



北京大学

## 硕士研究生学位论文

题目： 实时区块链实名交易监督系统  
的设计与实现

姓      名： \_\_\_\_\_

学      号： \_\_\_\_\_ 1601210903

院      系： \_\_\_\_\_

专      业： \_\_\_\_\_

研究方向： \_\_\_\_\_

导      师： \_\_\_\_\_

二〇一八年六月



## 版权声明

任何收存和保管本论文各种版本的单位和个人，未经本论文作者同意，不得将本论文转借他人，亦不得随意复制、抄录、拍照或以任何方式传播。否则一旦引起有碍作者著作权之问题，将可能承担法律责任。



## 摘要

比特币是一个集成网络学、密码学、货币银行学并以区块链为基础的加密货币。有别于其他加密货币的是比特币特有的去中心化与匿名化。区块链技术虽能保有交易信息的透明性与交易数据的不可变动性，但也因为其匿名之特性存在着三项问题。第一，在比特币交易系统中，没有落实实名制，造成流动资金的不透明性。其次，现今的国家并无支持以比特币交易相关的支付系统或是制定出相关的税务标准，使得政府无法从加密货币这方面的金融交易获得税收。第三，现有的比特币交易模型皆为匿名与匿名之间的交易，并无开立交易凭据，亦无法保障消费者权益。

本论文设计与实现一个比特币的实时交易监督系统-BRTMS，论文工作包括分析五种交易模型，发现由匿名顾客支付给实名商家的交易监督可同时实践保障消费者权益、保护顾客隐私以及使政府可以课征税收。由需求分析将系统模块划分为用户注册与登入、产品管理、职工管理、商家交易管理，以及顾客交易管理五种模块，同时采用 Java 编程语言和比特币开源钱包，以五种模块为基础实现主要系统，以及商家和商品信息管理、商家手持移动装置收款及交易、客户端行动支付和交易三个子系统，并且为解决区块链交易速度缓慢问题引入多重签章算法构建实时监督系统。实现系统后，对本系统进行功能测试，且对原始监督系统与有多重签章算法支持的实时监督系统进行性能测试与分析。比特币通过本系统的实时交易监督，当顾客使用手持移动装置客户端可拥有详细的交易明细，也保障了消费者权益；借由商家和商品信息管理子系统，让商家可以容易进行商品管理及库存管理；在政府端实现多重签章算法，让商家可避免双重支付攻击，且大幅减少交易确认时间，更能使政府主管机关有效获得税收。

**关键词：**比特币，区块链，多重签章算法



# **Design and Implementation of Real-time Blockchain Real-name Transaction Monitoring System**

Chen Po Wei (Data Mining and Business Intelligence)

Directed by Prof. Lijie, Lecturer Liujing

## **ABSTRACT**

Bitcoin is a blockchain-based cryptocurrency that integrates networking, cryptography, and money banking. Unlike other cryptocurrencies, Bitcoin has unique decentralization and anonymization. Its applied blockchain technology can preserve the transparency of transaction information and the irreversibility of transaction data, but it also has three problems because of its anonymous nature. First, identifying users in Bitcoin trading has not been implemented. Therefore, it results in the opaqueness of transaction liquidity. Second, current countries don't support the cryptocurrency payment systems for the Bitcoin, or formulate relevant tax standards, making it impossible for the government to obtain tax revenues from cryptocurrency transactions. Third, the existing Bitcoin transaction models are all based on anonymity. There are no receipts and even the protection of consumer right.

This thesis designs and implements a Bitcoin real-time transaction monitoring system-BRTMS. The research activities of this thesis include analysis of five trading models , finding the supervision of transactions between the anonymous customers and the real-name businesses, in that way can simultaneously protect consumer right, customer privacy, and enable the government to collect taxes. From the system requirements analysis, the system is divided into five modules: user registration and login, product management, employee management, merchant transaction management, and customer transaction management. Besides, the Java programming language and Bitcoin open source wallet are applied to realize the main system based on the five modules, as well as the three subsystems including store and merchandise information management subsystem ,store mobile payment collection and transaction subsystem, client mobile payment and transaction subsystem. Furthermore, to solve the problem of slow blockchain transactions, multiple signatures algorithm is introduced to build the real-time monitoring system. After the implement of the system, the functional testing is conducted. Then, the performance testing and analysis between the original supervisory system and the

real-time supervisory system with multiple signatures algorithm are performed. With these evaluation results from the system, Bitcoin can let the customers know more detailed information of transaction when using the mobile device subsystem. It can also protect the consumer right. Meanwhile, the store mobile payment collection and transaction subsystem can help businesses easier to carry out commodity management and inventory management. Implementing multiple signatures algorithm on the government side prevents businesses from double-spending attacks and drastically reduces the time required in transactions. Finally, the system can even enables government agencies to effectively obtain tax revenues.

**KEYWORDS:** Bitcoin, Blockchain, Multiple signatures algorithm

# 目录

|                               |           |
|-------------------------------|-----------|
| <b>第一章 绪论</b>                 | <b>1</b>  |
| 1.1 选题背景 . . . . .            | 1         |
| 1.2 研究目标与内容 . . . . .         | 10        |
| 1.3 论文组织结构 . . . . .          | 11        |
| <b>第二章 相关理论与技术</b>            | <b>13</b> |
| 2.1 比特币地址 . . . . .           | 14        |
| 2.2 区块链 . . . . .             | 24        |
| <b>第三章 系统需求分析</b>             | <b>27</b> |
| 3.1 交易模型分析 . . . . .          | 27        |
| 3.2 功能性需求分析 . . . . .         | 31        |
| 3.3 非功能性需求分析 . . . . .        | 34        |
| <b>第四章 系统概要设计</b>             | <b>37</b> |
| 4.1 BTMS 架构与运作流程 . . . . .    | 39        |
| 4.2 实时 BTMS 架构与运作流程 . . . . . | 42        |
| <b>第五章 系统详细设计与实现</b>          | <b>45</b> |
| 5.1 BTMS 数据库设计 . . . . .      | 45        |
| 5.2 系统模块设计 . . . . .          | 47        |
| 5.3 系统实现 . . . . .            | 61        |
| <b>第六章 系统测试</b>               | <b>65</b> |
| 6.1 功能测试 . . . . .            | 65        |
| 6.2 性能测试 . . . . .            | 74        |
| <b>第七章 总结与展望</b>              | <b>79</b> |
| <b>参考文献</b>                   | <b>81</b> |
| <b>致谢</b>                     | <b>85</b> |
| <b>北京大学学位论文原创性声明和使用授权说明</b>   | <b>87</b> |



# 第一章 绪论

现金法定货币，交易凭据及交易数据库存在着一些缺点。如现金很难杜绝假钞的横行，交易凭据有着伪造的可能，在交易数据库中信息不一致，数据库被 DDOS 攻击，交易数据被窜改，数据库损毁，都是在传统交易过程中曾出现过的窘境。

于 2009 年加密货币 - 比特币<sup>[1]</sup> 的问世，以密码学、网络学与货币银行学为基础创建了新一代的网络货币。各式网络货币中又以比特币最为广泛使用，其防堵被窜改、公开交易数据查看、使用者具匿名性、自动运作不须人为运营等等多项特性深受现今用户喜爱。至今区块链技术已成为 IBM、摩根大通、微软、谷歌与英特尔等重点开发项目，被视为改善银行运作效率、降低运营成本、提升信息安全、创建公开数据的最佳方法。为解决现金、收益及交易数据库存在之问题，本文采用以区块链为基础的加密货币比特币为基础，进行商业化收银系统开发。不仅是基于比特币算法稳定、交易公开透明、不可被窜改等特性，同时本论文更加入监督标签，促使在匿名交易转为部分实名交易之过程中，监管部門能有更好的新兴货币技术的提升，亦可创建自动化的税务审查机制，大幅降低人事成本，进而实现提升交易系统信息之可靠度与其稳定性。

## 1.1 选题背景

追溯着加密货币市场的演进，于 2009 年时，比特币并非第一个加密货币，在比特币之前已经有着很多类似的加密货币开发实验，但是一直无法做出一个稳定点对点式的电子现金系统，关于其制作瓶颈之部分将于后段章节阐述。在比特币稳定发展之后，有着许多对比特币有兴趣的研究者，以稳定的比特币系统为基础修改了许多基本的协议。于 2011 年相继创造出了货币，将其称之为山寨币。山寨币早期较为著名包括有莱特币 (Litecoin, LTC)<sup>[2]</sup>、狗币 (Dogecoin, DOGE)<sup>[3]</sup>、域名币 (Namecoin, NMC)<sup>[4]</sup>，于 2014 年也有人认为比特币挖矿使用到了大量的哈希运算，这样的大量运算也浪费了许多的社会资源，进而开发出较具意义的工作量证明挖矿算法，其中较为著名的如素数币 (Primecoin, XPM)<sup>[5]</sup>。于 2015 年底也诞生了现在最为著名的以太坊经典 (Ethereum Classic, ETC)<sup>[6]</sup>、以太坊 (Ethereum, ETH)<sup>[7]</sup>，以太坊最重大突破设计在于将编程语言虚拟机移植到了区块链架构上，这使得区块链技术不再仅止于点对点的电子现金系统，也创造出了属于以太坊的编程语言 Solidity<sup>[8]</sup>，使以太坊在虚拟机 (Ethereum Virtual Machine, EVM)<sup>[9]</sup> 中可以使用 Solidity 创建智能合约，合约可以建构去中心化的应用程序，如去中心化的交易所，将交易所去中心化可以有效的防治 DDOS 攻击<sup>[10]</sup>，降低交

交易所因为黑客攻击而倒闭的可能性。

### 1.1.1 加密货币市场

加密货币中最具代表性的是比特币，但除了比特币之外也存在许多模仿比特币的加密货币，有的是为其利益，有的是鉴于比特币的各种不足，进而希望借由其他货币改善比特币不够完美之处。加密货币市场中有成千上万种的加密货币，其中较广为人知的加密货币会在 Cryptocurrency Market Capitalizations<sup>[11]</sup> 的排行榜中出现，截至 2018 年 2 月 8 日该排行榜已经收入了 1510 种加密货币。在 Cryptocurrency Market Capitalizations 统计的数据当中，可知整体的加密货币市场，如图1.1所示，于 2018 年 1 月 7 日创下了历史新高，加密货币市场的总市值也高达了 829,579,000,000 美金，相当于五兆人民币的总市值。

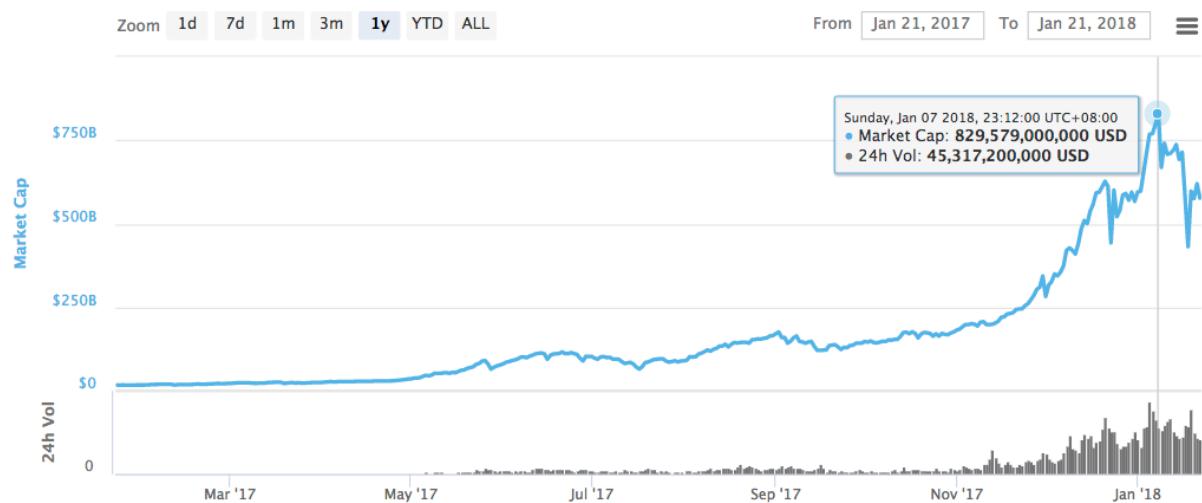


图 1.1 2017 年 1 月 21 日到 2018 年 1 月 21 日期间加密货币总市值走势图<sup>[11]</sup>

经由 Cryptocurrency Market Capitalizations 数据显示，整体加密货币市场自 2013 年起已经高达 150 亿美金，2014 年与 2015 年间总市值减少到近乎 2013 年的一半。针对比特币的价格波动，论文 "Have the security flaws surrounding Bitcoin effected the currency's value?."<sup>[12]</sup> 作出详尽的市场调研，致力于探讨在各个比特币市场大事件中对比特币价格的波动影响，针对影响的程度该论文给出影响指数，当中影响最为严重的是于 2014 年 2 月发生的日本交易所 Mt.Gox 倒闭事件，因为早期的加密货币市场中无完善的法律规范，各国对加密货币的接受度有所不同，日本对金融科技的接受度相较于较为开放的情况下成立了全世界第一家比特币交易所 Mt.Gox，也因为交易所不够普及，使得大部分的加密货币交易都集中在 Mt.Gox 交易所中，使 Mt.Gox 倒闭事件成为震荡市场价格重大因子之一，也造成 2014 与 2015 年的加密货币市场低迷。而在 2017 年，比特币又以 2016 年总市值之 35 倍的姿态攀上新高点，主要是因为美国最大的期权交易中心

芝加哥期权交易所（Chicago Board Options Exchange, CBOE）于 2017 年 12 月 10 日宣布支持比特币期货交易，此举将比特币价格推升到 20,000 美金的历史新高，图1.2为 2013 年至 2018 年间比特币总市值 K 线图。



图 1.2 2013 年至 2018 年间比特币总市值 K 线图<sup>[11]</sup>

### 1.1.2 加密货币的优势

于 2009 年 Satoshi Nakamoto 发布了比特币系统，成为全世界第一个加密货币的雏形。其透明的交易信息、区块链交易数据无法修改和删除、匿名与自治系统等特性，促使区块链技术冲破现存传统中心化之金融机构技术上的藩篱。以下将逐一说明加密货币的七项优点，分别为区块链结算系统不间断的运行、远距离支付、货币为用户持有、开放和透明的交易信息、区块链交易数据无法修改和删除、交易匿名性以及自治性系统。

第一，区块链结算系统不间断的运行。基于区块链技术与点对点网络的架构，以比特币为例，自 2009 年至今，所有的比特币交易事件皆会存储在比特币区块链当中，区块链既无法删除也无法修改，比特币区块链会以点对点网络的方式存储在比特币网络中的全节点<sup>[13]</sup>，目前比特币网络中的全节点高达 11147 个。与传统中心化的银行数据库相比，可能会因为银行的服务器维护，导致交易无法顺利进行，甚至可能有黑客的入侵导致银行或是个人资产有重大的损失。点对点网络提供稳定的数据库元数据，不

会因为数据库的停机而无法继续使用，实现其 24 小时不间断之运作。

第二，基于点对点网络架构完成远距离支付。于跨国汇款从美国转帐至中国一百万美金的场景中，需要经过的手续较为繁琐，资金有可能需要经过多个国家才可以抵达目的地，在经过各个国家的过程中，需要支付各国的手续费，也需要等待各个国家办理该业务的时间，即使当资金顺利抵达了目的地银行，目的地银行也需要花将近三至五日的工作日确认该笔金额的来源。届时领款人亦需要前往银行核实完整的身份验证、解释资金用途，才得以领取这笔跨国资金。比特币系统当中，有着 24 小时不间断运作的优点，也因为点对点网络架构，使得比特币无需经由传统金融机构繁琐的步骤完成国际汇款，于比特币系统中无系统壅塞的情况下，平均 10 分钟即可入帐，实现其短时间内即可完成远距离支付之运作。

第三，加密货币为用户持有。传统的金融体系中，资金的存储、流动往往需要经过银行，用户将所有的资产存入银行，拿到的是一串数字的银行余额，银行是一个中心化的机构，有着最高的权利。中央社的新闻<sup>[14]</sup>指出，台湾各地于 2017 年接连于土地银行、日盛银行、彰化银行、京城银行、兆丰银行皆传出银行行员监守自盗的行为，总金额高达一亿三千万新台币。在比特币系统中，比特币有如金币般存放在个人的比特币地址当中，用户为真实持有着货币，即使是比特币系统亦无权利动用该笔比特币资产，唯有比特币地址的私钥持有者，才可以转移该笔比特币地址中的比特币资产。

第四，公开的交易信息。基于区块链架构，所有的交易信息皆以公开的方式存储于区块链中，并且可信任与方便的取得元数据，以下将针对可信任与元数据进行探讨。

1. 可信任：在公有链的基本架构上，所有的交易记录都是公开透明的存储在区块链当中，比特币网络的用户都可以查看该笔交易，所有人都可以检查每个交易记录的正确性，公开的交易信息亦提升交易数据之可信性。
2. 元数据：除了以区块链技术为基础建构出可信任的系统之外，开放和透明的特性让更多的开发商或新公司得以更容易获得交易的元数据。毕竟，在传统金融体系中，所有交易记录均由中央金融机构存储，从中央金融机构提取原始交易信息并不容易，区块链的开放性和透明性促使金融公司降低了获取原始数据努力的门槛。公司或学者可以透过元数据制定出可视化的开发计划，甚至可以运用大量数据来分析前所未达的新价值观点。

第五，区块链交易数据无法修改和删除。在区块链结构中，通过严格验证的所有信息都记录在区块链中，且用户及系统平台都不赋与删除及修改之权限。根据区块链的特点，旧区块的哈希值在连接区块链的过程中，旧区块的哈希值会被存储在新区块。只要区块中的值被修改，即使仅有 1 bit 的变化，也会产出完全不同的哈希值，这也就是所谓的雪崩效应 (Avalanche effect)<sup>[15]</sup>。由于上述结构特性，区块链中所有的信息都

会被系统纪录且都不会被改变，倘若区块中记载的比特币交易在其中一个比特币全节点验证的结果被窜改，则该区块将不被比特币系统接受。因此，所有已经存储在区块链中的交易记录将不能被修改和删除，进而实现其架构安全之稳固。

第六，区块链系统中所有的用户皆为匿名。现今社会中，个人信息保护已成为企业最重要的课题。在区块链系统中创建的所有帐户都不会与真实世界中的实体创建直接关联，也因为没直接关联所以创建匿名。区块链系统中的所有帐户都是由匿名个体创建，匿名的设计可以有效保护消费者的隐私。然而，VISA 交易与比特币系统截然不同，在使用 VISA 支付系统前，用户必须向 VISA 公司的主机提交大量个人信息，这可能会产生个人信息泄露的风险。在区块链技术中，其匿名之特性可以有效地避免这个问题。

第七，自治系统。在区块链系统中，区块链的运作依赖于一些算法，包括共识算法。因此，在这种自治系统中，没有人（例如节点或矿工）可以直接改变系统运作的规则。如果在比特币系统中发现需要更正的严重错误，可以使用比特币改进提案（Bitcoin Improvement Proposals, BIP）<sup>[16]</sup> 升级比特币系统。在实施比特币改进提案之前，提议的比特币改进提案需要得到比特币系统中超过一定数量的矿工算力支持。由于这种以投票机制升级系统的门槛相当高，使得区块链系统通常不会有大的变化，因为变化不大也突显其相对稳定性。

### 1.1.3 加密货币的劣势

在区块链技术中，有着三项瓶颈，分别为每秒处理的交易量（Transactions Per Second, TPS）仅为 7 笔的限制、洗钱防治困难、低可扩展性，以下将逐一探讨。

第一，每秒处理的交易量上限仅为 7 笔。图1.3为国际上较为广泛使用的支付系统之每秒支持交易量比较图，以 VISA 为例，其以公司中心化运营的方式可以支持高达每秒 2,000 笔交易。但是以区块链技术为基础的比特币最大能够接受的每秒处理交易量仅为 7 笔。一般认为只要提升区块大小的限制，就可以提升每秒处理的交易量。提升每秒处理交易量的同时将产生下述的两项问题，分别为区块链成长速度过快造成节点崩溃以及区块同步延迟造成区块链分岔：



图 1.3 Bitcoin、Western Union<sup>[17]</sup>、PayPal<sup>[18]</sup> 以及 VISA 每秒支持交易量比较图<sup>[19]</sup>

图 1.4 比特币区块链成长走势图<sup>[20]</sup>

1. 区块链成长速度过快造成节点崩溃。区块链成长速度过快会造成比特币全节点不堪负荷：从 2009 年至今的比特币区块链大小已达到 188.89 GB，这样的成长速度因为比特币区块大小的最大值被设置为 1 MB。图1.4为过去比特币区块链大小，图中可以发现，于 2016 年开始，比特币区块链的成长速度为一直线，这表示着比特币网络中持续维持在供不应求的状况。为解决比特币每秒支持交易量上限的窘迫，现今对比特币的每秒处理交易量有许多优化的方案，其中包括解除比特币区块大小 1 MB 的限制。在一个区块上限为 1 MB 的限制下，满载的比特币系统中，比特币区块链平均每十分钟会增 1 MB，每小时会增加 6 MB，每天会增加 144 MB，每月会增加 4.2 GB，每年会增加高达 50 GB，要达到 1 TB 的区块链大小还需要 8 年，在 8 年后的未来存储 1 TB 的数据量应该不会有太大的负担。倘若解除 1 MB 的区块限制，在系统的每秒处理交易量看似可以接受更多的交易成倍成长，面临 1 TB 的比特币区块链数据会在更短的时间内出现，倘若存储区块链的成本超过了摩尔定律的成长曲线，会进一步造成用户自愿成为比特币全节点的意愿度降低，使得比特币网络的全节点数变少，导致比特币点对点网络逐渐转向中心化网络发展，失去一开始点对点网络的意义。
2. 造成区块链最新区块同步延迟而使区块链分岔。对于区块链的区块同步延迟同时也会造成比特币网络的影响，J. Göbel 于 "Increased block size and Bitcoin blockchain dynamics"<sup>[21]</sup> 有着详细的研究，在上修区块大小上限的议题上，用户因系统占用量提高，自愿成为比特币全节点意愿度下降，此举可能对比特币点对点网络建构

出的区块链同步上造成延迟，在 11147 个比特币全节点当中，平均每十分钟会有矿工于其中一个全节点生成一个最新的区块，该最新的区块会以点对点网络协议同步到其他 11146 个节点上。在比特币系统中，长年来的过程经验可以发现在矿工生成 1 MB 的区块后同步到全网节点可以在创造下一个区块之前完成。倘若将区块大小修改为 2 MB 或是更大，会使得比特币全节点的最新区块同步延迟现象更加明显，同步延迟会使得区块链分岔，造成 11147 个比特币全节点的信息不一致，进一步造成整个比特币点对点网络崩溃。

第二，洗钱防治困难。匿名性为比特币系统一大特色，比特币的地址生成的熵是 256 bits，乱数是在  $2^{256}$  的组态空间中随机选取，这样的地址与现实生活中的身份并无任何关联，使得黑市交易、洗钱防治变的困难，甚至有更为前沿的加密货币 Monero<sup>[22]</sup> 导入了环签章（Ring Signature）<sup>[23]</sup> 算法、Zcash<sup>[24]</sup> 导入零知识证明算法<sup>[25]</sup>，使得原本公开透明的区块链，变得无法查看，进而造成加密货币在洗钱防治上更加的困难。2017 年由 Thibault de Balthasar and Julio Hernandez-Castro 所提出的论文“An Analysis of Bitcoin Laundry Services.”<sup>[26]</sup>，致力探究比特币匿名交易下的资金流动模型，试图以机器学习的方法找出比特币洗钱模型作为洗钱的工具，图 1.5 为该论文针对黑市交易中的洗钱服务运营商 Darklaunder 进行洗钱机



图 1.5 Darklaunder 洗钱模型<sup>[26]</sup>

第三，低可扩展性。区块链架构为了防止各式的攻击，所以在架构订定以及接口设计都有着严谨的规范。以下将阐述造成比特币低可扩展性的原因：

1. 修改比特币协议制作添加外部信息的区块链：比特币区块链技术是一个严谨的架

构，倘若要创造可以支持外部信息的结构需要重新创造全新的加密货币，大部分的加密货币不支持外部输入，由于外部的信息输入无法保证其信息的正确性，进一步造成垃圾进垃圾出（Garbage In, Garbage Out, GIGO）的问题，倘若错误的信息存储在无法删除、修改的区块链下，只是强化该笔错误信息的错误。如食品履历区块链，致力于将食品生产到超市的过程逐一记录在区块链上，但如果一开始在输入信息时，无法保证其信息之正确性，则该食品履历区块链则毫无意义，错误的信息内容甚至可能使该错误被强化。

2. 于区块头或交易信息添加外部信息：比特币区块链上，可以添加一些信息于区块上，该信息会永久保存于区块链上，除了在区块上添加信息，在比特币单笔交易信息上，亦可填写一些私人信息，但这样的空间大小有限，且现今的比特币价格日趋上涨，比特币交易手续费是以单笔交易大小计算，这将使得在交易中添加些个人信息变得更加昂贵。

在比特币系统中，其区块链仅用于记录交易记录，不能扩展更多功能和应用程序。有很多开发者希望将比特币系统扩展到智能合约等其他应用程序。但是，后来发现改变原始比特币系统框架是具有挑战性的工作。因此，全球第二大加密货币以太坊（Ethereum, ETH）的作者 Vitalik 选择了创建了以太坊虚拟机（Ethereum Virtual Machine, EVM），而非改变原始的比特币系统框架。以太坊虚拟机所创建的智能合约可以在统一的以太坊平台上运行，而以太坊也借此突破了比特币之技术瓶颈。

#### 1.1.4 国际情势分析

比特币是一种全新的价值交换的媒介，因为区块链技术相当新颖，在各个国家所秉持的态度皆有所不同，大致可将各个国家对比特币的政策分为两大类，分别为接纳与禁止；在接纳加密货币的分类当中又分为欢迎、放任以及监管。

在欢迎、放任的政策下的国家包括伊朗、以色列、法国、美国、沙特阿拉伯王国、乌克兰以及新加坡，上述国家政策上皆已抱持着开放且不监管的方式接纳加密货币，这些国家认为加密货币对于国家金融科技的发展具有前景；在分类于接纳但是实施监管的国家中，包括韩国、菲律宾、英国、马来西亚、俄罗斯、日本以及香港地区皆认为加密货币技术可能存在的高风险，且针对加密货币的洗钱防制以及非法集资进行严格管理，也进一步制定相关的法律，使得国家可以应对新型加密货币的纠纷；在禁止加密货币的分类中包括津巴布韦、摩洛哥、印尼以及中国，这些国家认为比特币是一个非法的货币，因为涉及到洗钱问题、难以实施外汇管制。但是对于区块链技术层面上之探讨，在所有国家中都是以蓬勃发展的正面态度看待。上述各国政府对比特币态度如表1.1：

表 1.1 各国政府对比特币态度统整表

| 序号 | 国家      | 接纳              |                 | 禁止          |
|----|---------|-----------------|-----------------|-------------|
|    |         | 欢迎/放任           | 监管              |             |
| 1  | 伊朗      | 监管得当欢迎比特币发展     |                 |             |
| 2  | 以色列     | 欢迎：欲发展国际 ICO 中心 |                 |             |
| 3  | 法国      | 放任：与实体经济无关      |                 |             |
| 4  | 美国      | 暂不构成威胁          |                 |             |
| 5  | 新加坡     | 不监管但注意周边活动      |                 |             |
| 6  | 沙特阿拉伯王国 | 加密货币尚未成熟，不监管    |                 |             |
| 7  | 乌克兰     | 不属于货币           |                 |             |
| 8  | 韩国      |                 | 很快将监管交易所        |             |
| 9  | 菲律宾     |                 | 计划监管 ICO        |             |
| 10 | 英国      |                 | 考虑监管            |             |
| 11 | 马来西亚    |                 | 范式监管框架          |             |
| 12 | 俄罗斯     |                 | 2018 年 7 月前完成立法 |             |
| 13 | 日本      |                 | 任命加密货币监察长       |             |
| 14 | 香港      |                 | ICO 需受法规监管      |             |
| 15 | 津巴布韦    |                 |                 | 定法前，属非法     |
| 16 | 摩洛哥     |                 |                 | 禁止加密货币交易    |
| 17 | 印尼      |                 |                 | 2018 年前全面禁止 |
| 18 | 中国      |                 |                 | 禁止加密货币      |

## 1.2 研究目标与内容

比特币在各个国家的蓬勃发展且应用在相当多的领域，应用的领域包括交易所、去中心化交易所、兑币所、游戏点数、网络购物、资产的保存、募资以及远距离支付，甚至是各国的银行和金融机构都逐步投入人力及资金研究区块链相关技术，甚至是订定各国相关的法律。在比特币资产的流动上存在着许多的问题，第一，比特币的帐户是由乱数产生器生成，该比特币帐户与在现实生活中的用户并无直接关联，使得比特币洗钱变得更加盛行。第二，现今的比特币交易皆为匿名对匿名支付，并无正式的交易凭据开立，使得消费者在购物后，倘若商品存在问题，无法得到法律上的保障。第三，因为比特币是匿名支付给匿名的交易，使得政府无法从交易行为中进一步课征税收，滞碍国家经济发展。

