**实 验 报 告**



**课程名称 密码学基础**

**学 院 计算机学院**

**专 业 保密管理**

**姓 名 王君可**

**学 号 17300240009**

**开 课 时 间 2019 至 2020 学年第 二 学期**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | 实验项目  名 称 | **Blockchain** | 成绩 |  |   **一、实验目的**  1. Understanding basic concepts in blockchain.理解区块链的基本概念  2. Implementing a blockchain web application. 实现一个区块链应用  **二、实验内容**  The goal is to build an application that allows users to share information by posting. Since the content will be stored on the blockchain, it will be immutable and permanent. Users will interact with the application through a simple web interface.    **三、实验步骤**  **1. Store transactions into blocks**  A post(transaction) is a message that’s posted by any user. Each post will consist of three elements: Author、Content、Timestamp. Storing transactions (and any data) in blockchain in JSON format. The transactions are packed into blocks. A block can contain one or many transactions. The blocks containing the transactions are generated frequently and added to the blockchain.  **2. Add digital fingerprints to the blocks**  Store the hash of the block in a field inside our Block object, and it will act like a digital fingerprint of transactions contained in it.  **3. Chain the blocks**  The blockchain is supposed to be a collection of blocks. We need a way to make sure that any change in the previous blocks invalidates the entire chain. The Bitcoin way to do this is to create dependency among consecutive blocks by chaining them with the hash of the block immediately previous to them. By chaining here, we mean to include the hash of the previous block in the current block in a field called . **4. Implement proof of work (POW) algorithm**  To make the task of calculating the hash difficult and random, we add a constraint that our hash should start with n leading zeroes where n can be any positive integer.  **5. Add blocks to the chain** To add a block to the chain, we’ll first have to verify that:  The data has not been tampered with (the proof of work provided is correct). The order of transactions is preserved (the field of the block to be added points to the hash of the latest block in our chain).  **Mining**  The transactions will be initially stored as a pool of unconfirmed transactions. The process of putting the unconfirmed transactions in a block and computing proof of work is known as the mining of blocks. Once the nonce satisfying our constraints is figured out, we can say that a block has been mined and it can be put into the blockchain.  **6. Establish consensus and decentralization**  We need the data to be distributed, we need multiple nodes maintaining the blockchain. So, to transition from a single node to a peer-to-peer network, let’s first create a mechanism to let a new node become aware of other peers in the network.  **Consensus**  Due to intentional manipulation or unintentional reasons (like network latency), the copy of chains of a few nodes can differ. In that case, the nodes need to agree upon some version of the chain to maintain the integrity of the entire system. In other words, we need to achieve consensus.  A simple consensus algorithm could be to agree upon the longest valid chain when the chains of different participating nodes in the network appear to diverge. The rationale behind this approach is that the longest chain is a good estimate of the most amount of work done.  **7. Create interfaces**  Using a Python microframework Flask to create a REST API that interacts with and invokes various operations in blockchain node.  **8. build and run the application**  We’ve used **Jinja2** templating to render the web pages and some **CSS** to make things look nice.  **9. Add Merkle tree support**  Instead of computing the hash of the whole block, every individual transaction in the block are hashed and then stored to form a **Merkle tree**. The root of the tree represents the hash of the block.    Merkle root的实现是基于double-SHA256算法的，也就是对于交易A，用SHA256(SHA256(交易A))来表示merkle树中A节点；然后对于交易A和B，分别计算节点值后拼接并通过double-SHA256计算它们的父节点，最终得到merkle root。  **10. Add signature support**  Now in this application, anyone can change any name and post any content. Also, the post is susceptible to tampering while submitting the transaction to the blockchain network. One way to solve this is by creating **user accounts** using public key cryptography. Every new user needs a public key (analogous to username) and a private key to be able to post in the application. The keys are used to create and verify the digital signature.    实验中我才用了RSA签名算法，先对交易内容计算hash，然后对hash用私钥进行签名；最终转换为base64签名的字符串形式。验签的过程则是将签名转为base64解码后，先进行hash计算，然后用公钥进行验签。  **四、实验结果及分析**  1. 区块链应用界面：    2. 挖矿成功界面：    3. 获取区块链内容（其中包括对交易的**数字签名**）：    运行结果详见result.mp4。  **五、实验总结**  1. 每一个区块中实际都记录着一段时间内的所有交易，同时这些区块有顺序，也就是每个区块都有唯一的父区块，多个区块连接形成了区块链，连接的机制是每个区块的区块头的previous\_hash字段记录着父区块的区块头的hash值。  2. 区块内的字段包括：index标记区块的序号、transactions记录区块的交易、timestamp记录区块生成的时间戳，merkle\_hash记录merkle根，最后是nonce随机数用于工作量证明。  3. 区块链的第一个区块是创世区块，之后再添加区块的时候需要完成两个确认：确认该区块的previous\_hash字段确实是前一区块的hash、确认该区块的nonce满足工作量证明的难度要求。  4. 工作量证明即proof\_of\_work的过程就是将nonce从0开始逐渐增大，最终使得hash满足难度值要求，也就是hash的前difficulty位是0。这个difficulty的值应当是被过一段时间后会被更新。  5. 挖矿的过程其实需要完成以下步骤：获取当前未被确认的交易、验证交易记录的正确性包括签名是用付款人的私钥以及付款人的付款金额小于钱包金额（在添加新交易的时候付款人用私钥对交易内容签名，然后矿工挖矿的时候进行确认），确认无误后通过工作量证明算法计算得到本区块的nonce，最终将区块加入区块链。  6. 区块链的维护并不是一个线性，而是树状，所以区块链运行的一个重要基础是共识机制，也就是每个参与者都承认当前的最长的区块链是主链。具体的过程就是，参与者获得每个节点当前挖矿所得到的区块链，验证其有效性之后选取最长链作为主链。验证的过程就是逐个检验区块链的每个区块都满足工作量证明，并previous\_hash字段是前一区块的hash。  7. 交易记录的数据传输都是用json，原因是json是一种轻量的数据交换格式，易于人阅读和编写。  8. 学习了使用flash框架搭建web应用。  9. 签名算法使用python的Crypto中封装的签名算法，为每个参与者生成自己的公钥和私钥，然后对交易记录计算得hash之后对其用私钥签名，最后挖矿的时候可以用公钥进行验证。 |