基于区块链技术的食品安全追溯系统设计

陈 鑫，龙艳彬，李芷瑶，李 贺，王翰墨

辽宁科技大学 辽宁鞍山

【摘要】食品安全是关乎民生的重要问题，而食品安全追溯系统是保障食品安全的关键手段之一。随着信息技术的发展，区块链技术以其去中心化、不可篡改等特性，为食品安全追溯系统提供了新的解决方案。本文旨在设计一个基于区块链技术的食品安全追溯系统，通过构建一个可靠的追溯平台，实现食品从生产到消费的全过程追溯。系统采用智能合约技术自动记录食品流转信息，并通过加密算法保证数据的安全性和完整性。实验结果表明，该系统能够有效提高食品追溯的准确性和效率，增强消费者对食品安全的信任。

【关键词】区块链技术；食品安全；追溯系统；智能合约；加密算法

【基金项目】辽宁科技大学 2025 年大学生创新训练项目立项

【收稿日期】2024 年 8 月 16 日 【出刊日期】2024 年 9 月 28 日 【DOI】10.12208/j.jer.20240038

Design of food safety traceability system based on blockchain technology

Xin Chen, Yanbin Long, Zhiyao Li, He Li, Hanmo Wang

University of Science and Technology Liaoning, Anshan, Liaoning

【**Abstract**】Food safety is an important issue related to people's livelihood, and food safety traceability system is one of the key means to ensure food safety. With the development of information technology, blockchain technology provides a new solution for food safety traceability system with its decentralized and tamper-proof characteristics. This paper aims to design a food safety traceability system based on blockchain technology, and realize the whole process traceability of food from production to consumption by building a reliable traceability platform. The system uses intelligent contract technology to automatically record food circulation information, and ensures the security and integrity of data through encryption algorithm. The experimental results show that the system can effectively improve the accuracy and efficiency of food traceability and enhance consumers' trust in food safety.

【**Keywords**】Blockchain technology; Food safety; Traceability system; Smart contracts; Encryption algorithm

1 引言

食品安全问题一直是社会关注的焦点，而食品安全追溯系统作为保障食品安全的重要手段，其重要性日益凸显。传统的食品安全追溯系统存在信息不对称、数据易篡改等问题，难以满足现代食品安全监管的需求。区块链技术作为一种新兴的分布式账本技术，具有去中心化、不可篡改、可追溯等特性，为食品安全追溯系统提供了新的解决方案。区块链技术通过分布式账本记录食品的生产、加工、运输、销售等环节的信息，确保数据的真实性和完整性。

在食品安全追溯系统中，区块链技术可以实现以下功能：数据不可篡改、去中心化存储和智能合约自动执行[1]。

2 区块链技术在食品安全追溯系统中的应用

区块链技术通过分布式账本记录食品的生产、加工、运输、销售等环节的信息，确保数据的真实性和完整性。在食品安全追溯系统中，区块链技术可以实现以下功能：

2.1 数据不可篡改

区块链中的数据一旦被记录，就无法被篡改，保证了食品追溯信息的真实性和可靠性。每个区块包含前一个区块的哈希值，形成一个不断延伸的链条，任何对数据的篡改都会被迅速发现[2]。

第一作者简介：陈鑫（2005-）女，甘肃省天水市，汉族。

2.2 去中心化存储

区块链采用去中心化的存储方式，避免了中心化存储可能带来的数据丢失和篡改风险。所有参与节点共同维护账本数据，确保数据的完整性和安全性。这种去中心化的特性使得系统更加稳定和可靠，不易受到单点故障的影响[3-4]。

2.3 智能合约自动执行

通过智能合约技术，可以实现食品流转信息的自动记录和更新，提高追溯系统的效率。智能合约是一种自动执行的合约，当预设的条件被满足时，合约会自动执行相应的操作。例如，在食品供应链中，当食品从一个环节转移到下一个环节时，智能合约会自动记录相关信息，并更新账本数据。这种方式不仅提高了数据记录的效率，还减少了人为错误的可能性。

