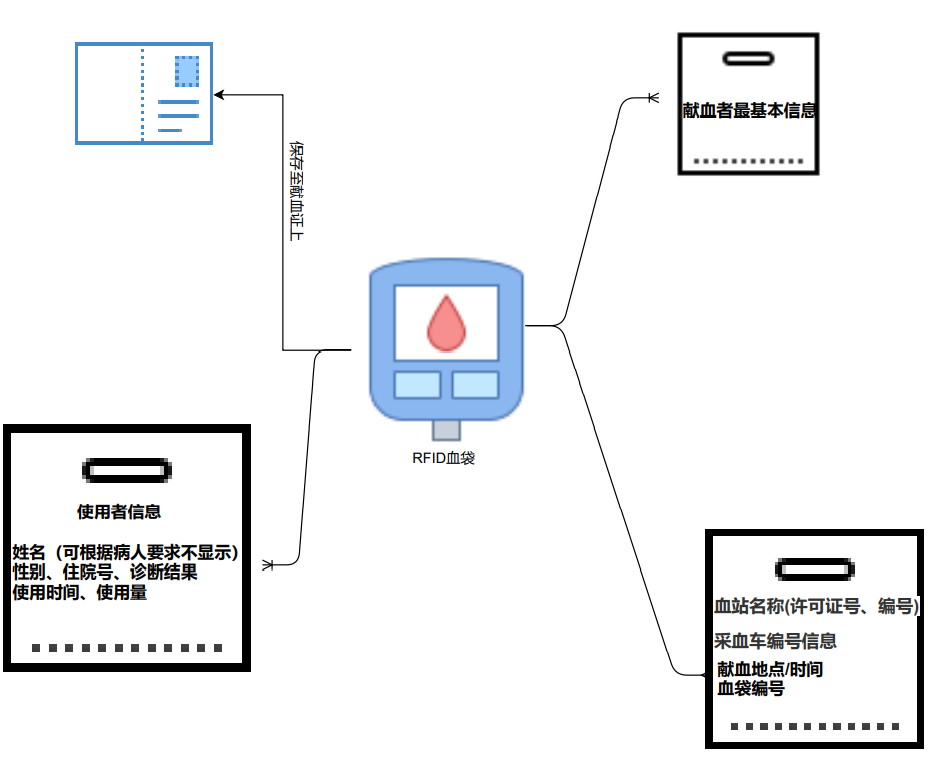
**4.献血**

**献血点在确认献血者身份，准备好配有RFID的血袋，该血袋配对应唯一血液ID，通过计算机端向电子献血证写入献血者献血信息和血液的全流程数据，上传至区块链平台**

**3.1.1.2 RFID血袋**

**无线射频识别即射频识别技术（Radio Frequency Identification，RFID），是自动识别技术的一种，通过无线射频方式进行非接触双向数据通信，利用[无线射频](https://baike.baidu.com/item/%E6%97%A0%E7%BA%BF%E5%B0%84%E9%A2%91/4142085" \t "https://baike.baidu.com/item/%E5%B0%84%E9%A2%91%E8%AF%86%E5%88%AB%E6%8A%80%E6%9C%AF/_blank)方式对记录媒体（[电子标签](https://baike.baidu.com/item/%E7%94%B5%E5%AD%90%E6%A0%87%E7%AD%BE/6976650" \t "https://baike.baidu.com/item/%E5%B0%84%E9%A2%91%E8%AF%86%E5%88%AB%E6%8A%80%E6%9C%AF/_blank)或射频卡）进行读写，从而达到识别目标和数据交换的目的。**

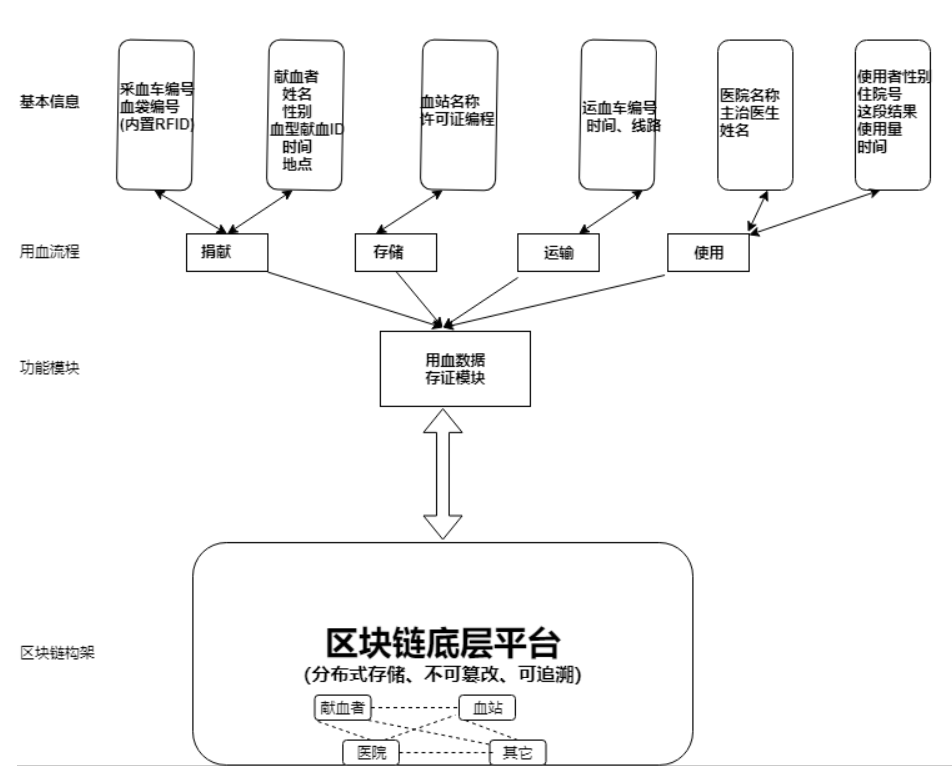
**所设的RFID血袋，对应唯一献血ID，可通过计算机读写端提前关联所属血站名称（许可证号、编号）、采血车编号信息；不仅如此，可通过计算机前端及时录入献血者基本信息（姓名、性别、血型）、献血时间及地点、运血车编号、使用机构名称（编号）、主治医生姓名。其中，最为关键的是，使用者的信息（姓名、性别、住院号、诊断结果、使用时间、使用量）这些平时我们无法得知的信息将得以查询。RFID血袋还将记录着“捐献-存储-运输-使用”的全流程信息上传至区块链平台。如下图3-2**



