|  |  |  |
| --- | --- | --- |
| **本科生毕业论文（设计）** | | |
|  | 题 目: | 基于区块链技术的数字版权管理系统 |
|  |  |  |
|  | 姓 名: | 朱新凡 |
|  | 学 院: | 信息科学技术学院 |
|  | 专 业: | 计算机科学与技术 |
|  | 班 级: | 计科151 |
|  | 学 号: | 19215108 |
|  | 指导教师: | 赵青松 职称: 讲师 |
| 2019 年5月20 日 | | |
| 南京农业大学教务处制 | | |

**目 录**

[摘要 1](#_Toc10129672)

[关键词 1](#_Toc10129673)

[Abstracts: 1](#_Toc10129674)

[Key words 1](#_Toc10129675)

[1 绪论 1](#_Toc10129676)

[1.1 选题背景意义 1](#_Toc10129677)

[1.2 国内外研究概况 2](#_Toc10129678)

[1.3 研究目的和内容 2](#_Toc10129679)

[2 区块链及其他相关开发技术 2](#_Toc10129680)

[2.1 区块链简介 2](#_Toc10129681)

[2.2 区块链解决的信任问题 3](#_Toc10129682)

[2.3 以太坊区块链 3](#_Toc10129683)

[2.4 智能合约 3](#_Toc10129684)

[2.5 连接以太坊区块链的工具以及使用 4](#_Toc10129685)

[2.5.1 Truffle和Ganache 4](#_Toc10129686)

[2.5.2 Remix-Solidity IDE 4](#_Toc10129687)

[2.5.3 Metamask 5](#_Toc10129688)

[2.5.4 JavaScript库Web3js 5](#_Toc10129689)

[2.6 星际文件系统IPFS 6](#_Toc10129690)

[2.7 本章总结 6](#_Toc10129691)

[3 基于区块链的数字版权管理系统设计方案 6](#_Toc10129692)

[3.1 基于区块链的数据版权管理系统架构 6](#_Toc10129693)

[3.2 基于智能合约的数字版权管理协议 8](#_Toc10129694)

[3.2.1 合约内容总览 8](#_Toc10129695)

[3.2.2 数字版权SHA-256查重和上传合约协议 10](#_Toc10129696)

[3.2.3 数字版权设置和查询合约协议 10](#_Toc10129697)

[3.2.4 数字版权交易合约协议 10](#_Toc10129698)

[3.3 基于本地以太坊私链和IPFS的数字版权存储机制 10](#_Toc10129699)

[3.3.2 数字版权存储机制步骤 10](#_Toc10129700)

[3.3.3 以太坊本地私有链网络 10](#_Toc10129701)

[3.3.4 身份认证存储 11](#_Toc10129702)

[3.4 基于SHA-256、Simhash查重和IPFS hash值的数字版权确权机制 11](#_Toc10129703)

[3.4.1 文本确权 11](#_Toc10129704)

[3.4.2 图片和视频确权 12](#_Toc10129705)

[4 基于区块链的数字版权管理系统实现 13](#_Toc10129706)

[4.1 区块链服务端实现 13](#_Toc10129707)

[4.1.1 创建以太坊本地私链 13](#_Toc10129708)

[4.1.2 Metamask与ganache连接 14](#_Toc10129709)

[4.1.3 Remix-Solidity IDE合约编译和部署 15](#_Toc10129710)

[4.1.4 Web3js连接到私有链 18](#_Toc10129711)

[4.2 IPFS上传、访问实现 19](#_Toc10129712)

[4.2.1 IPFS安装、初始化和启动 19](#_Toc10129713)

[4.2.2 访问IPFS的方法 19](#_Toc10129714)

[4.2.3 建立IPFS节点 19](#_Toc10129715)

[4.3 客户端实现 19](#_Toc10129716)

[5 实验与测试 20](#_Toc10129717)

[5.1 区块链用户个人信息 20](#_Toc10129718)

[5.1.1 区块链用户登录 20](#_Toc10129719)

[5.1.2 区块链用户个人中心初始化或修改信息 21](#_Toc10129720)

[5.2 上传版权信息 23](#_Toc10129721)

[5.2.1 文字版权查重和上传 23](#_Toc10129722)

[5.2.2 图片版权查重及上传 25](#_Toc10129723)

[5.2.3 视频版权查重及上传 27](#_Toc10129724)

[5.3 查询版权信息 27](#_Toc10129725)

[5.4 版权设置费用和交易 29](#_Toc10129726)

[6 结束语 31](#_Toc10129727)

[6.1 总结 31](#_Toc10129728)

[6.2 展望 32](#_Toc10129729)

[致谢 32](#_Toc10129730)

[参考文献 32](#_Toc10129731)

基于区块链技术的数字版权管理系统

计算机科学与技术专业学生 朱新凡

指导老师 赵青松

摘要：目前数字版权网络侵权严重，保护措施乏力，近年来区块链的出现可以有效解决这一难题，区块链不可篡改的特性可以记录的所有数据变化的过程，有利于实现版权保护的透明化。本系统提出了一种基于区块链技术的数字版权管理系统设计方案，包括智能合约协议、文件存储链上链下机制和基于SHA-256、Simhash海量查重算法和IPFS hash值的确权机制，采用VUE设计实现前端、Remix-Solidity IDE和 Ganach部署区块链私链、Web3js和MetaMask与以太坊交互数据、Solidity 0.5.0编写智能合约、IPFS星际文件系统保存大文件，基本实现了数字版权的查重、上传、查询和交易。

关键词：区块链；智能合约；版权保护；Solidity语法；Web3js

Digital Rights Management System Based on Blockchain

Student majoring in Computer Science and Technology Zhu Xin-fan

Tutor Zhao Qing-song

Abstracts: At present, the digital copyright network is seriously infringed and the protection measures are weak. In recent years, the emergence of the blockchain can effectively solve this problem. The process of all data changes that can be recorded by the non-tamperable feature of the blockchain is conducive to the transparency of copyright protection. This system proposes a digital copyright management system design scheme based on blockchain technology, including intelligent contract protocol, file storage chain chain-down mechanism and confirmation mechanism based on SHA-256, Simhash massive check algorithm and IPFS hash value. VUE design for front-end, Remix-Solidity IDE and Ganach deployment blockchain private chain, Web3js and MetaMask and Ethereum interactive data, Solidity 0.5.0 to write smart contracts, IPFS interstellar file system to save large files, basically realize digital copyright Check, upload, query and trade.

Key words**:**Blockchain; Solidity; Digital copyright protection; Solidity Grammar; Web3js

# 1 绪论

## 1.1 选题背景意义

数字版权是在各种出版网络上发布版权或分发数字媒体（图片，视频，文章）的权利 [1]。

目前对数字版权的保护正遭遇严峻的挑战：

（1）网络数字版权数量激增。互联网和手机端正在飞速发展，微博、知乎、微信、抖音、快手等新媒体每天都产出大量的原创文章、图片、短视频，

（2）保护措施乏力。网络盗版不同于传统盗版，借助于链接、社交分享、P2P 技术和云端网盘具有上传迅速，具有下载快捷，扩散广泛，成本低，时间短的特点[2]。在互联网时代，除官方出版物外，还有大量的微视频，图片，网络文学等，需要去获得版权保护，如果你去中国版权保护中心进行版权登记，你需要支付注册费，这需要花费很多时间。

目前的数字版权保护主要是依赖第三方机构的集中式管控，缺点：流程繁琐，效率低，版权归属模糊，集中化的版权管控机制容易遭受单点故障，数据存储安全性低。将区块链技术的引入，利用区块链的不可篡改性，可以完全记录产品的所有变化，有利于版权交易的透明度。版权交易对手方购买版权时不用再怀疑每一笔交易数据的真实性[3]。[构建基于区块链的数字版权管理服务平台，解决我国数字版权保护问题](file:///C:\Users\pigfafa\Desktop\report_wx_slzywkfv_2019-5-20-19-25-46\PaperPass-%E6%97%97%E8%88%B0%E7%89%88-%E6%A3%80%E6%B5%8B%E6%8A%A5%E5%91%8A\htmls\sentence_detail\42.html)，对于解决“确权难、收益难，维权难”等问题是非常有利的[4]。

利用区块链技术做数字版权保护是针对所有需要保护的数字版权，是希望所有的版权都上传到区块链之中。例如让同一类的所有艺术家都上传自己的原创艺术到系统中，作者、上传时间、认证结果的信息都可以很好的储存在区块链上，所以可以做到很好的确权。

## 1.2 国内外研究概况

许多成熟的基于数字版权的探索已经出现在国外，越来越多的公司开始尝试将这种技术应用于自己领域的发展。例如，以色列公司Colu推出了基于区块链技术的版权保护平台，以帮助那些不了解区块链的开发商和消费者。他们还可以通过平台建立和交换电子资产[5]。如金融资产，文件记录或专有交易[6]。还有像Binded，Desent，Minelabs等其他外国公司一直在争取数字领域的版权保护和交易[7]。

我国目前也有在区块链数字版权上做出系统，如One Fair平台的主要内容是建立公开透明的原创作品交易平台，解决艺术品交易的问题等。2017年1月份，中国版权保护中心华夏微电影微视频区块链版权（交易）服务平台正式上线，它将建立一个结合了生产，点击和利益的新生态链。从而在互联网时代推广视频内容交易更加方便和公平。

