单位代码：10293　密　级：公开

南京邮电大学

硕士学位论文

论文题目：基于区块链的供应链溯源系统关键技术研究

学　　号	1019010307
姓　　名	张斌
导　　师	李大鹏
学科专业	通信与信息系统
研究方向	区块链应用
申请学位类别	工学硕士
论文提交日期	2022.04

Research on key technologies of blockchain-based supply chain traceability system

Thesis Submitted to Nanjing University of Posts and Telecommunications for the Degree of

Master of Science in Engineering

By

Bin Zhang

Supervisor: Prof. Dapeng Li

April 2022

摘要

随着社会经济的不断发展和人民消费水平的提升，消费者对物质生活水平的需求不再仅仅体现在数量上，同时对商品的质量也有了更高的要求。供应链溯源技术是保障商品质量的首要方案，但在传统中心化存储的供应链溯源系统中存在信息孤岛、恶意企业对溯源信息进行修改不易被察觉和溯源难的问题。区块链具有去中心化、不可篡改性和可追溯性等特点，这些特点使其在商品溯源方面具有不可替代的优势。然而由于供应链溯源管理的参与方较多且存储信息量较大，如果直接使用区块链技术对供应链溯源平台进行实现，共识算法本身所存在的缺陷和区块链网络数据存储压力大的问题将会成为二者结合的主要瓶颈。因此本文针对这两个问题分别进行优化与改进，以提高区块链技术在供应链溯源中的可用性。论文主要研究工作如下：

（1）针对委托权益证明共识算法存在记账节点的选举受拥有数字货币数量多的节点影响较大、区块验证时间长和对被选为记账节点的恶意节点不能及时处理的问题。本文提出一种基于工作量证明和信誉值的委托权益证明共识算法。该算法首先通过引入工作量证明的方式选举出本轮的共识节点，之后非共识节点只能对共识节点进行投票选举出记账节点，以提高区块链网络的去中心化程度。在投票选举阶段，每个非共识节点都需完成两次投票，第一次投票是投给共识节点选举记账节点，第二次投票是投给非共识节点选举验证节点。最终的选举结果也会把该节点的信誉值考虑在内，保证记账节点和验证节点的可靠性。最后在共识阶段由验证节点对产生区块进行验证以降低交易延时，同时引入记账节点更换机制提升对恶意节点的处理速度，保证区块链网络的健壮性。实验结果表明，本文所提共识算法相较于原始算法在记账节点分布率、交易时延和系统吞吐量上都有一定的提升。

（2）为减轻区块链网络的存储压力，提出面向区块链溯源的链下扩展存储方案。该方案中，数据的不可篡改性由哈希函数的不可逆向推导和区块链的不可篡改共同保证，数据的有效性由 SM2 签名算法保证。企业通过部署智能合约将数据哈希和签名结果上链，数据明文则存在链下数据库中。本企业的每条数据库记录包含了数据明文，链上数据在区块链上存储的地址，对应上游企业相关数据的数据库索引和链上数据存储地址，实现溯源。最后基于该方案采用以太坊区块链通过 SSM（Spring+SpringMVC+Mybatis）开发框架对溯源系统进行实现，系统设计了详细的数据库表和智能合约，为区块链技术和供应链溯源更有效地结合提供了一种思路。

关键词：供应链溯源 ， 区块链 ， 共识算法 ， SM2 ， 智能合约

Abstract

With the continuous development of social economy and the improvement of people's consumption level, consumers' demand for material living standard is no longer only reflected in quantity, but also has higher requirements for the quality of commodities. Supply chain traceability technology is the primary solution to ensure the quality of commodities, but there are problems in the traditional centralized storage supply chain traceability system, such as information island, malicious enterprises' modification of traceability information is not easy to be detected and traceability is difficult. Blockchain technology has the characteristics of decentralization, immutability and traceability, which make it irreplaceable in the traceability of commodities. However, due to the large number of participants in supply chain traceability management and the large amount of storage information, if the supply chain traceability platform is directly implemented using blockchain technology, the defects of consensus algorithm itself and the large pressure of blockchain network data storage will become the main bottleneck of the combination of the two. Therefore, this paper optimizes and improves the two problems respectively to improve the usability of blockchain technology in supply chain traceability. The main research work of this paper is as follows:

(1) For the Delegated Proof of Stake consensus algorithm, the election of accounting nodes is greatly affected by nodes with a large number of digital currencies, the block verification time is long, and the malicious nodes selected as accounting nodes cannot be dealt with in time. This paper proposes a Delegated Proof of Stake consensus algorithm based on Proof of Work and Reputation. The algorithm first elects the consensus nodes of the current round by introducing proof of work, and then non-consensus nodes can only vote for consensus nodes to elect accounting nodes to improve the decentralization of the blockchain network. In the voting stage, each non-consensus node needs to complete two votes. The first vote is to vote for the consensus node to elect the accounting node, and the second vote is to vote for the non-consensus node to elect the verification node. The final election result will also take into account the reputation value of the node to ensure the reliability of the accounting node and the verification node. Finally, in the consensus stage, the verification node verifies the generated block to reduce the transaction delay. At the same time, the account node replacement mechanism is introduced to improve the processing speed of the blockchain network against malicious nodes and ensure the robustness of the blockchain network. The experimental results show that, compared with the original algorithm, the consensus algorithm proposed in this

paper has a certain improvement in the distribution rate of accounting nodes, transaction delay and system throughput.

(2) In order to reduce the storage pressure of the blockchain network, an off-chain expansion storage scheme for blockchain traceability is proposed. In this scheme, the immutability of data is guaranteed by the irreversible derivation of the hash function and the immutability of the blockchain, and the validity of the data is guaranteed by the SM2 signature algorithm. The enterprise uploads the data hash and signature result on the chain by deploying the smart contract, and the data plaintext is stored in the off-chain database. Each database record of this enterprise contains the data plaintext, the address where the data on the chain is stored on the blockchain, the database index corresponding to the relevant data of the upstream enterprise and the storage address of the data on the chain, so as to realize traceability. Finally, based on the scheme, the Ethereum network is used to implement the traceability system through the SSM (Spring+SpringMVC+Mybatis) development framework. The system designs detailed database tables and smart contracts, which provides an idea for a more effective combination of blockchain technology and supply chain traceability.

Key words: Supply Chain Traceability, Blockchain, Consensus Algorithm, SM2, Smart Contract

目录

第一章 绪论

1.1 研究背景与意义

供应链是通过上游和下游进行联系，把商品从生产端流向消费者的一个网状结构，它将供应链流程中各环节企业进行串联，提高了企业间的协同效率，让企业的经济效益实现最大化[1]。在一条完整的供应链上至少包含以下几类参与者：原材料供应商、运输商、商品制造商、零售商和最终消费者[2]。供应链的高效运作是经济发展的重要保障，因此如何保障供应链管理中信息更有效和准确的存储是很多学者都在研究的课题。由于在原材料到最终的消费者中间的各类企业数量不断增加，商品的生产流程也变得更加复杂化，而且随着经济全球化的发展和消费市场的不断扩张，各企业都在不断扩大自身产品生产规模，以更好满足市场的需求。这将会对商品溯源的难度提出进一步的考验，传统的中心化供应链管理模式也不再能适应当下发展现状[3]。在传统的溯源模式中通常面临着以下瓶颈：

（1）存在信息孤岛。一件商品在其从原材料到商品完成生产过程参与方较多，且每个企业都把生产数据存储在各自的数据管理系统中，这会使得每个企业之间的信息都是孤立存在的[4]。

（2）溯源信息真实性不可靠。目前供应链信息都是存储在中心化数据库当中。然而这种存储方式可能会对数据的完整性和可用性造成威胁，系统内数据容易遭到恶意篡改且数据造假成本低[5]。

（3）溯源信息追踪流程难。一件商品从原材料到最终的成品之间需要经过多家企业，各企业生产产品信息构建方法不同，传统的溯源模式不易进行追踪[6]。且企业的数字化程度也有所不同，这更增加了商品溯源信息追踪的难度。

近年来，全球多地出现“药物”、“粮食”、“商品”等方面的产品质量和安全问题也让每一个消费者认识到商品信息的可追溯性和供应链有效监管的重要性。可追溯性和可见性相关问题一直是企业向消费者提供高质量商品服务的关键问题[7]，建立一个数据完整的、可靠的供应链管理平台，让消费者对其购买商品从原材料、生产、运输、销售各阶段的信息能够全面掌握是供应链管理发展的首要大事。这不仅可以降低消费者对其购买的商品产生严重的信任危机、减少因为虚假商品导致市场秩序产生不稳定因素同时也能为各企业更好地承担自身社会责任提供良好的经营环境。

区块链具有去中心化、不可篡改性和可追溯性等特点，这些特点使其在商品溯源方面具

有不可替代的优势[8]。区块链技术可以在不完全可信的环境中实现可信的数据管理，同时也能够记录两个主体之间的交易，而无需中介机构的干预[9]。随着区块链技术的发展，人们普遍认为其可以颠覆性地改变社会现有的信任问题[10]。区块链技术可以对供应链溯源产生很多方面的影响，其中包括增强商品安全性；减少非法伪造；降低供应链中交易的信任风险等[11]。因此如何利用区块链技术与供应链溯源进行结合，实现供应链生产中各个环节信息更有效的形成关联，减少供应链企业之间的信息孤岛，提高各企业之间合作效率，进而使得整个供应链中商品信息的变得透明化是当前甚至今后很长一段时间需要讨论的热门话题。

在区块链发展的初期，其大多数使用的场景都是与数字货币这一概念相结合，与实际生活中的联系并没有其设想的那么多[12]。但随着以太坊区块链首次对智能合约这一概念进行实现，为区块链网络可以对除交易以外其他格式的数据进行保存提供了可能[13]。以太坊为开发者提供了图灵完备的开发语言，用户可以根据上传数据的需求对智能合约进行开发并把其部署到以太坊区块链上进行保存[14]。

区块链技术由于发展时间不长，使得其与实际生活中的场景相结合时仍会有诸多不足。在利用区块链技术对商品信息进行溯源时，主要存在以下二个问题：一是供应链中参与方多，现存的共识算法很难即满足去中心化强又满足吞吐量大的特点[15]。二是由于区块链是一种分布式的系统，对于区块链中存储的数据所有节点都会进行备份，然而商品的溯源信息存在数据量大的特点，这必然会导致溯源数据占用内存过大的问题[16]。因此只有对这两个问题进行优化，才能让区块链技术得到更好的应用，进而提高区块链技术与供应链溯源管理产生更好“化学反应”的可能性。

1.2 国内外研究现状

1.2.1 区块链技术研究现状

区块链采用的是一种非集中式、可靠且不易被篡改的数据库存储形式[17],它的发展过程可以分为三个阶段[18]。第一个阶段是数字货币，其中最具有代表性的是由中本聪所提出的比特币，它是一种数字货币，这种数字货币通过区块链公共账簿在点对点网络上进行交易[19]。比特币是使用区块链的应用之一，在这个阶段中，人们只是把区块链作为一种数字货币交易的方式，并没有过多认识到区块链背后所具有的价值。第二个阶段是可编程化区块链，其中最具有代表性的是由 Vitalik Buterin 提出的以太坊区块链，它在数字货币的基础上引入了智能合约等重要概念，使区块链真正的实现了可编程[20]。在这个阶段中，区块链在一定程度上实现

了与不同场景结合的可能性。第三个阶段是超越货币、金融范围的区块链应用场景，其中最具有代表性的是由 IBM 和 DAH 联合开发的一款区块链 Fabric[21]。随着区块链的发展，人们逐渐也认识到区块链可以与生活中的不信任环境进行结合，这将会为我们生活中的方方面面带来巨大的变化，比如物联网、能源、金融、医疗保健和政府等领域[22]。区块链网络为我们构建了一个完全可信的网络环境，这将大大提高我们的办事效率[23]。

共识算法是区块链中最重要的部分，它是区块链中所有节点在没有第三方约束的条件下对保存数据达到最终一致性的关键[24]。在定义区块链网络的基础架构时，出于不同角度的考虑会把其划分成四层和六层两种情况，但不管是以那种划分进行方式，共识层都是单独存在的一层，这也从另一方面体现出一个好的共识算法对于区块链技术来说是至关重要的。在区块链的发展过程中，众多学者都在对共识算法能够更好地适应实际的生产需求而提出自己的解决方案。

比特币使用到的共识算法是工作量证明（Proof of Work，PoW）[25]，它是通过计算找到一个符合一定难度值的随机数来进行区块生产的，这种方式是以浪费大量算力和电力的基础上完成网络中节点的共识[26]。后续学者针对此问题，在 PoW 共识算法的基础上提出了一些优化方法。文献[27]中为了降低工作量证明方法在物联网网络中的延迟问题，把可编程门阵列这一概念引入进来用以提高系统共识速度。在文献[28]中为了降低工作量证明共识算法浪费大量算力和电力的问题，结合权益证明共识算法提出了一种基于工作量证明和权益证明改进的区块链共识机制 PoWaS，实验表明该算法不仅可以减少算力竞争，同时也可以提高出块效率。在文献[29]中为了解决高功耗和 51%攻击的问题，引入几个验证轮来改进 PoW 共识协议。任何给定的共识节点都应该在参与下一轮游戏之前解决当前回合的游戏，任何解决了当前轮的节点只有在其他节点找到了预定数量的解时才能进入下一轮。

2011 年一位名为 Quantum Mechanic 的网友针对 PoW 共识算法共识时间长且会造成大量资源浪费的问题在比特币社区上首次提出了权益证明（Proof of Stake，PoS）共识算法。PoS 共识算法会根据每个网络节点所拥有数字货币的多少和数字货币拥有的时间两方面对其挖矿难度进行动态调整，也就是说节点拥有数字货币时间越长或者占有数字货币越多，则其产生区块的概率就越大。PoS 共识算法在一定程度上对 PoW 共识算法进行了优化，但是仍然存在吞吐量较小的问题而且容易产生出现节点集中化的现象[30]。文献[31]中为了对 PoS 共识算法中产生区块节点的收益更合理，利用博弈论中的沙普利值对 PoS 共识算法中的出块奖励进行优化，减小了网络节点的经济分层，提高了股权较少节点获取奖励的机会和区块链网络去中心化的特性。在文献[32]中为了实现区块链的一致性工作和存储的并行化，作者提出了一种基于分片的 PoS 区块链协议，该协议采用了公平和动态分片管理克服现有基于分片的共识协议的

这种局限性。

随后 Dan Larimer 在其公司创立的 Bitshares 中提出了委托权益证明（Delegated Proof of Stake，DPoS）共识算法，它不再采用挖矿的形式，而是采用投票的方式选出见证人节点进行出块[33]。每个节点拥有的票数与其自身所拥有的数字货币数量相关。文献[34]中针对 DPoS 共识算法中存在见证人节点分布不均匀和区块验证较慢等问题提出一种基于节点行为和民意的委托权益证明（QDPoSB）共识算法，该算法通过引入 Borda Count 选举出记账节点，采用实用拜占庭容错共识算法加快了区块验证的速度提高了系统的确认延迟。文献[35]中作者为使区块链与工业互联网相结合，在 DPoS 共识算法的基础上提出委托邻近性证明(DPoP)，该算法利用现有的传感器位置，认定接近传感器发出消息的节点被选为记账节点，DPoP 旨在减少浪费，通过限制共识操作所需的节点数量来提高吞吐量，并随着工业互联网网络的不断增长，提高区块链共识算法的可伸缩性和灵活性。文献[36]中在车联网中为了确保车与车之间数据共享的安全性和可追溯性，在 DPoS 共识算法的基础上提出一种两阶段安全增强方案：记账人选择和出块验证。在第一阶段中设计了一个基于声誉的选择方案，以保证记账人的安全性。在第二阶段，为防止记账节点之间内部串通产生恶意区块，新生成的区块由备选节点进一步核实和审计。结果表明该方案能够在车联网的场景中保证数据共享的有效性。

1.2.2 区块链溯源存储研究现状

随着社会经济的发展，在当今物质生活不再那么匮乏的情况下，消费者也开始对所购买商品的质量上也有了更高的要求。然而假冒伪劣产品不易被查别一直是供应链行业急需解决的痛点问题。区块链技术的出现为解决商品信息溯源提供了一个有效的方案[37]，但如果直接将区块链技术与数据溯源进行结合，区块链网络中的数据存储压力将成为二者结合的主要瓶颈。针对区块链在数据溯源中的应用问题，众多学者也从理论和技术的层面提出了自身的见解。

目前提高区块链存储的扩展性的方案主要有三种：分别为侧链存储、分片存储和有向无环图存储。侧链存储是指整个系统中包含主链和侧链二部分，侧链用来分担主链的存储和计算压力[38]。分片存储包含两种方式：第一种是将节点分成不同的区域并行处理区块链网络中的交易以提高交易的处理速度，这被称为交易分片[39]。第二种是将区块链中的完整数据分别存储不同的节点中以减少单个节点的存储原理，这被称为状态分片。有向无环图是一种拓扑排序的数据结构[40]，通过这种方式进行数据存储时不在存在区块的概念，而是把交易作为独立的单元进行共识，减少了数据打包的时间。

文献[41]中认为当前区块链系统无法在不损失速度和时间效率的情况下对数据量庞大的物联网数据进行存储与扩展，因此使用侧链和离线数据存储的概念来缓解区块链的可伸缩性问题。提出一种使用区块链技术的物联网系统安全和维护的架构框架，通过智能合约用于执行数据身份验证、授权和跟踪所有活动。结果表明所提架构可以有效的用于物联网系统。文献中[42]为解决现有分片处理协议中通信量大，安全性弱的问题。提出了 RapidChain，它是第一个基于分片的公共区块链协议。在该方案中采用了委员会共识算法，增加区块链网络的吞吐量和安全性。通过跨分片事务验证技术，减少区块链中事务的传播范围和通信量。文献[43]中为缓解区块链中存储压力，作者将区块链中不同的节点组织成一个集群，并将整个链的所有块分配给每个集群中的节点。在集群内部将区块链中完整数据分配给不同的节点，为使总查询代价最小化，提出了一种启发式算法 GAPG 来解决该问题。

部分学者也采用了其他方式对区块链的存储能力进行扩展。文献[44]中提出了一种基于 IPFS 的区块链存储模型，解决了溯源信息在区块中的存储问题以及访问问题。在提出的存储模型中，系统将溯源信息存储在 IPFS 分布式文件系统存储上，并将溯源信息的 IPFS 哈希值返回到区块链的块中。溯源数据所占内存大小远大于其哈希值，因此通过引入 IPFS 网络作为溯源数据存储层可以减少区块链中的溯源数据占用内存大小。文献[16]中对于数据量大的数据溯源场景采用了数据分割的形式进行存储以减少链上数据占用内存。该方案依据溯源数据的重要程度对数据进行划分，重要数据直接保存在链上，不重要的数据则存储在链下数据库中，并把链下数据进行哈希运算，然后把得到的哈希值与重要数据一同存储在链上。文献[45]中为解决农产品溯源数据全部上链增加区块链网络存储压力的问题，通过结合 LZ78 和哈夫曼编码的方式提出一种数据压缩方法（LZ-Hf）对农产品各个阶段的溯源数据进行压缩以减小溯源数据所占用存储大小的问题。结果表明所提方案具有较高的压缩效率，可以减小区块链的存储压力。

1.3 论文内容及章节安排

本文主要工作是以区块链与商品溯源进行结合为背景。为保证区块链网络的共识效率与去中心化特性，对委托权益证明共识算法现有问题进行改进。由于区块链是一种分布式技术，在供应链中参与的企业较多且溯源数据量大，如果溯源数据全部直接上链会导致占用大量内存的问题，提出一种链下扩展存储方案。具体工作主要包含下面几点：

（1）根据委托权益证明共识算法中存在的不足，提出一种基于工作量证明和信誉值的委托权益证明共识算法。该算法首先通过引入工作量证明的方式选出本轮的共识节点，之后非

共识节点只能对共识节点进行投票选举出记账节点，以降低拥有数字货币数量多的节点对记账节点选举的影响，提高区块链网络的去中心化程度。在投票选举阶段，每个非共识节点都需完成两次投票，且信誉值也会被作为选举结果的一部分进行考量。第一次投票是投给共识节点选举记账节点。第二次投票是投给非共识节点选举验证节点，保证了验证节点具有较高的可靠性。最后在共识阶段由验证节点对产生区块进行验证以降低交易延时，同时引入记账节点更换机制提高对恶意节点的处理速度。通过实验分析表明改进算法在一定程度具有更好的实用性。

（2）根据区块链技术在面对供应链规模较大和数据量较多时存在占用大量内存的问题，提出一种面向区块链溯源的链下扩展存储方案。该方案中，数据的不可篡改性质由哈希函数的不可逆向推导和区块链的不可篡改共同保证，SM2 签名算法保证了数据上传者身份的可靠。企业通过部署智能合约将数据的哈希值和对哈希值签名的结果保存在链上，数据明文则存在链下数据库中。本企业的每条数据库记录包含了数据明文，链上数据在区块链上存储的地址，对应上游企业相关数据的数据库索引和链上数据存储地址，实现溯源。最后基于该方案采用以太坊网络通过 SSM 开发框架对溯源系统进行实现，系统设计了详细的数据库表关系和智能合约。

