**课 程 报 告**

|  |  |
| --- | --- |
| **课程名称：** | **《区块链技术与应用》** |
| **学生姓名：** | **石夏源** |
| **学生学号：** | **202230091036** |
| **学生专业：** | **软件工程** |
| **授课教师：** | **许可** |
| **开课学期：** | **2024-2025-1** |

**华南理工大学软件学院**

**2024年9月**

|  |  |  |  |
| --- | --- | --- | --- |
| 评分 | 探索实践报告评分 | 学习报告评分 | 总分 |
|  | /40 | /60 | /100 |

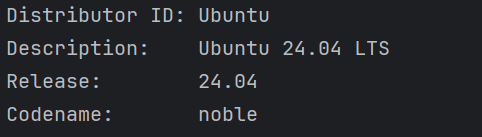
(\*\*\*\*\*\*注意：本页内容学生请勿改动\*\*\*\*\*\*\*\*)

1. **探索实践报告（ 40分）**

**Understanding Blockchain Cryptography**

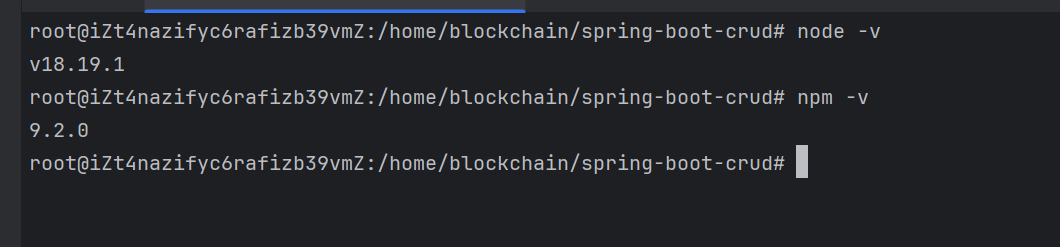
**Install prerequisites（10分）**

1. 你实践用的操作系统是 Ubuntu 24.04 LTS 。（1分）



1. 所安装的软件版本是： node版本V18.19.1 npm版本9.2.0

，并提供node -v, npm -v, 以及VS Code的运行截图（9分）



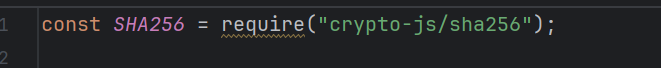
**Create the development environment（10分）**

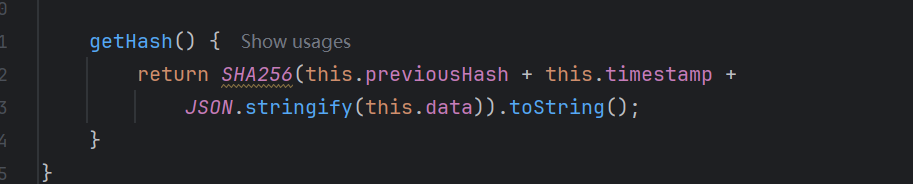
1. 安装Crypto-JS的命令是： npm install crypto-js ，给出执行结果截图。（4分）
2. 简述Crypto-JS的作用和使用方法。（6分）

crypto模块的目的是为了提供通用的加密和哈希算法。crypto-js 支持多种对称加密算法（如 AES、DES）和散列函数（如 MD5、SHA1、SHA256），以及消息认证码（MAC）算法。它提供了一套完整的加密工具，能够满足不同的安全需求。

在区块链中，每个区块都包含一个唯一的散列值，这个散列值是通过对区块的内容进行散列产生的，本实验中包括其data、时间戳和前一个区块的散列。这个独特的哈希值可以作为区块的指纹，使其内容易于识别和验证。

使用示例（使用sha256加密算法）:





哈希结果:



