**c单位代码：10610**

**送审编号：2017223040095**



硕士研究生学位论文

（学术学位）

|  |  |
| --- | --- |
| 题目： | 区块链环境中木马命令控制信道构建 |
|  | 关键技术研究 |

|  |  |
| --- | --- |
| 学位类别： | 科学硕士 |
| 学科专业： | 计算机科学与技术 |

学位论文完成时间

二〇二〇年四月

# 摘 要

近年来，木马攻击技术作为高级持续性威胁（Advanced Persistent Thread，简称APT）攻击的重要手段，给政府、企业、机构带来了巨大的威胁，严重影响着国家安全。随着木马通信技术的发展，逐步形成了以IRC、HTTP、P2P等协议作为主要通信协议的传统木马命令控制信道（Command and Control，简称C&C），此类C&C信道通常存在单点失效、认证不完备等缺陷。针对此问题，本论文主要研究区块链上木马命令控制信道的构建方法及其运行机制，在区块链上构建安全、可靠、隐蔽的木马命令控制信道，解决传统C&C信道中存在的不足。本论文从攻击者的角度还原一种恶意利用公有链构建木马命令控制信道的技术场景，为防御人员研究此类攻击活动提供理论基础，对研究此类木马的防御方法和反制技术具有十分重要的意义。本论文的主要研究工作如下：

（1）设计动态端到端加密方案，确保木马通信数据的机密性。在区块链上构建的木马命令控制信道中，针对任何用户均可读取记录在区块链上木马通信数据的问题，结合对称加密技术和非对称加密技术设计动态端到端加密方案，防止机密信息泄露，确保信道中木马通信数据的安全。

（2）设计数据分片还原算法，实现任意长度数据在区块链上的传输。针对区块链上单笔交易信息中可嵌入的数据长度限制问题，基于哈希分类及校验设计数据分片还原算法，实现任意长度的数据在区块链上的可靠传输，增强信道的传输能力。

（3）提出通过匿名网络技术，提高木马控制端的抗溯源能力。木马控制端是木马命令控制信道的控制中心，一旦暴露将给信道带来严重威胁。针对此问题，提出通过匿名网络技术隐藏控制端的真实地址信息，提高木马控制端的隐蔽性和抗溯源能力。

（4）研究在区块链中基于转账交易和基于智能合约的木马通信数据传输方法。在基于转账交易实现木马命令控制信道中，通过交易信息中的可定义字段实现木马通信数据的传输，该方法通用性强；在基于智能合约实现的木马命令控制信道中，通过状态转移实现木马通信数据的传输，该方法利用智能合约代码能提高信道工作的效率。

通过在以太坊Ropsten区块链网络中进行实验，评估研究内容的正确性和可行性。实验结果表明在区块链上构建木马命令控制信道的方案是可行的，该信道具有较高的传输效率、可靠性、匿名性和鲁棒性。

**关键字**：木马；命令控制信道；区块链；以太坊；智能合约

# Abstract

In recent years, Trojan attack technology, as an important means of APT (advanced persistent threat) attacks, has brought huge threats to the government, enterprises and institutions and it affected national security seriously. With the development of Trojan communication technology, IRC, HTTP, P2P and other protocols as the main communication protocols of the traditional Trojan Command and Control channel (C&C) has gradually formed. Those C&C channels usually have defects such as single-point failure and incomplete authentication. In view of those problems, this paper mainly studies the construction method and operation mechanism of Trojan command and control channel on the blockchain, and builds a safe, reliable and hidden Trojan command and control channel on the blockchain, so as to solve the deficiencies in the traditional C&C channel. In this paper, from the attacker's point of view to show a technology scene which is a malicious use of the public blockchain to build a Trojan command control channel. It can provide a theoretical basis for the defense personnel to study this kind of attack activities, and it has great significance for the study of this kind of Trojan defense methods and countermeasures. The main research work of this thesis is as follows.

(1) Design dynamic end-to-end encryption scheme to protect the confidentiality of Trojan communication data. The Trojan command and control channel on the blockchain, anyone can read the Trojan communication data which is recorded on the blockchain. To solve this problem, a dynamic end-to-end encryption scheme is designed by combining symmetric encryption technology and asymmetric encryption technology to prevent the disclosure of confidential information and ensure the security of Trojan communication data in the channel.

(2) Design the data shard and restoration algorithm to realize the transmission of data with arbitrary length on the blockchain. Aiming at the length limitation of data which can be embedded in a single transaction information on the blockchain, design the data shard and restoration algorithm which is based on hash classification and validation to realize the reliable transmission of data of arbitrary length on the block chain and enhance the transmission capacity of the channel.

(3) Propose to improve the anti-traceability ability of Trojan control side through anonymous network technology. If the Trojan control side which is the control center of the Trojan command control channel was exposed, the channel will be seriously threatened. In order to solve this problem, it is proposed to hide the real address information of the Trojan control side by using anonymous network technology. This can improve the concealment and anti-tracing ability of the Trojan control side.