## 1.3 研究目的和内容

①认证用户和区块链用户注册登录模块。需要验证实名身份或者私钥等其他证明信息。②用户数字版权信息的上传和登记模块。

③文字、图片、视频内容查重模块。利用SHA-256和Simhash算法，系统需要对即将上传到区块链上的相应作品进行查重确权审核通过。

④交易模块。作者将带有版权编号的作品上传成功后，可以自行设置作品的权限和版权费用，其他普通账号的成员可以按价格购买版权费用并提出转账。

⑤智能合约模块。上传，确认，设置权限和版权作品交易的过程由智能合约自动执行，并且根据指定交易的信息执行操作。

⑥区块信息模块。每一条对于区块链操作的信息都会被记录，并且不能篡改。

# 2 区块链及其他相关开发技术

## 2.1 区块链简介

区块链技术是比特币技术应用的底层技术，是一种分布式数据库，通过去中心化、共识信任的方式，集体维护一个可靠的数据库[8]。“去中心化”是指整个网络不存在中心管理机构，各节点间权利义务对等，任意节点的损坏或灭失都不会威胁整个系统的正常运作；“共识信任”是指整个系统遵循公开透明的运行规则，所有数据的记录与传递使用非对称加密和哈希算法来确保真实、不被篡改，从而促成系统各节点间达成一致信任；“集体维护”是指系统中的数据块由系统所有节点共同维护，运用分布式数据库技术方便每个参与节点都能及时获得一份完整数据拷贝[9]。

从根本上说，区块链是一个共享数据库，由一个交易分类账组成。就像银行一样，简单区块链的分类账跟踪货币（在这种情况下，加密货币）的所有权。与集中式银行不同，每个人都有一份分类账，可以验证对方的账户。具有分类帐副本的每个连接设备称为“节点”。

## 2.2 区块链解决的信任问题

（1）完全分散（Full decentralization）：读取/写入数据库是完全分散和安全的。没有任何一个人或团体控制区块链。

（2）极限容错（Extreme fault tolerance）：容错是系统处理损坏数据的能力。虽然容错并非区块链所独有，但通过让每个共享数据库的帐户验证其更改，它将概念置于逻辑极端。

（3）独立验证（Independent verification）：任何人都可以验证交易，无需第三方。这有时被称为“去中介”。

## 2.3 以太坊区块链

以太坊是一个区块链，允许您在其受信任的环境中运行程序。这与比特币区块链形成对比，后者只允许您管理加密货币。以太坊有一个名为以太坊虚拟机（EVM）的虚拟机。EVM允许在区块链上验证和执行代码，从而保证它将在每个人的机器上以相同的方式运行[10]。此代码包含在“智能合约”中（详情参考2.4智能合约一节）。除了跟踪帐户余额之外，以太坊还在区块链上维护EVM的状态。所有节点都处理智能合约，以验证合同及其输出的完整性。

任何人都可以创建自己的以太坊网络。可以在本地模拟以太坊区块链以进行开发。本地测试网络立即处理交易，并且可以根据需要分配以太网。开发人员在最终部署到主网络之前使用公共测试网络（或测试网络）来测试以太坊应用程序。这些网络上的以太网仅用于测试目的，没有任何价值。广泛使用的公共测试网络有三种：

Ropsten：[以太坊基金会](https://www.ethereum.org/foundation)创建的测试网。

Kovan：使用称为“权威证明”的共识方法的网络。这意味着其事务由select成员验证，从而导致一致的四秒块时间。此testnet上的以太网供应也受到控制，以减轻垃圾邮件攻击。

Rinkeby：由以太坊基金会创建的权威证明的一个测试网。

私有以太网网络允许各方共享数据而不会公开访问。私有链可以共享敏感数据，例如医疗保健记录；由于网络规模较小，可扩展以处理更高的读/写吞吐量

本设计是基于以太坊创建的本地私有链。在前期测试性能时，使用过Kovan以及Rinkeby测试网络。

## 2.4 智能合约

智能合约是在EVM上可以运行的代码合集。智能合约可以接受和存储以太网，数据或两者的组合。然后，使用编程到合同中的逻辑，它可以将该以太分配给其他账户甚至其他智能合约[11]。

以太坊区块链用的语言是solidity，是一种具有以下特征的面向对象语言：

（1）Solidity有一种特殊类型的地址用于定位用户，合同和合同代码，因为合同本身也是一个帐户。

（2）Solidity语言有着一个十分特殊的异常处理机制。如果过程中出现任何异常，所有执行都将被强制追溯。

（3）Solidity的运行环境建立在去中心化的网络上，一个函数调用变为一个网络中节点的代码执行，有着分布式的特点。

本系统设计用的是solidity 0.5.0版本的语法，solidity的版本很重要，需要在合约的开头描述，虽然在实际情况中可以这样写：pragma solidity >= 0.4.0 < 0.6.0，没有语法错误，并且兼容了0.4.0到0.6.0所有的版本，但实际上如果使用Remix-Solidity IDE编译的话，最终也是需要选择具体版本的编译器。如果使用truffle编译的话，也是依据truffle的版本会限制你的solidity版本，目前如果使用最新的truffle编译，只能是0.5.0以上版本的solidity语言，另外solidity语法更新很快，会修改很多语法内容。

## 2.5 连接以太坊区块链的工具以及使用

### 2.5.1 Truffle和Ganache

Truffle：目前最流行的以太坊开发框架，支持智能合约的编译、部署和测试。要求nodeJS v8.9.4或更高版本。官方建议搭配使用[Ganache](https://truffleframework.com/ganache)，这是在桌面上运行的以太坊开发的个人区块链。Ganache将合同和交易放在前面和中心来简化dapp开发。使用Ganache，可以快速了解详细信息，如帐户，余额，合同创建和gas成本。还可以微调Ganache的先进采矿控制，以更好地满足需求。

Ganache启动了个人以太坊区块链，可用于运行测试，执行命令、检查状态以及控制链的运行方式，页面参见图1。

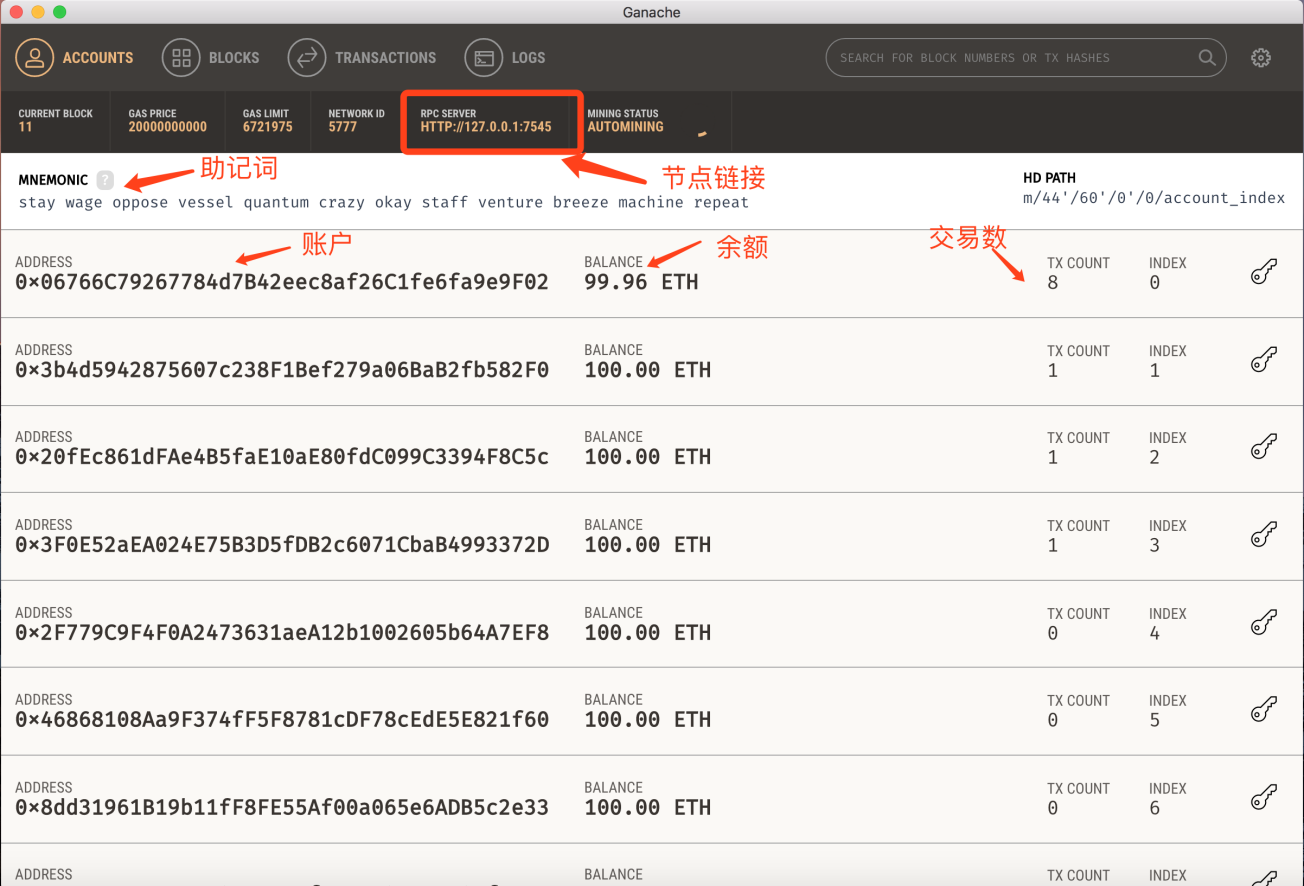


图1 ganache页面介绍

### 2.5.2 Remix-Solidity IDE

Remix-Solidity IDE是Solidity智能合约的开发环境。访问地址（https://remix.