本文由五个章节组成，具体安排如下：

第一章为绪论。首先对本文的研究背景与意义进行介绍。然后分别对区块链技术和区块链溯源存储研究现状进行介绍。最后阐述了本文主要工作及章节安排。

第二章为相关背景知识介绍。首先给出了区块链基本概念与相关技术介绍。然后阐述了密码学基础，其中包括：哈希算法、非对称加密算法、SM2 加密算法。最后对本文第四章搭建溯源系统使用到的以太坊区块链及智能合约的概念及原理进行介绍。

第三章为基于委托权益证明的共识算法研究。首先对委托权益证明共识算法原理与问题进行分析。接着对所提基于工作量证明和信誉值的委托权益证明共识算法进行了详细介绍，其中包括：节点类型、共识节点选举阶段、信誉模型与投票选举阶段、共识阶段与记账节点更换机制。最后从节点参与度、交易延迟和系统吞吐量三个方面对所提算法进行实验分析。

第四章为面向区块链溯源的链下扩展存储方案研究。首先给出了基于区块链的供应链溯源模型。然后对链下扩展存储方案设计进行详细阐述，其中包括：链上数据存储设计、链下扩展存储方案、溯源数据验证与问题子企业定位。最后基于该方案采用以太坊区块链通过 SSM 开发框架对溯源系统进行实现。

第五章为总结与展望。对本文所做工作进行回顾与总结，分析并指出本文中存在的不足以及对未来的研究内容进行展望。

第二章 相关背景知识介绍

2.1 区块链基本概念与相关技术

2.1.1 区块链定义与结构

区块链是网络数字化和现金技术化的一种安全底层应用技术，它被定义为一种用来对交易按照时间为顺序的数据进行分布式存储的数据库，是由一系列数据块通过加密方式连接而成的分布式账本，系统中的每一个节点都对该账本进行保存和维护[46]。在使用区块链技术发起交易时，所有的用户都是通过区块链网络为其分配的特定账户进行处理的，因此交易双方并不知道对方的真实身份，所以区块链中所包含的信息是完全匿名的[47]。区块链中有新的区块加入到链上时，链的长度会不断增加[48]，如果有两个区块同时产生会造成分叉的问题，那么区块链会选择最终区块数目最长的那条链作为主链来维护网络中的有效交易。区块链工作在一个没有中心机构管理的环境中，该环境主要由加密算法、P2P 网络和分布式共识算法等技术进行维护。也正是因为以上所述构建方法使得区块链具有去中心化、不可篡改和可溯源的特性[49]。

比特币系统作为区块链的最早应用，其区块的底层结构可以清楚地对区块链中数据存储的方式进行展示，图 2.1 展示了比特币区块链中每个区块内部的具体结构及其形成链状结构的方式。每个区块中包含区块头和区块体两部分[50]。

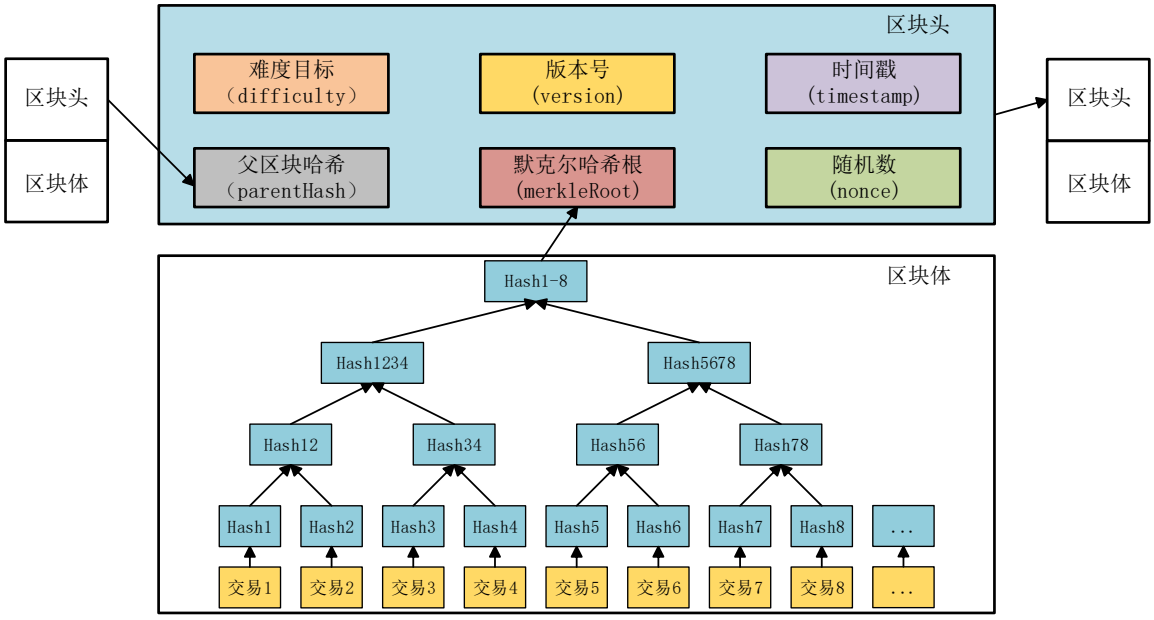

图 2.1 区块结构图

区块头由难度目标、版本号、时间戳、父区块哈希、默克尔根哈希、随机数六部分组成。难度目标占用内存大小为 4 字节，它表示当前时间段工作量证明难度的目标值，区块链中的节点也被称之为矿工需要完成该计算，才能发布区块，进而获得出块奖励，如果其难度值为 2，则表示矿工需要计数哈希结果的前二位为 0。版本号占用内存大小为 4 字节，它对当前区块链的版本进行规定。时间戳占用内存大小为 4 字节，它表示当前区块产生的时间。父区块哈希占用内存大小为 32 字节，它表示该区块上一个区块的哈希值，通过该字段可以从当前区块一直向前查询，直到查询到创世区块为止，从而使区块链中的区块数据最终形成链式结构。默克尔根哈希占用内存大小为 32 字节，它是区块体中所包含交易按照 Merkle 树的形式生成的散列值。随机数占用内存大小为 4 字节，它是矿工完成工作量证明时所计算出的数字，网络中的其他节点在收到该区块后需要通过该随机值对本区块的有效性进行验证。

区块体中包含了该区块所打包的交易数据[51]，然后区块中的交易按照 Merkle 树的形式生成一个默克尔根哈希保存在区块链头中。Merkle 树是一颗完美二叉树，形成默克尔根哈希的过程如图 2.1 中区块体中所示。每个叶子节点中所存储的都是区块链中交易的哈希值，然后把相邻节点的哈希值通过再次进行哈希运算，最后得到了一个默克尔根哈希，并把其保存在区块头中[52]。采用这种方式存储有二大优势：1）区块头中不再对各种交易信息进行存储，只存储交易最后产生的默克尔根哈希，对于网络中的轻节点可以选择只对区块头进行同步，这将有效减少轻节点加入区块链网络时所同步的数据量。2）如果区块中的某一笔交易被恶意节点所修改，则可以通过与正确节点的 Merkle 树从根节点往下对比哈希值的方式快速定位到被修改的那笔交易，不需要对区块体中的其他交易进行获取[53]。

虽然不同版本的区块链其所针对解决问题的场景有所不同，但它们的系统基础架构基本一致，其结构如图 2.2 所示。

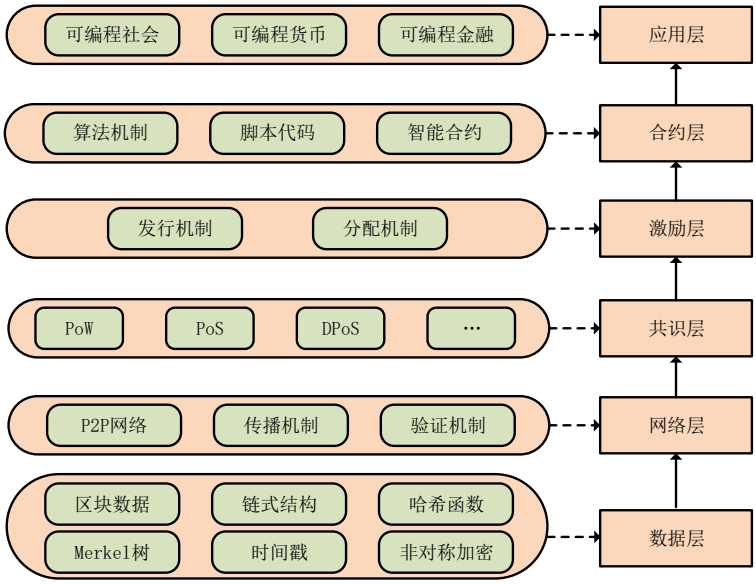

图 2.2 区块链系统基础架构图

其结构从上到下分别为应用层、合约层、激励层、共识层、网络层和数据层[54]。数据层是用来进行区块链数据保存的，每个区块都由区块头和区块体二个部分组成的，其具体结构已在上文中进行阐述。网络层主要包含 P2P 网络架构、传播机制和验证机制，保证了区块链上各个节点数据的同步性，是矿工挖矿成功后把区块同步出去的基础。共识层的主要功能是实现区块链中所有交易的确认，不同的共识算法其所针对的使用场景不同。激励层的主要功能是对发行机制和分配机制的实现，以此保证了数字货币的不断发行和对挖矿用户的奖励。合约层主要完成智能合约的部署与运行，是实现区块链网络可编程化的基础，它为区块链技术可以与实际场景结合提供了可能。应用层主要对区块链的应用场景和案例进行实现。区块链通过这些技术的结合，解决了一直存在的信任构建问题[55]。

2.1.2 共识算法

目前公有区块链中的共识算法主要包含 PoW、PoS、DPoS 等算法。在区块链中产生的交易都会被共识节点进行验证，该交易一旦被验证通过则无法在对其删除或者修改，因此共识算法对区块链网络的稳定性和可靠性来说是重中之重[56]。区块链技术发展的同时，共识算法的种类也在日益增加，不同的共识算法其面对的使用场景也不相同。在本节中，将会介绍区块链中常用的共识算法。

（1）工作量证明（PoW）共识算法

工作量证明算法作为一种适合于货币加密的共识方法已经得到了广泛的应用。在这种共识机制中，网络中的每一个节点都同时通过哈希运算去解决一个“数学难题”，用于对该区块中存储的交易进行确认。矿工会尝试寻找不同的随机数然后对其进行哈希运算使其满足一个预定难度目标值。当一个矿工找到符合挖矿难度的目标值时，它会把产生的区块在网络中进行广播，让其他矿工对该区块的正确性进行检查。如果区块中的交易验证通过，所有矿工都会把该区块记录下来，并在该区块的基础上再次进行下一个区块的验证。

工作量证明算法具有高安全性和去中心化的优势，但是由于区块链中的所有节点都在进行哈希运算，这会造成电力大量浪费的问题。在求解哈希函数的过程中需要大量时间进行计算，这导致了该算法存在吞吐量小、区块创建时间长、特殊的硬件依赖等问题，这些问题也使得它不能够在大型和交易数量快速增长的网络中使用。

（2）权益证明（PoS）共识算法

权益证明算法是对工作量证明算法的一种优化，它是针对工作量证明算法的能源效率低下问题进行改进。权益证明算法的思想是把节点拥有的权益大小作为区块产生的另一种衡量

标准，对每个节点的出块难度进行动态调整。节点所拥有的权益也叫做币龄，它是节点拥有代币的个数与该币所持有时间的乘积。当节点通过币龄对网络中的交易进行打包时，该节点相应的币龄也会被消耗。51%攻击是指当某一节点掌握网络中的 51%的权利时即可对区块的出块产生影响，比如利用分叉进行双花交易。由于 PoS 算法中区块的产生与币龄相关，因此这对以想要产生 51%攻击的节点来说将会产生更大的难度，随着区块链网络运行时间的增加，51%算力攻击发生的可能也几乎为 0。

权益证明算法相较于工作量证明算法具有区块创建速度快和吞吐量高相对较高的特点，但由于区块的产生与节点所拥有数字货币的总量有关，因此在去中心化的程度上，该算法不如工作量证明算法好。权益证明算法虽然引入币龄概念降低了区块产生的难度但仍然需要通过挖矿的方式对区块链中的交易进行打包。

（3）委托权益证明（DPoS）共识算法

委托权益证明算法是由 Dan Larimer 在 2014 年提出的，它是对权益证明算法的一种改进和优化。该算法首先需要网络中的节点互相投票，并按照投票数把获得投票数最高的前 N 个节点作为记账节点轮流产生区块，其中 N 的大小可以根据不同的业务需求进行调整。记账节点负责对网络中的交易进行打包出块并完成共识，如果记账节点在轮到它进行区块生产时没能正常产生区块或者产生错误区块，则会对其进行跳过让下一个节点进行出块，在下一轮进行投票选举时可以通过投票取消其记账资格。这种共识方式类似于民主大会，记账节点都是由持币人选举出来的。由于记账节点数是一个固定大小的值，因此在系统吞吐量上 DPoS 共识算法远远高于 PoW 共识算法和 PoS 共识算法，更符合供应链溯源的需求场景，但该共识算法存在记账节点的选举受拥有数字货币数量多的节点影响较大、区块验证时间长和对被选为记账节点的恶意节点不能及时处理的问题。因此，本文第三章将针对该算法中存在的问题进行优化以使其具有更好的实用性。

2.1.3 P2P 网络

P2P 网络常被称为对等网络[57]，它是区块链技术实现的基础。在 P2P 网络中每一个节点即是服务器同时也是客户端，它们之间的地位完全平等具有相同的权利，并且也执行相同的任务。P2P 网络中没有中央服务器进行统一管理，因此每个节点都在本地存储着网络中的所有数据。由于每个节点都会对网络中的数据进行转发、存储和接收，因此网络的整体性能会随着节点数的增加而变得更快更高效[58]。P2P 网络随着自身发展的 4 个阶段可以被分为 4 种网络模型如图 2.3 所示。下面分别对其进行简单介绍。

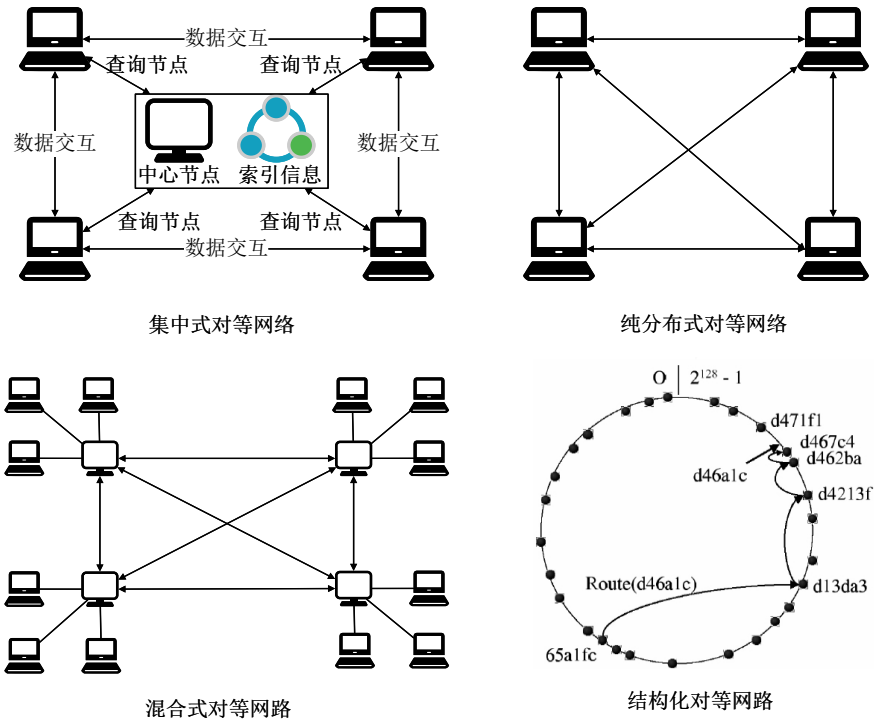

图 2.3 P2P 网络结构

集中式对等网络，这类网络中包含一个中心节点，该节点对网络中所有节点的索引信息进行保存，索引信息主要存储了各节点的 IP 地址及端口等。这类对等网络的优点是拓扑结构简单，可以快速定位到想要找到的节点。纯分布式对等网络，这类网络移除了集中式对等网络中的中心节点，各节点之间通过随机构建的方式形成拓扑结构。当有新节点加入时，它会随机选择一个网络中的节点建立关联关系，新节点可以通过网络中所有节点的 IP 地址列表与其他节点进行通信同时该节点加入网络的消息也会被邻居节点向全网进行广播。混合式对等网络，这类网络是前面二类对等网络的融合，主节点之间形成分布式对等网络，主节点与多个普通节点相邻组成集中式对等网络。混合式对等网络是一种效率更高且更灵活的组网方式，但其去中心化的程度相对较低。结构化对等网络，这类网络也是采用了分布式网络结构，但其在应用层的网络节点被组织成一种特定的有序结构，并且保证任何节点都可以对网络中的所有资源进行获取。

2.2 密码学基础

2.2.1 哈希算法

哈希算法贯穿着区块链中的方方面面，比特币中账户的交易地址、Merkle 树的形成，挖矿算法的难度计算等都是通过该算法进行实现的。哈希算法又称为消息摘要算法[59]，它是密

码学算法中的一种。该算法无需借助任何密钥即可将任意长度的数据映射成一个固定长度的二进制值，这个固定长度的二进制值通常称为哈希值。哈希算法具有单向不可逆的特性，即对于任意值不管进行多少次哈希运算都会得到相同的结果，但是不能通过哈希结果反推原始数据的值，如果输入值产生微小的变化其得到的结果也是截然不同的，因此可以利用此特性保证数据的完整性和防伪造性。

密码学中常用的哈希算法有 SHA-1、SHA-2 和 MD5 等。由于 MD5、SHA-1 已经被证明是一种不安全的算法[60]，因而区块链中使用了更可靠的 SHA-256 算法，该算法是 SHA-2 散列算法的“家族”中的一种特定算法。SHA-256 加密算法的加密过程如图 2.4 所示。

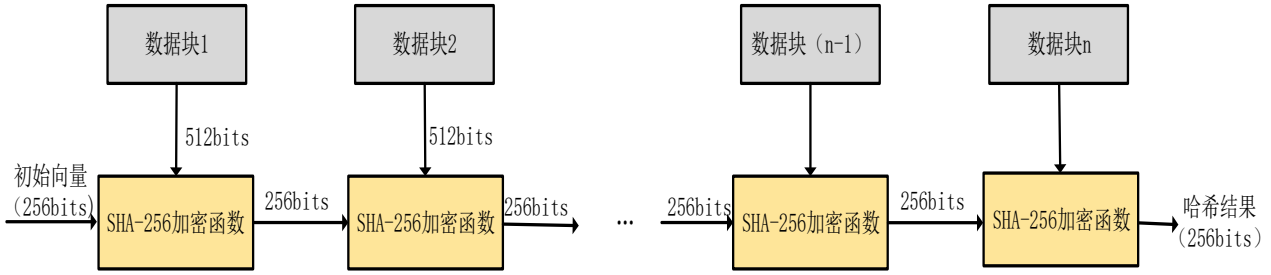

图 2.4 SHA256 加密算法执行流程

其加密过程可以分为二步：明文数据的预处理和迭代加密。在明文数据的预处理阶段，SHA256 加密算法是对明文数据的长度进行填充，保证明文数据可以被分解为 n 个 512 比特大小的数据块。迭代加密阶段是对所分解的 n 个数据块进行循环加密，上一个数据块的输出会被作为下一个数据块的输入进行计算。通过 n 次计算，最后得到的 256 比特的数据就是最终的加密结果。

2.2.2 非对称加密算法

非对称加密算法是相较于对称加密算法来说的，如果加密和解密时使用的是同一把密钥则为对称加密算法，反之则为非对称加密算法。在非对称加密算法中密钥包含两部分：私钥和公钥。私钥需要用户自身进行管理，它是用户身份的唯一证明。公钥则可以对任何人公布，其他用户可以使用公钥对两者所传递的消息进行加密或解密。虽然非对称加密算法的加密和解密速度不如对称加密算法快，但是在针对区块链中的复杂场景，非对称加密算法可以提高交易系统的安全性和完整性。在区块链中，一个账户对外暴露的公钥是由私钥经过椭圆加密算法生成的，账户使用的交易地址则是由公钥通过哈希算法得到的，它们之间是一种单向性关系，即无法从被推出的一方反推回上一方。

数字签名是指通过密码学技术运算产生各种符号及代码形成一个密码对消息进行签名，它只能由消息发送者生成，任何人不能对其进行伪造，它是对消息发送者身份的一种可靠保

障[61]。在区块链中通过该技术可以确保交易产生方的身份，当网络中的节点发起一笔交易时，该节点会对这笔交易用自身的账户私钥进行签名，当网络中参与共识的节点收到这笔交易时会通过交易发起方的公钥对该交易进行验证，如果验证失败，则不对该交易进行处理。由于每个用户的私钥都是由自身进行保管的，通过这种技术手段可以对网络中的资金安全产生有力的保障。其具体的实现过程如图 2.5 所示。