本文将解决上述三项问题，设计与实现一个比特币的实时交易监督系统达到下列几点目标：

1. 导入匿名顾客支付给实名商家的交易模型到加密货币交易系统。
2. 设计与实现在加密货币交易中匿名顾客支付给实名商家的交易监督系统。
3. 于政府端导入多重签章算法，使比特币的实时交易监督系统比透过 Green Address 机构验证更加快速。
4. 利用多重签章算法的机制可以让商家预防比特币双重支付的攻击。
5. 商家和商品信息管理子系统让商家可进行库存管理及商家商品管理。
6. 实现于加密货币交易中，消费者保持匿名同时也保障消费者权益。
7. 通过对商家进行实名制，比特币的交易监督系统让政府主管机关可以有效获得税收。

为实现上述目标，本文首先将详细探讨区块链技术的优势，借由深度了解区块链技术的优势可以取之优点，如区块链是一个不间断、不可窜改、去中心化、公开数据库、稳定的系统。第二，探讨区块链技术的劣势，区块链技术并非完美，存在着洗钱、逃漏税、无法保障消费者权益以及交易速度慢的问题，借由了解问题，并设计系统解决。第三，交易模型分析，透过交易模型分析探讨在现金、电子货币以及加密货币存在的交易模型，并在诸多交易模型中发现，匿名支付给实名的交易行为，可以保障消费者隐私，同时保障消费者权益，更使得政府可以对商家的营收进行查看。第四，系统设计，完成交易分析，并着手设计数据库模型。第五，系统详细设计与实现，利用 Java 编程语言实现主系统以及三个子系统。第六，系统优化，引入比特币的多重签章算法，实现比特币的实时交易监督系统。第七，功能测试，测试以 Java 编写程序所有的功能是否能够完整运作。第八，性能测试，测试在未进行优化的系统与已经优化的系统两者之间性能的差异。

### 1.3 论文组织结构

在本节中将逐一说明本论文的组织结构共分为七章，分别为绪论、相关理论与技术、系统需求分析、系统概要设计、系统实现、系统测试以及总结与展望：

第一章：简述区块链技术解决现有数据库存在的信息不一致、信息安全问题以及数据库损毁问题。说明在加密货币市场中不仅只是比特币，亦有许多新颖技术为特色的货币竞争。简要探讨区块链技术的优点，逐一探讨比特币的缺点及区块链技术瓶颈。最后于研究目标与内容当中，阐述本文的待解决问题以及解决方法。

第二章：介绍比特币的发展，阐述比特币地址生成运用到的算法、生成过程以及多重签章算法。并详细说明区块链结构及所有字段变量与区块链相关的意义。

第三章：首先对五种交易模型进行分析，进而得知匿名对实名的交易，既可以保障消费隐私且兼具保有消费者权益，并透过需求分析探讨本系统需构建的功能模块。

第四章：概要设计比特币交易监督系统的架构，以及说明三个子系统的模块功能。设计本系统的数据库结构，设计 ER 实体联系模型（Entity-relationship model），并阐述本系统的运作流程。最后则将多重签章算法引入本系统，使得本系统成为比特币的实时交易监督系统。

第五章：详细设计商家和商品信息管理子系统、商家手持移动装置收款及交易子系统以及客户端行动支付和交易子系统的介面。

第六章：透过功能测试逐一查看所有功能是否能够正常运行，经由性能测试分析原始系统与优化后的系统性能之间的差异。

第七章：说明比特币区块链系统透过本文所提出之系统，可以达到保障消费者权益同时也兼顾保有顾客隐私，也使得政府可以透过该系统查看交易明细课征税收，而商家亦可对库存及产品进行管理，达到三赢的局面。



## 第二章 相关理论与技术

比特币（Bitcoin, BTC）是一个点对点式的电子现金系统，集成了非对称式密钥密码学（Asymmetric Key Cryptography）<sup>[27]</sup>、签章密码学（Signature Cryptography）<sup>[28]</sup>、零知识证明密码学（Zero Knowledge Proof Cryptography）<sup>[25]</sup>、哈希函数密码学（Hash Function Cryptography）、共识算法（Consensus Algorithm）<sup>[29]</sup> 诸多技术建构而成一个分布式、不需要仰赖中心化机构加以维护的交易帐本，表2.1是比特币系统的相关参数简介。在接下来的章节中将逐一进行详尽的说明每个技术在各个环节中所扮演的角色。

表 2.1 比特币系统的相关参数简介

|              |                       |
|--------------|-----------------------|
| 第一个区块生成时间    | 2009 年 1 月 3 日        |
| 比特币预计总产量     | 21,000,000 BTC        |
| 比特币目前总产量     | 16,921,800 BTC        |
| 最新区块高度       | 513743 块              |
| 比特币总产值       | 5 兆人民币                |
| 比特币全节点的数量    | 11147 个               |
| 单日比特币交易金额    | 124,017.02430718 BTC  |
| 单日交易笔数       | 196,606 笔             |
| 比特币区块链大小     | 188.89 GB             |
| 平均区块大小       | 0.75 MB               |
| 平均生成单一区块所需时间 | 9.67 分钟               |
| 单日产出比特币数量    | 1,775 BTC             |
| 挖矿难度参数       | 3,290,605,988,755     |
| 全网挖矿算力       | 23,555,075.18 THash/s |

去中心化的加密货币系统给社会和传统中心化的金融体系以及政府带来了重大的冲击，Satoshi Nakamoto 建构了一个不需要经过中央银行发行货币的货币系统，在比特币的货币发行上全靠区块链既定的算法。除了货币发行，也将交易记录的帐本以明文的方式存储在去中心化的区块链中，以比特币为例，现今完整的比特币区块链帐本已经高达 188.89 GB，这样保存完整交易数据的计算机称之为全节点，在比特币去中心化的网络中，如图2.1所示，截至 2018 年 1 月 25 比特币网络中全节点数量为 11147 个<sup>[30]</sup>，全节点的数量决定了比特币帐本的可靠度，越多的全节点数量越能巩固比特币网络中过去的交易数据，使其难以被窜改。

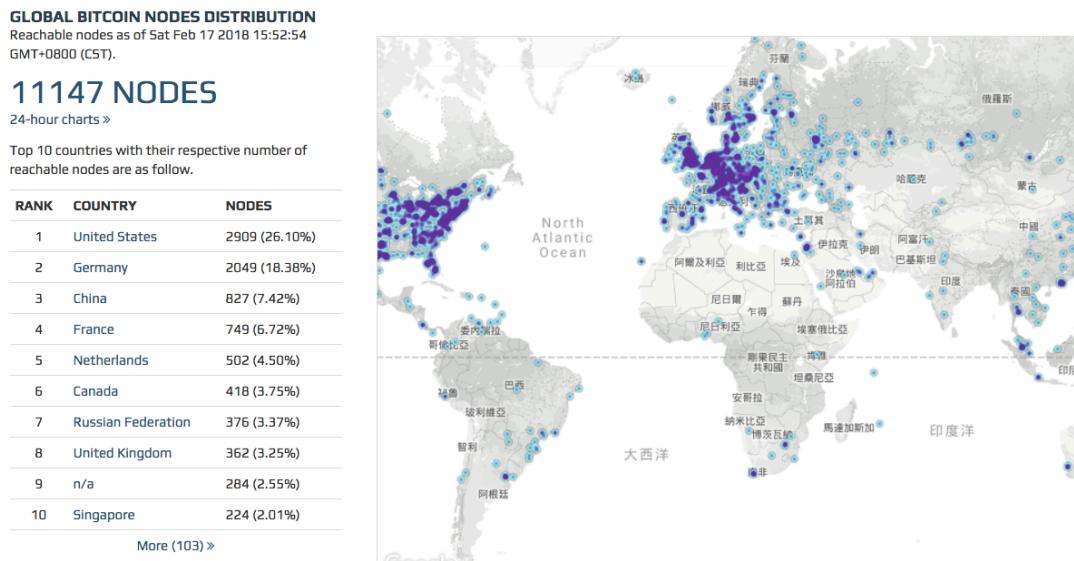


图 2.1 截至 2018 年 2 月 17 日止, 比特币全节点分布图<sup>[30]</sup>

## 2.1 比特币地址

比特币地址为比特币的载体，深入了解比特币的地址生成相关算法、比特币地址生成过程、多重签章，可以进一步应用在比特币的交易监督系统。

在点对点的现金系统中，首先必须先生成一个地址，在比特币的协议中有着既定的程序生成地址。运用到的技术包括乱数产生器（Random number generator）、非对称式加密算法 Secp256k1<sup>[31]</sup>、哈希算法 SHA-256<sup>[32]</sup> 以及 Base58 编码<sup>[33]</sup>。接下来将详述每一个函数的运作过程以及意义，最后说明比特币交易地址生成的每一个步骤。

### (一) 乱数产生器 (Random number generator)

乱数在密码学中是个相当重要的一环，在比特币系统中更是重要，毕竟生成的乱数会变成比特币的私钥，私钥是加密货币中是签署资产转移的唯一方式，在比特币地址中的乱数产生器会产出一个 256 bits 长度的乱数成为私钥，256 bits 的长度可以表现的组态空间为  $2^{256}$ ，换算成十进位表示为  $1.1579209 \times 10^{77}$ ，要在这组态空间中，以乱数产生同样的一把私钥是一件困难的事，但也有国际的实验室<sup>[34]</sup> 团队正在努力的穷举比特币  $2^{256}$  的组态空间，如图2.2所示，根据 LBC 公布的数据显示，目前已经完成了  $2.330109 \times 10^{16}$  个地址探索。虽然  $10^{16}$  的级别与  $10^{77}$  的级别相距甚远，但 LBC 已探索的组态空间中击中了 15 个比特币地址，该团队也成功将这 15 个地址下的 1.180899 个比特币转走。

如何建构一个乱数，在过往的乱数产生器往往会加入时间作为参数，但对于一个攻击者而言，只需要去猜测在这段时间内目标者所有生成的可能性有极高的机率可猜

2018-01-31 12:32:48

## 24h Pool Performance: 1588.74 Mkeys/s



图 2.2 LBC 穷举比特币私钥算力状态图<sup>[34]</sup>

出乱数。而乱数在密码学中常会是一把私钥的生成，在 HTTPS 协议中，服务器端与客户端建立一个加密连接的过程中，也需要一个乱数去建立高安全性的加密信道-传输层安全性协定（Transport Layer Security，TLS）<sup>[35]</sup>，在 SSH 协议<sup>[36]</sup>中也采用了乱数。

于 2013 年 8 月比特币开发者 Mike Hearn 于论文"All private keys generated on Android phones/tablets are weak and some signatures have been observed to have colliding R values"<sup>[37]</sup> 中提及在过去的历史事件中，发现 Android 手持移动装置版以及平板版的乱数产生器中存在着不随机，在不同的手持移动装置中有机会生成同样的私钥。Bitcoin.org 也发布了警告<sup>[38]</sup> 简要说明该乱数不随机事件的原因，以及表明受到该事件影响的比特币钱包客户端包含有 Bitcoin Wallet、BitcoinSpinner、Mycelium Bitcoin Wallet、blockchain.info。这样的错误源于 Android 本身支持的乱数产生器并不随机，随后 Android 解释了乱数的问题并加以修正。在这 Android 手持移动装置乱数不够乱的事件中，有自愿者自发性地公布自己的损失状态，总金额为 55.82152538 个比特币<sup>[39]</sup>，但因为比特币属于被动的性质，无人主动回报即不会加入统计中，所以总损失估计会超过 55.82152538 个比特币。

## （二）非对称式加密算法 Secp256k1

在密码学中有分对称式加密与非对称式加密，对称式加密又分为信息流加密与信息块加密，信息流加密著名的是由美国密码学家 Ron Rivest 教授发表，包括 RC2（1987 年）<sup>[40]</sup>、RC4（1987 年）<sup>[41]</sup>、RC5（1994 年）<sup>[42]</sup>、RC6（1998 年）<sup>[43]</sup>；信息块加密著名的有数据加密标准（Data Encryption Standard，DES，1975 年）<sup>[44]</sup>、三重数据加密算法（Triple Data Encryption Algorithm，Triple DES，1998 年）<sup>[45]</sup>、高级加密标准（Advanced Encryption Standard，AES，1998 年）<sup>[46]</sup>；非对称式加密最广为人知的有 RSA（Rivest – Shamir – Adleman，1977 年）<sup>[47]</sup>、椭圆曲线密码学（Elliptic Curve Cryptogra-

phy, ECC, 1985 年)<sup>[48]</sup>。非对称式加密与对称式加密最大的不同，在于对称式加密在加密解密的过程中只需要一把钥匙，而非对称式加密会生成两把钥匙，分别为私钥与公钥，在算法的设计上一开始会以乱数产生一把私钥，再经由非对称式加密算法推导出公钥，推导出的公钥在非对称式密码学中并无直接的方法可以反推至私钥，如此一来确立私钥的安全性。非对称式密码的使用场景有两种，第一种是希望收到加密信息的用户 Alice，Alice 会生成私钥存储在自己本地端的电脑中，并将推导出的公钥公布在网络上，这时希望联系 Alice 的用户 Bob 在网络上取得公钥后，Bob 会以 Alice 的公钥进行加密，之后将密文寄送给 Alice，在传递信息的过程中，即使网络存在着监听，也无法将信息顺利解密，唯有 Alice 收到信息后使用 Alice 原本产生该公钥的私钥，才可以解出明文。第二种则应用在比特币的交易之数字签名以及验证比特币交易，比特币地址的创建过程中会透过 secp256k1 生成私钥公钥对，在创建比特币交易的过程中，使用该地址的私钥对该地址未花费的输出（Unspent Transaction Output, UTXO）<sup>[1]</sup> 进行数字签名，完成数字签名后会与公钥以及交易信息一起广播到比特币网络的交易缓存池当中，比特币交易缓存池存在于所有比特币全节点当中，主要存储所有未被收录到比特币区块链内的所有交易，也就是零确认交易，等待矿工将该笔交易收入至比特币区块链当中。比特币采用的 secp256k1 是属于椭圆曲线密码学中的一个版本，不同的椭圆曲线版本的差异在于不同的初始参数，包括椭圆曲线方程  $y^2 = x^3 + ax + b$ 、 $p=FFFFFFFFFFFFFFFFFFF...FFFFFEFFFFFC2F$  为巨大的素数、 $G$  点被称为生成点的常数点亦称为基点。至于为什么选择 ECC 而非 RSA 的主要原因，其一在于 ECC 在生成密钥对所需的时间更加快速，根据表2.2显示，Nicholas Jansma 于 2004 年针对 ECC 与 RSA 的密钥对生成时间与数字签名所需时间的论文<sup>[49]</sup> 显示，当 ECC 产生 571 bits 的密钥长度，要达到相同安全等级的情形下，RSA 则需要生成 15360 bits，这也让两种非对称式的加密方式之间，生成的时间产生了高达 471 倍之差距。

表 2.2 ECC 与 RSA 相同安全等级的密钥对生成时间比较表<sup>[49]</sup>

| 密钥长度 (bits) |       | 时间 (秒) |        |
|-------------|-------|--------|--------|
| ECC         | RSA   | ECC    | RSA    |
| 163         | 1024  | 0.08   | 0.16   |
| 33          | 2240  | 0.18   | 7.47   |
| 283         | 3072  | 0.27   | 9.8    |
| 409         | 7680  | 0.64   | 133.9  |
| 571         | 15360 | 1.44   | 679.06 |

除了在密钥对生成时间 ECC 有着比 RSA 更高效的算法外，在安全性上 ECC 可以更短的密钥长度达到与 RSA 相同的安全强度，Léo Ducas 针对 ECC、RSA、BLISS

(Bimodal Lattice Signature Scheme)<sup>[50]</sup> 做出了深度的安全性探讨, 表2.3同样达到 80 bits 的安全性级数, RSA 1024 需要 1024 bits, ECDSA-160<sup>[51]</sup> 仅需要 160 bits, 该篇论文除了探讨 RSA 与 ECDSA (Elliptic Curve Digital Signature Algorithm) 之外, 更大的部分在阐述量子计算机对于既有的传统密码带来的抨击, 有机会快速穷举  $2^{256}$  的比特币私钥, 在未来量子计算机的蓬勃发展拥有 2000 qbits 运算能力, 量子计算机可以快速穷举破解所有的比特币私钥。因此发展针对量子计算机设计的数字签名算法成为密码学上崭新的议题, 而 BLISS 则为针对量子计算机所设计的抗量子计算的签章算法。

表 2.3 算法 BLISS、RSA 及 ECDSA 安全级数比较图表<sup>[50]</sup>

| 算法        | 安全性            | 签名大小    | 私钥大小    | 公钥大小    | 签名(微秒) | 签名/秒 | 验证(微秒) | 验证/千秒 |
|-----------|----------------|---------|---------|---------|--------|------|--------|-------|
| BLISS-0   | $\leq 60$ bits | 3.3 kb  | 1.5 kb  | 3.3 kb  | 0.241  | 4k   | 0.017  | 59    |
| BLISS-I   | 128 bits       | 5.6 kb  | 2 kb    | 7 kb    | 0.124  | 8k   | 0.03   | 33    |
| BLISS-II  | 128 bits       | 5 kb    | 2 kb    | 7 kb    | 0.48   | 2k   | 0.03   | 33    |
| BLISS-III | 160 bits       | 6 kb    | 3 kb    | 7 kb    | 0.203  | 5k   | 0.031  | 32    |
| BLISS-IV  | 192 bits       | 6.5 kb  | 3 kb    | 7 kb    | 0.375  | 2.5k | 0.032  | 31    |
| RSA-1024  | 72-80 bits     | 1 kb    | 1 kb    | 1 kb    | 0.167  | 6k   | 0.004  | 91    |
| RSA-2048  | 103-112 bits   | 2 kb    | 2 kb    | 2 kb    | 1.18   | 0.8k | 0.038  | 27    |
| RSA-4096  | >128 bits      | 4 kb    | 4 kb    | 4 kb    | 8.66   | 0.1k | 0.138  | 7.5   |
| ECDSA-160 | 80 bits        | 0.32 kb | 0.16 kb | 0.16 kb | 0.058  | 17k  | 0.205  | 5     |
| ECDSA-256 | 128 bits       | 0.5 kb  | 0.25 kb | 0.25 kb | 0.106  | 9.5k | 0.384  | 2.5   |
| ECDSA-384 | 192 bits       | 0.75 kb | 0.37 kb | 0.37 kb | 0.195  | 5k   | 0.853  | 1     |

### (三) 哈希算法 SHA-256

SHA-256 是 SHA (Secure Hash Algorithm, FIPS 182-2)<sup>[32]</sup> 哈希算法的家族之一。SHA 家族当中有着四大分支, 分别为 SHA-0、SHA-1、SHA-2 和 SHA-3。各种哈希算法的差异在于运算初始变量、算法所采用的运算子、接受的信息长度以及循环数的不同, 如表2.4所示。上述的参数差异皆由联邦信息处理标准 (Federal Information Processing Standards, FIPS) 中定义。表2.4中 MD5 不为 SHA 家族成员之一, 但 MD5 为最早被广泛使用的哈希算法, 因此此表将其作为借鉴的标准。SHA-0 为 SHA 家族中被最早提出的架构, 输出的长度为 160 bits, 而 SHA-1 提出后并无太大的变动。哈希算法的主要功能在于将信息或是文件建立专属的对应指纹, 也就是哈希值, 依照不同的算法设计, 该指纹的输出长度也略有不同, 表2.4显示了 MD5 与 SHA 家族的输出哈希值长度, 而长度也意味着该指纹的组态空间所映射的大小。倘若有着不同的输入, 但映射到了相同的指纹, 则将此现象称之为碰撞。通常在信息安全领域中, 只要发现该哈希算法存在着碰撞, 就会被弃用。有些专家甚至会提早数年建议用户尽早更换该哈希算法, 并更换上新制定的哈希算法做应用。哈希算法的功能包括借由生成哈希值进行文件校验、

工作量证明算法设计以及区块链中的哈希指针<sup>[52]</sup>。

表 2.4 MD5、SHA-0、SHA-1、SHA-2 和 SHA-3 比较表

| 算法        | 分支       | 输出哈希值长度(bits) | 最大输入信息长度(bits) | 循环(次数) | 使用到的运算子  | 碰撞攻击(bits)  |  |
|-----------|----------|---------------|----------------|--------|--|-------------|--|
| MD5<br>参考 | -        | 128           | $\infty$       | 64     | And, Xor, Rot, Or,<br>Add (mod $2^{32}$ )      | < 64<br>已碰撞 |  |
| SHA-0     | -        | 160           | $2^{64} - 1$   | 80     |  | < 80<br>已碰撞 |  |
| SHA-1     | -        |               |                |        |  | < 80<br>已碰撞 |  |
| SHA-2     | SHA-224  | 224           | $2^{64} - 1$   | 64     | And, Xor, Rot, Or, Shr,<br>Add (mod $2^{32}$ ) | 112         |  |
|           | SHA-256  | 256           |                |        |  | 128         |  |
|           | SHA-384  | 384           | $2^{128} - 1$  | 80     | And, Xor, Rot, Or, Shr,<br>Add (mod $2^{64}$ ) | 192         |  |
|           | SHA-512  | 512           |                |        |  | 256         |  |
| SHA-3     | SHA3-224 | 224           | $\infty$       | 24     | And, Xor, Rot, Not                             | 112         |  |
|           | SHA3-256 | 256           |                |        |  | 128         |  |
|           | SHA3-384 | 384           |                |        |  | 192         |  |
|           | SHA3-512 | 512           |                |        |  | 256         |  |

- 生成哈希值进行文件校验：哈希值在过往的应用中，往往作为文件完整性的校验，软件供应方会在网站中提供各式不同算法的哈希值，让使用者在下载完软件之后输入使用者方以哈希算法所生成的哈希值进行比对。但倘若该哈希算法存在碰撞的发生，这会使得不同的文件存在着同样的哈希值。这也意味着软件供应方所提供的软件可能存在着掺入恶意代码后，还能生成同样的哈希指纹，失去了文件完整性的校验功能，这便成为信息安全中的重大漏洞，由表2.4可以看出软件提供方若使用 MD5、SHA-0 以及 SHA-1 会造成发生碰撞导致指纹失去校验意义成为安全漏洞，因此 MD5、SHA-0 以及 SHA-1 已被弃用。
- 工作量证明算法设计：起初工作量证明算法的概念为设计一个相当困难耗时的解，但验算的过程中却相当简单快速。如对一个大数做因式分解是一个相当困难的事，但要验算其结果仅需要将所有的解相乘确认是否为该大数即可证明。同样的理念在比特币系统中是以 hash-puzzle 的方式实现，hash-puzzle 是利用哈希函数有着不可预期的特性，不可预期指的是假设输入连续性的数值 1 到  $n$  进入到哈希函数生成哈希值，而生成的哈希值无法观察出关联性，可说是完全不相关的数值，且无法预期下个输入的输出哈希值。比特币系统中的困难度变量，在 hash-puzzle 中定义了何谓真正的答案。在 SHA-256 当中输出的值为 256 bits 的哈希值，困难度变量则规定了在这 256 bits 当中，自最左边起必须为零的位数的门槛，而

在困难度变量要求必须为零的位数变多，则意味着要在连续性的数值 1 到  $n$  的 hash-puzzle 中，符合门槛的解越少。在 hash-puzzle 中，求得一个符合困难度变量的哈希值是困难的，但要验算求得的解是否符合困难度的门槛相当快速，这符合起初工作量证明算法的概念。

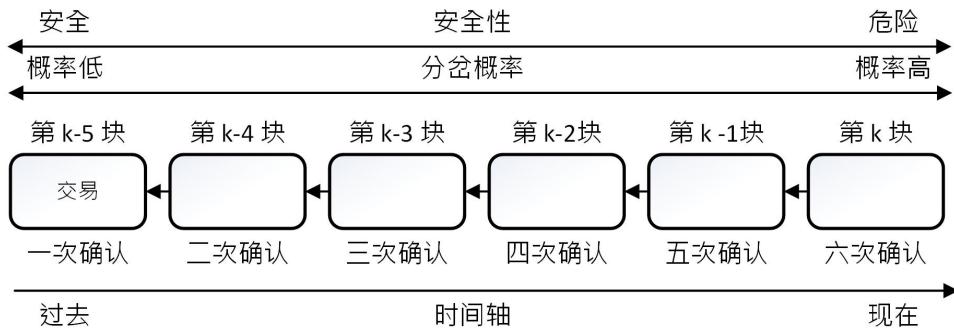


图 2.3 区块叠加示意图

3. 区块链中的哈希指针：比特币区块链的区块头当中，当前区块存储着前区块头的双重哈希的哈希值。当前区块则会如数据结构链结串列一样，直到链结到第一个比特币区块的创世区块，区块链中的哈希指针将区块链结在一起，也就形成了比特币区块链的基础。在比特币挖矿的过程中，矿工参与着最新区块的 hash-puzzle 解题，寻找一个符合困难度变量的输入。但 hash-puzzle 的工作证明当中解可能不只一个，但只要符合困难度变量的解都可以创建一个新的区块，届时就会在当前的区块上同时产生分岔，分岔的区块在比特币系统中皆为有效，但经过多个区块的叠加之后，便可以抉择出最长的链，该最长链则成为主链，而其他的分岔链，则成为孤儿块被丢弃，当中曾记载的交易信息变为无效，造成该分岔链的交易信息回溯。比特币区块链的分岔好发于最新的区块，如图2.3所示，而越旧的区块则越不易发生，所以在比特币交易中默认该笔交易要在六个区块确认后，该公司才会承认该笔交易有效。

#### (四) Base58 编码

表2.5为 Base58 编码表，Base58 编码的首次出现自 Satoshi Nakamoto 所提出的论文<sup>[1]</sup>当中，Base58 编码是源自于 Base64 编码表，如表2.6中所示。Base64 编码中包括大写英文 26 个字母、小写英文 26 个字母、阿拉伯数字以及字符"/" 和 "+"。在比特币地址生成过程中，Base58 的功能是将比特币地址的公钥哈希值重新编码，比特币地址的公钥哈希值是二进制的数据型态，即使是将二进制码转换成十六进制码输出也是对人类辨识上有一定的不便性。倘若采用 Base64 对比特币公钥哈希值进行编码有效缩短二进制码的长度，但 Base64 的编码存在着不适于作为地址的特殊字符 "+" 和 "-"。在 Base58

编码中移除了特殊字符 "+" 和 "-" 之外，也移除了人类较不易判读的相关字符，数字 "0" 与大写英文字母 "O"，因为该两字符在不同字体的体现相当相似，大写英文字母 "I" 以及小写英文字母 "l"，在人类判读上也有些为相似。因此于 Base64 移除上述 6 个字符后，便形成了 Base58 编码表。值得一提的是，Base58 编码与 Base64 编码表中的字符排序有些许异动，Base58 编码表将数字的部分移置到最前面，这也是比特币地址在以 Base58 编码后所呈现的第一个字符为 "1" 的主要原因。

表 2.5 Base58 编码表

| 数值 | 字符 | 数值 | 字符 | 数值 | 字符 | 数值 | 字符 |
|----|----|----|----|----|----|----|----|
| 0  | 1  | 16 | H  | 32 | Z  | 48 | q  |
| 1  | 2  | 17 | J  | 33 | a  | 49 | r  |
| 2  | 3  | 18 | K  | 34 | b  | 50 | s  |
| 3  | 4  | 19 | L  | 35 | c  | 51 | t  |
| 4  | 5  | 20 | M  | 36 | d  | 52 | u  |
| 5  | 6  | 21 | N  | 37 | e  | 53 | v  |
| 6  | 7  | 22 | P  | 38 | f  | 54 | w  |
| 7  | 8  | 23 | Q  | 39 | g  | 55 | x  |
| 8  | 9  | 24 | R  | 40 | h  | 56 | y  |
| 9  | A  | 25 | S  | 41 | i  | 57 | z  |
| 10 | B  | 26 | T  | 42 | j  |    |    |
| 11 | C  | 27 | U  | 43 | k  |    |    |
| 12 | D  | 28 | V  | 44 | m  |    |    |
| 13 | E  | 29 | W  | 45 | n  |    |    |
| 14 | F  | 30 | X  | 46 | o  |    |    |
| 15 | G  | 31 | Y  | 47 | p  |    |    |

于上述章节中已经详细说明比特币地址生成过程中使用到的所有相关算法技术，以及算法在现今的比特币系统中存在的问题。于本节中将阐述如何将比特币地址生成的相关算法实际带入比特币地址生成的过程中，地址的生成总共分为八个步骤，分别为生成私钥、生成公钥、生成公钥 SHA-256、生成公钥 SHA-256 的 RIPEMD-160、取得版本号、生成校验码、版本号与公钥 SHA-256 的 RIPEMD-160 和校验码合并，最后一步则是将合并的结果以 Base58 编码创建出一个比特币地址。

1. 生成私钥：使用乱数产生器产生一个长度为 256 bits 的乱数，而此乱数即成为该比特币地址的私钥。在比特币的系统当中，私钥可以透过椭圆曲线签章算法 secp256k1 签署一笔交易，广播到比特币网络当中。由于私钥为该地址资金转移的关键，黑客攻击的对象皆会聚焦在比特币私钥，因此私钥的保存成为比特币系统中最为热门的课题之一。
2. 生成公钥：在以乱数产生器生成私钥之后，接下来将运用到非对称式密码学中的