**图3-2**

**3.2.1 架构设计**

**我们采用以太坊技术作为支撑，结合新型管理模式的需要，我们设计了如下图3-3技术架构，主要为区块链构架、功能模块、用血流程。**

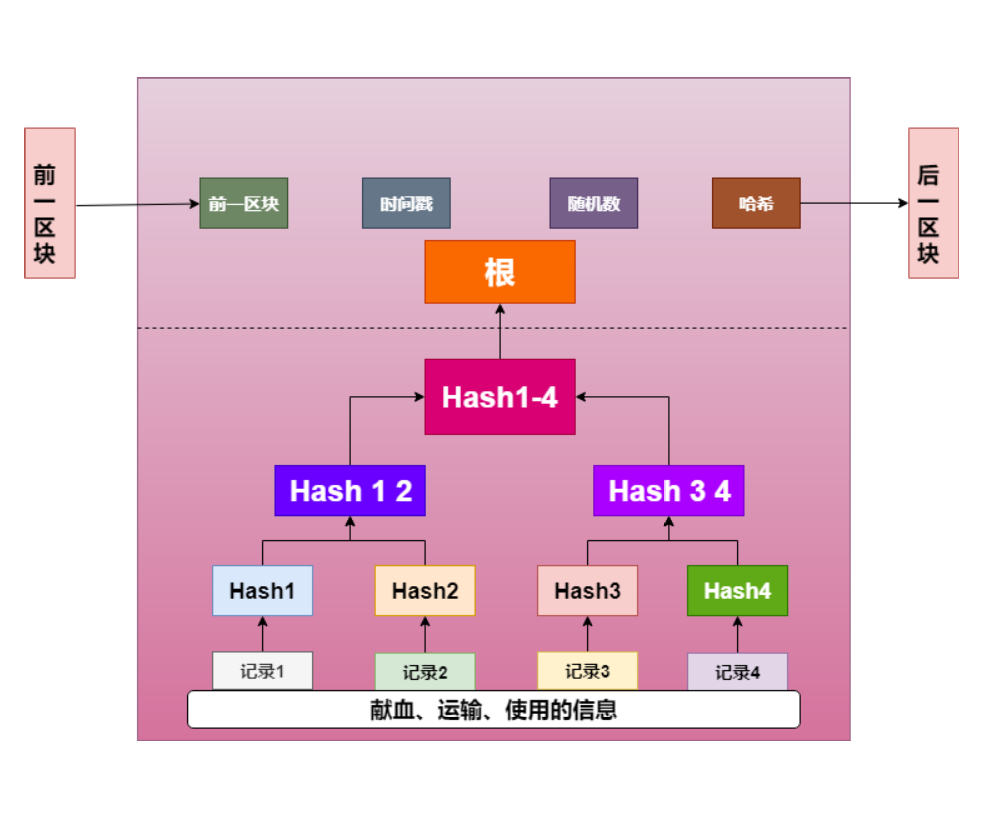


**图3-3**

**区块链构架，将献血者、血站、医院、血液使用情况进行区块化封装，连接成区块链；**

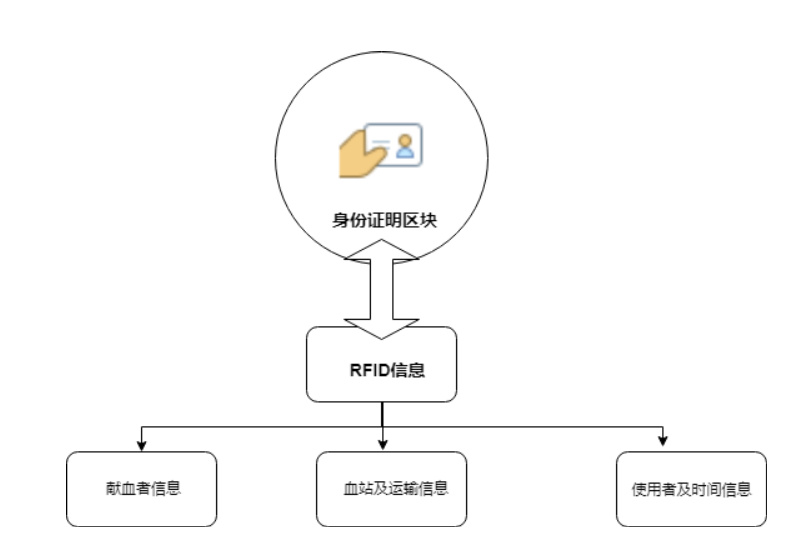
**功能模块以区块链构架为基础，提供区块链服务；**

**用户层表现形式为APP，机构前端，以实现用户与用户之间，用户与机构之间的信息交互。**



**图3-4**

**如上图3-4所示，我们对数据进行区块化封装，每个区块链有区块头和区块体两部分。其中，如图所示，迁移区块、时间戳、随机数、哈希值放在区块头里面。当前区块的交易以及通过验证的、每一个区块的创建过程中生成的所有记录封装在区块体中。区块 体由Merkle树组成，在Merkle树的叶子节点上，保存着每一笔发证、收 证、查证和换证的记录，这些数据通过两两Hash计算向上形成Merkle树 的根记入区块头。**



**图3-5**

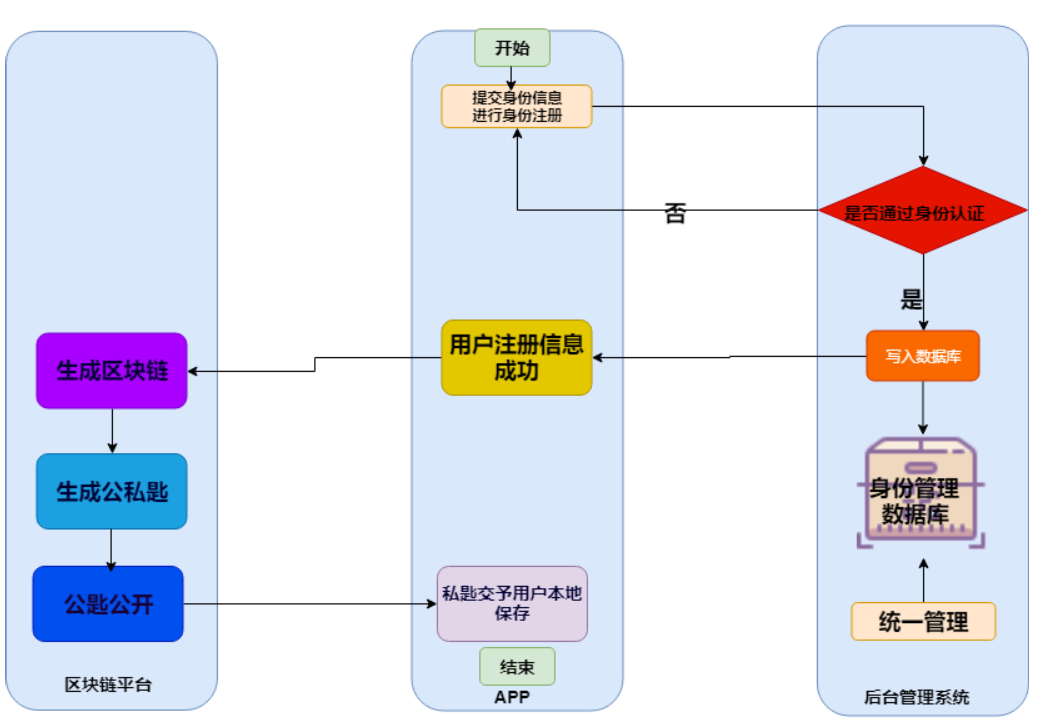
**根据新型献血的管理体系的类型，采用主链和侧链的双向锚定的设计加以区分。结果如上图3-5所示。如图显示，主链是用于身份验证，用于验证并存储献血者的信息。侧链则是血袋上的RFID里面所存储的信息。献血者完成身份认证后并献血之后，通过主链锚定其RFID信息，其每次献血都会有一个新的RFID，每个RFID都会对应一个主链。**

**侧链的双向锚定拘束允许信任在不同网络之间传递，建立起新人体系；同时，如果遇上有新的版本更新，也是可以发布试用新版本，这对主链和侧链不会造成影响。**

**3.3.1 交互设计**

**3.3.1.1身份认证阶段**

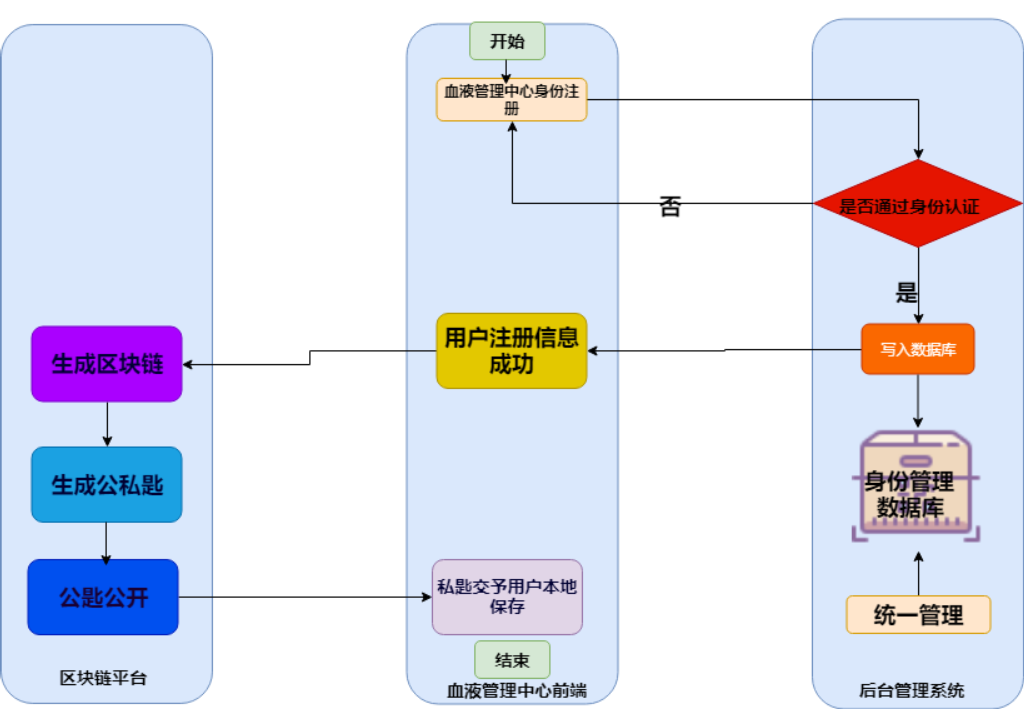
**身份认证系统主要涉及区块链平台、用户App和后台管理系统的交互。对于献血者注册系统有三个角色：献血者、血液管理中心、查验机构。如下图3-6是普通用户认证过程。**



**图3-6**

**首先，献血者登陆注册系统，提交身份信息之后，后台管理系统进行身份认证。如果认证成功，则将献血者的信息写入身份管理数据系统，并返回注册成功信息，否则要重新注册；注册完成之后将生成一个区块链并得到一个区块链的ID，然后再生成公私匙，公匙公开私匙交于献血者保存。**

