|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 分类号 | TP391 |  | | | 学校代码 | 10590 |
| **UDC** | 620 |  |  |  | 密 级 | 公开 |

|  |
| --- |
| 深圳大学硕士学位论文 |
| 加密货币钱包的分布式破解系统的 |
| 设计与实现 |
|  |
| 马 超 |

|  |  |
| --- | --- |
| 学 位 类 别 | 工程硕士 |
| 专 业 名 称 | 电子与通信工程 |
| 学院（系、所） | 信息工程学院 |
| 指 导 教 师 | 喻建平 教授 |

**深圳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 加密货币钱包的分布式破解系统的设计与实现现 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**

（必须装订在印刷本首页）

本学位论文作者完全了解深圳大学关于收集、保存、使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属深圳大学。学校有权保留学位论文并向国家主管部门或其他机构送交论文的电子版和纸质版，允许论文被查阅和借阅。本人授权深圳大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（涉密学位论文在解密后适用本授权书）

论文作者签名： 导师签名：

日期： 年 月 日 日期： 年 月 日

# **摘 要**

比特币概念的是一位化名为中本聪的人在一篇名为《Bitcoin:A Peer-to-Peer Electronic Cash System》的论文率先提出的。中本聪研究和结合了之前多个数字货币的发明，例如HashCash和B-money，创建了一个完全去中心化、分布式点对点的电子现金系统，此现金系统不依赖于任何通货保障或是结算交易验证保障的中央权威。

比特币的所有权是通过数字密钥、比特币地址和数字签名来确立的。数字密钥实际上并不是存在于比特币网络中，而是由用户生成并存储在一个文件或者简单的数据库当中，称之为钱包。存储在用户钱包中的数字密钥完全独立于比特币协议，可由用户的钱包软件生成并管理，而无需区块链或者网络连接。密钥实现了比特币许多特性，包括去中心化的信任和控制、所有权认证和基于密码学证明的安全模型。

随着信息时代的快速发展，人们对数字财产的安全保护措施越来越重视。对于比特币而言，钱包文件就是用户重点保护的对象。用户在使用比特币钱包时，出于对财产的保护，往往会对钱包文件进行多次备份或者加密，而在用户遗忘某个钱包文件密码时，情况就会变得十分糟糕。由于比特币离线钱包并不具备密码找回功能，当密码被遗忘的情况发生时，钱包文件里的财产是无法使用的，因此这种情况会带来不必要的损失。甚至有人在网上发布悬赏请求他人帮忙破解遗忘密码的钱包文件。

本论文就是针对比特币桌面钱包Bitcoin-Qt所生成的钱包文件无法提供密码找回功能的缺陷，从而提出一种破解及其优化方案。构建并实现一个分布式的密码破解系统，然后利用图形处理器（GPU）在通用并行计算上强大的计算能力，优化破解方案，缩短破解需要的时间、提高破解的效率。为遗忘密码的用户提供一项实用性较强的应用服务。

**关键词：**比特币钱包；暴力破解；分布式系统；GPU加速运算

# **Abstract**

The concept of Bitcoin was first brought by Satoshi Nakamoto in his thesis “Bitcoin: A Peer-to-Peer Electronic Cash System”, To combine the features of other digital currencies created before, like HashCash and B-money, he created a digital currency system that is decentralized and peer to peer, the system doesn’t rely on any currency based platform or settlement verification centralized authority.

The ownership of Bitcoin is confirmed by the technologies of digital keys, addresses and digital signature. The digital keys are generated in a document or in the database instead oft existing in the Bitcoin network, which is called wallet. The keys that stored in the wallet is completely independent of the Bitcoin protocol. It is managed by the wallet software that the user is using and it doesn’t need to be connected by blockchain or other network. The key technology has many features just like Bitcoin, including the authorization and control of decentralization and the security module based on cryptology.

With the development of information era, people are paying more and more attention to the safety of their digital assets. For example, some people will multi encrypt or have more back up documents for their wallets to keep it safe, however, it gets more troublesome when people forgot the password. For there is no way to get back the password of the off-line wallet, people can not access to their digital assets when they forgot the password, some people even post a reward to help them crack their password. Therefore, it is not a smart choice to have too many passwords.

This thesis is mainly focused on improving the disadvantage of desk wallet when the password is lost. To establish a distributed system to crack the password and shorten the processing time by the using of GPU processor. Thus, the aim is to provide the users a practical application to find their password.

**Key word：**Bitcoin wallet ; Brute-force break; Distributed system; GPU computing acceleration

**目 录**

[摘 要 I](#_Toc4016)

[Abstract II](#_Toc443)

[第1章 绪论 1](#_Toc11398)

[1.1 背景与意义 1](#_Toc16209)

[1.2 国内外研究现状 2](#_Toc19362)

[1.3 论文主要内容与结构 3](#_Toc7685)

[第2章 比特币的原理与特点 4](#_Toc12420)

[2.1 比特币的原理 4](#_Toc19713)

[2.1.1 比特币产生原理 4](#_Toc26829)

[2.1.2 区块链 5](#_Toc14865)

[2.1.3 交易 6](#_Toc12034)

[2.1.4 比特币钱包 7](#_Toc20963)

[2.2 钱包构成 7](#_Toc28752)

[2.2.1 数据结构 7](#_Toc6984)

[2.2.2 交易信息有效性 7](#_Toc30034)

[2.2.3 交易信息合法性](#_Toc19793) **[错误！未定义书签。](#_Toc19793)**

[2.3 比特币通讯模型 9](#_Toc28204)

[2.4 本章小结 12](#_Toc16257)

[第3章 系统方案设计和关键技术 13](#_Toc25731)

[3.1 系统设计原理 13](#_Toc19564)

[3.2 系统需求分析 13](#_Toc27151)

[3.3 系统架构设计 13](#_Toc17023)

[3.4 关键技术 15](#_Toc20696)

[3.4.1 网络协议分析技术 15](#_Toc7102)

[3.4.2 远程过程调用 17](#_Toc25194)

[3.4.3 网络监控 18](#_Toc15547)

[3.4.4 MySQL数据库 18](#_Toc30072)

[3.4.5 建立比特币vps节点 19](#_Toc15390)

[3.5 本章小结 19](#_Toc32201)

[第4章 系统模块设计与实现 20](#_Toc32612)

[4.1 比特币节点编译 20](#_Toc20583)

[4.2 网络数据包捕获模块设计与实现 22](#_Toc7891)

[4.2.1 设计流程 22](#_Toc26212)

[4.2.2 实现流程 22](#_Toc20901)

[4.3 协议分析设计 24](#_Toc31413)

[4.3.1 设计流程 24](#_Toc27900)

[4.3.2 实现流程 25](#_Toc1009)

[4.4 区分新交易设计与实现 28](#_Toc4902)

[4.4.1 设计流程 28](#_Toc20726)

[4.4.1 实现流程 30](#_Toc24111)

[4.5 存储子模块设计与实现 33](#_Toc1981)

[4.5.1 设计流程 33](#_Toc26581)

[4.5.2 实现流程 34](#_Toc32754)

[4.6 本章小结 35](#_Toc25184)

[第5章 系统测试 36](#_Toc32507)

[5.1 系统功能和性能测试 36](#_Toc20710)

[5.2 结果评价与方案设计总结 43](#_Toc18845)

[5.3 本章小结 44](#_Toc8964)

[第6章 总结与展望 45](#_Toc24412)

[6.1 论文总结 45](#_Toc22296)

[6.2 后续工作展望 46](#_Toc5705)

[参 考 文 献 47](#_Toc8055)

[致 谢 50](#_Toc16864)

# 第1章 绪论

## 1.1 背景与意义

2008年，一位化名Satoshi Nakamoto的人，在一篇名为《Bitcoin: A Peer-to-Peer Electronic Cash System》的文章中首次提出比特币的概念。比特币是一个基于数字货币生态系统而形成的理念与技术的集合。

在比特币的网络中，比特币也作为参与者之间的价值存储和传递。虽然也可以使用其他传输网络，但是比特币用户还是主要集中在因特网上，利用比特币协议相互交流。作为一款开源软件，比特币的协议栈可以在包括笔记本电脑和智能手机在内的多种设备上运行这使得比特币技术易于获取，用于买卖商品、给组织或者汇款，或是延长贷款期限。比特币在特殊的货币交易所可以购买、出售，以及与其他币种进行兑换。从一定意义上来说，比特币就是互联网货币的完美形态，因为它的快捷、安全以及无国界的特性。

用传统货币能做的事情，用户在网络上利用比特币都可以做到，包括发送给新的接收者。这些密钥通常存储在每个用户的电脑桌面钱包中。每一笔交易都需要用密钥解密，这是比特币的唯一先决条件，它完完全全掌握在每个用户手里。

对比特币的钱包来说，最关键的东西非私钥莫属。但是私钥是一个随机生成的64位长字符串（未压缩），想要记住它是比较困难的。因此比特币钱包程序通过对钱包文件设置相对简单的用户口令来加密包括私钥在内的钱包信息，用户口令既可以用来解密钱包文件，也可以用来签名交易（实际上还是使用私钥签名）。极大程度上方便和简化了钱包的使用。

由于两方面的原因，比特币钱包程序无法提供用户口令找回的功能，这使得用户在遗忘口令时，情况会变得十分糟糕：

（1）钱包文件的生成实际上完全脱离比特币网络，钱包文件没有口令是无法提取内部的信息，而且也很难通过钱包文件名区分不同的钱包文件（钱包文件可以重名）；

（2）比特币网络是去中心化的信任网络，可以理解为每一个节点都是100%可以信任，但是也可以认为每一个节点都是无法信任的，也就是无法拥有一个可信第三方为比特币用户提供这个功能。

虽然，用户可以选择基于信誉机制的钱包应用提供平台或者APP，但是对于长久保存的目的，大部分用户会选择自己备份离线钱包文件。而且由于比特币特有的找零机制，为了安全起见，官方建议每发出100次交易就应当备份一次钱包文件。那么在一些频繁备份钱包文件或者是长久存储备份钱包文件的背景下，遗忘密码的情况就难以避免地会发生。

因此，许多用户期待有一款可以帮助用户恢复其钱包文件密码的应用。由于比特币钱包加密算法是现阶段较安全的AES-256加密算法，靠寻求算法的漏洞来破解的可能性暂时来说是非常渺茫的，暴力破解和密码表破解是唯一的选择。这就导致普通的破解效率及其低下，一些必要的技术手段和优化方案是我们研究的方向。

本论文就以比特币官方发布的Bitcoin-QT桌面钱包所生成的钱包文件为例，基于分布式网络来构建一个可行的钱包破解系统，借用GPU强大的并行运算能力，进一步提出优化方案。

本论文讨论的破解系统与比特币领域中的矿池较为类似，借助大量的“矿机”，我们可以大大降低破解所需的时间，将之前几乎无法完成的任务，变成在可以承受的时间段内完成，给遗忘密码的用户带来了希望。

## 1.2 国内外研究现状

从2008年到现在，研究比特币的文章逐年增长：从2008年的1篇到2014年的#篇，越来越多的人投入比特币的研究当中。在目前所有研究比特币的文章中，技术研究仅仅只有#篇，对比特币钱包的研究也基本都是往安全方向发展，暂时没有研究比特币钱包破解的文章。

虽然没有研究比特币钱包破解的文章，但是却可以借鉴其他研究破解的文章。

## 1.3 论文主要内容与结构

本系统的主要思路是在用暴力破解钱包密码的基础上，将其与任务分发系统结合，实现一个分布式破解钱包文件的系统；之后再对破解环节进行优化，使用GPU提高运算效率，达到加速破解的效果。该系统主要用到任务分发、OpenCL、LAMP等技术。

本论文主要论述和验证此破解系统理论的可行性。论文分为下面六章。

第1章，介绍课题研究的背景与意义，密码破解研究的国内外研究现状，以及论文的主要内容和工作安排。

第2章，介绍比特币的原理，包括产生原理、区块链、挖矿、钱包等；重点阐述了比特币钱包的构成、以及钱包信息的存储方式；最后，详细分析了比特币钱包所用的加密算法。

第3章，介绍该系统的方案设计原理，对系统需求进行分析以及对系统进行架构设计，并对本系统实现所需用到的各项关键技术进行了简单的介绍。

第4章，描述方案的各个模块的设计过程，包括所用到的工具、软件选择、算法。并对系统的具体开发实现流程进行详细的介绍。

第5章，在原有系统基础上，提出GPU破解方案，对关键环节进行改进，实现并测试新方案的效率，与原有方案的效率进行对比。

第6章，总结与展望。

# 第2章 比特币的原理与特点

## 2.1 比特币的原理

比特币是一个全新的货币系统，基于密码学算法信任而非个人信任模式，在系统中的双方，能够进行相对零信任的交易。这种交易是不可撤销、按照点对点方式进行。可以将比特币系统理解成一个基于强有力密码学算法保障的账簿，它有如下特点：

(1) 去中心化 ：比特币系统是一个P2P的网络，整个网络中的节点都是相同的，并没有一个绝对的中心，所有的通信都是通过广播验证的方式进行，保证通信的公平与透明。

(2) 匿名：比特币的私钥是近似随机的生成，并不与比特币网络直接关联，且使用地址也无需实名认证，因此比特币系统具有一定的匿名性。

(3) 基于密码学算法的信任：比特币几乎所有环节都与密码学息息相关，从挖矿（POW工作证明）到交易再到钱包，强有力的算法保障使基于零信任模式的系统得以有条不紊地运行。

