文本复制检测报告单(全文标明引文)

№:ADBD2018R_2018052419254943646504837 6　　检测时间:2018-05-24 19:25:49

检测文献:　基于区块链技术的虚拟货币矿机的实现

作者:　王朋飞

检测范围:　中国学术期刊网络出版总库

中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库

中国重要会议论文全文数据库

中国重要报纸全文数据库

中国专利全文数据库

图书资源

优先出版文献库

大学生论文联合比对库

互联网资源(包含贴吧等论坛资源)

英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)

港澳台学术文献库

互联网文档资源

CNKI大成编客-原创作品库

个人比对库

时间范围:　1900-01-01至2018-05-24

指导教师　邰非

审阅意见　通过，经过第一轮修改后，王朋飞同学所撰写的论文已基本符合本科毕业论文规范，水平达到中上等程度。建议通过。

检测结果

总文字复制比：7.5%　　跨语言检测结果：0%

去除引用文献复制比：7.2%　　去除本人已发表文献复制比：7.5%

单篇最大文字复制比：1.4%

重复字数：[1854]　　总段落数：[7]

总字数：[24799]　　疑似段落数：[4]

单篇最大重复字数：[346]　　前部重合字数：[754]

疑似段落最大重合字数：[724]　　后部重合字数：[1100]

疑似段落最小重合字数：[54]

指　标：☐疑似剽窃观点　☑疑似剽窃文字表述　☐疑似自我剽窃　☐疑似整体剽窃　☐过度引用

表格：0　　公式：0　　疑似文字的图片：0　　脚注与尾注：0

0%（0）　基于区块链技术的虚拟货币矿机的实现_第1部分（总2767字）

0%（0）　基于区块链技术的虚拟货币矿机的实现_第2部分（总509字）

54.2%（724）　基于区块链技术的虚拟货币矿机的实现_第3部分（总1335字）

14.1%（664）　基于区块链技术的虚拟货币矿机的实现_第4部分（总4701字）

18.3%（412）　基于区块链技术的虚拟货币矿机的实现_第5部分（总2255字）

0%（0）　基于区块链技术的虚拟货币矿机的实现_第6部分（总11956字）

4.2%（54）　基于区块链技术的虚拟货币矿机的实现_第7部分（总1276字）

（注释：无问题部分　文字复制比部分　引用部分）

1. 基于区块链技术的虚拟货币矿机的实现_第1部分　　总字数：2767

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

学号 142010020132

年级 14级
本科毕业论文
基于区块链技术的虚拟货币矿机的实现
专业应用物理学
姓名王朋飞
指导教师郜非
评阅人
2018年5月
中国南京
BACHELOR'S DEGREE THESIS
OF HOHAI UNIVERSITY
Realization of virtual currency mining machine based on blockchain technology
College ：Faculty of Science
Subject ：Applied Physics
Name ： Wang Pengfei
Directed by ： taifei Professor
May 2018
NANJING CHINA
学术声明：
郑重声明
本人呈交的毕业论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计（论文）的研究成果不包含他人享有著作权的内容。对本设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计（论文）的知识产权归属于培养单位。
本人签名： 日期：

摘要
虚拟货币从诞生到现在，已被越来越多的人接收，并有很多商家开始尝试以虚拟货币作为支付方式，这完全得益于虚拟货币去中心化的属性，这种属性能保证货币不会滥发、超发，也不受任何行政干预，从而能有效应对通货膨胀，通货膨胀正是当前各国央行无法解决的难题。从技术层面上来看，去中心化属性的实现是基于区块链技术，这项技术是结合加密、解密、链表、Hash处理、树及工作量证明来实现，当前这项技术已成为应用研究的热点领域。各国政府、机构、公司已投入大量的人力、物力去研究、开发基于区块链技术的相关应用。
论文中，我们选择了最为著名的虚拟货币：BTC比特币作为研究对象，通过对比特币源代码的分析和研究，同时利用更通俗易懂的编程语言Python来进行直接简单地模拟比特币的运行原理。使用SDK集成平台Anaconda，IDE编辑器为IntellJ IDEA,实现比特币的挖矿程序，从而完成比特币去中心化中的核心功能。
关键词：BitCoin；BlockChain；Python；flask；web；
ABSTRACT
Since the birth of virtual currency, it has been accepted by more and more people, and many businesses have begun to try to use virtual currency as a payment method. This is entirely due to the decentralized nature of virtual currency, which guarantees that currency is not It will be spamming, hyperventilation, and without any administrative intervention, so that it can effectively cope with inflation. Inflation is precisely the problem that central banks cannot solve. From a technical perspective, the realization of the decentralized attributes is based on blockchain technology. This technology is combined with encryption, decryption, linked lists, Hash processing, trees, and workload proofing. The current technology has become an applied research. Hot areas. Various governments, agencies, and companies have invested a lot of human and material resources to research and develop related applications based on blockchain technology. In the paper, we chose the most famous virtual currency: BTC Bitcoin as a research object, through the analysis and research of the bitcoin source code, and at the same time using a more straightforward programming language Python to directly and simply simulate the operation of Bitcoin. principle. Bitcoin's mining process is implemented to complete the core functions of Bitcoin decentralization.
Key words: BitCoin, BlockChain, Python, flask, web.

目录

2. 基于区块链技术的虚拟货币矿机的实现_第2部分

3. 基于区块链技术的虚拟货币矿机的实现_第3部分

总字数：1335

相似文献列表　文字复制比：54.2%(724)　疑似剽窃观点：(0)

1	环境感知的风扇自动控制系统设计与实现 张凯 -《大学生论文联合比对库》- 2017-04-04	22.2%（296） 是否引证：否
2	基于安卓平台的聊天机器人系统的设计与实现 曹达 -《大学生论文联合比对库》- 2016-05-26	17.1%（228） 是否引证：否
3	20115521_赵祥麟_面向短文本的网络舆情分析系统设计 赵祥麟 -《大学生论文联合比对库》- 2015-05-18	17.1%（228） 是否引证：否
4	基于响应式web的电子商务网站的设计和实践 童陈陈 -《大学生论文联合比对库》- 2016-04-03	17.1%（228） 是否引证：否
5	基于响应式web的电子商务网站的设计和实践 童陈陈 -《大学生论文联合比对库》- 2016-04-07	17.1%（228） 是否引证：否
6	安利APP的设计与实现 胡志巍 -《大学生论文联合比对库》- 2016-04-27	17.1%（228） 是否引证：否
7	基于响应式web的电子商务网站的设计和实现 童陈陈 -《大学生论文联合比对库》- 2016-05-29	17.1%（228） 是否引证：否
8	基于IOS的学生社交应用软件的设计与开发 付吉 -《大学生论文联合比对库》- 2016-04-16	17.1%（228） 是否引证：否
9	NI设备管理系统的设计和实现 王苏振 -《大学生论文联合比对库》- 2016-05-19	17.1%（228） 是否引证：否
10	NI设备管理系统的设计和实现 王苏振 -《大学生论文联合比对库》- 2016-05-23	17.1%（228） 是否引证：否
11	基于主动发现与被动扫描技术的web应用漏洞检测系统 孟晨宇 -《大学生论文联合比对库》- 2017-05-08	17.1%（228） 是否引证：否
12	817320_司徒晓欣_校园移动信息推送平台的设计与实现 司徒晓欣 -《大学生论文联合比对库》- 2017-05-24	17.1%（228） 是否引证：否
13	基于交通大数据的车辆调度算法实现 黄楚冰 -《大学生论文联合比对库》- 2017-05-25	17.1%（228） 是否引证：否
14	基于Docker的ERP信息管理系统后台实现 刘晨辉 -《大学生论文联合比对库》- 2017-05-25	17.1%（228） 是否引证：否
15	2220132452_袁牛飞_基于flask框架的测试环境自动化部署系统 袁牛飞 -《大学生论文联合比对库》- 2017-06-10	17.1%（228） 是否引证：否