表 2.6 Base64 编码表

| 数值 | 字符 | 数值 | 字符 | 数值 | 字符 | 数值 | 字符 |
|----|----|----|----|----|----|----|----|
| 0  | A  | 16 | Q  | 32 | g  | 48 | w  |
| 1  | B  | 17 | R  | 33 | h  | 49 | x  |
| 2  | C  | 18 | S  | 34 | i  | 50 | y  |
| 3  | D  | 19 | T  | 35 | j  | 51 | z  |
| 4  | E  | 20 | U  | 36 | k  | 52 | 0  |
| 5  | F  | 21 | V  | 37 | l  | 53 | 1  |
| 6  | G  | 22 | W  | 38 | m  | 54 | 2  |
| 7  | H  | 23 | X  | 39 | n  | 55 | 3  |
| 8  | I  | 24 | Y  | 40 | o  | 56 | 4  |
| 9  | J  | 25 | Z  | 41 | p  | 57 | 5  |
| 10 | K  | 26 | a  | 42 | q  | 58 | 6  |
| 11 | L  | 27 | b  | 43 | r  | 59 | 7  |
| 12 | M  | 28 | c  | 44 | s  | 60 | 8  |
| 13 | N  | 29 | d  | 45 | t  | 61 | 9  |
| 14 | O  | 30 | e  | 46 | u  | 62 | +  |
| 15 | P  | 31 | f  | 47 | v  | 63 | /  |

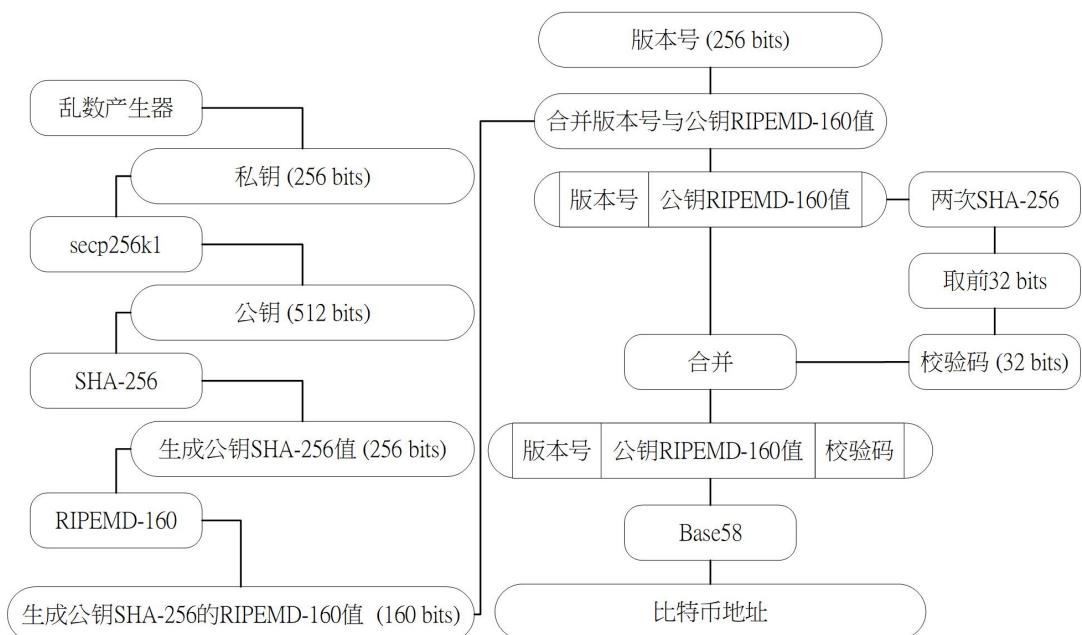


图 2.4 比特币地址生成流程图

私钥公钥转换算法，于比特币系统中采用的是 secp256k1，secp256k1 是椭圆曲线密码学中的其中一个版本，不同的版本差异在于采用不同的变量。于比特币地址生成的过程中，secp256k1 负责将上述步骤所生成的私钥透过椭圆曲线算法计算出公钥，生成的公钥长度为 512 bits，这比第一个步骤所生成的 256 bits 私钥多出一倍的長度。在比特币交易中，私钥可以签署一笔交易，在签署完之后便会将公钥、签名以及交易信息广播至比特币网络中，等待被收入到比特币区块链当中。此时该笔交易在比特币交易缓存池中等待 secp256k1 算法对其公钥、签名以及交易信息进行验算，若计算出的值为真，则该笔交易被视为有效且继续待在交易缓存池中，等待被收入区块链内。

3. 生成公钥 SHA-256：SHA-256 为 SHA 家族之一，也是哈希算法的一种，因此符合哈希算法的特征包括雪崩效应、不可预测、不可逆（单向性）以及校验文件是否完整的诸多特性。将前一个步骤所生成的長度为 512 bits 的公钥作为 SHA-256 的输入，产出长度仅为公钥一半的公钥 SHA-256 哈希值。这个步骤将使即使知道公钥 SHA-256 的攻击者，更难以用穷举攻击推导出该地址的公钥。
4. 生成公钥 SHA-256 的 RIPEMD-160：RIPEMD-160 也是哈希算法的一种，和其他哈希算法的特征相符，与 SHA-256 不同的部分在于 RIPEMD-160 所生成的哈希值長度为 160 bits。在前一步骤中进行 SHA-256 计算的公钥已经有了第一层的保护，而在第四个步骤中，再次透过 RIPEMD-160 取得哈希值。这将使得攻击者既使取得了比特币地址，也必须先针对 RIPEMD-160 进行破译，再进一步对 SHA-256 破译，才有机会取得该地址的公钥。因为这样的设计，让只收取却不花费的比特币地址，在黑客攻击上造成相当大的难度。
5. 取得版本号：在比特币系统初始的设计中，已经定义了些不同功能的比特币地址，这些特殊功能的比特币地址有着特殊的版本号，最为常见的为以"1" 为头的比特币地址，该地址为比特币系统中最早被使用且最为普遍的地址版本，该种地址为一把私钥进行比特币地址推倒，所以仅需要一把钥匙就可以移动该地址下的比特币资产；第二种是以"3" 为地址开头的比特币地址，该地址采用多重签章技术，该技术为后续经过多项 BIP 才完成落实于比特币系统。在第五个步骤中会加入版本号加以区分不同的地址。
6. 生成校验码：校验码是比特币地址生成过程中重要的一环。校验码可降低比特币的支付过程中因手误而转入到不存在（不符合比特币地址生成规则）地址的可能性。在第六个步骤中对公钥 SHA-256 的 RIPEMD-160 再做两次 SHA-256，取该哈希值前 32 bits 的值作为校验码。
7. 版本号与公钥 SHA-256 的 RIPEMD-160 和校验码合并：即将第五步骤产生的版

本号、第四个步骤产生的公钥 RIPEMD-160 及第六个步骤产生之校验码三者进行合并。

8. 合并的结果以 Base58 编码：Base58 修改自 Base64，其与 Base64 最大不同之处在于移除了"0"、"O"、"I"、"l"、"+"、"/" 的字符，可以降低人工在判读地址的错误率。在这个步骤中使用 Base58 将第七步骤合并的组合结果进行编码。

比特币区块链技术，虽然已经利用工作量证明的方式解决了双重支付（Double-spending）问题<sup>[53][54]</sup>，但工作量证明的算法所设置之题目困难度会直接影响到每一个比特币区块的产出时间，这个比特币区块的产出时间也考虑到比特币全节点于全世界各地的网络同步状况，倘若今天的区块生成时间过短会造成全世界的比特币节点之区块数据不一致，这样的数据不一致将导致比特币区块链出现分岔，再更严重一点甚至会造成比特币网络的瓦解。

现今的比特币区块产出速度为十分钟一块，附上足够的手续费也须等待将近十分钟的交易时间，若是在手续费较低的情况下，该笔比特币交易甚至可能会在交易缓存池中滞留一周的时间才被矿工运算。即使在手续费足够的情况下，十分钟的确认时间会对实体商家的小额交易处理非常不友善，为了在既有的比特币区块链框架底下能够提升交易速度，因此产生了多重签章（Multi-Signature）技术。多重签章于在交易开始创建的同时管控双重支付交易的发生，采用了 2-of-2 多重签章创建一个特殊的比特币地址，这个地址由两个代表人所持有，分别为使用者与 Green Address<sup>[55]</sup> 机构节点，每笔交易的建立必须要双方同时签署才被允许广播至比特币网络中。若是遇到交易塞车，且节点缓存池空间不足的情况时，比特币节点会优先遗弃手续费最低的交易，视同该笔交易不曾存在过。若真的遇到交易被遗弃的情况，Green Address 机构节点也会在之后透过内部的数据库记录再次广播此笔交易，并确保此笔交易可以被收入至区块内。Green Address 机构节点因此成为了交易创建的把关者，过滤所有双重支付攻击的发生，也避免交易因为比特币网络塞车而被矿工遗弃的情形。在这样的机制下，只要是使用 Green Address 钱包交易就可避免双重支付攻击的发生，对商家或是收款人而言，可以在即时交易中同时得到双重支付攻击的保护，也提高了交易在未进入区块链前的可确定性，达到即时交易的可能。

### （一）Green Address 钱包生成过程

此节将详细阐述 Green Address 钱包生成过程的重要步骤，如图2.5所示。

1. 用户安装 Green Address 比特币钱包。
2. 用户于本地端透过乱数产生器生成一个比特币私钥。
3. 向 Green Address 机构节点请求创建 2-of-2 多重签章比特币地址。

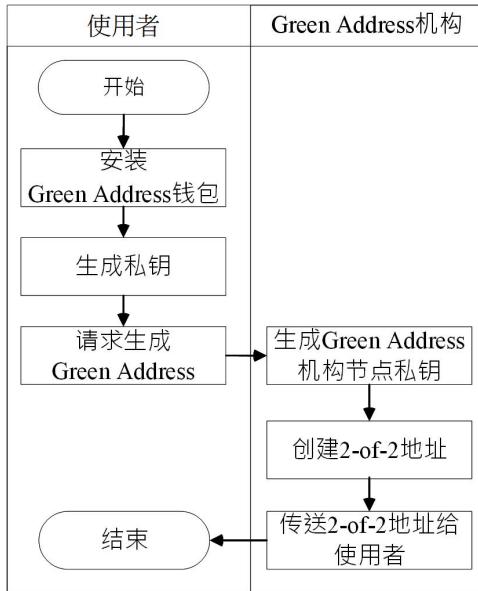


图 2.5 Green Address 钱包生成流程图

4. Green Address 机构节点使用乱数产生器生成私钥。
5. 创建 2-of-2 多重签章比特币地址如"2NGXNJrxq2qH6SuGEuJ57GjvMAmLUwPbZfQ"，该地址的为多重签章地址，因此在地址的第一个字为"2"，倘若为一般非多重签章地址，则地址的第一个字为"1"。
6. 将生成的 2-of-2 多重签章比特币地址传回用户的 Green Address 比特币钱包。

## (二) Green Address 交易发起流程

说明完 Green Address 地址是如何创建之后，本节将详细说明如何运用多重签章地址发起交易至比特币网络中，如图2.6所示。

1. 用户使用原本创建 Green Address 的私钥，并完成签署交易。
2. 因为是多重签章地址，所以该交易需发送至 Green Address 机构节点。
3. Green Address 机构节点收到交易信息后检查该交易的发起地址是否存在双重支付，倘若有双重支付则遗弃该笔交易；若无双重支付则往下一个步骤。
4. Green Address 机构以 Green Address 的私钥签署该笔交易。
5. 将该笔交易封包广播至比特币网络。

## 2.2 区块链

自 2009 年以来，加密货币比特币的诞生引发了新的货币革命浪潮，基于密码学，点对点网络，共识算法和区块链技术，它们被结合成比特币加密货币。到目前为止，它在九年内发生大量的袭击和欺诈事件后仍然在积极努力。比特币一直是互联网上最具

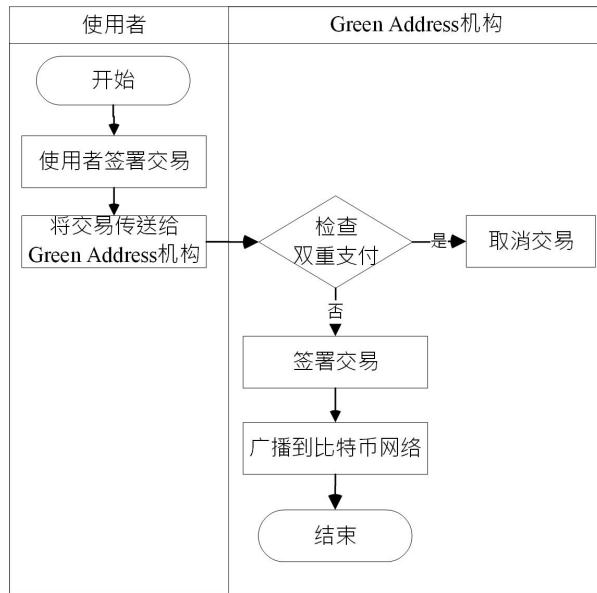


图 2.6 Green Address 交易发起流程图

代表性的加密货币，同时是区块链技术极为重要的应用之一。比特币区块链可视为一种专门存储交易信息的数据库，该数据库的结构严谨。图2.7 为比特币区块链结构图，区块链之所以称为链是因为由许多区块构成，区块头存在于区块中记录区块中的重要信息共六项，分别为区块版本、前区块的哈希值、Merkle Root、难度、时间戳以及Nonce，以下将逐一说明：

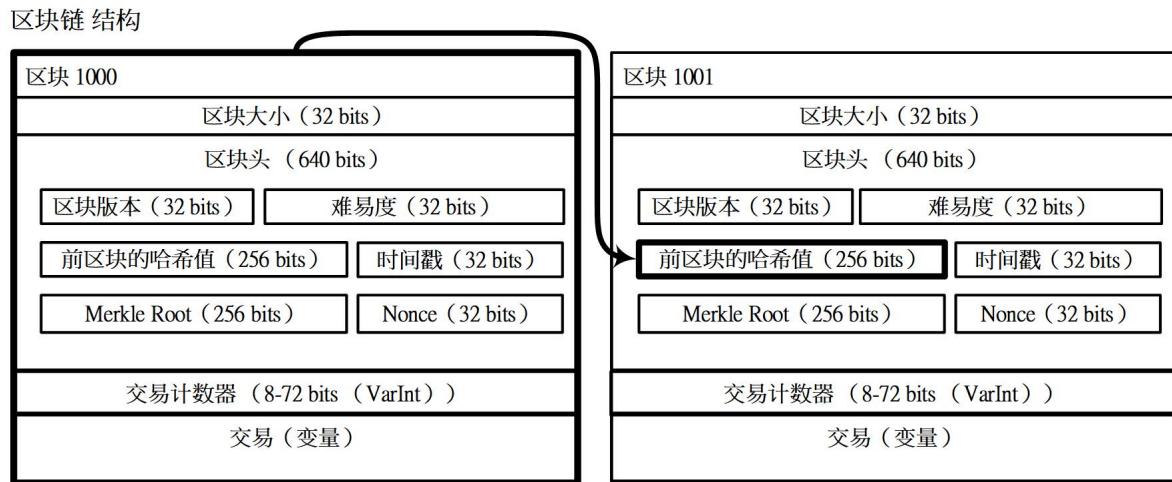


图 2.7 比特币区块链结构图

1. 区块版本 (32 bits)：该字段存储比特币区块链中的区块版本。
2. 前区块的哈希值 (256 bits)：记录前一个区块的哈希值。根据当前区块的前一个区块哈希值进而形成哈希指针，所有块可以因为哈希指针连接在一起形成比特币链。

区块链，不仅可以在区块与区块间创建虚拟链结，还可以使得区块更难以被篡改。而通过新区块不断叠加在旧区块的过程，旧区块的哈希值将继续传递到最新的区块上。若区块上面堆叠更多的区块，促使哈希值的间接引用越多次，因此较早创建的区块更难以修改。

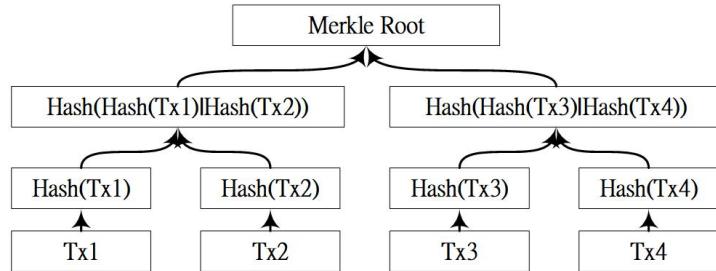


图 2.8 Merkle Tree 示意图

3. Merkle Root (256 bits): Merkle Root 的生成方法是将当前区块的所有交易为  $n$  个进行排序后，Merkle Root 为 Merkle Tree 的树根，交易为树叶  $n$  个，将每个树叶进行两次 SHA-256 哈希算法取得哈希值得到  $n$  个哈希值，再将哈希值两两配对合并进行两次 SHA-256，得到  $n * 2^{-1}$  个哈希值后，在  $k$  轮后会使得  $n * 2^{-k} = 1$  时，合并到只剩下一个哈希值，最后一个哈希值则为 Merkle Root，如图所示2.8，图中的 Hash() 函数为双重 SHA-256，在区块链中的 Merkle Root 可用于快速检查当前区块中所有存储交易的正确性。
4. 难易度 (32 bits)：难易度参数主要调控比特币挖矿过程中采用工作量证明算法的变量，值得一提的是比特币的难易度参数为动态调整。在过去加密货币的设计中，有着因为没有动态修改区块难度，而导致区块链生成速度太快，甚至导致区块链系统崩溃。
5. 时间戳 (32 bits)：以年、月、日、小时、分钟和秒的格式记录区块生成时间。
6. Nonce (32 bits)：Nonce 记录着矿工在进行挖矿时，必须要不断的尝试 Nonce 参数，直到符合难易度参数，才可以创建一个全新的比特币区块。该值为 32 bits，意为着矿工尝试的组态空间为  $2^{32}$  个可能性。

## 第三章 系统需求分析

透过特性要因分析可以将比特币的交易监督系统大致分为四个主题，如图3.1所示，分别为信息安全技术、加密货币钱包、近场通信技术以及数据库。作为一个金流系统，四项主轴之中信息安全是不可或缺的环节，着重于商家认证机制、用户权限控管、身份识别管理、用户访问控制四个方向；本系统致力于奠定匿名对实名的加密货币系统，必须对区块链技术、公钥私钥生成算法、点对点交易技术、钱包地址产出以及货币发行技术五个方向进行探讨；在交易场景中，本系统采用近场通信技术，因此需要对商品RFID标签建置、读取商品RFID标签以及Android Beam传输商品交易进行基础的API调研；为了使加密货币实名制的实现，数据库必须存储与政府和商家相关的信息。此时数据库加密、个人信息去识别化安全以及数据库连接便相当重要。

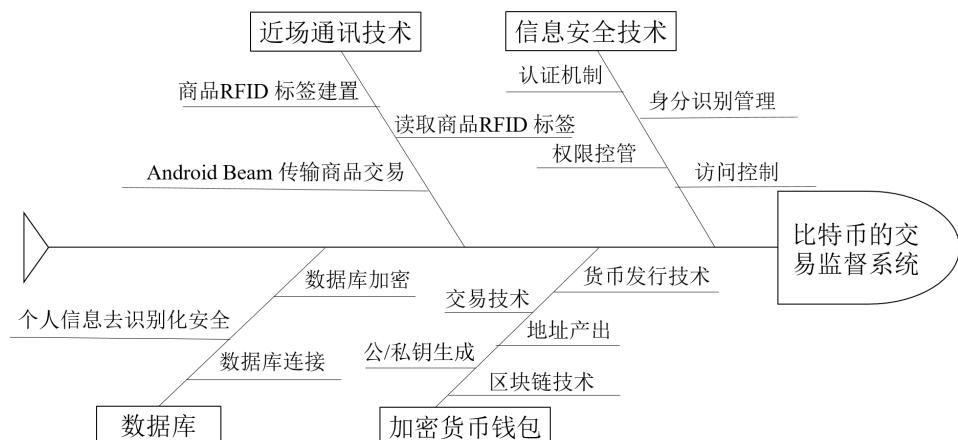


图 3.1 比特币的交易监督系统特性要因分析图

为了使本文所提出的系统设计之模块更加明确，对系统进行详细的需求分析可以使应确立的方向更加分明，在本章将分为三节进行分析，分别为交易模型分析、功能性需求分析以及非功能性需求分析。

### 3.1 交易模型分析

在设计比特币的交易监督系统之前必须针对现今社会中人与人之间的交易方式进行分析，在支付的方式依照使用人数比例大致可区分为现金交易以及电子支付两种类别，图3.2为各种交易模型示意图。

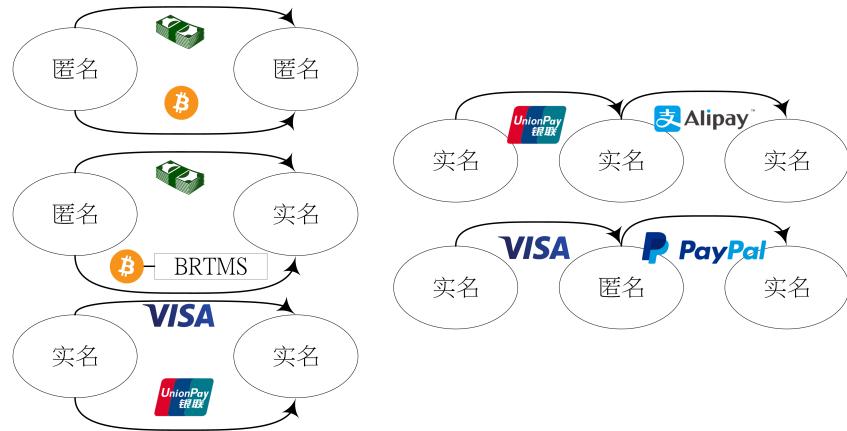


图 3.2 各种交易模型示意图

### 3.1.1 现金交易类型

在现金交易模型中大致分为匿名顾客支付匿名商家以及匿名顾客支付实名商家两种，在过去的社会中，贝壳货币逐渐的被黄金取代，但黄金过于沉重在交易上相当不便。纸钞渐渐取代黄金，法定货币的概念因此形成。法定货币包括钞票以及钱币，在过往的法定货币存在着黄金价值支撑，使得法定货币具有价值，但经过历史的变迁，已无大量的黄金作为法定货币的价值支撑，使法定货币渐渐地走向信用本位，纸钞以及铜币的使用皆不带有持有者的真实姓名，因此在现金交易模型分析中将顾客支付皆归类为匿名支付。

#### (一) 匿名顾客支付匿名商家交易模型

现金法定货币的本身不存在用户信息以及交易信息。现金是一种匿名的支付方式，早期有许多的商家并无向政府进行注册，当顾客完成挑选商品进行结帐的同时，顾客只是将现金货币交付给商家，商家将商品卖给顾客，但在这过程中并无开立交易凭据。在商家无开立交易凭据的情况下，顾客完成此次的交易后，倘若该商品存在着瑕疵，进而引起商家与顾客之间的消费纠纷，使其在追溯的过程中存在没有凭据的困扰。对于匿名顾客支付匿名商家可以保有顾客的信息隐私，但无法保障顾客应有的顾客权益。政府因没有交易凭据而无法确认消费行为存在，使政府在课征税收的过程变得困难，且商家所贩售的商品并无得到明确的信息纪录，对于商家库存管理只能依靠过去的经验。对政府而言，商家贩售的商品并无完整的商品信息，无法对商家商品进行安全性检验，使得国民购买的商品存在的许多安全疑虑。

## (二) 匿名顾客支付实名商家交易模型

现金最为常见的为匿名顾客支付给实名商家的交易模式，在匿名顾客以现金法定货币进行交易时，商家只要开立交易凭据，即被归类为匿名顾客支付实名商家交易模型。商家在开立交易凭据的同时，凭据上所记载的信息包括职工信息、商家信息以及产品信息；职工信息记录该笔交易的担当人员，商家信息记录商家与顾客创建交易行为的时间与地点，产品信息详细记录顾客于该商家购买的商品；交易凭据因此成为记录交易事实的重要证明。对于顾客，因现金法定货币的匿名性使顾客依旧保有顾客信息隐私，但却因为商家开立交易凭据得以保障顾客权益。商家因为开立交易凭据使商家为实名，因为实名使得商家可以经营品牌形象。除此之外，因为凭据记录了产品信息让商家在库存管理以及收支计算变得更加容易。对政府而言，可以查看商家的所得以及商家贩售的商品，使政府可以更有效率的课征税收以及管理产品安全。

### 3.1.2 电子支付类型

电子商务的日趋盛行，电子支付交易模式也日渐普及，电子支付与加密货币的区块链技术不同的是采用数据库存储着所有的交易信息，而用户要能够使用电子支付也必须接受电子支付运营商实名制的条件。在电子支付的数据库中存在着黑客攻击、信息不一致以及中心化运营的风险，但电子支付有效解决了难以现金支付远距离之困境，也使得现今的网络购物变得更加发达。在电子支付交易模型中分为三种交易分别为实名顾客支付实名商家、实名顾客透过实名第三方再实名商家以及实名顾客透过匿名第三方再实名商家，以下将逐一说明。

#### (一) 实名顾客支付实名商家交易模型

典型的实名顾客支付实名商家交易模式以 VISA 国际电子支付公司为例，VISA 国际电子支付公司（以下简称为 VISA 公司）为中心化的运营，用户在使用 VISA 电子支付（以下简称为 VISA 服务）前，必须做详尽的用户真实身份认证，这使得 VISA 公司可以得到用户的所有信息，因此 VISA 公司拥有了允许或冻结相关用户账户的使用权限，并且为了保护用户隐私，所有的交易信息记录皆被存储在 VISA 公司的数据库当中。

VISA 公司透过不断的优化电子支付技术已经可以接受每秒两千笔的交易量，为了维持稳定的运营服务，VISA 公司对每笔交易收取固定比例的手续费。在使用 VISA 服务的过程中，顾客因为必须进行实名验证，使得顾客必须透露顾客个人信息，甚至可以针对这些个人信息进行商业上的交易。商家必须承担 VISA 服务所需的交易手续费，成本因此提高，使得商品售价必须做出相对应的调整。对政府而言，因为交易信息皆

以中心化的方式存储，可以方便的查看，也可以快速的查阅商家收入课征应缴的税收。

## (二) 实名顾客透过实名第三方支付实名商家交易模型

电子支付方式日趋普及也使用户可选择的电子支付渠道更加多元，较为常见的包括 VISA 电子支付以及银联电子支付，而每一家银行为了让用户能在电子商务的支付上更为便利，会同时推广电子支付，使得银行卡支持电子支付，但更多的银行卡林立使得用户在卡片管理上更加繁琐，第三方的支付集成平台开始出现成为解决方案。顾客可以先在实名第三方支付后台添加多张卡片，再以第三方支付统一进行付款。实名顾客透过实名第三方支付给实名商家时，顾客可以得到更方便的银行卡管理，但却在第三方支付上透露了个人交易信息，面对仅支持第三方支付的商家却同时接受多家不同银行卡的支付方式，政府仅需要调阅第三方支付公司就可查阅该公司所有用户的交易信息，且可更完整的采集到商家的收入信息。

## (三) 实名顾客透过匿名第三方支付实名商家交易模型

为了让用户可以保有更多的隐私，Paypal 公司提出了实名顾客透过匿名第三方支付实名商家的模式，顾客欲支付一笔款项给商家时，首先使用 VISA 服务支付一笔资金到 Paypal 公司，Paypal 公司再以 Paypal 的名义支付该笔款项给商家。换言之，商家不会取得相关的顾客个人信息，也无法对顾客交易信息进行数据挖掘。值得一提的是，原本透过 VISA 公司支付款项给商家，顾客就必须先负担一笔的交易手续费给 VISA 公司，而再透过 Paypal 公司支付款项给商家时必须多经过 Paypal 公司，顾客必须再支付第二笔交易手续费给 Paypal 公司，这使得交易手续费更加的昂贵，但也因为多一个程序的资金转移，使得顾客的个人信息可以得到隐匿。实名顾客透过匿名第三方支付实名商家的交易模型，对顾客而言商家可以开立交易凭据使得顾客保有顾客权益，同时也保有顾客的匿名性，但缺点是顾客必须承担交易之间所增加的手续费；对于商家，可以透过交易信息管理商品库存，快速计算商家的收益；而政府则可以快速地查看商家交易信息以及商家所得。

### 3.1.3 交易比较

在上述两种现金交易模型与三种电子支付模型中可以看出，初始的交易型态为匿名顾客支付匿名商家交易模式，因无法保障顾客与商家之间的权益而发展出匿名顾客支付实名商家交易模式，这使得顾客与商家之间可以兼顾双方权益同时也保有顾客个人隐私，且可以让政府快速地查看税务相关信息。

支付技术的日新月异与 VISA 公司的出现，使得电子商务可以快速的运行，但也

因为 VISA 服务会透露太多的个人信息，顾客需求逐渐转型成以实名顾客透过匿名第三方支付实名商家为基础的交易模式，兼具以电子支付作为付款方式，且同时保有个人隐私。但其缺点是需要多经过一个资金转移的程序，成为上述五种交易模型中交易手续费最为高昂的一种，但因其能保留顾客匿名性，让透过匿名第三方支付的模式在电子支付类型中依然成为顾客的第一选择。由上述分析可知，现金法定货币最佳的交易模型为匿名顾客支付实名商家交易模式，而电子支付类型的交易模式也趋向实名顾客透过匿名第三方支付给实名商家的方向演进。