### 2.1.1 比特币产生原理

比特币产生原理图如图2-1所示；



图 2-1 比特币产生的原理图

### 2.1.2 区块链

区块链，又称帐链。对用户来说，区块链记录着所有比特币交易的详细信息，而对计算机而言，区块链是由包含交易信息的区块从后向前有序链接起来的数据结构。它可以被储存为Flat File（一种包含没有相对关系记录的文件），或是储存在一个简单的数据库中。比特币核心客户端使用Google的LevelDB数据库来存储这些区块链数据。区块有序地链接在一个链条里，每一个区块都指向钱一个区块，最终构成区块链。



图 2-2 区块链

上图所示的是一个简单版本的[区块链](http://8btc.com/" \l "block-chain)，仅仅包含整个区块链的35998到36000三个区块。

### 2.1.3 交易链

简单来说，比特币交易就是告知整个网络：比特币已经从交易的输入转到了交易的输出。为了防止双花的可能性，比特币的交易输入只能被引用一次。下图是一个简化了的交易链。



图 2-3 比特币交易链

交易是将钱从输入移至输出，输入是指钱币的来源，通常也是指之前一笔交易的输出。交易的输出则通过关联一个密钥的方式将钱赋予一个新的所有者。目的密钥被成为是安全锁。这样就给资金强加了一个要求：有签名才能在以后的交易中拿回资金。也就是说交易输出需要验证之后才能拿回输出给他的币。

交易费实际上是输入和输出的差值，多余的币是不能留在交易中，没有设置输出地址，因此也无法输出到某一个地址上。比特币网络把这样的币定义为交易费，奖励给挖到含有这个交易区块的矿工。例如上图所示，交易输出比交易输入少了0.0001BTC，这些币就成为了交易费。

### 2.1.4 比特币钱包

从广义上来说，比特币钱包是基于比特币网络协议，用来存放、发送、接收比特币的一款软件；从狭义上来说，比特币钱包是存储用户数字密钥的载体，这里的载体可能是文件、硬件设备甚至是一串字符。本论文主要讨论的是狭义上的比特币钱包。

存储在钱包中的数字密钥是完全独立于比特币协议与网络，用户身上会直接看到数字密钥，都是借用相应的钱包程序可以对钱包进行管理。对于比特币技术爱好者来说，可以通过转换数字密钥的格式让钱包在各个不同公司的钱包程序中使用。

## 2.2 钱包介绍

钱包信息主要由密钥对：私钥、公钥和地址构成。

### 2.2.1 私钥

比特币的私钥实际上是一个随机选出的数字而已，但是却控制着一个或者多个比特币地址中的所有资金。生成私钥是钱包中较为关键的一步，是要足够安全的熵源，即随机性来源。生成一个比特币私钥在本质上与在1到2256选取一个数字无异。只要选取的结果是没有办法预测或者没有办法重复，那么如何选出这个随机数并不重要。比特币软件使用操作系统底层的随机数生成器来生成256的熵。通常情况下，操作系统随机数生成器由人工的随机源进行初始化。

相同的私钥有不同的压缩格式，如下表所示：

|  |  |
| --- | --- |
| **格式** | **私钥** |
| Hex | 1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD |
| WIF | 5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn |
| Hex-compressed | 1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD01 |
| WIF-compressed | KxFC1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ |

表2-1 私钥格式

### 2.2.2 公钥和地址

比特币的公钥与私钥互为一个密钥对，是私钥通过椭圆曲线算法产生的。产生过程是不可逆的，也就是说无法通过公钥得到与其对应的私钥。由于比特币的某种程度上是会公布出去，因此不可逆的产生方式是安全的保障。

以一个私钥为k例，公钥K是通过私钥k乘以曲线上生成点G得到，生成点G是secp256k1标准的一部分：



公钥是用来解开私钥签名的数据，使用私钥签名交易之后，会把自己的公钥一起发送，私钥签名的数据可以使用公钥解密，发送公钥之后旷工才能验证私钥的签名的正确性（能不能解开），私钥和公钥是成对出现的，一个私钥签名的数据，只有对应的公钥才能解开，而地址也是从公钥生成的，这样就可以验证花费的交易是不是属于这个地址。

比特币地址是由公钥进过单向加密哈希算法生成的以数字“1”开头的字符串，例如：

1D2BmyoXzMP3mXECyG6ihNda3dDXpYwvdY

哈希算法是一种单向函数，接收任意长度的输入产生指纹摘要。生成比特币地址时使用SHA-256和RIPEMD160算法，用公式可以近似表示为：



但是实际上生成地址并不是这么简单，进行哈希运算的除了公钥还有前缀和校验码，这样做是为了方便钱包判断地址是否合法，下图为钱包生成地址的详细过程。

图 2-4 地址生成



## 2.3 算法介绍

比特币钱包加密环节主要用到的算法包括AES-256和Secp256k1 两种算法，下面就分别对这两种算法做简要介绍。

### 2.3.1 AES-256

111111

### 2.3.2 Secp256k1

准确来说，Secp256k1并不是一种算法，而是椭圆曲线数字签名算法（ECC）里的一个曲线标准。ECC最早是由Neal Koblit和Victor Miller于1985年首先提出，从那时起ECC的安全性和实现效率就被众多的数学家和密码学家所广泛研究。所得的结果表明，较之RSA算法，ECC具有密钥长度短，加解密速度快，对计算环境要求低，在需要通讯时，对带宽要求低等特点。近年来，ECC被广泛应用于商用密码领域，这可由ECC被许多著名的国际标准组织所采纳所佐证，如ANSI（American National Standards Institute）、IEEE、ISO、NIST（National Institute of Standards and Technology）。

在理解椭圆曲线之前需要理解两个概念：平行线和射影平面坐标系。

1. 平行线

故名思意就是两条永不相交的直线。但是在近代数学基础中却提出了这样的概念：既然可以假设平行线是永不相交，也可以假设平行线在很远很远的地方相交了，既平行线相交于无穷远点。

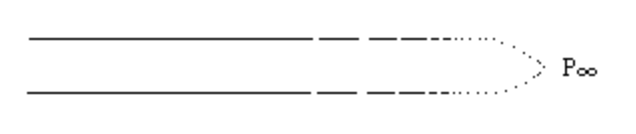


图 无穷远点示例

直线上出现点，所带来的好处是所有的直线都相交，有且只有一个交点。为与无穷远点相区别把原来平面上的点叫做平常点。

平行线的特性：

1. 直线L的无穷远点有且只有一个
2. 平面上一组互相平行的直线有公共无穷远点
3. 平面上任何相交的两直线有不同的无穷远点
4. 平面上全体无穷远点构成一条无穷远直线
5. 平面上全体无穷远点与全体平常点构成射影平面
6. 射影平面坐标系

射影平面坐标系是针对普通平面坐标系的扩展，可以理解为射影平面坐标系就是可以表示无穷远点坐标的坐标系。

设普通平面坐标系上的坐标为，然后令。则A点可以表示成，含有三个参数的坐标点，形成了一个新的坐标体系。这个新的坐标体系能够表示射影平面上所有的点，我们就把这个能够表示射影平面上所有点的坐标体系叫做射影平面坐标系。

在弄清楚上述概念之后，那么就可以用方程来表示椭圆曲线的定义：



所有满足上述方程的点的集合，组成了一条椭圆曲线。且曲线上的每个点都是非奇异（或光滑）的。

定义的解释：

1. 上述方程式Weierstrass方程，是一个齐次方程
2. 虽说是椭圆曲线，但是并不是椭圆
3. 所谓“非奇异”或“光滑”的，在数学中是指曲线上任意一点的偏导数不能同时为0，即是曲线上任意一点都存在切线
4. 椭圆曲线上有一个无穷远点

知道了椭圆曲线上的无穷远点。我们可以把椭圆曲线放到普通平面直角坐标系上。因为普通平面直角坐标系只比射影平面坐标系少无穷远点。我们在普通平面直角坐标系上，求出椭圆曲线上所有平常点组成的曲线方程，再加上无穷远点，就可以构成椭圆曲线。

令，带入上述Weierstrass方程得：



也就是说满足Weierstrass方程的光滑曲线加上一个无穷远点，组成了椭圆曲线。因此简化的运算和表述方式都是以普通平面坐标系表示。下图是一个椭圆曲线的示例，类似于比特币所用的曲线。

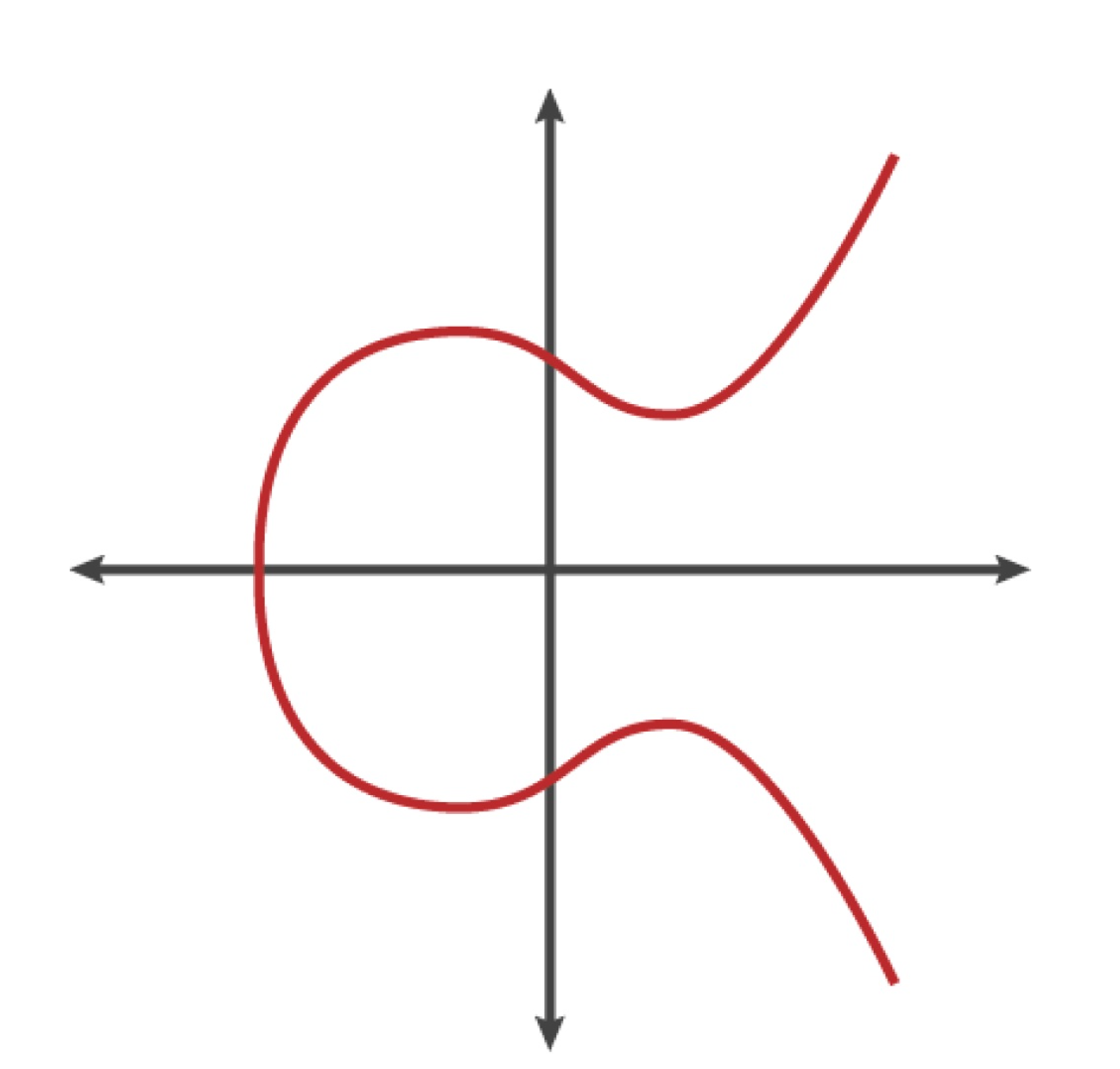


图 椭圆曲线示例

以上提到的椭圆曲线都是连续的，而在密码学中需要的是离散的点。

比特币使用的是secp256k1标准所定义的一条特殊的椭圆曲线和一系列数学常数。该标准由美国国家标准与技术研究院（NIST）设立。Secp256k1曲线由下述函数定义，该函数可产生一条椭圆曲线：

 或 

上述表明该曲线实在素数阶的有限域内，也写作，其中，这是非常大的一个素数。

这条曲线被定义在一个素数阶的有限域内，而不是定义在实数范围内，它的函数图像看起来像分散在两个维度上的散点，因此很难画图表示。不过其中的数学原理与实数范围的椭圆曲线相似。下面用素数阶为17的有限域内椭圆曲线作示例，其形式为网格上一系列的散点。Secp256k1可以理解成一个更大更复杂的散点图。