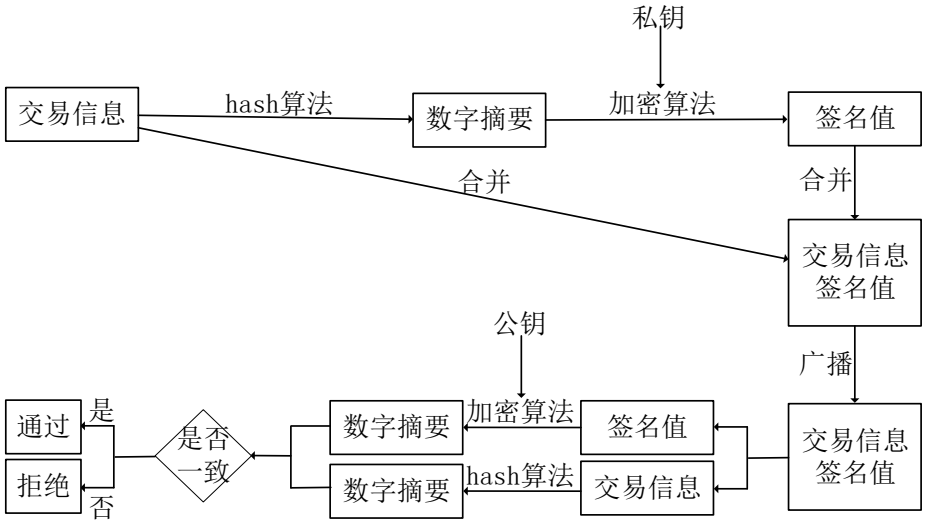

图 2.5 签名与验证过程

对于交易发送方，首先对交易信息进行哈希运算，生成数字摘要。然后交易发送方通过自身保存的私钥对数字摘要进行签名，生成签名值。最后把交易信息和签名值打包后发送给消息接收方。

对于消息接收方，首先通过交易发送方的公钥对签名值进行验证，即可判断用户身份是否正确。然后消息接收方对交易发送方发送的明文消息进行哈希运算。最后通过把签名值中的哈希值与交易信息哈希运算后的哈希值进行对比即可判断该交易是否是交易发送方所产生的原始交易数据，保证区块链中交易的安全性和真实性。

2.2.3 SM2 加密算法

SM2 加密算法是由国家密码管理局在 2010 年年底发布的，它是在国际标准下的 ECC 椭圆曲线密码算法的基础上优化改进而得来的，是非对称加密算法的一种。在 SM2 加密算法没有出现之前比较常用的非对称加密算法是 RSA 加密算法，从各个方面来说，我国提出的 SM2 加密算法都要比 RSA 加密算法要好。SM2 加密算法相对 RSA 加密算法来说密钥长度更短，但其安全性更高，而且在处理速度也有所提高[62]。

SM2 加密算法的签名过程可以分为三个过程：密钥对生成、签名生成和签名验证阶段。

（1）密钥对生成的流程如图 2.6 所示。

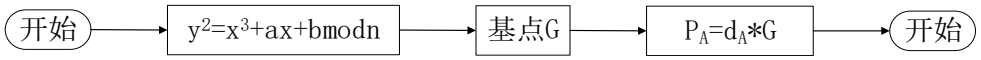

图 2.6 密钥对生成流程

$y^2 = x^3 + ax + bmodn$为椭圆曲线，G 是椭圆曲线上的基点，n 为基点 G 的打点次数。d_A表示私钥，它的取值范围为 1 到 n-1。P_A表示所生成的公钥，由 P_A 和 G 是不能够推出私钥 d_A 的。

（2）签名值生成流程如图 2.7 所示。

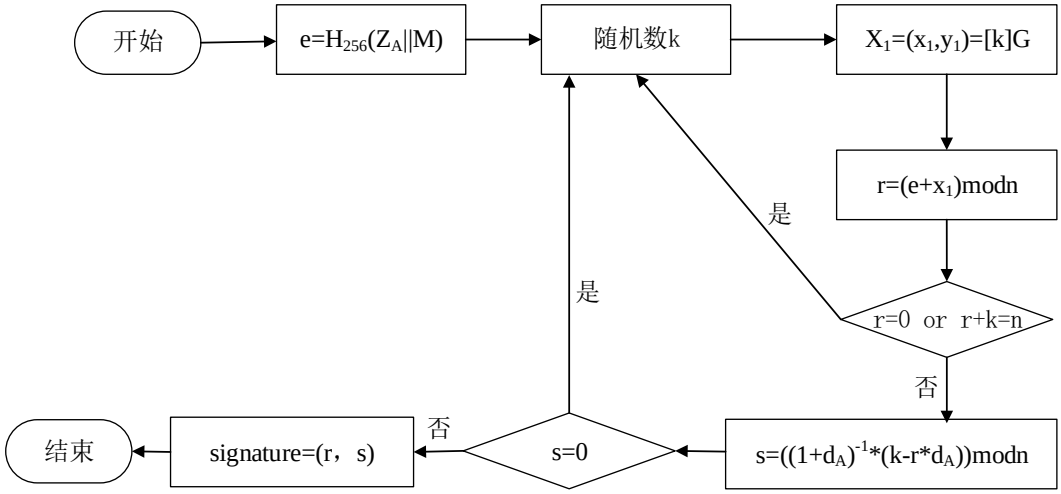

图 2.7 签名值生成流程

M表示需要进行签名的明文信息。Z_A表示杂凑值它是按照公式 2.1 进行定义的。

$$Z_A = H_{256}(IDL_A||ID_A||a||b||x_G||y_G||x_A||y_A) \tag{2.1}$$

式中ID_A是对用户进行区分的标识，IDL_A是ID_A的长度。a和b是椭圆曲线中的系数。x_G和y_G是基点G的坐标值。x_A和y_A是公钥的坐标值。

签名值生成的具体步骤如下：

1）通过杂凑值与明文信息进行哈希运算得到哈希摘要 e。

2）生成随机数k，且$k \in [1, n-1]$并和基点G进行运算得到椭圆上的点x_1。

1）根据x_1和e计算签名参数r。如果r不符合要求则重新选取随机数进行计算。

2）根据私钥等值计算签名参数s，如果s不符合要求则重新选取随机数进行计算。

3）最后输出签名值$signature = (r, s)$，签名流程结束。

（3）签名验证流程如图 2.8 所示：

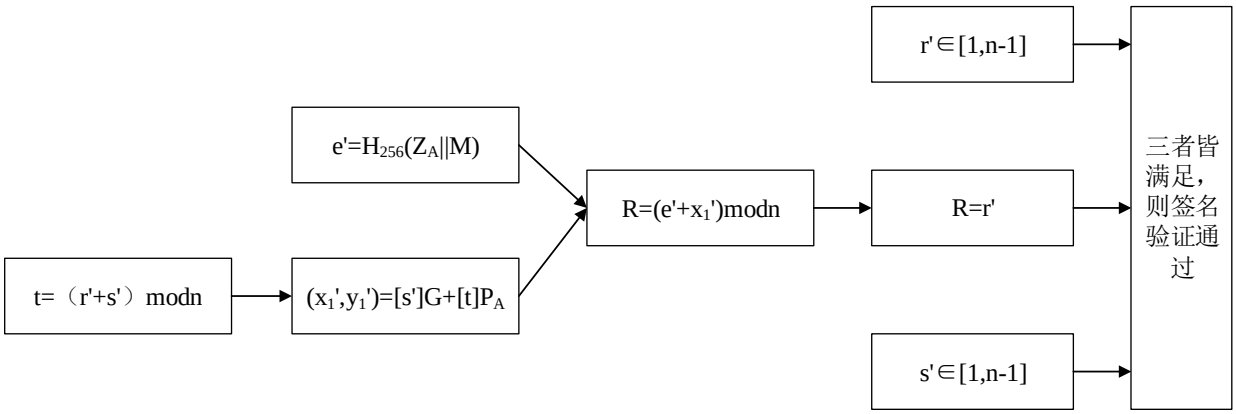

图 2.8 签名值验证流程

1）将签名值(r,s)转化为二个数r'和s'，并对其进行验证如果r'或s'任意一个数小于 1 或者大于 n-1 则验证直接失败直接返回。

2）对明文数据 M 按照签名流程的方法与杂凑值生成e'。

3）将r'和s'转化成整数，并生成t如果t为 0 则直接验证失败。

4）计算椭圆曲线上的点x_1'。

5）通过e'和x_1'计算R把它与r'行比较，如果相等则标签该签名正确，不相等则验证失败。

2.3 以太坊和智能合约

2.3.1 以太坊

以太坊[63]是由 Vitalik Buterin 受比特币的影响在 2013 年提出，它是公有链的一种。以太坊网络中的账户可以分为两类：用户账户和智能合约账户。用户账户是指网络中的节点在加入区块链时为其创建的账户，这个账户是由一对公钥和私钥所控制的。智能合约账户是指存储在区块链中的一段处理逻辑代码的地址。这两类账户的区别是：用户账户中不包含代码，它是网络中各节点与其他节点进行交易的凭证。智能合约账户不能主动向其他账户发起交易，它只允许被用户账户调用，但在被调用时可以向其他账户发起内部交易请求。

相比于比特币区块链来说，以太坊区块链提供了大量外部接口，用户可以通过编写智能合约的方式对其在区块链中保存的数据格式进行定义。以太坊去中心化的实现离不开以太坊虚拟机(EVM)，它是以太坊中智能合约部署和调用的基础，被称为以太坊的“心脏”。EVM 的具体实现如图 2.9 所示，堆栈位宽为 256 比特最大深度为 1024，内存是变量存储的区域，程序计数器保证了代码的有序执行，可用 Gas 是油费池，当油费用完后如果交易还没完成则交易产生失败。当用户对网络中与已经部署的合约进行数据交互时，EVM 将会调用相应的指令去执行该请求，并对数据结果进行更新。

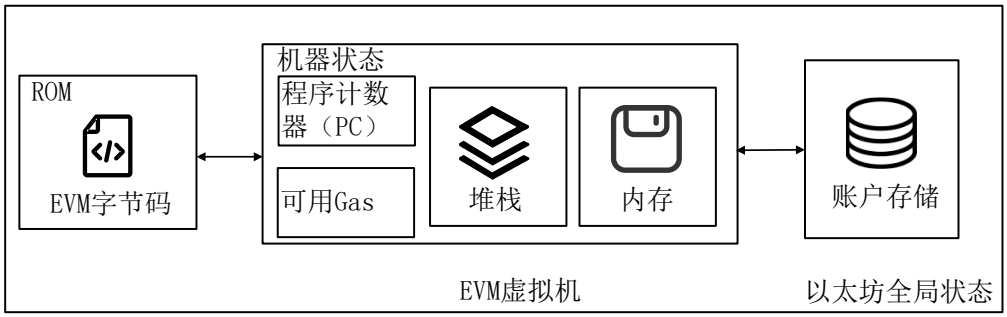

图 2.9 以太坊虚拟机结构图

以太坊中引入了一个比较特殊的定义：Gas，无论是用户账户之间的转账交易还是用户账户对网络中的合约进行调用都会产生一定的 Gas 消耗，它是为了对用户所提交请求的工作量大小的一种限制，这笔消费也会随着交易的完成奖励给对其进行打包上链的矿工。在 Gas 中包含二个概念需要了解，首先是 GasLimit，它表示这笔交易所允许消耗 Gas 值的最大数量，其次是 GasPrice，它表示一个 Gas 的单价，如果交易用户给予他发送请求的 GasPrice 越高，则矿工将优先对其进行打包，这笔交易所花费的总交易值 price = GasLimit * GasPrice。如果用户发出的交易执行完成，但其设定的 Gas 没有全部消耗完，则剩余的 Gas 会返还给交易用户。但是当这些 Gas 全部用完，交易仍没有完成时，系统将会返回给用户一个错误提示，并且相应的 Gas 值也不再退还，这避免了一些合约中出现死循环或者恶意用户输入无效请求等问题而浪费大量算力去计算无效交易的问题。

2.3.2 智能合约

智能合约这一概念是由尼克•萨博（Nick Szabo）首次提出的[64]，他给予智能合约如下定义：“一个智能合约是一套以数字形式定义的承诺，包括合约参与方可以在上面执行这些承诺的协议。”但在当时由于缺少完全可信的网络环境，因此这一概念始终也只是停留在文字上并没有被真正的应用到实际生产中。区块链的出现为智能合约的实现提供了可能性[65]，在以太坊区块链中就将该概念与其自身技术相结合，通过图灵完备的 Solidity 语言可以完成智能合约的构建。智能合约可以对多方用户的交易进行自动化执行不需要任何第三方的介入，而且当智能合约部署到区块链后，任何人都不能再对其进行修改，因此在进行合约发布前需要对所编写的合约逻辑进行全方面的测试。以太坊网络本身具有很好的健壮性，但所部署的合约却未必。“TheDao”事件就是因为合约漏洞而使得用户造成了大量的经济损失，以太坊网络也因为此事件产生了分叉。智能合约程序的运行流程如图 2.10 所示。

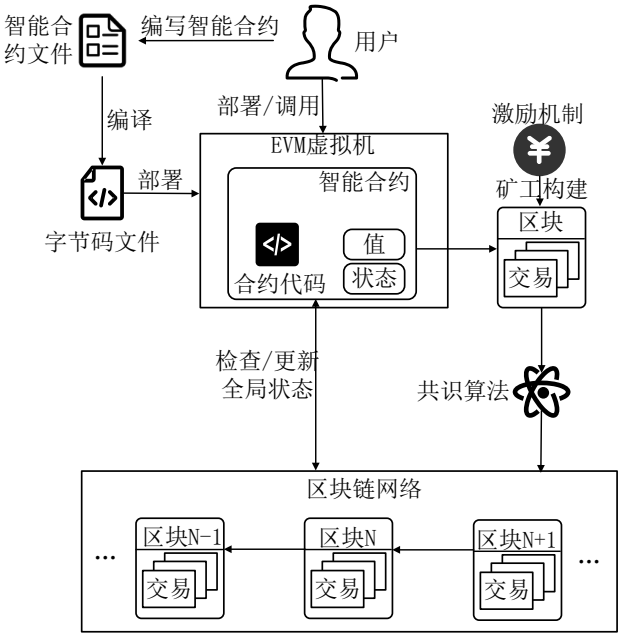

图 2.10 智能合约运行机制

首先需要根据具体的业务需求对智能合约进行代码开发生成 Solidity 文件，然后需要把合约编译生成 EVM 虚拟机能够运行的字节码，最后把其部署到 EVM 上即可通过合约提供的接口对其进行操作。如果想要通过其他编程语言对所实现的智能合约进行调用，可以通过以太坊提供的 Web3j 开发库采用 RPC 调用的方式对智能合约进行操作。

2.4 本章小节

本章分为三个部分对基础概念进行介绍。第一部分是区块链的基本概念与相关技术，该小节对区块链定义与结构和区块链中使用到的共识算法与点对点网络进行说明。第二部分是对密码学基础进行详细阐述，其中包括哈希算法、非对称加密算法和 SM2 加密算法三部分。第三部分是以太坊区块链的基本概念以及智能合约的实现方式和运行机制进行介绍。通过对这三部分基础知识进行总结，为下文的技术实现做好基础铺垫。

第三章 基于工作量证明和信誉值的 DPoS 共识算法研究

针对委托权益证明（DPoS）共识算法中存在记账节点的选举受拥有数字货币数量多的节点影响较大、区块验证时间长和对被选为记账节点的恶意节点不能及时处理的问题，在结合工作量证明和委托权益证明各自优点的基础上提出一种基于工作量证明和信誉值的委托权益证明（Delegated Proof of Stake base on Proof of Work and Reputation, DPoSPR）共识算法。本章节重点介绍了该算法的实现方案和执行流程，最后对其进行性能分析。

3.1 DPoS 共识算法

3.1.1 DPoS 共识算法原理

共识算法是区块链技术在去中心化条件下保证各参与节点数据一致的关键因素，区块链中节点所产生的每一笔交易都需要网络中参与共识的节点达成一致后才能被打包进区块中。在对委托权益证明共识算法进行分析之前，首先对该共识算法中所涉及的概念进行阐述。委托权益证明共识算法的共识流程可以分为两个阶段，其共识流程如图 3.1 所示。

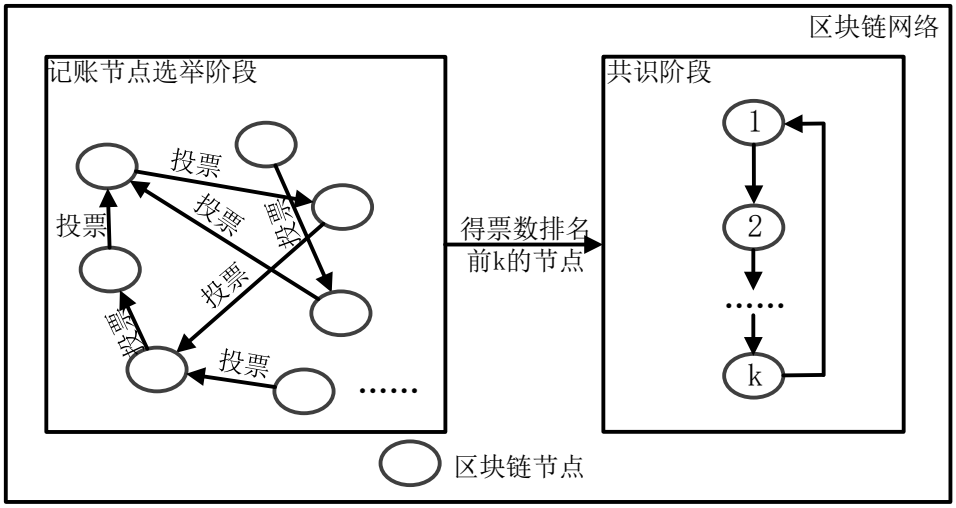

图 3.1 委托权益证明共识流程

第一个阶段为记账节点选举阶段：记账节点是由区块链中的所有节点每隔一段时间进行互相投票选举产生的（选举间隔由区块链自行决定），其中投票由节点向区块链中发送一笔交易来实现，这笔交易中包含投票者的身份信息、被投票者的身份信息和所投票数，该交易会通过 P2P 网络发送给区块链中参与共识的节点最后打包进区块中。区块链中的每个节点都可以参与到投票和被选为记账节点的过程中，且每个节点所拥有的投票数与其自身拥有的数字货币数量相关。记账节点的个数是由不同的区块链决定的，如果区块链中需要 k 个记账节点，

则投票结束后得票数排在前 k 名的节点被认定为记账节点集合，它们所拥有产生区块的权利是完全等价的。记账节点需要交纳一些数字货币当作保证金，以保证其能在担任记账节点的周期内认真履行自己的职责。本轮投票结束到下一轮投票结束的这段时间称之为共识周期。

第二个阶段为共识阶段：当记账节点选举完成后，区块链就进入到了共识阶段。在委托权益证明共识算法中，区块的生产是由记账节点集合中的节点轮流对区块链中的交易进行打包完成的，记账节点在其区块生产的这段时间称之为生产周期。当记账节点在其生产周期完成出块后需交由其后续其他记账节点进行区块验证，当验证通过的记账节点数超过记账节点总数的 2/3 时该区块被认定为不可逆的状态即该区块中所包含的交易生效，生成该区块的记账节点将会获取一定数字货币奖励。如果记账节点在其生产周期因为各种原因不能产生正常区块或者产生错误区块，系统则会跳过该区块的当前生产周期交由下一个记账节点完成出块任务，并没收该记账节点的保证金。

3.1.2 DPoS 共识算法问题分析

委托权益证明共识算法具有吞吐量高、能耗低、网络带宽要求低等优点，但其也存在以下问题：

1）在委托权益证明共识算法中，区块链中每个节点所拥有的投票数与其自身持有的数字货币数目成正比，这将造成拥有数字货币数量多的节点会对记账节点的选举产生较大影响的问题，使选举制度的公平性遭到破坏，弱化了区块链网络的去中心化程度。

2）在选举出的记账节点集合中存在不诚实的节点时，委托权益证明共识算法只是跳过该节点本轮的生产周期并对其保证金进行没收，并没有对其进行替换或惩罚。该不诚实节点仍然可以参与当前共识周期后续的区块生产且被选举为记账节点的权利没有受到影响。

3）当记账节点产生区块后，该区块需要其后续超过记账节点总数的 2/3 个记账节点在其生产周期验证通过后才能把该区块确认为不可逆的状态，这会增加区块链网络中的交易时延，且更容易产生区块分叉（同一时刻产生多个区块将会出现区块分叉现象），进而影响区块链的整体性能。

为解决上述问题使委托权益证明在区块链中发挥更好的作用，本文提出一种基于工作量证明和信誉值的委托权益证明（DPoSPR）共识算法，其具体实现方案将在下文进行详细阐述。

3.2 DPoSPR 共识算法的设计与实现

3.2.1 节点分类

在 DPoSPR 共识算法中将区块链中的节点分为七类，分别为普通节点、候选节点、共识节点、非共识节点、记账节点、备选节点和验证节点。

普通节点集合 N_o：普通节点不参与区块链中的共识阶段，其进出区块链网络时不需要任何的授权。用户可以在使用系统提供的服务时看到整个网络的交易状况，虽然不能参与投票与竞选，但可以对消息和区块进行转发。普通节点可以通过身份认证成为候选节点，但每个身份 id 只能认证一次。