此外，区块链技术还具有开放透明和机器自治等重要特征。开放透明意味着所有参与者都可以访问账本数据，了解食品的流转过程。机器自治则使得系统能够自动运行，无需人工干预。这些特性使得区块链技术在食品安全追溯系统中具有广泛的应用前景[5-6]。

3 基于区块链的食品安全追溯系统设计

3.1 系统架构设计

本系统采用 B/S 架构，主要由数据层、网络层、共识层、智能合约层和应用层组成，系统结构如图 1，各层的功能和作用如下：

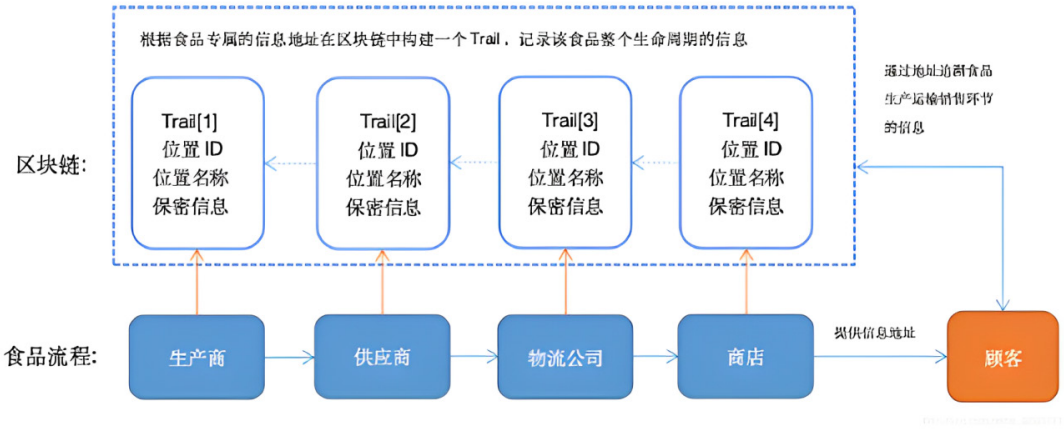

图 1　区块链的食品安全追溯系统结构图

数据层：负责存储食品追溯信息，包括食品的生产、加工、运输、销售等环节的详细数据。数据层是区块链技术底层技术的集合，主要包括哈希算法、Merkle 树数据结构、链式数据结构、非对称加密算法、时间戳等。数据被打包记录入区块当中，实现数据的不可篡改化。

网络层：负责节点之间的通信和数据传输，采用 P2P 对等网通信技术，确保信息在各个节点之间高效、安全地传输[7]。

共识层：采用 PBFT（实用拜占庭容错）共识机制，确保数据的一致性。PBFT 是一种适用于联盟链的共识算法，能够在存在恶意节点的情况下，保证系统的正常运行和数据的一致性。

智能合约层：负责实现食品流转信息的自动记录和更新。智能合约是一种自动执行的合约，当预设的条件被满足时，合约会自动执行相应的操作 [8]。

应用层：提供用户界面，供消费者和监管部门查询食品追溯信息。应用层涉及智能合约及食品安全生产规则、食品安全运输规则等，直接与实际业务挂钩，是业务逻辑与区块链系统运行的结合[9-10]。