ethereum.org/#optimize=true）也可以下载下<https://github.com/ethereum/remix-ide>

到安装到本地，但是在Windows平台上安装到本地会产生许多不兼容问题，如果需要本地下载尽量使用ubuntu或者mac系统。

这里只做简单介绍，具体使用参照测试部分。简单来说就是可以将通过编译的合约部署到区块链的节点上。设计的应用访问合约需要知道合约的地址和ABI定义。Remix-Solidity IDE的运行界面参见图2。

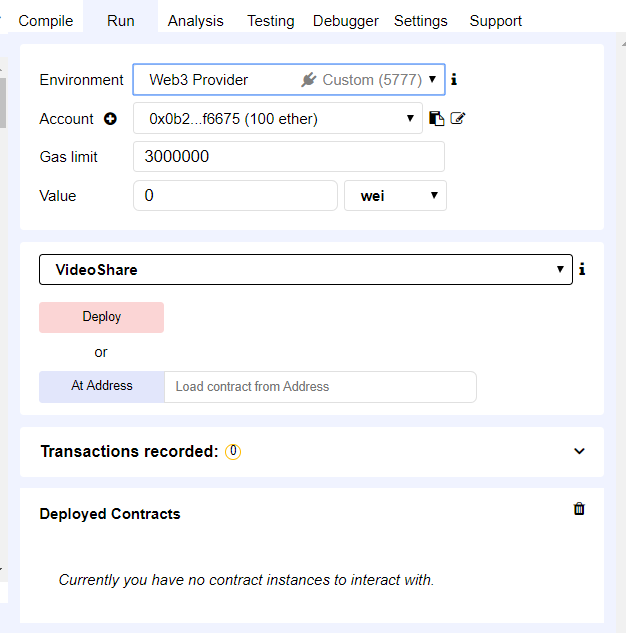


图2 Remix Solidity IDE 运行界面

### 2.5.3 Metamask

MetaMas是一款基于浏览器的插件Ethereum轻量级钱包，其扩展程序已添加到Google Chrome中。可以跟以太坊区块链的智能合约互动，它是链接本设计与钱包的方式。

MetaMask不存储钱包信息，所有钱包的私钥和密码都由用户持有。即使MetaMask停止更新，您也可以使用私钥将自己的货币退回到其他钱包。注意：Metamask作为google的扩展程序安装时需要vpn，MetaMask界面参见图3。

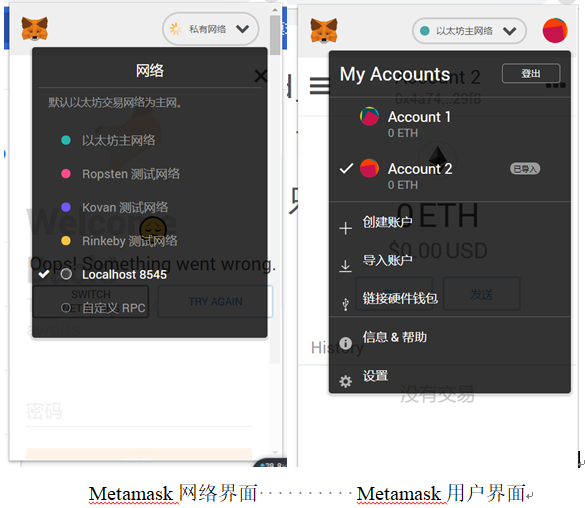


图3 Metamask界面

### 2.5.4 JavaScript库Web3js

Web3.js是以太坊提供的一个Javascript库，封装了以太坊的JSON RPC API，提供了与区块链交互的对象和函数，包括查看网络状态，查看本地账户、查看交易和区块、发送交易、编译/部署智能合约、调用智能合约等，其中最重要的就是与智能合约交互的API。可以使用HTTP或IPC建立与以太坊节点旳连接。可以检查Web3是否存在。若存在就使用web3.currentProvider作为它的提供者。

Web3.js 需要两个东西来和合约对话: 地址和ABI。详情可以参照3.2 Remix-Solidity IDE的章节。

## 2.6 星际文件系统IPFS

IPFS全名叫星际文件系统，是一个旨在创建持久且分布式存储和共享文件的网络传输协议。它是一种内容可寻址的超媒体分发协议。在IPFS网络中的节点将构成一个分布式文件系统。

简单来说，以本设计对它的使用来说理解，它其实就是一个文件系统；支持点对点的传输。存储结构是哈希链，会将文件分解为多个block，每个block都有单独的hash值，在读取文件时要把分解过后的block重新组合成逻辑上的文件。

## 2.7 本章总结

本章作了区块链的基本简介，基本技术的介绍，还介绍了本设计用到的开发基本工具，这里只做介绍，具体创建私链以及部署合约尚未涉及，工具间相互联调将在第四部分重点描述。

# 3 基于区块链的数字版权管理系统设计方案

## 3.1 基于区块链的数据版权管理系统架构

下图图4为系统架构图



图4基于区块链的数据版权管理系统架构图

其中客户端为VUE前端，用于身份认证用户注册登录、区块链用户修改个人信息、发起版权上传、版权查询和版权交易的请求。

在数据传入到区块链上之前要在后端做数据的查重和权限审查保证上传到区块链上的数据准确无误。

区块链服务端结构一共分为3层，首先是合约层：最为关键，需要制定规则，主要是需要制定版权查询权限、交易的规则；网络层：遵循以太坊eth的网络协议和p2p网络；存储层: 区块链不适宜存储大文件，设计采用链上链下结合的方式，区块链上存储用户信息、文字版权的内容、图片和视频的hash值、权限等信息，区块链下通过 IPFS存储图片以及视频作品。

## 3.2 基于智能合约的数字版权管理协议

部署在以太坊区块链上的合约需要地址，由Remix-Solidity IDE中提供。这里使用了solidity语言来描述智能合约。合约内容主要包括以下三个大类：设置、上传和获取链上内容、设置权限和版权交易。

### 3.2.1 合约内容总览

合约设计的结构体参见表1。

表1结构体

|  |  |
| --- | --- |
| 结构体 | 含义 |
| User | 区块链用户 |
| Txt | 文字 |
| Picture | 图片 |
| Video | 视频 |
| Comment | 评论 |
| Lable | 标签 |
| Gratuity | 打赏 |
| Album | 合集 |

其中主要介绍文字和视频结构体的构建。

struct Txt {

文字版权内容

版权是否公开权限

文字上传时间戳

（这里的时间戳now是unix的时间戳，是记录从1970年1月1日一来的秒数）

版权费用

版权所有者

}

struct Video {

标题

封面

视频简介

视频持续时间

视频上传时间戳

视频版权所有者

评论数量

打赏数量

标签数量

版权是否公开权限

版权费用

文件ipfs hash值

标签映射

评论映射

打赏映射

}

solidity语法中function的描述很严谨，举例来说必须在定义函数时规定函数的调用方式，根据不同的调用情况一共有external, public, internal, private四种方式，在这里不多加赘述。在返回函数时也要注意,solidity返回值最多只能返回11个,太多的话会导致栈溢出。

以下表2介绍一些有特征的funtion信息，类似的方法省略。

表2 部分function

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 操作分类 | 方法名 | 调用类型 | 作用 | 参数 | 返回 |
| 用户操作 | setMyInfo | public | 设置用户信息 | nickname, \_profile, \_avatar（昵称,简介,头像） | 无返回 |
| 用户操作 | getMyinfo | public | 获取用户信息 | Address(用户地址) | nickname,profile.avatar,  （昵称,简介,头像） |
| 文字操作 | setTxt | public | 上传文件版权信息 | \_txtId, \_content, \_permission（版权号，内容，权限） | 无返回 |
| 文字操作 | getTxt | public | 获取版权文件信息 | \_txtxtid（版权号） | content, num, time, author  （内容，权限，时间戳，作者地址） |
| 文字操作 | setTxtFee | public | 设置文字版权费用 | \_txtid,fee  （版权号，费用） | 无返回 |
| 文字操作 | bugTxtRight | public payable | 购买文字版权 | \_txtid（版权号） | 无返回 |
| 图片操作 | getImgOwner | public | 获取已有版权数目和版权号 | Address(用户地址) | num,,imgid  （数量，版权号） |
| 视频操作 | getVideo | public | 获取视频信息 | videoid（版权号） | title, cover, info, author,commentsNum, gratuityNum, gratuitySum  （标题，封面，作者，评论数，打赏数，打赏总额） |
| 视频操作 | getVideoFile | public | 获取视频文件信息 | videoid（版权号） | filename, fileinfo, size, width, height, fps, time  （文件hash，文件信息，大小，长，宽，fps，时间戳） |
| 评论操作 | makeComment | public | 评论视频 | \_videoId,\_videotimestamp, \_comment（视频版权号，时间戳，评论内容） | 无返回 |
| 打赏操作 | reward | public payable | 打赏视频 | \_videoId,  （视频版权号） | 无返回 |

### 3.2.2 数字版权SHA-256查重和上传合约协议

①获取区块链上所有版权的SHA-256算法后的hash值

②比较现在上传的内容是否有重复

③无重复则上传版权内容、作者及其他信息

④记录此刻时间戳记录在版权信息内

### 3.2.3 数字版权设置和查询合约协议

①检测权限，当前账户是否有权限操作版权

②根据输入的参数设置版权内容或者返回版权内容及其他属性

### 3.2.4 数字版权交易合约协议

①根据版权号查询交易版权的费用和版权所有者

②向合约地址发送版权费用和gas

③合约地址发送费用到版权所有者

④获得版权许可

## 3.3 基于本地以太坊私链和IPFS的数字版权存储机制

目前区块链的存储问题：

现在如要在以太坊区块链上存储大量数据，是非常昂贵的，大约gas8字节需要20,000的gas。大约存储1Mb数据需要10以太，目前1以太=1264美元（2019.5.13日的价格），所以存储图片、视频或者其他大文件到以太坊区块链上很不现实，价格过于昂贵。