候选节点集合 N_c：候选节点可以在区块链网络中发出工作量证明请求后，工作量证明的定义如 2.1.2 小节所述，对该请求进行验算，验算成功后则把其验算结果进行签名并广播到区块链网络中。

共识节点集合 N_d：共识节点是由候选节点通过工作量证明的方法选举出来的。假设系统中需要产生区块的节点数为 k，那么需要在候选节点中选取前 2k 个完成工作量证明的候选节点，把其定义为共识节点。

非共识节点集合 N_r：非共识节点是指候选节点集合中未能成为共识节点的节点。

记账节点集合 N_w 和备选节点集合 N_a：这两类节点是由非共识节点对共识节点进行投票选举出来的，但在投票选举时不再仅仅只考虑共识节点获得的票数，同时也会把共识节点的信誉值当作选举结果考量的一部分，信誉值的计算方法将在 3.2.3 小节进行展示。共识节点中排名前 k 的节点被选为记账节点，k+1 到 2k 的节点被选为备选节点。记账节点需要对网络中的交易进行验证与打包，并把验证成功的交易数据生成区块。备选节点的职责是当记账节点因作恶或者其他原因不能正常出块时对记账节点进行更换，其具体实现方法将在 3.2.4 小节中进行展示。

验证节点集合 N_v：验证节点是非共识节点互相投票选举出来的，这类节点的数量与记账节点相同，它负责对记账节点所产生的区块进行验证。验证节点的选举方法和原始算法中的记账节点选举阶段类似，只是在选举过程中节点的信誉值也需要被作为选举结果考量的一部分。由于区块链中的非共识节点都可以参与选举，拥有数字货币数量多的节点出于对自己利益的保护会把其手中的票数投给其信任的节点，因此其具有较高的可靠性。当验证节点正确完成验证后也会给与其少量的数字货币奖励。

区块链中各类型节点类型转化图如图 3.2 所示。

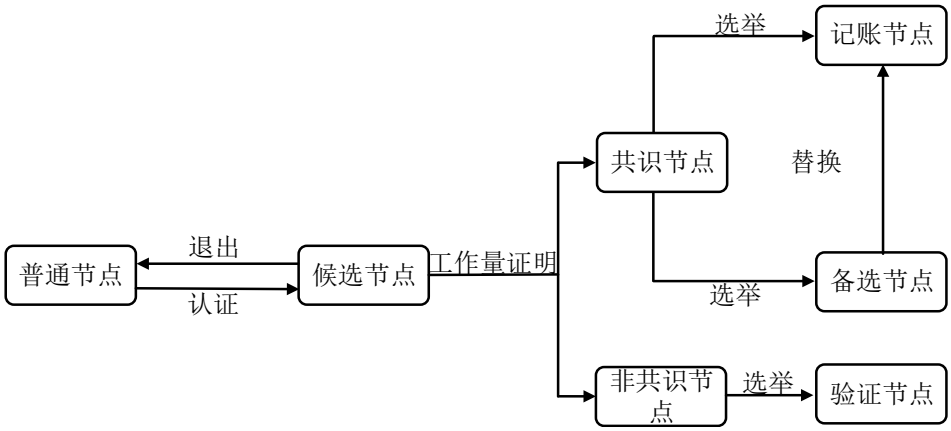

图 3.2 节点类型转化图

3.2.2 共识节点选举阶段

共识节点选举是由工作量证明的方式实现的，选择该方案的原因有二个：首先，由于 DPoS 共识算法在记账节点选举中受拥有数字货币数量多的节点影响较大，工作量证明方法具有更高的去中心化特性，因此引入该方法可以保证更多候选节点有成为共识节点的可能性。其次，工作量证明是对节点计算性能的一种体现，通过该方式可以选择出计算性能相对较好的节点，记账节点的性能对区块链整体性能会产生较大的影响。

工作量证明的原理如图 3.3 所示。区块头的结构已经在 2.1.1 小节中进行详细介绍，此处不再阐述。

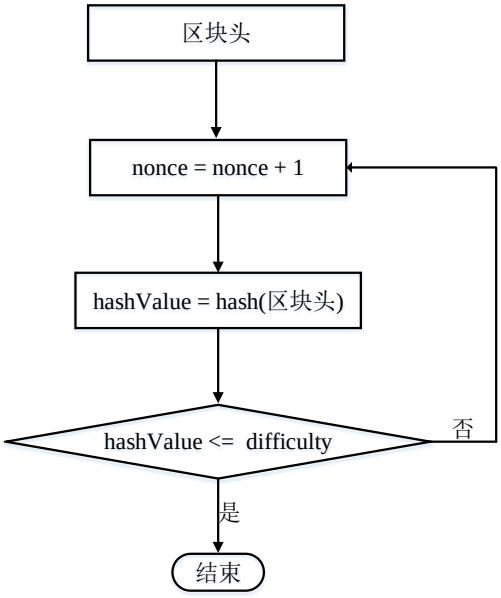

图 3.3 工作量证明原理

工作量证明的过程是通过更新区块头中随机数的值进行实现的。本方案引入工作量证明是为了选取共识节点，因此需要对区块信息的部分值进行调整。首先是默克尔哈希根的值，因为该区块并不是真的对网络中的交易进行打包，因而该值设置为 0。其次是父区块哈希值，

为避免候选节点对区块信息中的父区块哈希进行预测干扰随机选择的结果，因此把其设置为工作量证明请求发出时刻当前最新区块的哈希值和一个随机数产生的哈希结果。最后是在难度目标值的选取上，通过 Java 语言对工作量证明方法进行实现并测试后，认为本次实验选择难度值在 6 时，平均验证时间为 15 秒即可满足实验要求，在实际使用中难度的具体值可以根据不同情况进行设置或者将其根据节点数等其他值定义为动态变化的值。

共识节点选举流程如下：

第一步：区块链网络广播工作量证明请求，请求中包含候选节点所验证区块头中的父区块哈希值，该值为当前时刻最新区块的哈希值与一个随机数哈希后的结果。

第二步：网络中的候选节点收到工作量证明请求后按照上述工作量证明原理对随机值进行验证，当验证出符合难度值的随机数后候选节点发送验证成功消息到区块链中，该消息中包含验证成功区块头，验证成功节点会对该信息进行签名。

第三步：当系统收到 2k 条验证成功消息并验证成功后向网络中其它节点广播停止验证消息，该消息中包含本轮 2k 共识节点的地址信息，网络中其他候选节点收到该消息后便停止本轮工作量证明，本轮共识节点选举结束。

在共识节点选举阶段，其具体实现方法如表 3.1 所示。

表 3.1 共识节点选举算法

Input:记账节点数目 k

Output:共识节点集合N_d

```
broadcast<工作量证明请求>
num = 0 // 已选验证有效节点个数统计
N_d // 共识节点集合
while(num < 2k)
    wait() // 等待候选节点发送验证
     if(verifyBlock(验证N_i成功消息) == true)
          N_d.add(N_i)
          num++
    else
          continue
    end if
end while
broadcast<停止验证消息>
return  N_d
```

3.2.3 信誉机制与投票选举阶段

在 DPoS 共识算法中，恶意节点在其生产周期产生无效区块或者不能按时产生区块时只是对其本轮所缴纳的保证金进行没收，该恶意节点被选举为记账节点的权利没有受到影响。因此本小节提出信誉机制，使用信誉值来对每个网络节点的可信度进行定量的描述。通过每个节点的信誉值，可以对网络节点的行为可靠度进行评估，减少恶意节点被选中为记账节点和验证节点的概率，提升整个网络的安全性。另一方面，信誉值也可以作为各节点遵守区块链规定的一种约束。下文信誉值的量化标准是根据本次实验设定出的合理值，该值的大小可以根据具体系统运行环境自行改变。

候选节点的初始信誉值为 60，最大值为 100。在投票选举阶段，该信誉值与得票数一样都是决定该节点是否能够成为记账节点或者验证节点的重要组成部分。信誉值考虑记账节点和验证节点在出块和验证方面的因素：

1) 记账节点生成有效区块数C，该值是由记账节点在一个共识周期内生成的有效区块数决定的，其具体计算方法如公式 3.1 所示。

$$C = \sum_{t=1}^{n} \frac{actualCount}{theoryCount} c_t \tag{3.1}$$

在上式中，c_t表示记账节点在第 t 个生产周期是否正常出块，如果正常出块则c_t的值为 0.2，如果不能出块或者产生区块有问题则c_t的值为 0。$theoryCount$表示理论上一个生产周期区块中包含的交易数量，$actualCount$表示区块中实际包含的交易数量。

2) 验证节点验证区块数V，该值是由验证节点在一个共识周期内有效验证的区块数决定的，其具体的计算方法如公式 3.2 所示。

$$V = \sum_{t=1}^{n} \frac{1}{2^{\frac{time}{T}}} v_t \tag{3.2}$$

在上式中，v_t表示当前验证节点第 t 个生产周期是否对区块进行验证，如果当前周期已经向记账节点发送验证结果则v_t=0.02，如果本周期没有向记账节点发送验证结果则v_t=0。$time$表示验证完成所消耗的实际时间，T表示区块验证周期（验证节点完成区块验证的理论值）。

3）惩罚系数P，该值是系统对记账节点和验证节点的一种行为约束，如果在其验证周期或生产周期该节点不能正常工作则会对其信誉值进行扣除。其具体计算方法如公式 3.3 所示。

$$P = \sum_{t=1}^{n} p_t \tag{3.3}$$

对于记账节点集合，如果记账节点在该生产周期内无法正常出块或者出块被验证为无效，

则需要把其从记账节点集合中踢出且其所获得的信誉值C将会被清零，并从备选节点中选择一个节点替换当前记账节点。惩罚系数P分别根据不同行为进行递减，如果出块被记账节点验证为无效则p_t= -10，在生产周期内无法正常出块则p_t= -8。

对于验证节点集合，如果验证节点在验证周期验证错误或者没有验证区块则对其V值进行减半处理。惩罚系数P分别根据不同行为进行递减其，如果验证节点认为该区块存在问题但区块最后验证通过或者验证节点认为该区块不存在问题但区块没有通过验证，在该验证周期都会对其惩罚因子进行累加此时p_t=-3，如果在验证周期内未能进行验证则p_t=-2。

在一轮共识周期结束后，将会对记账节点集合和验证节点集合的信誉值进行更新，更新公式如公式 3.4 所示。

$$trustValue = oldTrustValue + C + V + P \tag{3.4}$$

在上式中$oldTrustValue$表示当前节点在本轮共识周期前的信誉值。$trustValue$表示本轮共识周期结束后的信誉值，如果更新后的节点信誉值小于 40，则该节点将被踢出候选节点，且对该节点身份 id 进行封禁，不再能够担任候选节点。当共识进行 50 轮后，会对系统中候选节点信誉值大于 60 的节点进行信誉值扣除，其信誉值大于 60 的部分将会减少为原来的一半，避免因引入信誉值导致记账节点产生集中化的现象。

在 DPoS 共识算法中，记账节点的选举是由投票实现的。但由于每个节点的投票数与其手中持有的数字货币数量相关，这会造成拥有大量数字货币的节点对记账节点的选举产生较大影响较的问题。因此在 DPoSPR 共识算法的投票选举阶段中每个非共识节点可以进行两次投票。第一次投票是选举记账节点，为了保证选举结果具有随机性，在投票选举阶段各非共识节点只能把自身拥有的票数投给完成工作量证明的共识节点。第二次投票是选举验证节点，验证节点是非共识节点互相投票选举出来的，这保证了验证节点具有较高的可靠性。投票选举阶段也不再只对节点获取的票数作为唯一参考标准，而是把上文所提的信誉值作为另一个角度对候选节点进行评估。当节点的信誉值较高时，其被选为记账节点或验证节点所需的票数小于信誉值较低的节点。节点最终得票数计算公式 3.5 如所示。

$$finalValue = \alpha voteValue + \beta voteValue \frac{trustValue}{100} \tag{3.5}$$

在上式中$voteValue$表示当前节点获得的投票得票数。$finalValue$表示当前节点的最终得票数。为防止区块链因运行时间较长或较短时导致信誉值的大小对选票结果影响过小或过大的问题，把其按照百分比的形式对共识节点得票数进行加权。α和β表示获得投票得票数和信誉值的权重，且$\alpha + \beta = 1$。它可以根据区块链对记账节点和验证节点的要求所设定，如果倾向于所选节点为投票得票数高的节点，则$\alpha > \beta$，如果更信任诚信度较高的节点，则$\beta > \alpha$，

如果希望二者等比例考虑，则$\alpha = \beta$。本次实验将这两个值均设置为 0.5。具体的投票选举算法表 3.2 所示。

表 3.2 投票选举算法

Output:记账节点集合N_w, 备选节点集合N_a, 验证节点集合N_v

```
broadcast<投票选举请求>
wait()     // 等待投票结束
N_w    // 记账节点集合
N_a    // 备选节点集合
N_v    // 验证节点集合
for N_i in N_c
        voteValue = getVoteValue(N_i)
        trustValue = getTrustValue(N_i)
        N_i.finalValue = α * voteValue + β * voteValue * trustValue / 100
end for
sortByfinalValue(N_d) // 根据最终得票数对共识节点进行排序
sortByfinalValue(N_r) // 根据最终得票数对非共识节点进行排序
N_w = N_d(d ∈ [0,k - 1])
N_a = N_d(d ∈ [k,2k - 1])
N_v = N_r(r ∈ [0,k - 1])
return N_w,N_a,N_v
```

3.2.4 共识阶段与记账节点更换机制

在投票选举阶段完成后，区块链进入共识阶段。在原始的 DPoS 共识算法中，记账节点新生成的区块会在后续其他记账节点的生产周期进行验证，当该区块被其后续超过记账节点总数的 2/3 个记账节点在其生产周期验证通过才能把该区块确认为不可逆的状态，这会增加区块链网络中的交易时延，影响区块链的整体性能。针对这一问题，采用验证节点对记账节点产生区块进行验证的方法，在保证交易可信的前提下减少交易的确认时间。DPoSPR 共识算法的共识阶段流程如图 3.4 所示。

1）记账节点集合轮流对区块链中的交易进行打包，当记账节点收到交易请求后对其签名和交易地址的合法性等信息进行校验，如果校验成功则把其加入当前区块的交易体中，如果校验失败则直接拒绝其交易请求。

2）当前记账节点生产周期结束时，记账节点把其生成的区块进行签名后发送给验证节点集合中的所有节点进行验证。如果在生产周期内未能正常产生区块，系统会从备选节点中选出一个节点对当前记账节点进行更换，并将记账节点替换消息进行广播，广播消息中包含新

的记账节点信息。

3）当验证节点收到区块验证请求后验证区块签名和交易体内的每笔交易是否正确，验证通过则向记账节点发送验证通过消息。如果发现该区块中有交易是错误的则向记账节点发送验证失败消息。

4）当记账节点收到来自验证节点的$\frac{2}{3}k+1$个验证通过请求或者在验证时延（验证时延由区块验证时延和区块传播时延构成）结束时验证通过的消息不少于$\frac{1}{2}k+1$，则认为该区块有效。记账节点会把该区块对区块链中的所有节点进行广播，广播消息中包含当前区块和验证节点对当前区块的验证结果。如果记账节点收到$\frac{2}{3}k+1$个拒绝请求或者在验证时延结束时验证通过的消息小于$\frac{1}{2}k+1$，则该区块被认定为无效，不再对其进行广播。系统对记账节点进行替换，并将记账节点替换消息进行广播，广播消息中包含新的记账节点信息。当每一轮共识周期结束后，共识节点集合和验证节点集合的信誉值会根据 3.2.3 小节公式 3.4 进行更新。

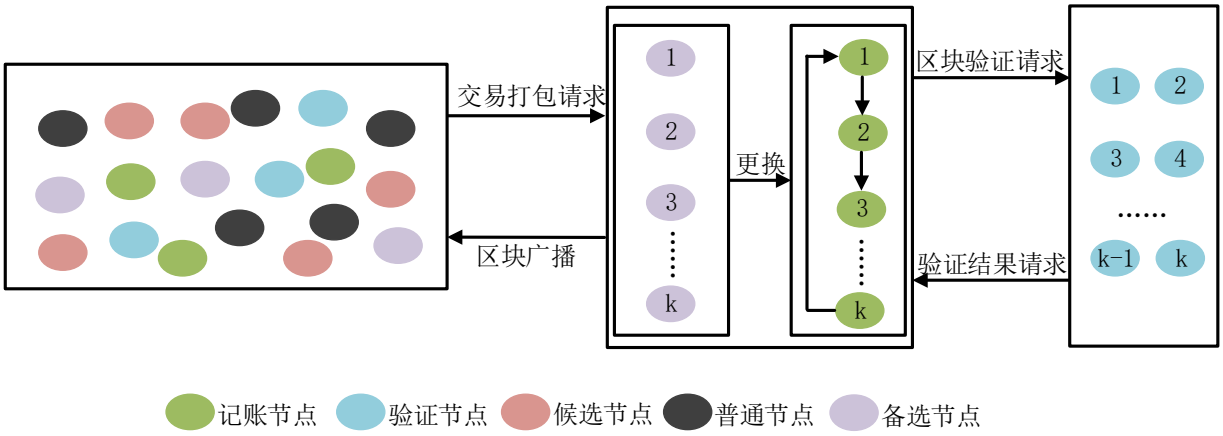

图 3.4 共识流程

在记账节点集合中，如果当前记账节点因为某些原因宕机不能按时产生区块或者产生区块包含不正确的交易时，该区块生产周期将会被浪费。如果该记账节点在下一生产周期仍然不能正常对区块链中的交易进行打包，这会对区块链的整体性能产生较大的影响，因此如果该记账节点在其生产周期内不能按时产生区块或者产生区块包含不正确的交易时，为保证区块链网络的健壮性，本共识方案将会从备选节点集合中按顺序选取节点对该记账节点进行更换，通过这种方式对区块链网络的稳定性及出块效率都有一定的提高。记账节点出块算法如表 3.3 所示。

表 3.3 记账节点出块算法

Input: 备选节点替换位置index, 记账节点位置i

```
valueNum // 区块验证有效票数
invalueNum // 区块验证无效票数
block = generateBlock(N_w^i) // 当前记账节点产生区块
nextGenerator = (i + 1) % k
if(block == null) // 未能产生区块
    broadcast(通知记账节点 nextGenerator 产生区块消息且当前节点未能产生区块)
    启动记账节点更换流程(index,i)
    broadcast(记账节点更换消息)
    return
end if
broadcast(向验证节点发送验证区块消息)
while(true)
    wait() // 等待验证节点验证
    if(收到验证通过消息)
        valueNum++
        if(valueNum >= k * 3 / 2 + 1 || (time > 验证时延 && valueNum >= k * 2 + 1)) // 验证通过
            broadcast(通知记账节点 nextGenerator 产生区块消息和当前产生区块)
            broadcast(广播区块)
            break
        end if
    end if
    if(收到验证未通过消息)
        invalueNum++
        if(invalueNum >= k * 3 / 2 + 1 || (time > 验证时延 && valueNum < k * 2 + 1)) // 验证无效
            broadcast(通知记账节点 nextGenerator 产生区块消息且当前节点产生错误区块)
            启动记账节点更换流程(index,i)
            broadcast(记账节点更换消息)
            break
        end if
    end if
end while
```

3.2.5 DPoSPR 共识算法

基于前四小节中对DPoSPR共识算法的介绍，本节对提出的DPoSPR共识算法进行总结，其整体结构如图 3.5 所示。在 DPoSPR 共识算法主要包含共识节点选举、投票选举、共识和记账节点替换四个阶段。在共识节点选举阶段，由于工作量证明具有较高的去中心化特性，

它的思想是尽可能地让区块链中的每一个节点都享有产生区块的可能性，因此通过引入该方法，让区块链中的候选节点对一道“数学题”进行计算，这不仅平衡了记账节点的分布同时也是对网络中计算性能和网络带宽较好节点的一种筛选。在投票选举阶段，对记账节点和验证节点的选举不再仅仅只对该节点获取的投票数作为唯一的衡量手段，而是引入信誉机制，把该节点的历史信誉度也考虑在内。在共识阶段，为提高记账节点产生区块能够更快地成为不可逆状态减少区块分叉的可能，选择由验证节点对区块进行验证，提升区块链网络中的交易确认速度。当记账节点自身网络出现问题或者产生恶意区块时，就会进入记账节点更换阶段，以保证区块链网络的健壮性。