在使用加密货币作为交易货币的支付中，大部分的交易类型皆为如现金交易类型中匿名顾客支付匿名商家的交易模式，商家与顾客交易的过程中并无开立交易凭据，使得顾客与商家发生消费纠纷时因无法提出交易事实证明而无法拥有顾客权益保障。为了在使用加密货币时让用户可以同时兼顾顾客隐私与拥有顾客权益保障，也让商家不需要支付运营商比过去更为高昂的手续费并能以公开匿名的方式查看所有的交易信息，同时让政府在课征税收的业务上可以更加便利，本文致力于设计出三者都能同时兼顾的加密货币交易系统。表3.1为交易关系比较表。

表 3.1 交易关系比较表

|        | 顾客 | 中介单位 | 商家    | 商品    |
|--------|----|------|-------|-------|
| 现金     | 匿名 | 无    | 匿名/实名 | 匿名/实名 |
| VISA   | 实名 | 无    | 实名    | 实名    |
| 支付宝    | 实名 | 实名   | 实名    | 实名    |
| PayPal | 实名 | 匿名   | 实名    | 实名    |
| 加密货币   | 匿名 | 无    | 匿名/实名 | 匿名/实名 |

## 3.2 功能性需求分析

在现今的加密货币交易系统中采用匿名对匿名支付的交易模式，这样的交易模式顾客无法拥有权益保障，商家需支付高昂的交易手续费，政府也无法在交易中课征税。传统交易模式中所采用实名支付给实名的交易模式虽然可以有效地保障顾客权益，但在这对顾客个人隐私日趋重视的世代中，个人信息越来越有价值，个人信息的保护更是成为重要的课题。在本系统中，将以加密货币-比特币为基础，设计一个以匿名顾客支付给实名商家为基础的比特币交易监督系统。在实现匿名支付给实名的交易模型同时，也将商家的库存信息同时加入，使商家也可以轻松地使用本系统管理产品库存。如图3.3所示，在本系统中，参与者总共有三种，分别为商家、职工以及顾客。

商家，在本系统中为第一个参与者，因为必须要有商家的注册参与，才可以进一步的添加职工以及商家产品信息，如此一来商家才有商品可以贩售。参与者商家本身

有三项需求，分别为用户注册与登入、职工管理以及商家产品管理。以下将逐一说明：

1. 用户注册与登入：为了得到政府的认证，商家必须接受政府的查看，提交相关的信息到本系统中。在用户注册的页面当中，进行用户的注册或是登入都需要用户信息，因此需要包括加载用户信息。
2. 职工管理：在完成用户注册与登入之后，才得以进行职工管理，商家本身会有大于或等于一个职工的帐号，商家的注册者本身会是一个职工。在职工管理当中，需要有查找职工帐户与修改职工帐户两项功能，查找职工帐户的功能必须包括加载职工信息。在进行职工帐户修改时需要包括职工信息以及商家信息，才得以对职工信息进行修改。
3. 商家产品管理：商家需要添加产品信息至商家产品信息中，产品为政府认可的产品认证编号，在本系统中产品本身不存在价格，需要参与者商家将产品添加至商家产品信息中才可以添加价格，这样的设计可以让不同的商家在不同的产品设置不同的价格。除了商家对商家产品价格需求，同时也需要对产品库存进行管理，透过区块链加密货币公开交易信息的特性，可以使得库存管理以及交易信息更快地核对。在商家产品管理中需要查找产品信息，在查找产品信息的同时包括加载产品信息，使得查找过程可以顺利运作。在查找完成产品信息后，商家需要将产品信息与商家信息添加到商家产品信息当中，在这过程中需要包括加载商家信息以及商家产品信息。

职工，需要进行用户注册并且通过政府的审查，在完成审查之后需要进入商家交易管理建置移动收银机。参与者职工本身有两项需求，分别为用户注册与登入、及商家产品管理。以下将逐一说明：

1. 用户注册与登入：要成为职工之前职工需要进行用户注册，等待政府的审查之后，并经由商家进行职工管理将用户与商家一并提交到职工信息，才得以成为正式的职工，职工的用户注册与登入皆需要包括加载用户信息，才能使得注册与登入功能顺利运行。
2. 商家交易管理：在完成登入后，商家交易管理将会加载相关信息，包括加载职工信息，在交易信息中添加职工编号，加载商家信息使得在进行交易的过程中可以将商家的比特币地址发送给顾客等待接收款项以及加载商家产品信息使得职工的手持移动装置上可以显示所有的商家产品信息，在完成信息加载后职工的手持移动装置已经成为一台移动收银机。在职工在进行扫码的过程会创建交易清单并且将交易清单发送给顾客等待顾客的付款。在等待付款时，商家交易管理需要认证该笔交易是否有效，因此需要加载交易信息，

顾客，为了保持顾客的身份匿名，参与者顾客与参与者职工和商家不同，顾客在

参与本系统的同时不需要注册帐户以及登入用户帐户，顾客主要需求为加载过去与顾客相关的交易信息，以及使用比特币支付进行付款：

顾客交易管理：顾客需要加载交易信息，但因为交易信息中的信息并无详细阐述商家产品信息，因此需要进一步加载商家产品信息使得交易明细更加的清楚。除了显示过去的交易信息的需求，顾客更需要创建一笔交易以比特币进行付款。待付款完成之后，等待参与者商家向顾客回复支付完成即确立该笔交易完成。

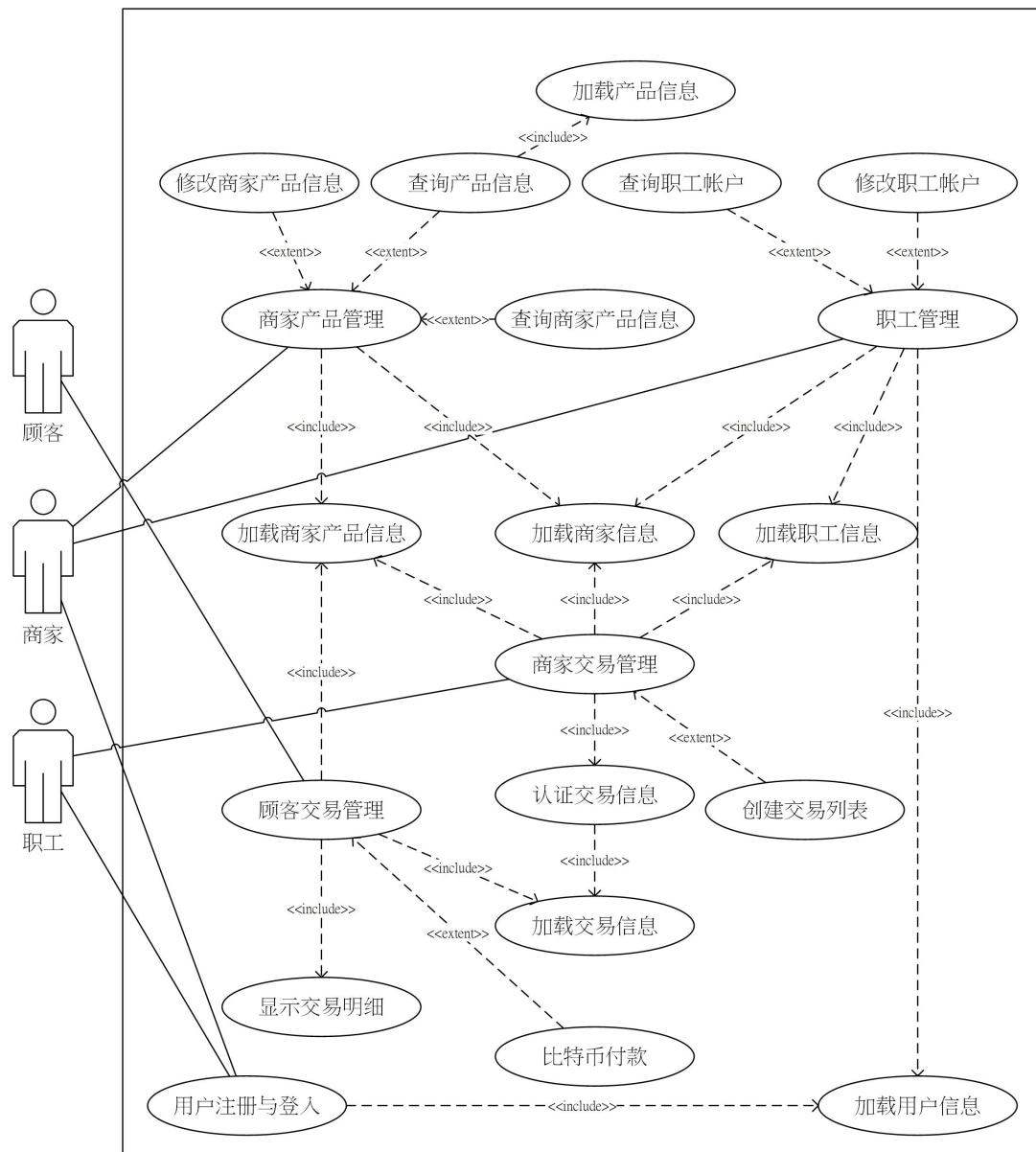


图 3.3 比特币交易监督系统用例图

### 3.3 非功能性需求分析

#### (一) 性能需求

本文所实践之系统为比特币的交易监督系统，参与者商家需要使用到商家和商品信息管理子系统，在参与者商家操作该子系统时，系统在设置完成后，于存储的过程中应于三秒内完成。于商家和商品信息管理子系统，商家端行动收银与交易明细系统应达到于两秒内完成子系统与数据库之间的信息传递以及信息存储，倘若商家与顾客在完成交易后，却无法得到快速的系统响应，这使得参与者顾客与职工收到支付完成的回复，甚至导致商家的交易塞车影响用户的消费体验。

#### (二) 可拓展性

本论文提出的系统为比特币的交易监督系统雏形，于系统设计中将优先考虑最必要功能并加以实现，包括商家和商品信息管理子系统、商家手持移动装置收款及交易子系统以及客户端行动支付和交易子系统，于上述三个子系统中皆可以进行数据库字段的添加，使得交易信息、商家信息、商品信息、商家产品信息、用户信息以及职工信息更加的完备且更符合实际上业务需求。在未来甚至可以加入更完整的税务信息，使政府在税务计算方面可以大幅降低人力资源成本且更快速地进行税务统计，而库存管理方面可以加入库存预测的服务。

#### (三) 可操作性

在本系统中的三个子系统中，为了将商家端行动收银与交易明细系统，以及客户端行动支付与交易明细系统实现在手持移动装置 Android 操作系统上，于界面设计方面应给予商家用户以及顾客用户人性化的操作交互设计，以及于代码撰写方面应更加简洁，使得用户在启动本移动端应用程序时可以在最短的时间内完成初始化程序加载。除了针对用户交互方面的优化，也需要添加各个功能项目的使用说明，以及用户操作问题反馈，给予使用本系统的手持移动装置在应用程序上能快速的修正以提升用户体验。商家端建置与管理商品信息子系统是以 Java 编程语言实现，针对参与者商家操作进行设计，该子系统需简单明确的呈现商家信息、产品信息以及商家产品信息，使参与者商家可以快速地查看所有商品，并添加、修改以及删除相关产品信息，包括价格、库存以商家产品描述。

#### (四) 软件与硬件环境需求

表3.2为本文提出系统的软件与硬件环境需求，商家交易客户端与顾客交易客户端是建置在 Android 手持移动装置的应用程序，本文提出的交易方式为比特币，因此导

入 bitcoinj<sup>[56]</sup> 对比特币钱包进行管理，该套件可以生成比特币地址、同步比特币区块链、透过 AES 加密算法保护比特币钱包以及发起比特币交易至比特币网络。针对数据库的控制数据传输选用 JDBC<sup>[57]</sup> 实现数据库的添加、修改、删除以及查找的功能，商家交易客户端与顾客交易客户端的手持移动装置必须要内置 NFC 监听器用来发送与接收交易信息。商家管理客户端是以 Java 编程语言实现，软件环境中需支持 Java 工作环境，服务器端则需要 SQL 环境进行数据库搭建。

表 3.2 环境需求表

| -               | 操作系统                    | 软件   | 硬件   |
|-----------------|-------------------------|--|--|
| 商家交易客户端与顾客交易客户端 | Android 4 或以上           | Java 7 或以上<br>bitcoinj v0.14.6<br>JDBC 4.2 或以上<br>Maven 3+         | 存储容量: 32 GB<br>内存容量: 2 GB<br>网络需求: 10 兆或以上<br>传感器: NFC 监听器         |
| 商家管理客户端         | Windows 7 或以上           | Java 7 或以上<br>JDBC   | 硬盘容量: 100 GB<br>内存容量: 4 GB<br>处理器: Core 2 Duo 或以上<br>网络需求: 10 兆或以上 |
| 服务器端            | Windows Server 2003 或以上 | Microsoft SQL Server<br>bitcoin-qt 0.8.X 或以上<br>Apache HTTP Server | 硬盘容量: 1000 GB<br>内存容量: 16 GB<br>处理器: Xeon E3 或以上<br>网络需求: 100 兆或以上 |



## 第四章 系统概要设计

本论文提出比特币的交易监督系统(Bitcoin Transaction Monitoring System, BTMS)<sup>[58]</sup>, 以下简称 BTMS, BTMS 以加密货币比特币实作, BTMS 包含三个子系统, 商家和商品信息管理子系统 (Store and Merchandise Information Management Sub-System, SMIMSS)、商家手持移动装置收款及交易子系统 (Store Mobile payment Collection and Transaction Sub-System, SMCTSS) 、客户端行动支付和交易子系统 (Client Mobile Payment and Transaction Sub-System, CMPTSS), 图4.1为主系统与子系统的功能模块图。

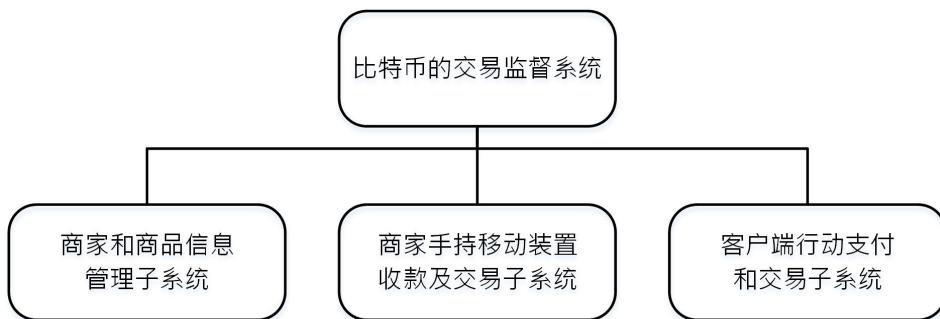


图 4.1 BTMS 系统功能模块图

本论文设计让商家能够结合商品与 RFID 标签, 以达到快速建构与管理商品数据库之系统, 并且让商家及顾客可以运用手持移动装置 NFC 功能来实际运作比特币的行动支付流程。商家只需要扫描商品上的 RFID 标签, 即可快速创建交易清单, 再利用 NFC 功能与顾客之行动设备进行信息交换, 轻松地将商家的收款地址以及交易数据发送给顾客, 收到数据后便能快速地以顾客之比特币行动电子钱包付款, 并将交易细项保存下来, 以便未来商家与顾客能够快速查找比特币行动支付的交易记录。本系统主要是以完成区块链之加密货币的收款监督系统为主要目标, 本论文进而将积极利用自由软件的利基: 使用成本低、进入门槛低、开放源代码、社区能力强、共通性及移植性强、资通安全性高等优势来开发本论文收款监督系统的应用服务平台。本系统范围包含建置下各项子系统如下:

1. 商家和商品信息管理子系统 (Store and Merchandise Information Management Sub-System, SMIMSS): 本系统可以让商家在进货时, 快速地将 RFID 标签之识别码与进货商品信息集成在一起, 并且透过本系统添加、修改或删除数据库内部的信息, 包括产品名称、详细信息, 存货数量等信息, 商家与顾客便可依照该数据库取得当前商品信息与状态。不仅让商家的存货信息更加清楚明了, 也可以提供顾客更多的即时服务, 图4.2所示。



图 4.2 商家和商品信息管理子系统功能模块图

2. 商家手持移动装置收款及交易子系统（Store Mobile payment Collection and Transaction Sub-System, SMCTSS）：本系统使商家在结帐时，能够以手持移动装置 NFC 功能扫描商品上的 RFID 标签，即可简单地创建交易清单，并透过 NFC 与顾客手持移动装置碰触，将交易清单以及商家之比特币收款地址等等重要交易信息一并传递给顾客，可以减短结帐的速度，使结帐效率大幅提升，图4.3所示。



图 4.3 商家手持移动装置收款及交易子系统功能模块图

3. 客户端行动支付和交易子系统 (Client Mobile Payment and Transaction Sub-System, CMPTSS): 顾客在结帐时, 不必再麻烦的拿出信用卡或是零钱包, 只需要拿出手持移动装置让职工以 NFC 将交易清单与比特币地址转送给自己, 即可自动链接至比特币电子钱包的应用程序当中, 并且自动填妥相关数据, 如: 交易金额、收款地址等等与此同时也能将交易纪录保存于客户端, 以便日后顾客快速取得过往的交易纪录, 除此之外亦可让广大的民众体验加密货币与行动支付带来的便利生活, 图4.4所示。

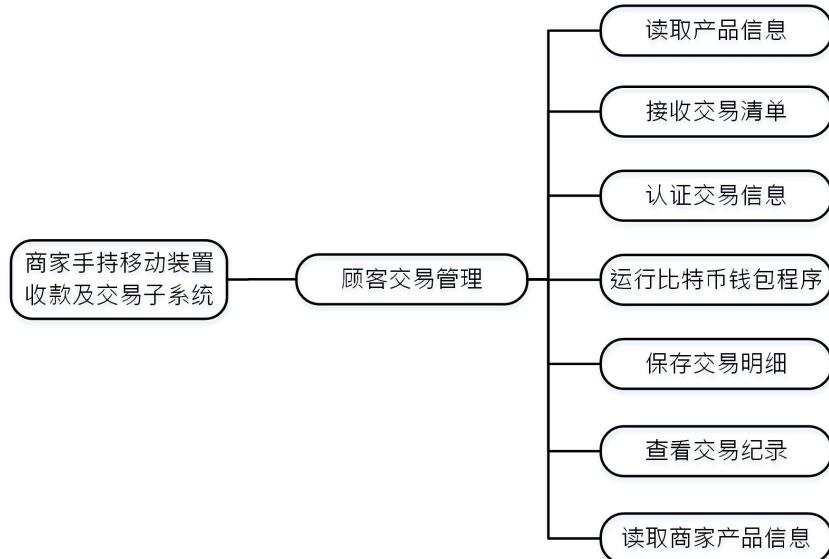


图 4.4 客户端行动支付和交易子系统功能模块图

## 4.1 BTMS 架构与运作流程

图4.5为 BTMS 和商家注册流程架构图, 商家需要在以下 4 个步骤中对用户注册与登入:

1. 商家必须在 BTMS 注册一个用户, 并附有政府法规的商业证明。
2. 比特币的交易监督系统将自动向相应的政府金融监管机构提交商业申请, 以审查该商家的加密货币交易业务。
3. 如果政府批准商家的加密货币业务申请, 服务器将激活商家在该收集监控系统中创建商家用户。
4. 商家可以自由地登入帐户并添加商家想要出售的产品信息, 亦可透过职工管理添加修改删除职工信息。

具体的加密货币商家收银金流监控系统运行过程如图4.6所示, 以图4.5所设计出的系统架构进行扩增, 首先需要与区块链检视器 (Blockchain Explorer)<sup>[59]</sup> 对接, 而之所

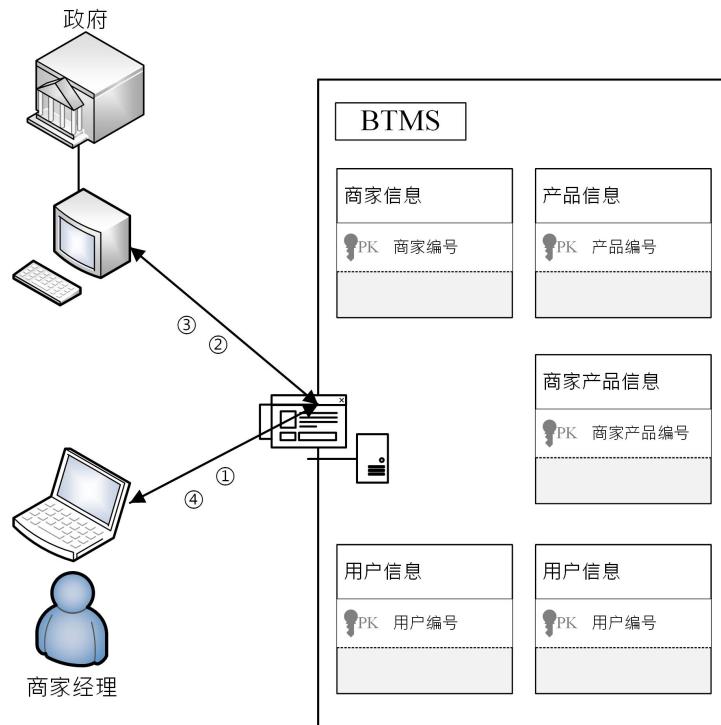


图 4.5 BTMS 和商家注册流程架构图

以该监控系统需要与区块链检视器对接是为了能够最直接的比对交易被记录的成交状况，能够达到即时性以及正确性，倘若担忧单方面的依赖相信区块链检视器的成果，亦可以使用多家区块链检视器进行交叉参考，以避免因为一家公司的错误所带来的影响。区块链检视器的目的是为了能够快速地准确地比对该笔交易的成交，做为整笔交易提出到结算的环节之一。

图4.6比特币的交易监督系统创建步骤描述如下：

1. 商家职工将登入到图4.5中的第一个步骤，所示的先前步骤创建的用户，使用手持式平板电脑或手持移动装置访问 SMCTSS 中的服务。如前所述，在能够登入到系统之前，商家用户必须由政府机构审计。
2. 在成功登入 SMCTSS 用于商家加密货币金流监控系统时，手持移动装置将加载通过 SMIMSS 注册的商家产品信息，然后创建产品目录。商家的职工可以根据顾客的需求选择所需的产品和数量。
3. 职工使用设备完成顾客选定商品的产品信息后，手持移动装置上的 NFC 技术可用于将产品信息从附近职工的手持移动装置传递给顾客的手持移动装置，而无需物理交互。然后，顾客可以很容易地将自己的消费信息记录成为交易凭证等参考。在接收从商家职工设备向顾客设备购买产品信息之后，顾客设备将向商家的手持移动装置发送其自己的比特币支付地址的信息。

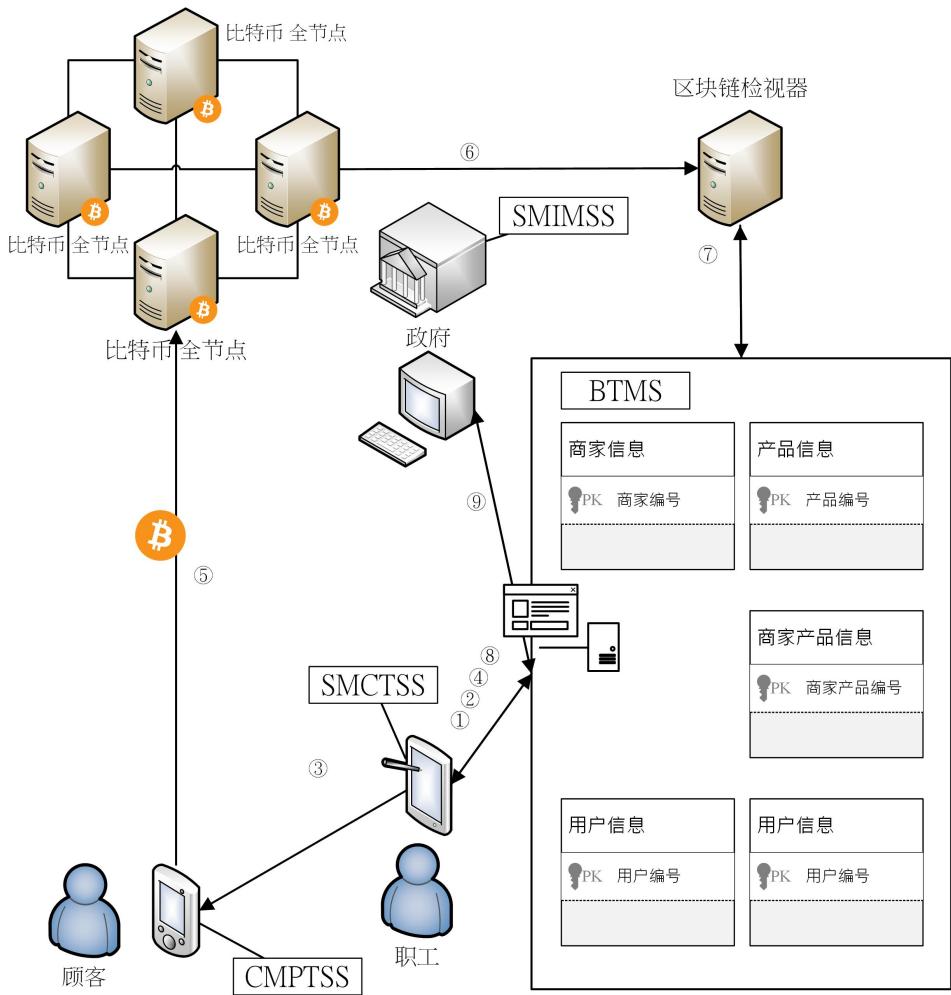


图 4.6 BTMS 的整体流程架构图

4. 商家的手持设备收到顾客确认购买所选产品的相应信息后，会将交易信息的副本发送给 SMIMSS 监控系统。顾客信息包括交易串行号、商家 ID 号码、商品号码、购买的商品的数量以及加密货币的收款人地址以及顾客的支付地址。
5. 收到顾客交易信息后，即完成此次的加密货币支付。同时，此次交易的加密货币将透过客户端 CMPTSS 发布到比特币网络中进行验证和记录。
6. 区块链检视器将开始分析在比特币网络中缓存池的所有交易以及区块链中记录的交易。
7. 拟议的交易监控系统 BTMS 将向区块链检视器提出请求。这个请求数据不仅包括存储在 BTMS 中的交易副本之加密货币收款人地址，如图4.6的第四步骤，还包括顾客预期付款的加密货币支付地址。区块链检视器使用请求数据来检查交易是否存储在区块链中，或者交易还在等待确认。如果交易已被确认并存储在区块链中，则交易数据库中校验值字段的值将更改为"1"，否则其默认值为"0"。
8. 当校验值字段中的值为"1" 时，交易已完成消息可发送至商家平板电脑上运行的商家和商品信息管理子系统（SMIMSS）。
9. 政府财政监督部门可以审查拟议 BTMS 中的所有交易信息，以作为税收审计参考。

## 4.2 实时 BTMS 架构与运作流程

在 BTMS 的机制前提下，提出比特币多重签章算法实践于政府端，将该方法称为 Government Green Address，将优化后的系统称为比特币的实时交易监督系统（Bitcoin Real-time Transaction Monitoring System, BRTMS），以下则将该实时系统简称为 BRTMS，BRTMS 与过去 Green Address 比特币钱包地址不同的在于 Green Address 生成需要生成两把私钥分别为用户私钥以及 Green Address 机构的私钥，并将两把私钥加以合并形成 Green Address 钱包地址。在用户发起基于 Green Address 机制的比特币交易时，用户生成交易并使用数字签名，再将该笔交易发送至 Green Address 机构进行下一个步骤的数字签名，在这过程中完成 Green Address 机制交易必需依赖 Green Address 机构，且该机构位于海外，在封包传输上会有一定程度的延迟。为了提升国内用户可以更快速地透过多重签章技术提升交易速度，BRTMS 将取代原有的 Green Address 机构，使得国内的用户拥有更快的网络传输速度，更快的完成多重签章算法签名。