16	软通-0123848-胡志巍-安利APP的设计与实现 软通 -《大学生论文联合比对库》- 2016-10-11	17.1%（228） 是否引证：否
17	基于树莓派的室内环境控制系统的设计和开发 周展弘 -《大学生论文联合比对库》- 2017-03-17	17.1%（228） 是否引证：否
18	基于flask的个人博客系统 李志航 -《大学生论文联合比对库》- 2017-04-11	17.1%（228） 是否引证：否
19	基于响应式web的电子商务网站的设计和实现 童陈陈 -《大学生论文联合比对库》- 2016-06-06	12.4%（165） 是否引证：否
20	基于多源软件数据的开发人员推荐技术研究与实现 齐鑫 -《大学生论文联合比对库》- 2016-06-01	11.6%（155） 是否引证：否
21	邰默涵-12051639-网络工程 邰默涵 -《大学生论文联合比对库》- 2016-01-04	11.2%（150） 是否引证：否
22	3120931046-李铭贤-基于Docker的在线实验管理系统的设计与实现 李铭贤 -《大学生论文联合比对库》- 2016-06-13	10.4%（139） 是否引证：否
23	【Python进入山东小学教材】吴恩达：孩子会识字后立马教她Python！_新智元 -《网络（http://chuansong.me/）》- 2017	10.2%（136） 是否引证：否
24	基于Python的专题搜索引擎 胡宇帆 -《大学生论文联合比对库》- 2017-05-09	9.3%（124） 是否引证：否
25	11303070312-唐琴-安全类新闻内容管理与发布系统设计与实现 唐琴 -《大学生论文联合比对库》- 2017-05-25	7.5%（100） 是否引证：否
26	反思比特币 顾继东 -《第一财经日报》- 2017-05-18	5.8%（78） 是否引证：否
27	比特币:模糊了货币的边界 孙柏; -《金融博览(财富)》- 2013-07-23	4.5%（60） 是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第一章相关介绍

1.1. Bitcoin

Bitcoin,缩写BTC，其概念最初由中本聪在2009年提出，比特币是一种P2P形式的数字货币。点对点的传输意味着比特币是一个去中心化的支付系统。

与传统货币相比，比特币有如下的三个优势：

1. 去中心化。比特币不依靠特定的货币发行机构，它根据特定的算法，通过大量的计算产生，产生的比特币通过众多节点构成的分布式数据库进行确认，使用密码学来保证比特币被真实的拥有者进行交易。

2. 隐私性。比特币账号由特定摘要算法根据用户公钥生成，不需要进行实名登记，即没有任何的身份信息。对于比特币来说，交易是支付给地址，而不是某个人，而用户可以根据私钥来申明账户的所有权。一个人可以拥有多个地址。

3. 公开性。比特币系统其实就是一个公开账本系统，任何人只要想查看比特币系统的交易记录，都可以从它附近的节点进行下载这些交易记录。每个账户的余额也是通过这些公开的交易记录中计算得来的。

1.2. Python

Python 是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。作为一种解释型语言，Python的设计哲学强调代码的可读性和简洁的语法（尤其是使用空格缩进划分代码块，而非使用大括号或者关键词）。相比于C++或Java，Python让开发者能够用更少的代码表达想法。不管是小型还是大型程序，该语言都试图让程序的结构清晰明了。Python 的设计具有很强的可读性，相比其他语言经常使用英文关键字，其他语言的一些标点符号，它具有比其他语言更有特色语法结构。同时，Python也是一种面向对象的语言，为代码编写的灵活性提供了充足的保障。

另外，Python有相对较少的关键字，结构简单，和一个明确定义的语法，学习起来更加简单，同时用Python书写的代码定义更加清晰，非常易于阅读，对于源代码的维护也非常容易，而对于Python来说，它最大的优势之一就是拥有丰富的库，跨平台的，在UNIX，Windows和Macintosh兼容很好。同时它还有可移植，可扩展，可嵌入等特点。因此，在这篇论文中，我选择使用Python来进行系统的模拟，原因之一就是因为使用Python书写的代码更容易让阅读者理解，同理，对于代码的编写也是较其他语言为简单的。

1.3. Flask

Flask是一个使用Python编写的轻量级Web应用框架。基于Werkzeug WSGI工具箱和Jinja2 模板引擎。Flask使用BSD授权。

Flask被称为“MicroFramework”，因为它使用简单的核心，用extension增加其他功能。Flask没有默认使用的数据库、窗体验证工具。然而，Flask保留了扩增的弹性，可以用Flask-extension加入这些功能：ORM、窗体验证工具、文件上传、各种开放式身份验证技术。

1.4. Anaconda

Anaconda 是一种Python语言的免费增值开源发行版，其包含了conda、Python等180多个科学包及其依赖项，用于进行大规模数据处理, 预测分析, 和科学计算,致力于简化包的管理和部署。Anaconda使用软件包管理系统Conda进行包管理。

指　标
疑似剽窃文字表述

1. 比特币不依靠特定的货币发行机构，它根据特定的算法，通过大量的计算产生，产生的比特币通过众多节点构成的分布式数据库进行确认，
2. Python
   Python 是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。
3. 相比于C++或Java，Python让开发者能够用更少的代码表达想法。不管是小型还是大型程序，该语言都试图让程序的结构清晰明了。Python 的设计具有很强的可读性，相比其他语言经常使用英文关键字，其他语言的一些标点符号，它具有比其他语言更有特色语法结构。同时，Python也是一种面向对象的语言，
4. Python有相对较少的关键字，结构简单，和一个明确定义的语法，学习起来更加简单，同时用Python书写的代码定义更加清晰，非常易于阅读，对于源代码的维护
5. Python来说，它最大的优势之一就是拥有丰富的库，跨平台的，在UNIX，Windows和Macintosh兼容很好。
6. Flask
   Flask是一个使用Python编写的轻量级Web应用框架。基于Werkzeug WSGI工具箱和Jinja2 模板引擎。Flask使用BSD授权。
   Flask被称为“MicroFramework”，因为它使用简单的核心，用extension增加其他功能。Flask没有默认使用的数据库、窗体验证工具。然而，Flask保留了扩增的弹性，可以用Flask-extension加入这些功能：ORM、窗体验证工具、文件上传、各种开放式身份验证技术。