(4) Study the method of transferring Trojan communication data based on transfer transaction and smart contract on the blockchain. In the Trojan command control channel, the transmission of Trojan communication data is realized by the definable fields in the transaction information. In the Trojan command control channel based on smart contract, the Trojan communication data is transmitted by state transfer.

The correctness and feasibility of the research content were evaluated by conducting experiments in Ethereum Ropsten blockchain. The experimental results show that it is feasible to construct a Trojan command control channel on the blockchain, and it has high transmission efficiency, reliability, anonymity and robustness.

**Key Words**: Trojan; Command Control Channel; Blockchain; Ethereum; Smart Contract

目 录

[摘 要 I](#_Toc37241302)

[Abstract III](#_Toc37241303)

[第1章 绪论 1](#_Toc37241304)

[1.1 研究工作的目的与意义 1](#_Toc37241305)

[1.2 国内外研究现状 2](#_Toc37241306)

[1.3 主要研究内容与论文结构 3](#_Toc37241307)

[1.3.1 主要研究内容 3](#_Toc37241308)

[1.3.2 论文结构 4](#_Toc37241309)

[第2章 木马通信技术发展 5](#_Toc37241310)

[2.1 基于IRC实现的木马命令控制信道 5](#_Toc37241311)

[2.2 基于HTTP实现的木马命令控制信道 6](#_Toc37241312)

[2.3 基于P2P实现的木马命令控制信道 8](#_Toc37241313)

[2.4 其他方式实现的木马命令控制信道 10](#_Toc37241314)

[2.5 本章小结 10](#_Toc37241315)

[第3章 区块链技术 11](#_Toc37241316)

[3.1 区块链技术的发展 11](#_Toc37241317)

[3.2 区块链技术的原理 13](#_Toc37241318)

[3.2.1 数据层 14](#_Toc37241319)

[3.2.2 网络层 16](#_Toc37241320)

[3.2.3 共识层 17](#_Toc37241321)

[3.2.4 合约层 17](#_Toc37241322)

[3.3 智能合约 17](#_Toc37241323)

[3.4 公有链上构建木马命令控制信道的优势 19](#_Toc37241324)

[3.5 本章小结 20](#_Toc37241325)

[第4章 区块链上木马命令控制信道研究 21](#_Toc37241326)

[4.1. 木马命令控制信道模型设计 21](#_Toc37241327)

[4.2. 数据隐私保护 24](#_Toc37241328)

[4.2.1 加解密设计 24](#_Toc37241329)

[4.2.2 密钥协商 26](#_Toc37241330)

[4.3. 数据分片还原 27](#_Toc37241331)

[4.3.1 数据分片算法 28](#_Toc37241332)

[4.3.2 数据重组算法 29](#_Toc37241333)

[4.4. 匿名接入 31](#_Toc37241334)

[4.5. 数据传输 33](#_Toc37241335)

[4.5.1 基于转账交易传输 33](#_Toc37241336)

[4.5.2 基于智能合约传输 34](#_Toc37241337)

[4.6. 木马命令控制信道健壮性分析 36](#_Toc37241338)

[4.6.1 数据隐私保护机制 36](#_Toc37241339)

[4.6.2 数据分片还原算法 36](#_Toc37241340)

[4.6.3 匿名接入机制 37](#_Toc37241341)

[4.6.4 数据传输机制 37](#_Toc37241342)

[4.7. 本章小结 38](#_Toc37241343)

[第5章 区块链上木马命令控制信道的实现 39](#_Toc37241344)

[5.1 实验环境部署 39](#_Toc37241345)

[5.1.1 开发环境 39](#_Toc37241346)

[5.1.2 测试环境 40](#_Toc37241347)

[5.2 数据安全传输 40](#_Toc37241348)

[5.2.1 动态加解密 41](#_Toc37241349)

[5.2.2 数据分片还原 42](#_Toc37241350)

[5.2.3 通信数据上链 49](#_Toc37241351)

[5.2.4 匿名接入 55](#_Toc37241352)

[5.3 功能测试及分析 56](#_Toc37241353)

[5.3.1 信道基本功能测试 56](#_Toc37241354)

[5.3.2 信道指标测试 58](#_Toc37241355)

[5.4 本章小结 61](#_Toc37241356)

[第6章 结论与展望 63](#_Toc37241357)

[6.1 结论 63](#_Toc37241358)

[6.2 展望 64](#_Toc37241359)

[参考文献 65](#_Toc37241360)

图表目录

[图1.1 木马通信模型 1](#_Toc37241477)

[图2.1 IRC木马实现原理 5](#_Toc37241481)

[图2.2 HTTP木马实现原理 7](#_Toc37241482)

[图2.3 Fast-Flux命令控制流程 8](#_Toc37241483)

[图2.4 域名生成算法(DGA)原理 8](#_Toc37241484)

[图2.5 P2P木马命令控制流程 9](#_Toc37241485)