**Create a basic hash chain（10分）**

1. IBM教程使用以下代码定义单个区块及区块链。当然，现实情况要复杂得多，但基本思想是相通的。在计算区块的哈希值时，为什么要包含前一个区块的哈希值？请详细解释。（5分）

const SHA256 = require("crypto-js/sha256");

class Block {

    constructor(data, previousHash) {

        this.data = data;

        this.timestamp = Date.now();

        this.previousHash = previousHash;

        this.hash = this.getHash();

    }

    getHash() {

        return SHA256(this.previousHash + this.timestamp +

            JSON.stringify(this.data)).toString();

    }

}

class Blockchain {

    constructor() {

        this.blockchain = [new Block("Genesis Block", '')];

    }

    getLastBlock() {

        return this.blockchain[this.blockchain.length - 1];

    }

    createBlock(data) {

        this.blockchain.push(new Block(data, this.getLastBlock().hash));

    }

}

通过包含其哪一个区块的哈希值，让块之间形成链式结构。通过对区块头进行哈希运算，可以得到一个哈希值，这个哈希值作为区块的唯一标识符，可以确保数据的完整性。如果区块中的数据被算改，那么计算得到的哈希值也会发生变化，从而使得区块无效。因此，通过对比哈希值，可以验证数据是否被算改。这种链式结构具备发散传导性，越往历史以前的篡改，越容易导致大面积的影响。

1. 使用以下代码对区块链进行测试，请在划线处补充上你喜欢的任何测试数据，要求以自己的名字开头（形式不限）。（2分）

myChain = new Blockchain();

console.log('\n-----\nNew blockchain created');

console.log(myChain);

myChain.createBlock("first set of transaction data");

console.log('\n-----\nAdded a block');

console.log(myChain);

**myChain.createBlock(" ");**

console.log('\n-----\nAdded another block');

console.log(myChain);

shixiayuan\_Chain = new Blockchain();

console.log('\n-----\nNew blockchain created');

console.log(shixiayuan\_Chain);

shixiayuan\_Chain.createBlock("first set of transaction data");

console.log('\n-----\nAdded a block');

console.log(shixiayuan\_Chain);

shixiayuan\_Chain.createBlock("石夏源\_block");

console.log('\n-----\nAdded another block');

console.log(shixiayuan\_Chain);

1. 运行程序的命令是node MyChain.js ，请给出首次运行的结果（截图或文本）。（3分）

**Help prevent modification of the hash chain（10分）**

1. 在Blockchain类中添加以下代码检查区块链的有效性，请补充划线处代码。（2分）

    isBlockchainValid() {

        for (let i = 1; i < this.blockchain.length; i++) {

            let currentBlock = this.blockchain[i];

            let previousBlock = this.blockchain[i - 1];

**if ( ) {**

                return false;

            }

        }

        return true;

    }

isBlockchainValid() {

        for (let i=1; i<this.blockchain.length; i++) {

        let currentBlock = this.blockchain[i];

        let previousBlock = this.blockchain[i-1];

        if ((currentBlock.previousHash !== previousBlock.hash) ||

        (currentBlock.hash !== currentBlock.getHash())) {

        return false;

        }

        }

        return true;

        }

1. 在测试代码后面添加以下内容，请在划线处补充上你喜欢的任何测试数据，要求以自己的名字开头（形式不限），且不能与前面相同。（2分）

console.log('Is the chain valid: ' + myChain.isBlockchainValid());

**myChain.blockchain[1].data = ' ';**

console.log('Modified transaction data in a block');

console.log('Is the chain valid: ' + myChain.isBlockchainValid());

shixiayuan\_Chain = new Blockchain();

console.log('\n-----\nNew blockchain created');

console.log(shixiayuan\_Chain);

shixiayuan\_Chain.createBlock("first set of transaction data");

console.log('\n-----\nAdded a block');

console.log(shixiayuan\_Chain);

shixiayuan\_Chain.createBlock("石夏源\_block");

console.log('\n-----\nAdded another block');

console.log(shixiayuan\_Chain);

console.log('Is the chain valid: ' + shixiayuan\_Chain.isBlockchainValid());

shixiayuan\_Chain.blockchain[1].data = ' 石夏源的DATA';

console.log('Modified transaction data in a block');