大多数区块链设计选项都选择通过加密数据在链下存储更大的数据。散列操作和其他防止数据被篡改的方法，只在区块链上引用存储数据的散列值。

区块链用户在区块链中无法分辨，识别个体采用地址，只要你申请账号，并且可以申请以太币，就可以成为区块链用户，没有任何的限制，所以设计需要单独设立一种用户，需要经过身份认证。

### 3.3.2 数字版权存储机制步骤

数字版权存储机制包含如下步骤：

（1）发起数字版权上传请求: 用户通过客户端上传文字、图片或视频到区块链的请求时立即可以调用部署在区块链上的智能合约，区块链上会及时存储智能合约执行的结果。

（2）IPFS存储图片和视频内容:通过客户端网关上传数字作品到 IPFS，若上传成功，生成唯一的hash值存入区块链上，区块链用户可以通过工作的哈希值访问存储在IPFS上的图片和视频信息。

（3）数字版权设置权限、查询请求:调用部署在区块链上的智能合约，区块链上会及时存储智能合约执行的结果。

（4）数字版权交易请求:启用在区块链上的智能合约，获取购买版权的费用，再调用智能合约和Metamask交易，交易记录会保存在区块链上。

（5）构造区块:只要进行对区块链修改数据call()，都会生成新的区块并按时间顺序连接到私链上。

### 3.3.3 以太坊本地私有链网络

在一般的以太坊书籍和教程中，会让你使用go-ethereum客户端，简称Geth，是基于Go语言环境，在构建私有链时，需要定义创世区块。之后Geth会自动连接主网，并且同步所有的节点，大致需要几百G的数据，默认保存在C:\用户\用户名\AppData\Roaming\Ethereum目录下，这样十分消耗本机的资源。

但是现在没有必要自己在底层创建私链。并且下载几百G的数据到本地，可以使用ganache来构建以太坊私有链，现在的以太坊开发几乎都使用ganache。

### 3.3.4 身份认证存储

这里可以调用阿里云的身份一致性查询接口，可免费调用10次，之后需要付费，但是官方提供的java和php的调用代码，由于本人的在学习VUE前端时水平有限，无法加入到项目当中，最后选择了另一种方式判断此身份证号是否是中国大陆第二代身份证号码。

引入一个身份认证的javascript库validid，用于验证中国，台湾，香港和韩国的身份证号码。

用户身份的信息可以存储到区块链上，但是吧身份信息上传到区块链对于隐私来说是很不安全的，因为区块链很透明，所以最终本设计想将身份认证用户和区块链用户区分开来，所以选择使用Mysql来存储用户信息，使用Navicat可视化操作Mysql，只是存放用户基本信息。但是这个交互的过程很不理想，需要将区块链用户的address作为与Mysql数据库交互的桥梁。

## 3.4 基于SHA-256、Simhash查重和IPFS hash值的数字版权确权机制

数字版权的确权是一件很困难的事，即使是国家的专利申请，都会依据谁先申请的才判断，严格来说数字版权的确权甚至还需要其他网站资源作为库。在本设计中由于技术限制，只是基于提交的数据进行查重和确权，输入的文字内容为短文本，图片和视频采用hash存储的形式，所以采用了安全散列算法SHA-256和海量数据相似度计算Simhash算法来作为本设计的确权技术。

### 3.4.1 文本确权

1、SHA-256

算法描述：

（1）信息预处理

此部分有两个步骤，填充比特和附加长度。

Step1: 在消息的结尾处进行添加，使得在模512之后的消息长度的剩余部分是448。将第一位填充为1，然后加0，直到长度等于512.余数为448。如果长度满足模512之后的余数是448的情况，则还执行补码，并且此时填充512位。

Step2: 原始数据的长度信息（填充的第一步之前的消息）被添加到已填充的消息中。SHA-256使用64位数据来表示原始消息的长度。

（2）计算信息摘要

消息M被分成N个512位数据块，然后迭代次数是N，并且N次迭代的结果是最终散列值。

通过第一数据块和映射函数获得哈希的初始值H(0)以获得 H(1)，并且通过第二数据块和映射函数获得 H(1)以获得 H(2)…[顺序处理，最后得到H(N)，然后将8位32位H(N)连接成256位消息摘要。](file:///C:\Users\pigfafa\Desktop\report_wx_slzywkfv_2019-5-20-19-25-46\PaperPass-%E6%97%97%E8%88%B0%E7%89%88-%E6%A3%80%E6%B5%8B%E6%8A%A5%E5%91%8A\htmls\sentence_detail\398.html)

迭代的过程如图5所示：Mi为第i个512bit的数据块，Hi为第i次迭代后的信息摘要。其中映射算法十分复杂，先将每一块（512位）分解为16个32位的字，记为w[0], …, w[15]。然后是64次加密循环，其中ABCDEFGH这8个字（word）在按照一定的规则进行更新。

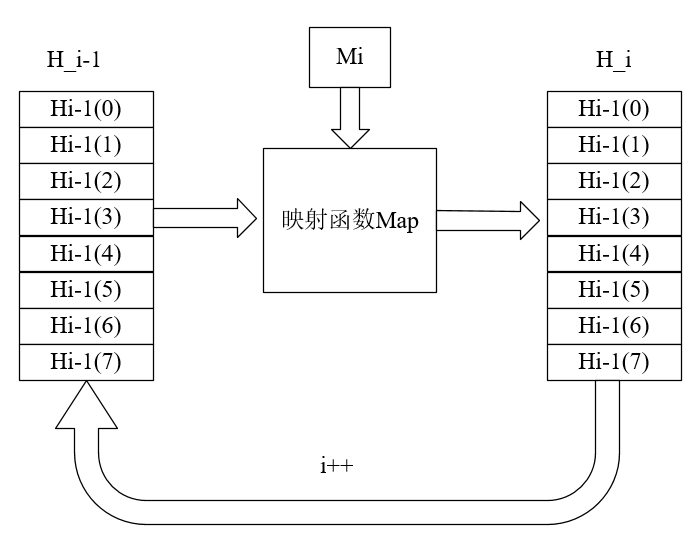


图5 sha-256 迭代过程

2、Simhash查重

Simhash是一种局部敏感的散列算法，其主要思想是降维，它将高维特征向量映射到低维特征向量。两个矢量的海明距离用于确定物品是重复还是高度近似。[海明距离越小，相应文本越相似。](file:///C:\Users\pigfafa\Desktop\report_wx_slzywkfv_2019-5-20-19-25-46\PaperPass-%E6%97%97%E8%88%B0%E7%89%88-%E6%A3%80%E6%B5%8B%E6%8A%A5%E5%91%8A\htmls\sentence_detail\408.html)simhash算法分为5个步骤：分词、hash、加权、合并、降维。可以用来计算短文本相似度。

（1）分词。给出一个段落，执行分词，获得有效的特征向量，然后为每个特征向量设置1-5和其他五个权重级别。

（2）hash。[通过散列函数计算每个特征向量的散列值，并且散列值是由二进制数01组成的n位签名。](file:///C:\Users\pigfafa\Desktop\report_wx_slzywkfv_2019-5-20-19-25-46\PaperPass-%E6%97%97%E8%88%B0%E7%89%88-%E6%A3%80%E6%B5%8B%E6%8A%A5%E5%91%8A\htmls\sentence_detail\415.html)

（3）加权。在hash值的基础上，给所有特征向量进行加权，即W = Hash \* weight，且遇到1则hash值和权值正相乘，遇到0则hash值和权值负相乘。

（4）合并。累积上述各个特征向量的加权结果，使其仅成为一个序列串。

（5）降维。对于n位签名的累积结果，如果大于0则设置为1，否则设置为0，从而获得语句的simhash值。

最后，在引用的库函数当中，降维后得到64位hash值，可以根据不同句子simhash的海明距离来判断它们的相似性，出现不同的位置则+1，最后得到一个文本相似度，就是海明距离/64。

综上两种算法，在设计文本确权机制的时候，决定将已确权的版权信息上传到区块链的版权库中。在上传新版权的时候需要进库一一做查重比对，例如simhash，需要和已确权的文本一一做文本相似度比较，并返回一个最大的文本相似度值，如果超过一定的阈值则属于侵权行为不予上传的请求。