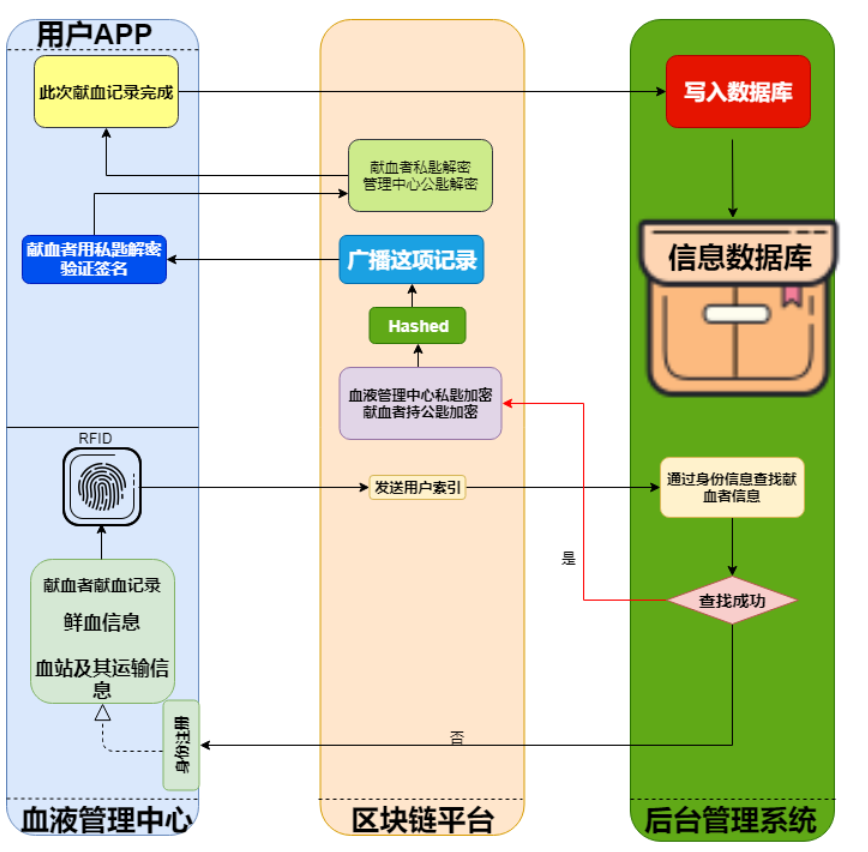
**其中，生成区块链的ID，我们将利用Hash算法将献血者信息（身份证上的基本信息）生成全网的唯一身份标识，根据这一个标识，我们可以判断该献血者是否是第一次献血。原则上同样的输入会有相同的Hash值产生，但是献血者的献血信息重复性就很低，所以可以完全避免该身份标识发生重复的现象。不仅如此，后台管理系统所保存的身份标识可以将认证机构根据用户信息生成的全网唯一身份标识与存储过的全网唯一身份标识进行对比，来判断该献血者是否是第一次献血：若未找到存储过的全网唯一身份标识则为第一次献血，否则为再次献血。后学，则由发证机构进行身份认证如图3-7**



**图3-7发证机构身份认证**

**3.3.1.2献血认证阶段**

**该阶段主要涉及用户APP、血液管理中心、区块链平台、后台管理系统之间的交互。如图3-8所示**



**图3-8**

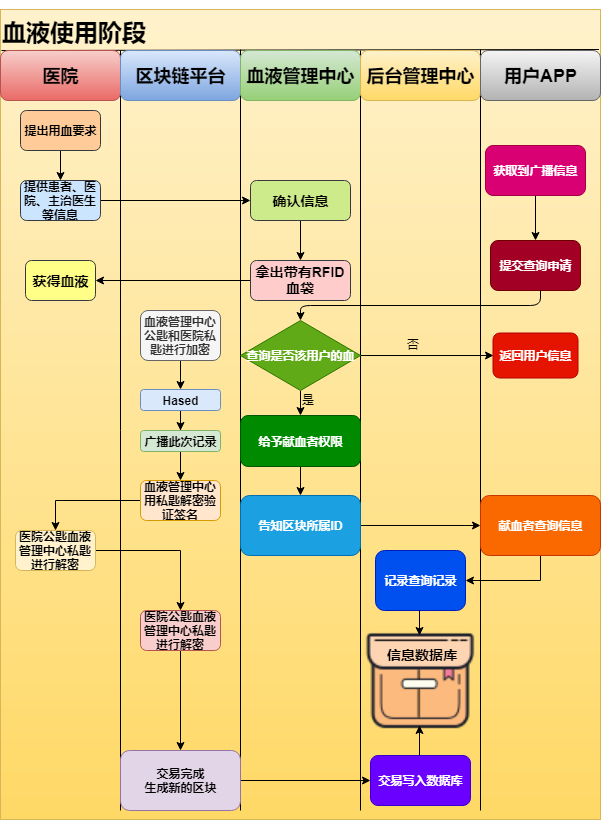
**首先用户献血之后，将其献血记录、血液信息、血站及其运输信息记录到RFID中**

**其次，血站使用其公匙和用户的私匙进行两次加密，加密之后平台输出该笔交易的Hash值，并全网广播这笔交易，记录这次献血的信息的区块链编号，写入区块链。其它机构收到了血液管理中心发出的广播，并对献血者的身份进行了确认。**

**最后，献血者在APP上验证血液管理中心的签名，确保信息的准确性，在区块链上面用私匙解密完成确认，血液管理中心公匙解密作为存证。**

**3.3.1.3血液使用阶段**

**该阶段主要涉及用户APP、血液管理中心、区块链平台、后台管理系统以及医院之间的交互。如图3-9所示**

****

**首先，医院发出用血要求，然后将患者、主治医生等信息提供给血液管理中心以供日后验证使用，血液管理中心拿出带有RFID血袋，交予医院。由于生命刻不容缓，所以验证加密工作可以在日后完成。**

**患者用血之后，将使用记录记录在RFID上，然后医院用私链血液管理中心用公链对此进行加密，加密之后区块链平台对此输出Hash值，并且广播此次交易记录，血液管理中心使用私匙解密验证是否是医院的签名，然后医院用公匙、血液管理中心用私匙进行解密，交易完成并且生成新的区块。**

**因为RFID也是献血者本次献血的血液的唯一ID，所以献血者会收到这个广播，如果他想查询信息可以向血液管理中心提出查询申请，血液管理中心则根据其申请者的信息与RFID上所对应的信息进行验证，通过验证给与申请者权限并告知其区块的所属ID以供申请者查询；反之则返回用户信息界面。**

**四：关键技术与方法**

**4.1.1 以太坊账户体系**

**4.1.1.1以太坊的钱包体系涉及的名词：**

**account：一个帐号的总概念，本质上对应着一个唯一私钥key，由这个私钥key可以算出公钥（也就是公开的address）。私钥key代表着帐号的总控制权。**

**address：帐号地址，相当于私钥key算出来的公钥。**

**wallet：钱包，管理着帐号地址以及一个密码对应着的keystore，钱包的作用就是提供方便，让用户只需要记住帐号地址和密码即可。**

**keystore：一个帐号的keystore与一个密码对应，由密码+keystore可以算出私钥key，从而拿到帐号的控制权。在钱包中对一个帐号密码进行修改之后，生成的keystore也会改变。**

**以太坊中，拥有一个私钥key就相当于控制以太坊帐号空间中的一个帐号，无论是私链、公链或者是联盟链，这个私钥key对应的帐号不变，这也是以太坊可以离线申请帐号的原因。或者也可以说以太坊体系下的所有链共享一个帐号地址空间，你控制一个以太坊帐号，那么这个帐号在公链、私链中都是你所控制**。

**4.2.1 以太坊的技术架构**

**以太坊采用五层架构实现，从上到下分别是数据层、网络层、共识层、激励层、智能合约层、如图4-1所示**

****

**图4-1**

**1.智能合约层**

**智能合约赋予账本可编程的特性，区块链2.0通过虚拟机的方式运行代码实现智能合约的功能，比如以太坊的以太坊的虚拟机（EVM）。同时，这一层通过在智能合约上添加能够与用户交互的前台界面，形成去中心化的应用（DAPP）。当然，在某些技术文档中认为DAPP应该在智能合约上单独为应用层，也是有一定道理，只要不影响理解即可。**

**2.激励层**

**激励层主要实现以太币的发型和分配机制，以太币不是数字货币，而是定位于平台运行的燃料，运行智能合约和发送交易都需要向矿工支付一定的以太币。目前以太币可以通过挖矿获得，矿工每挖到一个区块都会奖励其以太币（数量不一定）。**

**3.共识层**

**共识层主要实现全网所有节点对交易和数据达成一致，以太坊采用两种共机制，初期采用工作量证明机制（POW），待网络中的以太币充分流通和分散后，改为采用交易速度更快、无资源消耗的权益证明机制（POS），从而有效地避免额纯POS机制导致的初期权益分配不公平的情况。**