本节将详述本系统之商家注册、Government Green Address 钱包创建与验证交易的运作流程与相关数据库架构，如图4.7所示。

1. 商家以通过政府机构的审查审核的用户登入该系统。
2. 系统加载该商家注册的商品信息，职工可以依照顾客的需求进行点单选取数量。

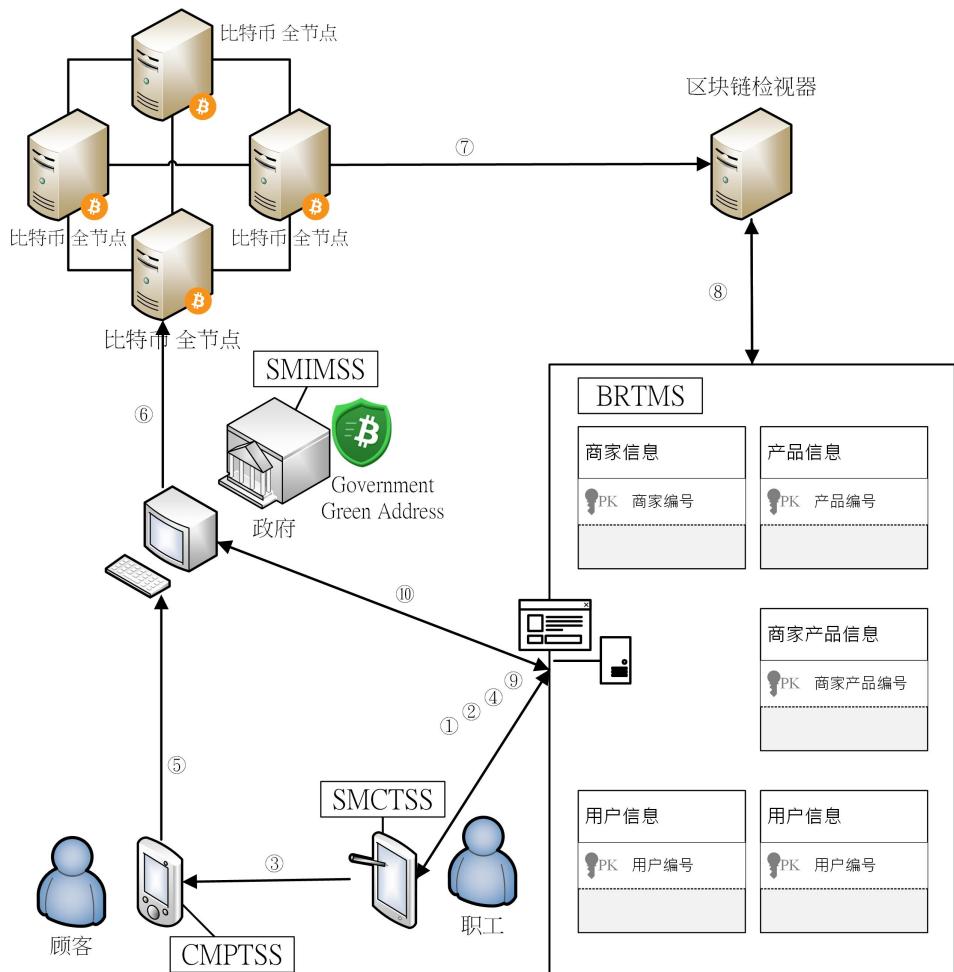


图 4.7 基于 Government Green Address 的 BRTMS 整体流程架构图

3. 快速创建交易清单，并透过 **Government Green Address** 创建一个全新的比特币收款地址，再以 **Android Beam** 的方式将交易信息轻松地传达给顾客。
4. 在商家职工的平板电脑收到这笔交易信息之后，会对本监督系统重送一个副本进行存盘。该交易信息包括由监督系统所提出的交易流水号、商家编号等信息。
5. 顾客收到交易信息后，手持移动装置会自动开启 **Government Green Address** 的付款页面，确认金额无误之后便能进行支付，此时便会以顾客的比特币私钥签署交易，并等待 **Government Green Address** 机构节点的认证及发布。
6. **Government Green Address** 节点收到交易请求，并完成验证非双重支付攻击后，以代理节点对应地址的私钥签署本次交易，并广播至比特币节点中。
7. 区块链检视器便会开始分析网络中所有存在缓存池中的交易，以及已经被记录到区块链中的交易。
8. 本交易监督系统会向区块链检视器查找，检查该笔交易是否已经存在于缓存池当中，若已经确认进入缓存池，则认定该笔交易成立并完成付款。
9. 在交易确认之后，便向商家职工的平板电脑送出交易已经成交的信息，此时完成交易，于此同时也将该笔交易信息建置于系统数据库内。
10. 后续政府可以连入 **BRTMS** 系统查看交易信息。

## 第五章 系统详细设计与实现

在前章中已将本文的系统设计概要架构说明，在本章中将对数据库的内容信息设计进行详细的说明、数据库 ER 模型架构、各个模块的类与类之间的关系、每个模块运作的顺序以及模块与模块之间的响应关系，最后则是系统实现的平台展示。

### 5.1 BTMS 数据库设计

比特币的交易监督系统应用了六个信息表，分别为商家信息、产品信息、交易信息、用户信息、职工信息以及商家产品信息，以下将逐一说明：

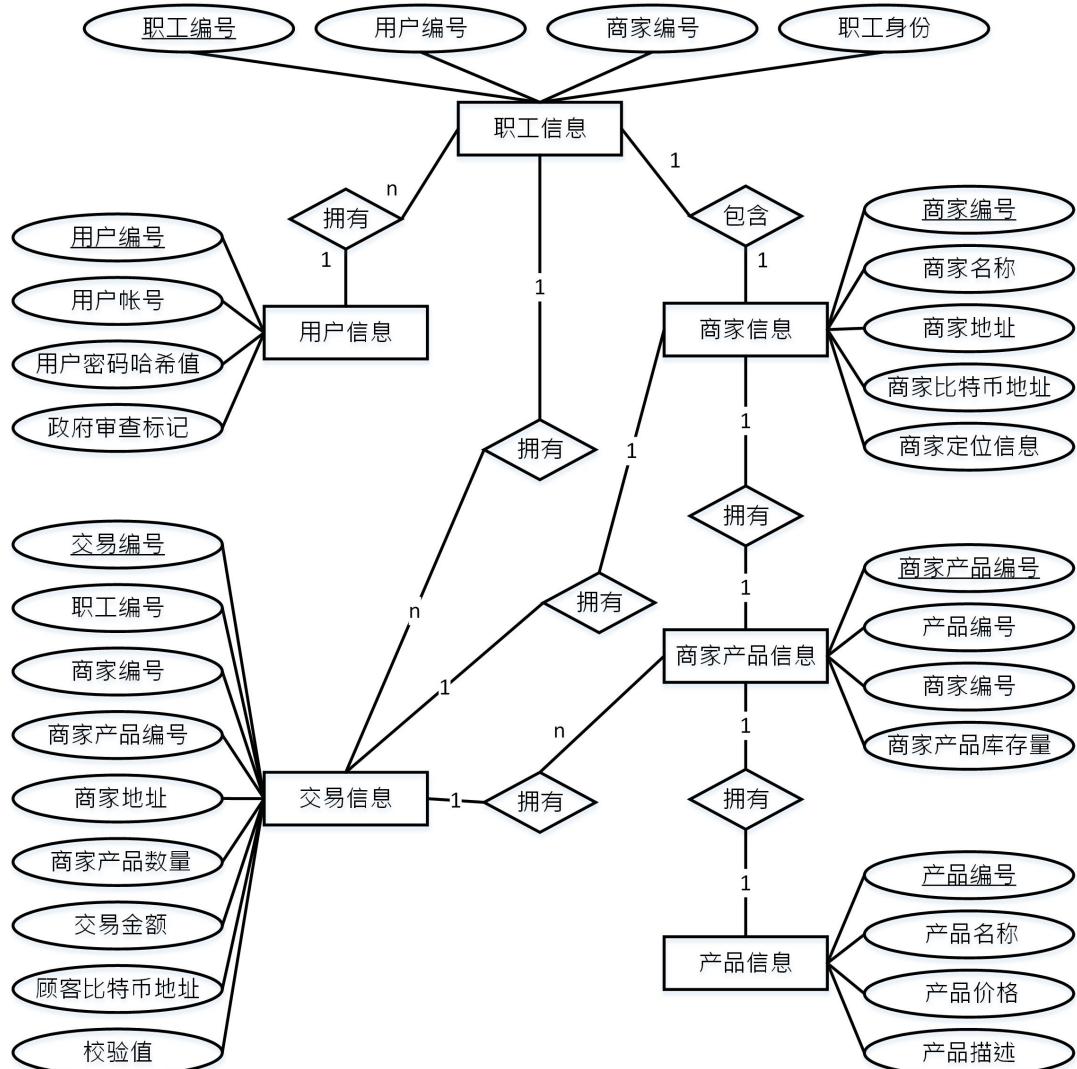


图 5.1 BTMS 数据库实体关系图

1. 商家信息表：存储正在审核中的企业信息或已经过审核的企业信息。表5.1存储的信息包括商家编号、商家名称、商家地址、商家比特币地址以及商家定位信息。

表 5.1 商家信息表

| 序号 | 字段名               | 字段说明    | 类型            | 是否为空 | 主外键 |
|----|-------------------|---------|---------------|------|-----|
| 1  | STORE_ID          | 商家编号    | int           | 否    | PK  |
| 2  | STORE_NAME        | 商家名称    | navarchar(20) | 否    |     |
| 3  | STORE_ADDRESS     | 商家地址    | navarchar(50) | 否    |     |
| 4  | STORE_BTCAADDRESS | 商家比特币地址 | navarchar(50) | 否    |     |
| 5  | STORE_GPS         | 商家定位信息  | navarchar(30) | 否    |     |

2. 产品信息表：只有授权用户才能登入添加或修改交易产品信息。表5.2产品信息表内容包括产品编号、产品名称、产品价格以及产品描述信息。

表 5.2 产品信息表

| 序号 | 字段名                 | 字段说明 | 类型            | 是否为空 | 主外键 |
|----|---------------------|------|---------------|------|-----|
| 1  | PRODUCT_ID          | 产品编号 | int           | 否    | PK  |
| 2  | PRODUCT_NAME        | 产品名称 | navarchar(10) | 否    |     |
| 3  | PRODUCT_PRICE       | 产品价格 | float         | 否    |     |
| 4  | PRODUCT_DESCRIPTION | 产品描述 | navarchar(50) | 是    |     |

3. 交易信息表：表5.3记录包括交易编号、职工编号、商家编号、商家产品编号、商家地址、商家产品数量、交易金额、顾客比特币地址和最后确认字段的校验值。

表 5.3 交易信息表

| 序号 | 字段名                   | 字段说明    | 类型            | 是否为空 | 主外键 |
|----|-----------------------|---------|---------------|------|-----|
| 1  | TX_ID                 | 交易编号    | int           | 否    | PK  |
| 2  | STAFF_ID              | 职工编号    | int           | 否    | FK  |
| 3  | STORE_ID              | 商家编号    | int           | 否    | FK  |
| 4  | STOREPRODUCT_ID       | 商家产品编号  | int           | 否    | FK  |
| 5  | STORE_ADDRESS         | 商家地址    | navarchar(50) | 否    | FK  |
| 6  | STOREPRODUCT_QUANTITY | 商家产品数量  | int           | 否    |     |
| 7  | TX_AMOUNT             | 交易金额    | float         | 否    |     |
| 8  | CONSUMER_BTCAADDRESS  | 顾客比特币地址 | navarchar(50) | 否    |     |
| 9  | CHECK                 | 校验值     | bool          | 否    |     |

4. 用户信息表：表5.4存储所有用户信息，包括政府、商家及顾客之个人的帐户编号与帐号，而用户密码则以哈希值的方式保存以增加用户安全性，最后则是政府审查值，倘若通过为"1"，未通过为"0"。
5. 职工信息表：表5.5信息表存储各个商家拥有的职工信息，包括各职工编号、用户编号、商家编号及职工身份。

表 5.4 用户信息表

| 序号 | 字段名               | 字段说明    | 类型           | 是否为空 | 主外键 |
|----|-------------------|---------|--------------|------|-----|
| 1  | USER_ID           | 用户编号    | int          | 否    | PK  |
| 2  | USER_ACCOUNT      | 用户帐号    | nvarchar(30) | 否    |     |
| 3  | USER_PASSWORDHASH | 用户密码哈希值 | nvarchar(30) | 否    |     |
| 4  | GOVT_AUTH         | 政府审查    | bool         | 否    |     |

表 5.5 职工信息表

| 序号 | 字段名          | 字段说明 | 类型           | 是否为空 | 主外键 |
|----|--------------|------|--------------|------|-----|
| 1  | STAFF_ID     | 职工编号 | int          | 否    | PK  |
| 2  | USER_ID      | 用户编号 | int          | 否    | FK  |
| 3  | STORE_ID     | 商家编号 | int          | 否    | FK  |
| 4  | STAFF_STATUS | 职工身份 | nvarchar(20) | 否    |     |

6. 商家产品信息表：表5.6存储各家商家当前商家产品存货信息，由商家产品编号、产品编号、商家编号及商家产品库存量所组成。

表 5.6 商家产品信息表

| 序号 | 字段名                    | 字段说明    | 类型  | 是否为空 | 主外键 |
|----|------------------------|---------|-----|------|-----|
| 1  | STOREPRODUCT_ID        | 商家产品编号  | int | 否    | PK  |
| 2  | PRODUCT_ID             | 产品编号    | int | 否    | FK  |
| 3  | STORE_ID               | 商家编号    | int | 否    | FK  |
| 4  | STOREPRODUCT_INVENTORY | 商家产品库存量 | int | 否    |     |

## 5.2 系统模块设计

在本系统中共有三种用户，分别为顾客、商家以及职工，首先是用户注册与登入模块，其余总共有四个管理模块分别为商家产品管理模块、职工管理模块、商家交易管理模块和顾客交易管理模块，以下将说明各个模块类图设计以及时序图运作。

### (一) 用户注册与登入模块

在本系统中仅职工与商家需要进行注册，并且需要经过政府的审查批准。职工与商家皆为用户，皆可使用用户注册模块。图5.2为用户注册与登入模块类图，在 LoginManagement 类当中分别需要调用 RegistNewUser 类实现用户注册以及 LoginUser 类实现用户登入。RegistNewUser 类中 hash() 方法是将用户输入的用户密码使用哈希算法生成用户密码哈希值，addnewuser() 方法则是将用户帐号和密码发送至 User 类。在 LoginUser 类中 getuserpasswordhash() 方法是向 User 类取得该用户帐号的密码哈希值。

图5.3为职工与商家注册时序图，以下为流程说明：

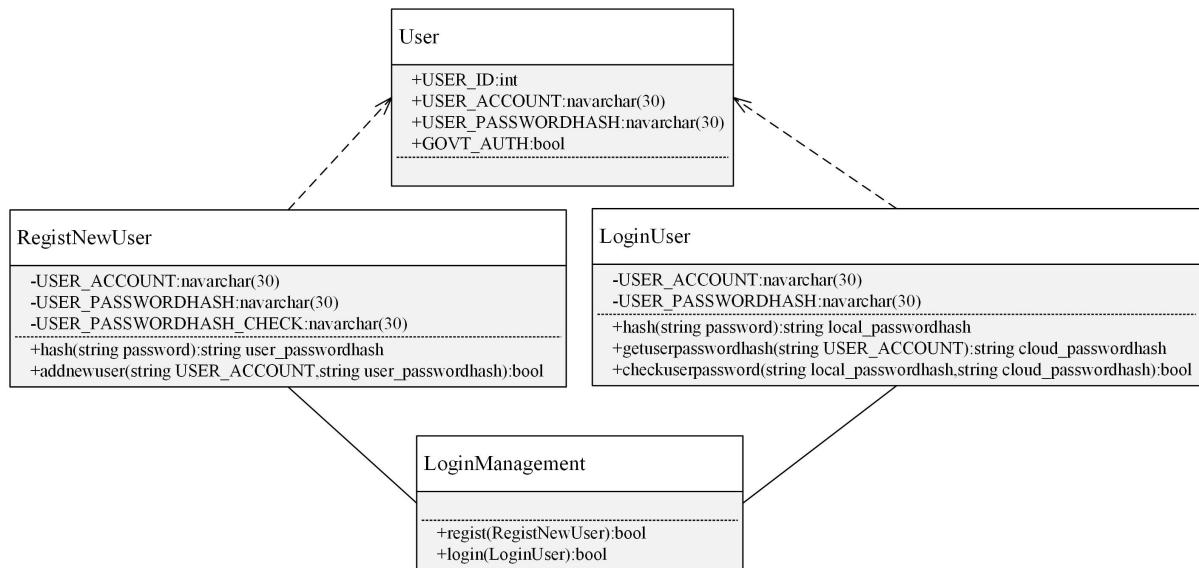


图 5.2 用户注册与登入模块类图

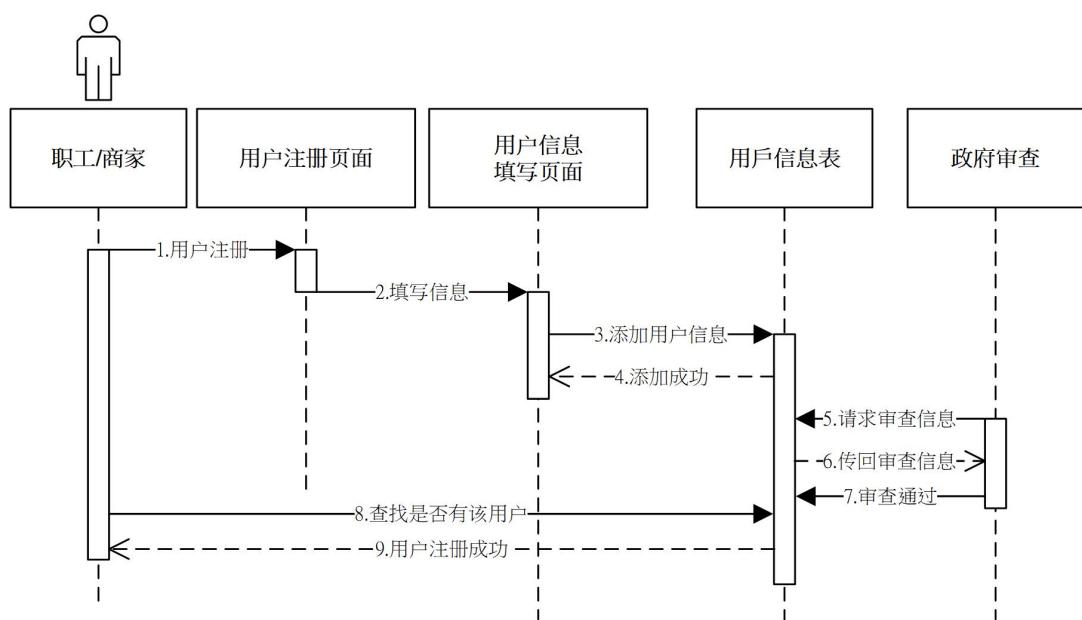


图 5.3 职工与商家注册时序图

1. 首先职工或是商家前往用户注册页面。
2. 前往用户信息填写页面填写用户帐户信息以及用户密码。
3. 完成填写后，为了提升用户密码信息安全，将用户密码以哈希算法计算后以密码哈希值保存。将填写完的帐号信息以及密码哈希值提交到用户信息表。
4. 用户信息表则传回添加成功。届时的用户信息已经存储到用户信息表内但是尚未被激活，需要等待政府进行审查。
5. 政府向用户信息请求表拿取所有还未被激活的用户信息。
6. 数据库中的用户信息表传回，政府请求的用户信息清单。
7. 政府进行用户审查，在完成用户审查后，则向用户信息表内输入审查通过。
8. 此时，用户再次访问用户信息表当中，用户帐号是否已经激活成功。
9. 用户信息表回传已经验证通过。

## (二) 商家产品管理模块

商家的运营需要管理本身所需贩售的商品。在商家完成注册用户帐号并且通过政府审核之后，便可查看所有的产品，进一步可以透过产品管理模块添加、修改以及删除商家商品。图5.4为商家产品管理类图，其中 `StoreProductManagement` 类中有四个方法需要实现，分别为显示商家信息的 `showstoreinfo()` 方法、显示商家产品信息的 `showstoreproductinfo()` 方法、显示产品信息的 `showproductinfo()` 以及修改商家产品信息的 `edit()` 方法。其中 `StoreProductCompare` 类与 `EditStoreProduct` 类需要向 `StoreProduct` 请求商家产品信息，`StoreProductCompare` 类中的 `compare()` 方法是向商家产品信息表取得与 `STORE_ID` 相符的所有相关商家产品。`EditStoreProduct` 类中的 `addstoreproduct()` 方法、`editstoreproduct()` 方法、以及 `deletestoreproduct()` 方法分别为添加、修改、以及删除商家产品信息。`StoreCompare` 类中的 `compare()` 方法是向 `Store` 类请求与 `STORE_ID` 相符的商家信息。

图5.5为商家产品管理时序图，以下为流程说明：

1. 商家在完成用户注册后并完成政府的审核之后，便可填写使用用户与用户密码到登入页面。
2. 登入页面会将用户提交之密码透过哈希算法生成密码哈希值。
3. 用户登入页面向用户信息表询问该用户帐号的密码哈希值。
4. 用户信息表将该帐户的密码哈希值传回给用户登入页面保存。
5. 用户页面将本地密码哈希值和从数据库信息表中请求保存的密码哈希值进行比对是否相同。
6. 倘若本地与数据库中的哈希值一致，则为登入成功。

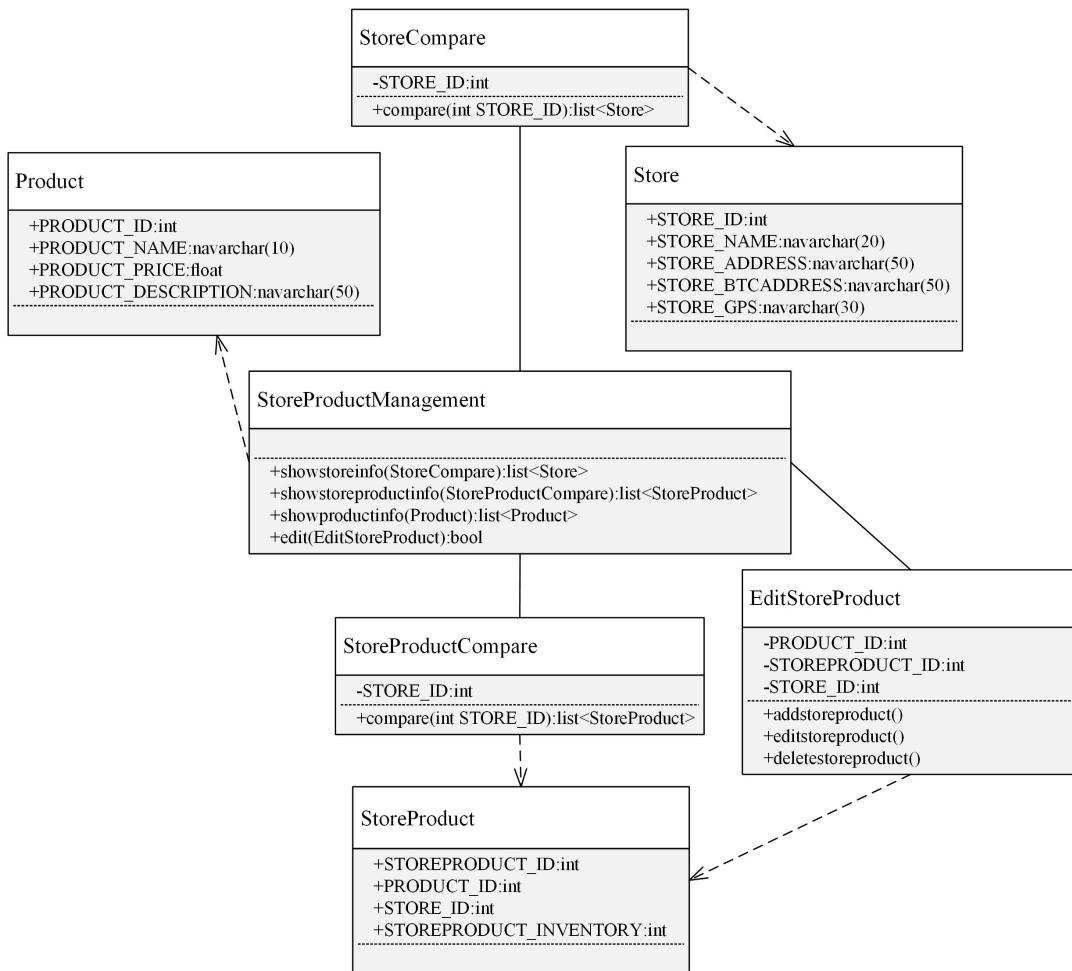


图 5.4 商家产品管理类图

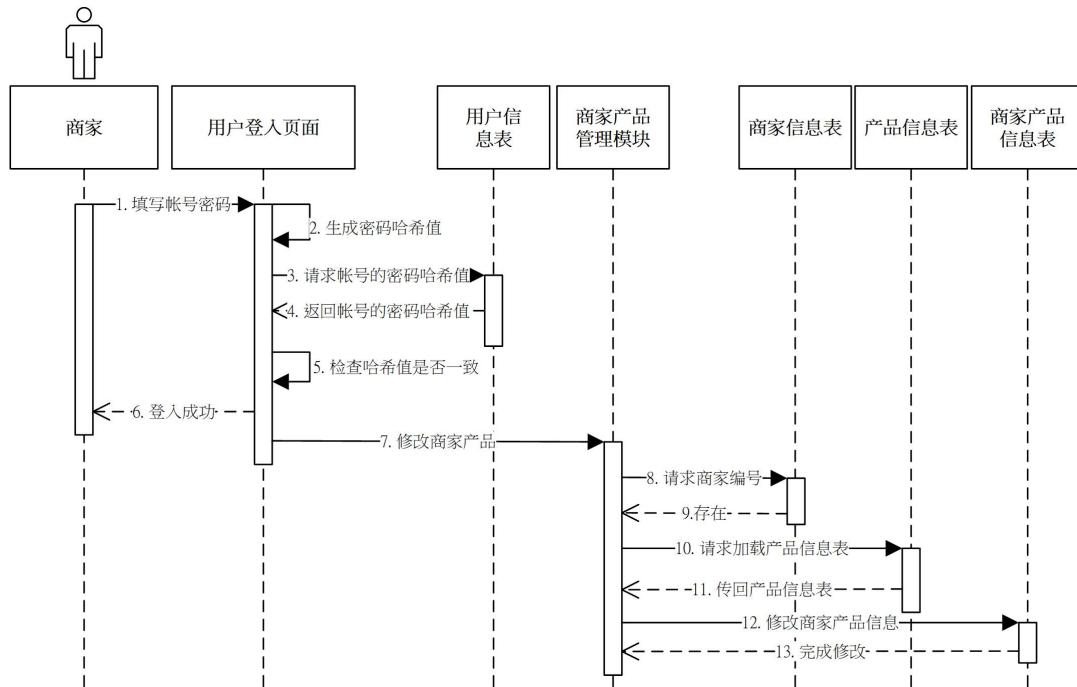


图 5.5 商家产品管理时序图

7. 届时可以进入商家产品管理模块。
8. 为确认欲修改的商家是否存在，所以向数据库中的商家信息表请求该商家编号是否存在。
9. 商家信息表返回存在的信息。
10. 向数据库中的产品信息表当中请求所有的产品信息。
11. 产品信息表回传所有的产品信息。
12. 此时商家已经确认商家信息是否存在，且已经取得所有的产品信息。商家向商家产品信息表提交商家要增加的产品编号以及商家本身的商家编号，此时生成商家产品编号。
13. 商家信息表在完成添加商家产品信息之后，传回添加成功的信息到商家产品管理模块。

### (三) 职工管理模块

商家需要职工进行商家的运营。在商家用户完成政府审查之后，便可以进入职工管理模块，透过提交用户编号以及商家编号添加、修改以及删除职工信息。图5.6为职工管理模块类图，StaffManagement 类当中有四种方法，前三种分别为显示与该商家 STORE\_ID 相符的完整信息 showstore() 方法、显示符合该商家 STORE\_ID 职工信息的 showstorestaff() 方法、显示用户编号的 showuser() 方法。而在第四种方法中，UserCompare

类需要向 User 类提取用户相关信息实现查找, StoreCompare 类需要向 Store 类请求商家信息相关数据, 其中 EditStoreStaff 类与 StoreStaffCompare 类需要使用到 Staff 类修改职工信息表的内容, EditStoreStaff 类中的 addstaff()、editstaff() 以及 deletestaff() 方法分别为添加、修改以及删除职工, StoreStaffCompare 类中的 compare 方法是取得符合 STORE\_ID 的所有职工信息。