4. 基于区块链技术的虚拟货币矿机的实现_第4部分

总字数：4701

相似文献列表　文字复制比：14.1%(664)　疑似剽窃观点：(0)

1	精通比特币-第7章 区块链 - 码农的专栏 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	6.7%（315） 是否引证：否
2	基于比特币块链技术的数字护照系统的设计实现 张佳程 -《大学生论文联合比对库》- 2016-05-18	5.2%（246） 是否引证：否
3	什么是区块链技术？ - lzhl1993的专栏 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	5.2%（245） 是否引证：否
4	Blockchain - 码农的专栏 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	5.0%（234） 是否引证：否
5	taifei的专栏 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	4.8%（227） 是否引证：否
6	精通比特币 - 第5章 交易 - taifei的专栏 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	4.5%（210） 是否引证：否
7	精通比特币-第8章 挖矿与共识 - 码农的专栏 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	2.3%（106） 是否引证：否
8	比特币的现状及未来发展研究 湛嘉俊 -《大学生论文联合比对库》- 2015-05-17	1.3%（62） 是否引证：否
9	1113800033 湛嘉俊 -《大学生论文联合比对库》- 2015-05-18	1.3%（62） 是否引证：否
10	比特币交易追踪机制及技术研究 梅溢 -《大学生论文联合比对库》- 2016-05-27	0.6%（29） 是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第二章比特币系统介绍

2.1. 比特币地址

在上面，我们提到，比特币进行交易的时候是向比特币地址进行交易的，这个比特币地址就相当于我们银行卡的账号一般，谁拥有这个比特币的地址，那么在这个比特币地址下面的比特币就属于谁，那么地址是怎么生成的呢？

下面是比特币地址的生成过程详细图解：

图2.1 比特币地址计算过程

上面的过程中，涉及到九个步骤，分别使用了不同的算法，包括椭圆曲线加密算法（ECDSA），SHA256，RIPEMD160消息摘要算法，BASE58加密算法。其中ECDSA，SHA256，PIPEMD160算法是不可

逆的。BASE58算法是可逆的。

具体过程介绍如下：

第一步。随机生成私钥，私钥取值范围为32个字节的数字，即范围为0~2^256,这个范围大到比地球上原子数还要大，可以保证私钥唯一性。秘钥可以是任意的一种形式，只要最后经过变换后能够对应于这32个字节的数字即可。例如为方便记忆秘钥可以是一句话。但最终使用的秘钥是这句话经过变换对应的32个字节的数字。

第二步。利用椭圆曲线加密算法ECDSA加密私钥生成公钥。由于使用非对称加密，外人无法通过公钥推算出私钥的内容，因此公钥可以对外公开而无需担心安全问题。

第三步。计算公钥SHA256摘要。获取Hash值。

第四步。对第三步的结果再进行RIPEMD160摘要，获取Hash值。

第五步。在第四步的结果前面添加地址的版本号，这个版本号一般是一个字节的。

第六步。对第五步的结果进行SHA256摘要，获取Hash值。

第七步。对第六步的结果再进行SHA256摘要，获取Hash值。

第八步。获取第七步结果的前四个字节（我们称之为校验码），并添加到第五步的结果后面。因此这个时候数据结果为“版本号+第五步结果+校验码”。

第九步。对第八步的结构进行BASE58编码。这个结果就是我们的比特币地址了。例如：“16UwLL9Risc3QfPqBUvKofHmBQ7wMtjvM”。

上面的过程中，私钥就相当于银行卡的密码，比特币地址就相当于银行卡的账号，比特币地址是从私钥经过密码学运算得到的，我们可以看到第二、三、四、六、七这几个步骤是不可逆的，因此是没有办法从比特币的地址获取到比特币的私钥的，也就是说虽然银行卡账号是从密码得来的，但是却没有办法从账号计算得到密码。

一个人可以生成多个比特币地址，并用在不同的交易上面，而且除非使用者自己披露外人是没有办法看出这些地址之间的联系的。

2.2. 比特币交易

2.2.1. 简介

比特币交易是比特币系统中最重要的部分。根据比特币系统的设计原理，系统中任何其他的部分都是为了确保比特币交易可以被生成、能在比特币网络中得以传播和通过验证，并最终添加入全球比特币交易总账簿（比特币区块链）。比特币交易的本质是数据结构，这些数据结构中含有比特币交易参与者价值转移的相关信息。比特币区块链是一本全球复式记账总账簿，每个比特币交易都是在比特币区块链上的一个公开记录。

2.2.2. 交易的输入输出

比特币交易中的基础构建单元是交易输出。 交易输出是比特币不可分割的基本组合，记录在区块上，并被整个网络识别为有效。 比特币完整节点跟踪所有可找到的和可使用的输出，称为“未花费的交易输出”（unspent transaction outputs），即UTXO。 所有UTXO的集合被称为UTXO集，目前有数百万个UTXO。 当新的UTXO被创建，UTXO集就会变大，当UTXO被消耗时，UTXO集会随着缩小。每一个交易都代表UTXO集的变化（状态转换）。

2.3. 比特币账簿、区块链、区块

2.3.1. 比特币账簿

比特币账簿上面记录着整个比特币网络的所有的交易记录，比特币账簿保存在彼此链接运行的比特币网络上面。前面提到，比特币网络是一个P2P网络，因此任何链接这个网络的人只要有意愿即可从比特币网络上面下载到比特币账簿上的记录，即上面提到的公开性。比特币账簿上面的已经被打包的交易记录不可被更改。每个比特币地址下的比特币数量是通过这个比特币的账簿的交易记录计算得来的。通俗来说即：查询比特币地址支出和获取的比特币交易记录，那么对这些记录进行运算就可以的到比特币账户的剩余比特币数量。

2.3.2. 区块链

提到比特币账簿就不得不提区块链技术，比特币账簿中的所有交易记录都以区块链这么一种数据结构进行存储，而区块链又是一个一个的区块连接起来的，你可以把它想象成一个环环相扣的锁链。每一个区块里面都包含着整个比特币网络约十分钟内的交易记录。这是因为打包计算的个区块的信息，将近要花费全部网络将近十分钟的算力。