1. **网络层**

**网络层主要实现网络节点的链接和通信，又称“点对点技术”，是没有中心服务器、依靠用户群交换信息的互联网体系。与有中心服务器的中央网络系统不同，对等网络的每个用户端既是一个节点，也有服务器的功能，其具有去中心化与壮健性等特点。**

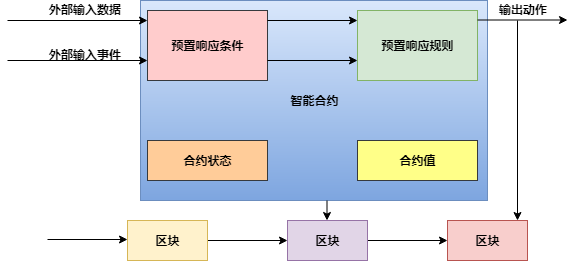
1. **数据层**

**数据层最底层，是一切的基础，主要实现了两个功能，一个是相关数据的存储，另一个是账户和交易的实现与安全。数据存储只要基于Merkle树，通过区块的方式和链式结构实现，大多数以KV（Key-Value）数据库的方式实现持久化，比如以太坊采用LevelDB。账号和交易的实现基于数字签名、哈希函数和非对称加密技术等多种密码学算法和技术、保证了交易的去中心化的情况下能够安全地进行。**

**4.3.1 以太坊的智能合约**

**4.3.1.1智能合约简介**

**典型的智能合约模型如图4-2所示**

****

**图4-2**

**从本质上来讲，智能合约也是一段程序，但是与传统的IT系统不同，智能合约继承了区块链的三个特性：数据透明、不可篡改、永久运行。**

**（1）数据透明**

**区块链上所有的数据都是公开透明的，因此智能合约的数据处理也是公开透明的，运行时任何一方都可以查看其代码和数据**

**（2）不可篡改**

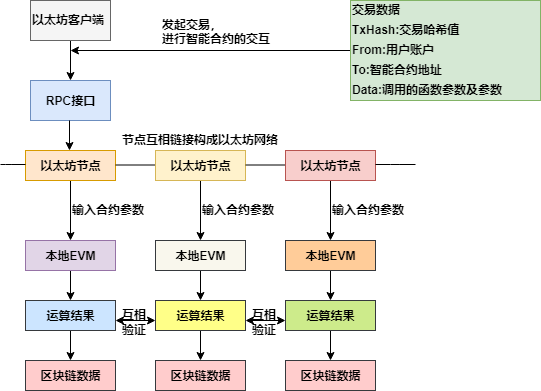
**区块链本身所有数据不可篡改，因此部署在区块链上的智能合约代码以及运行产生的数据输出也是不可篡改的，运行智能合约的节点不必担心其它节点恶意修改代码与数据。**

**（3）永久运行**

**支撑区块链网络节点往往达到数百甚至上千，部分节点的失效不会导致智能和玉的停止，其可靠性理论上接近于永久运行，这样确保了智能合约能像纸质合约一样每时每刻都有效。**

**4.3.1.2以太坊智能合约的运行原理**

**智能合约是部署在区块链的代码，区块链本身不能执行代码，代码的执行是每个节点通过虚拟机实现，智能合约的运行原理如图4-3所示。**

****

**图4-3**

1. **以太坊虚拟机**

**以太坊虚拟机是以以太坊中智能合约的运行环境。智能合约可以比作是java程序，java程序通过java虚拟机（JVM）将代码解释字节进行执行，以太坊的智能合约通过以太坊虚拟集解释成字节码进行执行。EVM被沙箱封装起来，也就是说在EVM内部的代码不能接触到网络、文件系统或者其它进程，甚至智能合约与之智能合约之间的调用也有些限制。**

**（2）RPC接口**

**RPC接口是以太坊与其它IT系统交互的接口，以太坊节点在8545端口提供了JSON RPC API接口，数据传输采用JSON格式，可以执行Web3库的各种命令，可以向前端，比如Mist等图形化客户端提供区块链的信息。**

**如图4-3可以看出，部署在区块链上的智能合约是一段能够在本地产生原智能合约代码的数据串，可以理解区块链为一个数据库，而客户端通过发起一笔交易，告诉以太坊节点需要调用的函数及相关参数，所有的以太坊节点都会接受到这笔交易，然后从区块链这个数据库中读取了存储智能合约的运行代码，在本地EVM运行出结果。为避免节点作恶，节点运行智能合约的结果将与其它以太坊节点进行对比，最后确认无误之后才将结果写入区块链中，从而实现智能合约的正确执行。**

**4.4.1 以太坊的可信存证机制**

**所有的数据及操作都被永久地记录在数据区块供用户访问。这些存证数据所在地数据区块被同步存储在系统地每一个参与运算的节点中，所有这些节点构成了可信存证系统及其坚韧的分布式数据管理系统，任何一个节点上的数据被破坏可以通过同步节点或访问其他哈希节点得到验证；同时任何一个节点上的数据区块被破坏都不会影响整个可信存证系统的正常运转。下面我们对系统的可信性进行分析。**

**4.4.1.1 存储可信性**

**存储可信性是指数据存储的正确性，即使系统在存储、通信过程中发生故障，甚至被恶意攻击时，只要数据处理结果被确认，存储的数据就不被篡改或者丢失。以太坊平台整体上可划分为网络层、共识层、数据层、智能合约层和激励层(应用层) 5 个层次。**

**在网络层，以太坊平台以完全分布且可容忍单点故障的 P2P 协议作为网络传输协议，用于节点间传输交易数据和区块数据。节点之间以扁平拓 扑结构相互连通，不存在中心化节点，具有平等、自治、分布等特性。在数据传输过程中，网络中广播的数据由节点负 责时刻监听，当接收到相邻节点发来的新交易和新区块时，节点首先会通过工作量证明或数字签名等来验证交易和区块 的有效性，验证通过后，新交易加入到正在构建的区块，新区块连接到区块链当中，确保了数据的有效性与可信性。**

**在共识层，以太坊依赖于工作量证明机制来解决区块的容错一致问题。PoW 机制依靠分布式节点的算力竞争来保证全网数据的一致性和安全性，它要求每个节点基于自身算力解决求解复杂但验证简单的 SHA256 计算难题，即寻找一个合适的随机数，使得区块头部元数据的SHA256哈希值小于区块头中难度目标的设定值。如果有人想要篡改或者伪造其中一个区块，就必须针对该区块及其后所有的区块，重新寻找区块头的随机数，构造出一条比当前被公认的区块链主 链更长的链，这个恶意过程所需要消耗的算力超过了整个区块链中正在进行正常挖矿运算的算力总和，因此恶意攻击的 难度和成本很高。PoW机制融合了经济激励政策与共识算法，促使更多以太坊节点参与到挖矿运算，主动增强了网络的可靠性与安全性。**

**在数据层，以太坊中每个区块包含区块头和区块体两部分，区块体存放交易数据，区块头存放父区块哈希、时间戳、 Merkle 根等数据，其中父区块哈希构建了区块链以区块为单位的链式结构，保证了数据数据的不可篡改性，时间戳是该区块的生成时间，基于交易哈希生成的 Merkle 根确保了区块内交易数据的完整性与可信性。因此，区块链特殊的数据结构保证了所有区块环环相扣，任一区块被篡改，都会引起其后续所有区块哈希指针的连环变化。**

**在智能合约层，以太坊实现了Casper共识。从前所述，PoW 牺牲性能来换取数据的一致性和安全性，针对PoW 机制低效耗能的缺陷，以太坊平台引入 PoS 机制，根据矿工在区块链中拥有的股权来决定其挖矿的难度，当矿工拥有越多的以太币，越容易找到合适的随机数，挖矿的整体难度就会越低。以太坊又基于 PoS 机制提出 Casper 共识，要求矿工购买以太币作为抵押，根据抵押的以太币数量和时间，按比例分配区块的记账权和经济奖励。如果攻击者想要篡改数据，需要先在全网达成共识，篡改操作将记录在区块链当中，篡改结果也会被记录在状态数据库中，因此共识机制的存在 既提高了以太坊平台的性能，又保障了数据的一致性和安全性。**