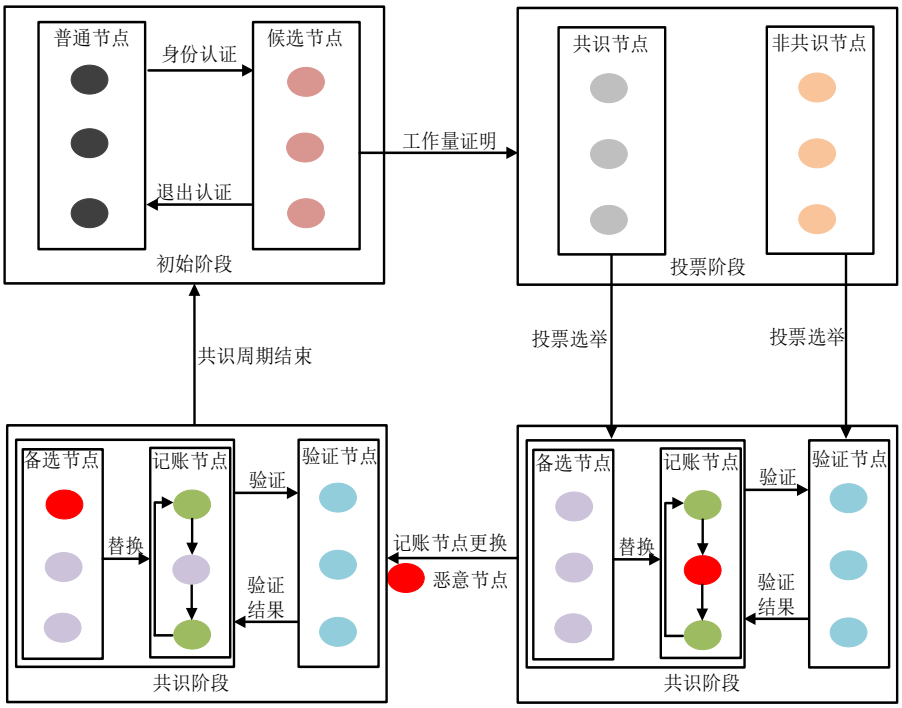

图 3.5 DPoSPR 共识算法整体结构

3.3 实验结果与分析

本小节对上文所提出的基于工作量证明和信誉值的委托权益证明共识算法进行性能评估，分别从记账节点分布率、交易时延、吞吐量三个方面进行测试与分析，验证 DPoSPR 共识算法的实用性。

3.3.1 实验环境

为验证所提共识算法与原始共识算法的性能差异，通过 Java 语言对 DPoS 共识算法进行实现，然后在此基础上进行修改，实现了 DPoSPR 共识算法。在对性能进行测试时，采用 Socket 网络编程的方式对区块链中的节点进行网络通信模拟。通过对比多次试验记录，对试验结果

进行分析。具体使用到的软/硬件环境如表 3.4 所示。

表 3.4 实验环境

软件/硬件	版本
操作系统	Windows10 64位
CPU	AMD R7 4800H
内存	16G
开发工具	IntelliJ IDEA 2018.3
JDK	1.8.0_231

3.3.2 记账节点分布率分析

区块链的重要特征就是其具有去中心化特性，网络中参与记账的节点越多则去中心化程度越高，这不仅能够保证数据记录在不完全可信环境下的安全，也可以让系统中参与节点数量变多，提高区块链中各节点的参与度和系统活跃度。本文按照公式 3.6 对记账节点的分布进行定义。

$$distributionRate = (count_{N^w} / count_{N^c}) * 100\% \tag{3.6}$$

上式中，$distributionRate$表示区块链中参与记账节点的分布率，它是对节点参与记账分布的量化方式。$count_{N^w}$表示区块链中参与过记账节点的个数。$count_{N^c}$表示区块链中候选节点的总数。本次实验中都会将k值设定为 11，同时设定一轮共识周期中每个节点包含 20 轮生产周期且一轮生产周期的时间定义为 300 毫秒。在对记账节点分布率测试时，系统通过启动 100 个节点来模拟区块链中的候选节点,其中 10 个节点为恶意节点，每个节点拥有的票数是通过随机数产生的。分别对DPoSPR共识算法和DPoS共识算法 60 次记账节点选举中参与记账的数目进行统计，实验多次取平均值得到的实验结果如下图 3.6 所示。

从图 3.6 中可以看到，在选举开始时由于二者都是选择 11 个记账节点因此二者记账节点的分布率均为 11%。通过 60 次选举后二者之间的记账人分布率的值产生了明显的差异。通过分析，认为产生这种现象的原因有二个。一是由于对记账节点只能在完成工作量证明的共识节点中产生，而工作量证明本身就是一个随机事件，谁都无法预测到下一次验证是否会成功，因此相比于原算法本方案可以有更多的节点参与到共识过程中。二是由于当记账节点发生故障或者产生恶意区块时，本方案可以立即通过备选节点集合对其进行替代，这也增加了不同节点记账的可能。综上所述，采用DPoSPR共识算法可以保证记账节点分布得更加广泛，这将对区块链网络的去中心化程度有更好的保障。

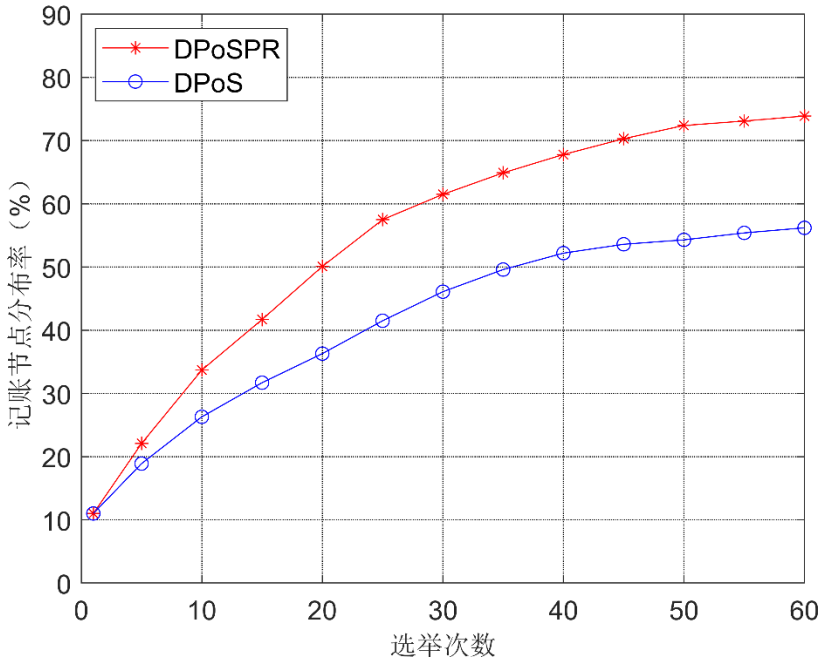

图 3.6 记账节点分布率

3.3.3 交易时延分析

交易时延是指区块链中的节点把产生的交易从提交到该交易被确认为不可逆所耗费的时间。一笔交易所花费的交易时延越短，网络产生区块分叉的可能性越小，整个网络的安全性也就越强。本文按照公式 3.7 对交易时延进行定义。

$$transactionDelay = broadcast_{time} + verify_{time} + generateBlock_{time} \quad (3.7)$$

上式中，$transactionDelay$表示一笔交易的交易时延。$broadcast_{time}$表示节点广播交易的时间，它由二部分组成：记账节点将生成区块广播给验证节点集合的时间和验证节点对区块进行验证后把验证信息发送给记账节点的时间。$verify_{time}$表示验证节点对区块验证所花费的时间。$generateBlock_{time}$表示一笔交易被打包进区块的时间。为了对交易时延测试，在区块链中的候选节点数为 100 的条件下，设置 10 个恶意节点。从系统正常运行时开始每隔 10 个共识周期在每个记账节点的生产周期内向区块链中发送交易进行交易延时测试，多次测试后取平均值得到的实验结果如下表 3.5 所示。

表 3.5 交易时延

算法 \ 周期	1	10	20	30	40	50
DPoS	2528ms	2514ms	2545ms	2506ms	2522ms	2531ms
DPoSPR	441ms	413ms	387ms	358ms	368ms	361ms

从表 3.5 中可以看到，本文所提共识算法在交易时延上相比原始共识算法要低，随着时间的增加交易延时在逐渐减小且慢慢趋于稳定。之所以会产生这样的结果是因为 DPoS 共识

算法在区块生成后需要等到超过 2/3 个其他记账节点逐一进行区块验证后才能确定区块状态的不可逆，而在 DPoSPR 共识算法中，记账节点出块成功后只需要把该区块广播给验证节点集合进行验证即可，因此 DPoSPR 共识算法在$verify_{time}$所花费的时间小于 DPoS 共识算法，从而使得整个区块链网络中的交易时延有所下降。本文所提算法的交易延时逐渐减小且慢慢趋于稳定的原因是由于引入了信誉机制，在系统运行一定时间后恶意节点的信誉值将会降低，其被选为记账节点的概率逐渐减小。

3.3.4 吞吐量分析

吞吐量是指在单位时间内区块链网络所完成交易同步的数据量。在相同条件下，共识算法在单位时间内处理和同步的交易数越多，则代表该共识算法所能承受的网络流量越大，这对于区块链技术在大数据时代能够得到更广泛的应用来说至关重要。吞吐量大小的定义如公式 3.7 所示。

$$TPS = \frac{sum（transaction）}{Time} \tag{3.8}$$

上式中，TPS表示区块链网络的吞吐量，$Time$表示共识时间，$sum（transaction）$表示共识时间内完成同步的交易数量。本实验从节点数为 35 个开始，每次增加 5 个节点，直到增加到 80 个节点为止。在相同节点数量时，通过客户端不断向区块链中发送交易并进行吞吐量测试，取不同时刻的$Time$值计算平均值后得到的实验结果如下图 3.7 所示。

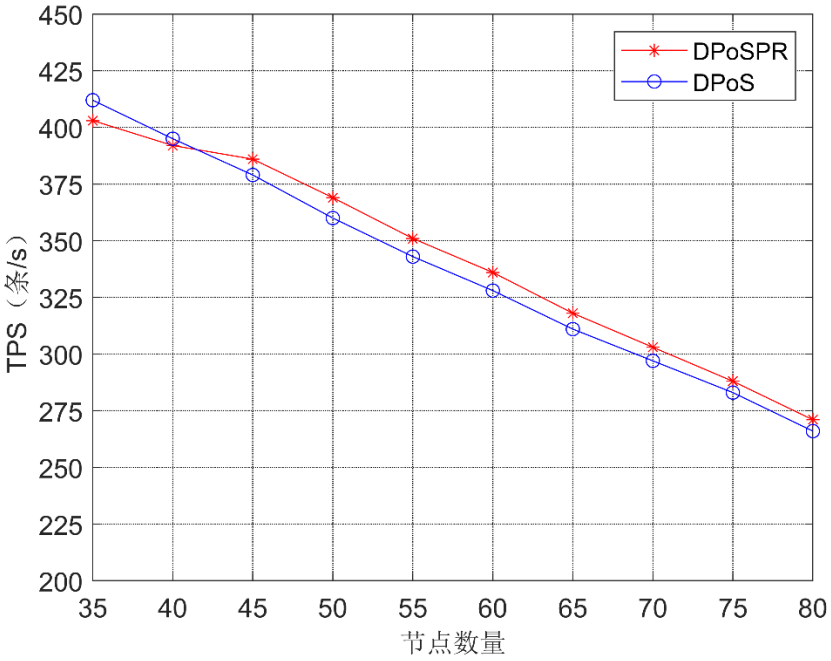

图 3.7 共识算法吞吐量对比图

从图 3.7 中可以看到，共识算法在网络中节点数量不断增加的过程中整体的吞吐量是下

降的。DPoSPR 共识算法在系统节点数较少时的系统吞吐量略小于 DPoS 共识算法，其原因是因为记账节点的选举需要通过工作量证明的方式首先选举出共识节点，当节点数目不多时，记账节点和验证节点部分算力会受到这方面的影响。随着节点数的增多，工作量证明选举共识节点的过程中不再需要当前记账节点和验证节点进行参与。在 DPoSPR 共识算法中区块验证是由验证节点完成的，因此相同时间内同步的交易数略大于原始算法。在本次实验中并未设置恶意节点，如果记账节点中存在恶意节点，由于本方案中引入了记账节点替换机制和信誉机制，DPoSPR 共识算法在吞吐量上会有更明显的提升。

3.4 本章小结

本章节通过结合工作量证明和信誉值的方式针对 DPoS 共识算法存在的问题进行优化，提出 DPoSPR 共识算法。该算法通过引入工作量证明机制选举出共识节点，降低了拥有数字货币数量多的节点对记账节点选举的影响，提高区块链的去中心化程度。在投票选举阶段，每个非共识节点都需完成两次投票最终。第一次投票是选举记账节点。第二次投票是选举验证节点。最终的选举结果也会把节点的信誉值考虑在内，保证记账节点和验证节点具有较高的可靠性。最后在共识阶段由验证节点对产生区块进行验证以减低交易时延，同时引入记账节点更换机制加快对恶意节点的处理速度。经实验分析表明，本文所提共识算法相较于原始算法在记账节点分布率、交易时延和吞吐量上都有一定的提升。

第四章 面向区块链溯源的链下扩展存储方案研究

区块链自身所具有的特性使得它在解决传统溯源系统中带有天然的信任优势。但由于区块链本身是一种基于分布式的系统，对于区块链中存储的数据所有节点都会进行备份，而供应链溯源存在数据量大的特点，这会导致溯源数据占用大量内存增加溯源系统维护成本且区块链的存储压力最终会成为数据溯源的主要瓶颈。因此本章通过数据库存储和密码学技术提出一种面向区块链溯源的链下扩展存储方案。

4.1 基于区块链的供应链溯源模型

考虑到供应链流程中涉及的企业数量较多，且企业规模的大小也不同，因此本次设计不再按照企业进行分类，而是进行更细粒度的区分。如果企业规模较大，则把企业内部按照生产产品的不同设置多个部门，企业中的核心部门拥有该企业区块链账户的私钥，代表该企业参与区块链网络的维护以降低私钥泄漏的风险。企业中其他部门的上链数据需要交由其所属企业的核心部门审核，审核通过后由核心部门上传到区块链中。如果企业规模较小，这类企业往往稳定性不强且诚实度也有待考察，因而不再单独地作为一个区块链节点。规模较小企业首先需要向其企业性质类型相同的规模较大企业做出申请，申请成为该企业的一个部门参与供应链溯源，等到其形成一定规模后再独立参与区块链的维护。通过这种方式可以使规模较小企业快速参与到区块链溯源中，其自身不再需要对区块链的软硬件环境进行部署与搭建。在下文中为方便描述，统一将企业中的各部门称为子企业，为区别核心部门与其他部门，核心部门称为核心子企业，其他部门则称为非核心子企业。企业组成关系如图 4.1 所示。

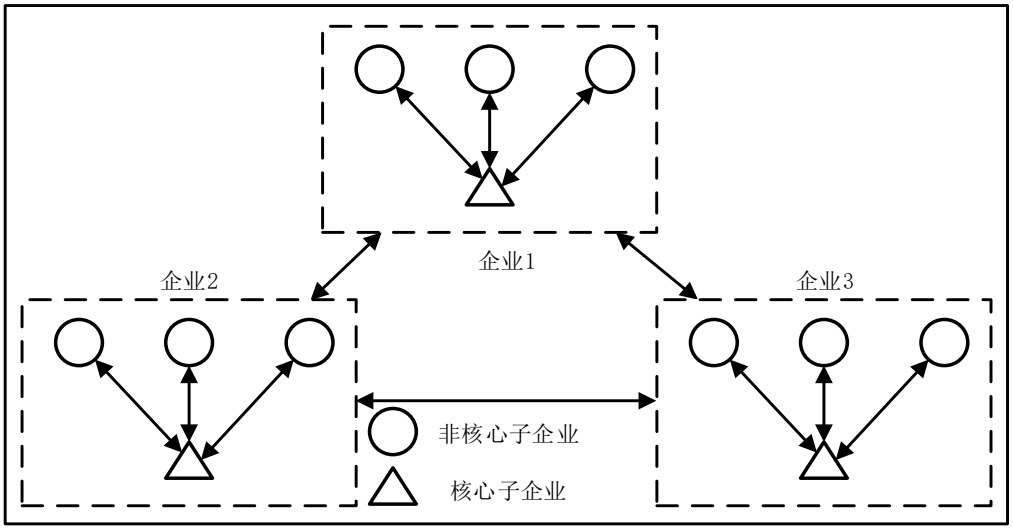

图 4.1 企业组成关系

在一条完整的供应链中主要由原材料生产企业、运输企业、商品生产企业、商品销售企业四类企业构成。一条供应链中所涉及的子企业集合为$Q=\{q_1,q_2,....,q_n\}$，其中q_i子企业对其加工产生产品的信息集合由$M=\{m_1,m_2,....,m_k\}$表示。一件商品从原材料到加工完成所产生相关数据的传递顺序是按照Q集合中的顺序依次进行的，q_1是指原材料生产企业，q_n表示的是销售企业。如果节点q_i在节点前面q_j即$i+1=j$则把q_i称为q_j的上游企业，反之则称q_j为q_i的下游企业。

在基于区块链的供应链溯源中，由于区块链技术是一种基于分布式的系统，对于区块链中存储的数据所有节点都会进行备份。而供应链溯源数据存在数据量大的特点，如果所有子企业所产生的产品信息都直接存储在区块链网络中，这会导致溯源数据占用大量内存增加溯源系统维护成本且区块链网络会承受较大的存储压力。因此本文提出一种面向区块链溯源的链下扩展存储方案，该方案通过减少子企业上链数据占用存储空间大小，进而达到扩展区块链网络存储的目的，其具体实现方案将在下文进行详细阐述。

4.2 面向区块链溯源的链下扩展存储方案

4.2.1 链上数据存储设计

本文提出链上数据存储设计方案是基于以下两点考虑的：1）保证链下数据的不可篡改性。2）确定数据上传者的身份信息可靠性。针对第一点，采用 2.2.1 小节所描述的哈希算法，利用哈希算法的单向性，保证链下数据的不可篡改。如果链下数据有部分被修改，则其得到的哈希结果是不一样的。对于第二点，选择 2.2.3 小节所描述的 SM2 加密算法保证，子企业方通过 SM2 加密算法产生的私钥对数据产生的哈希值进行签名，保证上传数据者身份是确定的。上链信息包含以下两部分：哈希值和签名值，其中哈希值如公式 4.1 所示，签名值如公式 4.2 所示。

$$Hash_i = hash(Hash_p, P_i) \tag{4.1}$$

$$Sign_i = Signature(Hash_i, SK_i) \tag{4.2}$$

$Hash_p$表示当前子企业的上游企业数据的哈希值，其值可以直接从区块链上获取，但在使用前需要利用上游企业的签名值对其进行校验，验证通过后才能使用，这保证了上游企业哈希值的可靠性，其校验方法如公式 4.3 所示，如果签名值未通过校验，则拒绝接受上游企业产品。P_i为子企业q_i的链下数据。$hash$表示通过 SHA-256 哈希算法对括号内数据进行哈希运算

的函数。$Hash_i$表示当前子企业q_i数据的哈希值。SK_i表示当前子企业q_i由 SM2 加密算法产生的私钥。$Signature$表示通过私钥对括号内的哈希值进行签名的函数，子企业q_i利用自身私钥SK_i对$Hash_i$进行签名得到当前子企业q_i数据签名值$Sign_i$。在计算当前子企业明文数据的哈希值时会包含其上游企业的哈希值，使数据库中存储的关于一件商品的溯源数据形成链式结构。选择这种存储方式是因为当消费者在进行商品验证时，在获取链下明文数据后只需要读取最后一个子企业的链上哈希值与签名值即可完成验证而不必将所有子企业的链上数据都进行获取。通过这种存储设计可以提高溯源系统的验证效率。

$$res = verifySign\left(Sign_{i-1}, Hash_p, PK_{i-1}\right) \tag{4.3}$$

$Sign_{i-1}$为子企业q_{i-1}的签名值，$Hash_p$表示子企业q_{i-1}存在链上的哈希值，PK_{i-1}表示子企业q_{i-1}的公钥，$verifySign$是对当前签名值进行校验的函数，res为$true$时则表示验证通过。

4.2.2 链下扩展存储方案

基于上小节的链上数据存储设计，提出链下扩展存储方案。为说明该方案的流程，首先给出其整体结构图如图 4.2 所示。

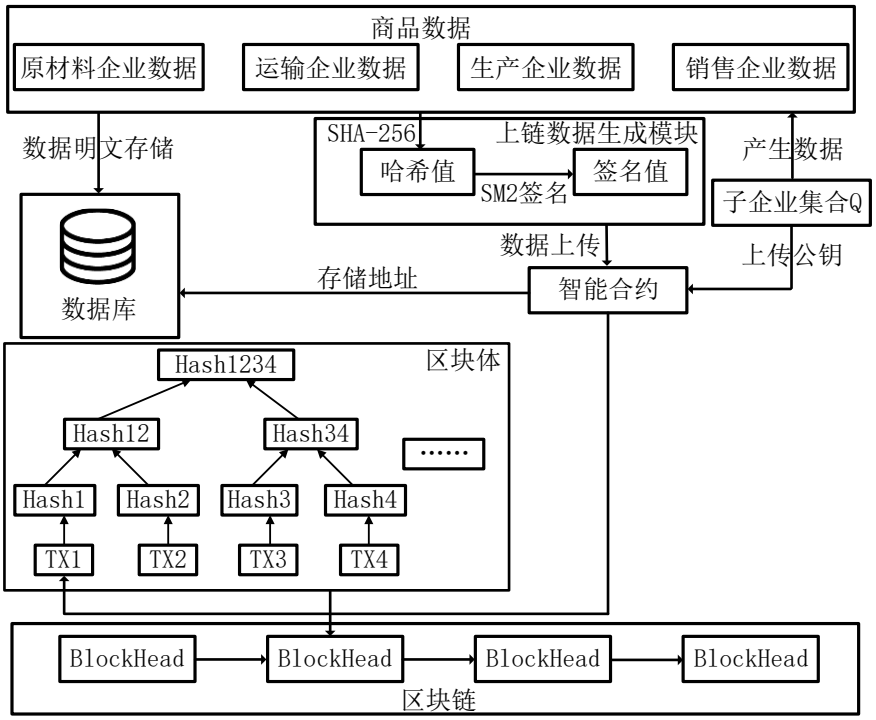

图 4.2 链下扩展存储方案结构图

图中上链数据生成模块的实现方法如 4.2.1 小节所示，它对商品各个环节产生的链下数据进行处理并把链下数据产生的哈希值和签名值结果通过部署智能合约的方式存储在区块链中。在子企业集合 Q 中，各子企业数据存储流程如图 4.3 所示。