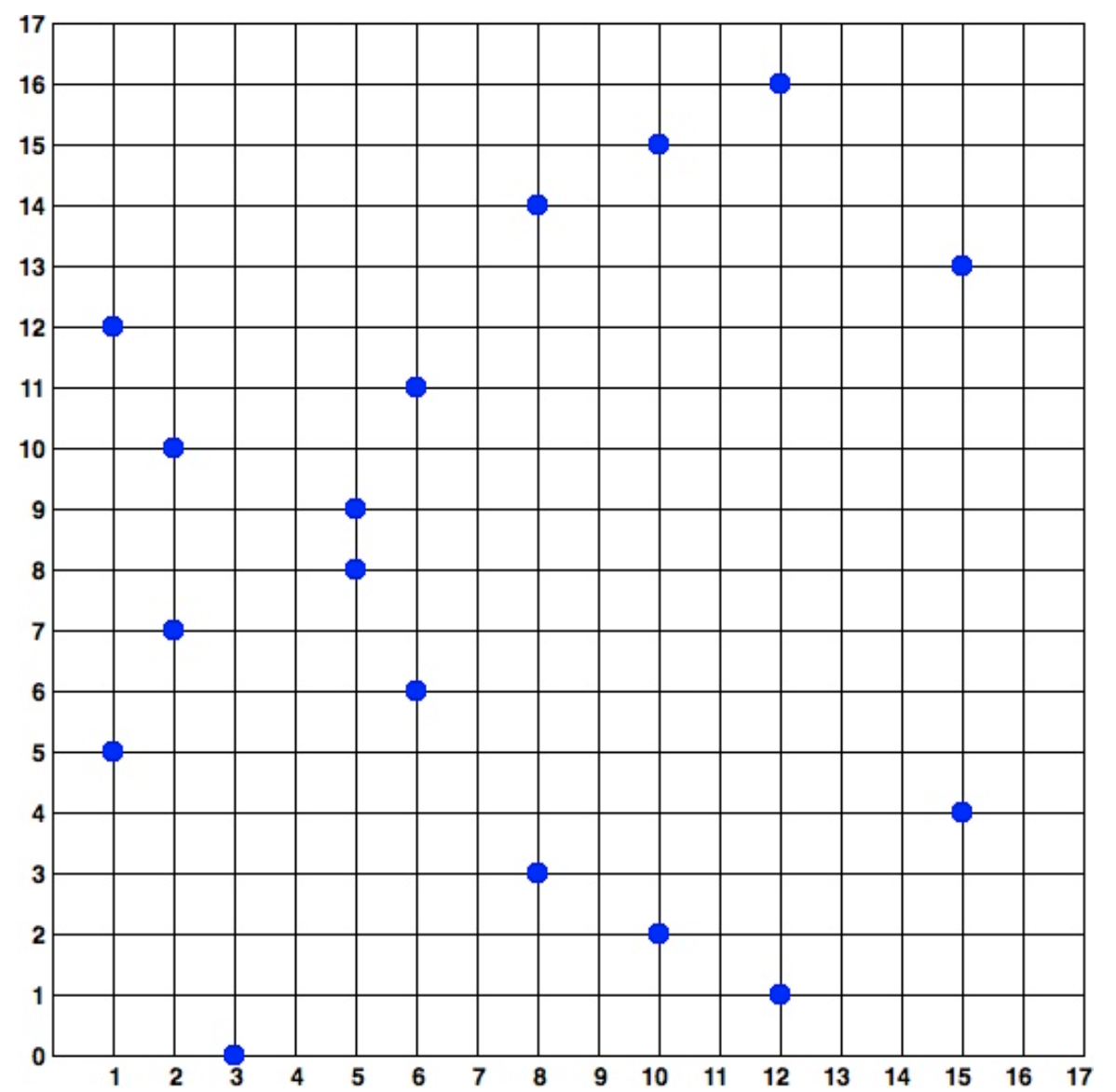


图 上的椭圆曲线，

对于椭圆曲线的运算，我们只需要了解加法以及乘法运算法则，因为在生成公钥时只需要用到乘法。

加法运算法则：任意取椭圆曲线上两点、 （若、 两点重合，则做点的切线）做直线交于椭圆曲线的另一点，过做y轴的平行线交于。我们规定。这里的加法并不是实数范围内的普通加法，而是抽象出来的加法。

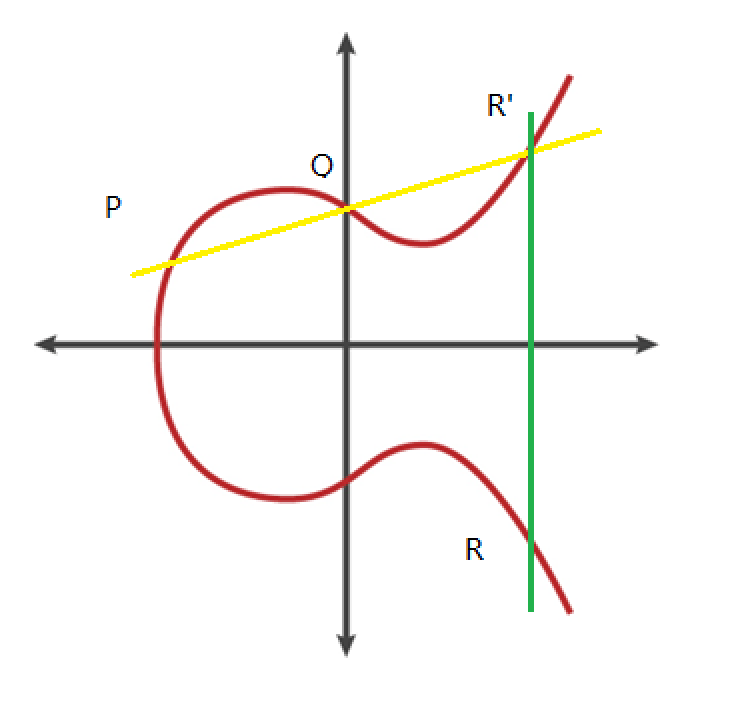


图 加法运算

简单的乘法规则可以认为是：，一共个相加的结果。

利用上述的加法与乘法的定义，就可以理解之前提到的公钥生成公式。椭圆曲线的特性使得在拥有公钥的情况下很难逆推出其对应的私钥。这也是在本论文中用到的特性。

## 2.4 本章小结

本章首先就比特币的系统原理做了一个比较详细的阐述，包括其产生原理、挖矿机制、比特币钱包、区块链等；分析了比特币的技术特点，以及简单介绍了根据其技术特点可以尝试的应用场景；其次，重点阐述了比特币的伪匿名性，为本系统的实现提供了理论依据。对比特币交易的构成以及判断比特币交易的合法性做了简单介绍。最后，分析了比特币的通讯模型。

# 第3章 系统方案设计和关键技术

## 3.1 系统设计原理

节点自身创建的交易（如Alice打10个BTC到Bob账户）在第一次向比特币网络发送时，网络上比特币块链中的所确认的交易（blockhain中的记录）和等待被确认交易记录（在节点内存池中保存的新的交易信息）中必然不存在该交易信息。所以，如果在Alice的网络接口端对比特币交易数据进行监控，当发现Alice向外发送了一条比特币网络上（mempool和比特币块链中）并不存在的交易时，就可以认为该交易信息就是Alice使用的ip地址创建发送的，就可以把Alice使用的比特币地址与使用的ip地址绑定，达到监控的目的。（比特币协议并未要求对数据加密，数据内容对外可见，可以对数据进行分析）。

新交易定义：某一比特币交易信息在比特币网络上的某一节点，向全网广播的尚未出现在比特币网络系统中的的交易信息，称为新交易。

监控目的是发现属于该节点自身创建的交易信息，即网络上某一交易信息的发送源。新交易判定是比特币监控系统最重要的一个环节，是把比特币地址与一具体的ip地址/MAC地址对应起来重要依据。

## 3.2 系统需求分析

1. 用户需求

本系统基于比特币网络的匿名性，针对在网络上传播的比特币交易信息进行实时监控，在比特币客户端节点网络中安装监控软件，根据解析的交易信息内容绑定IP地址与比特币地址的关系，追查交易信息发送源、判断感兴趣的实体的信息，对洗黑钱、非法交易、逃税漏税等在比特币网络上的非法活动进行追踪监控，达到打击犯罪并规范比特币市场行为目的。可为监管机构提供一种监控比特币的方案。

1. 功能需求

将比特币网络中的比特币数据进行捕获过滤，并能筛选出交易相关信息，发现交易的比特币地址与ip地址之间的联系。

## 3.3 系统架构设计

在目标网络与互联网连接的接口处，如路由器、交换机、网关，设置监控节点，运行监控程序，将监控区域内节点对外发送的交易相关信息（“inv”与“tx”）实时捕获，并立刻通过RPC远程过程调用方法向比特币网络上的VPS节点查询服务器查询网络上是否存在该交易，若查找不成功，说明这是一条节点向外广播的新交易，判定该交易是比特币新交易。查询服务是一个与比特币网络交易信息同步率高的VPS完全节点，拥有较完整的内存池信息。综上，本监控系统拓扑图3-1：

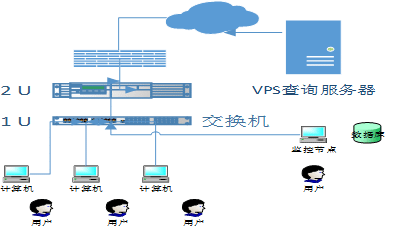


图 3-1 系统网络拓扑图

本系统实现可分为以下4个步骤：

(1) 编译并配置VPS新交易查询服务的比特币节点，并让节点同步网络上的比特币交易信息。为的是保证监控程序存储的交易消息不能落后于被监控节点交易消息，否则会把转发的交易当成新交易。如果发生断网等意外情况导致信息落后，则要重新同步信息后再进行新交易判定。

(2) 捕获节点发送和接收的交易信息（“inv”和“tx”消息），为保证捕包质量，采用多线程方法，建立消息缓存队列，将捕包与消息处理流程分开。

(3) 从线程取出捕获比特币交易相关数据，先判定数据是否是已经判定过的新交易信息，若不是再通过json RPC向查询服务器查询该交易是否已经存在，判断新交易。

(4) 将解析后的数据设计数据库进行存储。

综上，根据本系统实现的步骤，可将本系统的分模块设计如图3-2所示。分模块做简单介绍：

1. 网络数据包捕获模块

此模块是本系统最开始的部分，监控系统操作对象是含有Bitcoin协议的数据包，首先应该设置过滤条件，捕获需要的比特币数据包。

1. 比特币协议解析模块

本模块是该系统中一个重要的组成部分，只有在完成对捕获的数据包内容详细分析之后，才能根据捕获数据具体内容，设置相应的监控过滤条件。

1. 区分新交易

此模块是本系统设计的核心，监控程序只捕捉由被监控节点发起的新的交易，才能有效地发现将IP 地址与比特币地址之间的联系。

1. 存储模块

此模块主要是存储解析后的数据，将捕获数据记录保存。

比特币新交易监控系统

存 储 模 块

区 分 新 交 易

网络数据包捕获模块

比特币协议解析模块

图3-2 系统模块设计图

## 3.4 关键技术

### 3.4.1 网络协议分析技术

网络协议分析[13]就是指对网络上的数据进行相应的协议分析。协议分析的过程主要分为三部分内容：捕获数据包、过滤数据包和具体协议解析。

图3-3所示为常用的以太网数据包截获程序 tcpdump 的数据包捕获流程图[24]。

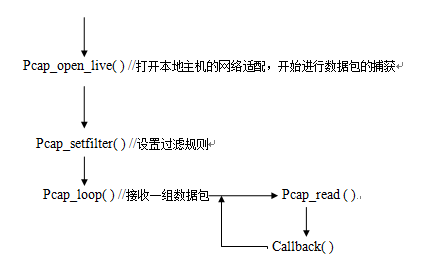


图 3-3 数据包捕获流程图

其中当程序进入 Pcap\_loop 循环之后上层调用Libpcap 的程序通过 callback 方式进行调用。也就是说上层调用程序将提供一个函数接口给 Libpcap，当 Libpcap 读到数据将自动调用上层提供的接口函数进行下一步处理。

由于在网络上的数据信息量是非常惊人的，不可能对每个数据都进行详细的协议分析，并且在实际应用中也会有这种情况：只分析某一特定具体的协议，而其他协议都不在考虑范围。所以，这就需要对捕获到的数据包进行过滤[25]。

TCP/IP 协议簇是目前应用最广的协议簇，它是网络通信协议的核心[26]。TCP/IP 协议簇分层模型如图3-4所示。每个层次包含许多协议，且下层协议在实现时可以体现上层协议的一些细节。TCP/IP各层数据报格式如图 3-5所示

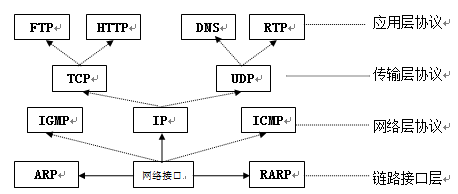


图3-4 TCP/IP协议簇结构示意

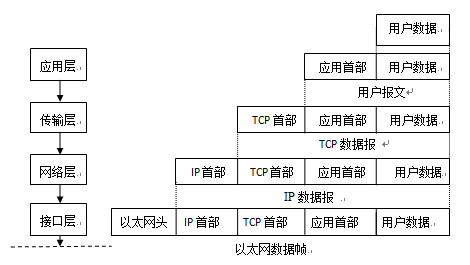


图 3-5 TCP/IP各层数据报格式示意

为了有效的解析含有bitcoin交易的数据包，需要此系统的捕获数据包模块能对网络上所传送的数据包进行有效监听与捕获，这是协议解析的第一个环节，也是整个系统的数据基础[27]。该系统本模块选用开发包libnids。Libnids是基于Windows平台，免费且公开的网络抓包工具。

libnids是用于网络入侵检测开发的专业编程接口。它不但实现了基于网络的入侵检测系统的基本框架，而且实现了入侵检测系统的底层功能，这使得开发者可以专注于高层的功能开发[28]。

### 3.4.2 远程过程调用

比特币监控节点在与VPS节点通过RPC（Remote Procedure Call，远程过程调用）进行通信。RPC协议是指某一程序可以在不知道网络细节的前提下，可以基于此协议对网络内另一台计算机程序请求服务[29]。RPC使用的模式是客户机/服务器模式。服务器和客户端的通信过程分为请求过程和响应过程两部分。它的系统模型如图3-6所示。服务器客户端请求响应流程如图3-7 所示。