### 3.4.2 图片和视频确权

参考3.3.1的存储方式，图片和视频上传到IPFS后会返回hash值，这里采用了IPFS的技术，通过IPFS的hash值来查重。

IPFS产生的hash值经过以下步骤产生：

Step1:通过SHA-256基于内容产生hash值

Step2: 使用multihash这种hash表达方法。由三部分组成：HASH算法编码，HASH值长度（字节），HASH值

Step3: 采用Base58编码，所以得出的hash值以Qm开头。

所以确权的思想与文本相似，也是传入区块链的版权库，在上传之前，只要逐一比较区块链中的hash值是否相似，即可确权。

# 4 基于区块链的数字版权管理系统实现

## 4.1 区块链服务端实现

### 4.1.1 创建以太坊本地私链

创建私链过程：

Step1: 安装ganache的前提环境：nodejs在8.6.0版本以上（版本越高越好）。还需要npm，版本也是越高越好。

Step2：在官方github上 (<https://github.com/trufflesuite/ganache/releases>) 符合

你操作系统的ganache 。本设计使用的是ganache 1.3.1版本。（目前最新是2.0.2版本，以太坊更新换代特别快）

Step3：打开ganache，设置私有链的信息，主要是端口号，见图6

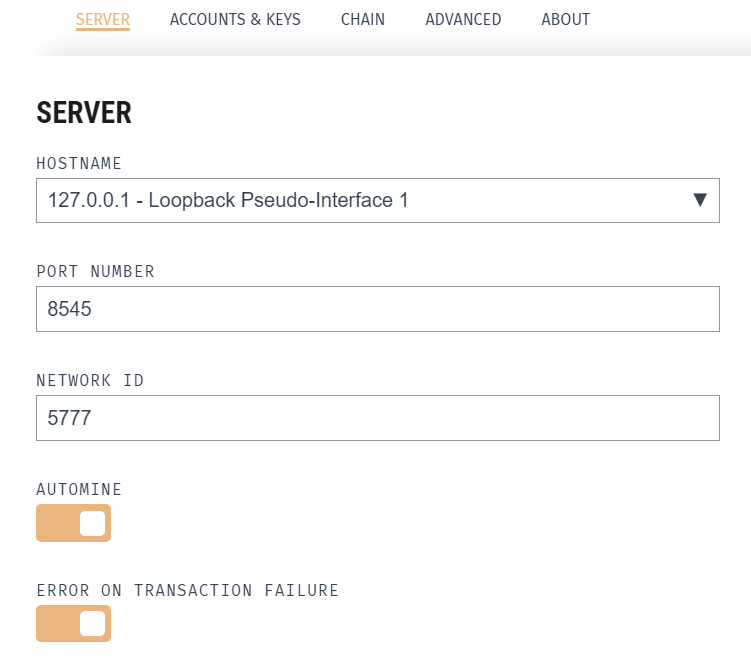


图6 ganache服务接口设置

总结：至此ganache创建私链已经成功，如下图所示ganache在<http://127.0.0.1:8545>运行，并自动创建创世区Block 0，运行后默认创建10个账号，每个账号里有100ETH的余额。可以自主设置需要多少个账号和初始以太币的数量，注意虚拟的以太币是不可以在测试网络或者以太坊的主网络中使用的，就算你想要去使用，在切换网站之后，你的余额也会自动清零。但是如果想用测试网完成设计或者做其他的区块链项目，需要根据以太坊测试网的要求去申请测试以太币，这里本设计尝试过，获得了很好的效果，具体不多赘述，私有链的创世块和提供的账户信息参见图7和图8。

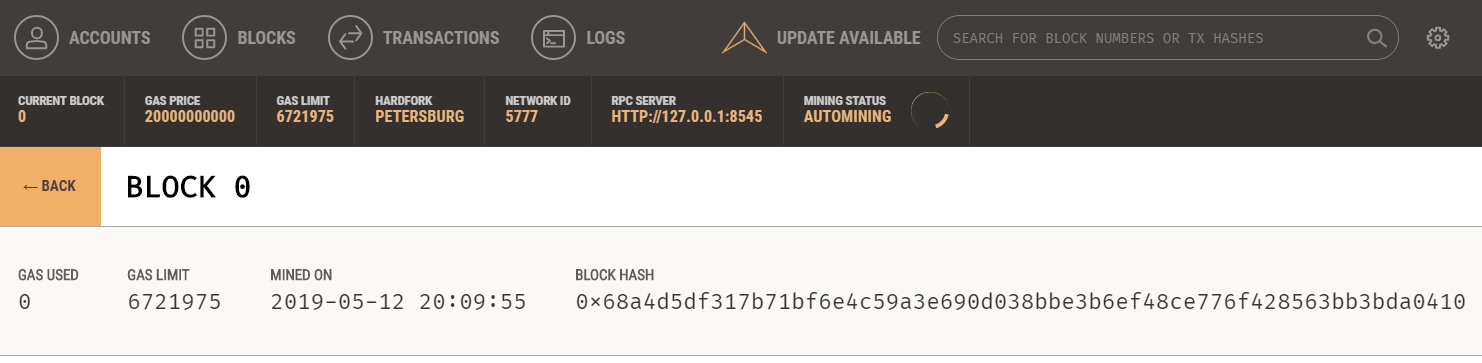


图7 ganache创世块内容

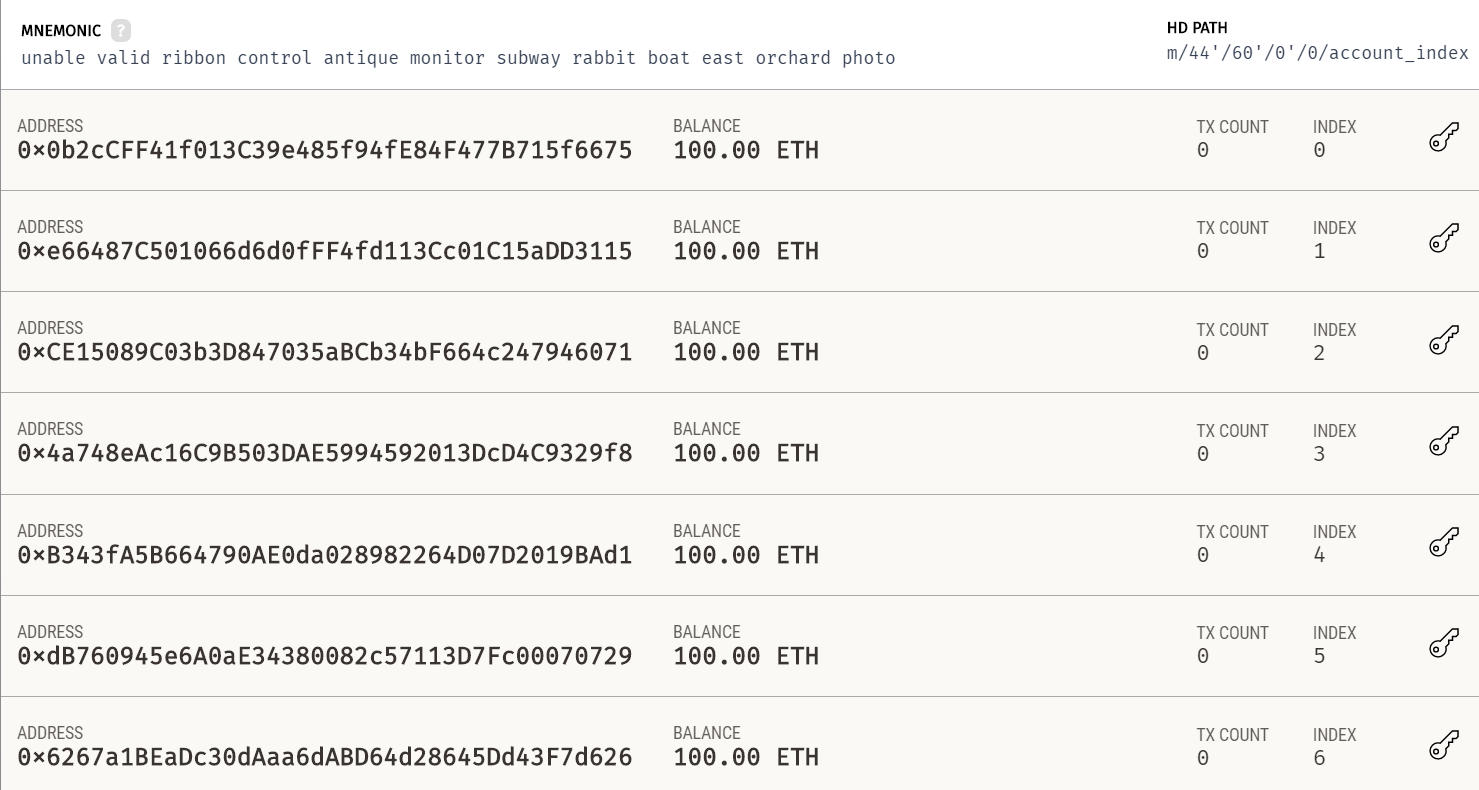


图8 ganache提供的账户信息

### 4.1.2 Metamask与ganache连接

Step1:打开Metamask，网络选择ganache所运行的服务端口，这里就是<http://127.0.0.1:8545> 相当于localhost 8545。