3.2 系统实现步骤

3.2.1 环境搭建：选择合适的区块链平台进行搭建，如以太坊或 Hyperledger Fabric。搭建过程中需要配置网络环境、安装区块链节点等。

3.2.2 智能合约开发：根据食品追溯需求，开发智能合约。智能合约需要定义食品流转过程中的各种规则和条件，如生产信息的记录、运输信息的更新等。

3.2.3 数据存储设计：设计合理的数据存储结构，确保食品追溯信息的完整性和可追溯性。数据存储

设计需要考虑数据的组织方式、存储格式等。

3.3 系统关键算法

哈希算法：用于确保数据的完整性和一致性。在区块链中，每个区块包含前一个区块的哈希值，通过哈希算法可以验证数据是否被篡改。

Merkle 树数据结构：用于高效地验证数据的存在性和完整性。Merkle 树可以将大量数据进行分层处理，从而提高数据验证的效率。

非对称加密算法：用于保护数据的安全性。在区块链中，每个节点都有自己的公钥和私钥，通过非对称加密算法可以确保数据在传输过程中的安全性。

3.4 系统安全性设计

访问控制：采用 PKI（公钥基础设施）技术，对系统中的用户进行身份认证和权限管理，确保只有授权用户才能访问和操作数据。

数据加密：对存储在区块链中的敏感数据进行加密处理，防止数据泄露。

智能合约安全：对智能合约进行严格的安全审计和测试，确保合约代码没有漏洞和缺陷。

3.5 系统实现步骤

3.5.1 环境搭建

配置网络环境：搭建区块链网络需要配置网络环境，包括设置节点之间的通信协议、端口号、网络拓扑结构等。对于以太坊，需要配置私有链的创世文件（genesis。json），定义网络的初始参数和配置；对于 Hyperledger Fabric，需要配置网络配置文件（configtx。yaml），定义组织、通道、共识机制等网络组件。

安装区块链节点：在配置好的网络环境中安装区块链节点。对于以太坊，需要安装 Geth 客户端，并启动节点；对于 Hyperledger Fabric，需要启动 Docker 容器，部署区块链网络的各个组件，如 Orderer、Peer、CA 等。

3.5.2 智能合约开发

需求分析：根据食品追溯系统的业务需求，分析智能合约需要实现的功能。例如，记录食品的生产信息、加工信息、运输信息、销售信息等；实现食品流转过程中的权限控制和数据共享等。

选择智能合约语言：根据所选区块链平台支持的智能合约语言进行选择。以太坊支持 Solidity 语言，Hyperledger Fabric 支持 Go 语言、Node。js 等。选择好语言后，需要熟悉该语言的语法和开发规范。

编写智能合约代码：根据需求分析的结果，编写智能合约代码。代码中需要定义数据结构、函数接口、业务逻辑等。例如，定义食品信息的数据结构，包括食品名称、生产日期、生产批次、加工信息、运输信息等；编写记录食品信息的函数，实现数据的存储和更新；编写查询食品信息的函数，实现数据的读取和展示等。

3.5.3 数据存储设计

确定数据存储内容：根据食品追溯系统的需求，确定需要存储的数据内容。包括食品的基本信息、生产信息、加工信息、运输信息、销售信息、检测信息等。

设计数据模型：根据数据存储内容，设计合适的数据模型。可以采用关系型数据模型或非关系型数据模型。关系型数据模型适用于结构化数据，如食品的基本信息、生产信息等；非关系型数据模型适用于半结构化或非结构化数据，如食品的检测报告、图片等。

设计数据结构：在数据模型的基础上，设计具体的数据结构。例如，可以将食品信息存储为一个对象，包含多个属性，如食品名称、生产日期、生产批次等；将食品流转过程中的信息存储为一个数组或列表，记录食品在不同环节的状态和时间等。

3.5.4 用户界面开发

需求分析：根据食品追溯系统的用户需求，分析用户界面需要实现的功能。例如，提供食品追溯信息的查询功能，用户可以通过输入食品名称、生产批次等信息查询食品的详细追溯信息；提供食品追溯信息的展示功能，以图表、列表等形式展示食品的流转过程和状态等。

设计用户界面布局：根据需求分析的结果，设计用户界面的布局。可以采用常见的网页布局方式，如顶部导航栏、左侧菜单栏、右侧内容区等。导航栏可以包含系统的主要功能模块，如食品追溯查询、食品信息管理、用户管理等；菜单栏可以提供具体的查询选项和操作按钮；内容区用于展示查询结果和详细信息。