客户进程 服务器进程

客户程序

客户存根

通信模块

服务器存根

调度程序

服务过程

通信模块

服务接口

图3-6 RPC系统模型

客户程序

8 编码

5 解码

6响应请求

服务过程

11获得结果

1发出请求

7 返回结果

服务器存根

2 编码

客户存根

10 解码

调度程序

4 解码并调度

3发往外部格式的数据

通信模块

通信模块

9发往外部格式的数据

图3-7 RPC服务器客户端请求响应流程

[json rpc](http://baike.baidu.com/view/12822065.htm" \t "_blank) 是一种以json为消息格式的远程调用服务，使用[http](http://baike.baidu.com/view/9472.htm" \t "_blank)作为[传输协议](http://baike.baidu.com/view/441895.htm" \t "_blank)，也可以使用其它传输协议，传输的内容是json消息体。相对[xml rpc](http://baike.baidu.com/view/1158493.htm" \t "_blank)来说，json rpc 具有很多优点：体积小巧、格式简单、占用带宽小，解析较易[29]。

比特币提供超过50种RPC调用方法，运行“bitcoin –server”或“bitcoind'”使bitcoin以[HTTP JSON-RPC](http://json-rpc.org/wiki/specification|)服务器模式运行，但与其通信时必须设置必须设置 rpcuser 和 rpcpassword 确保 JSON-RPC 的安全运行。使用cURL或其他命令行HTTP-fetching可以发送命令并查看结果，例如：

curl --user user --data-binary '{"jsonrpc": "1.0", "id":"little proj", "method": "getgetbestblockhash", "params": [] }' -H 'content-type: text/plain;' http://127.0.0.1:8332/

将会看到类似下面的信息:

{"result":"0000000000000005217e3a0547a464e6b8972912cde54ff6c98d03519e369e69", "error":null, "id":"little proj"}

### 3.4.3 网络监控

随着互联网的高速发展，互联网的使用愈加频繁[30]。它不仅是企业内部的沟通桥梁，也是企业与外部进行各种业务往来的重要管道。网络监控是对局域网内的计算机进行监视和控制[31]。它有以下几种工作模式：网关模式、网桥模式、旁听模式。

网关模式采用的是存储转发的模式，与路由器的工作方式类似，效率可以得到保障。这种方式可以允许拦截消息，这对判定新交易的实时性上有帮助，但其很容易造成单点故障，在没有有效方法解决情况下，不适合本方案。网桥模式虽可以避免单点故障，但是由于本系统监听的是TCP协议属于网络层，故该方法也不适用于本方案。旁听模式是通过在交换机上配置镜像端口，不需要改变现有的网络状态。当监控设备意外停止工作时，不会影响网络的正常运行[32]。综上所述，本设计选用旁路监听的模式最佳。

### 3.4.4 MySQL数据库

MySQL是一种关系型数据库管理系统，由于它体积小、速度快、成本低、开放源代码等优势，被广泛应用于中小网站的研发中[33]。MySQL是一个多线程、多用户、结构化查询语言的数据库服务软件，它以客户机/服务器的结构形式实现[34]。本系统选择 MySQL 作为系统数据库，有以下几个原因。

(1) 速度快，使用简单。使用多重 SQL 查询优化，这能有效的提高查询速度。

(2) 可移植性，成本低。MySQL 可运行在多种不同的系统上，例如 Window、Linux等，兼容性很强，用户可以根据自己的需求来选择操作系统或者多系统之间的移植。它不仅开放源代码，而且对大多数个人用户来说是免费的。

(3) 兼容性好，MySQL 为多种编程语言提供了接口，方便开发者使用这些语言进行数据库操作[34]。

(4) 管理工具多，现市场上出现多重 MySQL 管理工具，例如：Navicat 等，这些管理工具对开发者实行数据库的管理非常受用。

(5) 集成化平台，MySQL 受到众多开发者的喜爱，现有多重集成 MySQL 的开发运行环境，例如 Lamp，Wamp。这方便开发者进行运行环境搭建。

本监控系统需要将监控的数据进行存储，有IP地址以及对应的比特币交易id，输入输出地址和交易量等信息，可通过mysql设计数据表以供查询。

### 3.4.5 建立比特币vps节点

比特币网络上的节点一般分为完全节点和普通节点，其中，完全节点有比较高的传输上行带宽和下行带宽，开放端口与其它比特币节点连接进行数据交互[35]，存放所有可能在下一个区块被确认的交易信息，并能为SPV节点和矿工节点提供内存池信息查询功能，2014年度全球有接近7000个完全节点。一个大的完全节点能和几百上千个比特币节点相连接。

普通节点一般不开放8332端口，且没有完整的内存池信息，内存池信息更新比较慢，比特币新交易监控对等待确认的交易信息的集合实时性要求很高，若是用普通节点来判断比特币新交易，就会出现漏判、误判，故选用完全节点。而普通网络带宽达不到要求，所以本实验需要使用vps来编译一个完全节点作为新交易查询服务器[36]。

本设计需要一个完全节点提供完整mempool的实时信息的查询服务，故在方案设计中需要租用一个VPS并对编译该比特币节点，这部分是方案设计很重要的一个环节。

## 3.5 本章小结

本章首先对方案设计原理进行描述，并对系统设计进行了需求分析，进而根据需求，对系统架构进行设计。接着介绍了实现本系统各模块功能所需要用到的关键技术，如网络协议分析技术、远程过程调用RPC、网络监控以及MySQL数据库等。

# 第4章 系统模块设计与实现

本章首先根据本文3.3节可知，应该先为系统设计准备条件，即编译并配置VPS新交易查询服务的比特币节点，让节点同步网络上的比特币交易信息。接着针对各个模块进行设计，包括分析数据流、设计的重难点及解决方案。并对系统的具体开发实现流程进行详细的介绍。

## 4.1 比特币节点编译

SecureCRT是一款支持[SSH](http://baike.baidu.com/view/16184.htm" \t "http://baike.baidu.com/_blank)（Secure Shell)的[终端仿真](http://baike.baidu.com/view/190233.htm" \t "http://baike.baidu.com/_blank)程序，能够在window下登录Linux服务器主机进行命令操作的一款软件[37]。SecureCRT通过使用内含的VCP命令行程序可以进行[加密](http://baike.baidu.com/view/40927.htm" \t "_blank)文件的传输。有流行CRT Telnet客户机的所有特点，包括：自动注册、对不同主机保持不同的特性、打印功能、颜色设置等等。SecureCRT的SSH[协议支持](http://baike.baidu.com/view/63557.htm" \t "_blank)DES,3DES和RC4密码和密码与[RSA](http://baike.baidu.com/view/7520.htm" \t "_blank)鉴别。

通过使用SecureCRT对VPS节点编译比特币查询服务器上的程序，使用已经购买的阿里云VPS服务后，进入SecureCRT操作界面如图4-1所示。

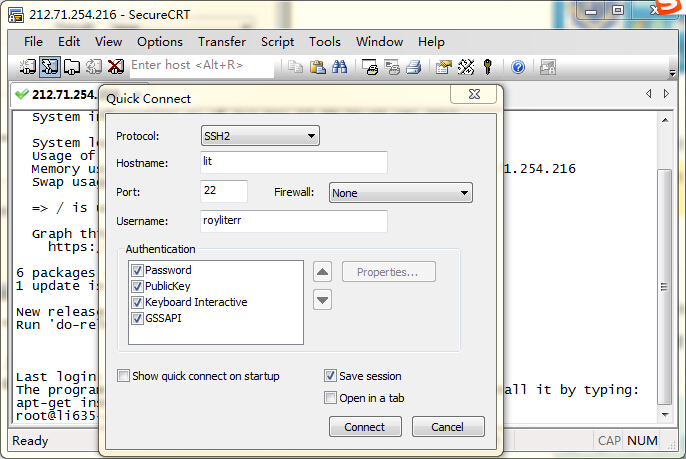
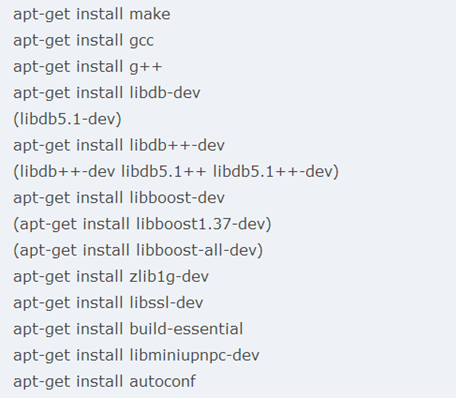


图 4-1 SecureCRT操作界面

VPS节点编译是监控系统的基础部分，需要建立一个比特币的完全节点，在VPS（ubuntu）上编译比特币的源码。具体过程如下：

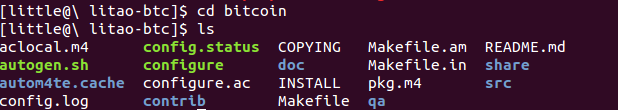
1. 在linux的系统下编译代码，需要一些依赖库，首先进行进行下列库的安装；



(2) 下载比特币源码；



(3) 阅读比特币源码中 doc/build-unix.md，根据提示，开始编译；



(4) 编译成功，显示节点可使用；

(5) 配置bitcoin.conf文件：

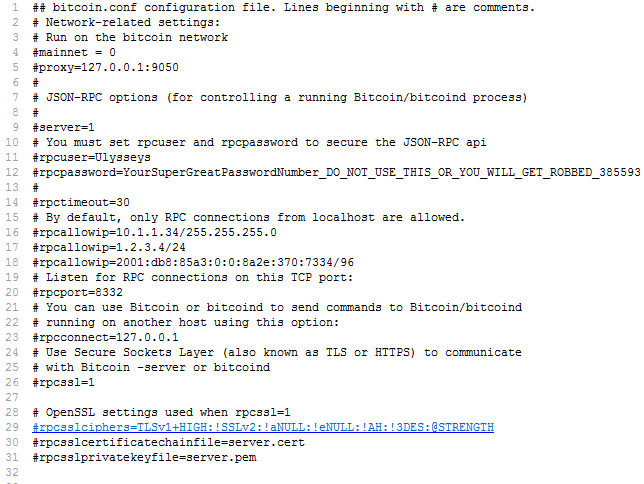


图 4-2 Bitcoin.conf 配置

比特币系统配置参数说明：

#proxy=127.0.0.1:9050 通过一个 Socks5 代理服务器连接

# server=1告知Bitcoin-QT 接受 JSON-RPC 命令

# RPCuser和RPC password 设置rpcuser和rpcpassword 确保JSON-RPC的安全

#rpcport=8332 在8332端口监听 RPC连接

#rpcallowip=10.1.1.3 为安全考虑默认仅允许来自本机的 RPC 连接。

#rpcconnect=127.0.0.1 其它主机远程运行的 Bitcoin/bitcoind 客户端

#rpcssl=0 不使用ssl套接层，在本地调用为效率考虑不用加密

#rpctimeout=30 客户端在HTTP连接建立后，等待多少秒以完成一个RPC HTTP请求

## 4.2 网络数据包捕获模块设计与实现

### 4.2.1 设计流程

由前面的系统需求分析可以知道，数据包捕获模块需要实现以下功能：设置过滤条件在以太网中捕获含有bitcoin 协议的数据包。数据包捕获流程如下图所示：

将消息存入缓存队列

tcp 重组

设置捕获数据包回调函数

设置网卡模式

图 4-3 数据包捕获流程图

(1) 设置网卡模式：网卡有四种接收模式：广播模式、组播模式、直接模式、混杂模式；本方案需要接收所有的数据帧，故将libnids默认网卡工作模式设置为混杂模式(Promiscous Mode)，工作网卡设置为1[38]。

(2) 设置捕包回调函数nids\_register\_tcp (bitcoin\_ protocol\_callback)；调用bitcoin\_ protocol\_callback()方法对对TCP连接的比特币交易信息进行过滤、分析；

(3) 对接收到的信息进行重组[39]；

(4) 当重组完毕或者TCP连接关闭或者重置[40]，将重组接收的消息送入缓存队列。

### 4.2.2 实现流程

由于要处理的交易信息量可能非常大，如果交易判断工作在libnids回调函数中进行，会引起丢包问题[40]。丢包会引起数据不完整，出现新交易漏判、误判。这个模块使用多线程编程，设置消息缓存队列，把过滤的数据包送入其中，把捕获的数据包和新交易判断过程分开处理，保证数据包捕获的性能[41]。其数据处理流程图如图4-2 所示。

nids\_init( )

nids\_register\_tcp ( bitcoin\_ protocol\_callback)

开 始

若为NIDS\_JUST\_EST

表示建立tcp连接

若为NIDS\_DATA

表示有数据到达

若为NIDS\_CLOSE

/RST 时，表示tcp连接关闭或重置

判断nids\_state 的状态

A\_tcp->saddr==moned\_addr ||a\_tcp->daddr==moned\_addr&& a\_tcp->dest == 8333

(a\_tcp->client.count\_new)

|| (a\_tcp->server.count\_new)

Tcp 连接

客户端数据

Tcp 连接

服务端数据

送入TCP数据重组链表TCP\_data

将重组后的消息中交易信息送入消息缓存队列等待process\_data()线程处理

结 束

图 4-4 数据包捕获模块数据处理流程图

1. libnids 进行初始化；设置网卡工作模式[42]；
2. 设置捕包回调函数nids\_register\_tcp ( bitcoin\_ protocol\_callback)，bitcoin\_ protocol\_callback()方法对对TCP连接的比特币交易信息进行过滤；
3. 判断连接状态nids\_state，若为NIDS\_JUST\_EST表示建立TCP连接，设置过滤条件判断条件判断是否是与被监控的ip建立的TCP连接，是否是比特币程序端口号。若为NIDS\_DATA表示有数据到达；若为NIDS\_CLOSE 或者NIDS\_RST，表示TCP 链接关闭或重置；
4. count\_new判断数据来源并通过送入重组队列 ；这是由于比特币交易信息的ip包长度可能超过1518，所以要先进行重组才能把完整的一条交易信息送入缓存队列；
5. 当重组完毕或者TCP连接关闭或者重置，将重组接收的消息送入缓存队列。