**4.4.1.2 访问可信性**

**访问可信性是指数据存证后的可审计及可追溯。本文提供了2种访问方式，一是调用智能合约方法，二是依据智能合约的编码规则对交易信息进行解码，两种方式都保证了数据被访问时的正确性。**

**在数据层，以太坊以 Key-Value 数据库存储索引数据与状态数据，使得系统可以基于交易哈希检索交易数据，或者基于区块哈希检索区块数据。这与传统数据库的预写式日志 非常类似，区块数据按照日志文件格式存储，以追加形式写 入，支持数据重放，维护了所有的历史操作记录。另外，Merkle树的结构维护一个区块内的事务之间的关系，以太坊区块数据具备了预写式日志不具备的不可篡改性和可追溯性。**

**在智能合约层，本文系统中的合约方法提供了查询操作，无需共识，也不会更改合约状态值，调用取证函数则就获取到最新存入区块的数据。此外，本文依据智能合约的编码规则还原数据的存储过程，基于区块哈希检索区块数据，又基于区块数据中的交易哈希检索交易数据，通过解码交易数据中的Input字段来还原存证数据。**

**在激励层（应用层），以太坊除了基于以太币的数字货币交易，还支持去中心化应用，文本基于Node.js语言构建Web前端应用**通**过 JSON-RPC 与运行在以太坊节点上的智能合约进行 通信，支持访问可信的区块数据、交易数据和状态数据，使 应用程序拥有良好交互性的用户界面。**

**4.4.1.3 处理可信性**

**处理可信性是指数据处理的正确性，系统需要保存数据的最终状态。在传统的数据库管理系统中，用户通过提交事务来处理数据，事务常由过程型语言与SQL语句组合共同构成，事务执行的过程或结果通过日志进行记录。区块链的逻辑结构与传统数据库管理系统中的日志有所相似，但比特币区块链仅支持挖矿与转账功能，功能上适用于数字货币，对于数据的处理具有很大的局限性。以太坊的智能合约提供了区块链中的数据处理功能，但如果直接将以太坊作为传统的数据库系统来处理数据，会有无法处理不同文件格式、无法 提供查询功能及存储成本昂贵效率低下的缺陷。**

**因此，可以借助MongoDB 数据库对存证数据进行集中式统一处理，将任意文件格式的数据分割为块，块集合存储于fs.chunks，文件元数据存储于fs.files，提取数据的ID和MD5值进行后续的分布式智能合约存储。MD5 算法具有压缩性、容易计算、抗修改性、弱抗碰撞和强抗碰撞的特点，如果在分布式存储的过程中数据被某些人做了改动，那么得到的MD5值将有很大变化。此外，原始文件经过MD5加密后为 32 位的字符串，能够给定固定大小的空间存储、传输和验证，大大地降低了存证成本，也提供了通道验证数据的完整性，确保系统的处理可信性。**

**4.5 小结**

**本小结提出了一种在以太坊平台上基于智能合约的可信存证系统。存证后数据全链条每个节点都有存证，数据分布式完整、不可篡改、安全可信，适用于有信任公关危机的机构，可以直接从运维的节点获取和验证数据，把存证数据视为直接证据，不再需要第三方机构出具证明。**

**在未来的工作中，可以考虑在智能合约中增加访问限制，只有特定的用户可以调用合约方法，但任然允许任何人访问合约，确保可信存证系统的公开公正。**

**五：搭建及部署私有以太坊**

**如3.2.1提到，新型的献血管理体系以区块链技术作为底层技术。我们都知道，区块链可以分为联盟链和私有链，下面我们将讨论并且搭建一条属于自己的以太坊私有链。通过以太坊JSON RPC和JavaScript API编程接口将其部署到所创建的以太坊私有链，并可调用合约获得正确的合约执行结果。**

**5.1 启动命令参数**

**详情请见官网https://geth.ethereum.org/docs/interface/command-line-options**

**5.1.1命令：**

**account - - - - - - - - - - - - - - - - - -管理账户**

**console- - - - - - - - - - - - - - - - - - 启动交互式javascripts环境**

**js- - - - - - - - - - - - - - - - - - -执行指定的javascripts文件（多个）**

**init- - - - - - - - - - - - - - - -- - - - -启动并初始化一个新的创世纪块**

**copydb- - - - - - - - - - - -- - - - - - - 从文件夹创建本地链**

**export- - - - - - - - - - - - - - - - - - -导出区块链到文件**

**import- - - - - - - - - - - - - - - - - - -导入一个区块链文件**

**makecache- - - - - - - - - - - - - - - - - -生成ethash验证缓存(用于测试)**

**makedag- - - - - - - - - - - - - - - - - -生成ethash 挖矿DAG(用于测试)**

**5.1.2 ETHEREUM选项：**

**--keystore- - - - - - - - - - - - - - keystore存放目录(默认在datadir内)**

**--config value- - - - - - - - - - - -TOML 配置文件**

**--datadir “xxx”- - - - - - - - - --数据库和keystore密钥的数据目录**

**--lightpeers value- - - - - - - - - - - 最大LES client peers数量(默认值:20)**

**5.1.3 API和控制台选项：**

**--rpc- - - - - - - - - - - - -- - - - -- - - 启用HTTP-RPC服务器**

**--ws- - - - - - - - - - - - - - - - - - - - -启用WS-RPC服务器**

**--jspath loadScript -- - - -- - - JavaScript加载脚本的根路径(默认值:“.”)**

**--exec value- - - - - - -执行JavaScript语句(只能结合console/attach使用)**

**5.1.4 交易池选项：**

**--txpool.accountslots value- - - - - -每个帐户保证可执行的最少交易槽数量**

**--txpool.globalqueue value- - - - - - - - 所有帐户非可执行交易最大槽数量**

**--txpool.lifetime value- - - - - - - - - -非可执行交易最大入队时间**

**--txpool.journal value - - - - - - - - - -本地交易的磁盘日志：用于节点重启**

**5.2 以太坊环境搭建**

**5.2.1安装go环境**

**1.官网网址：<https://golang.org/dl/>**

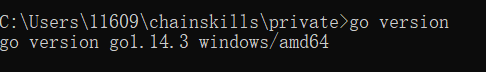
**2.安装完成之后配置环境变量**

export PATH=$PATH:/usr/local/go/bin

vim ~/.bash\_profile 如下图



1. 查看是否配置成功命令：go version

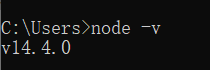


**5.2.2安装Node.js、Npm**

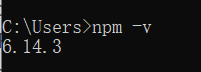
**1.官网网址：<https://nodejs.org/en/download/>**

**2.查看是否安装成功：**

**Node -version或者node -v 如下图**



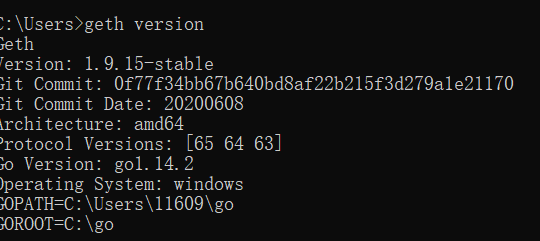
NPM是随同Nodejs一起安装包的管理工具



**5.2.3以太坊Ethereum安装**

**apt install ethereum**

**查看以太坊是否安装成功:geth version**

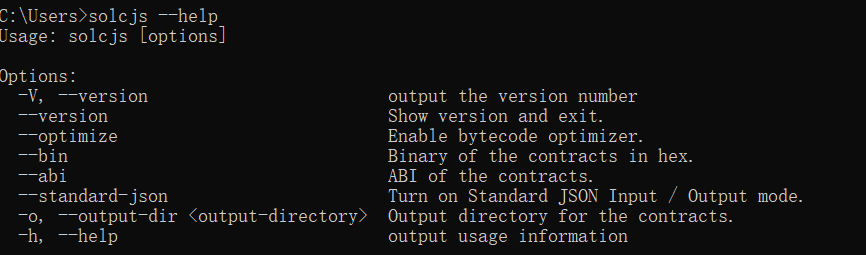


**5.2.4 solc编辑器安装**

**npm install -g solc**

**查看solc支持的功能**

**solcjs --help 如图**



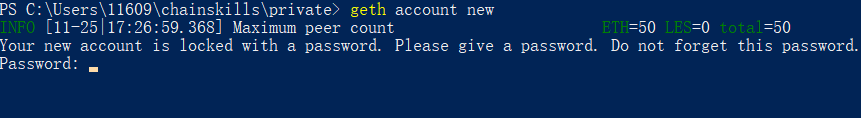
**注意：solcjs命令与Solidity编译器提供的solc包并不兼容，因此不能通过eth.compile.solidity()RPC的方式与以太坊客户端结合使用**