对每个区块头的信息进行SHA256摘要，可以得到一个Hash值。这个Hash值就对应着区块链中的区块。每一个区块都会保存上一个区块的Hash值，一直追溯到创世区块（中本聪创造的第一个区块）。区块链结构如下所示：

图2.2 区块链结构

从上面的区块链结构中看出，每一个区块包含有上一个区块的Hash，而当前区块的Hash的计算又依赖者上一个区块的Hash。也就是说，如果当前区块信息发生改变，就会导致所有它之后的区块都会相应的改变，一旦某区块之后有很多区块后，这种瀑布效应将会保证区块的信息不可变，因为如果要修改某一区块信息，则必须重新计算当前区块和其之后所有区块的信息，而上面提到，每一个区块的计算要耗尽全网络将近十分钟的算力。因此，要想更改区块的信息需要极大的算力，是一件不可能完成的事情。所以一个长区块链的存在可以让区块链的历史不可改变，这也是比特币安全性的一个关键特征。

2.3.3. 创世区块

区块链中的第一个区块创建时间为2009年，称为创世区块。它是区块链里面所有区块的共同祖先，这意味着你从任一区

块，循链向后回溯，最终都将到达创世区块。创世区块中隐藏着一条信息*"The Times 03/Jan/2009 Chancellor on brink of second bailout forbanks."*。消息由比特币的创立者中本聪嵌入创世区块中。这句话是泰晤士报当天的头版文章标题，引用这句话，既是对该区块产生时间的说明，也可视为半开玩笑地提醒人们一个独立的货币制度的重要性，同时告诉人们随着比特币的发展，一场前所未有的世界性货币革命将要发生。

2.4. 比特币挖矿

2.4.1. 简介

上面已经介绍了比特币的交易以及比特币账簿，上面提到比特币是一个去中心化的网络，即没有"中央"服务器或者控制点。也就是说比特币账簿的记账权不在某个中心机构身上，而在所有参与者手中。但是并不是参与比特币网络的任何人都有比特币的记账权，因为比特币网络的价值"比特币"就包含在这些账簿的记录中，比特币的记账权分配将决定这比特币系统的安全性。那么就涉及到比特币的挖矿了。

挖矿是使得比特币与众不同的发明，他实现了去中心化的机制。挖矿即参与比特币记账权竞争的所有参与者中，要比赛解决一道数学难题，谁先得到这道题的正确答案谁就有比特币的记账权，那么他需要将比特币网络中未被打包的交易记录进行打包成一个区块，并挂载到总的区块链（总账簿）上面，同时他将有机会向区块中写入一条交易记录，即给自己一定数量的比特币作为奖励。等到打包的区块被其他参与者确认后，拥有者就可以使用他写入区块中的那笔交易中的比特币了。因此新比特币的生成过程被称为挖矿。而参与挖矿的参与者被称为矿工，由于比特币交易中会付出一部分的交易费。因此矿工不但可以获取创建的新币的奖励还可以获取到他所打包的所有交易中包含的交易费。

2.4.2. 比特币总量

按照挖矿的奖励机制，可以知道随时间的进行，比特币的数量将会越来越多，最终将会有无限多个比特币被创造出来，然而实际情况却并不是如此。比特币的数量上限约为2,100万比特币。这是因为初始的比特币奖励为50枚，之后每四年奖励的比特币将会减半。比如在2012年时比特币奖励为25枚，2016年的时候变为12.5枚，预计将会在2020年变为6.25枚。

图2.3 比特币总量变化趋势（摘自精通比特币第二版图10-1）

从上图中可以看到，比特币最终总量将会接近2100万，预计将在2140年左右会存在这么多的比特币。在那之后，新的区块不再包含比特币奖励，矿工的收益全部来自交易费。

2.4.3. 工作量证明机制

上面提到矿工们要想获得比特币网络的记账权，就需要去解答一个数学难题，而这个数学难题有一个特点就是会衡量当前整个网络的算力，保证全网络的参与者算出这个数学难题的时间在10分钟左右，这个数学难题就是工作量证明机制。

工作量证明机制使用Hash函数来进行，Hash函数输入一个任意长度的数据会输出一个长度固定的值，对于相同输入会有相同输出，但是如果输入只是存在细微的不同，例如多了一个标点，少了一个字母，或者多了一个数字其输出结果就会截然不同。因此Hash的结果将是不可预测的，也就是说，凭空设想一个输入来获取到一个想要的Hash值是不可能的。而工作量证明机制就是找到特定的组合使得这个组合Hash后的结果小于某个值。即过程如下：

图2.4 工作量证明机制公式

最终的目的就是要找到这么一个随机值使得上面的式子成立。可以看到Target越小，找到这么一个随机值将会越难，这也就可以做到动态调整公式求解的难度了。另一方面，如果某个矿工找到了一个随机值使得上面的式子成立，那么其他矿工去验证是非常简单的，他们之后对输入进行Hash就可以和结果进行比对。

2.5. 比特币运行原理总结

经过上面的介绍，基本上已经将比特币系统中的关键性的组成部分介绍完毕，当然比特币的系统中还有许许多多的细节部分支撑着比特币系统安全有序的进行，例如比特币系统中的区块链的分叉择长原则，挖矿难度调整细节等等。另外还有随着比特币发展而出现的一些新的技术，如矿池。由于本文的目的并不是完整再现比特币系统，而是从比特币运行的宏观方面模拟比特币运行比特币的原理。同时也为避免比特币知识占据本文的绝大部分篇幅，因此只选择了能快速理解和搭建比特币系统的关键性知识。

总结来说，比特币作为去中心化的对等共识网络，其特有的账号系统确保了其隐私性，另外公共账单即区块链技术和工作量证明机制保证了比特币的安全性以及公开性。使得虚拟货币能够在这几年飞速发展，总结比特币运行原理如下：

比特币参与者是比特币网络的参与者所创建的比特币账号。

首先：比特币网络参与者通过比特币软件随机生成的秘钥生成一个比特币地址。（当然这个秘钥是要一定记住的）

之后：拥有账号的参与者就可以使用这个账号与其他账号进行交易，账号之间的每一笔交易都会被比特币网络进行缓存（这些创建的交易还没有被矿工打包）。

与此同时：全世界参与比特币网络的矿工会对这些没有打包的交易记录整理并去争分夺秒求解一个数学难题，即找到上面提到的随机值。这个数学难题被解答出来的时间约为10分钟。一旦某个矿工解答出了这个数学题，它便将打包好的区块向网络中传播，等到被其他节点的参与者确认后，这个区块就会被永久的链接入比特币系统总的账簿中去。同时工作量证明机制和区块链结构将会保证这个区块不会被恶意更改。