## 4.3 协议分析设计

本模块协议分析主要是针对比特币交易相关信息（tx和inv）进行解析[43]，故本节主要针对比特币交易做相对应的数据结构设计，解析流程设计，以及实现解析。

### 4.3.1 设计流程

首先要设计数据结构来解析比特币交易相关数据，根据第2章比特币交易构成叙述的比特币交易信息tx和向量清单inv的构成，为保证与源代码数据的一致性，参考比特币bitcoin-master中的core.h和protocol.h的源码，数据结构设计如图4-5所示：



图4-5 数据结构设计

协议分析的数据流程图如下图4-6所示。

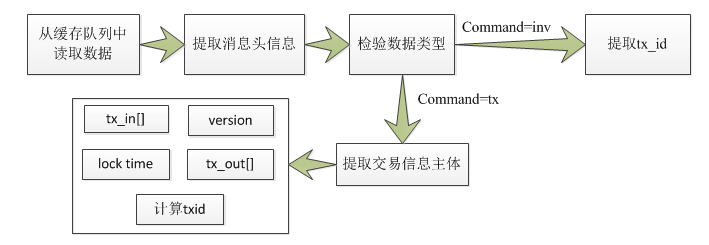


图 4-6 数据处理流程图

1. 首先，缓存队列中读取数据，开始进行解析；
2. 提取消息头信息，验证该消息是否为比特币消息；
3. 检验数据类型，如果command=inv，调用parse\_inv()方法解析比特币交易消息；

如果command=tx调用parse\_tx()方法解析比特币交易消息，包括交易的消息头，消息体，消息体里各部分具体交易信息，包括输入输出地址等信息；在解析数据内容的同时并检验各部分是否有效，若为无效数据，则判定该交易无效。

### 4.3.2 实现流程

此模块实现了对比特币相关交易数据的解析；包括txid、tx\_in[]、tx\_out[]等数据，获取其中关键的比特币地址和交易id等信息。其开发环境是windows平台，运行环境是visio stidio 2012系统平台，开发语言为C。

依据上节设计流程可得，该模块的整体程序设计图如图4-7所示；

(1) 首先，利用btc\_MSG类从缓存队列的获取对象实例，进行解析；

(2) 提取消息头信息msg.hdr；验证magic是否等于F9BEB4D9；

(3) 检验数据类型，如果command=inv，调用parse\_inv()方法解析比特币交易消息；

如果command=tx调用parse\_tx()方法解析比特币交易消息，包括交易的消息头，消息体，消息体里各部分具体交易信息，包括输入输出地址等信息；在解析数据内容的同时并检验各部分是否有效。

其中子程序parse\_inv的算法流程图如图4-8所示；子程序parse\_tx的算法流程图如图4-9所示；

图 4-7 程序流程图



图 4-8 parse\_inv的算法流程图

inv消息协议解析的步骤为：

* 判断command指令是否为“inv”；
* 清点清单数目count是否为零，不为零则读入下面的清单数据；
* Type为1时才是与交易有关的数据，提取交易数据，清单数量count减一，当count不为零，循环解析数据；
* Type为不为1时，清单数量count减一，循环解析数据。

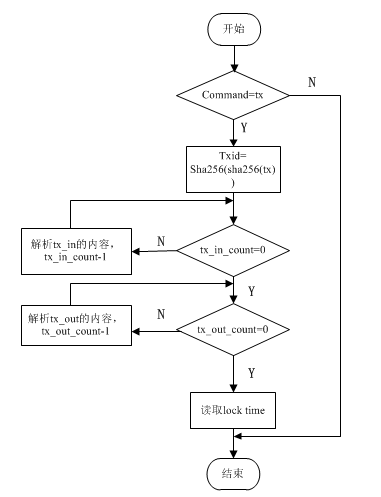


图 4-9 parse\_tx算法流程图

tx消息解析的步骤为：

* 判断command指令是否为“tx”；
* 首先计算消息的txid，从消息中获取交易的输入数目tx\_in\_count，循环读取交易输入的比特币地址信息，直到tx\_in\_count为零，交易输入解析完毕；
* 获取交易的输出数目tx\_out\_count，循环读取交易输出的比特币地址信息、对应的交易值，直到tx\_out\_count为零，交易输出解析完毕；

将最后新交易需要记录的数据内容详尽解析并提供调用接口；

程序运行的执行结果如图4-10所示



图 4-10 协议解析程序程序运行结果

## 4.4 区分新交易设计与实现

### 4.4.1 设计流程

此模块是系统功能实现的核心部分，本系统只提取被监控节点所发起的交易，才能有效的将IP 地址与比特币地址对应。该模块的程序设计过程有以下两个难点：

(1) 数据存储问题

理论上需要监控被监控节点网络接口流经的所有交易信息，保证对每一个比特币交易数据的判别。等待确认交易信息的量不大，与每个区块产生的新交易数据量基本一致，从blockchain.info搜集2015年1月5日的节点交易数量时间图，可以得到每个块大概发生1000笔交易，记录每个交易ID需要32字节，每10分钟一共需要32K内存，这对存储来说不是很大问题；但由于网络中还会有旧的区块交易信息在传播，并且比特币块链中交易信息量非常巨大（2015年总数据量已经超过20G），如果监控节点流经大量的已确认的交易信息，在最坏的情况的监控程序内存无法记录全部流经节点的交易数据，也就无法判断交易是不是新产生的[45]。

解决方法：比特币客户端提供了对交易的查询功能，通过其提供的RPC接口，当收到一笔交易信息时，程序通过json-RPC的交易查询指令getrawtransaction对一笔交易进行判断是不是已经被块链确认或者已经在mempool中的交易信息，就可以判断这是否是一笔新交易。 另外内存池中的新交易信息记录会不断膨胀，需要定期刷新（使用循环buffer）该记录队列。”tx”交易信息数据量较大且同一交易的“inv”消息的广播要先于“tx”消息，优先处理“inv”消息。

(2) 信息实时正确处理问题

比特币消息是在全网广播，一般10到60秒就会传送至全部节点，只有完全节点才有可能保存完整的内存池信息，故需要构架一个VPS完全节点作为查询服务器提供新交易判定的查询一部分[46]。

由于整个网络拓扑环境复杂，各个节点在接收广播的信息时间快慢会有差别，从整个设计过程和比特币网络信息传输协议来看，设计与信息的传输与响应的速度密切相关，对实验中信息处理时间过程进行分析：

设查询服务器更新被监控节点的新交易信息的时间为，查询服务器更新被监控节点转发的交易的时间是，监控节点查询时间，分别针对被监控节点的新交易和转发的交易鉴别过程中可能出现的情况进行分析。



(1) < ，查询时交易未被服务器更新，鉴定为新交易； //正确



(2) > ，查询时交易已经被服务器内存池更新，鉴定为转发的交易；



//误判

(3) < ，查询时交易已经被服务器内存池更新，鉴定为新交易；



//误判

(4) > ， 查询时交易已经被服务器内存池更新，鉴定为转发的交易；



//正确

漏判和误判的两种情况：

(1) 查询服务器更新新交易时间先于查询时间，将新交易错判为非新交易\_\_漏判；

(2) 查询服务器更新转发交易时间落后于查询时间，将转发信息错断为新交易\_\_误判；

设计的性能关键归结为三点： (1) 查询费时要短、及时； (2) 服务器更新交易信息速度要快、全； (3) 查询时间要先于新交易更新的时间。

节点在处理“tx”消息之前会先通过“inv”进行通信，并且处理inv时不会更新内存池信息，这将有助于解决在得到查询结果前信息被更新导致误判问题。使用处理性能高的计算机，保证消息解析的速度能跟上比特币网络对交易实时处理的速度。

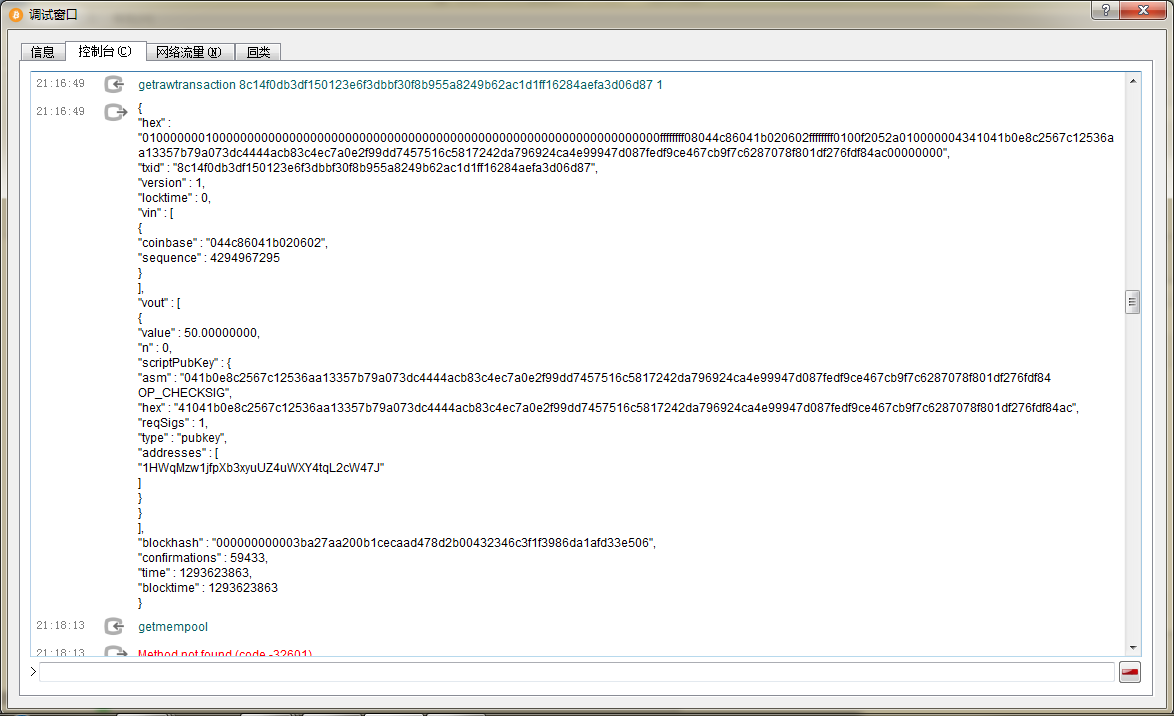
此外，节点向外发送交易信息时一般都向几个节点同时发送，inv必有重叠部分，故设计一个新交易信息容器newtxids外，另设一个循环buffer vtxid\_rct存储最近一段时间内接收到的txid，信息先通过与新交易容器数据检验，再通过对最近接受的消息比较，就能先过滤一些亢余的数据包，不用每个包都向服务器发送RPC命令。

Bitcoind提供的RPC调用方法API中，可以获取“tx”信息的调用方法主要有以下：

表4-1 tx信息调用方法

|  |  |  |  |
| --- | --- | --- | --- |
| **命 令** | **参 数** | **描 述** | **unlocked** |
| getrawmempool |  | 返回内存池中全部的交易信息的id——txids，这个方法可以知道当前时刻unconfirmed txs的情况 | N |
| getrawtransaction | <txid> [verbose=0] | verbose参数为0返回hex格式数据，为1返回json格式的交易数据信息，查找失败返回error | N |
| gettransaction | <txid> | 返回钱包里id为<txid>的交易信息，输出json数据 | N |
| listtransactions | [account] [count=10] [from=0] | 返回一定数目交易信息，如果不足，则列出全部交易信息 |  |

新交易判定必须是实时的，mempool信息时刻变化，mempool提交当前内存池信息，信息量大，不适合本方案。选用getrawtransaction方法，查询目标交易信息是否在blockchain或者mempool中。需要在bitcoind startup settings里设置 txindex=1而非默认值，才能确保getrawtransactions能查找全部的交易信息。在比特币客户端调试窗口使用查询命令“getrawtransanction”检验RPC查询功能如图4-11所示：



**图 4-11** getrawtransanction指令查询结果

### 4.4.1 实现流程

bitcoin使用[HTTP JSON-RPC](http://json-rpc.org/wiki/specification|)，本程序基于c++语言开发，选用JSon.Net开发包。待bitcoind配置和数据同步完成后，数据捕获程序进行实时捕包，获取过滤比特币交易信息（MSG\_inv或者tx）后放入缓存队列vrecv，新交易判定线程process\_data()实时从消息缓存队列vrecv中提取消息，判定捕获的交易是否是新交易。该程序设计流程图如4-12所示。



图 4-12 新交易判定Process\_data线程流程

过程说明：

1. 判断比特币缓存队列（rvec）是否为空，如果不是，读取信息对象实例；
2. 解析比特币消息头Parse\_bitcoin()；
3. 判断信息类型，分别对inv和tx类型消息进行处理，其它消息类型忽略。

如果是inv消息类型，则进行parse\_inv()数据解析，从而获取MSG\_inv的txid；

如果是tx消息类型，则求该交易的哈希值get\_txhash()；

1. 检验消息是否存在接受的消息队列里，将txid轮询与缓存队列buffer vtxid\_rct一一比较；如果不存在，进行下一步骤；如果已经存在，返回删除此消息，返回程序，重新开始；
2. 检验新交易是否已经存在新交易记录中，将txid轮询与缓存队列newtxids一一比较；

如果不存在，进行下一步骤；如果已经存在，返回删除此消息，返回程序，重新开始；

1. 调用RPC指令getrawtransaction方法查询交易是否属于新的交易；如果不存在，进行下一步骤；如果已经存在，返回删除此消息，返回程序，重新开始；
2. 将新交易记录到newtxids 容器中；
3. 程序结束。