**5.3 以太坊集成开发环境**

1. **创建账户**

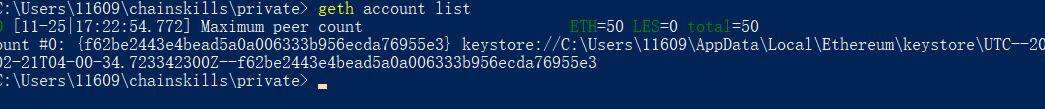
geth account new

**输入两遍密码后，生成账户地址。如下图**



**2.查看账户**

geth account list **如下图**



**3.初始化创世块的文件**

geth --datadir "./" init genesis.json

**注意：“./”表示以太坊的路径**

**运行以太坊私有链，首先要定义自己的创世区块，创世区块信息写在一个json格式的配置文件中，就是genesis.json。**

**其中，chainID制定了独立的区块链网络的ID。网络ID就在连接到其它节点的时候会用到，以太坊的网络ID是1，为了不与公有网络冲突，运行私有链节点的时候要指定自己的网络ID。不同ID网络的节点无法相互连接。配置文件还对当前挖矿难度difficulty、区块Gas消耗限制gasLimit等参数进行了设置。**

**4.启动以太坊**

geth --datadir "./" console

1. **启动挖矿qizhong**

miner.start()

**其中start可以设置参数，表示挖矿使用的线程数。第一次启动挖矿会先生成挖矿所需的DAG文件，当进度达到100%后，就会开始挖矿**

**6.停止挖矿**

miner.stop()

**挖矿所得的奖励会进入矿工的账户，这个账户叫做coinbase，默认情况下coinbase是本地账户中的第一个账户，可以通过miner.setEtherbase()将其他账户设置成coinbase。另外注意第一次停止挖矿需要一段时间，因为节点需要为工作证明算法生成1GB数据集。**

**5.4 以太坊编程接口**

**5.4.1通过json rpc编译合约**

**1.通过json rpc编译合约**

curl --data '{"jsonrpc":"2.0","method": "eth\_compileSolidity", "params": ["contract Multiply7 {event Print(uint);function multiply(uint input) returns (uint) {Print(input \* 7);return input \* 7;}}"], "id": 5}' localhost:8545

**提示:port8545:Connection refused**

**2.开启rpc端口**

geth --datadir "./" --rpcport "8545" --rpc console

1. **开启8545端口之后执行编译命令**

curl --data '{"jsonrpc":"2.0","method": "eth\_compileSolidity", "params": ["contract Multiply7 {event Print(uint);function multiply(uint input) returns (uint) {Print(input \* 7);return input \* 7;}}"], "id": 5}' localhost:8545

1. **添加头部后执行编译命令**

curl -H "Content-Type: application/json" --data '{"jsonrpc":"2.0","method": "eth\_compileSolidity", "params": ["contract Multiply7 {event Print(uint);function multiply(uint input) returns (uint) {Print(input \* 7);return input \* 7;}}"], "id": 5}' localhost:8545

**注意：推荐使用Remix进行合约的部署任务**

**5.4.2通过JavaScripts API编译合约**

1. **定义合约**

var source = 'contract Multiply7 {event Print(uint);function multiply(uint input) returns (uint) {Print(input \* 7);return input \* 7;}}'

1. **编译合约**

var compiled = web3.eth.compile.solidity(source)

**5.4.3 搭建Remix**

**Remix是一个[开源](https://link.juejin.im/?target=https://github.com/ethereum/remix)的用于[Solidity](https://link.juejin.im/?target=https://solidity.readthedocs.io/en/develop/)智能合约开发的Web端IDE，提供基本的编译、部署至本地或测试网络、执行合约等功能。Solidity是以太坊Ethereum 官方设计和支持的开发语言，专门用于编写智能合约。remix是基于浏览器的在线编译器,我们可以选择在线编译，但是有时候我们会遇到一些问题，所以我们要搭建自己的本地remix**

**官方在线编译器**

**<https://ethereum.github.io/browser-solidity>**

**[https://remix.ethereum.org](https://remix.ethereum.org/)**

**两种方法安装本地Remix(windows)**

**官方地址:**https://github.com/ethereum/remix-ide

**方式一：**

git clone https://github.com/ethereum/remix-ide.gitcd remix-ide

npm install

npm run build && npm run serve

**方式二：已经做为npm的一个module了**

npm install remix-ide -g

remix-ide

**启动命令：**

**npm start**

**访问Remix：**

**<http://127.0.0.1:8080>**

**5.4.4 Truffle**

**Truffle是基于Solidity语言的一套开发框架，它简化了去中心化应用（Dapp）的构建和管理流程。本身是采用Javascript编写，支持智能合约的编译、部署和测试。使用 Truffle 开发有一以下优点：**

**1.内置智能合约编译，链接，部署和二进制（文件）管理。**

**2.可快速开发自动化智能合约测试框架。**

**3.可脚本化、可扩展的部署和迁移框架。**

**4.可管理多个不同的以太坊网络，可部署到任意数量的公共主网和私有网络。**

**5.使用 [ERC190](https://github.com/ethereum/EIPs/issues/190) 标准，使用 EthPM 和 NPM 进行包装管理。**

**6.支持通过命令控制台直接与智能合约进行交互。**

**7.可配置的构建管道，支持紧密集成。**

**8.支持在Truffle环境中使用外部脚本运行器执行脚本。**

**我们将简单说明下如何使用Truffle**

# **5.4.4.1安装 Truffle**

**只要通过以下命令就可以安装Truffle**

**npm install -g truffle**

# **5.4.4.2编译合约**

**所有合约都位于项目的contracts/目录中。由于合约是用[Solidity语言](https://learnblockchain.cn/docs/solidity/)编写的，所有包含合约的文件都将具有.sol文件扩展名。相关的 Solidity [库](https://learnblockchain.cn/docs/solidity/contracts.html" \l "libraries)也将有一个.sol扩展名。使用truffle init命令创建的空Truffle[工程](https://learnblockchain.cn/docs/truffle/quickstart.html)会生成一个用于部署的Migrations.sol合约文件。如果我们使用[Truffle Box](https://truffleframework.com/boxes)来创建工程，则会有多个合约文件。**

**5.4.4.2.1编译命令**

**要编译Truffle项目里的合约，请切换到项目工程所在根目录，然后在终端中键入以下内容：**

**truffle compile**

**首次运行时，将编译所有合约。 在后续运行中，Truffle将仅编译自上次编译以来有更改的合约。 如果我们想覆盖此行为，可以使用 --all 选项运行上面的命令。**

**5.4.4.2部署合约**

**迁移脚本（JavaScript文件）可帮助我们将合约部署到以太坊网络。 这些文件负责暂存我们的部署任务，并且假设我们的部署需求会随着时间的推移而发生变化。 随着项目的发展，我们将创建新的迁移脚本，以进一步推动区块链的发展。 先前运行的部署记录通过特殊的Migrations迁移合约记录在链上，详细信息如下。**

**5.4.4.2.1部署命令**

**Truffle migrate**

**这将部署在项目的migrations目录中的所有迁移文件。最简单的迁移只是一组管理部署脚本。如果我们的迁移先前已成功运行，则truffle migrate将从上次运行的迁移开始执行，仅运行新创建的迁移。如果不存在新的迁移，truffle migrate将不会执行任何操作。我们可以使用--reset选项从头开始运行所有迁移。对于本地测试，确保在执行migrate之前安装并运行了[Ganache](https://truffleframework.com/ganache)等 测试区块链。**

**5.4.4.3调试合约**

**Truffle 集成了一个调试器，以便我们可以调试合约进行的交易。此调试器和传统开发环境中使用的命令行调试程序有点像。调试区块链上的交易与调试传统应用程序（例如，用C++或Javascript编写的应用程序）不同。 在区块链上调试交易时，没有实时运行代码; 相反，我们将逐步执行该交易的历史执行，并将该执行映射到其关联的代码上。 这为调试提供了许多自由，因此我们可以随时调试任何交易，只要我们拥有交易与之交互的合约的代码和工件(artifacts)即可。 可以将这些代码和工件（artifacts）类似于传统调试器所需的调试符号。**