Step2: 用ganache提供的私钥（私钥信息见图9）导入账户，导入方法见图10。

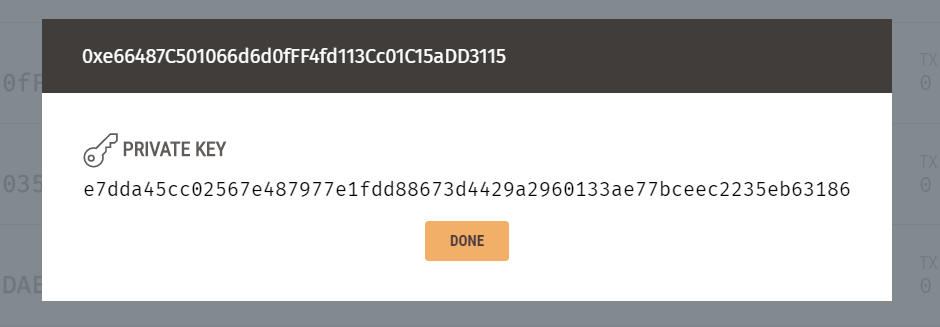


图9 账户的私钥

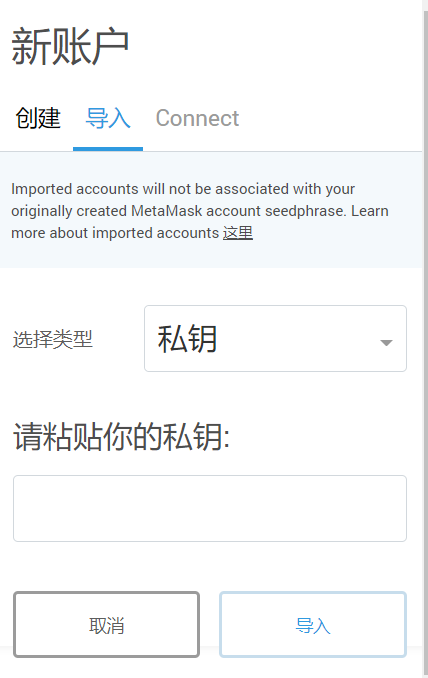


图10 Matamask中导入私钥

Step3: 导入两个账户之后可以相互转账SEND 以太币，并且ganache可以获取交易记录即可。

### 4.1.3 Remix-Solidity IDE合约编译和部署

部署智能合约的流程图11如下：



图11部署智能合约的流程图

Step1: Remix-SolidityIDE需要和Metamask先产生连接，如图12所示

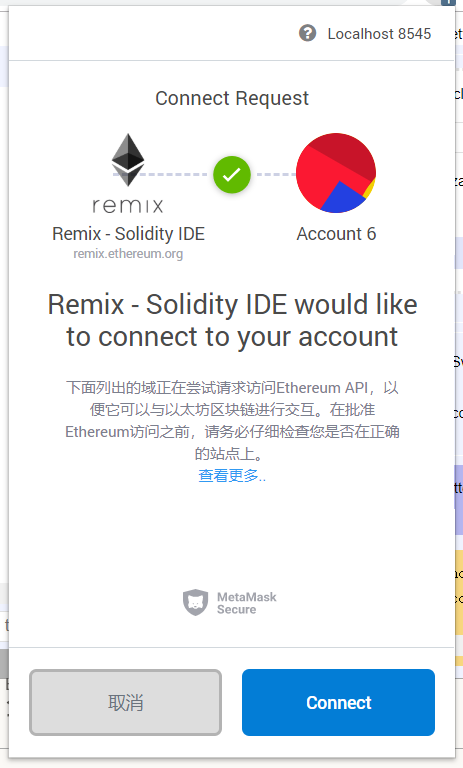


图12Remix-SolidityIDE、Metamask连接图

Step2: 选择编译器版本进行编译，这里本设计采用的是0.5.0+commit.1d4f

565a.Emscripten.clang 的编译器可以适配0.5.0到0.6.0版本之间的语法。编译环境见图13。

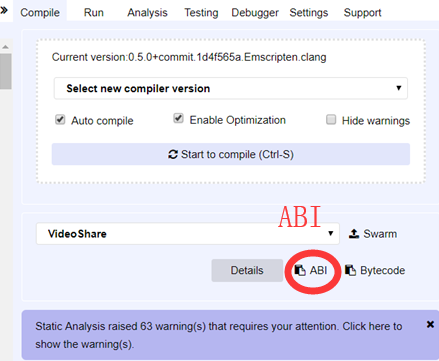


图13Remix-Solidity IDE编译器环境

Step3: 编译成功后将图13中的ABI复制并加入到项目中。

Step4: 点击run，选择web3 provider连接到本地私链，见图14、15。

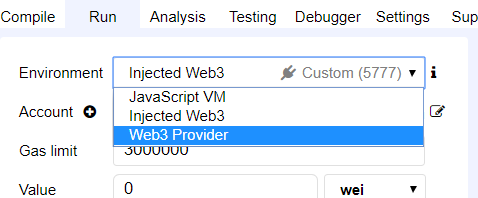


图14Remix-Solidity IDE运行环境选择

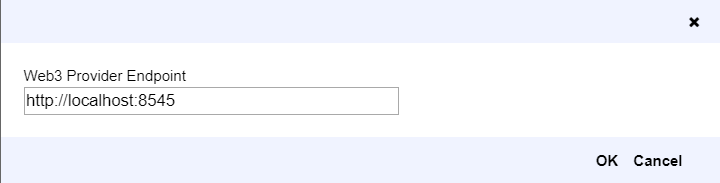


图15Remix-Solidity IDE运行环境选择端口

Step5: 点击图16上的Deploy部署，花费一些gas部署合约到私有链上

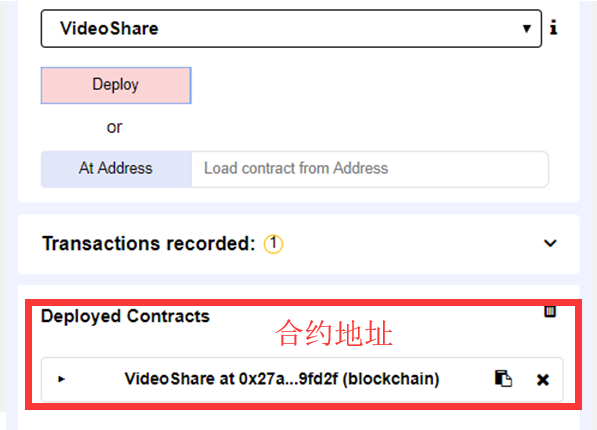


图16Remix-Solidity IDE部署合约

在这里BLOCK1中已经存入信息，显示操作为部署合约，通过图16和图17，可以看出合约地址一致，说明操作正确



图17区块1中的内容

### 4.1.4 Web3js连接到私有链

①需要通过npm下载web3 1.0.0beta版本到项目中

②其次需要先检查是否开启metamask插件并与之相连

if (typeof web3 !== 'undefined') {

web3js = new Web3(web3.currentProvider);

} else {

//使用外置的ethAPI

web3js=newWeb3(new Web3.providers.HttpProvider(config["network"]

[0]["httpApi"]));

③通过web3js获取合约信息，参数是编译后的abi以及合约地址。

const videoShare = new web3js.eth.Contract(videoShareAbi,

config["network"][0]["contractAddress"]);

④检测是否有账户登录

web3js.eth.getAccounts((error, result) => {

if (result.length > 0) {

this.$store.commit('setUserAccount', result[0]);

this.userAccount.address=result[0];

}

## 4.2 IPFS上传、访问实现

### 4.2.1 IPFS安装、初始化和启动

IPFS的安装在这里不多做赘述，在官网上下载并且按步骤即可安装成功，安装之后进入到下载地址的命令行，执行命令ipfs init对ipfs仓库进行初始化，可以设置本地仓库的大小，如果不设置，系统的默认大小是10G，还可以设置服务器端口（默认8080），但本设计基于的电脑在8080有system系统占用，所以这里选择了8081端口作为服务器端口。命令行ipfs daemon启动。其他的细节参照ipfs的教程即可，注意可能会遇到跨域问题。

### 4.2.2 访问IPFS的方法

可以通过命令行添加文件到IPFS，但是这样对用户的要求过高，并且也不符合项目需要集成性的要求，所以需要将访问IPFS的任务加入到后端中。

设计采用调用IPFS-API的方式上传文件到IPFS的节点，采用gateway的方式访问IPFS节点信息这个是公共网关，将https://ipfs.github.io/public-gateway-

[checker/gateways.json](https://ipfs.github.io/public-gateway-checker/gateways.json)文件包含进项目，这样可以不需要连接到[ipfs.github.io](http://ipfs.github.io/)这个域名进行获取资源。此外还有3个本地网关，分别是127.0.0.1、当前访问的域名、当前访问域名的8081端口，这个端口是ipfs节点默认的网关端口。

"http://127.0.0.1:8081/ipfs/:hash",

"http://" window.location.host "/ipfs/:hash",

"http://" document.domain ":8081/ipfs/:hash"

运行后 this.$store.state.gateway 这个全局变量中就有了一个最快返回的网关，还可以切换这里的网关。

### 4.2.3 建立IPFS节点

在项目中通过npm可以下载ipfs的API，通过命令定义本地节点

const ipfsAPI = require('ipfs-api');

const ipfs = ipfsAPI({host: 'localhost', port: '5001', protocol: 'http'});

在这里复制编译好的ipfs.js文件到public目录，然后在index.html引入，

其中需要描述repo: '/ipfs-' + Math.random() 每次都需要新开一个空间，否则会有权限错误。再设置一个检测，每10秒，输出一下当前的状态。

## 4.3 客户端实现

客户端采用vue-cli 3.0的框架，虽说是框架，其实主体页面还是自己写的，主要利用到他的路由以及便于与第三方库连接的特点，安装时用npm install安装依赖即可。这里只是简单应用了vue的版型，并没有深入应用。

其中

public文件夹内是保存的静态资源

src/api 与后端交互使用相关方法和配置

src/service 存放所使用api的方法和数据处理

views 所有路由组件（本设计的界面）

App.vue 设计的顶层路由

main.js为vue的入口函数

router,js所有的路由

每个vue文件都是一个组件，但是在本设计中相当于界面，其中每一个vue都包含了template、script、和style这些其实就是web开发传统意义上的html，javascript和css。

这里介绍设计的App.vue文件的内容，App.vue显示的是整个系统的最上端部分，系统的一些初始化可以放在这个部分来完成，在上面描述的ipfs，web3js，智能合约的初始化可以在这一部分完成。

在<template></template>之间描述的是页面最顶上的部分。<script>描述一些函数，例如按钮触发什么条件，产生什么结果。虽然其中主要是javascript的语法，但是vue在这个script部分是最基本要写data、methods两个数据操作的函数。可以在data内定义很多类型的数据。引用时需要通过this引用，相当于是整个页面是对象，引用数据要用this去调用，除非你是定义的临时变量。