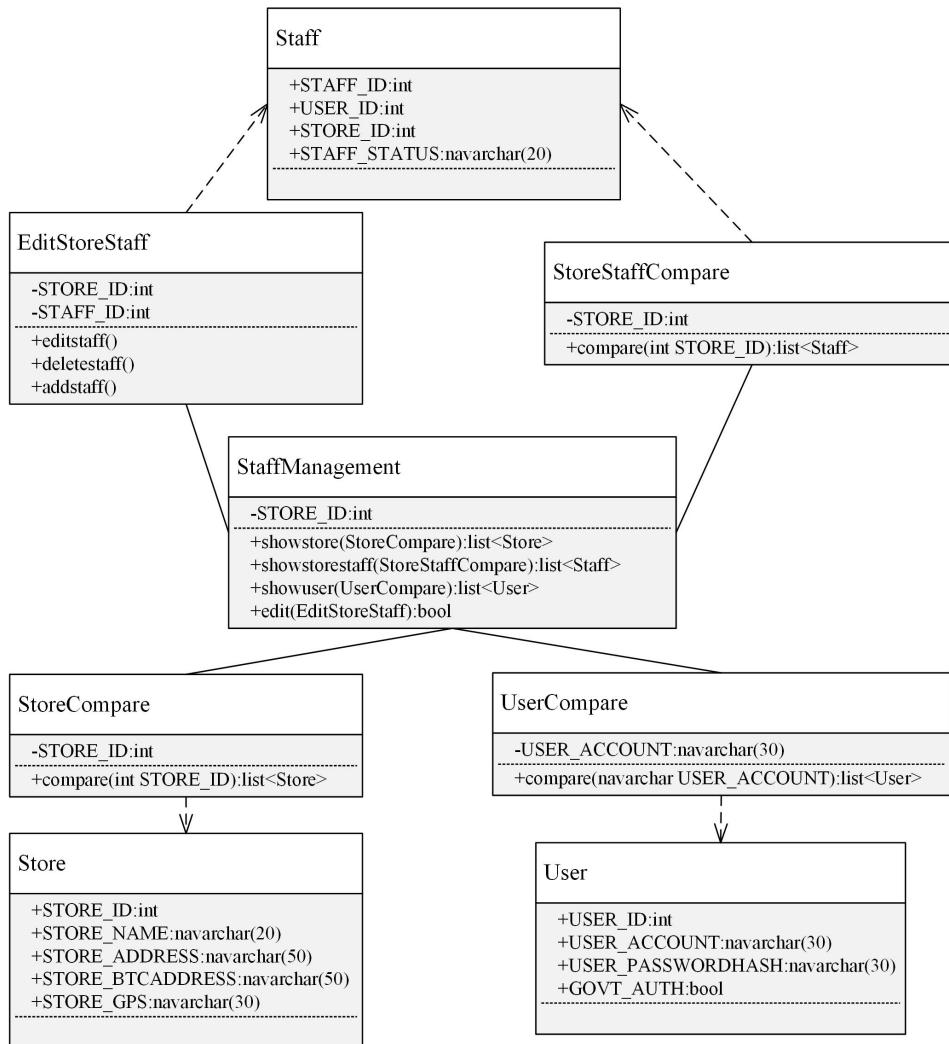


图 5.6 职工管理模块类图

图5.7为商家职工管理时序图，以下为流程说明：

1. 商家前往用户登入页面输入用户帐号密码。
2. 用户登入页面自动使用哈希算法将用户密码转换成密码哈希值。
3. 用户登入页面向数据库中的用户信息表请求该用户帐号的密码哈希值。
4. 数据库用户信息表回传用户密码哈希值。
5. 用户登入页面比对本地端的用户密码哈希值与数据库中的密码哈希值是否一致。

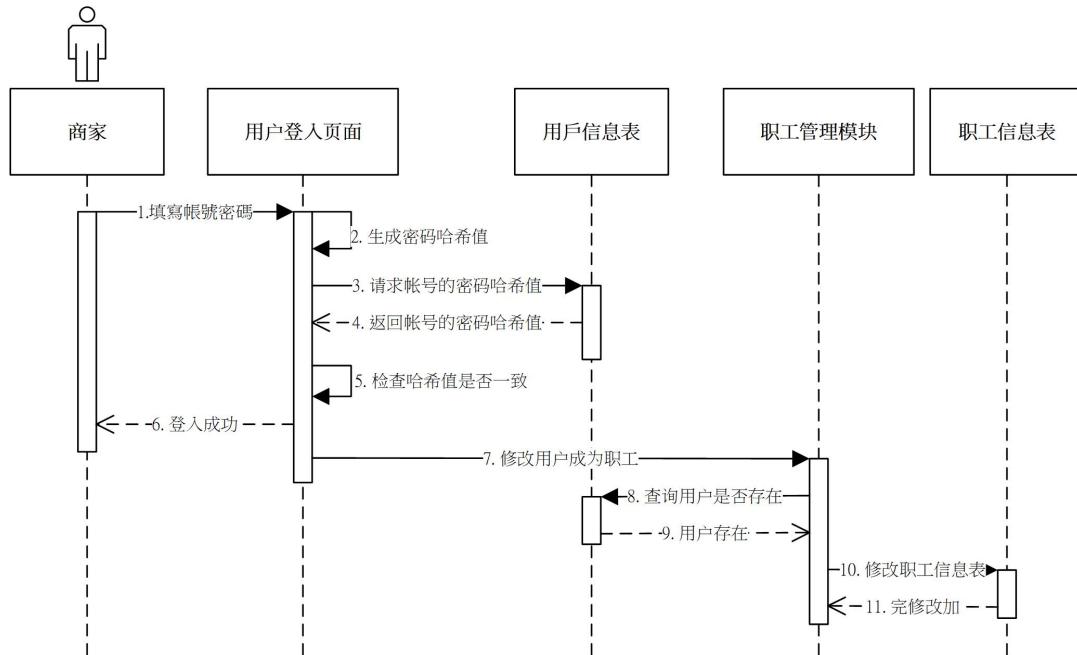


图 5.7 职工管理时序图

6. 倘若数据库中的密码哈希值与用户登入页面生成的密码哈希值一致，则登入成功。
7. 商家提交商家编号以及欲添加的用户编号。
8. 职工管理模块向用户信息表查找该用户是否存在。
9. 用户信息表传回用户存在信息。
10. 职工管理模块向数据库中的职工信息表提交用户编号、商家编号以及职工编号修改职工信息。
11. 职工信息表传回修改完成。

#### (四) 商家交易管理模块

在用户完成注册后且在商家将该用户加入成为公司职工后，该职工可以透过用户帐号进入到商家交易管理模块，该模块为手持移动装置的应用程序。在进入到本模块后会自动加载商家商品、商家信息以及职工信息，制成一个基于区块链加密货币的移动收银机，可以扫描带有RFID标签的商家产品，创建交易后等待用户交易管理模块读取以及验证交易是否完成的信息。图5.8为商家交易管理模块类图，在StoreTransactionManagement类当中有五个方法，分别为显示所有的商家产品信息的showstoreproduct()方法、显示职工信息的showstorestaff()方法、显示商家信息的showstore()方法、检查交易是否完成的check()方法以及透过NFC传输协议传输信息的nfc()方法。StoreCompare类中的compare()方法是向Store类取得与STORE\_ID相符的商家信息，StoreProduct-

Compare 类中的 compare() 方法是向 StoreProduct 类请求与 STORE\_ID 相符的所有商家产品信息。StoreStaffCompare 类中的 compare() 方法是向 Staff 类中请与 USER\_ID 相符的职工信息。在 NFCMessage 类中有两个方法，receivefcmessage() 方法使得商家可以快速的扫描商品的 RFID 标签，sendnfcmessage() 方法使得职工在创建完交易清单后可以将交易信息发送给顾客。CheckTx 类中有两个方法，分别为取得于 Tx 类中交易信息的 compare() 方法，以及 getcheck() 方法是检查该笔交易的 CHECK 值是否已经在 Tx 类中被修改为"1"，倘若被修改为"1" 则表示该笔交易已经被写入区块链中。在本模块要做到验证比特币交易是否已经被存储在比特币区块链当中，必须使用 BlockchainExplorer 类中的 getblockchain() 方法取得所有区块链相关的信息，再透过 getbitcoinctx() 方法取得该笔比特币交易的状态以及详细信息。Tx 类中可以调用 txcheck() 方法对 BlockchainExplorer 类检查该笔交易是否已经进入了区块链，如果已经进入区块链，则将相关的交易信息的 CHECK 值修改为"1"。值得一提的是，在此将 CHECK 值修改为"1" 的标准，为该笔交易进入区块链的时间。由于比特币区块链的生成速度平均为每十分钟一块，这将使得交易确认速度不够即时。

图5.9为商家交易管理时序图，以下为流程说明：

1. 首先职工填写用户的帐号密码到用户登入页面。
2. 用户登入页面模块将用户填写的密码透过哈希算法生成本地端的用户密码哈希值。
3. 将用户填写的帐号发送到用户信息表请求远程的用户密码哈希值以及商家编号。
4. 用户信息表传回远程的用户密码哈希值。
5. 用户登入页面模块进行本地用户密码哈希值和远程用户密码哈希值比对是否一样。
6. 倘若一致则登入成功。
7. 此时用户登入页面向商家交易管理模块发送商家编号的信息，商家交易管理模块将信息保存。
8. 商家交易管理模块向商家产品信息表请求与商家编号相关的所有商家产品信息。
9. 商家产品信息表传回所有与商家相关的商家产品信息至商家交易管理模块。
10. 商家交易管理模块向商家信息表请求商家比特币地址信息。
11. 商家信息表将商家比特币地址信息传回到商家交易管理模块。
12. 商家产品信息表将所有与商家编号有关的商家产品信息传回到商家交易管理模块，此时商家交易管理模块已经有完整的商家产品信息以及商家信息。将带有 RFID 标签的商品透过 NFC 线圈进行传感对应到相关的商家产品信息。
13. 将产品信息创建交易清单显示在手持移动装置的屏幕上，等待顾客的手持设备

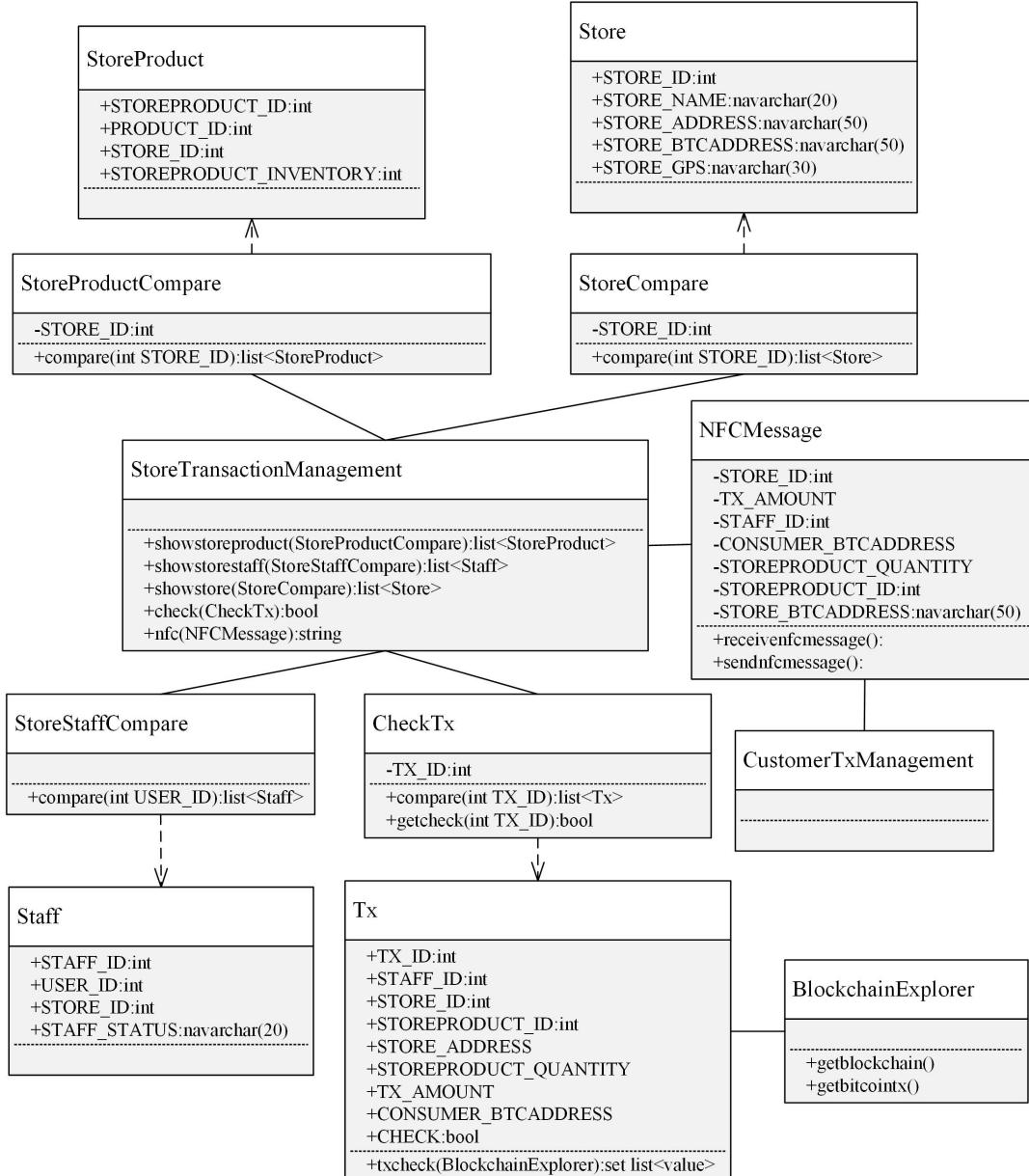


图 5.8 商家交易管理模块类图

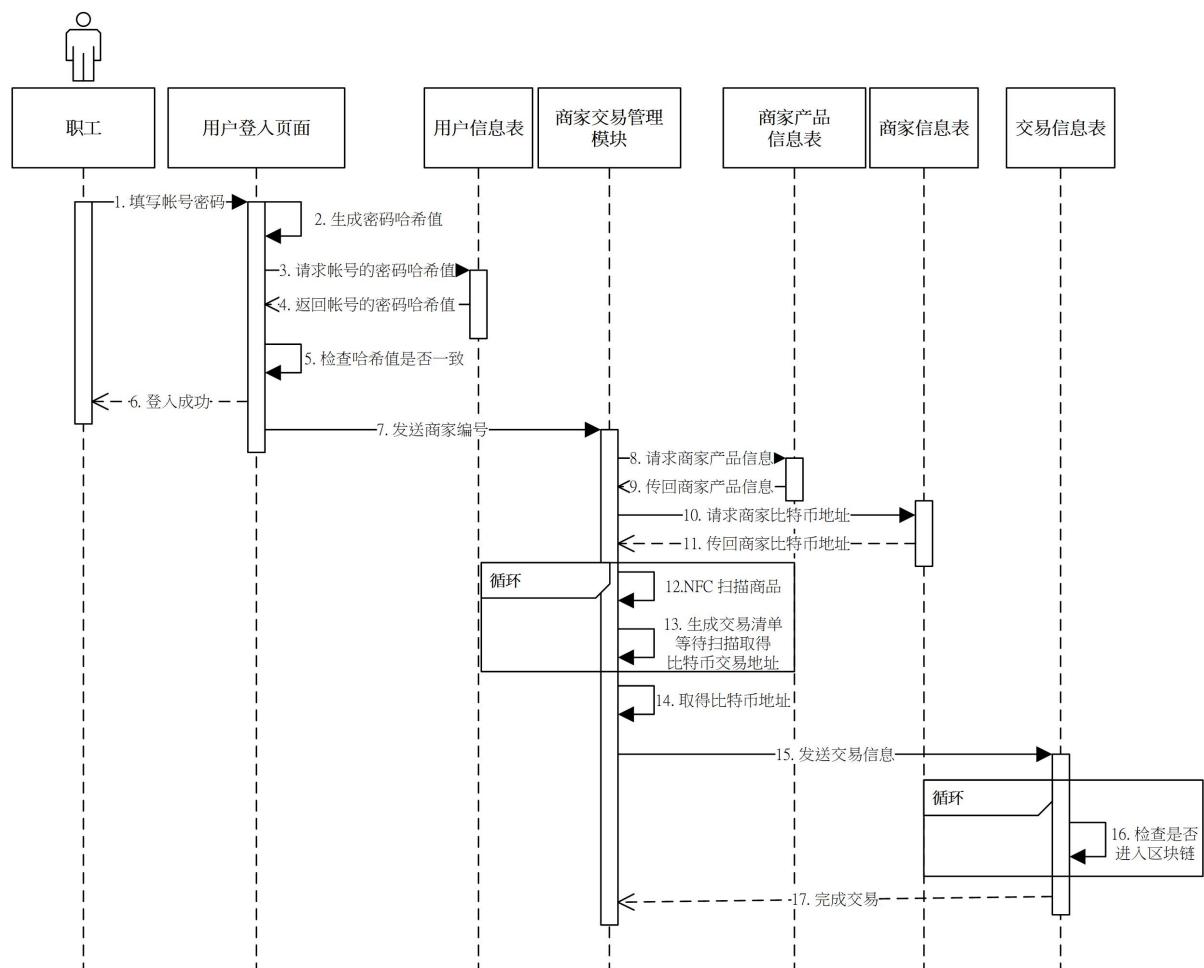


图 5.9 商家交易管理时序图

进行感应，感应的同时会将交易清单以及商家比特币地址发送到顾客交易管理模块。

14. 在感应的同时顾客交易管理模块会将比特币钱包模块中的比特币地址提交给商家交易管理模块。
15. 商家交易管理模块发送信息至交易信息表查找该笔交易信息。
16. 交易信息表不断的向区块链检视器查找该笔比特币交易是否已经被写入比特币区块链内，倘若写入区块链则将交易信息的 CHECK 信息由"0" 改为"1"。
17. 交易信息表传回该笔交易的 CHECK 已经为"1" 则完成交易。

在比特币系统中，使得一笔比特币交易得到比特币网络的认可，需要等待六个区块的交易验证，相当于需要费时 60 分钟的时间等待一个交易的完成，这使得比特币交易在商家进行小额交易造成很大的不便，因此为了在 BRTMS 中改善此问题，便导入了基于 Green Address 钱包的多重签章算法，使得交易时间可以从 60 分钟的等待时间，缩短为一秒钟左右。图5.10为商家 Government Green Address 交易管理模块类图，在此模块类图中针对 Tx 类中添加 ggatxcheck() 方法，使得商家交易管理模块可以支持 Government Green Address 地址格式的辨别，当检测到为 Government Green Address 地址格式，将交易信息中的 CHECK 字段修改为"1"，这使得比特币交易信息可以在平均一秒钟的时间得到交易确认，ggatxcheck() 方法需要调用 BlockchainExplorer 类中的 getblockchain() 取得比特币区块链的信息以及 getbitcointx() 方法得到比特币交易信息。在 GovernmentGreenaddressCheckTx 类中提供 compare() 方法可以针对 Government Green Address 地址向 Tx 类查找相关信息以及 ggatxcheck() 方法检测该笔交易是否存储于交易信息中的 CHECK 值已经被修改为"1"。

## (五) 顾客交易管理模块

在本系统中顾客需要在手持移动端安装手持移动装置程序包括顾客交易管理模块，使得顾客手持移动装置可以支持比特币支付、查找过去的交易信息、以 NFC 通信协议接收商家交易管理模块创建的交易信息以及详细的商家商品信息。图5.11为顾客交易管理模块类图，在 CustomerTxManagement 类中包括六个方法，分别为显示所有该用户拥有比特币交易信息相关的交易明细的 showtx() 方法、取得交易信息后取得相对应的商家产品信息的 showstoreproduct() 方法、检查于交易信息表中的交易信息 CHECK 字段的值是否已经被填入"1" 的 check() 方法、发送详细的交易信息至交易信息表保存的 sendtx2table() 方法、透过 NFC 协议接收于 StoreTransactionManagement 类所创建的交易信息以及发送用户比特币地址的 nfc() 方法，最后是控制比特币钱包的 bitcoinpayment() 方法。CheckTx 类中的 getcheck() 方法可以向 Tx 类询问该笔交易是否已经得到认证，

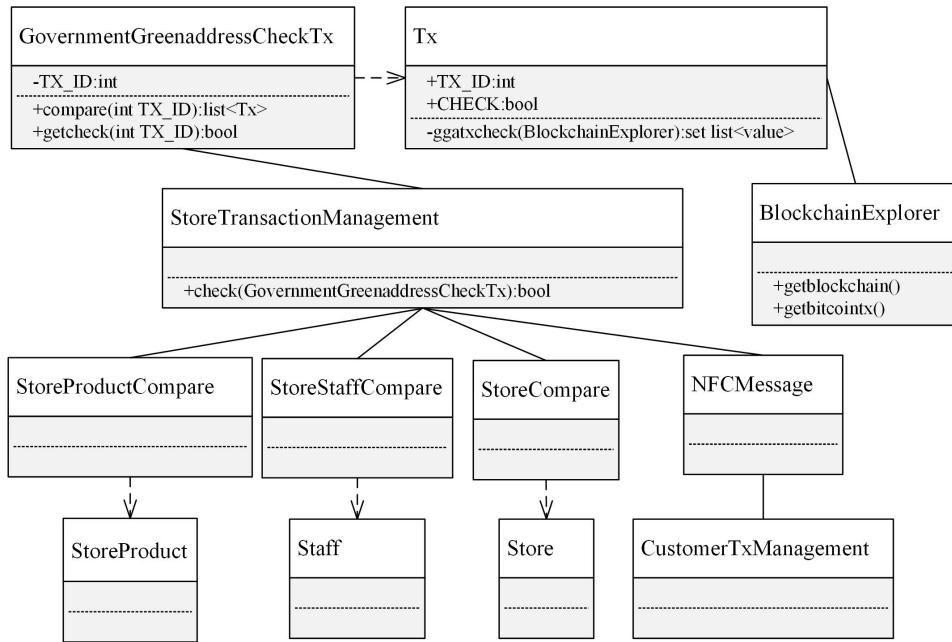


图 5.10 商家 Government Green Address 交易管理模块类图

`TxCompare` 类中的 `compare()` 方法提供用户向交易信息表查找与用户交易信息相关的交易，`TransferTx` 类中的 `sendtx()` 方法是将顾客完成交易的交易信息提交到交易信息表，`BitcoinPay` 类使用 `pay()` 方法支付比特币，`NFCMessage` 类中的 `receivenfcmessage()` 和 `sendnfcmessage()` 分别为透过 NFC 协议发送以及接收交易信息和顾客比特币地址。`StoreProductCompare` 类则是向 `StoreProduct` 类请求顾客手持移动装置中所有交易信息中相关的商家商品。

图5.12为顾客交易管理时序图，以下为流程说明：

1. 顾客启动比特币钱包管理模块，届时比特币钱包管理模块再同步最新的区块链信息至手持移动装置本地端，并将与手持移动装置本身存在的比特币地址进行核对。
2. 将所有在比特币钱包模块的地址提交到顾客交易管理模块。
3. 顾客交易管理模块将所有拿到的比特币地址发送到交易信息表，请求完整的交易信息内容。
4. 交易信息表将概要的交易信息传回顾客交易管理，此时的交易信息并非相当完整，只有商家产品编号。
5. 为了使得交易信息更加的完整可以显示更多的商家产品说明，便将手持移动装置本地端数据库存在的商家产品编号向商家产品信息表提出请求更详细的说明。
6. 商家产品信息表传回详细的商家产品信息到顾客交易管理模块。
7. 顾客交易管理模块将收到的信息显示在画面上。

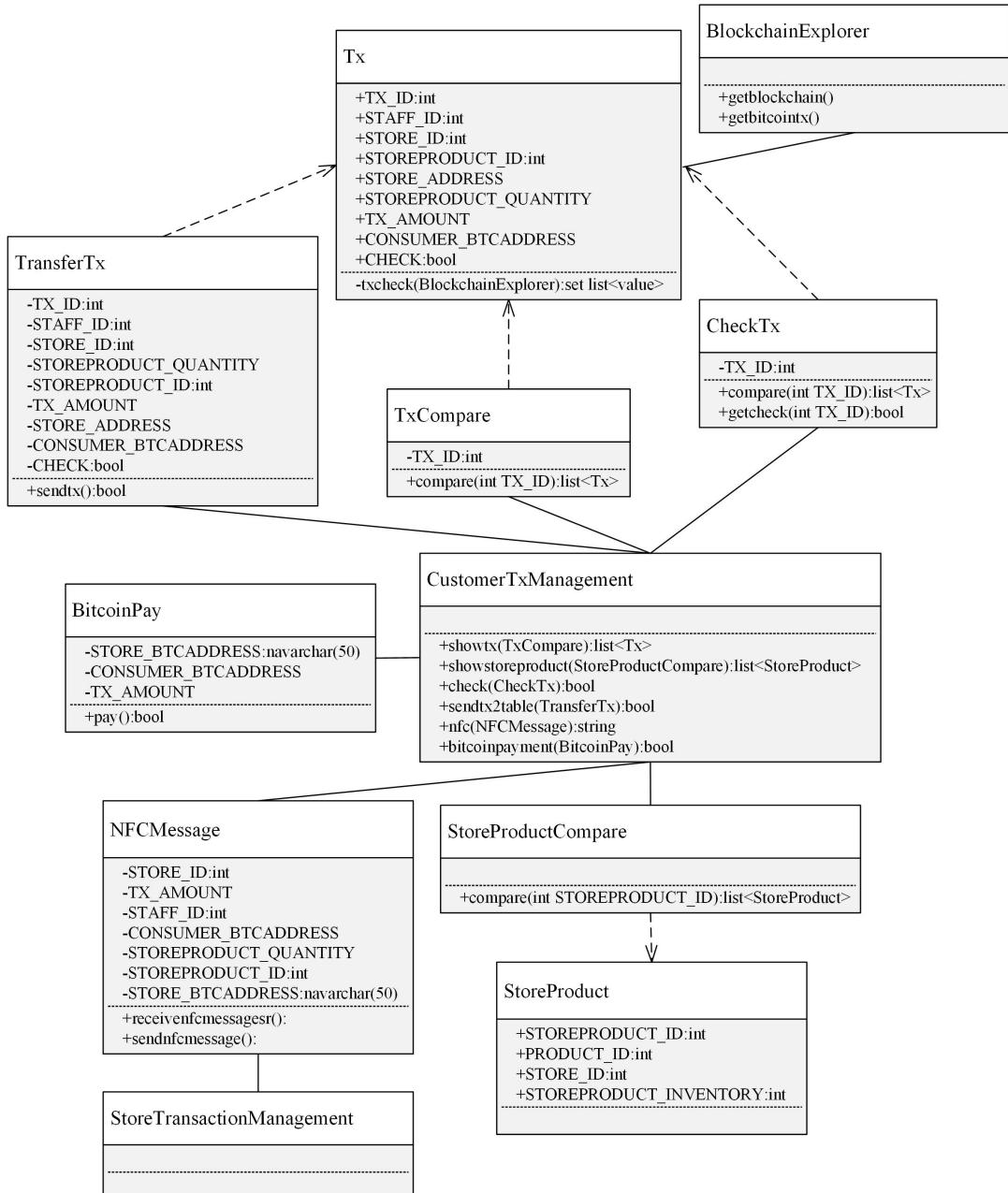


图 5.11 顾客交易管理模块类图

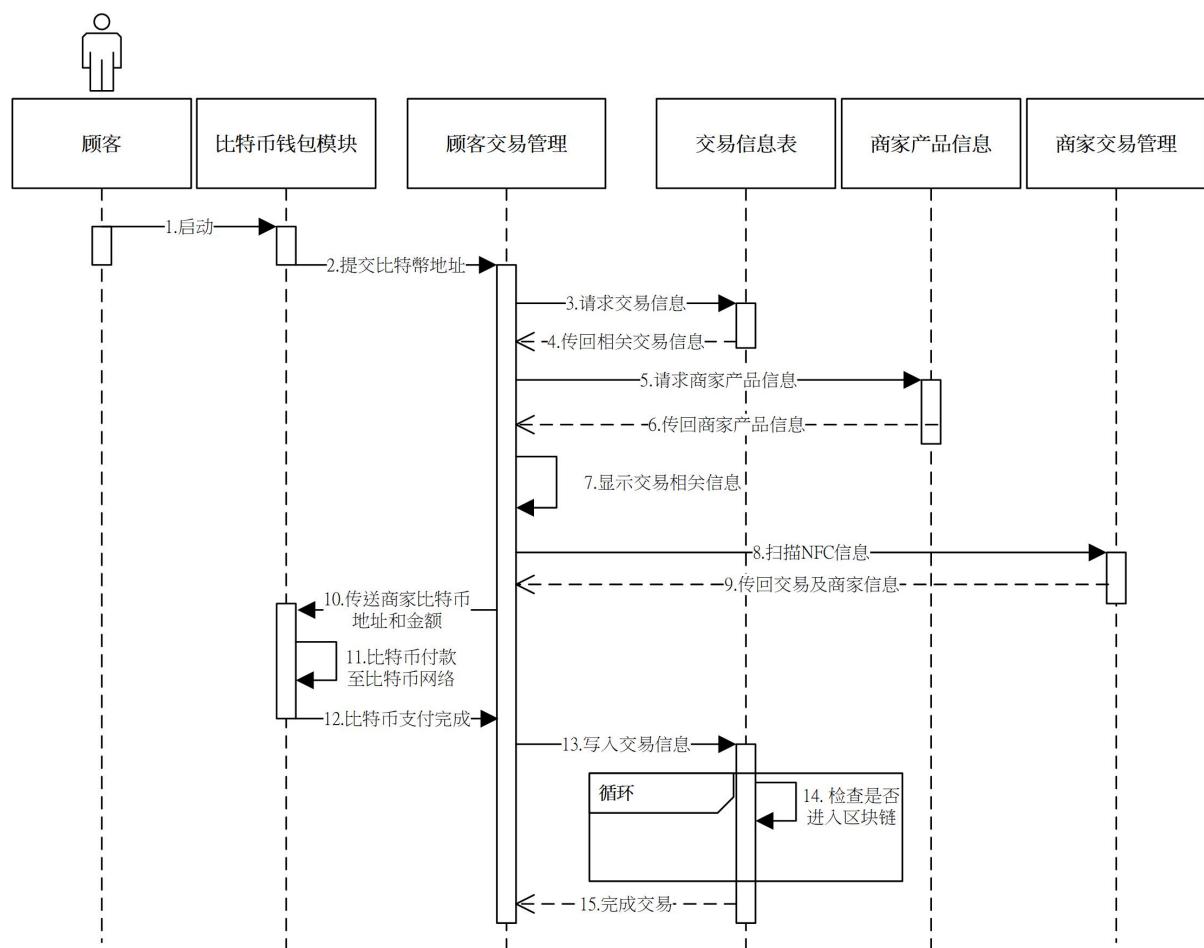


图 5.12 顾客交易管理时序图

8. 当顾客前往商家时，商家扫描完所有顾客欲购买的商家比特币地址、商家商品、数量、价格以及应付金额之后，便向商家交易管理启动 NFC 准备等待顾客的手持设备进行信息传输，此时的顾客将手持设备与商家手持设备靠近。
9. 商家管理模块透过 NFC 将交易清单信息发送到顾客交易管理模块。
10. 在商家交易管理模块重送的交易清单信息当中，包括商家的比特币地址信息与应付金额，此时顾客交易管理模块将比特币地址信息发送到比特币钱包模块。
11. 在比特币钱包模块当中使用 secp256k1 算法签署比特币交易信息，并将交易信息广播到比特币网络当中，此时该笔交易信息会进到比特币网络中的交易缓存等待矿工解出工作量证明的问题，将该笔交易信息存储到比特币区块链。
12. 在完成比特币签名后，会生成比特币的交易哈希值，比特币钱包管理模块将比特币交易哈希值发送到顾客交易管理模块。
13. 顾客交易管理模块请求交易信息表将比特币交易哈希值写入。
14. 完成写入后，交易信息表模块会调用比特币区块链检视器的相关函数，不断检查该笔比特币交易是否已经从未确认交易转变成已确认交易。倘若发现已经确认，则将交易信息表中 CHECK 字段的信息从"0" 修改为"1"。
15. 发现交易信息中的 CHECK 值为"1" 之后，便发送完成交易的信息到顾客交易管理模块。