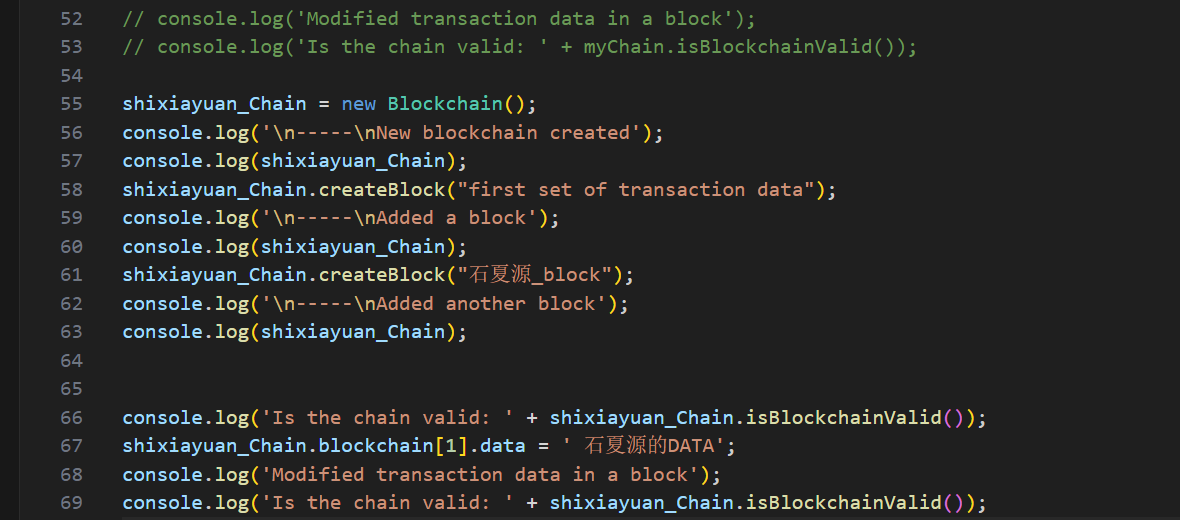
console.log('Is the chain valid: ' + shixiayuan\_Chain.isBlockchainValid());