第一步，子企业需要把公钥通过智能合约上传到链上，利用区块链的不可篡改性保证下游企业得到的上游企业公钥是完全可信的。

第二步，子企业q_i从区块链上获取上游子企业q_{i-1}的$Hash_{i-1}$、$Sign_{i-1}$和PK_{i-1}，并对其进行签名校验。

第三步，如果签名验证成功则子企业q_i将明文数据M_i、上游企业数据在其对应表的存储位置pid和上游企业链上数据存储位置$address_{i-1}$保存在数据库中，id 是由数据库按照自增原则自动生成的。这些数据组成了子企业q_i的链下数据P_i。

第四步，子企业q_i把链下数据按照 4.2.1 小节公式 4.1 和公式 4.2 的数据处理方式对其明文数据P_i进行处理，生成数据$Hash_i$和$Sign_i$，并把其交由核心子企业进行验证。核心子企业验证成功后会把哈希值和签名值上传到链上，然后把链上数据保存的合约地址存储到对应的链下数据中。

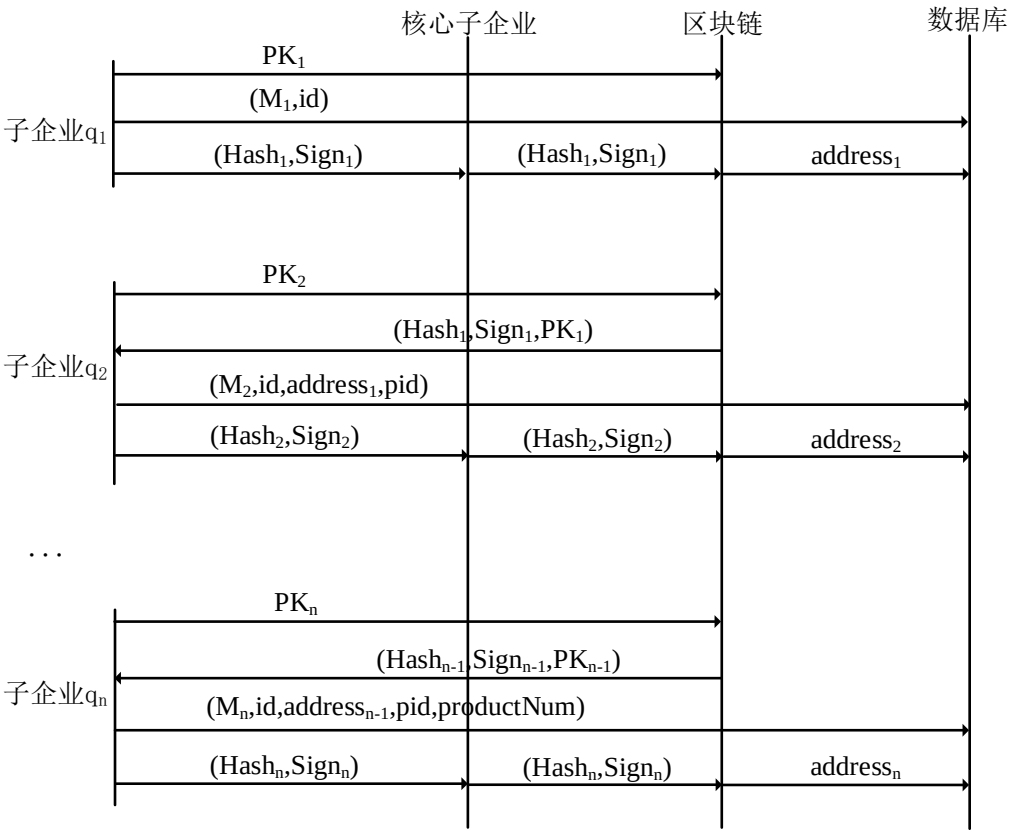

图 4.3 数据存储流程

由于q_1子企业没有上游企业，因此它没有对链上数据进行获取的步骤。q_n子企业增加了一个$productNum$字段，该字段可找到q_n子企业的数据并且可以通过链下存储位置pid和链上存储地址$address_{i-1}$可以唯一确定一个中间环节信息完成向上追踪，通过这种方式，可以获得商品从原材料到销售的全部数据，用于对商品的数据进行溯源。当前子企业获得其上游企业数据后，通过 4.2.1 小节公式 4.3 所示的验证方法对其哈希值进行校验，如果验证未通

过校验，则拒绝接受上游企业产品。

4.2.3 溯源数据验证与问题子企业定位

基于区块链的溯源系统主要是保证商品数据的真实性并对错误数据进行溯源，找到问题子企业。本文提出的链下扩展存储方案溯源数据验证流程如图 4.4 所示。

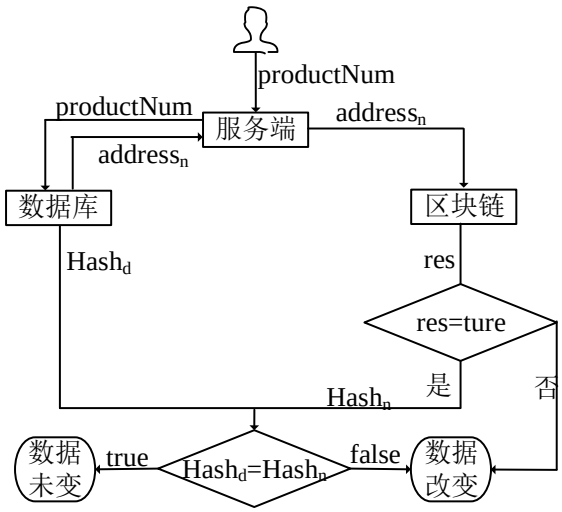

图 4. 4 溯源数据验证流程

消费者把商品编号$productNum$输入到服务器端后，服务器端首先从数据库中得到q_n子企业链上信息存储的位置和商品从q_1到q_n子企业的所有链下明文信息。商品信息按照 4.2.1 小节公式 4.1 的处理方式进行哈希运算生成哈希值$Hash_d$，然后通过链上信息存储位置$address_n$获得$Hash_n$和$Sign_n$，同时从链上获取企业q_n的公钥，并对其按照 4.2.1 小节公式 4.3 进行校验。如果res校验结果为$true$则将$Hash_d$和$Hash_n$进行对比，相同则返回$true$代表商品数据未发生改变，不同则返回$false$代表商品信息有误发生改变。如果res校验结果为$false$则直接返回，代表商品信息有误发生改变。

如果消费者验证商品信息后发现商品信息存在部分被篡改的情况，则溯源系统会把数据库中关于该商品的所有信息按照表 4.1 中的方法对数据有问题的子企业进行定位。由于当前子企业链上存储的哈希值与其上游企业的哈希值有关，因此验证过程从第一个子企业开始进行验证。其中 getPK 表示从链上获取子企业q_i公钥，getHash 表示从链上获取子企业q_i链下数据的哈希值，getSign 表示从链上获取子企业q_i链下数据的签名值。

表 4. 1 问题子企业定位方法

Input: 链下各子企业数据集合P_i和$address_i$，公钥存储合约地址$address_p$
Onput: 数据错误子企业集合E
1 E // 数据错误子企业集合

```
2  Hashb1 = getHash(address1)
3  if(hash(P1) != hashb1)
4        E.add(q1)
5  end if
6  Hashbpre  = Hashb1 //  保存上游企业链上哈希
7 for(i = 2; i <= n - 1; i++)
8        Hashbi = getHash(addressi)
9        Hashdi = hash(Hashbpre,Pi)
10       if(Hashbi != Hashdi)
11             E.add(qi)
12       end if
13        Hashbpre = Hashbi
14  end for
15  PK = getPK(addressp,qn)
16  Hashbn = getHash(addressn)
17  Signn  =  getSign(addressn)
18  res = verifySign(signn,Hashbn,PK)
19  if(res == false || hash(Hashbpre,Pn) != Hashbn)
20        E.add(qn)
21  end if
22  return E
```

q_1子企业没有上游企业，所以单独对其数据进行校验。除q_n子企业外其他子企业的链上签名值已经在数据进行传递的过程中完成了校验，如果其签名值有误，则不会再有后续流程，因而在进行问题子企业定位时直接将链下数据的哈希值与链上存储的哈希值进行对比即可。对于q_n子企业没有下游企业，其链上签名值未经过校验，需要单独对其执行一次签名值校验过程。

在对当前子企业获取链上哈希值时，子企业应该采用其下游企业存储的链上地址值。其原因有两个：1）如果使用子企业自身存储的链上数据地址，那么可能会存在当前子企业为恶意子企业修改了链下数据，同时也修改了链上数据存储地址，则会影响其下游所有子企业的计算结果。其自身数据不会受到任何影响。2）如果当前子企业修改了其上游子企业的链上存储地址，由于当前子企业进行验证时该子企业的链上数据存储地址是通过其下游子企业获取的，因此其自身的运算也会受到影响。综上所述选择下游子企业保存的链上存储地址更能够保证溯源信息的可靠性。q_n子企业没有下游企业，其链上数据存储地址需要使用自身存储的值，但其所保存的值对商品本身不会产生较大的影响且该子企业的值也不会影响到其他子企业。如果子企业对溯源结果有异议，认为自身数据的异常是由于其下游企业存储的链上存储地址值被修改过造成的，则可以通过数据变更日志进行确认。当前子企业进行哈希运算

时带入的上游企业哈希值是上轮校验时从链上得到的，每轮校验结束时需要对其进行更新。

4.3 基于链下扩展存储方案的区块链溯源系统实现

4.3.1 系统架构设计

在对系统进行搭建前，首先需要对开发技术进行选型，合适的系统架构设计不仅可以在开发中对项目有一个整体的把握，而且在进行后续功能扩展或者性能优化时也可以更高效。基于链下扩展存储方案溯源系统的技术架构如图 4.5 所示。

为提高溯源系统的可维护性和扩展性，本系统使用的是 B/S 服务器系统架构，其具有以下优势。第一，用户直接使用本地的浏览器即可完成操作，方便用户对平台的接入。第二，后期在系统进行升级与迭代时用户无需进行任何操作，兼容性强。系统技术架构可以分为三部分：用户展示层、服务层和数据存储层。

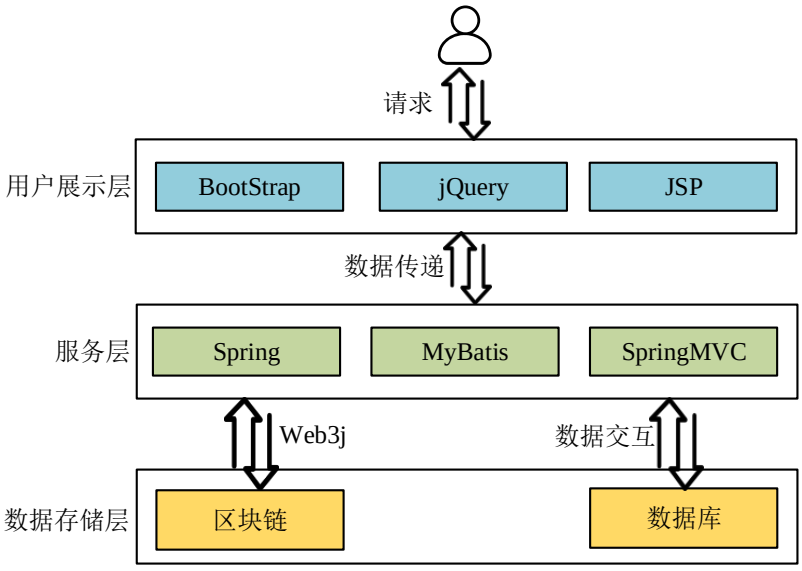

图 4.5 系统技术构架

用户展示层选择 BootStrap+jQuery+JSP 完成与用户交互界面的搭建，它为企业和消费者对溯源数据进行上传和查询的操作进行了简化，使用者无需了解底层具体是如何实现的，只需要对其进行简单的了解就可以很容易的上手操作。用户展示层主要实现用户注册、用户登录、各企业信息录入、溯源信息查询等页面。

服务层选择主流的 Java 语言进行构建，采用 SSM（Spring+SpringMVC+Mybatis）的开源框架对用户端发送的信息进行分层处理减少后台系统对数据的耦合性。企业通过用户展示层完成数据录入后，其输入的信息将会通过 HTTP 请求传递给服务层，服务层首先需要对用户展示层传递的数据进行校验，如果有不符合规定的信息传入则直接向用户返回错误结果。符

合要求则执行相应的添加或删除操作把数据持久化存储到数据存储层，然后向用户返回执行结果，把结果在用户展示层进行展示。Web3j 是 Java 应用与区块链网络进行数据交互的一个轻量级 Java 开发库，实现了 JSON RPC 接口的协议封装，它为与智能合约进行交互提供了函数和对象，也可以对区块链中的交易信息、账户信息等进行查看与操作。服务层为前端提供可用的服务接口和实现与数据存储层的数据交互，其中主要包括用户管理模块、各企业数据管理模块、溯源信息查询管理模块。

在本次设计中数据存储层分为两部分：数据库和区块链。链下数据库是为了提高区块链网络的扩展存储能力，减少链上数据存储的负担。区块链对明文数据产生哈希值和签名值的存储与读取是通过部署和调用智能合约实现的，其具有不可篡改的特点，因而它是保证整个溯源系统数据是否可信的关键因素。本次实验选用 Ganache 作为以太坊网络，它是基于本地内存的以太坊测试网络，主要用于开发和测试，实现了真实以太坊网络的所有功能。

4.3.2 数据库表设计

结合 4.2 节所述的面向区块链的链下扩展存储方案设计，本次各企业登录平台的账户信息及其产生的商品相关数据的明文数据都存储在链下数据库中。因此本小节首先对数据库表进行设计。

首先是供应链参与子企业信息管理表如表 4.2 所示，每个子企业首先通过以下信息进行注册，其中 eth_address 字段表示该子企业所属企业的区块链地址。duty 字段表示当前子企业的身份，其中 1 表示原材料生产企业，2 表示运输企业，3 表示商品生产企业，4 表示商品销售企业。identification_code 为子企业身份识别码，每个子企业的身份识别码唯一。status 表示当前子企业用户的状态，子企业用户通过申请后该值设置为有效，如果该子企业多次被举报产生恶意数据则把其所属的企业内所有子企业都置为无效状态，只有子企业的状态为有效时其才能够登录平台。

表 4.2 子企业信息管理表

字段名称	字段说明
id	子企业序号
login_account	子企业登录用户名
password	子企业登录密码
name	子企业名称
email	子企业邮箱
create_time	子企业创建时间
eth_address	所属企业区块链地址
identification_code	子企业身份识别码
address	子企业地址
core	是否是核心子企业

duty	子企业身份
status	子企业状态
num	当前子企业在所属企业内编号

消费者需要单独的表进行存储，每位消费者只能对其所购买商品进行溯源查询。消费者信息管理表如表 4.3 所示。其中 identification_code 为消费者的唯一身份识别码，当消费者购买商品后，销售企业需要为该商品设置相应买家。

表 4.3 消费者信息管理表

字段名称	字段说明
id	消费者序号
login_account	消费者登录用户名
password	消费者登录密码
identification_code	消费者身份识别码

在供应链管理中，各类企业需要上传关于商品相关的溯源数据字段，因此首先需要对各企业上传的数据字段进行明确。表 4.4 是对原材料生产企业信息、运输企业信息、商品生产企业信息、商品销售信息溯源信息的整理。在下表中每个企业信息表中存储的信息合约地址是其溯源明文数据所产生的哈希值与签名值在区块链中存储的位置，通过 Web3j 提供的 load 方法加上该合约地址即可对该合约中所存储的数据进行读取。除了原材料生产企业外，其他企业都有 pid 字段，该字段是当前企业与其上游企业的数据所在 id 进行映射，通过商品销售企业的 pid 向上查找即可获得该商品所涉及的所有链下明文存储数据。

表 4.4 溯源信息管理表

企业类型	字段名称	字段说明
原材料生产企业	id	记录原材料序号
	material_address	原材料信息合约地址
	material_type	原材料所属种类
	img_url	原材料图片链接
	material_name	原材料名称
	material_creator	原材料方身份识别码
	address	原材料生产地址
	create_time	原材料生产时间
运输企业	id	记录运输序号
	transport_address	运输信息合约地址
	pre_address	上游企业数据链上地址
	transporter	运输方身份识别码
	number_plates	运输方车牌
	delivery_time	出库时间
	img_url	运输图片链接
	delivery_number	运送单号
	pid	上游企业数据链下编号
	id	记录生产序号
	prodcuct_address	商品信息合约地址
	pre_address	上游企业数据链上地址

商品生产企业	producer	生产方身份识别码
	img_url	商品图片链接
	product_type	商品类型
	product_name	商品名称
	create_time	生产时间
	shelf_life	保质期
	address	商品生产地址
	pid	上游企业数据链下编号
商品销售企业	id	记录销售序号
	seller_address	销售信息合约地址
	pre_address	上游企业数据链上地址
	seller	销售方身份识别码
	receive_time	商品收到时间
	address	销售地址
	img_url	销售图片链接
	product_num	商品编号
	pid	上游企业数据链下编号

核心子企业需要对其所属企业内的生产数据进行上链审批。待验证数据表如表 4.5 所示。核心子企业通过区块链地址来识别属于该企业的待验证数据。

表 4.5 待验证数据表

字段名称	字段说明
id	待验证数据序号
enterprise_number	子企业编号
product_number	产品编号
hash_value	哈希值
sign_value	签名值
affiliated_enterprise	所属企业区块链地址

4.3.3 智能合约的实现与部署

在区块链溯源系统中，智能合约是用户与区块链交互的桥梁也是对上链数据的一种规范约束，它与溯源系统的服务层通过 Web3j 进行交互，核心子企业通过部署智能合约把数据上传到区块链中，保证了哈希值和签名值得不可篡改性，同时也可以通过加载智能合约对已经上传了的数据进行读取。智能合约使用的编程语言为 Solidity，用户可以在开源的 Remix 平台上进行开发。在智能合约的设计上，本次设计总共分为 2 个合约：子企业公钥保存合约和信息存储合约。

子企业公钥保存合约（EnterpriseManagement）是用来对各子企业公钥进行保存的，利用区块链技术的不可篡改性保证参与供应链溯源的子企业获得的公钥是可靠的，它由监管机构进行创建的，也只有监管机构方才能够对子企业公钥进行上传。其变量和函数如图 4.6 所示。

其中 manager 表示监管机构的以太坊地址，enterprisePK 是企业地址与其内部子企业公钥的一个映射，ifRegister 是对该子企业是否已经注册进行判定的映射。监管机构通过 setEnterprisePK 函数对子企业进行注册并上传子企业公钥。ifLegitimate 函数用于对其企业内部子企业当前状态进行查询。getPK 函数用于指定子企业的公钥进行获取。当企业内部产生变动时，需要调用 changeRegisterStatus 函数对子企业状态进行变更，保证系统的安全性。

变量		
变量类型	变量名称	作用
address	manager	合约的构建者
mapping（address => string[]）	enterprisePK	企业内所有子企业公钥存储
mapping（address => bool[]）	ifRegister	企业内子企业是否注册
函数		
函数输入参数	函数名称	函数功能
企业地址，企业公钥	setEnterprisePK	对企业进行合法注册并保存公钥
企业地址,在企业中的编号	ifLegitimate	判断企业是否注册
企业地址，在企业中的编号	getPK	获取企业公钥
企业地址,在企业中的编号	changeRegisterStatus	修改企业状态

图 4.6 子企业公钥保存合约

信息存储合约（EnterpriseMessage）是由核心子企业在完成其所属企业的上链信息校验时进行创建的。其变量和函数如图 4.7 所示。

其中 owner 表示当前合约所属企业的以太坊地址。signValue 和 hashValue 表示当前产品数据的签名值和哈希值。为保证同一信息只能够使用一次，合约中设置一个标志 finishFlag，当把物品交由下游企业后该子企业需调用 setComplete 函数对该标志状态进行更改，任何人都不能再对哈希值和签名值进行更改，且下游企业在对当前子企业数据及其签名值进行验证时，finishFlag 必须是 false，避免同一标签再次复活。在还没有对物品交付前，防止子企业因为疏忽输错数据，允许通过 updateValue 函数对其哈希值和签名值进行修改，但只允许修改一次，修改完成后需要把标志 modifyFlag 设置为 true。identificationCode 表示当前子企业所交付的下游企业或者消费者的身份识别码，当前子企业调用 setNextOwner 函数设置其下游企业或者消费者的身份识别码后，下游企业或者消费者才能对当前合约中的哈希值和签名值进行获取。当下游企业对当前子企业产品进行接收时首先通过 getFinishFlag 函数对产品状态进行确认，