图5.13为 BRTMS 中的顾客 Government Green Address 交易管理模块类图，以下将说明采用 Government Green Address 与未采用 Government Green Address 技术的顾客交易管理类图设计之间的差异。于 Tx 类中添加 ggatxcheck() 方法，该方法可以不断检查交易信息表中的交易是否来自 Government Green Address 地址，倘若是则不需要等待区块链的验证时间可以即刻认定该笔交易为有效，快速提交比特币交易速度。GovernmentGreenaddressCheck 类中 compare() 方法是为了查找与 TX\_ID 相符的交易信息，getchek() 方法则是向交易信息表中询问符合 TX\_ID 的交易信息是否已经被修改为"1"。其中 GovernmentGreenaddressBitcoinPay 类中的 multiplesignature() 方法可以实现多重签章算法，使得本系统可以支持即时交易。governmentgreedaddresspay() 方法是将多重签章算法广播报导比特币网络。

### 5.3 系统实现

为了验证和证明所提议的 BTMS 用于比特币支付收款监督的可行性和有效性，将其运行在用于商家商品管理和维护的 Java 应用程序的 SMIMSS 子系统，用于商家职工的运行在 Android App 上的 SMCTSS 以及运行在 App 上的用于顾客的 CMPTSS。如图5.14所示，SMIMSS 的 Java 应用程序可以帮助商家登入到系统或创建一个新帐户。授

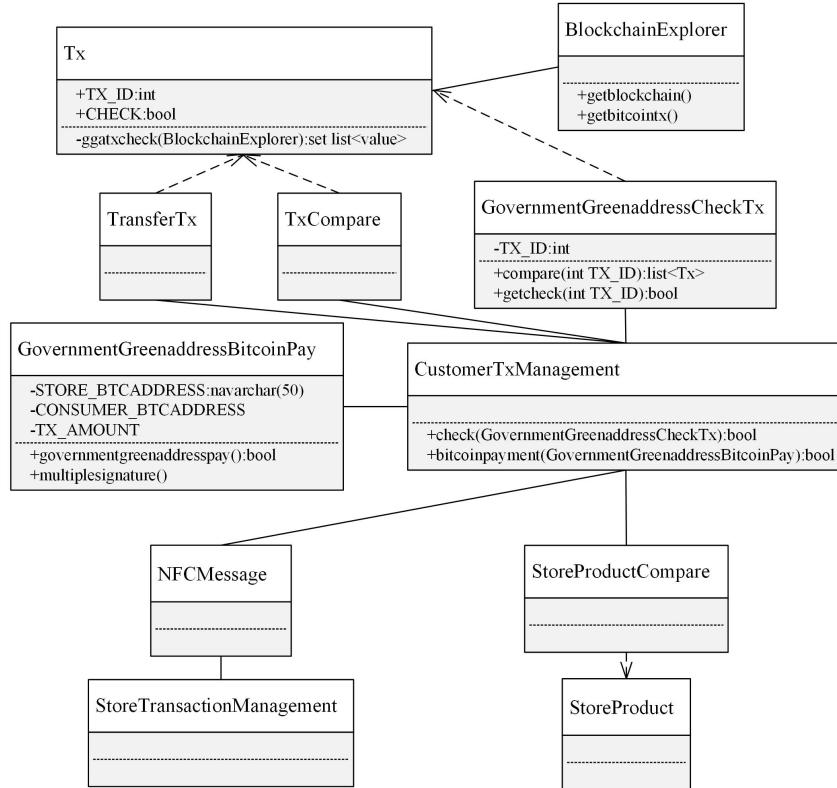


图 5.13 顾客 Government Green Address 交易管理模块类图

权商家成功登入系统后，商家可以插入或更新产品列表，如图5.15所示。实现的 SMIMSS Java 应用程序运行前面部分中所述的功能。

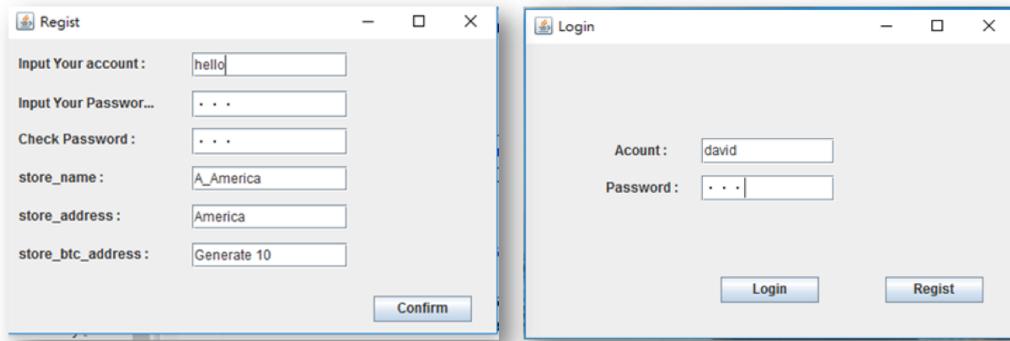


图 5.14 SMIMSS 的 Java 应用程序的注册和登入界面

商家的产品信息可以通过 RFID 标签扫描，存储到云端数据库中，商家职工可以使用实现的 SMCTSS Android 客户端，启用 NFC 监听器，从购物车中的顾客购买产品中读取 RFID 标签信息。在如图5.16所示的第一项活动中，商家职工必须登入才能获得授权访问 SMCTSS 功能。然后，在第二项活动中，SMCTSS 应用程序可以通过使用 SMIMSS 中应用的云数据库检查产品 RFID 标签信息并将其展示给顾客，从而将扫



图 5.15 在 SMIMSS 中插入或更新授权商家的产品目录

描的产品列入购物车。在图5.16的第三项活动中，顾客可以要求职工删除购买物品以，并确认最终交易清单。最后，SMCTSS 应用程序将自动使用比特币测试网络（Bitcoin Testnet）<sup>[60]</sup> 帮助职工确认发布此比特币交易的收款人地址，如图5.16所示。

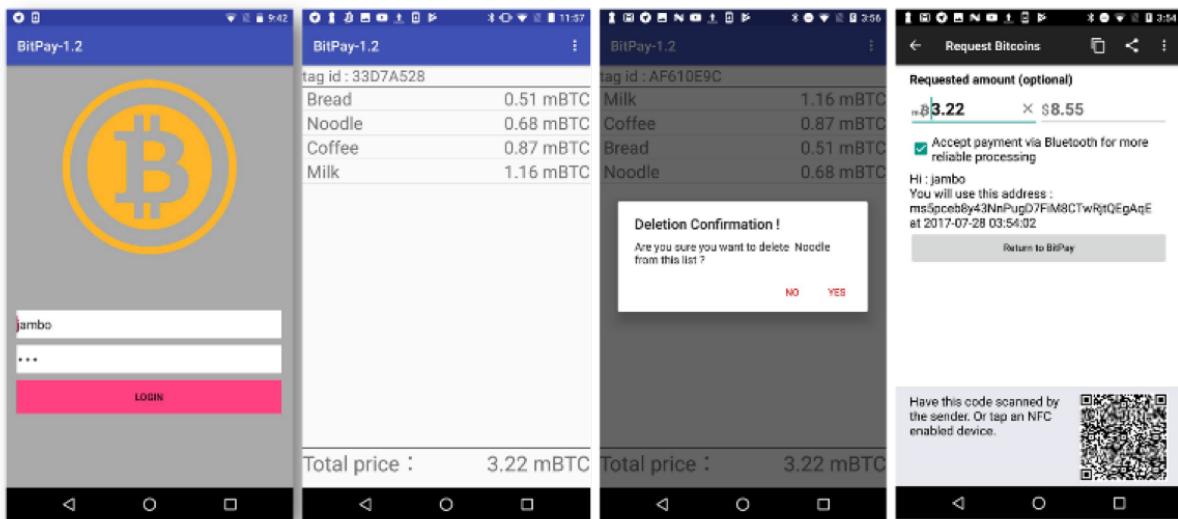


图 5.16 登入、等待结帐的商品、删除商品及支付确认

同时，顾客将使用与 SMCTSS App 相对应的 CMPTSS Android App 通过比特币完成采购产品交易。如图5.17所示，第一个活动表示顾客确认购买产品创建交易数据库的交易清单，第二个活动显示包括金额和付款人比特币地址在内的付款确认，第三个活动显示该钱包交易的历史记录，凡是出入该钱包的所有交易都会被显示出来，钱包可以同时作为买方和卖方，最后在第四项活动中显示了该笔交易详细采购产品的交易凭据。

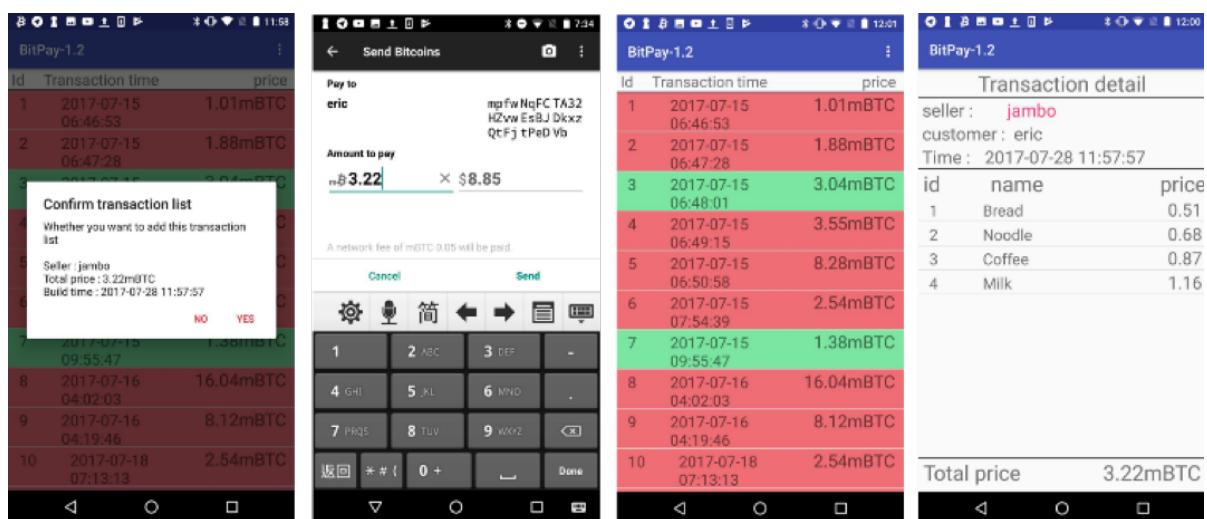


图 5.17 在 CMPTSS App 中，交易确认，付款确认、交易历史记录和交易凭证

## 第六章 系统测试

于本章将进行功能测试与性能测试，运行功能测试以确认本文所提出之系统的所有功能是否皆顺利运行，透过性能测试可知本系统的交易速度在优化前与优化后之间的差异，在本章中以手机作为手持移动装置的测试环境。

### 6.1 功能测试

本段主要是描述比特币的交易监督系统的测试计划。确认在系统集成前，必须先确认所有的设计组件均可正确的输出，在此着重于集成系统测试（Integration Test）及验收测试（Acceptance Test）。本章节内容将依据系统需求规格书与系统设计，描述关于集成测试的相关计划与内容。并希望透过此章节之描述与实践，达到顺利进行测试工作之目的。

1. 接受标准：本测试计划需要满足下列的测试接受准则：
  - (a) 本系统需要对所有列为必要（Critical、Important、Desirable）之需求作完整测试。
  - (b) 测试程序需要依照本测试计划所订定的程序进行，所有测试结果需要能符合预期测试结果方能接受。
  - (c) 以测试案例为单位，当测试未通过时，需要进行该单元的测试，其接受的准则与前一项规定相同。
2. 测试环境说明包括所采用的硬件与软件规格，分别如下：
  - (a) 硬件规格分为系统主机以及周边设备：
    - i. 系统主机：一台以上主机，每台主机 CPU 为 Intel Pentium 4 1.0 GHz 或以上，256 MB RAM 或以上，60 GB 以上硬盘空间。
    - ii. 周边设备：一台以上手持移动装置，与用来代表虚拟商品的数个 RFID 标签；已可供测试 NFC 用的手持移动装置包含小米 3 WCDMA 版，详细规格于表6.1，Google Nexus 5X，详细规格于表6.2。
  - (b) 软件规格：关于测试环境所需的软件规格說明，如下列所示：操作系统：Windows 10、Android 6.0.1/7.1.1。
3. 测试地点：在铭传大学桃园校区资工系实验室，透过 Android 手机进行的交易仿真实验，测试环境如图6.1的示意。
4. 测试时间：

表 6.1 小米 3 手机规格表

|      |  |
|------|--|
| 系统频率 | GSM 四频、 WCDMA  |
| 操作系统 | Android 4.3  |
| 处理器  | Qualcomm Snapdragon 800 2.3 GHz 四核心                  |
| 内存   | 2 GB RAM 、 16 GB ROM                                 |
| 记忆卡  | 不支持  |
| 显示屏幕 | 5 吋 1670 万色 IPS (1920 x 1080 pixels) 、 441 ppi       |
| 相机   | 1300 万像素后置镜头 ( $f/2.2$ 、 28 mm) 、 200 万像素前置镜头、 1080p |
| 电池   | 3050 mAh (不可换)                                       |
| 尺寸   | 144 x 73.6 x 8.1 mm                                  |
| 重量   | 145 g  |

表 6.2 Google Nexus 5X 手机规格表

|      |  |
|------|--|
| 系统频率 | GSM 四频、 WCDMA  |
| 操作系统 | Android 6.0  |
| 处理器  | Qualcomm Snapdragon 800 1.8 GHz 六核                   |
| 内存   | 2 GB RAM 、 16 GB ROM                                 |
| 记忆卡  | 不支持  |
| 显示屏幕 | 5 吋 1670 万色 IPS (1920 x 1080 pixels) 、 441 ppi       |
| 相机   | 1300 万像素后置镜头 ( $f/2.2$ 、 28 mm) 、 200 万像素前置镜头、 1080p |
| 电池   | 2700 mAh (不可换)                                       |
| 尺寸   | 147 x 72.6 x 7.9 mm                                  |
| 重量   | 136 g  |

- (a) 各子系统之内部组件集成测试 (Module Test) (2017 年 2 月 25 日到 2017 年 6 月 8 日)
- (b) 比特币的交易监督系统集成测试 (Integration Test) (2017 年 6 月 8 日到 2017 年 6 月 21 日)
- (c) 比特币的交易监督系统接受度测试 (Acceptance Test) (2017 年 7 月 10 日到 2017 年 7 月 21 日)

5. 查核点:

- (a) 各子系统之内部组件集成测试 (2017 年 5 月 10 日)
- (b) 比特币的交易监督系统集成测试 (2017 年 7 月 1 日)
- (c) 比特币的交易监督系统接受度测试 (2017 年 7 月 1 日)

6. 集成测试规划 (Integration Testing):

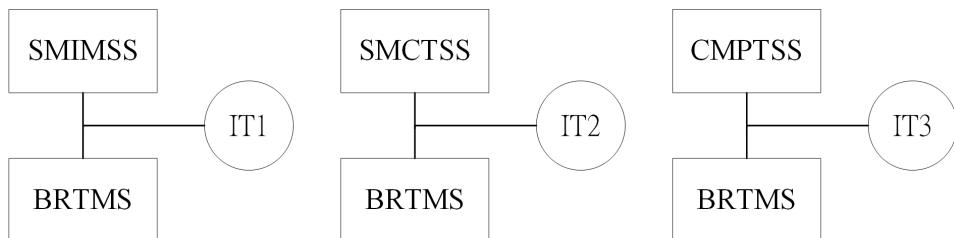


图 6.1 集成子系统测试

7. 验收测试规划 (Acceptance Testing, AT): 本系统须达成以下三组接受用例陈列的所有功能，测试本论文设计与搭建的系统功能是否能够顺利运行。测试的角色有两个，分别为管理员以及用户，如图6.2为 BTMS 用例示意图，预计测试服务器的组态设置、手机的组态设置以及数据库的组态设置：

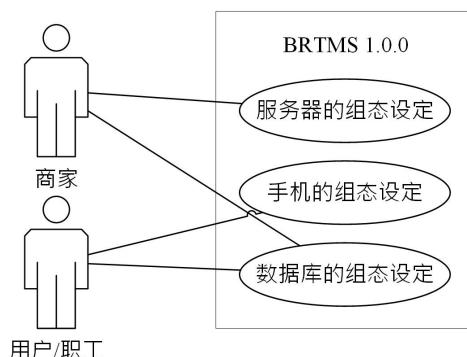


图 6.2 BTMS 用户用例图

图6.3为三组验收测试的示意图，对本比特币的交易监督系统进行测试。

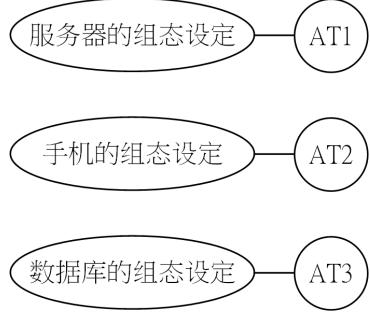


图 6.3 验收测试 (Acceptance Testing)

### 6.1.1 测试用例

#### 1. 集成测试 (Integration Test) :

- (a) IT1 测试用例：表6.3为 IT1 测试用例，测试对象为商家和商品信息管理子系统。目的：验证 [SMIMSS 1.1.0] 子系统能否正确管理商品信息。

表 6.3 IT1 测试用例

|         |   |
|---------|---|
| 用例 ID   | IT1   |
| 用例名称    | 集成 SMIMSS 至 BTMS  |
| 测试目标    | [SMIMSS 1.1.0] 、 [BTMS 1.0.0]   |
| 依赖关系    | SMIMSS-F-001~ SMIMSS-F-005  |
| 严重程度    | 1 (Critical)  |
| 用例描述    | 1. 能够添加店家帐户<br>2. 能够添加/修改/删除店员帐户<br>3. 能够添加/删除/修改商品信息<br>4. 能够取得产品信息<br>5. 能够接收交易信息 |
| 预期结果    | 1. 成功添加店家帐户<br>2. 成功添加/修改/删除店员帐户<br>3. 成功添加/修改/删除商品信息<br>4. 成功取得产品信息<br>5. 成功接收交易信息 |
| Cleanup | 无   |

- (b) IT2 测试用例：表6.4为 IT2 测试用例目标，测试对象为商家手持移动装置收款及交易子系统。目的：验证 [SMCTSS 1.2.0] 子系统是否能够完成一笔行动支付之交易。

- (c) IT3 测试用例：表6.5为 IT3 测试用例，目标测试对象为客户端行动支付和交易子系统。目的：验证 [CMPTSS 1.3.0] 能正确接收 SMCTSS 所发送的交易数据，并以其交易信息运行以比特币付款之动作。可以查找商品信息，且能够存储并且查看用户过往之交易纪录。

表 6.4 IT2 测试用例

|         |  |
|---------|--|
| 用例 ID   | IT2  |
| 用例名称    | 集成 SMCTSS 至 BTMS   |
| 测试目标    | [SMCTSS 1.2.0] 、 [BTMS 1.0.0]  |
| 依赖关系    | SMCTSS-F-001~ SMCTSS-F-007   |
| 严重程度    | 1 (Critical)   |
| 用例描述    | 1. 能够登入店员帐户<br>2. 能够扫描 NFC 标签<br>3. 能够读取商品信息<br>4. 能够创建交易清单<br>5. 能够发送交易信息<br>6. 能够认证交易信息<br>7. 能够存储交易明细 |
| 预期结果    | 1. 成功登入店员帐户<br>2. 成功扫描 NFC 标签<br>3. 成功读取商品信息<br>4. 成功创建交易清单<br>5. 成功发送交易信息<br>6. 成功认证交易信息<br>7. 成功存储交易明细 |
| Cleanup | 无  |

表 6.5 IT3 测试用例

|         |   |
|---------|---|
| 用例 ID   | IT3   |
| 用例名称    | 集成 CMPTSS 至 BTMS  |
| 测试目标    | [CMPTSS 1.3.0] 、 [BTMS 1.0.0]   |
| 依赖关系    | CMPTSS-F-001~ CMPTSS-F-007  |
| 严重程度    | 1 (Critical)  |
| 用例描述    | 1. 能够登入顾客帐号<br>2. 能够读取商品信息<br>3. 能够接收交易清单<br>4. 能够认证交易信息<br>5. 能够运行行动支付<br>6. 能够存储交易明细<br>7. 能够查看交易纪录 |
| 预期结果    | 1. 成功登入顾客帐号<br>2. 成功读取商品信息<br>3. 成功接收交易清单<br>4. 成功认证交易信息<br>5. 成功运行行动支付<br>6. 成功存储交易纪录<br>7. 成功查看交易纪录 |
| Cleanup | 无   |

2. 验收测试用例 (Acceptance Testing Cases) : 验收测试用例目的在于测试母系统 BTMS、子系统 SMIMSS、SMCTSS 与 CMPTSS 是否能够顺利的进行信息传递完成交互。

(a) AT1 测试用例: 表6.6所示, 目标测试管理人员是否能够顺利使用子系统 SMIMSS 顺利与母系统 BTMS 交互。目的: 验证使用用例 (Use case) 1, 透过组态文件的修改对服务器进行组态设置。

表 6.6 AT1 测试用例

|         |                                |                          |
|---------|--------------------------------|--------------------------|
| 用例 ID   | AT1                            |                          |
| 用例名称    | 服务器的组态设置                       |                          |
| 测试目标    | [SMIMSS 1.1.0]<br>[BTMS 1.0.0] |                          |
| 依赖关系    | BTMS-F-001                     |                          |
| 严重程度    | 1 (Critical)                   |                          |
| 用例描述    | 用户操作                           | 系统响应                     |
|         | 1. 管理人员依照环境设置服务器组态。            |                          |
|         |                                | 2. 服务器依照管理人员所做的组态设置启动服务。 |
| 预期结果    | 成功启动服务器的相关服务。                  |                          |
| Cleanup | 无                              |                          |

(b) AT2 测试用例: 如表6.7所示, 参与者为用户。目的: 验证用例 (Use case) 2 透过组态文件的修改对手机进行组态设置。

表 6.7 AT2 测试用例

|         |                                  |                         |
|---------|----------------------------------|-------------------------|
| 用例 ID   | AT2                              |                         |
| 用例名称    | 手机的组态设置                          |                         |
| 测试目标    | [SMCTSS 1.2.0]<br>[CMPTSS 1.3.0] |                         |
| 依赖关系    | BTMS-F-002~ BTMS-F-003           |                         |
| 严重程度    | 1 (Critical)                     |                         |
| 用例描述    | 用户操作                             | 系统响应                    |
|         | 1. 用户修改手机组态设置参数。                 |                         |
|         |                                  | 2. 手机依照用户在设置档中所填入的数值运作。 |
| 预期结果    | 成功完成手机的组态设置                      |                         |
| Cleanup | 无                                |                         |

- (c) AT3 测试用例：如表6.8所示，目的：验证用例（Use case）3，透过组态文件的修改对数据库进行组态设置。

表 6.8 AT3 测试用例

|         |  |                          |
|---------|--|--------------------------|
| 用例 ID   | AT3  |                          |
| 用例名称    | 数据库的组态设置   |                          |
| 测试目标    | [SMIMSS 1.1.0]<br>[SMCTSS 1.2.0]<br>[CMPTSS 1.3.0] |                          |
| 依赖关系    | BTMS-F-001~ BTMS-F-003                             |                          |
| 严重程度    | 1 (Critical)                                       |                          |
| 用例描述    | 用户操作   | 系统响应                     |
|         | 1. 管理者设置数据库组态。                                     |                          |
|         |  | 2. 数据库依照管理人员所做的组态设置启动服务。 |
|         | 3. 用户修改数据库之数据及文件。                                  |                          |
|         |  | 4. 数据库依照用户所做的组态设置启动服务。   |
| 预期结果    | 成功设置完成数据库的相关设置。                                    |                          |
| Cleanup | 无  |                          |

## 6.1.2 测试结果和分析

1. 集成测试用例（Integration Testing Cases）表6.9为 IT1、为 IT2、为 IT3 的集成子系统测试结果，皆顺利运作。
2. 验收测试用例（Acceptance Testing Cases）表6.10为前节所设计的三种 AT1、AT2 与 AT3 的验收测试结果，皆顺利运行。
3. 可追踪性（Traceability）
  - (a) 子系统与测试用例：表6.11为测试组 IT1、IT2、IT3、AT1、AT2、AT3 与子系统 SMIMSS、SMCTSS、CMPISS 的关系表。
  - (b) 需求与测试用例：表6.12为本系统需求与集成测试用例的关系表，表6.13为需求与验收测试案例的关系表。

表 6.9 集成子系统测试结果

| 测试用例 | 结果 (通过/不通过) | Comment  |
|------|-------------|--|
| IT1  | 通过          | 1. 成功添加店家帐户<br>2. 成功添加/修改/删除店员帐户<br>3. 成功添加/修改/删除商品信息<br>4. 成功取得产品信息<br>5. 成功接收交易信息                      |
| IT2  | 通过          | 1. 成功登入店员帐户<br>2. 成功扫描 NFC 标签<br>3. 成功读取商品信息<br>4. 成功创建交易清单<br>5. 成功发送交易信息<br>6. 成功认证交易信息<br>7. 成功存储交易明细 |
| IT3  | 通过          | 1. 成功登入顾客帐号<br>2. 成功读取商品信息<br>3. 成功接收交易清单<br>4. 成功认证交易信息<br>5. 成功运行行动支付<br>6. 成功存储交易纪录<br>7. 成功查看交易纪录    |
| RATE | 90%         | BTMS 开发透过手机让商家及顾客以手机发送交易信息，如：商品名称、商品金额，商家收款地址。并且及时将商品信息更新至服务器之数据库，以便商家控管商品信息状态，同时让顾客可以享受数字加密货币的方便性。      |

表 6.10 验收测试结果

| 测试用例 | 结果 (通过/不通过) | Comment                       |
|------|-------------|-------------------------------|
| AT1  | 通过          | 成功启动服务器的相关服务。                 |
| AT2  | 通过          | 成功完成手机的组态设置。                  |
| AT3  | 通过          | 成功设置完成数据库的相关设置。               |
| RATE | 100%        | BTMS 可透过组态设置的方式来设定各个子系统的环境参数。 |

表 6.11 子系统与测试用例关系表

|     | SMIMSS 1.1.0 | SMCTSS 1.2.0 | CMPISS 1.3.0 |
|-----|--------------|--------------|--------------|
| IT1 | X            |              |              |
| IT2 |              | X            |              |
| IT3 |              |              | X            |
| AT1 | X            |              |              |
| AT2 |              | X            | X            |
| AT3 | X            | X            | X            |

表 6.12 需求与集成测试用例的关系表

|              | IT1 | IT2 | IT3 |
|--------------|-----|-----|-----|
| BTMS-F-001   | X   |     |     |
| BTMS-F-002   |     | X   |     |
| BTMS-F-003   |     |     | X   |
| SMIMSS-F-001 | X   |     |     |
| SMIMSS-F-002 | X   |     |     |
| SMIMSS-F-003 | X   |     |     |
| SMIMSS-F-004 | X   |     |     |
| SMIMSS-F-005 | X   |     |     |
| SMCTSS-F-001 |     | X   |     |
| SMCTSS-F-002 |     | X   |     |
| SMCTSS-F-003 |     | X   |     |
| SMCTSS-F-004 |     | X   |     |
| SMCTSS-F-005 |     | X   |     |
| SMCTSS-F-006 |     | X   |     |
| SMCTSS-F-007 |     | X   |     |
| CMPTSS-F-001 |     |     | X   |
| CMPTSS-F-002 |     |     | X   |
| CMPTSS-F-003 |     |     | X   |
| CMPTSS-F-004 |     |     | X   |
| CMPTSS-F-005 |     |     | X   |
| CMPTSS-F-006 |     |     | X   |
| CMPTSS-F-007 |     |     | X   |

表 6.13 需求与验收测试案例的关系表

|            | AT1 | AT2 | AT3 |
|------------|-----|-----|-----|
| BTMS-F-001 | X   |     | X   |
| BTMS-F-002 |     | X   | X   |
| BTMS-F-003 |     | X   | X   |