程序中Json RPC调用的方法关键代码如下图所示：

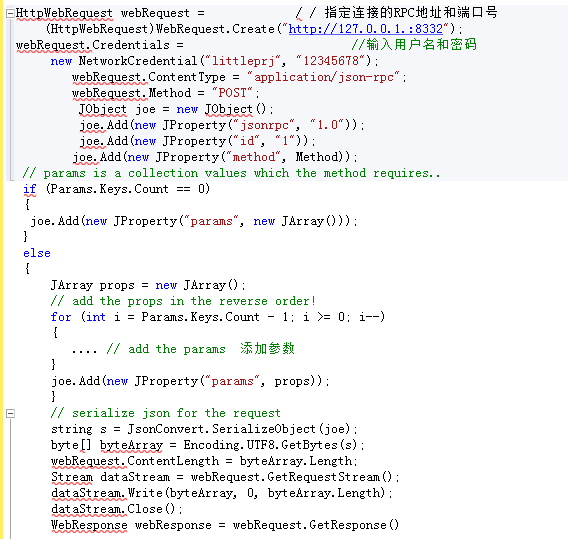


图4-13 Json RPC调用实现代

## 4.5 存储子模块设计与实现

### 4.5.1 设计流程

设计采用MySQL作为后台数据库[49]，将比特币新交易监测信息存储于数据库中，通过调用数据库的命令进行信息的存储、查询与修改。根据系统需要记录的数据情况，设计4个table用于存储查询数据：

•mon\_node\_tb表 存储网络上的监控节点状态基本信息。

•bttx\_tb表 存储比特币新交易的部分信息。

•bttx\_txoutmsg\_tb表 存储新交易输出的基本信息。

•bttx\_txin\_msg\_tb表 存储新交易输入的基本信息。

因为每个交易信息的输出地址和输入地址都可能是多个的，所以对这部分内容单独建表方便查询；mon\_node\_tb记录网络监控节点的基本信息。其bttx\_tb、bttx\_txin\_tb 和bttx\_txout\_tb表的信息如表4-2、表4-3和表4-4所示，表关系图如图4-14所示。

表 4-2 bttx\_tb表

|  |  |  |  |
| --- | --- | --- | --- |
| **字 段** | **类 型** | **长 度** | **说 明** |
| txid | char | 64 | 主键not NULL |
| ipaddr | varchar | 50 | not NULL default |
| time | varchar | 20 | not NULL default |

表 4-3 bttx\_txin\_tb 表

|  |  |  |  |
| --- | --- | --- | --- |
| **字 段** | **类 型** | **长 度** | **说 明** |
| txid | char | 64 | 主键 not NULL |
| txid\_pre | char | 64 | not NULL default |
| tx\_index | int | 4 | not NULL default |
| txin\_addr | Varchar | 150 | not NULL default |

表 4-4 bttx\_txout\_tb表

|  |  |  |  |
| --- | --- | --- | --- |
| **字 段** | **类 型** | **长 度** | **说 明** |
| txid | char | 64 | 主键not NULL |
| txout\_addr | varchar | 100 | not NULL default |
| txout\_values | varchar | 20 | not NULL default |

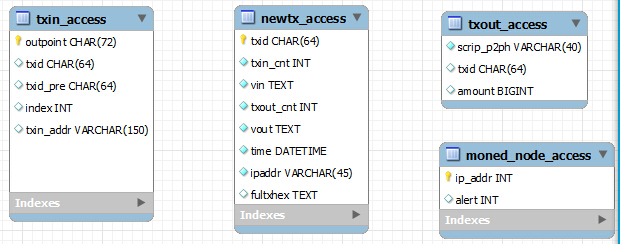


图 4-14 数据表关系图

### 4.5.2 实现流程

数据库实现过程如下：

(1) 下载并安装最新的mysql数据库；

(2) 使用mysql提供的C API，连接数据库方法如下：

mysql\_real\_connect(conn,"localhost", "root", "12345678", NULL, 0, NULL, 0) == NULL);

创建一个数据库btc\_snif\_db方法如下：mysql\_query(conn, "create database btc\_snif\_db:"));

为读取方便，数据块中的数据存储是已经转变成hex形式的数据，创建数据表bttx\_db、bttx\_txinmsg\_db、bttx\_txoutmsg\_db的方法如下：

mysql\_query(con, "create table bttx\_db(txid char(64), ip\_addr varchar(50), time varchar(50), txin\_cnt int, txout\_cnt int)");

mysql\_query(con, "create table bttx\_txinmsg\_db(txid char(64), txid\_pre char(64), txin\_index int, txin\_addr varchar(200))");

mysql\_query(con, "create table bttx\_txoutmsg\_db(txid char(64), txout\_addr char(40),txout\_values bigint)")；

在c程序中插入记录数据，不能直接插入变量，通过以下方法实现插入变量：

sprintf(tmp,"insert into bttx\_txinmsg\_db2(txid, txid\_pre, txin\_index, txin\_addr) values('%s', '%s', %d, '%s')", tmp1, tmp2,ptx\_msg->tx\_in[count].index,tmp3);

rt = mysql\_real\_query(con, tmp, strlen(tmp));

查询时通过mysql命令行＂use btc\_snif\_db＂选用存储的数据库，再使用命令：

＂select （查询条件） from （查询条件）＂按条件查询即可获取相关查询信息[50]。

## 4.6 本章小结

本章首先实现了本系统所需要的准备阶段-----编译比特币节点。接着根据上一章所述的系统模块设计图分模块进行设计与实现，网络数据包捕获模块选取windows平台libnids开发包捕获含有Bitcoin协议的数据包；协议分析模块设计程序对含有tx（交易）的协议按字段进行解析，得出每笔交易的输入、输出、交易金额信息；区分新交易模块实现鉴定所捕获的交易是否属于被监控节点发起的交易功能；存储模块针对区分新交易模块得到的新交易的IP 地址与比特币地址数据设计相应的数据库存储。

# 第5章 系统测试

前面的章节设计和实现了该方案，本章将对该方案进行功能测试和性能测试。这里分成三个部分来对系统功能（VPS查询功能、捕包功能、新交易捕捉功能）的实现情况进行测试。

## 5.1 系统功能和性能测试

这一节主要对该系统进行功能测试和性能测试。测试内容如下：

1. VPS完全节点查询服器查询功能测试
2. 数据包捕获程序功能及性能测试
3. 新交易捕获功能及性能测试

测试环境及部署说明：本实验是在深圳大学的一网络出口端设置一个监控点监控内部网络的比特币交易信息。使用内部网络一主机发送比特币交易作为被监控对象，网内另一主机安装监控软件，检验监控结果。VPS选用阿里云服务提供的某一远端服务器作为查询服务器，借助blockchain.info查询相关信息。本实验测试环境网络拓扑如图5-1所示。



图 5-1 实验环境网络拓扑图

本测试实验所用设备配置情况如表5-1所示，主要包含被监控节点、监控节点以及查询服务节点的配置情况。

表5-1 实验环境配置信息

|  |  |  |  |
| --- | --- | --- | --- |
| 设备及工具 | Ip地址 | 基本配置 | 用途 |
| 神舟k580s  （监控主机A） | 210.39.1.8  //内部网络地址:  192.168.253.12 | Cpu：i5 3210  内存：金士顿8g  操作系统：window7 64 | 作为监控软件搭载节点，抓取数据包，并进行新交易捕捉、存储等工作 |
| 联想567  （被监控主机B） | 210.39.1.8  //内部网络地址:  192.168.253.4 | Cpu:Intel G1610  内存：2.0Gb  操作系统:windows7 32 | 作为实验被监测的测试节点，发送比特币交易信息，被监控的对象。 |
| 阿里云某vps服务器 | 212.71.254.216 | Cpu: 未知  内存：未知  操作系统：Ubuntu | 比特币完全节点，查询服务器，提供比特币新交易查询服务 |
| blockchain.info | 190.93.243.195 | 未知 | 辅助分析工具，可查询最近区块的交易信息 |

1. VPS完全节点查询服器查询功能测试

从blockchain.info上查询1条比特币块链已经存在的txid，一条存在于mempool中但还未被确认的txid，另外创建一个不存在的txid，。使用RPC指令发送3条命令：

getrawtransaction 7081818a01e32c9dcde602b18f90c56dc3de279210a4f04e12868de61593a9fd 0

getrawtransaction fa9b09609d1c39528415db0ddfb6cb17d0c83b71c038022f585e988c245682b7 0

getrawtransaction fffb09609d1c39528415db0ddfb6cb17d0c83b71c038022f585e988c245682b7 0

查询交易信息，结果如图5-2所示



图 5-2 VPS完全节点查询服务器RPC查询功能测试

结果分析：

第一个交易是一条已经被确认的交易，指令getrawtransaction获取了其交易的hex信息；

第二个交易位于内存池但未被确认的交易，同样也能查询到该交易信息；

第三个是一条不存在真的交易，返回错误代码（code -5）。

结果表明，该VPS节点查询服务器的RPC调用功能正常，能够正确识别查询指令getrawtransaction，查询内存池或者区块链上的交易时均能返回相应的交易信息，输入不存在的交易则查询失败。

1. 数据包捕获程序功能及性能测试

首先不设对交易是否是新交易设置捕捉条件限制，检验捕捉比特币交易数据包功能，在捕包功能正常条件下再进行新交易的捕捉，排除软件捕捉功能不正常导致捕捉不到新交易可能性。启动比特币客户端和捕包程序，结果如图5-3所示：

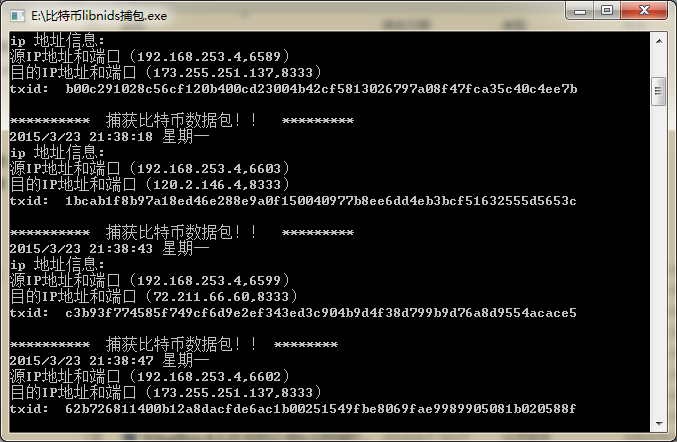


图 5-3 捕获数据报结果

结果分析：在发送新交易实验前，比特币客户端程序也在和其它比特币网络节点进行比特币交易数据的传输，并且测试实验环境下程序捕包功能正常，可进行下一步新交易的捕捉实验。

1. 新交易捕获功能及性能测试

为测试不同网络流量环境下捕捉新交易的性能，实验利用比特币块链数据未更新完会有大量交易数据传输的特点，分别对有少量交易数据和有大量交易数据的网络环境下新交易监控的质量进行检查。

表 5-2 新交易监控实验测试内容

|  |  |  |  |
| --- | --- | --- | --- |
| **检测项目** | **检测方法** | **检测目的** | **检测内容** |
| 新交易的数量 | 在一段时间内输出一定量的交易，记录捕捉结果 | 捕捉功能是否正常 | 捕捉到的新交易的数量 |
| 新交易输入的数量txin数目、txout数目 | 对新交易数据捕捉后，配合blockchain.info提供查询的交易数据信息，检验不同输入输出的比特币地址解析的正确性 | 解析功能是否正常 | 解析输出内容 |
| 网络流量和交易数据量 | 在另一台主机上运行区块高度落后较多的比特币客户端，选择开启更新区块链或不开启 | 在网络中有大量交易数据条件下捕捉新交易质量 | 网络流量和交易数量对监控程序的干扰 |

在vps节点工作足够时间后，开启监控主机A的btc\_tx\_snif程序，而后在被监控主机B使用比特币客户端创建交易进行简单的功能测试，可在被监控测试主机B上查看实验环境下比特币网络的基本信息如图5-4所示：

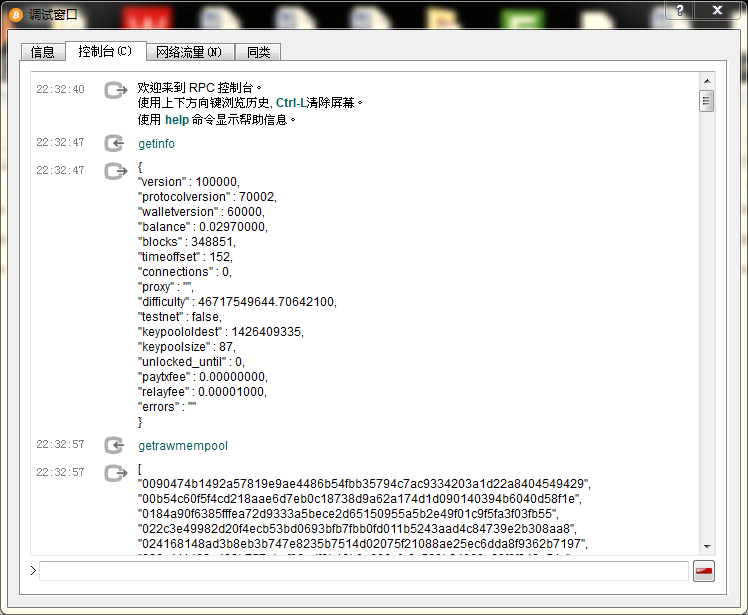


图 5-4 比特币网络的基本信息

不同网络流量环境检测方法：（1）先不开启干扰主机的比特币客户端，测试新交易捕捉性能，此时网络流量不大，需要检测的交易数量也小，网络流量环境如图5-5所示；（2）开启干扰主机的还未更新完块链的比特币客户端程序，再进行新交易监控测试，此时干扰主机网络流量环境如图5-6所示；