**5.4.4.2.1调试命令**

**要使用，获取要调试的交易hash，运行以下命令启动调试器：**

**$ truffle debug <transaction hash>**

**如果想打开调试器以使其准备调试接下来的交易，可以简单地运行：**

**$ truffle debug**

**无论哪种方式启动的调试器，一旦它开始运行了，它就不仅限于调试启动时附加的交易（号）; 可以卸载当前交易并加载新交易，**

**5.4.5通过在线或本地Remix编译合约**

1. **本地私有链监听RPC端口**
2. **以web方式连接私有链，必须开启-rpcrpccorsdomain参数，设置IP请求白名单 \* 为所有 成功后读取Account列表**

geth --datadir "./" --rpcport "8545" --rpccorsdomain "\*" --rpc console

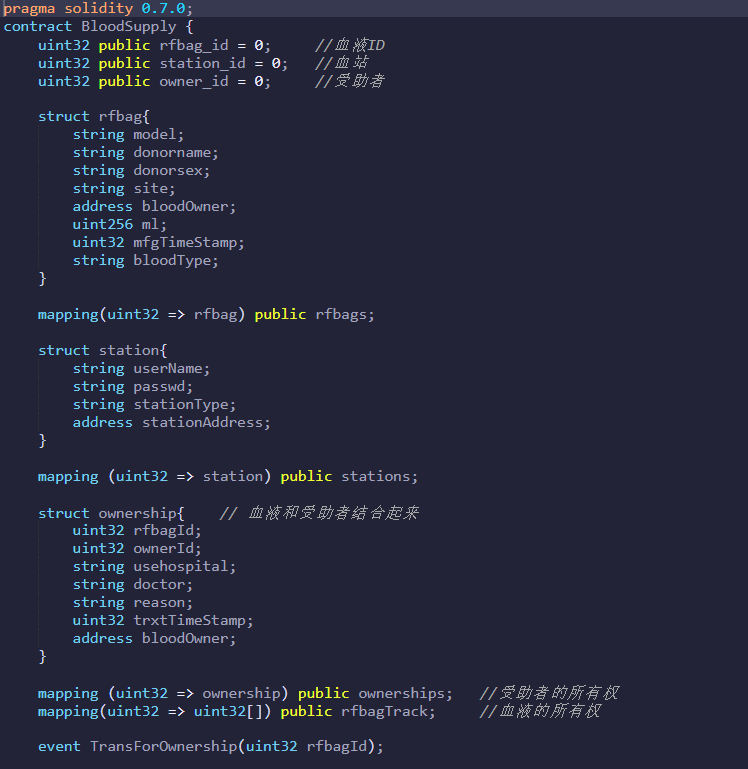
1. **成功后读取Account列表**
2. **部署信息**
3. **控制台查看合约部署的地址**
4. **发起交易**

curl -H "Content-Type: application/json" -X POST --data '{"jsonrpc":"2.0", "method": "eth\_sendTransaction", "params":[{"from":"address", "to":"address","data":"0xc6888fa10000000000000000000000000000000000000000000000000000000000000006"}], "id": 8}' localhost:8545

1. **交易提交**
2. **查看交易**

eth.getTransaction("address")

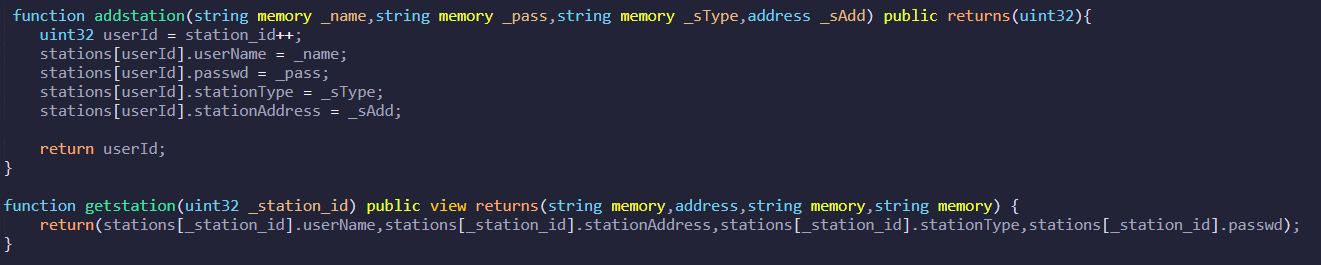
**5.5.1合约代码**



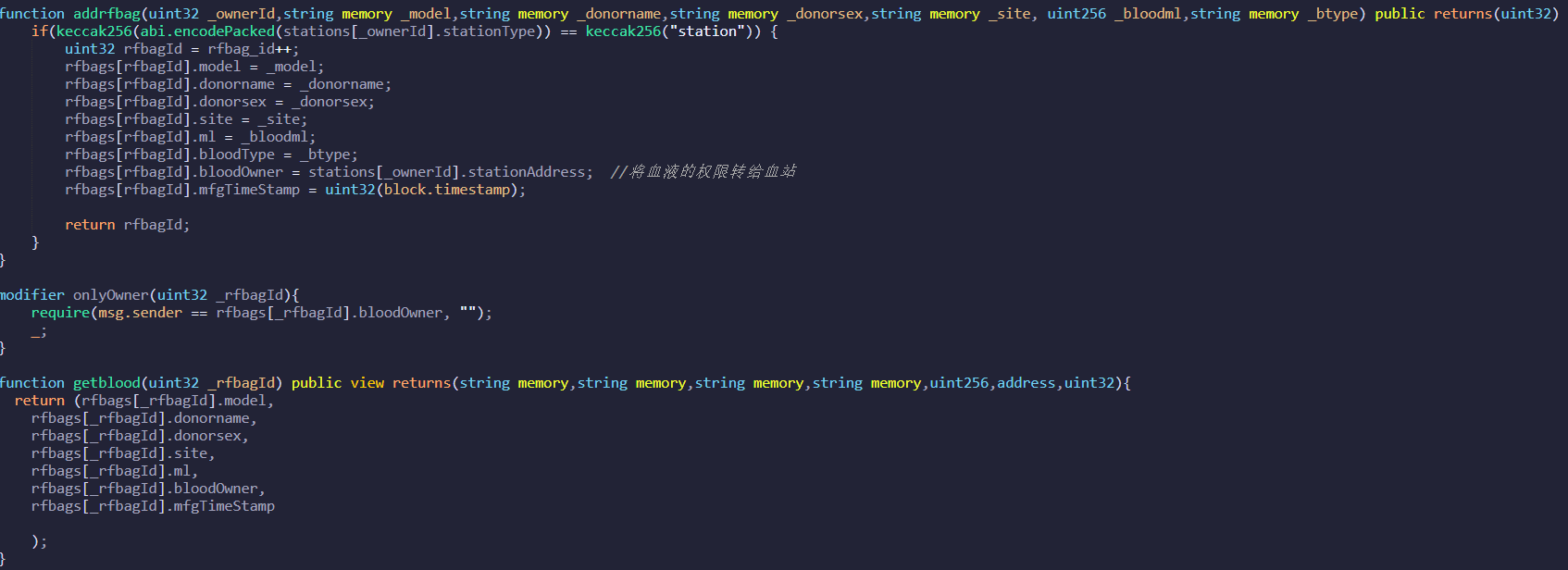
**第一步：我们创立了三个对象rfbag\_id;station\_id;owner\_id;这些分别对应着血液；血站和使用医院。其后，我们设置其结构体，第一个结构体struct rfbag结合本方案前文所提，对于存储血液信息并将上链的带有RFID标识的血袋我们设置了 string model（血型）;string donorname（献血者名字）;string donorsex（献血者性别）;string site（献血地点）;uint256 ml（献血量）。这些结构体将作为信息保存。address bloodOwner;uint32 mfgTimeStamp;string bloodType;这三个结构体是后面进行血液转移重要的部分；第二个结构体struct station，结合前文所描述的功能我们将有string userName（血站名字）；string passwd（入站时间）。String stationType（类型）;address stationAddress（血站地点）这两个是之后进行权限转移的重要部分；第三个struct ownership就是所有者（受助者）。结合本方案的应该要实现的那样，我们这边定义了uint32 rfbagID;uint32 onweID这两个重要结构将通过后面的功能实现返回所使用的血袋ID，和所有者ID。以便查询血袋信息。string usehospital（使用的医院）;string doctor（主治医生）;string reason（病因），这些将通过手动添加保存并上链。**

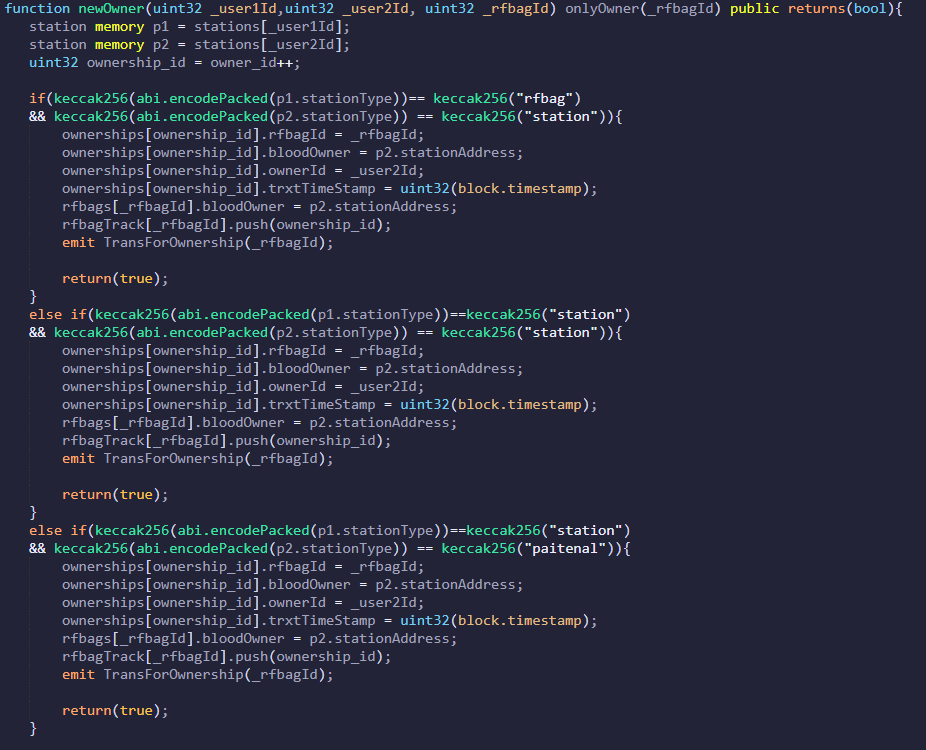
**第二步：前一步我们完成了该方案的三个重要角色的结构，而后我们将用一些功能实现这些结构体。根据逻辑顺序，第一个功能应该是function addstation（增加血站）**