最后：拥有记账权的矿工可以为自己写一笔交易，给予自己一定数量的比特币作为奖励。即向自己的地址写入一定数量的比特币。之后就可以使用自己获取的比特币了。

图2.5 比特币网络运行概览

指　标
疑似剽窃文字表述

1. 比特币交易的本质是数据结构，这些数据结构中含有比特币交易参与者价值转移的相关信息。比特币区块链是一本全球复式记账总账簿，每个比特币交易都是在比特币区块链上的一个公开记录。
2. 所以一个长区块链的存在可以让区块链的历史不可改变，这也是比特币安全性的一个关键特征。
   2.3.3. 创世区块
   区块链中的第一个区块创建时间为2009年，称为创世区块。它是区块链里面所有区块的共同祖先，这意味着你从任一区块，循链向后回溯，最终都将到达创世区块。创世区块
3. 消息由比特币的创立者中本聪嵌入创世区块中。这句话是泰晤士报当天的头版文章标题，引用这句话，既是对该区块产生时间的说明，也可视为半开玩笑地提醒人们一个独立的货币制度的重要性，同时告诉人们随着比特币的发展，一场前所未有的世界性货币革命将要发生。

5. 基于区块链技术的虚拟货币矿机的实现_第5部分

总字数：2255

相似文献列表　文字复制比：18.3%(412)　疑似剽窃观点：(0)

1	P29_理财小能手	15.3%（346）
	理财小能手 - 《大学生论文联合比对库》- 2016-05-23	是否引证：否
2	财经信息APP系统的设计与实现	13.9%（313）
	邬佳仁 - 《大学生论文联合比对库》- 2016-05-24	是否引证：否
3	序列化技术的综述和展望	5.9%（132）
	赵冬; - 《电脑与电信》- 2013-09-10	是否引证：否
4	科技管理系统设计与开发	5.8%（131）
	王鹏; - 《有色金属加工》- 2015-06-20	是否引证：否
5	科技管理系统设计与开发	5.8%（131）
	王鹏; - 《中国有色建设》- 2015-07-20	是否引证：否
6	触店高校学生信息服务平台 --Android客户端的设计与实现	5.6%（126）
	曾祥金 - 《大学生论文联合比对库》- 2014-06-11	是否引证：否
7	触店高校学生信息服务平台	5.5%（123）
	曾祥金 - 《大学生论文联合比对库》- 2014-06-05	是否引证：否
8	跑步应用iOS客户端设计与实现	5.5%（123）
	莫国远 - 《大学生论文联合比对库》- 2015-05-29	是否引证：否
9	跨平台手机移动中间件的设计与实现	3.1%（70）
	林远(导师：郭淑琴) - 《浙江工业大学硕士论文》- 2012-03-30	是否引证：否
10	基于IOS的手机应用程序开发—“友帮”二手市场	2.7%（60）
	谢俊杰 - 《大学生论文联合比对库》- 2015-06-09	是否引证：否
11	基于JSON的Android与SQLServer数据通信	2.3%（51）
	李志军; - 《电脑与信息技术》- 2015-08-15	是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第三章比特币模拟系统设计概要

3.1. 引言

3.1.1. 编写目的

本章用于描述“基于区块链技术的虚拟货币挖矿机”项目的系统设计。依据上面提到的比特币系统的主要组成部分设计一个新的基于区块链技术的挖矿机系统。为系统的概要设计，详细设计提供设计依据。

3.1.2. 项目背景

虚拟货币从诞生到现在，已被越来越多的人接收，并有很多商家开始尝试以虚拟货币作为支付方式，这完全得益于虚拟货币去中心化的属性，这种属性能保证货币不会滥发、超发，也不受任何行政干预，从而能有效应对通货膨胀，通货膨胀正是当前各国央行无法解决的难题。从技术层面上来看，去中心化属性的实现是基于区块链技术，这项技术是结合加密、解密、链表、Hash处理、树及工作量证明来实现，当前这项技术已成为应用研究的热点领域。各国政府、机构、公司已投入大量的人力、物力去研究、开发基于区块链技术的相关应用。

3.1.3. 定义

业务流程图：使用图表的方式描述系统人员业务关系、内作业顺序、管理信息流向各单位。使用业务流程图，人们可以很方便地找出业务流程中的不合理之处。

时序图：是UML图中的一种。它通过时间顺序显示多个对象之间的动态协作。它可以表示对象之间发送消息的行为顺序

，当执行一个用例行为时，一个类操作或状态机中引起转换的触发事件与时序图中的每条消息一一对应形成映射。

数据词典：是有组织的系统,用来定义和管理数据库文件，是为了方便数据库的存取和控制,加强系统的数据管理。

Json(JavaScript Object Notation):是一种轻量级的数据交换格式。基于JavaScript的一个子集。是一种独立于语言的文本格式，使用了类似于C语言家族包括C, C++, C#, Java, JavaScript, Perl, Python等的习惯。易于机器解析和网络传输，方便人们阅读编写。是一种理想的数据交换语言。

3.2. 任务概述

3.2.1. 系统目标

该系统以区块链为基本知识，以比特币运行特点为研究对象，目标是搭建一个简易的区块链交易系统。将区块链中的重要部分及其运行原理使用Python代码进行模拟实现。系统可以模拟区块链的运行机理。

3.2.2. 运行环境

开发工具 ：IntelliJ IDEA 2017.2.5， Anaconda

Python环境 ：Python3.6， Flask， Hashlib， Json， Request

运行环境 ：Chrome浏览器（IE、Firfox亦可）

3.3. 功能需求

3.3.1. 功能模块划分

通过上面的比特币系统功能介绍，相应，本文系统主要功能需求分为“比特币地址之间交易”，“查看整体区块链信息”，“查看区块信息”，“挖矿”，“查看未确认交易信息”，“自动生成比特币地址”等功能。这些功能通过一个主程序进行统一调用。

图3.1 模拟系统功能

3.3.2. 功能描述

点击运行项目后在浏览器输入相应地址“localhost:62001”,即可访问到项目运行界面。

点击随机生成比特币地址即可生成一个比特币地址。

输入两个比特币地址以及输入交易的比特币数量即可完成交易，交易结束后可以查看交易信息

点击查看未确认交易信息即可查看当前还未被矿工打包的交易的信息

点击挖矿即可实现模拟挖矿，自动求解工作量证明机制，打包未确认交易信息。

点击查看区块链信息即可查看所有的区块链中的所有区块。

点击查看区块即可查看要查询的区块。

3.3.3. 对象结构描述

在提到比特币信息的时候，比特币的区块以及比特币的交易都有着一定的结构，那么区块链系统中的对象结构借用比特币系统的结构，并对比特币系统的结构做一定的简化，一遍更容易理解。