如果该产品已经被接受，则拒绝此产品。如果未被确认则在对其签名值进行验证，如果验证通过，则通知产品所属子企业更改当前产品的交付状态。

变量		
变量类型	变量名称	作用
address	owner	合约的构建者
string	signValue	签名值
string	hashValue	哈希值
bool	finishFlag	数据是否交付下游企业
bool	modifyFlag	是否修改过
string	identificationCode	下游子企业识别码/消费者识别码
函数		
函数输入参数	函数名称	函数功能
数据哈希值，签名值	updateValue	对企业哈希值和签名值进行更新
无	setComplete	设置为已交付状态
身份识别码	setNextOwner	设置下游企业/消费者身份码
身份识别码	getHash	获取企业数据哈希值
身份识别码	getSign	获取企业数据签名值
无	getFinishFlag	获取交付状态

图 4.7 信息存储合约

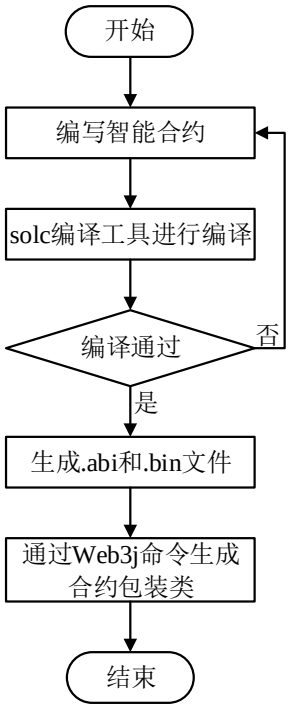

图 4.8 合约包装类生成流程

本文服务端是通过 Java 语言来实现溯源系统与区块链网络进行交互的，智能合约的加载与部署也需要通过 Java 语言来完成，因此首先需要把智能合约转化成相对应的 Java 类。具体流程如图 4.8 所示。

solc 是用于编译 Solidity 文件的命令行工具，智能合约通过编译后将会生成.bin 文件和.abi 文件。.bin 文件是字节码文件，它是最终运行在以太坊虚拟机中的可执行代码。.abi 文件是一个 JSON 对象，它是对智能合约接口的描述。当合约编译成功后，既可通过 Web3j 提供的命令工具来对.abi 和.bin 文件进行执行生成对应的 Java 合约包装类。

当合约生成对应的 Java 包装类后，既可把其复制到项目中进行使用。通过 Web3j 提供的 API 接口可以完成合约的部署工作。部署代码如下所示。

```
public static String setEnterpriseMessage(String hashValue,String signValue,String privateKey) throws Exception {
    Web3j web3j = Web3j.build(new HttpService("http://localhost:7545")); // 与区块链网络建立连接
    Credentials credentials = Credentials.create(privateKey); // 创建合约调用者
    EnterpriseMessage enterpriseMessage = EnterpriseMessage.deploy(web3j, credentials,
        Contract.GAS_PRICE, Contract.GAS_LIMIT, hashValue,signValue).send();
    // 合约部署，其中EnterpriseMessage表示合约包装类
    return send.getContractAddress();// 获取链上部署合约地址
}
```

其中 privateKey 表示相关企业的区块链账户私钥。在进行合约部署时，需要对传入合约构造函数中所需要的参数 hashValue 和 signValue。enterpriseMessage 为合约包装类对象，当获取该值后，即可把其当作一个普通的 Java 类进行使用。智能合约中所定义的方法与变量在对应的包装类中都有相应的映射，且该类中存储的变量值为当前合约在链上存储的值。

合约部署完成后既可在 Ganache 中进行查看，具体的信息如图 4.9 所示。其中最重要的是 CREATED CONTRACT ADDRESS 字段，它表示合约在链上对应的地址 0x9b009255f1dAb39c12e7D7c7893353ABF4C0d60C。在系统运行的过程中可以通过加载该合约的地址与区块链网中的数据进行交互。

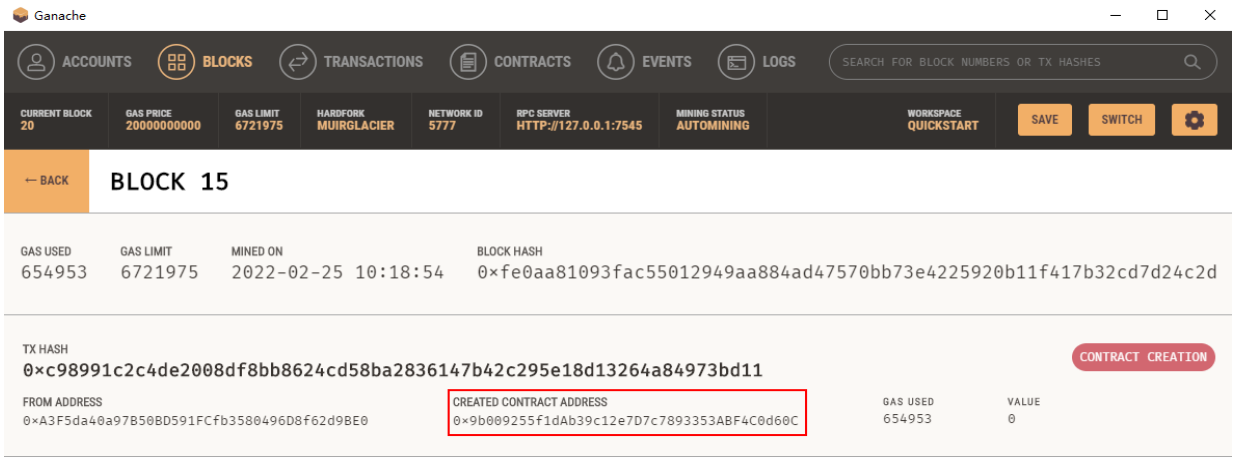

图 4.9 ganache 中合约部署成功展示

区块中其他字段所代表的含义如表 4.6 所示。

表 4.6 区块中字段含义

字段名称	字段说明
GAS USED	GAS 消耗
GAS LIMIT	GAS 最大消耗值
MINED ON	挖矿出块时间
BLOCK HASH	区块哈希
TX HASH	交易哈希
FROM ADDRESS	交易发起方

4.3.4 溯源系统实现

本系统实现是在 Windows 环境下进行，固态硬盘容量是 512GB，内存大小为 16GB，处理器使用的是 AMD R7 4800H。系统开发环境版本如表 4.7 所示。

表 4.7 系统开发环境版本

软件名称	使用版本
数据库	Mysql5.5.40
服务器	Tomcat-8.5.31
JDK	1.8.0_231
Solidity	0.4.25
Web3j	3.4.0
Ganache	2.3.0

根据溯源系统需求以及链下扩展存储方案分析，本文系统按照不同的用户身份可以分为 4 个模块：其中包含企业监管机构模块、核心子企业模块、非核心子企业模块、消费者模块，其具体功能如图 4.10 所示。

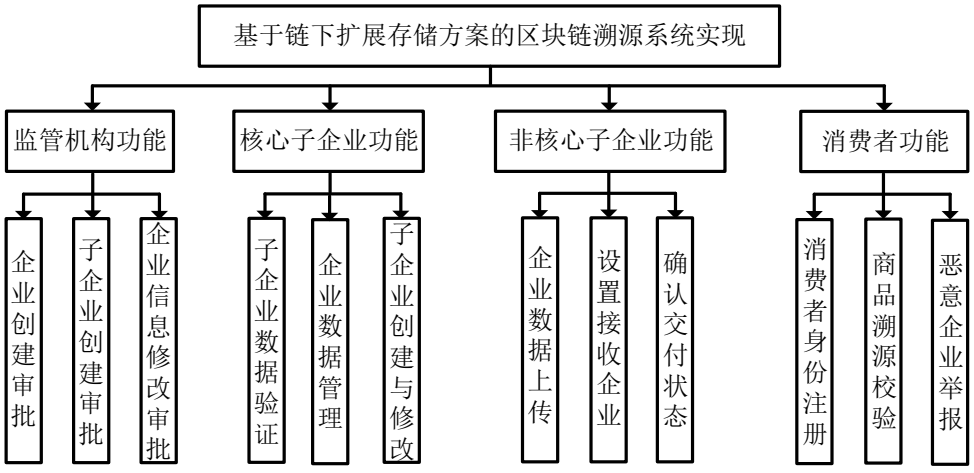

图 4.10 系统功能实现

监管机构具有企业创建审批、子企业创建审批、子企业信息修改审批功能。监管机构负责对子企业加入溯源平台的权限进行审核，只有当完成审批后子企业才能正常登陆。根据 4.1 小节中的模型构建，企业注册分为两类，第一类为企业注册，它是由当前企业内的核心子企

业完成的，这类注册需要在登录界面进行注册，其具体的注册页面如图 4.11 所示。

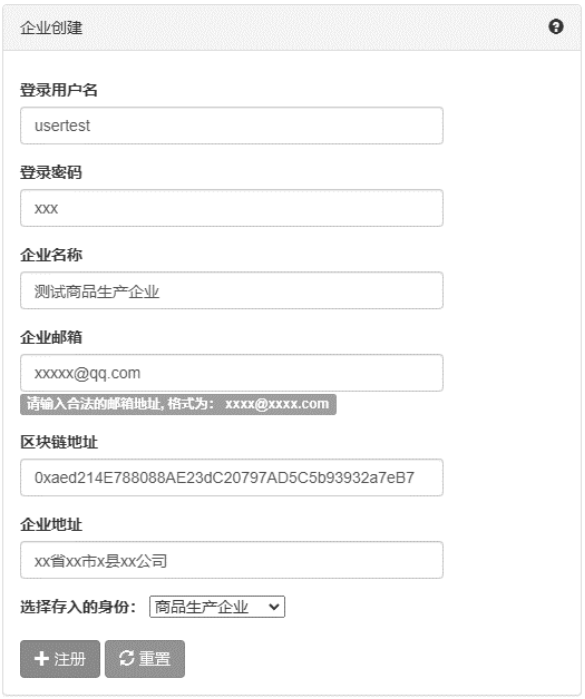

图 4. 11 企业创建页面

当企业注册完成，等待监管机构进行审批，如果审批通过，服务器端会把 SM2 加密算法生成的私钥通过发送邮件的方式通知核心子企业进行保存，公钥则上传到子企业公钥保存合约中，用于后续签名验证。注册时需要填写的字段都包含在了表 4.2 中，创建页面中没有要求输入数据库表中的字段都是由服务器端自动生成的，当完成审批后，子企业即可获取其身份识别码。当子企业通过注册的账号登录到平台上后，该子企业信息会存储在 session 域中，子企业在进行数据上传时不再需要对其自身信息进行输入，直接从 session 域中拿到当前登录的用户即可。

第二类为企业中的非核心子企业注册，这类注册是由其所属的核心子企业完成的，其注册页面与第一类几乎相同，由于创建子企业与核心子企业使用的是相同的区块链地址，因此不再需要输入。用户注册信息与溯源数据无关，因此保存在链下数据库即可。子企业信息修改审批是当子企业信息发生变化时需要提交申请，监管机构负责审批与修改。

核心子企业具有子企业数据验证、子企业创建与修改、企业数据管理功能。企业数据管理功能与非核心子企业功能相同，将在下文一同进行展示。子企业数据验证功能如图 4.12 所示。

核心子企业点击右侧的签名值验证按钮后，服务器端会对当前签名值进行校验。当核心子企业对数据验证完成后，核心子企业输入当前企业私钥既可将该子企业所产生数据的哈希值与签名值通过部署智能合约的方式保存在链上。

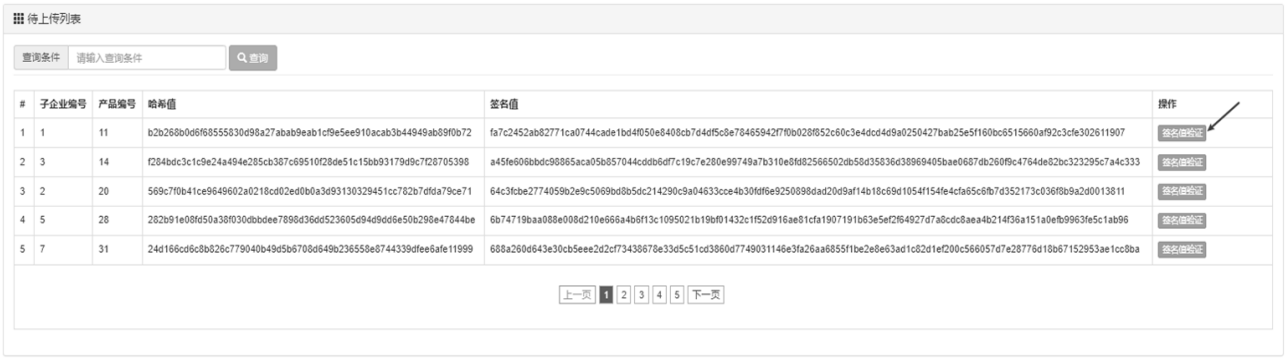

图 4.12 子企业数据验证页面

子企业创建与修改页面如图 4.13 所示。

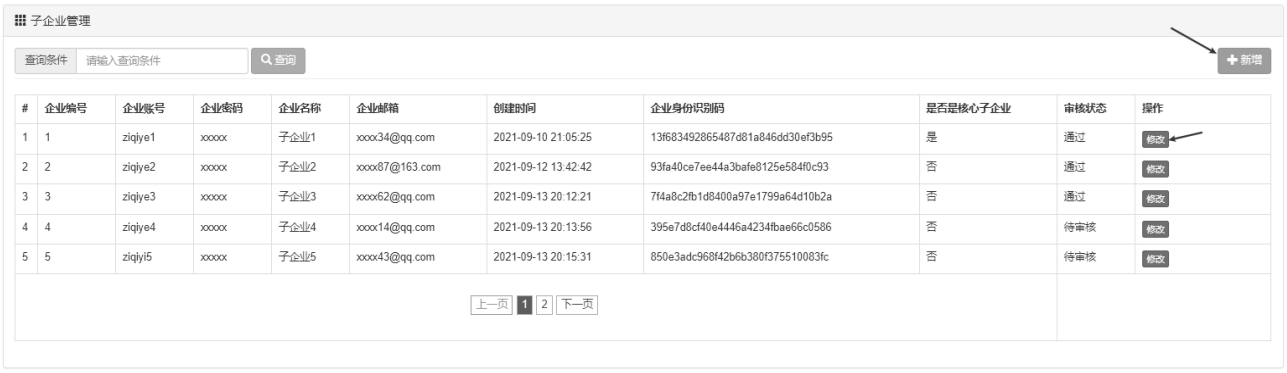

图 4.13 子企业创建与修改页面

当前企业中的所有子企业都是由核心子企业来进行管理的，右上角设有新增功能，当有新的子企业加入时，既可通过该按钮进入创建页面进行子企业创建。企业身份识别码是用于对不同子企业进行区分的，在信息存储合约中会设置该字段，保证企业的链上存储值只能由被授权后的子企业进行获取。当子企业信息发生变更时，点击对应子企业右侧的修改按钮对修改进行说明后，等待监管机构进行审批即可。

非核心子企业具有企业产品数据上传、设置产品接收子企业、确认产品交付状态功能。子企业数据处理流程如图 4.14 所示。

其中商品生产企业产品数据上传页面如图 4.15 所示。在数据上传时需要填写的字段都包含在了表 4.4 商品生产企业中，上游企业相关联的 id 是通过上游企业合约地址进行获取的，企业私钥是指子企业注册时 SM2 加密算法产生的私钥，它用于对当前子企业产生数据的哈希值进行签名。表 4.4 中在该页面没有上传的关于子企业的相关信息可以直接从登录用户中进行获取，当核心子企业对签名值进行校验通过后会把该产品的合约地址保存在数据库中。另外三类企业上传信息页面与本页面类似，只是上传的数据有所不同，不再进行展示。

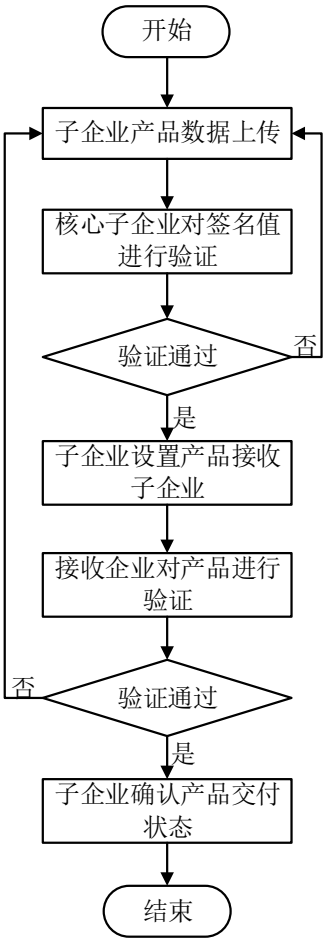

图 4.14 子企业数据处理流程

图 4.15 商品生产企业产品数据上传页面

消费者具有消费者身份注册、商品溯源校验和问题企业举报功能。其查询结果页面如图 4.16 所示。

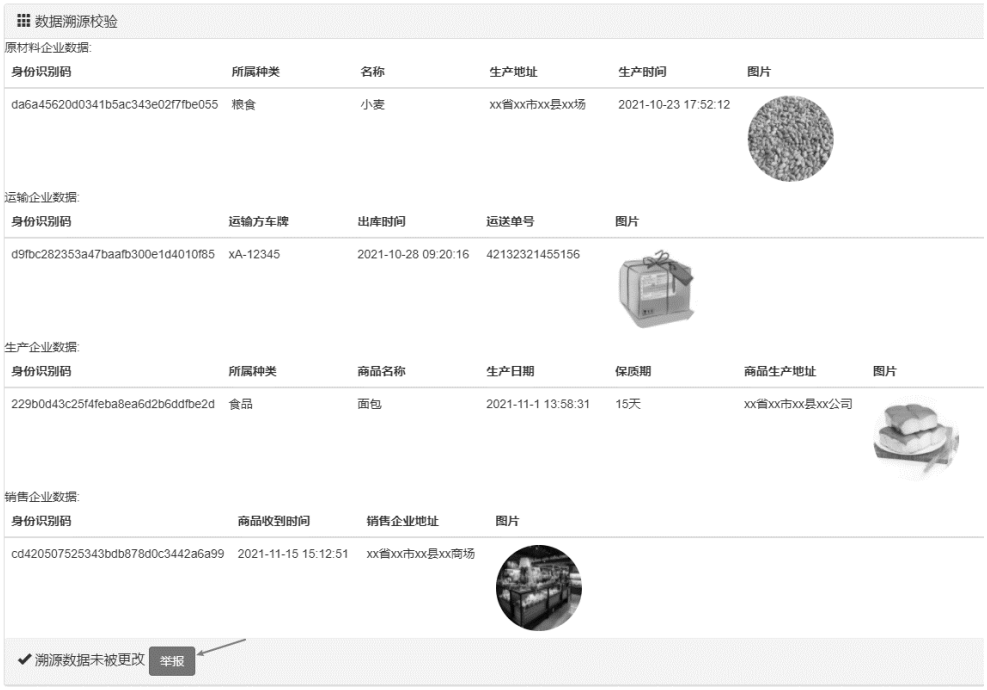

图 4. 16 溯源信息查询页面

消费者身份注册只需填写对应信息即可完成。当消费者购买商品后，可以通过商品的编号在溯源查询页面进行查询。为验证本方案的可行性，简化了溯源流程，每一类企业只包含了一个节点。服务器端会根据商品编号从销售企业数据表中找到该商品，然后按照 pid 对该商品的所有溯源数据进行获取，最后按照 4.2.1 小节公式 4.1 中的数据处理方式对明文数据进行计算得到明文数据的哈希值并与销售企业链上存储的哈希值进行对比。在页面的最下方展示了溯源数据的验证结果。如果数据正确则展示溯源数据未被更改。如果数据库中的明文数据部分有错误，则会按照 4.2.3 小节所示的问题子企业定位流程对问题数据进行追踪，并在最下方展示溯源数据被更改和数据有问题的子企业的集合。消费者可以通过点击举报按钮对相关问题进行举报。

4.3.5 溯源系统性能测试

本文所提方案是把溯源相关的所有数据存储在链下数据库中，区块链中存储的数据包含哈希值和签名值两部分，这两部分占用内存的大小仅有一百多字节远小于溯源明文数据直接上链，以此来分担区块链网络的存储压力。文献[16]中的数据分割存储方案也是通过数据库对区块链的存储进行扩展，因此本文按照数据分割存储的思路对商品溯源数据进行处理，并与本文所提方案从上链数据占用存储空间，系统响应时间两方面进行对比。

首先是上链数据占用存储空间对比，从溯源商品数量为 500 件时开始测试，每次增加 500 条商品溯源数据，直到溯源商品数量达到 3000 件为止。商品溯源数据按照本文方案和