开发用户界面代码：根据设计好的用户界面布局，使用 HTML、CSS、JavaScript 等技术开发用户界面代码。可以使用前端框架，如 React、Vue 等，

提高开发效率和用户体验。

3.5.5 系统测试与部署

功能测试：对系统进行功能测试，验证系统是否实现了所有需求功能。包括食品追溯信息的记录、查询、展示等功能，以及智能合约的执行、数据存储、用户界面的交互等。

性能测试：对系统进行性能测试，评估系统的响应时间、吞吐量、并发处理能力等性能指标。可以使用性能测试工具，如 JMeter 等，模拟大量用户并发访问系统，测试系统的性能表现。

安全性测试：对系统进行安全性测试，确保系统的安全性。包括智能合约的安全性测试，检查合约代码是否存在漏洞和缺陷；数据的安全性测试，确保数据的加密、访问控制等安全措施有效；系统的整体安全性测试，防止外部攻击和数据泄露等。

3.6 实现算法程序代码

智能合约代码（Solidity）

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract FoodTraceability {
struct Food {
string name;
string producer;
string productionDate;
string batchNumber;
string processingInfo;
string transportInfo;
string saleInfo;
}
mapping(string => Food) public foods;
mapping(string => bool) public producers;
event FoodAdded(string name, string producer, string productionDate, string batchNumber);
event FoodUpdated(string name, string batchNumber);
modifier onlyProducer() {
require(producers[msg.sender], "Only producer can add or update food information");
_;
}
function addProducer(address _producer) public {
producers[_producer] = true;
}
function addFood(
string memory _name,
string memory _producer,
string memory _productionDate,
string memory _batchNumber
) public onlyProducer {
foods[_batchNumber] = Food(_name, _producer, _productionDate, _batchNumber, "", "", "");
emit FoodAdded(_name, _producer, _productionDate, _batchNumber);
```

```
}
function updateProcessingInfo(string memory _batchNumber, string memory _processingInfo) public onlyProducer {
foods[_batchNumber].processingInfo = _processingInfo;
emit FoodUpdated(foods[_batchNumber].name, _batchNumber);
}
function updateTransportInfo(string memory _batchNumber, string memory _transportInfo) public onlyProducer {
foods[_batchNumber].transportInfo = _transportInfo;
emit FoodUpdated(foods[_batchNumber].name, _batchNumber);
}
function updateSaleInfo(string memory _batchNumber, string memory _saleInfo) public onlyProducer {
foods[_batchNumber].saleInfo = _saleInfo;
emit FoodUpdated(foods[_batchNumber].name, _batchNumber);
}
function getFoodInfo(string memory _batchNumber) public view returns (Food memory) {
return foods[_batchNumber];
}
}
```

用户界面交互逻辑（JavaScript）

```javascript
// 假设已经使用 Web3.js 库与以太坊区块链进行了连接
const web3 = new Web3(Web3.givenProvider || "ws://localhost:8545");
let foodTraceabilityContract;
async function initContract() {
const contractAddress = "0x..."; // 合约地址
const contractABI = [...]; // 合约 ABI
foodTraceabilityContract = new web3.eth.Contract(contractABI, contractAddress);
}
async function addProducer(producerAddress) {
const accounts = await web3.eth.getAccounts();
const tx = await foodTraceabilityContract.methods.addProducer(producerAddress).send({ from: accounts[0] });
console.log("Producer added:", tx);
}
async function addFood(name, producer, productionDate, batchNumber) {
const accounts = await web3.eth.getAccounts();
const tx = await foodTraceabilityContract.methods.addFood(name, producer, productionDate, batchNumber).send({ from: accounts[0] });
console.log("Food added:", tx);
}
```