1. 再次运行程序，请给出有效性检查的输出结果截图（要包含你的测试代码）。数据一旦加入区块链就能轻松防止篡改吗？为什么？（6分）

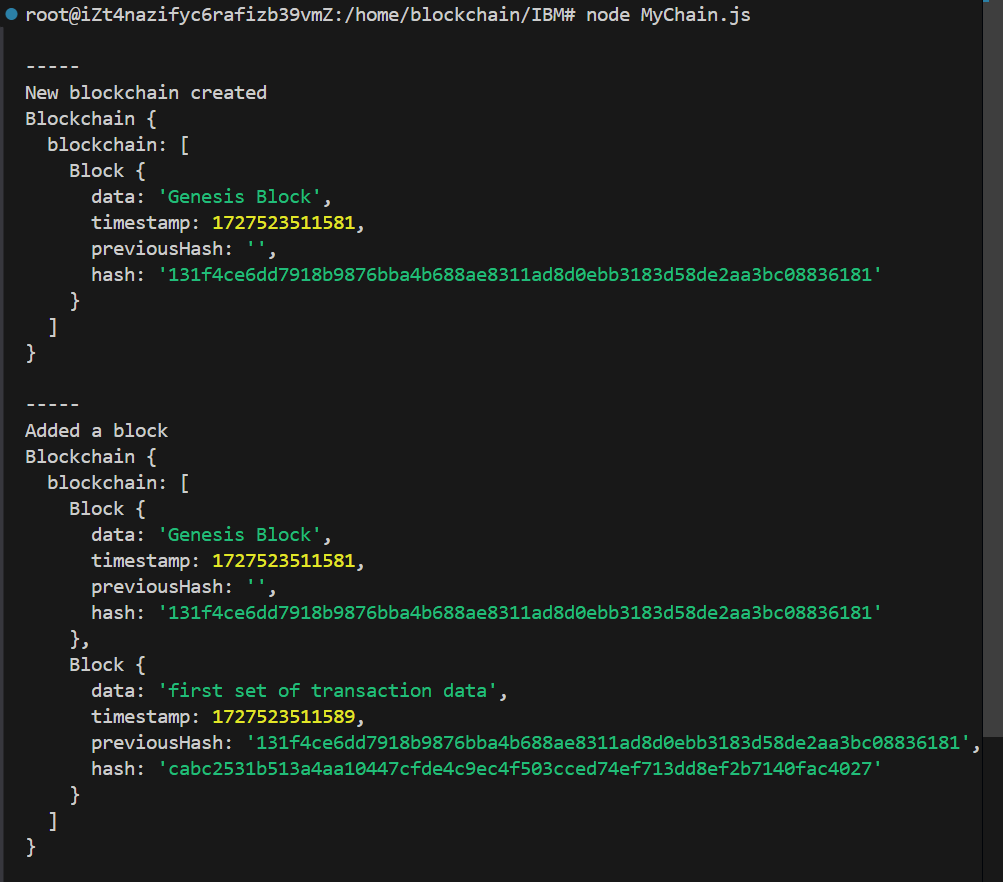
由于区块链的特性，一旦数据的哈希值被存储在区块链上，就无法篡改或删除。

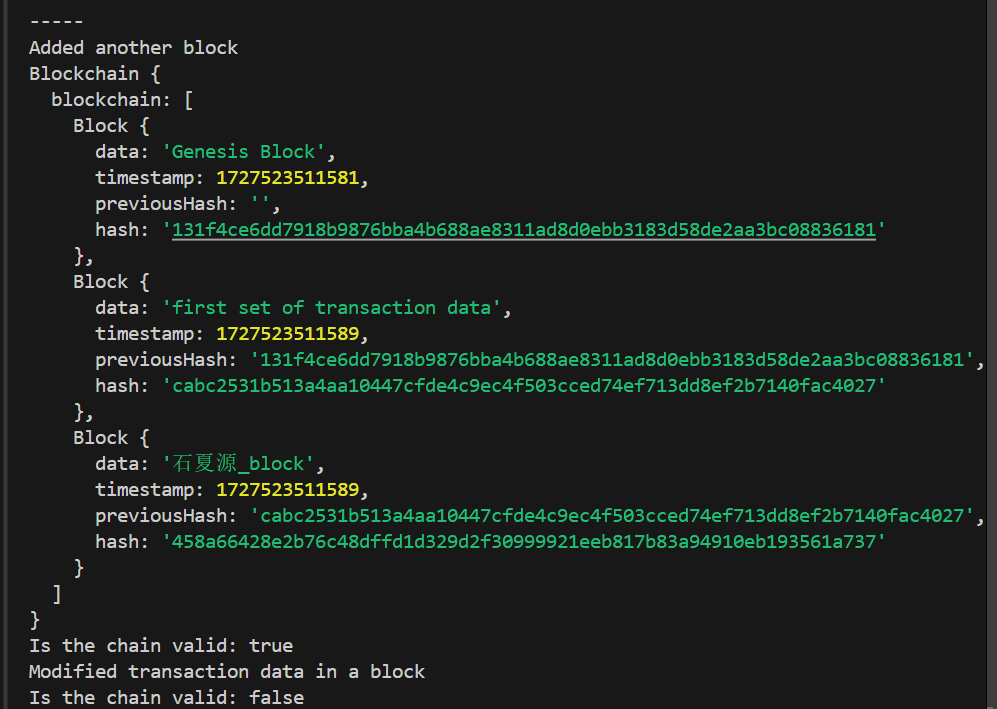
展开说：在区块链中，每个新区块都包含上一个区块经过科学方法算出来的数据指纹——哈希值。这个值让一个个区块之间形成了有着严格顺序关系的链条结构，一旦某个区块中的任何数据被篡改，该区块在下一个区块头部的数据指纹——哈希值就会变动，之后就无法衔接上来，也就不会被认可。