数据分割存储方案对溯源数据进行处理，其对比结果如图 4.17 所示。

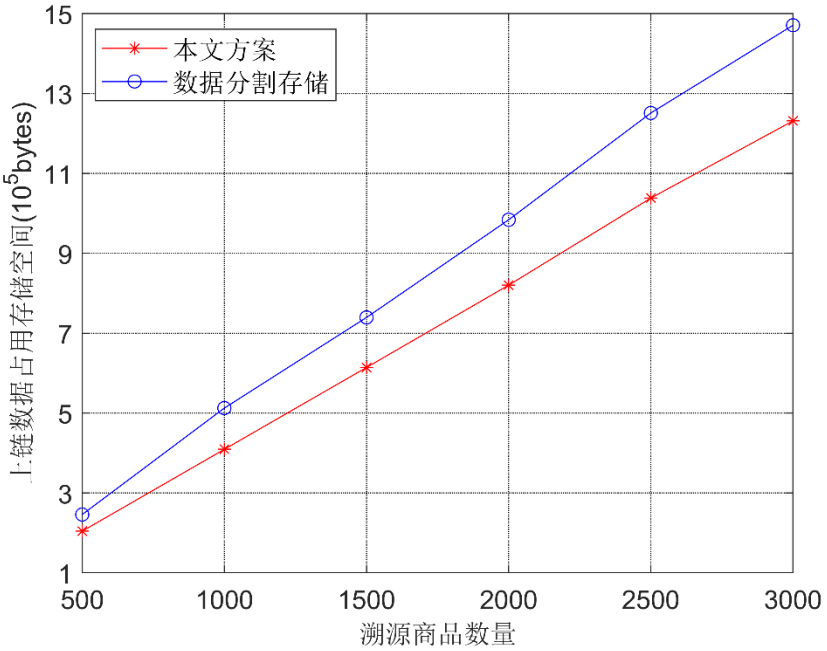

图 4.17 上链数据占用存储空间对比

本文方案上链数据包含哈希值和签名值两部分，因此随着溯源商品数量的增加其上链数据的大小几乎是呈线性变化的。数据分割存储方案中链上溯源数据分为核心数据和非核心数据的哈希值两部分。由于商品图片信息占用内存较大且对商品本身影响不大，因此放在链下存储。商品的属性信息与商品关联性强则存储在链上。每次输入数据是不同的，因此它的上链数据大小会有所波动，但整体上也为线性变化。

其次是商品溯源验证时间对比，本文对存储的溯源商品从 1 件时开始测试，每次增加 1 件，直到溯源商品数量达到 5 为止，其对比结果如图 4.18 所示。

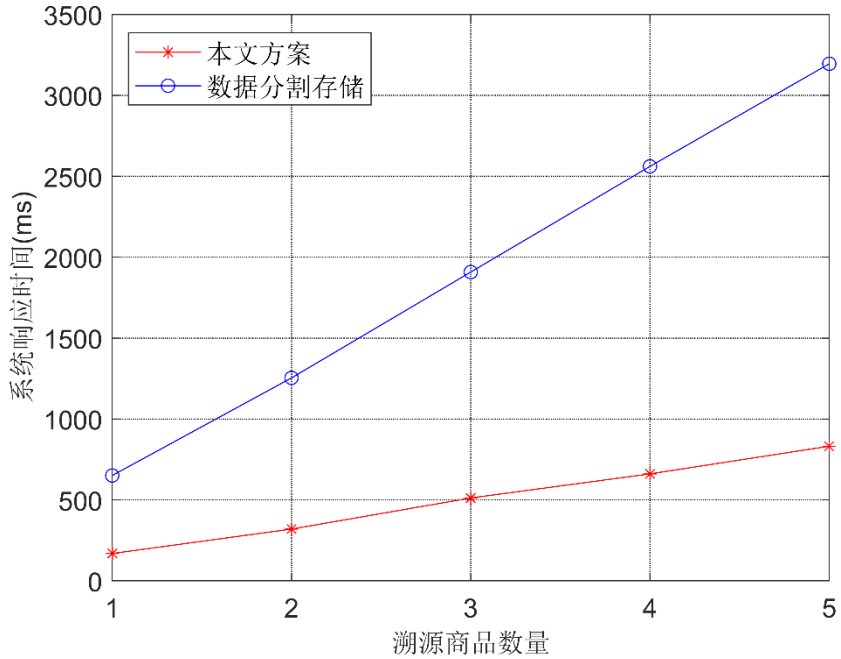

图 4.18 系统响应时间对比

本文方案在链下存储时对商品各阶段信息设计了链式存储结构，因此在溯源校验时只需要读取最后一个子企业的链上存储哈希值和签名值就可以实现商品的溯源校验，且数据库索引值的底层存储结构是通过 B+树进行实现的，这种索引结构使其具有较好的查询性能，因此本方案对商品的溯源校验所需要响应时间不高。数据分割存储方案中由于对各阶段信息分开存储，对于商品中的每一个环节都需要从数据库和区块链两部分进行读取，这就会增加溯源校验时间，因此本文所提方案在商品溯源上具有更快的响应速度。

4.4 本章小结

区块链技术的去中心化特性使得其在面对数据量较大的溯源场景时会存在占用内存较大的问题，因此提出一种链下扩展存储方案。该方案把溯源数据存储在数据库中，通过 SHA256 哈希算法和 SM2 加密算法，在保证数据的可靠性、可溯源性和数据不可被篡改的条件下减少了链上数据的存储大小，提高了区块链的扩展性。通过以太坊区块链根据溯源系统的实际结合 SSM 开发框架对所提方案进行实现，并对系统主要功能进行了展示。最后经过系统性能测试对比表明，本文方案在面对区块链溯源的场景中上链数据占用内存较低并具有更快的系统响应速度。

第五章 总结与展望

区块链技术特点可以为当今社会中存在的信任问题提供有效的解决方案，尤其是在供应链管理领域。但由于区块链技术自身设计实现并不适合直接将其与供应链进行结合，这包括是共识算法和数据存储两个方面。因此本文针对这两个问题完成以下工作。

1）对公有链中常用的共识机算法进行分析。结合供应链管理中参与机构多，交易数量大等特点，选择对吞吐量更高的委托权益证明共识算法中存在的问题进行改进。通过引入工作量证明的方法来选出共识节点增加区块链中参与记账的人数，保证了区块链网络的去中心化程度。在投票选举阶段，每个非共识节点都需完成两次投票，第一次投票是投给共识节点选举记账节点，第二次投票是投给非共识节点选举验证节点。最终的选举结果也会把该节点的信誉值考虑在内，保证记账节点和验证节点的可靠性。最后在共识阶段由验证节点对产生区块进行验证以降低交易时延同时为提高系统处理恶意记账节点的速度和系统健壮性提出记账节点更换机制。通过对原始算法与优化后的算法进行实验验证后表明，所提共识算法在一定程度上能够针对原始算法中存在的问题进行解决。

2）为减轻区块链网络的存储压力，提出一种面向区块链溯源的连下扩展存储方案。该方案首先利用 SHA-256 哈希算法的单向性对明文数据进行哈希运算得到哈希值；然后采用 SM2 加密算法产生的私钥对哈希值进行签名保证信息上传者身份的可靠；最后把哈希加值和签名结果通过智能合约保存在区块链，明文数据和其哈希值与签名值在区块链上存储的地址则存储在数据库中。接着对该方案如何验证溯源数据是否被篡改进行说明。最后采用以太坊区块链平台结合 SSM 框架对所提方案进行实现并对部分功能进行展示。

最后，本文为区块链技术能够更好地与商品溯源进行结合，对共识算法和溯源数据的存储提出相应的解决方案，但目前已经完成的工作仍然有部分值得优化和改进的地方。

1）本文所提出的基于工作量证明和信誉值的委托权益证明共识算法中，节点的信誉值只对记账节点和验证节点这两类节点进行考量，在后续过程中可以把节点的投票行为也纳入信誉值的考量中，对积极投票的节点进行信誉值奖励。并且本文所提方案中只允许候选节点投赞成票，后续也可以考虑更完善的投票模型。

2）本文所有的溯源数据都是通过人工对其进行输入的，这会导致很多人为因素产生不必要的错误，因此后期可以考虑通过物联网的技术对商品的信息进行自动化地采集，通过加入物联网设备使得整个流程更加的自动化，而无需人工进行参与，提高溯源系统的运行效率和数据的准确性。对于溯源系统的功能和页面美化程度上也可以进一步进行完善。

参考文献

[1] Mentzer J T, DeWitt W, Keebler J S, et al. Defining supply chain management[J]. Journal of Business logistics, 2001, 22(2): 1-25.

[2] Francisco K, Swanson D. The supply chain has no clothes: Technology adoption of blockchain for supply chain transparency[J]. Logistics, 2018, 2(1): 2-13.

[3] Aung M M, Chang Y S. Traceability in a food supply chain: Safety and quality perspectives[J]. Food control, 2014, 2014(39): 172-184.

[4] Liu L M, Qian H, Gao Y C, et al. Analysis and assessment of food traceability status in China[C]//Advanced Materials Research. Trans Tech Publications Ltd, 2012: 1353-1357.

[5] Wu H, Cao J, Yang Y, et al. Data management in supply chain using blockchain: Challenges and a case study[C]//2019 28th International Conference on Computer Communication and Networks (ICCCN). IEEE, 2019: 1-8.

[6] Galvez J F, Mejuto J C, Simal-Gandara J. Future challenges on the use of blockchain for food traceability analysis[J]. TrAC Trends in Analytical Chemistry, 2018, 2018(107): 222-232.

[7] Dujak D, Sajter D. Blockchain applications in supply chain[M]//SMART supply network. Springer, Cham, 2019.

[8] Saberi S, Kouhizadeh M, Sarkis J, et al. Blockchain technology and its relationships to sustainable supply chain management[J]. International Journal of Production Research, 2019, 57(7): 2117-2135.

[9] Dabbagh M , Sookhak M , Safa N S . The Evolution of Blockchain: A Bibliometric Study[J]. IEEE Access, 2019, 2019(7): 19212-19221.

[10] Lim M K, Li Y, Wang C, et al. A literature review of blockchain technology applications in supply chains: A comprehensive analysis of themes, methodologies and industries[J]. Computers & Industrial Engineering, 2021, 2021(154): 107133-107147.

[11] Cole R, Stevenson M, Aitken J. Blockchain technology: implications for operations and supply chain management[J]. Supply Chain Management: An International Journal, 2019, 24(4), 469-483.

[12] Hughes L, Dwivedi Y K, Misra S K, et al. Blockchain research, practice and policy: Applications, benefits, limitations, emerging research themes and research agenda[J]. International Journal of Information Management, 2019, 2019(49): 114-129.

[13] Wang X, He J, Xie Z, et al. ContractGuard: Defend ethereum smart contracts with embedded intrusion detection[J]. IEEE Transactions on Services Computing, 2019, 13(2): 314-328.

[14] Oliva G A, Hassan A E, Jiang Z M. An exploratory study of smart contracts in the Ethereum blockchain platform[J]. Empirical Software Engineering, 2020, 25(3): 1864-1904.

[15] 付瑶瑶, 李盛恩. 授权股份证明共识机制的改进方案[J]. 计算机工程与应用, 2020, 56(19): 48-54.

[16] Chen J, Lv Z, Song H. Design of personnel big data management system based on blockchain[J]. Future Generation Computer Systems, 2019, 2019(101): 1122-1129.

[17] Shahnaz A, Qamar U, Khalid A. Using blockchain for electronic health records[J]. IEEE Access, 2019, 2019(7): 147782-147795.

[18] Chen G, Xu B, Lu M, et al. Exploring blockchain technology and its potential applications for education[J]. Smart Learning Environments, 2018, 5(1): 1-10.

[19] Nanayakkara S, Rodrigo M N N, Perera S, et al. A methodology for selection of a Blockchain platform to develop an enterprise system[J]. Journal of Industrial Information Integration, 2021, 2021(23): 100215-100231.

[20] Panescu A T, Manta V. Smart contracts for research data rights management over the ethereum blockchain network[J]. Science & Technology Libraries, 2018, 37(3): 235-245.

[21] Androulaki E, Barger A, Bortnikov V, et al. Hyperledger fabric: a distributed operating system for

permissioned blockchains[C]//Proceedings of the thirteenth EuroSys conference. 2018: 1-15.
[22] Abou Jaoude J, Saade R G. Blockchain applications–usage in different domains[J]. IEEE Access, 2019, 1(7): 45360-45381.
[23] Efanov D, Roschin P. The all-pervasiveness of the blockchain technology[J]. Procedia computer science, 2018, 2018(123): 116-121.
[24] Gramoli V. From blockchain consensus back to Byzantine consensus[J]. Future Generation Computer Systems, 2020, 1(107): 760-769.
[25] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. Decentralized Business Review, 2008, 1(1): 21260-21269.
[26] Wang W, Hoang D T, Hu P, et al. A survey on consensus mechanisms and mining strategy management in blockchain networks[J]. IEEE Access, 2019, 2019(7): 22328-22370.
[27] Yazdinejad A, Srivastava G, Parizi R M, et al. Slpow: Secure and low latency proof of work protocol for blockchain in green iot networks[C]//2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring). IEEE, 2020: 1-5.
[28] 吴梦宇, 朱国胜, 吴善超. 基于工作量证明和权益证明改进的区块链共识机制[J]. 计算机应用, 2020, 40(08): 2274-2278.
[29] Kara M, Laouid A, AlShaikh M, et al. A compute and wait in pow (cw-pow) consensus algorithm for preserving energy consumption[J]. Applied Sciences, 2021, 11(15): 6750-6764.
[30] Lepore C, Ceria M, Visconti A, et al. A survey on blockchain consensus with a performance comparison of PoW, PoS and pure PoS[J]. Mathematics, 2020, 8(10): 1782-1808.
[31] 刘怡然, 柯俊明, 蒋瀚, 等. 基于沙普利值计算的区块链中 PoS 共识机制的改进[J]. 计算机研究与发展, 2018, 055(010): 2208-2218.
[32] Lee D R, Jang Y, Kim H. Poster: A proof-of-stake (PoS) blockchain protocol using fair and dynamic sharding management[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019: 2553-2555.
[33] Larimer D. Delegated proof-of-stake (dpos)[J]. Bitshare whitepaper, 2014, 1(1): 81-85.
[34] Tan C, Xiong L. DPoSB: Delegated Proof of Stake with node's behavior and Borda Count[C]//2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC). IEEE, 2020: 1429-1434.
[35] Ledwaba L P I, Hancke G P, Mitrokotsa A, et al. A delegated proof of proximity scheme for industrial internet of things consensus[C]//IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2020: 4441-4446.
[36] Kang J, Xiong Z, Niyato D, et al. Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory[J]. IEEE Transactions on Vehicular Technology, 2019, 68(3): 2906-2920.
[37] Min H. Blockchain technology for enhancing supply chain resilience[J]. Business Horizons, 2019, 62(1): 35-45.
[38] Singh A, Click K, Parizi R M, et al. Sidechain technologies in blockchain networks: An examination and state-of-the-art review[J]. Journal of Network and Computer Applications, 2020, 2020(149): 102471-102487.
[39] 孙知信, 张鑫, 相峰, 等. 区块链存储可扩展性研究进展[J]. 软件学报, 2021, 32(01): 1-20.
[40] Lewenberg Y, Sompolinsky Y, Zohar A. Inclusive block chain protocols[C]//International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2015: 528-547.
[41] Agarwal V, Pal S. Blockchain meets IoT: a scalable architecture for security and maintenance[C]//2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS). IEEE, 2020: 53-61.
[42] Zamani M, Movahedi M, Raykova M. Rapidchain: Scaling blockchain via full sharding[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 931-948.
[43] Li D, Dai J, Jiang R, et al. GAPG: a Heuristic Greedy Algorithm for Grouping Storage Scheme in

Blockchain[C]//2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops). IEEE, 2020: 91-95.
[44] Kumar R, Tripathi R. Implementation of distributed file storage and access framework using IPFS and blockchain[C]//2019 Fifth International Conference on Image Information Processing (ICIIP). IEEE, 2019: 246-251.
[45] 周家栋. 面向区块链的农产品溯源信息存储方法研究[D]. 安徽农业大学, 2021.
[46] Jin Z, Jian Z. Research on Application of Internet of Things Information Security Using Blockchain Technology[C]//2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS). IEEE, 2020: 402-404.
[47] Mettler M. Blockchain technology in healthcare: The revolution starts here[C]//2016 IEEE 18th international conference on e-health networking, applications and services (Healthcom). IEEE, 2016: 1-3.
[48] Monrat A A, Schelén O, Andersson K. A survey of blockchain from the perspectives of applications, challenges, and opportunities[J]. IEEE Access, 2019, 2019(7): 117134-117151.
[49] Siyal A A, Junejo A Z, Zawish M, et al. Applications of blockchain technology in medicine and healthcare: Challenges and future perspectives[J]. Cryptography, 2019, 3(1): 3-19.
[50] Leng K, Bi Y, Jing L, et al. Research on agricultural supply chain system with double chain architecture based on blockchain technology[J]. Future Generation Computer Systems, 2018, 1(86): 641-649.
[51] Manimaran P, Dhanalakshmi R. Blockchain-based smart contract for e-bidding system[C]//2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT). IEEE, 2019: 55-59.
[52] Vujičić D, Jagodić D, Ranđić S. Blockchain technology, bitcoin, and Ethereum: A brief overview[C]//2018 17th international symposium infoteh-jahorina (infoteh). IEEE, 2018: 1-6.
[53] Liu L M, Qian H, Gao Y C, et al. Analysis and assessment of food traceability status in China[C]//Advanced Materials Research. Trans Tech Publications Ltd, 2012: 1353-1357.
[54] 邵奇峰, 金澈清, 张召, 等. 区块链技术:架构及进展[J]. 计算机学报, 2018, 41(05): 969-988.
[55] Gai K, Wu Y, Zhu L, et al. Differential privacy-based blockchain for industrial internet-of-things[J]. IEEE Transactions on Industrial Informatics, 2019, 16(6): 4156-4165.
[56] 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.
[57] Androutsellis-Theotokis S., Spinellis D. A survey of peer-to-peer content distribution technologies[J]. ACM computing surveys (CSUR), 2004, 36(4): 335-371.
[58] 周文莉, 吴晓非. P2P 技术综述[J]. 计算机工程与设计, 2006, 2006(01): 76-79.
[59] Zhai S, Yang Y, Li J, et al. Research on the Application of Cryptography on the Blockchain[C]//Journal of Physics: Conference Series. IOP Publishing, 2019: 032077-032086.
[60] Wang X, Yu H. How to break MD5 and other hash functions[C]//Annual international conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 2005: 19-35.
[61] Fang W, Chen W, Zhang W, et al. Digital signature scheme for information non-repudiation in blockchain: a state of the art review[J]. EURASIP Journal on Wireless Communications and Networking, 2020, 1(1): 1-15.
[62] 杨洵, 王景中, 付杨, 等. 基于国密算法的区块链架构[J]. 计算机系统应用, 2020, 29(08): 16-23.
[63] Buterin V. A next-generation smart contract and decentralized application platform[J]. white paper, 2014, 3(37), 1-36.
[64] Szabo Nick. Formalizing and Securing Relationships on Public Networks[J]. First Monday, 1997, 2(9), 1-8.
[65] Macrinici D, Cartofeanu C, Gao S. Smart contract applications within blockchain technology: A systematic mapping study[J]. Telematics and Informatics, 2018, 35(8): 2337-2354.

附录 1 攻读硕士学位期间撰写的论文

[1]张斌，李大鹏，蒋锐，等。面向区块链溯源的链下扩展存储方案，计算机与现代化，已录用。

附录 2 攻读硕士学位期间申请的专利

[1] 李大鹏，张斌，朱天林. 基于区块链的商品供应链交易信息存储与追踪系统及方法，202110747080.7，2021.7。

致谢

一转眼，我已经到了研究生生活的最后阶段。回顾这二年半的生活，它将成为我人生中最重要的一笔财富。在这段生活中我有过迷茫、焦虑和欢笑，这些都教会了我如何更好的去成长，使我对自身有了更清晰的认识与定位。在此，我想向研究生生活中一直陪伴着我一起进步与成长的老师和同学们表示感谢，是你们的陪伴与鼓励让我在每一次遇到挫折时都能够有勇气去面对。

首先，诚挚地感谢我的导师李大鹏老师。本文从选题、研究、撰写到完成离不开李老师的悉心指导。每当我的论文遇到难题停滞不前时，李老师总会耐心地为我提供许多宝贵的意见和想法。李老师具有渊博的专业知识、严谨的科研态度、孜孜以求的工作作风，这些品质都深深地感染了我使我在工作和生活产生困难时具有砥砺前行的动力。在生活中，李老师平易近人，对学生关爱有加，这份师恩我将永记于心。在此向李老师致以真诚的谢意和崇高的敬意。

其次，感谢师兄师姐们在科研上对我的指点和工作方向上的引导，即使你们毕业后工作后很忙，在我遇到困难时，仍然会给予我相应的帮助。感谢教研室的兄弟姐妹们和朝夕相处的室友们，是你们的陪伴给我的研究生生活增添了不一样的色彩，优秀的你们也给我在学习和生活上为我树立了榜样。

最后，感谢我的父母与家人。感谢你们不辞辛苦地把我养大成人，给我提供了良好的学习机会。不管我做出什么决定你们都会给我最大的支持，你们的支持也是我前进路上最大的动力。祝愿我的家人们永远身体健康，幸福美满。