表3.1 交易结构

名称描述

Version 版本号（默认为0x01000000）

Time 时间戳

sender 发送方

recipient 接收方

anoumt 交易金额

表3.2 区块结构

名称描述

Version 版本号（默认为0x01000000）

Previous_hash 前一个区块的Hash

Merkle_root 交易列表组件的Merkle树的根Hash

index 区块在区块链中的索引

timestamp 时间戳

Tx_n 包含的交易的数量

tx 具体的交易列表的Hash索引

Difficulty_hash 此区块对应的难度值

proof 随机数，通过挖矿获取

3.4. 功能执行流程

3.4.1. 交易

参数：交易方、接收方、交易金额

由于是模拟系统，不会有参与者参与系统的运行，因此这里只模拟比特币系统的单个交易本身的结构，而没有模拟交易之间的结构信息。因此这里的交易作为模拟交易，只简单构建交易信息，既不会像比特币那样检索UTXO，也不会消费和构建UTXO，因此，最简单的交易只需要发送方，接收方，金额即可构建一个简单的交易。

图3.2 交易模块流程图

图3.3 交易模块时序图

3.4.2. 挖矿

主界面输入挖矿者地址后点击挖矿后，主模块会调用挖矿模块执行工作量证明机制算法，当该算法找到最佳proof值后即可证明挖矿成功，那么挖矿者即可以创建一个交易并添加到当前交易列表并将这个交易列表进行打包并添加到区块链后面。最后以Json格式返回新挖取到的区块的信息。

图3.3 挖矿流程图

图3.4 挖矿时序图

第四章系统实现

4.1. 概述

4.1.1. 编写目的

本章用于描述区块链模拟系统的架构设计及其功能实现。决定区块链模拟系统应该从比特币系统中汲取那些值得借鉴的内容，并使用这些内容组建起区块链模拟系统并使其运行。并对这些设计内容进行详细描述。

4.2. 功能模块

4.2.1. 主模块FlaskServer

定义系统的主要功能模块，提供各种功能实现模块的接口。模块提供接口如下：

1. Index

返回访问系统的主界面。主界面包含系统的各项操作

接口：/

2. full_chain

获取全部区块链区块信息接口

接口：/chain

Method: GET

3. mine

调用挖矿模块执行挖矿操作

接口：/mine

Method： POST

参数：address（挖矿成功后为该地址添加一定数量的虚拟币）

4. get_not_sure_tx

获取为被矿工打包的交易

接口：/tx/not_sure

Method: GET

5. new_transactions

指定地址之间的交易

接口: /tx/new

Method: POST

参数： sender（发送方）、recipient（接收方）、amount（交易总额）
4.2.2. 挖矿模块mine
挖矿模块，提供挖矿的一些算法
1. Proof_of_work
工作量证明机制算法，通过不断Hash与Target比较的方式找到最佳的proof 值。
参数：
Tx[] ： 未被打包的全部交易信息
Version: 版本号
Prehash: 前一个区块的hash值
2. Valid_proof
验证proof值是否正确
参数：
Version：版本号
Mercle_root: 所有交易组成的Mercle数的根rootHash值
Prehash: 前一个区块的hash
4.2.3. 交易模块Tx
1. New_transaction
新建交易，在列表中添加新交易后，返回交易被添加到的区块的索引，这个索引也就是下一个要被挖的区块的索引。
参数：
Block_chain: 总的区块链
Sender：发送方地址
Recipient：接收方地址
Amount：发送的金额。
2. Tx_hash
获取交易信息的Hash。这个Hash是存储在交易信息中的Hash字段的信息。同时Hash字段也是检索交易的重要字段。
参数：
Tx: 要获取Hash的交易内容
3. Get_txs_merkle_root
获取交易列表的Merkle树的根Hash值。因为构建区块时需要所有交易的Merkle树的根Hash值。
参数：
Txs：交易列表
4.2.4. 区块模块Block
1. new_block
创建新模块，挖矿者挖矿成功后会调用这个方法来进行创建新的模块并将之添加到整体区块链中。
参数：
Block_chain:区块链
Txs：新区快包含的交易列表
Proof：挖矿成功找到的随机数
Timestamp:时间戳
Previous_hash:上一个区块的Hash
2. Block_head_hash
计算对应区块区块头Hash
参数：
block 区块信息
4.2.5. Unit（工具）模块
1. Constant
定义系统中定义的各种参数的模块
2. Hash
用于项目中各种消息摘要算法模块
3. Param_check
用于检测参数是否合法的模块
4.3. 功能模块代码实现
4.3.1. FlaskServer

文件路径：blockchainmodel/FlaskServer.py
对应Index界面文件路径：
blockchainmodel/templates/index.html
blockchainmodel/templates/style.css
==========

```
from flask import Flask, jsonify, render_template
from blockchainmodel.util.param_check import param_check, get_params
import blockchainmodel.util.constant as constant
from blockchainmodel import MineModel
from time import time
from blockchainmodel import Tx
from blockchainmodel import Block
from blockchainmodel.Blockchain import Blockchain
app = Flask(__name__)
# 初始化Blockchain类
blockchain = Blockchain()
# 获取全部区块链信息
@app.route('/chain', methods=["GET"])
def full_chain():
 response = {
 'chain': blockchain.chain,
 'length': len(blockchain.chain)
 }
 return jsonify(response), 200
# 获取未被打包的交易列表
@app.route('/tx/not_sure', methods=["GET"])
def get_not_sure_tx():
 response = {
 'tx': blockchain.current_transactions,
 'length': len(blockchain.current_transactions)
 }
 return jsonify(response), 200
# 用户交易实现
@app.route("/tx/new", methods=["POST"])
def new_transactions():
 param_list = ['sender', 'recipient', 'amount']
 params = get_params(param_list)
 # Check that the required field are in the POST'ed data
 is_check, error_param_index = param_check(param_list)
 if is_check:
 return "Miss Param" + param_list[error_param_index], 400
 # Create a new Transaction
 new_transaction_index = Tx.new_transaction(block_chain=blockchain, sender=params[0],recipient=params[1],
amount=params[2])
 response = {"message": f'Transaction will be added to Block {new_transaction_index}'}
 return jsonify(response), 201
@app.route("/")
def index():
 return render_template('/index.html')
# 挖矿段的实现
"""
1.计算工作量证明
2.奖励矿工，新增一次交易就赚一个币
```