这里还要介绍一下在App.vue中的一些操作，例如如何显示和保持登录可以用sessionStorage，可以在输入登录数据时，临时保存，刷新或者关闭时自动删去。还有axio机制，他可以和后台交互数据，也可以读取json文件的内容，还可以访问服务器端口，例如监听3000端口，可以设置代理target: 'http://127.0.0.1:3000/api/', //对应自己的接口，在后台写好之后，然后通过axio进入。但是在这里本系统设计遇到了很大的阻力，因为都是逻辑错误，在开发者选项中，只能显示无法进入或者路径错误，具体哪里出现了问题，很难找出。

# 5 实验与测试

## 5.1 区块链用户个人信息

### 5.1.1 区块链用户登录

本设计是面向以太坊区块链用户的，所以只需要区块链用户只需要在Metamask内导入私钥，即代表用户注册已存在。如图16，目前Metamask登录的address尾号为3115与系统中显示的登录address一致。注意，这里由于将地址全部显示出来，会使得整个顶层路由很挤，没有其他按钮的地方，所以通过utils/assist.js中的函数，做出调整，使其显示出的效果可以像Metamask一样，中间用…来表示，首和尾两部分也足以看出是否一致，登陆信息参见图18，登录测试用例如表3所示。

表3 登陆测试

|  |
| --- |
| 用例 预期结果 实际结果 测试结果 |
| 进入界面 出现地址 出现地址 通过 |
| 未输入合约进入界面 显示未登录 未登录 通过  未开启私链进入界面 显示未登录 未登录 通过  未开启ipfs进入界面 出现地址 出现地址 通过 |



图18 区块链用户登录

### 5.1.2 区块链用户个人中心初始化或修改信息

进入到区块链个人中心界面，见图19。



图19 区块链用户个人中心

将测试信息填入，见图20。个人中心设置测试用例如表4所示。

表4 个人中心设置测试

|  |
| --- |
| 用例 预期结果 实际结果 测试结果 |
| 填写昵称保存 跳转至Metamask支付 跳转至Metamask支付 通过 |
| 填写简介保存 跳转至Metamask支付 跳转至Metamask支付 通过 |
| 上传头像保存 跳转至Metamask支付 跳转至Metamask支付 通过 |
| 全部保存 跳转Metamask支付gas为总和 跳转Metamask支付gas为总和 通过 |



图20 区块链用户个人中心保存信息后

全部保存后，因为要向区块链提交数据，所以要花费必要的gas用于上传，此时新的区块产生，记录了对合约进行了CALL操作，存入数据，见图21和图22。

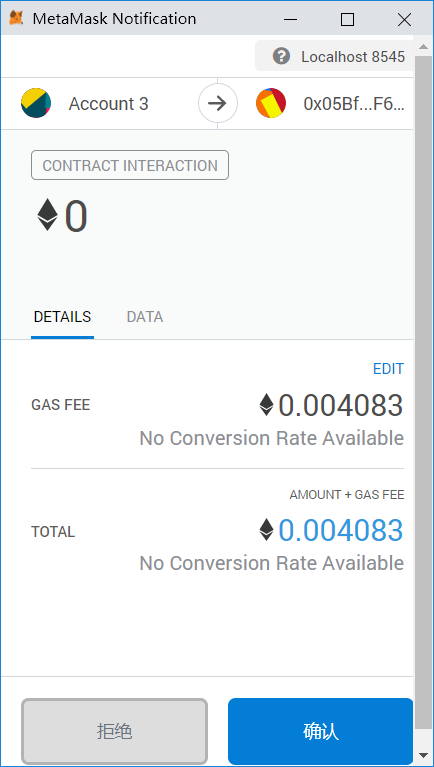


图21 支付相应的gas



图22 区块3的内容

刷新界面，可见个人信息已经存好并且可以展示（区别在于右上角），见图23。



图23 个人中心保存后变化（默认显示出）

## 5.2 上传版权信息

### 5.2.1 文字版权查重和上传

刷新后，输入短文本到框内，点击文本是否重复按钮，若显示可以上传至区块链说明无侵权，属于区块链用户的正版。若确认侵权则表明不可上传。见图24-28。

上传到区块链设定不花费eth，但是需要支付gas，支付成功后，可以在Metamask中查询history记录以及log。文字版权SHA-256查重及上传测试用例参见表5和表6。

表5 文字版权SHA-256查重及上传

|  |
| --- |
| 用例 预期结果 实际结果 测试结果 |
| 第一次输入123456 跳转Metamask支付 跳转Metamask支付且支付成功 通过 |
| 第二次输入123456 提示有重复 有重复侵权 通过 |
| 切换用户输入123456 提示有重复 提示有重复 通过 |
| 未刷新输入123456 会产生覆盖错误 产生覆盖错误 通过  未刷新输入1234 56（空格） 跳转Metamask支付 跳转Metamask支付且支付成功 通过 |



图24通过SHA-256侵权查重后成功提示



图25通过SHA-256侵权查重后失败提示

表6 文字版权Simhash查重及上传

|  |
| --- |
| 用例 预期结果 实际结果 测试结果 |
| 第一次输入I am pig 无重复可上传 无重复可上传 通过 |
| 第二次输入I am pig 提示相似度1 有重复,侵权 通过 |
| 切换用户输入I am pig 提示相似度1 有重复,侵权 通过 |
| 输入My name is pig 提示相似度0.5 有重复,侵权 通过 |
| 输入this is money 提示相似度0.27777 可上传 通过 |



图26通过Simhash侵权查重后失败提示



图27通过Simhash侵权查重后成功提示



图28通过Simhash侵权查重后失败提示

### 5.2.2 图片版权查重及上传

刷新后可以先上传预览图片，点击是否重复后，若如文字上传时一直，可以上传至区块链，证明没有侵权，可以上传，见图29。

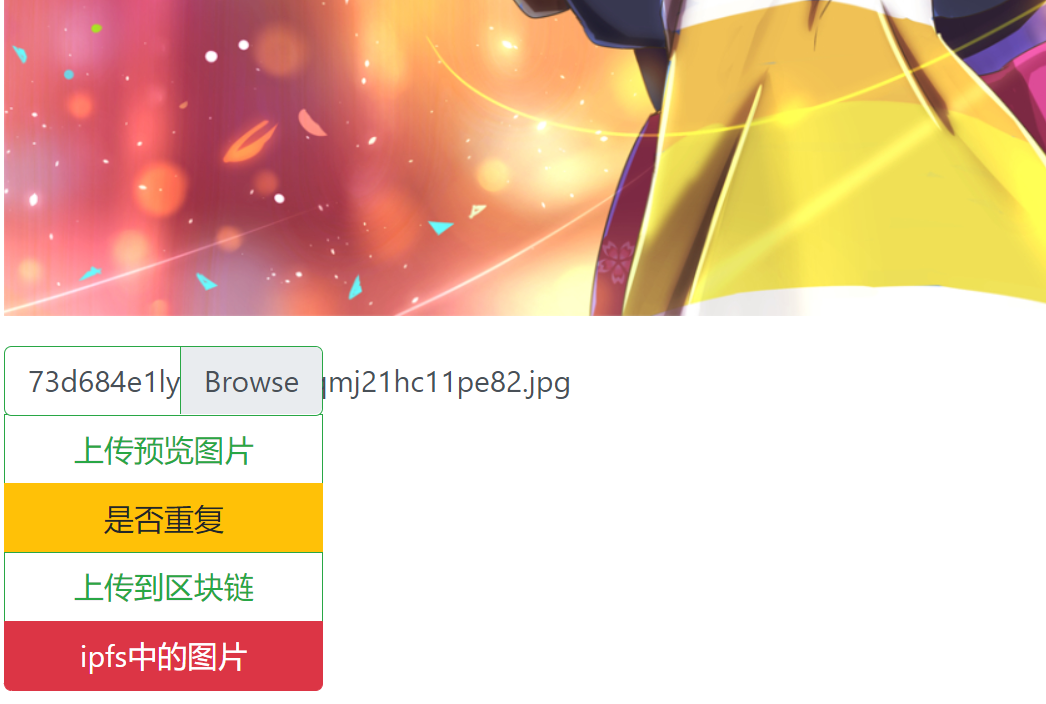


图29图片上传

点击上传到区块链后，同上支付一定数目gas。图片测试用例和结果与文字差不多。

见图30。图片版权重及上传测试用例参见表7

表7 图片版权重及上传

|  |
| --- |
| 用例 预期结果 实际结果 测试结果 |
| 第一次上传图像1 跳转Metamask支付 跳转Metamask支付且支付成功 通过 |
| 第二次输入图像1 提示有重复 有重复侵权 通过 |
| 切换用户输入图像156 提示有重复 有重复侵权 通过 |
| 未刷新输入图像1 会产生覆盖错误 产生覆盖错误 通过 |