**我们将添加第一步所述的结构体信息，并且通过get station来得到所提交的信息。**



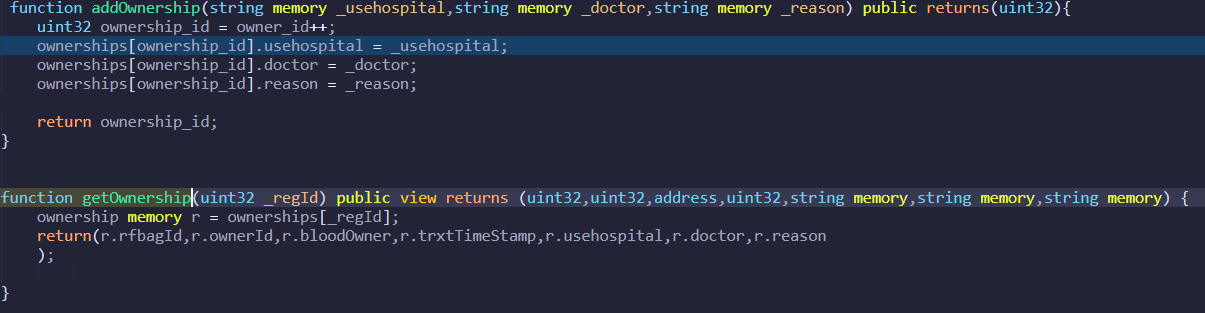
**我们将添加第一步所述的结构体信息，并且通过getstation来得到所提交的信息。**

**第二个功能则是添加血袋了，如果前面没有血站来以供血袋转移，交易则无法进行。这一步与添加血站一致，我们将提交第一步结构体中所构造的信息，再通过get blood方法得到所提交的方法，与第一个功能有所不同的是，这里有一个权限转移modifier onlyOwner(),这一步目的是为了将血液的权限转移给血站，以供后面血站将血液转交给医院使用。**



**在完成前面的准备之后，我们就可以进行血液权限的转移了，在这里血站将把存储在这里的血液转移给医院。**

**在完成转移后，根方案前面所提到的功能，我们将又有两个方法，一个是function addOwnership（添加受助者信息） function getOwnership（得到受助者信息）这个功能和前面add tation和getstation一样。**



这样，我们就可以开始设计前端等应用与实践。

**六：应用与实践设计**

**本文讨论了基于区块链的新型献血管理研究办法，设计了如下的功能界面**



**图6-1**

**如图6-1所示是基于区块链的电子献血证运行截图，其中左上角是献血者献血记录一览；右上角所示为证照的二维码，包含该证件在区块链上的业务信息、区块号、区块地址、交易地址等数据，查证人可通过扫描该二维码获取持证人在区块链上的证照信息；图左下角是使用RFID的通知推送，由于每个血袋都有且只有一个RFID也就对应唯一一个血液ID。并且血袋在运输过程中的日期、运输车、血站的编号使用者的住院号、主治医生等一列信息都将记录在RFID上，然后再保存到区块链上，从而保证其信息不可篡改，且可追溯。**

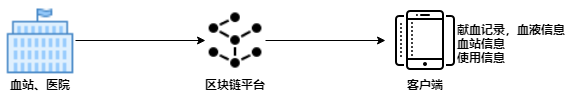
**七：新型献血管理的应用前景**

1. **实现办事“零跑动”**

**我们都知道，无偿献血者及其配偶和直系亲属享有如下用血权力：**

1. **、无偿献血的公民，本人有享受免费用血的权利。无偿献血者（不包括自身储血）累计献血量在800毫升以下的，按献血量的三倍免费用血；  
    (2)、累计献血量在800毫升（含800毫升）以上的，终身免费用血。无偿献血者的配偶和直系亲属（指父母、子女）需要临床用血的，可以按献血者实际献血量等量免费用血。**

**但是，现在这个社会证明“我爸是我爸”都如此的困难，如何证明我献过血，那是我的血就更加困难了，再加上原来的纸质献血证容易丢失，信息可信度不高，在用献血证获得报销或者减免的过程中，难免少不了官僚主义的现象的发生，并且需要病人加开各种材料的证明，需要人们在不同机构来回奔波；再加之，如果患者遗失一些证件，补办过程也是相当的繁琐，这些情况往往让一些病人放弃使用献血证。但是实行新的献血管理方式之后，献血证变为电子证存储在你的移动端上；信息变成数字信息存储在二维码上面方便携带和使用，可以实现证照的无纸化审批，减少了纸质证照的使用，降低了社会成本与资源浪费，缓解了证件多、用证烦等用证用信息的乱象。如图7-1所示**

****

**图7-1应用后**

1. **链上社会**

**基于区块链的新型献血管理模式在不改变政府现有管理模式及制度的前提下，通过系统构建了献血者、献血点、运输、血站、医院以及患者之间的联系，实现了申请、查证、验证以及用证的数字化管理，优化了献血流程的服务，推动了当今社会推进信息化建设和应用以达到深化行政审批制度改革的目的，对社会献血公益事业的发展起到了积极的作用，不仅如此，运用区块链技术我们不仅可以发展出的基于区块链的献血管理研究，我们可以运用这一技术推出许多技术，例如基于区块链的成绩认证、身份识别、体育产业、卖鞋、卖游戏账号等一系列相关技术。届时，区块链技术将成为智慧城市发展的重要组成部分，在城市发展过程中，社会和政府依托电子政务，在其管辖范围最大限度发 挥职能效能，创造更加智慧的城市环境**。**因区块链的不可篡改性、可追溯性可以提升政府部门行政服务效率，故其也会推动政府和社会向智慧转型，向着为民、务实、廉 洁、高效的服务型政府转变。这样也会提升服务机关在民众心中的满意水平如上图7-1所示。**

1. **助力诚信社会**

**以前因为血液的流通信息不明，使用献血证用血的流程繁琐等一系列问题，扰乱了献血公益事业的生态环境，降低了公民献血的积极性，同时也损失了社会的诚信，也给献血这一个公益事业带了不必要的麻烦。虚假的信息危害很多，因此保证证件和信息的真实性尤为重要，基于区块链的献血管理系统所运行的电子献血证因为其多重加密，并且需要配合身份证明信息共同使用的等因素，大大提升其安全性。首先丢失和伪造献证的概率被降到了最低，保障了献血证真实性。不仅如此，记录在献血证里面的信息，也是经过献血证验证而经过多重加密的，保持了信息的可信度，使得信息可追溯，这也极大地维护了献血者的权益。基于区块链的献血管理通过信息技术手段有力地加强了社会诚 信建设，符合十八届四中全会《中共中央关于全面推进依法治国若干重 大问题的决定》中对信用工作的要求。**

**七：方案总结**

**本篇方案主要结合我国献血管理现状,在分析我国现在献血业的不足之后，然后重点深入介绍了区块链技术，在介绍了区块链的底层技术之后，利用区块链的这些特点，研究区块链技术在献血公益事业领域中的应用。也就是本文的基于区块链的献血管理研究。**

**区块链技术并不是突然出现的新技术，而是已有技术的真个和，包括了基于密码学的隐私保护技术，即hash算法中的SHA256函数算法、加密算法中的非对称加密技术，以及解决拜占庭问题的分布式共识机制等。在本篇方案中，系统的开放是基于以太坊平台，其中编写智能合约主要是在remix平台下，利用献血管理的智能合约来实现本篇方案中内核的主要功能，然后利用truffle框架轻松构建了整个分布式应用系统。在献血管理系统中，献血者信息、献血者献血、血液运输、医院用血这些信息都永远的记录在了区块链之中，同时利用区块链的特性，保证了这些数据的真实可靠，数据的可追溯。**