3.将区块加入链就可以新建区块
"""
```
@app.route("/mine", methods=["POST"])
def mine():
 # get mine address, if mine success, this address will get some coins
 mine_address = get_params(['address'])[0]
 # check address
 if param_check([mine_address])[0]:
 return "Miss Mine Address", 400
 last_block = blockchain.last_block
 previous_hash = Block.block_head_hash(last_block)
 # We run the proof work algorithm to get the next proof...
 proof = MineModel.proof_of_work(constant.VERSION, blockchain.current_transactions, previous_hash)
 # We must receive a reward for finding the proof
 # The sender is "0" to signify that this node has mined a new coin
 Tx.new_transaction(block_chain=blockchain,sender=0,
 recipient=mine_address,amount=1)
 block = Block.new_block(block_chain=blockchain,
 txs=blockchain.current_transactions,proof=proof,
 timestamp=time(),previous_hash=previous_hash)
 response = {
 "message": "New Block Forged",
 "index": block['index'],
 'tx': block['tx'],
 'proof': block['proof'],
 'previous_hash': block['previous_hash']
 }
 return jsonify(response), 200
```
4.3.2. MineModel
文件路径: blockchainmodel/MineModel.py
```
=====================================================================
from blockchainmodel.util import hash
from blockchainmodel.util import constant
from blockchainmodel import Tx
def proof_of_work(version, txs, pre_hash):
 """
 简单的工作量证明机制算法
 Simple Proof of Work Algorithm
 - Find a number p' such that hash(pp') contains leading 4 zeroes, where p is the previous p'
 - p is the previous proof, and p' is the new proof
 :param pre_hash:
 :param txs All of not sure tx_hash
 :param version:
 :return:<int>
 """
 proof = 0
 merkle_root = Tx.get_txs_merkle_root(txs)
 while valid_proof(version=version, pre_hash=pre_hash,
 merkle_root=merkle_root, proof=proof) is False:
 proof += 1
 return proof
def valid_proof(version, pre_hash, merkle_root, proof):
 """
```

```
检测随机数的hash值是否满足工作量证明机制算法
 Validates the Proof: Does hash(last_proof, proof) contain 4 leading zeros?
 :param pre_hash:上一个区块的算法
 :param version:区块链算法版本号
 :param proof:<int> Current Proof 当前要验证的随机数
 :param merkle_root 交易列表的merkle树的根hash值
 :return:<bool> True if correct, False if not...
 """
 target = 2 ** (256 - constant.DIFFICULTY_BITS)
 guess = version + pre_hash + merkle_root + hash.hash_proof(proof)
 guess_hash = hash.hash2(guess)
 return int(guess_hash, 16) < target
```

4.3.3. Tx

文件路径：blockchainmodel/Tx.py

```
==============================================================
from time import time
import hashlib
import json
def new_transaction(block_chain, sender, recipient, amount):
 """
 Adds a new transaction to the list of transactions
 在列表中添加新交易后，返回该交易被加到的区块的索引，也就是下一个要挖的区块
 :param block_chain
 :param sender:<str> Address of the Sender
 :param recipient:<str> Address of the recipient
 :param amount:<int> Amount
 :return:<int> The index of the Block that will hold the transaction
 """
 transaction = {
 'ver': 1,
 'time': time(),
 'sender': sender,
 'recipient': recipient,
 'amount': amount
 }
 transaction['hash'] = tx_hash(transaction)
 block_chain.current_transactions.append(transaction)
 return block_chain.last_block['index'] + 1
def tx_hash(tx):
 """
 获取交易的Hash值
 :param tx:<dict> Transaction
 :return:<str> Hash result
 """
 tx_str = json.dumps(tx, sort_keys=True).encode()
 return hashlib.sha256(tx_str).hexdigest()
def get_txs_merkle_root(txs):
 """
 组件交易记录的merkle树，并返回根Hash
 :param txs:交易列表
 """
 txs_hash = [tx["hash"] for tx in txs]
 tmp_txs = []
```

```
while len(txs_hash) >= 2:
index = 0
while index < len(txs_hash):
hash1 = txs_hash[index]
index += 1
if index >= len(txs_hash):
hash2 = ""
else:
hash2 = txs_hash[index]
index += 1
p_hash = hashlib.sha256(hash1 + hash2)
tmp_txs.append(p_hash)
txs_hash = tmp_txs[:]
tmp_txs = []
if tmp_txs:
return tmp_txs[0]
else:
return ""
```

4.3.4. Block

文件路径：blockchainmodel/Block.py

```
==========================================================
from blockchainmodel.util import constant
from blockchainmodel.util import hash
from blockchainmodel import Tx
def new_block(block_chain, txs, proof, timestamp, previous_hash=None):
merkle_root = Tx.get_txs_merkle_root(txs=txs)
block_chain.transactions += txs
# get txs the hash columns, make a list as block params
txs_hash = [tx["hash"] for tx in txs]
block = {
'version': constant.VERSION,
'previous_hash': previous_hash or block_chain.chain[-1]["hash"],
'merkle_root': merkle_root,
'index': len(block_chain.chain) + 1,
'timestamp': timestamp,
'tx_n': len(txs_hash),
'tx': txs_hash,
'difficulty_target': constant.DIFFICULTY_BITS,
'proof': proof
}
# block_hash = hash.hash_block(block)
# block["hash"] = block_hash
# Reset the current list of transactions
block_chain.current_transactions = []
block_chain.chain.append(block)
return block
def block_head_hash(block):
"""
Creates a SHA-256 hash of a Block Header
:param block:<dict> Block
:return:<str> Hash result
"""
# We must make sure the Dictionary is Ordered, or we'll have inconsistent hashes
```

```
 ver = block["version"]
 pre_hash = block["previous_hash"]
 merkle_root = block["merkle_root"]
 proof = hash.hash_proof(block["proof"])
 print(ver + pre_hash + merkle_root)
 return hash.hash2(ver + pre_hash + merkle_root + proof)
```

4.3.5. BlockChain

文件路径：blockchainmodel/BlockChain.py

```
==============================================================
from time import time
from urllib.parse import urlparse
import requests
from blockchainmodel import MineModel
from blockchainmodel import Block
class Blockchain(object):
 """
 负责管理链。
 用来存储交易信息
 一些帮助方法来将新区块添加到链中
 """
 def __init__(self):
 # Save the Blockchain
 self.chain = []
 # Save the current transactions
 self.current_transactions = []
# Save all of the transactions
 self.transactions = []
 # Create the genesis block(创建创世区块）
 Block.new_block(
 block_chain=self,
 txs=self.current_transactions,
previous_hash="0000000000000000000000000000000000000000000000000000000000000000",
 proof=100,
 timestamp=str(time()).split(".")[0])
@property
 def last_block(self):
 # Returns the last Block in the chain
 return self.chain[-1]
```

4.3.6. Unit

1. Constant

文件位置：blockchainmodel/unit/constant.py

```
==============================================================
# 定义版本号
VERSION = "01000000"
# 定义难度值，这里使用固定的值
DIFFICULTY_BITS = 16
```