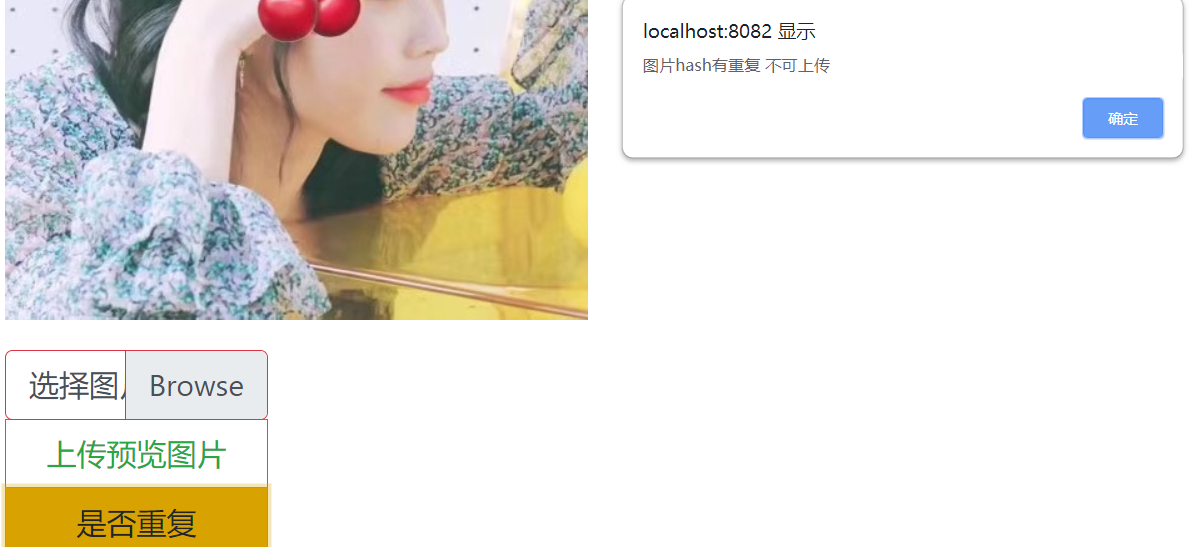


图30图片确权失败后提示

### 5.2.3 视频版权查重及上传

操作过程见图31、图32。



图31视频上传



图32视频上传侵权提示

## 5.3 查询版权信息

根据版权号可以查询相关版权的信息，可以先查询自己已有的版权号。

操作过程见图33-38。版权查询测试用例参见表8。

表8版权查询信息

|  |
| --- |
| 用例 预期结果 实际结果 测试结果 |
| 作者查询编号0（有权限） 版权内容展示 版权内容展示成功 通过 |
| 作者查询编号0（无权限） 版权内容展示 版权内容展示成功 通过 |
| 其他账户查询编号0（有权限） 提示error 提示error 通过 |
| 其他账户查询编号0（无权限） 版权内容展示 版权内容展示成功 通过 |
| 管理员6675查询编号0（有权限） 版权内容展示 版权内容展示成功 通过 |



图33个人版权编号查询

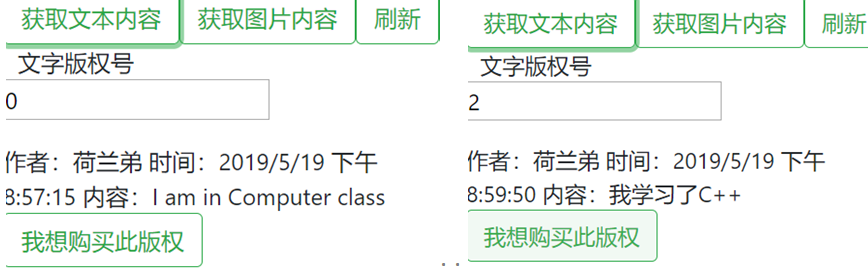


图34查询想知道的版权



图35管理员账户6675查询有权限的版权



图36其他普通账户查询有权限的版权error提示



图37作者查询有权限的版权



图38视频展示界面

## 5.4 版权设置费用和交易

设置作者所持有的2号版权为1eth币，见图39。

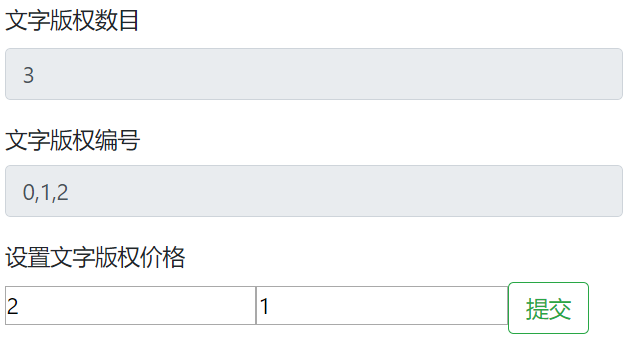


图39设置版权费用

切换账户address为下图的账号后去购买版权，会发现需要花费除去gas以外1eth币的费用。见图40、图41。

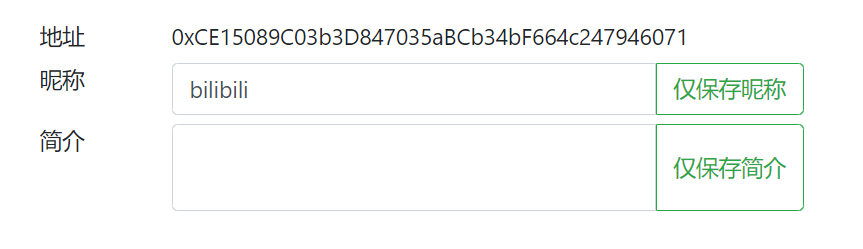


图40切换账号



图41支付费用

执行完之后，可以发现Account3账户收到转账，见图42。



图42收到转账

# 6 结束语

## 6.1 总结

本设计完成了基于区块链的数字版权管理系统的基本功能，研究了以太坊区块链的原理和机制，研究并应用了SHA-256和Simhash算法用于版权确权。以上的论文描述了本设计的选题背景、国内外研究现状以及研究目的和内容。介绍了本设计完成过程中包括区块链以及其他相关所使用到的开发技术，阐述了本设计的设计方案框架以及其中的一些原理和设计的机制，并完成了基本的实现方案，最后进行了相关功能应用的实验和测试。

在做毕业设计的初期，一直在找寻教程和资料，在windows和ubuntu平台上都做过很多尝试，由于没有在linux系统上开发的经历，搭建以太坊私有链网络的过程很不顺利，而且搭建的过程很底层，很多用户的操作都是在以太坊提供的javascript开发控制台上进行，对整个系统的开发造成了不小的阻力。

在选择了开发工具以及方法之后，慢慢的步入了开发的正轨，从底层开始慢慢设计和研究，也遇到了开发很多的困难，主要来源于资料很少，对开发技术的不熟练，例如solidity语法报错，如何设计智能合约，还有前端后端页面间的数据连接和路由。

通过这次毕业设计，学习到了很多东西，主要是学习能力和解决问题的能力，还有搜集资料的能力，经常在github和csdn的网站上学习别人的技术开发，学习区块链的道路上需要英语更加熟练，需要看很多白皮书和API方法，还有这些研究很多都是国外的研究，纯英文的，对自身来说还是很大的挑战。但是经过这次开发经历，对于以后的开发和学习做了很好的铺垫。

## 6.2 展望

本设计基于区块链的数字版权管理系统，由于系统和个人的能力，还有一些问题需要解决，以下是对系统的不足之处的整理和总结：

①设计的操作有冗余。例如给版权设置费用，完全可以在上传的时候就设置，没有必要另外设计函数。

②获取区块链上文字版权的个数时，需要刷新两次，才可以获得，这个原因还没有找到，目前推测是每一次点击按钮可能只会执行一次区块链function函数的调用。

③还可以加入其它的功能让界面更加美观和完善，例如评论、视频弹幕。

④系统的健壮性不够，在下一个操作之前必须要先执行一个必要的操作，例如上传版权时，必须要先刷新一下才可以。

⑤ganache每一次重新打开，区块上的信息就会消失，所以测试很不方便，而且也没有测试过并发环境。

综上的问题，希望今后可以在部署到以太坊主网络之前可以得到有效的解决。

致谢

本设计能够顺利完成首先要感谢指导老师赵青松老师，从开题开始就对本设计的课题十分关心，给了许多建议和帮助，会发很多关于区块链和数字版权的前沿论文，给予了很大的帮助，在中期时每隔一段时间就要检查所有同学的进度和代码。还要感谢周围同学的鼓励和支持，能得以坚持完成区块链的学习。有了他们的关心和支持，还有宝贵的设计建议，才得以完成毕业设计，在此对于他们给予衷心的感谢。

参考文献

[1]赵丰,周围.基于区块链技术保护数字版权问题探析[J].科技与法律,2017,(01):59-70.

[2]伍旭川,王鹏.区块链技术应用及展望[J].清华金融评论,2016,(10):23-25.

[3]吴健,高力,朱静宁.基于区块链技术的数字版权保护[J].广播电视信息,2016,(07):60

-62.

[4]袁勇,王飞跃.区块链技术发展现状与展望[J].自动化学报,2016,(04):16-20.

[5]李绍民,姚远.区块链多媒体数据版权保护方法研究[J].科技资讯,2015,13(35):13-15.

[6]S.Haber, W.S. Stornetta,“Secure names for bit-strings,”In Proceedings of the 4th ACM Conference on Computer and Communications Security[J]. Computer and Communications

Security,1997,(04):28-35.

[7]徐珉川.知识产权的“去中心化”—比特币与登记制度[J].科技与法律,2014,(03): 474-

497.

[8]Dwork, Cynthia, Naor, Moni. “Pricing via Processing, Or,Combatting Junk Mail, Advances in Cryptology”[J]. Lecture Notes in Computer Science,1993,(03):139-147.

[9] William Cornish. Intellectual property: patents, copyright, trademarks and allied rights [M]. London:Sweet and Maxwell,1999:87.

[10]赵国锋,何英,周继华.基于区块链的数字版权登记技术[J].信息技术与网络安全,2019,3(4):79-83.

[11]李超,戴炳荣,王泓机,王晓强.基于区块链的数字版权保护与交易系统现代计算机[J].2018,10(4):80-84.