测试代码：



运行结果如下：





1. **课程学习报告（ 60分）**

**Understanding Fabric and FISCO BCOS**

1. **请结合课内外所学以及个人理解，举一个实际应用例子，用自己的语言详细阐述Fabric的交易流程和FISCO BCOS的交易流程，并对比这两种交易流程，给出自己的见解。（30分）**

**FABRIC**

1. **客户端 A 发起一笔交易**
2. **背书节点验证签名并执行交易**
3. **检查提案响应**
4. **客户端将背书结果封装进交易**
5. **验证和提交交易**
6. **账本更新**

**FISCO BCOS**

1. **交易生成**
2. **交易池(验证)**
3. **交易广播**
4. **交易打包**
5. **交易执行**
6. **交易共识**
7. **交易落盘**

Hyperledger Fabric是一个模块化、可配置的区块链框架，提供了多种可插拔的组件，包括共识算法、加密算法、智能合约等。它支持多通道、多子网和多链等架构，支持与其他区块链平台的互操作性，如以太坊、比特币等，其可以与不同的区块链网络进行集成和交互，能够满足不同业务场景的需求。

FISCO BCOS则是一个专门为金融行业设计的区块链平台,与其他区块链平台的兼容性相对较弱。它的架构比较集中，通常采用单链单侧的设计。这种架构简单明了，容易部署和管理，但在扩展性和灵活性方面可能存在一定的限制。

Fabric使用模块化的共识机制，常见的是Kafka/Zookeeper或Raft，共识主要在排序服务中进行。FISCO BCOS使用PBFT（Practical Byzantine Fault Tolerance）共识算法，所有共识节点参与共识。

Fabric采用“先背书，后排序，再验证”的流程。交易先在背书节点模拟执行，然后排序，最后在所有节点验证和提交。FISCO BCOS采用“先共识，后打包，再验证”的流程，交易先在共识节点达成共识，然后打包成区块，最后在所有节点验证和提交。

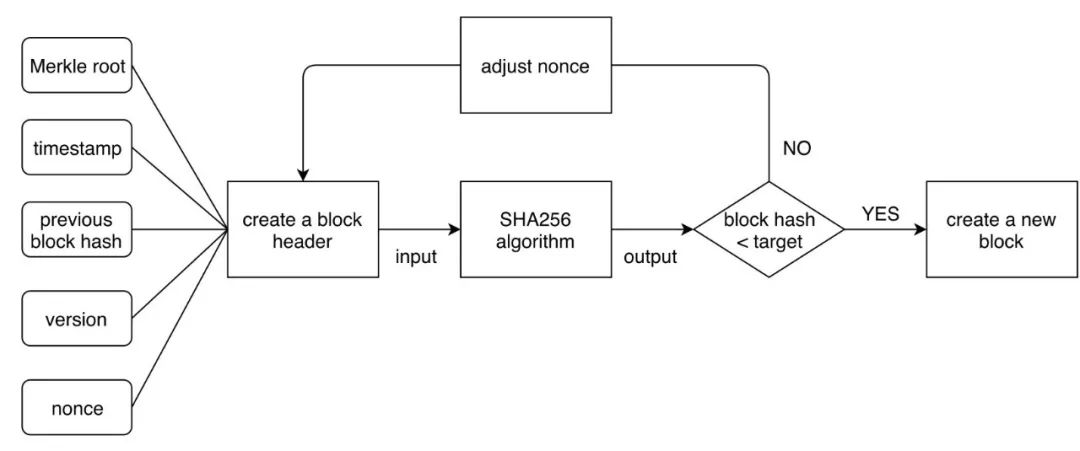
Fabric智能合约在背书节点上执行，结果在排序后广播。FISCO BCOS智能合约在共识节点上执行，结果在共识后广播。