2. Hash

文件位置：blockchainmodel/unit/hash.py

```
==============================================================
from hashlib import sha256
 def hash2(src):
 """
 两次hash，用于挖矿中hash头的生成
```

```
:param src:<str> 要进行hash的字符串
:return:hash结束的字符串
"""
src_bytes = bytes.fromhex(src)
hash1 = sha256(src_bytes).digest()
result = sha256(hash1).hexdigest()
return result
def byte2hex(b):
"""
将byte类型的数值或字符串转换为16进制
:return:返回十六进制字符串
"""
return hex(int(b))[2:]
def hash_proof(proof):
# 将随机值进行Hash
hex_proof = byte2hex(proof)
lens_sub = 8 - len(hex_proof)
if lens_sub <= 0:
return hex_proof[:8]
return "0" * lens_sub + hex_proof
```

3. ParamCheck

文件位置：blockchainmodel/unit/param_check.py

```
==================================================================
from flask import request
def param_check(params):
"""
检查所有的参数值，如果参数为Null或者参数长度为0则返回false，
:param params:要检测的参数列表
:return:参数列表是否有误，以及有误的参数的位置
"""
for param in params:
if param is None or len(param) == 0:
return True, params.index(param)
return False, 0
def get_params(params_str):
params = []
for params_str in params_str:
param = request.form.get(params_str, None)
params.append(param)
return params
```

4.4. 界面及功能展示

4.4.1. 主界面

运行项目代码。进入浏览器输入地址http://127.0.0.1:5002/即可打开项目界面。

图4.1 主界面

界面中的操作包括“交易”，“挖矿”，“获取全部交易记录”，“获取全部未确认交易信息”。

4.4.2. 交易功能

在测试交易一栏，如果有任意一栏没有填写信息而点击确认交易的话，会提示缺少参数，如下，如果没有填写发送者地址：

图4.2 缺少参数

在测试交易一栏填写发送者地址：“test1”，填写接收者地址“test2”, 填写交易金额“1”。点击确认交易。如果交易成功即可显示如下提示：提示中将会显示这笔交易将会被打包进哪一个区块。

图4.3 交易成功提示

4.4.3. 查看未被挖矿挖取的交易

同时，我们点击其他操作中的“未确认的交易”，即可查看所有未被打包的交易的信息列表。这里会显示有多少交易没有被打包。我们刚刚交易的信息就包含在这个未被确认的交易信息列表中。

图4.4 查看未被打包的消息

4.4.4. 查看创世区块

在执行挖矿之前，我们来看一看“查看所有交易信息”即可查看当前区块链的信息，如果系统运行之前没有进行挖矿，这里面将只有一个创世区块，即区块链中的第一个区块。点击“查看所有交易信息”即可见如下信息。

图4.5 区块链信息（创世区块）

4.4.5. 挖矿

这时候在挖矿模块填写挖矿者地址“test1”，点击挖矿按钮，即可进行挖矿，执行工作量证明算法。一段时间的运算后。找到最佳的proof值，即可将之前未被打包的交易进行打包。执行结束之后即可看见下面的提示，会显示出当前挖出的区块的详细信息，我们一开始进行的交易就在这个区块里面，另外，这个区块里面还有另一笔交易，即系统赠予挖矿者币的一笔交易，如下：

图4.6 挖矿成功

可以看到，上面的交易一共有两笔，第一个的Hash为：0f7f2dd23b849054f62afa5b2b8f9637e5335da402223a4561457f4cfc4a0d10，和一开始的交易的Hash值是一样的。这时候再查看未打包的交易，则可以看到未被打包的交易已经没有了。

图4.7 再次查看未被打包交易

4.4.6. 查看区块链信息

这时候再次查看区块链信息可以看到有除了开始的创世区块又添加了一个新的区块，而这个新添加的区块就是我们刚刚挖出来的区块。

图4.8 再次查看区块链信息

第五章总结

区块链系统是近几年极具热点性质的技术，从中本聪提出比特币系统开始到现在区块链技术已经不单单是应用于金融与货币贸易方面，可以展望的一方面是：未来的各行各业都可能会应用区块链技术从而来完成技术上面的转型。

应用区块链系统的安全性与隐私性特点会让一些现行系统中的安全性问题迎刃而解。例如现在的学生信息管理系统，学历信息管理系统等等。因此可以发现区块链技术的未来发展方向是方方面面的。

在这一方面来说研究基于区块链的虚拟货币挖矿技术有着十分重要的现实意义。同时又是以比特币这个区块链技术最成熟的项目来进行研究。正是由于未来区块链技术将会在社会的方方面面起到举足轻重的作用，因此有这么一套以区块链原理为基础的项目就显得尤为重要，因为随着技术的进步，那些以技术基础而构建的项目只要稍加修改就可以应用与很多方面。而不只是仅限于某一方面。例如比特币系统，如果要想把比特币系统修改成信息管理系统，相信还是需要修改很多东西，甚至要删除很多无关的代码。

因此，作为研究区块链挖矿的项目，希望能在未来起到一个基石作用，能够很方面的进行扩展，从而可以以零成本的方式实现某一功能，这与很多开源的编程框架有着共通的思想。

由于精力有限，对于系统的某些地方可能会考虑不周会有潜在的BUG，尽管系统的实现使用了很多Python现有的一些框架。因此现在项目已经开源到了GitHub上面，希望可以让项目得到更好的完善。

致谢

在论文的末尾，本人十分有成就感，而且给常高兴。通过大学四年的学习，我不仅扩充了自己的知识面，同时学习了许多专业之内和专业之外的理论与技能，同时也养成了自主学习的能力，掌握了一些学习方法与习惯。另外也自学了有关软件开发与计算机网络方便的相关知识，这对于我完成本篇论文有着莫大的帮助。

在此，非常感谢我的指导老师郤非老师。郤老师平日对我细心的指导以及平日里给我的帮助让我在计算机方面的理论知识的学习有了正确的方向和动力。郤非老师是我的本科生导师，同时也是我的C++以及数据结构的任课老师，在软件开发方面造诣颇深。能跟着郤非老师做毕业设计以及在大学期间跟随郤老师做一些软件工作是件非常开心的事情。这让我学到了很多东西，让我提高了许多。

另一方面，我要感谢许多的开源项目，比特币的开源代码能让我在论文研究中有据可循，另外还有BitCoin4J的开源代码，以及有关比特币系统介绍的博主的无私分享让我的论文能够顺利的完成。

参考文献
[1]《比特币白皮书：一种点对点的电子现金系统》 中本聪
[2]《精通比特币（第二版）》
[3]《区块链技术指南》 邹均等著 2016 机械工业出版社
[4] 从零到一用 Python 写一个区块链

附录
1. 相关Api文档
Python文档
地址：http://docs.oracle.com/javase/8/docs/api/
Flask文档
地址：http://docs.jinkan.org/docs/flask/
比特币项目
地址：https://github.com/bitcoin/bitcoin
2. 项目开源
本项目基于Apache Licene 2.0 协议开源于GitHub
项目地址：https://github.com/MengFly/blockchainmodel