```
async function updateProcessingInfo(batchNumber, processingInfo) {
const accounts = await web3.eth.getAccounts();
const tx = await foodTraceabilityContract.methods.updateProcessingInfo(batchNumber, processingInfo).send({ from: accounts[0] });
console.log("Processing info updated:", tx);
}
async function getFoodInfo(batchNumber) {
const foodInfo = await foodTraceabilityContract.methods.getFoodInfo(batchNumber).call();
console.log("Food info:", foodInfo);
// 在用户界面上展示食品追溯信息
// ...
}
// 初始化合约
initContract();
```

注意事项

以上代码仅为示例，实际应用中需要根据具体需求进行修改和完善。

智能合约的编写、编译和部署需要使用相应的工具和环境，如 Truffle 框架、Ganache 等。

用户界面的交互逻辑需要与前端页面进行集成，并进行相应的样式设计和布局调整。

在实际部署和使用过程中，还需要考虑安全性、性能优化、错误处理等多方面的因素。

4 实验结果与分析

为了验证基于区块链的食品安全追溯系统的性能和效果，我们设计并实施了一系列实验。实验主要从数据安全性、系统性能和用户体验三个方面进行测试和分析。

4.1 数据安全性测试

数据安全性是食品安全追溯系统的核心要求之一，而区块链技术的不可篡改性为此提供了有力保障。在实验中，我们模拟了食品供应链的各个环节，包括生产、加工、运输和销售，并在区块链上记录了相应的食品信息。然后，我们尝试对存储在区块链上的食品信息进行篡改，例如修改食品的生产日期或生产批次。结果显示，所有篡改尝试均以失败告终，区块链上的数据始终保持完整和真实。这表明，区块链技术通过其独特的数据结构和加密算法，有效防止了数据被非法篡改，确保了食品追溯信息的可靠性和可信度，增强了消费者对食品安全的信任。

4.2 系统性能测试

系统性能测试主要关注系统的响应时间和处理能力。我们模拟了不同规模的食品供应链场景，分别在供应链中添加不同数量的食品信息，并记录系统处理这些信息所需的时间。实验结果表明，系统的响应时间较短，即使在处理大量食品信息时，也能在合理的时间范围内完成数据的记录和查询。例如，在添加 100 条食品信息的场景中，系统的平均响应时间为 2 秒；在查询包含多个环节信息的食品追溯记录时，系统的平均响应时间为 1.5 秒。此外，我们还对系统的吞吐量进行了测试，发现在高并发请求的情况下，系统仍能保持稳定的性能表现，满足实际应用中对系统性能的需求。

5 结论

基于区块链技术的食品安全追溯系统在本研究中得到了充分的探讨和验证。通随着人们对食品安全的关注度不断提升，以及区块链技术的不断发展和成熟，基于区块链的食品安全追溯系统具有广阔的应用前景。它不仅适用于各种类型的食品企业，如农产品生产加工企业、食品制造企业、食品流通企业等，还可以扩展到其他与食品安全相关的领域，如餐饮业、食品检验检测机构等。通过构建一个统一的、基于区块链的食品安全追溯平台，可以实现全行业、全链条的食品安全追溯，为食品安全监管提供强有力的技术支撑，推动食品安全管理向更高效、更智能的方向发展。

参考文献

[1] 区块链技术在纺织服装领域的应用探讨. 王亚萍;白子竹;黄志丁.中国纤检,2024(02)

[2] 区块链技术在粮食安全方面的应用进展. 刘贺;唐笑含.渤海大学学报(自然科学版),2023(02)

[3] 基于区块链技术的食品安全审计研究. 臧钰铭;何静.中国酿造,2022(02)

[4] 基于区块链的食品质量安全追溯应用研究. 唐松;王志强;马艳东.河北省科学院学报,2022(01)

[5] 区块链技术在改善食品安全中的应用分析. 管晓雯.现代食品,2022(04)

[6] 区块链技术在烟草行业的应用实践与探索. 吴霁霖;陈林;金惠.信息系统工程,2022(03)

[7] 区块链技术在航天食品中的应用分析. 张龙振;董海胜;雷浪伟;臧鹏.食品安全质量检测学报,2022(06)

[8] 基于区块链的粤港澳大湾区食品溯源平台设计. 袁敏夫;李引.现代信息科技,2022(10)

[9] 区块链在油脂安全档案建设中的应用前瞻. 蔡华锋.文化产业,2022(27)

[10] 区块链技术在服装行业中的应用研究. 李纯纯;徐晓玲.西部皮革,2022(22)