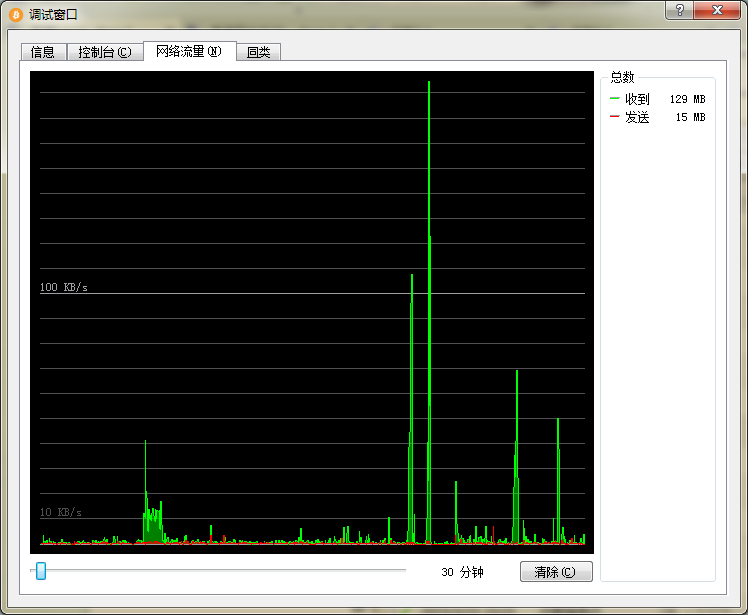


图 5-5 被监控主机（块链已更新完全）网络流量图

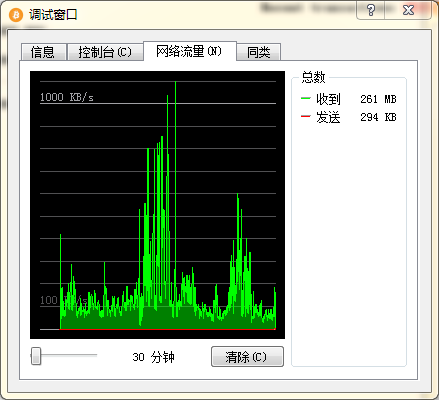


图 5-6 干扰主机（大量更新区块数据）网络流量图

运行被监控主机客户端，创建实验测试的比特币交易如图5-7所示：

分别创建不同数量交易输入与输出的交易，测试一段时间内比特币监控的效果与协议解析的准确性。为验证监控程序监控到的数据包是否出现误判、漏判的情况，可通过查看被监控主机客户端上的交易记录（如图5-8所示）来判断是否出现新交易信息判断错误。

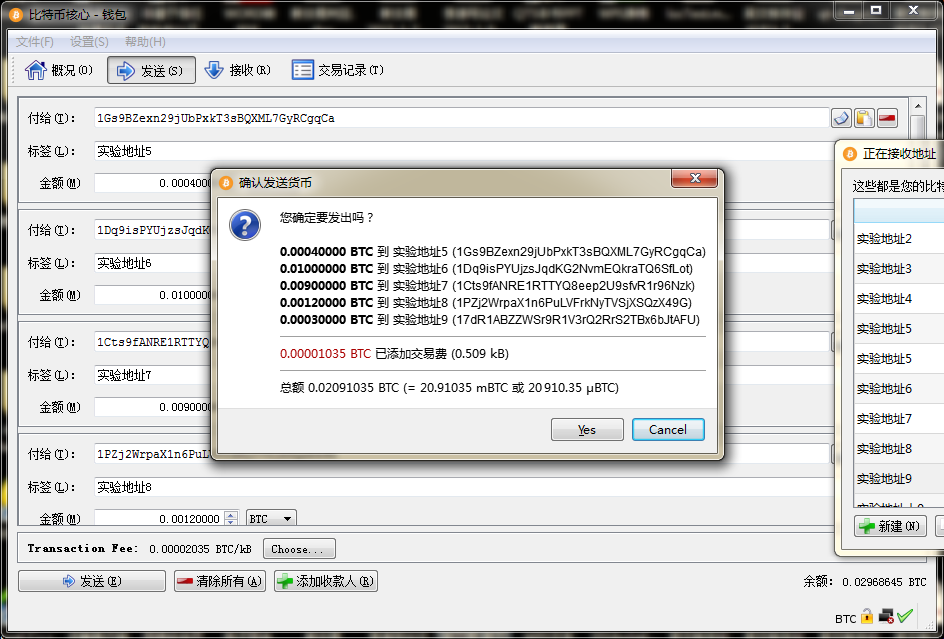


图5-7 实验测试被监控主机创建的交易记录

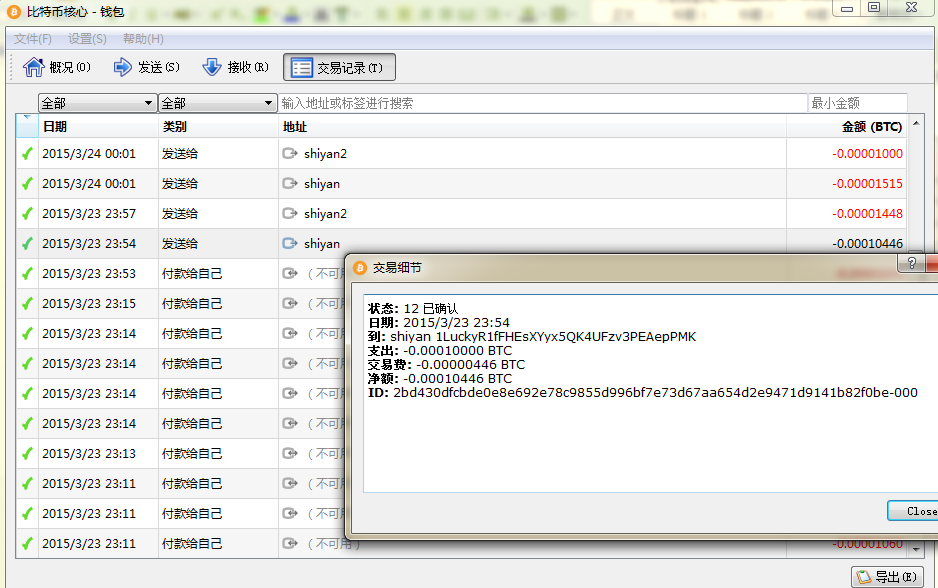


图5-8 实验测试被监控主机确认的交易记录

实验测试结果如图5-9所示：

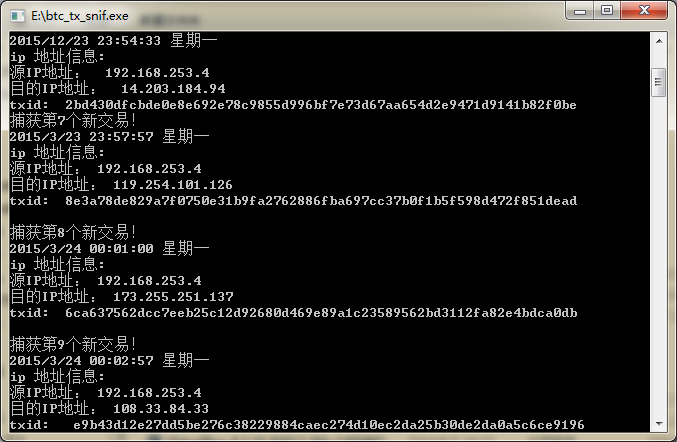


图 5-9 新交易监控程序捕捉界面

如图5-9所示，btc\_tx\_snif新交易监控程序在实验开始后，能够捕获被监控节点192.168.253.4的新交易，最后捕包的结果如表5-3与5-4所示。

表5-3 无大量交易信息网络环境下的新交易捕获结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 一定时间内连续创建的交易数目（个/分钟） | 新交易捕获数量 | 非新交易捕捉数量 | 新交易捕获率 | 误判率 |
| 1 | 1 | 0 | 100% | 0 |
| 10 | 10 | 0 | 100% | 0 |
| 20 | 20 | 0 | 100% | 0 |

表5-4 有大量交易信息网络环境下的新交易捕获结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 一定时间内连续创建的交易数目（个/分钟） | 新交易捕获数量 | 非新交易捕获数量 | 新交易捕获率 | 误判率 |
| 1 | 1 | 0 | 100% | 0 |
| 10 | 9 | 0 | 90% | 0 |
| 20 | 17 | 0 | 85% | 0 |

实验结果表明，在没有大量交易数据的网络环境下，监控程序对新交易捕获成功，没有出现误判、漏判交易信息情况。，在有大量交易数据的网络环境下，监控程序对新交易捕获成功，但捕获率并非100%，没有出现误判情况，但出现信息漏判情况，捕包程序的检测率降低。

通过对数据的分析处理和存储，最终本实验获得的比特币监控数据的效果如图5-10所示。使用MySQL的查询命令：select (查询条件) from (查询条件)，对某一查询的新交易的id号＂2bd430dfcbde0e8e692e78c9855d996bf7e73d67aa654d2e9471d9141b82f0be＂进行查询，获取相应的监控信息，包括：新交易对应的ip地址、比特币地址、交易的金额、时间等信息。

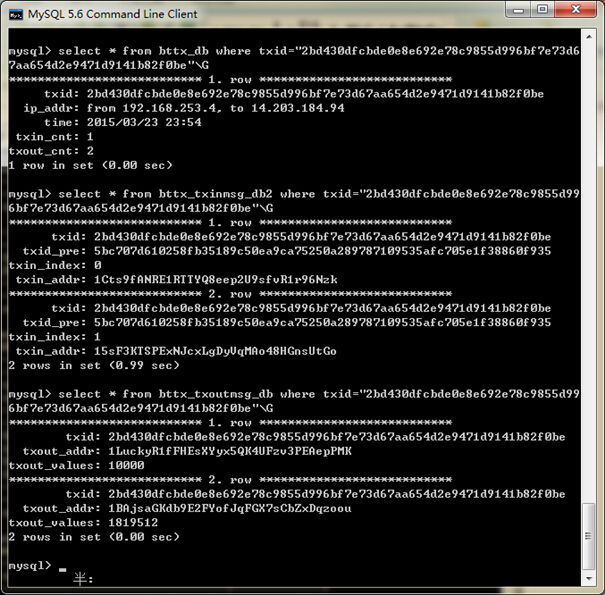


图 5-10 本监控实验MySQL查询获得的结果

## 5.2 结果评价与方案设计总结

从测试结果来看本设计基本完成对比特币新交易数据的监控，达成设计预期的目标。从测试的实验环境来看，该测试环境是一个学校的内部网络，监控的交易的信息量不大，对放入大型网络中的性能还未进行测试，目前完成的是功能上的研究，证明方案可行性，其它测试研究有待日后进行。

由于整个网络拓扑环境复杂，各个节点在接收广播的信息时间快慢会有差别，从整个设计过程和比特币网络信息传输协议来看，设计的理论前提是信息的传输与响应的速度密切相关

除了论文里已经提到的多线程与优先处理inv消息方法（服务器在处理inv时并不会更新mempool中的交易）外，还有以下几个方法可以提高系统性能，提高捕捉准确率：

(1) 使监控节点成为完全节点，本地调用速度要明显快于远程过程调用RPC云端服务器。vps完全节点越大越好，连接的网络节点越多，更新mempool信息将更快。

(2) 在不出现单点故障情况下，通过网关对交易信息的拦截，可以使查询服务器有时间更新自己的mempool信息，并且可以让查询信息先于该交易广播发送出去，能有效解决误判、漏判的可能性。

(3) 在监控节点设计一个内存池，块链上的交易信息可本地调用块链数据进行查询，mempool信息可以通过对流入流出的交易信息进行过滤筛选出unconfirmed txs信息，添加到自己设计的内存池中，在捕包质量高的情况下，监控节点的mempool信息不会落后于被监控节点的mempool信息，新交易被漏判、转发交易被误判的可能性就比较小。但要存储的信息量要大大超过于本方案设计，可行性尚待研究。

## 5.3 本章小结

本章在前面几章实现系统的基础上，对该方案进行了功能性测试。实验分成了两部分对代理功能的实现情况进行测试，证明了方案的可行性。最后总结了设计方案可能存在的一些问题和可改进的方法，方便日后明确研究方向。

# 第6章 总结与展望

本系统针对比特币网络的匿名问题，在比特币节点网络安装监控软件，对网络上传播的比特币交易信息进行实时监控，捕捉交易信息的发送源。根据监控软件解析的交易信息内容将特定的IP地址与比特币地址信息进行绑定，达到对某一比特币交易信息的实际拥有人进行定位的效果。该设计为监管比特币非法活动提供了一个追踪监控的方法，从而达到打击犯罪、规范比特币市场的目的。

## 6.1 论文总结

本文旨在实现整个比特币新交易监控系统功能，从全局统筹考虑出发，分别完成各个模块的功能实现从而实现整个监控系统的功能。课题的设计、实现过程中完成了如下几个部分的工作：

(1) 深入研究比特币系统基本原理，并根据比特币的匿名性存在的缺点，提出系统设计方案以使比特币网络能更加透明；

(2) 在分析系统需求的前提下，进行系统整体实现方案的设计构架。包括：数据捕获模块设计、协议解析模块设计、区分新交易模块设计以及存储模块设计。

(3) 分别针对不同模块进行详细的设计。网络数据包捕获模块选用了windows平台下免费的抓包工具libnids开发包，根据其所对应的端口号8333来设置过滤条件实现在以太网中捕获含有bitcoin 协议的数据包，然后将其放入到缓存队列；协议分析模块取出缓存队列数据，根据比特币交易的组成对tx和inv消息进行数据结构设计，从而实现对含有tx（交易）和inv(向量清单)的协议按字段进行解析，得出每笔交易的输入tx\_in[]、输出tx\_out[]、txid、交易金额等信息。