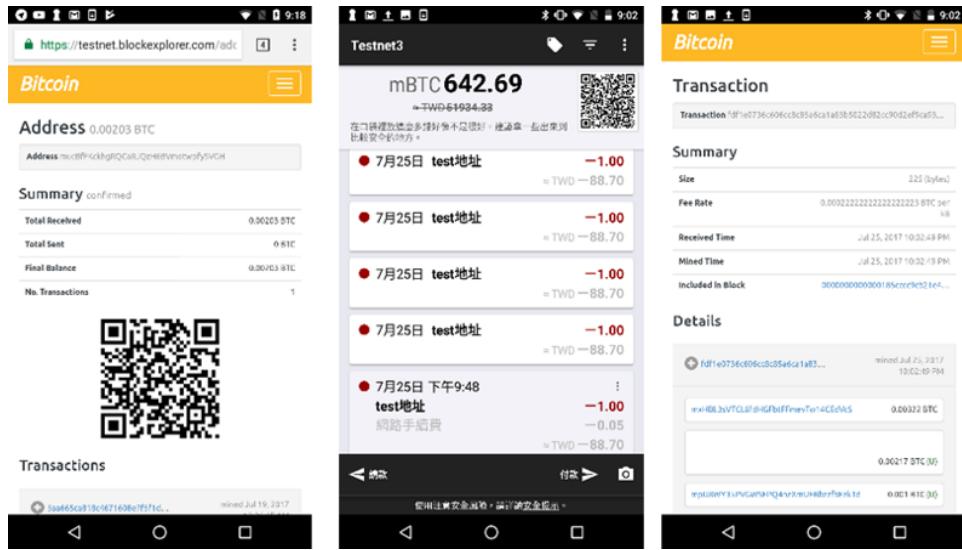


图 6.4 使用区块链检视器验证存储在比特币区块链中的交易过程

## 6.2 性能测试

根据比特币点对点架构，尽管顾客和商家之间的交易细节已经快速存储到云数据库，但官方确认交易与当前比特币区块链的交易通常需要更长的时间，因为需要确保确认的数量在交易广播比特币点对点网络并存储到缓存池后，是否存在双重支付。本文设计了比特币的交易监督系统以及引用多重签章算法重新设计的比特币的实时交易监督系统，以下将分别针对有无采用多重签章算法的性能测试。

### 6.2.1 系统性能测试

为了验证本文提出的 BTMS 不会通过使用比特币等加密货币影响交易完成时间，于 2017 年 7 月 25 日、2017 年 9 月 6 日以及 2018 年 3 月 21 日，在 Testnet 实验中连续记录了 20 笔交易信息，每 1 秒发出一笔交易，分别历时 20 分钟。

1. 初始测试（2017 年 7 月 25 日）：首先，使用区块链检视器（Blockchain Explorer）<sup>[59]</sup>，如 6.4 的快照所示，透过使用 Testnet 依序进行 20 笔比特币交易，如图 6.4 的中间快照所示，最后 20 笔交易完成时间全部记录在区块链检视器。实验结果显示，实验中的所有交易都在 3 秒钟左右（平均 2.97 秒，标准差小于 1 秒）发送到比特币网络缓存池，平均交易完成时间在比特币区块链中确认为 522.33 秒（小于 9 分钟），标准差大约为 339 秒。
2. 第一次测试（2017 年 9 月 6 日）：表 6.14 为第一次实验数据，该次的比特币交易发起广播至比特币交易缓存池的时间平均为 1.918 秒，标准差为 0.55586 秒。但为了预防双重支付攻击需要等待平均 654.8 秒（小于 11 分钟，标准差 346.63 秒）的时间。

表 6.14 第一次 Testnet 运行实验之数据分析（2017 年 9 月 6 日）

|          | 进入缓存池时间 (秒) | 进入区块链时间 (秒)     | 完成交易时间 (秒)      |
|----------|-------------|-----------------|-----------------|
| 平均       | 1.918       | 654.8           | 654.8           |
| 样本标准差    | 0.55586     | 346.63          | 346.63          |
| 95% 信赖区间 | 1.69 ~ 2.15 | 511.72 ~ 797.88 | 511.72 ~ 797.88 |
| 99% 信赖区间 | 1.61 ~ 2.23 | 460.9 ~ 848.7   | 460.9 ~ 848.7   |

3. 第二次测试（2018 年 3 月 21 日）：表 6.15 为第二次测试的实验原始数据，表 6.16 为将数据进行分析后的结果可以看出，平均进入缓存池的时间为 1.12 秒，进入区块链的时间为 287.12 秒，因为并未采用多重签章算法，所以交易完成时间以 287.12 秒计算。

表 6.15 第二次 Testnet 运行实验数据（2018 年 3 月 21 日）

| 交易次数 | 付款时间    | 进入缓存池所花时间 (秒) | 进入缓存池时间点 | 写入区块链所花时间 (秒) | 写入区块链时间点 |
|------|---------|---------------|----------|---------------|----------|
| 1    | 0:00:00 | 3             | 0:00:03  | 136           | 0:02:16  |
| 2    | 0:01:00 | 1             | 0:01:01  | 76            | 0:02:16  |
| 3    | 0:02:04 | 3             | 0:02:07  | 12            | 0:02:16  |
| 4    | 0:03:00 | 1             | 0:03:01  | 438           | 0:10:18  |
| 5    | 0:04:00 | 1             | 0:04:01  | 378           | 0:10:18  |
| 6    | 0:05:00 | 1             | 0:05:01  | 318           | 0:10:18  |
| 7    | 0:06:00 | 2             | 0:06:02  | 258           | 0:10:18  |
| 8    | 0:07:00 | 1             | 0:07:01  | 198           | 0:10:18  |
| 9    | 0:08:00 | 1             | 0:08:01  | 138           | 0:10:18  |
| 10   | 0:09:00 | 1             | 0:09:01  | 78            | 0:10:18  |
| 11   | 0:10:00 | 1             | 0:10:01  | 18            | 0:10:18  |
| 12   | 0:11:00 | 2             | 0:11:02  | 810           | 0:24:30  |
| 13   | 0:12:00 | 1             | 0:12:01  | 750           | 0:24:30  |
| 14   | 0:13:00 | 1             | 0:13:01  | 690           | 0:24:30  |
| 15   | 0:14:00 | 1             | 0:14:01  | 630           | 0:24:30  |
| 16   | 0:15:00 | 1             | 0:15:01  | 570           | 0:24:30  |
| 17   | 0:16:00 | 2             | 0:16:02  | 510           | 0:24:30  |
| 18   | 0:17:00 | 2             | 0:17:02  | 450           | 0:24:30  |
| 19   | 0:18:00 | 1             | 0:18:01  | 390           | 0:24:30  |
| 20   | 0:19:00 | 1             | 0:19:01  | 330           | 0:24:30  |

在未引用多重签章算法的监督系统中，第一次与第二次的测试相比并无太大的差异，交易完成时间与比特币区块生成所需时间相同，平均时间皆为十分钟。根据比特币 Testnet 上的初步实验结果显示，本文提出的 BTMS 可以快速有效地运行比特币的交易监督系统。

表 6.16 第二次以 Testnet 运行实验数据分析（2018 年 3 月 21 日）

|          | 进入缓存池时间 (秒) | 进入区块链时间 (秒)     | 完成交易时间 (秒)      |
|----------|-------------|-----------------|-----------------|
| 平均       | 1.12        | 287.12          | 287.12          |
| 样本标准差    | 0.68        | 246.59          | 246.59          |
| 95% 信赖区间 | 0.84 ~ 1.4  | 185.33 ~ 388.91 | 185.33 ~ 388.91 |
| 99% 信赖区间 | 0.74 ~ 1.5  | 149.18 ~ 425.06 | 149.18 ~ 425.06 |

## 6.2.2 实时系统性能测试

本节介绍比特币测试币于 Government Green Address 钱包进行交易的性能实验与结果分析，包含该实验的目的、方法及结果分析。实验的目的是要确认的系统在商家端进行行动支付时能快速、精准且高效率的进行交易，并了解使用一般 Testnet 钱包与 Government Green Address 钱包作为交易媒介的确认交易时间的差距。表6.17为 2017 年 7 月 25 日初步的采用多重签章算法的测试数据，数据显示交易存储到区块链的平均时间为 10 分钟，但 Government Green Address 采用的多重签章算法，可以有效拒绝双重花费的交易发起，因此可将交易的有效时间，从交易进入区块链的时间，重新订定为交易进入比特币交易缓存池的时间。

表 6.17 初步的 Government Green Address 实验测试数据（2017 年 7 月 25 日）

| 付款时间     | 进入缓存池<br>所花时间 (秒) | 进入缓存池<br>时间点 | 入块时间     | 写入区块<br>费时间 |
|----------|-------------------|--------------|----------|-------------|
| 06:36:25 | 2.14              | 06:36:27.14  | 06:52:00 | 16:25       |
| 06:38:25 | 1.075             | 06:38:26.075 | 06:52:00 | 14:25       |
| 06:40:25 | 1.722             | 06:40:26.722 | 06:52:00 | 12:25       |
| 06:42:25 | 2.953             | 06:42:27.953 | 06:52:00 | 10:25       |
| 06:44:25 | 1.511             | 06:44:26.511 | 06:52:00 | 08:25       |
| 06:46:25 | 2.269             | 06:46:27.269 | 06:52:00 | 06:25       |
| 06:48:25 | 1.831             | 06:48:26.831 | 06:52:00 | 04:25       |
| 06:50:25 | 1.227             | 06:50:26.227 | 06:52:00 | 02:25       |
| 06:52:25 | 2.026             | 06:52:27.026 | 07:12:01 | 20:24       |
| 06:54:25 | 1.257             | 06:54:26.257 | 07:12:01 | 18:24       |
| 06:56:25 | 1.511             | 06:56:26.511 | 07:12:01 | 16:24       |
| 06:58:25 | 2.815             | 06:58:27.815 | 07:12:01 | 14:24       |
| 07:00:26 | 1.544             | 07:00:27.544 | 07:12:01 | 12:25       |
| 07:02:25 | 1.767             | 07:02:26.767 | 07:12:01 | 10:24       |
| 07:04:25 | 1.52              | 07:04:26.52  | 07:12:01 | 08:24       |
| 07:06:25 | 1.953             | 07:06:26.953 | 07:12:01 | 06:24       |

本次的实验分为两部份，分别是透过比特币 Testnet 钱包以及使用本论文所采用的 Government Green Address 比特币钱包上运行 20 次付款，皆以相同地址收款，交易金额都设置为 0.00001 BTC，实验时间为 2017 年 9 月 6 日与 2018 年 2 月 6 日，每隔一分

钟运行一次付款的动作，总共历时 20 分钟。两款钱包同时发起交易，并透过区块链检视器进行记录时间，最后再比较使用一般比特币钱包及 Government Green Address 钱包两者之间的差距。

- 第一次测试（2017 年 9 月 6 日）：表6.18为第一次测试实验的数据分析结果，平均进入交易缓存池的时间为 2.11 秒，真正写入区块链的时间为 654.24 秒，采用多重签章算法，这些交易不存在双重支付的问题，因此完成时间记为 2.11 秒。

表 6.18 第一次以 Government Green Address 运行实验之数据分析（2017 年 9 月 6 日）

|          | 进入缓存池时间（秒）  | 进入区块链时间（秒）      | 完成交易时间（秒）   |
|----------|-------------|-----------------|-------------|
| 平均       | 2.11        | 654.24          | 2.11        |
| 样本标准差    | 0.65        | 346.9           | 0.65        |
| 95% 信赖区间 | 1.84 ~ 2.38 | 511.05 ~ 797.43 | 1.84 ~ 2.38 |
| 99% 信赖区间 | 1.75 ~ 2.47 | 460.19 ~ 848.29 | 1.75 ~ 2.47 |

- 第二次测试（2018 年 3 月 21 日）：表6.19为第二次以 Government Green Address 运行实验数据，表6.20为基于第二次测试的数据分析结果，进入缓存池的平均时间为 1.55 秒，进入区块链的平均时间为 431.4 秒，因为采用多重签章算法将完成交易时间以 1.55 秒计算。

本次实验分别记录以 Testnet 钱包及 Government Green Address 钱包运行 20 次交易的进入缓存池等待时间和写入区块等待时间。若以 Testnet 钱包交易，必须等到交易写入才能保证此笔交易不会被矿工遗弃，也算真的完成这笔交易；但若以 Government Green Address 钱包发起交易就大不相同，当交易进入缓存池，即使遇到交易被矿工遗弃的情况，Government Green Address 机构节点也会重新发起此笔交易，保证让交易写入区块，所以只要进入缓存池就可以视为交易完成，透过两者钱包的交易数据，本文分析两种钱包交易的时间数据。

透过本次的实验可以发现虽然以两种钱包交易进入区块的等待时间完全相同，但因为 Government Green Address 钱包的特性，只要进入缓存池就算完成交易确认，因此 Government Green Address 钱包的完成交易确认的时间远远快于一般 Testnet。相信以此方式作为主要支付管道，可以省去顾客在现金支付时掏零钱、算钱及找零等繁琐的动作及时间，以此达成提升日常生活中的便利性与安全性。

表 6.19 第二次以 Government Green Address 运行实验数据（2018 年 3 月 21 日）

| 交易次数 | 付款时间    | 进入缓存池所花时间 (秒) | 进入缓存池时间点 | 写入区块链所花时间 (秒) | 写入区块链时间点 |
|------|---------|---------------|----------|---------------|----------|
| 1    | 0:00:01 | 2             | 0:00:03  | 619           | 0:10:18  |
| 2    | 0:01:00 | 1             | 0:01:01  | 559           | 0:10:18  |
| 3    | 0:02:02 | 2             | 0:02:04  | 496           | 0:10:18  |
| 4    | 0:03:00 | 1             | 0:03:01  | 438           | 0:10:18  |
| 5    | 0:04:00 | 1             | 0:04:01  | 378           | 0:10:18  |
| 6    | 0:05:00 | 2             | 0:05:02  | 318           | 0:10:18  |
| 7    | 0:06:00 | 1             | 0:06:01  | 258           | 0:10:18  |
| 8    | 0:07:00 | 1             | 0:07:01  | 198           | 0:10:18  |
| 9    | 0:08:00 | 2             | 0:08:02  | 138           | 0:10:18  |
| 10   | 0:09:00 | 2             | 0:09:02  | 78            | 0:10:18  |
| 11   | 0:10:00 | 1             | 0:10:01  | 18            | 0:10:18  |
| 12   | 0:11:00 | 2             | 0:11:02  | 810           | 0:24:30  |
| 13   | 0:12:00 | 2             | 0:12:02  | 750           | 0:24:30  |
| 14   | 0:13:00 | 1             | 0:13:01  | 690           | 0:24:30  |
| 15   | 0:14:00 | 2             | 0:14:02  | 630           | 0:24:30  |
| 16   | 0:15:00 | 1             | 0:15:01  | 570           | 0:24:30  |
| 17   | 0:16:00 | 2             | 0:16:02  | 510           | 0:24:30  |
| 18   | 0:17:00 | 2             | 0:17:02  | 450           | 0:24:30  |
| 19   | 0:18:00 | 1             | 0:18:01  | 390           | 0:24:30  |
| 20   | 0:19:00 | 2             | 0:19:02  | 330           | 0:24:30  |

表 6.20 第二次以 Government Green Address 运行实验之数据分析（2018 年 3 月 21 日）

|          | 进入缓存池时间 (秒) | 进入区块链时间 (秒)   | 完成交易时间 (秒) |
|----------|-------------|---------------|------------|
| 平均       | 1.55        | 431.4         | 1.55       |
| 样本标准差    | 0.51        | 220.84        | 0.51       |
| 95% 信赖区间 | 1.34~1.76   | 340.24~522.56 | 1.34~1.76  |
| 99% 信赖区间 | 1.26~1.84   | 307.86~554.94 | 1.26~1.84  |

表 6.21 原始系统与实时系统比较表

|                     | 完成交易时间 (秒) |
|---------------------|------------|
| 原始系统 (BTMS) 交易平均时间  | 287.12     |
| 实时系统 (BRTMS) 交易平均时间 | 1.55       |

## 第七章 总结与展望

设计并实现一个名为 BRTMS (Bitcoin Realtime Transaction Monitoring System) 的比特币的实时交易监督系统，其工作包括背景调研比特币技术上的优势与劣势、匿名交易衍生出无法保障顾客权益、政府无法课征税收及存在着透过比特币洗钱的问题。为解决上述问题，在技术调研方面针对比特币地址的相关算法、地址的生成过程、多重签章算法以及区块链技术进行剖析。设计系统之前得进行详细的需求分析以及交易模型分析。设计 BRTMS 时，加入 Government Green Address 使该系统成为实时监督系统，并实现 BTMS 与 BRTMS 的客户端与服务器端，最后对 BTMS 进行功能性测试和性能测试。性能测试的成果中可知 BRTMS 比 BTMS (Bitcoin Transaction Monitoring System) 的交易完成时间从平均 287.12 秒缩减至 1.55 秒，达到更好的顾客消费体验。

在 BRTMS 中，顾客交易信息是匿名且公开透明，同时保障顾客消费权益；商家可以根据所有数字化交易信息为自己的业务目的进行统计和计算，减少手动操作计算结果中的错误。统计数据甚至可以与商家的库存管理相结合，使货物和资金更加便利地进行统计，进一步改善信息的准确性和降低人工成本；政府在解决交易纠纷的过程中拥有更多可信的证据供参考。数字交易交易凭据也可以解决纸本交易凭据丢失或破损的问题。本文提出的 BRTMS 架构包括以下特色如下：

1. 引入匿名支付给实名的交易模型至加密货币交易系统。
2. 设计与实现于加密货币中匿名支付给实名的交易监督系统。
3. 于政府端实现多重签章算法，使实时的交易监督比 Green Address 节点更快速。
4. 透过多重签章算法使得商家可以预防双重支付攻击。
5. 商家商品管理让商家可进行库存管理及商家商品管理。
6. 实现于加密货币中，消费者匿名同时也让消费者拥有消费者权益。
7. 借由将商家实名，BRTMS 使政府主管机关可以有效获得税收。

BRTMS 不仅兼顾顾客和商家同时协助府金融监督单位审计加密货币交易筹集税收。本系统中以 bitcoinj 库实作的 Java 应用程序和 Android 应用程序客户端，修改其在 Android 系统上的开放源代码应用程序，使得本论文阐述之概念能够实际在区块链上运行，未来加密货币不仅能维持目前的便利性，还可以让用户不用担心交易后找不到卖家，使一般民众能更安心使用加密货币作为日常生活的行动支付管道。



## 参考文献

- [1] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*, **2008**.
- [2] Charlie Lee. *Litecoin Official website*, **2011**. <https://litecoin.org>.
- [3] Billy Markus. *DogeCoin*, 2013-12. <http://dogecoin.com>.
- [4] Daniel Kraft. *Namecoin*, 2011-04. <https://namecoin.org>.
- [5] Sunny King. *Primecoin*, 2013-07. <http://primecoin.io/>.
- [6] Constantine Kryvomaz. *Ethereum Classic*, 2015-07. <https://ethereumclassic.github.io/>.
- [7] Ethereum Foundation Vitalik Buterin. *Ethereum*, 2015-07. <https://ethereum.org>.
- [8] *solidity*. <https://solidity.readthedocs.io/en/develop/>.
- [9] Gavin Wood. “*Ethereum: A secure decentralised generalised transaction ledger*”. *Ethereum Project Yellow Paper*, **2014**, 151: 1–32.
- [10] Rainer Böhme, Nicolas Christin, Benjamin Edelman *et al.* “*Bitcoin: Economics, technology, and governance*”. *Journal of Economic Perspectives*, **2015**, 29(2): 213–38.
- [11] *Cryptocurrency Market Capitalizations*. <https://coinmarketcap.com/all/views/all/>.
- [12] John Gregor Fraser and Ahmed Bouridane. *Have the security flaws surrounding BITCOIN effected the currency's value?*, **2017**: 50–55.
- [13] Kyle Torpey. “*You Really Should Run a Bitcoin Full Node: Here's Why*”. *Bitcoin Magazine*, **2017**.
- [14] 蔡怡杼. 银行员监守自盗手法有这些, 2017-10. <https://www.nownews.com/news/20171029/2633984>.
- [15] CM Adams and SE Tavares. *The use of bent sequences to achieve higher-order strict avalanche criterion in S-box design* [techreport], **1990**.
- [16] *Bitcoin Improvement Proposals*. <https://github.com/bitcoin/bips/blob/master/README.mediawiki>.
- [17] *Western Union*. <https://www.westernunion.com/us/en/home.html>.
- [18] *PayPal*. <https://www.paypal.com/>.
- [19] *Digibyte*. <https://www.digibyte.io/>.
- [20] blockchain.info. *Blockchain Size*, **2018**. <https://blockchain.info/charts/blocks-size?timespan=all>.
- [21] J Göbel and AE Krzesinski. “*Increased block size and Bitcoin blockchain dynamics*”. In: *Telecommunication Networks and Applications Conference (ITNAC), 2017 27th International*, **2017**: 1–6.
- [22] Shen Noether and Sarang Noether. “*Monero is not that mysterious*”. *Technical report*, **2014**.
- [23] Emmanuel Bresson, Jacques Stern and Michael Szydlo. “*Threshold ring signatures and applications to ad-hoc groups*”. In: *Annual International Cryptology Conference*, **2002**: 465–480.

- [24] Ming Zhong. “A faster single-term divisible electronic cash: ZCash”. *Electronic Commerce Research and Applications*, **2002**, 1(3-4): 331–338.
- [25] Uriel Feige, Amos Fiat and Adi Shamir. “Zero-Knowledge Proofs of Identity”. *J. Cryptology*, **1988**, 1(2): 77–94. <https://doi.org/10.1007/BF02351717>.
- [26] Thibault de Balthasar and Julio Hernandez-Castro. “An Analysis of Bitcoin Laundry Services”. In: *Nordic Conference on Secure IT Systems*, **2017**: 297–312.
- [27] Ayush Singh Panwar. “Asymmetric Key Cryptography”. *Browser Download This Paper*, **2014**.
- [28] Taher ElGamal. “A public key cryptosystem and a signature scheme based on discrete logarithms”. *IEEE transactions on information theory*, **1985**, 31(4): 469–472.
- [29] Andrew Miller and Joseph J LaViola Jr. “Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin”. Available on line: <http://nakamotoinstitute.org/research/anonymous-byzantine-consensus>, **2014**.
- [30] Addy Yeow. *Global Bitcoin Node Distribution*. <https://bitnodes.earn.com/>.
- [31] Don Johnson, Alfred Menezes and Scott Vanstone. “The elliptic curve digital signature algorithm (ECDSA)”. *International Journal of Information Security*, **2001**, 1(1): 36–63.
- [32] Dmitry Khovratovich, Christian Rechberger and Alexandra Savelieva; ed. by Anne Canteaut. “Bi-cliques for Preimages: Attacks on Skein-512 and the SHA-2 Family”. In: *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*. Springer, **2012**: 244–263. [https://doi.org/10.1007/978-3-642-34047-5\\_15](https://doi.org/10.1007/978-3-642-34047-5_15).
- [33] The Bitcoin Core developers. *Base58*, **2009**. <https://github.com/Bitcoin/bitcoin/blob/master/src/base58.cpp>.
- [34] *The Large Bitcoin Collider*. <https://lbc.cryptoguru.org>.
- [35] Tim Dierks. “The transport layer security (TLS) protocol version 1.2”. **2008**.
- [36] Tatu Ylonen and Chris Lonvick. “The secure shell (SSH) protocol architecture”. **2006**.
- [37] Android Developers Blog. *Some SecureRandom Thoughts*, 2013-08. <https://android-developers.googleblog.com/2013/08/some-securerandom-thoughts.html>.
- [38] bitcoin.org. *Android Security Vulnerability*, 2013-08. <https://bitcoin.org/en/alert/2013-08-11-android>.
- [39] BurtW. *Bad signatures leading to 55.82152538 BTC theft*, 2013-08. <https://bitcointalk.org/index.php?topic=271486.0>.
- [40] Lars R Knudsen, Vincent Rijmen, Ronald L Rivest *et al*. “On the design and security of RC2”. In: *International Workshop on Fast Software Encryption*, **1998**: 206–221.
- [41] Ron Rivest. “Rc4”. *Applied Cryptography by B. Schneier, John Wiley and Sons, New York*, **1996**.
- [42] Ronald L Rivest. “The RC5 encryption algorithm”. In: *International Workshop on Fast Software Encryption*, **1994**: 86–96.
- [43] RL Rivest, MJB Robshaw, R Sidney *et al*. *The RC6 block cipher. v1. 1, August 20, 1998*, **2016**.
- [44] Data Encryption Standard. “Data encryption standard”. *Federal Information Processing Standards Publication*, **1999**.

- [45] American Bankers Association *et al.* “*Triple Data Encryption Algorithm Modes of Operation*”. *ANSI X9*: 52–1998.
- [46] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, **2013**.
- [47] Ronald L Rivest, Adi Shamir and Leonard M Adleman. *Cryptographic communications system and method*. Google Patents, 1983-9 20.
- [48] Neal Koblitz. “*Elliptic curve cryptosystems*”. *Mathematics of computation*, **1987**, 48(177): 203–209.
- [49] Nicholas Jansma and Brandon Arrendondo. “*Performance comparison of elliptic curve and rsa digital signatures*”. *nicj.net/files*, **2004**.
- [50] Léo Ducas, Alain Durmus, Tancrede Lepoint *et al.* “*Lattice signatures and bimodal Gaussians*”. In: *Advances in Cryptology–CRYPTO 2013*. Springer, **2013**: 40–56.
- [51] Scott Vanstone. “*Deployments of Elliptic Curve Cryptography*”. In: *the 9th Workshop on Elliptic Curve Cryptography (ECC)*, **2005**.
- [52] Carlos Pinzón and Camilo Rocha. “*Double-spend Attack Models with Time Advantange for Bitcoin*”. *Electronic Notes in Theoretical Computer Science*, **2016**, 329: 79–103.
- [53] Christian Decker and Roger Wattenhofer. “*Information propagation in the bitcoin network*”. In: *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, **2013**: 1–10.
- [54] Ghassan O Karame, Elli Androulaki and Srdjan Capkun. “*Double-spending fast payments in bitcoin*”. In: *Proceedings of the 2012 ACM conference on Computer and communications security*, **2012**: 906–917.
- [55] GreenAddress. <https://greenaddress.it/>.
- [56] Rostislav Skudnov *et al.* “*Bitcoin clients*”. **2012**.
- [57] Graham Hamilton, Rick Cattell and Maydene Fisher. *JDBC database access with Java: a tutorial and annotated reference*. Addison-Wesley Longman Publishing Co., Inc., **1997**.
- [58] Po-Wei Chen, Bo-Sian Jiang and Chia-Hui Wang. “*Blockchain-based payment collection supervision system using pervasive Bitcoin digital wallet*”. In: *Wireless and Mobile Computing, Networking and Communications (WiMob)*, **2017**: 139–146.
- [59] Hiroki Kuzuno and Christian Karam. “*Blockchain explorer: An analytical process and investigation environment for bitcoin*”. In: *Electronic Crime Research (eCrime), 2017 APWG Symposium on*, **2017**: 9–16.
- [60] Bitcoin Testnet. <https://en.bitcoin.it/wiki/Testnet>.
- [61] 江柏宪、陈伯韦、王家辉、何建明. “匿名加密货币与实名商家交易的有效行动支付监督平台之建置与实作-以比特币为例”. *TANET2017 台湾因特网研讨会*, 2018-02.



## 致谢

首先得感谢李杰教授大力鼓励者我继续往科研的方向前进，同时给予我最多的论文资源与多次的论文指导，总是指引我研究方向，老师严谨的治学态度使我终生受益。刘京老师对比特币区块链技术的了解，在我寻求问题解答的过程中给予不同的见解。

在实习单位中央研究院信息科学所继续展开我的科研路，同时也延续着之前与铭传大学王家辉老师合作的科技部计划“比特币监督收银系统”，因为有着计划的补助也使得在求学的路上获得更充足的预算，也因为计划上的补助，使我能顺利地前往意大利罗马参加 IEEE WiMob 会议发表论文“Blockchain-based payment collection supervision system using pervasive Bitcoin digital wallet.”，参加台湾最大的计算机会议 TANET 发表论文“匿名加密货币与实名商家交易的有效行动支付监督平台之建置与实作-以比特币为例”<sup>[61]</sup>，也得到了 TANET 会议的最佳论文奖，在我人生道路中写下了崭新的一页。感谢李开晖教授愿意教导我并让我在其旗下做科学的研究。



## 北京大学学位论文原创性声明和使用授权说明

### 原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名： 日期： 年 月 日

### 学位论文使用授权说明

(必须装订在提交学校图书馆的印刷本)

本人完全了解北京大学关于收集、保存、使用学位论文的规定，即：

- 按照学校要求提交学位论文的印刷本和电子版本；
- 学校有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务，在校园网上提供服务；
- 学校可以采用影印、缩印、数字化或其它复制手段保存论文；
- 因某种特殊原因需要延迟发布学位论文电子版，授权学校在一年/两年/三年以后在校园网上全文发布。

(保密论文在解密后遵守此规定)

论文作者签名： 导师签名： 日期： 年 月 日