**（2）请结合课内外所学以及个人理解，用自己的语言详细阐述PoW共识算法过程和PBFT共识算法过程，思考为什么PoW共识安全性必须满足51%诚实节点？为什么PBFT共识安全性必须满足2f+1个诚实节点（假设有f个恶意节点，所有节点个数为3f+1）？（30分）**

PoW (工作量证明) 和 PBFT (拜占庭容错协议) 是两种主要的区块链共识算法，各自适用于不同的区块链网络环境。PoW 侧重于利用计算力来确保网络安全，而 PBFT 则通过复制和消息交换来防止拜占庭故障。

**PoW 共识算法过程**

在POW协议中，每个节点竞争着找到一个nonce的值来生成满足特定条件的哈希,找到这样的一个nonce值的难度可以根据哈希值的条件来计算。当找到这样的一个nonce值时，将生成一个块并向P2P网络广播。根据协议的不同类型，对等节点通常会接受最长的链或总难度最大的链来不断地扩展区块链，POW利用这一机制来确定哪个节点有资格打包一个块，这个过程也称为挖矿。



**PBFT共识**主要包括Pre-prepare、Prepare和Commit三个阶段：

1. Pre-prepare：负责执行区块，产生签名包，并将签名包广播给所有共识节点；
2. Prepare：负责收集签名包，某节点收集满2\*f+1的签名包后，表明自身达到可以提交区块的状态，开始广播Commit包；
3. Commit：负责收集Commit包，某节点收集满2\*f+1的Commit包后，直接将本地缓存的最新区块提交到数据库。

**为什么PoW需要51%的诚实节点？**

1. 计算能力：控制超过一半的计算能力可以使攻击者优先完成哈希计算，潜在地改变区块链的顺序或内容。
2. 双花攻击：如果攻击者拥有主导权，可以在不同的地方重复使用同样的资金。
3. 网络分叉：控制大部分算力的攻击者能够形成最长链，使其成为公认的有效链。

**为什么PBFT需要2f+1个诚实节点？**

主要是考虑到Primary节点的作恶问题。若Primary节点是诚实的，f+1就足够了。

考虑到Primary节点可能会作恶，必须要收到2f+1条消息。假如只要求收到2f条消息： Primary节点是作恶的，另外还有f-1个节点也是作恶的（假设这f个恶意节点构成集群B），剩下的2f+1个节点都是正确的。那么Primary节点坑会给f个正确节点发送a=1的PrePrepare消息（假设这f个正确节点构成集群C），给另外f+1个正确节点发送a=2的PrePrepare消息（假设这f个正确节点构成集群D）。同时，另外集群B中的恶意节点也会给集群C中的正确节点发送a=1的Prepare消息，给集群D中的f+1个正确节点发送a=2的Prepare消息。集群C和集群D中的节点也都会向彼此发送Prepare消息。此时，对于集群C中的节点，其可能收到1个a=1的PrePrepare消息，f-1个a=1的Prepare消息（从集群B收到），和f个a=1的Prepare消息（从集群C中收到，包括自己的）。总计2f条消息，从而得到a=1的结论。另一方面，对于集群D中的节点，其可能收到1个a=2的PrePrepare消息，f-1个a=2的Prepare消息（从集群B收到），和f+1个a=2的Prepare消息（从集群D中收到，包括自己的）。总计2f+1条消息，从而得到a=2的结论。也即集群C和集群D中的节点得出了不一样的结论，没能达成共识。