区分新交易模块系统功能实现的核心部分，因为本系统只提取被监控节点所发起的交易，对监控节点转发的交易不作处理。该模块的程序设计过程面临两个难点：一是数据存储问题，利用循环buffer来防止内存池中的新交易信息记录不断膨胀。其次，采用比特币客户端提供的RPC接口的查询功能来判断所接收的交易是否为新交易；此外，由于“tx”交易信息数据量较大且同一交易的“inv”消息的广播要先于“tx”消息，优先处理“inv”消息。另一难点则是，信息实时处理问题，本系统最终通过使用SecureCRT完成对VPS节点服务器的编译，作为新交易判定的查询服务器，可以有效的降低新交易的误判率； 存储子模块采用MySQL作为后台数据库，将新交易对应的ip地址、比特币地址、交易的金额、时间等记录存储。

(4) 对系统进行功能测试和性能测试。测试内容包括：①VPS完全节点查询服器查询功能测试 ；②数据包捕获程序功能及性能测试；③新交易捕获功能及性能测试。测试选择几个测试用例进行实验，实验结果表明：系统能够实现数据包捕获功能、VPS完全节点的交易信息查询功能以及对被监控节点发起的新交易信息捕获的功能。实现将节点所在IP地址与比特币地址进行关联，即对比特币地址与其所拥有人进行精准定位，从而实现撤销用户匿名性的目的。这为监管比特币网络提供了一种解决方案。

(5) 最后对实验存在的不足进行总结，并提出一些可能的解决方法。

## 6.2 后续工作展望

本论文在系统需求分析的基础上开发、设计实现了比特币新交易的监控系统。由于实验条件限制，无法对海量用户请求进行数据测试，这里只是在理论的基础上实现了论文中设计的功能。而且对于本文中实现的系统，还有一些值得改进、提升的空间。结合本系统实现以及对于比特币产业链的了解提出以下几点可改进的想法。

(1) 本监控系统实现了比特币地址与IP地址的一个基本的对应。但是当用户使用VPN 连接代理时，提取到的IP地址与比特币地址很难实现对应。设计何种方法实现抵抗使用代理隐藏ip地址信息值得进一步研究；

(2) 本监控系统能够实现在被监控节点捕获数据包的功能，但是往往大型网络的出口端的数据量是惊人的，如何高效无误的抓到含有Bitcoin交易协议的数据包，这是值得考虑的；

(3) 本系统实现了基本的所需功能，还有很多可以完善的地方，界面和功能都可以开发的更加友好，如：实现远程查询和控制功能、与服务器通信以及设置被监控节点的开关等。

**参 考 文 献**

1. S Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System. 2008, [DB/OL]. Available: <http://bitcoin.org/bitcoin.pdf>
2. Fergal Reid and Harrigan Martin. An analysis of anonymity in the Bitcoin system[C]//In Proceedings of the 3rd IEEE International Conference on Privacy, Security, Risk and Trust and on Social Computing,. New York: Springer, 2011: 1318~1326.
3. D Chaum. Untraceable electronic mail, return addresses, and digital pseudonym[J], Communications of the ACM, 1981.24(2): 84–90.
4. White GB, Fisch EA, Pooch UW. Cooperating security managers: A peer-based intrusion detection system [J]. IEEE Network, 1996,10(1):20-23.
5. Gregory B White, Eric A Fisch, U do W Pooch. Cooperating security Managers: A peer based intrusion detection system[R].IEEE Network. April 1975.
6. N Christin, Traveling the Silk Road: A Measurement Analysis of a Large Anonymous Online Marketplace[R] , Rio de Janeiro :In Proceedings of the 22nd International World Wide Web Conference, 2013.
7. 王刚, 冯志勇. 关于比特币的风险特征、最新监管动态与政策建议[J].金融与经济,2013,5(09): 22-24.
8. Eng Keong Lua,Jon Crowcroft, Marcelo Pias, Ravi Sharma ,Steven Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes[J] . IEEE Communications Survey and Tutorial, 2004,6(3): 72-93.
9. C Decker and R Wattenhofer, Information Propagation in the Bitcoin Network ,Proc[J].13th IEEE Int’l Conf. Peer-to-Peer Computing, 2013: 1–10.
10. D Ron and A. Shamir, Quantitative Analysis of the Full Bitcoin Transaction Graph[J]. Financial Cryptography and Data Security, LNCS7859, Springer, 2013: 6–24.
11. M Ober, S Katzenbeisser, and K Hamacher, Structure and Anonymity of the Bitcoin Transaction Graph[J] .Future Internet, 2013,5(2) : 237–250.
12. T Moore and N Christin. Beware the Middleman: Empirical Analysis of Bitcoin-Exchange Risk[J]. Financial Cryptography and Data Security ,LNCS 7859, Springer, 2013: 25–33.
13. I Miers, C Garman, M Green, and A D Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin[J]. In Proceedings of the IEEE Symposium on Security and Privacy, 2013:26-29.
14. S Zhioua. Anonymity attacks on mix systems: a formal analysis[J]. In Information Hiding, ser. Lecture Notes in Computer Science, Berlin Heidelberg: Springer, 2011 : 133–147.
15. 周宏达.比特币的前世今生[J].中国金融家,2013.6(2):21-23.
16. 徐平均. 基于Libnids分布式入侵检测系统的研究与实现[D].长沙.中南大学,2012:21-42.
17. 陈刚. 基于Libnids网络入侵检测系统的设计与实现[D].长沙:湖南大学,2010.15-26.
18. 娄耀雄, 武君.比特币法律问题分析[J].北京邮电大学学报: 社会科学版.2013( 4) : 25-31.
19. [张磊](http://www.cnki.net.ezproxy.lib.szu.edu.cn/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e5%bc%a0%e7%a3%8a&code=20756740;32046565;" \t "http://www.cnki.net.ezproxy.lib.szu.edu.cn/KCMS/detail/_blank), [马志剑](http://www.cnki.net.ezproxy.lib.szu.edu.cn/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e9%a9%ac%e5%bf%97%e5%89%91&code=20756740;32046565;" \t "http://www.cnki.net.ezproxy.lib.szu.edu.cn/KCMS/detail/_blank).比特币的市场监管体制研究[N].常州信息职业技术学院学报,2014(05):21-24.
20. 郑书雯,范摇磊. 基于P2P 网络Bitcoin 虚拟货币的信用模型[J].信息安全与通信保 密,2012(3):72-75.
21. W Diffie，M Hellman．New directions in cryptography．IEEE Transactions on Information Theory,1976，11，22(6)：644-654.
22. 郭志宇. 基于Winpcap的客户访问控制系统设计[D].重庆.重庆大学.2012:20-26.
23. 徐军. 网络监控系统的设计与实现[D].成都.电子科技大学.2010:19-27.
24. 吕楠. Bitcoin合作式矿区挖矿研究[J].计算机技术与发展.2014,24(2):39-40.
25. 朱海霞. [基于协议分析的入侵检测系统的研究与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=2006056076.nh&dbcode=CMFD&dbname=CMFD2006&v=" \t "_blank)[D]. 大连.大连海事大学.2006:21-24.
26. 陈洪泉,纪若愚.基于Libnids的网络入侵检测系统的设计与实现[J].Scientific Research: 2010 Asia-Pacific Conference on Information Network and Digital Content Security,2010:150-152.
27. 祝晓光. 网络安全设备与技术[M].北京:清华大学出版社,2004:23-45.
28. 刘文涛. 网络安全开发包详解[M].北京:电子工业出版社,2005:34-67.
29. 龚俭,董庆,陆晟.面向入侵检测的网络安全检测实现模型[J].小型微型计算机系统,2001,22(2) :145-148.
30. 廖俊云. 基于协议分析的网络入侵检测系统[D].成都：电子科技大学硕士学位论文, 2005: 9-15.
31. 陈丽君.基于应用协议解析的分布式网络入侵检测系统研究[D].北京.北京服装学院.2011:26-29
32. Larry L , Bruce D. 计算机网络[M]. 第2版. 叶新铭, 贾波, 吴承勇等译. 北京: 机械工业出版社, 2001. 292-336
33. 杨天路,刘宇宏,张文原,毅强. P2P网络技术原理与系统开发案例[M].北京:人民邮电出版社.2007,2(1):45-46.
34. 李玉波, 朱自强, 郭军. linux C 编程[M]. 北京：清华大学出版社, 2005.
35. 聂元铭, 丘平.网络信息安全技术[M]. 北京：科学出版社, 2001.
36. W. Richard Stevens,《TCP/IP详解卷l:协议》[M]. 北京：机械工业出饭社.2000.
37. 吴洪, 方引青, 张莹. 疯狂的数字化货币——比特币的性质与启示[J].北京邮电大学学报（社会科学版）,2013(06):34-43.
38. 兰旭辉, 熊家军, 邓刚. 基于Mysql的应用程序设计[J].计算机工程与设计 2004(03):12-15.
39. B Yang and H Garcia-Molin. PPay: Micropayments for Peer-to-Peer Systems[J]. In Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS’03), V Atluri and P Liu, Eds. ACM Press. 2003:300–310.
40. 梁晓欢.Diamond Circle:为你敞开比特币安全支付之门.电脑与电信[J].2014(02):10-15.
41. 杨波. 现代密码学[M].北京：清华大学出版社，2007：13-67.
42. 冯登国. 国内外密码学研究现状及发展趋势[J]. 通信学报 2002,23(5):18-23.
43. 卜凡尧. 比特币的安全问题[J]. 保密科学技术.2013(07):16-21.
44. 刘宁, 沈大海. 解密比特币[M].机械工业出版社. 2014:12-45.
45. 谢翊, 杨继勋. [加强对虚拟货币监管的建议](http://www.cnki.net/kcms/detail/detail.aspx?filename=HNRD200801038&dbcode=CJFQ&dbname=CJFD2008&v=)[J]. 华南金融电脑. 2008(01):12-21.
46. [叶檀](http://epub.cnki.net/kns/popup/knetsearchNew.aspx?sdb=CJFQ&sfield=%D7%F7%D5%DF&skey=%D2%B6%CC%B4&scode=). 比特币:在零市场监管下“裸奔”[J][.IT时代周刊.2014(](http://www.cnki.net/kcms/detail/search.aspx?dbcode=CJFQ&sfield=inst&skey=%A1%B6IT%CA%B1%B4%FA%D6%DC%BF%AF%A1%B7&code=0865966;)01):23-25.
47. 陈恺,魏仕民,肖国镇.电子现金系统的研究与发展[J].西安科技大学学报:自然科学版.2008(10):11-16.
48. 钟鸣, 杨义先. 基于零知识证明的电子现金[J].通信学报,2001(06):56-57.
49. 唐汉明, 翟振兴. 深入浅出MySQL-数据库开发、优化与管理维护[M]. 北京:人民邮电出版社， 2009.
50. 施瓦兹. 高性能 MySQL 第三版[M]. 清华大学出版社. 2012.

**致 谢**

岁月如梭，三年美妙的研究生生活即将结束。回首往昔，文山湖荡漾的碧波，杜鹃山清脆的鸟鸣，还有和老师、同学们相处的欢乐时光，都使我胸中充满温馨和甜蜜。

首先，特别要感激我敬爱的导师喻建平教授。喻老师治学严谨、十分平易近人，温润如玉。迷茫的时候，喻老师会耐心的给我们分析，提出建议。在学习上也会给我们足够的空间，在实验室每周的学术讨论会，创造条件让我们多学习，多了解专业内外的理论与技术。还记得在签工作时，有点犹豫，征询喻老师的意见，喻老师耐心又冷静地结合我的实际，就业前景给我提出了客观而且宝贵的意见，一席话，给予了我莫大的前行的勇气。三年里，喻老师在生活上、学习上、还有精神上都给予我极大地帮助和指导。三年时光虽然不长，但师恩如海，带给我的是不尽的财富。老师常鼓励我们，年轻人不要怕输，要勇往直前；老师常激励我们，做科研要细致认真，要静下心来，使我们在课题研究方面不敢有丝毫的懈怠。在此向喻老师致以最诚挚的感谢！

感谢谢维信教授、黄敬雄教授和黄建军教授，他们给予科研项目极大的关注以及对于科研工作的指导，都使我受益匪浅。感谢刘宗香老师、裴继红老师、张勇老师和李良群老师对我的教导、支持与关心。感谢李岩山老师、高志坚老师和严海花老师，他们为我从事学习和科研工作提供了良好的环境和帮助。

还要感谢带我做项目的申屠青春师兄。师兄提供给我项目需要的各种软硬件设施，当我在项目过程中遇到问题时提供悉心的指导，以及对我生活提供了许多帮助。跟随师兄的这几年，无论是学习方面，还是为人处世方面，我都学到了很多，再次感谢师兄！

此外，我还要特别感谢与我一起做比特币相关项目的小伙伴吴文栋同学和已经毕业的熊林博师兄。吴文栋同学专业基础扎实、为人耐心细致、温暖。他给我的毕业设计提出了很多建设性意见，当我的项目进展停滞不前的时候，与我一同探寻各种可能性，无论是在精神上还是实际上都给予了我很大的帮助。熊林博师兄为人低调，勤奋刻苦，与其同在申屠青春的师兄的公司实习时，深受其刻苦钻研精神的影响，在他的带动下，阅读了很多专业相关书籍，养成了良好的遇到问题解决问题的思维方式。虽然师兄已经毕业，但是每当我在实际项目中遇到问题，他总会很耐心及时的给予力所能及的指导，并鼓励我在科研的道路上勇往直前。感谢吴文栋同学和熊林博师兄无私的帮助与关怀！

最后，感谢爸妈、师兄师姐、师弟师妹和同学们对我无私的关爱和照顾，感谢所有关心我的人，并感谢美丽的深圳大学带给我美好的回忆。