[图3.1 区块链发展历程 11](#_Toc37241486)

[图3.2 区块链技术框架 14](#_Toc37241487)

[图3.3 区块结构 15](#_Toc37241488)

[图3.4 公钥加解密过程 16](#_Toc37241489)

[图3.5 区块链中P2P网络结构 16](#_Toc37241490)

[图3.6 智能合约模型 18](#_Toc37241491)

[图4.1 区块链木马总体设计 22](#_Toc37241492)

[图4.2 区块链木马通信模型 23](#_Toc37241493)

[图4.3 木马一对多控制模型 25](#_Toc37241494)

[图4.4 区块链交易过程 26](#_Toc37241495)

[图4.5 密钥协商过程 27](#_Toc37241496)

[图4.6 分片数据结构 28](#_Toc37241497)

[图4.7 数据分片过程 29](#_Toc37241498)

[图4.8 数据重组过程 30](#_Toc37241499)

[图4.9 数据分片与重组关系 31](#_Toc37241500)

[图4.10 控制端通过匿名网络接入区块链 32](#_Toc37241501)

[图4.11 交易信息结构 33](#_Toc37241502)

[图4.12 木马通信交易一般形式 33](#_Toc37241503)

[图4.13 智能合约的组成 34](#_Toc37241504)

[图4.14 不同用户角色权限分配 35](#_Toc37241505)

[图5.1 木马通信数据在传输过程中的形态变化 41](#_Toc37241506)

[图5.2 端到端动态加密实现 41](#_Toc37241507)

[图5.3 分片数据结构设计 42](#_Toc37241508)

[图5.4 转账交易方式中不同分片长度实验对比（粗粒度） 45](#_Toc37241509)

[图5.5 转账交易方式中不同分片长度实验对比（细粒度） 47](#_Toc37241510)

[图5.6 通过智能合约传输12500字节时的生成交易信息 48](#_Toc37241511)

[图5.7 智能合约方式中不同分片长度对比 49](#_Toc37241512)

[图5.8 交易信息 50](#_Toc37241513)

[图5.9 木马运行机制 51](#_Toc37241514)

[图5.10 目标主机信息数据结构 53](#_Toc37241515)

[图5.11 智能合约中发布命令的函数 55](#_Toc37241516)

[图5.12 木马控制端发送指令 56](#_Toc37241517)

[图5.13 木马受控端接收执行指令 57](#_Toc37241518)

[图5.14 木马控制端上传文件 57](#_Toc37241519)

[图5.15 木马受控端接收文件 57](#_Toc37241520)

[图5.16 事件定义 58](#_Toc37241521)

[图5.17 RPC请求格式 60](#_Toc37241522)

[图5.18 数据包过滤条件 61](#_Toc37241523)

[表2.1 不同协议实现C&C信道的优缺点 10](#_Toc37241524)

[表3.1 区块链的类型与特性 13](#_Toc37241525)

[表3.2 不同共识算法的特点 17](#_Toc37241526)

[表3.3 不同区块链平台上智能合约对比 19](#_Toc37241527)

[表4.1 比特币服务器节点信息 31](#_Toc37241528)

[表4.2 不同用户的用途 35](#_Toc37241529)

[表5.1 开发环境软硬件配置 39](#_Toc37241530)

[表5.2 测试环境软硬件配置 40](#_Toc37241531)

[表5.3 分片头部结构 42](#_Toc37241532)

[表5.4 数据分片算法代码实现 43](#_Toc37241533)

[表5.5 分片重组算法代码实现 43](#_Toc37241534)

[表5.6 转账交易方式中不同分片长度对比表（粗粒度） 45](#_Toc37241535)

[表5.7 转账交易方式中不同分片长度对比表（细粒度） 46](#_Toc37241536)

[表5.8 智能合约方式中不同分片长度对比 48](#_Toc37241537)

[表5.9 全局变量定义 53](#_Toc37241538)

[表5.10 函数定义 54](#_Toc37241539)

[表5.11 事件定义 55](#_Toc37241540)

[表5.12 不同用户权限测试 58](#_Toc37241541)

[表5.13 传输效率比较 59](#_Toc37241542)

# 绪论

## 研究工作的目的与意义

近年来，计算机技术和因特网技术（Internet）极速发展，因特网技术被广泛应用于社会生活的各个领域，给人们的生活和工作带来了极大便利，提高了人们的生活质量，改变了人们的工作习惯。但随之出现的网络安全问题也日益突出，恶意代码攻击层出不穷，严重威胁到国家安全和个人隐私。恶意代码通常包括僵尸程序、病毒、蠕虫、木马、间谍软件等，其中木马是与僵尸网络最为接近的恶意代码,尤其是在命令控制信道实现上。我国是网络攻击最大的受害国之一，长期遭受木马、病毒、勒索软件等恶意程序的攻击，造成巨大损失。

高级持续威胁[2]（Advanced Persistent Thread，简称APT）是由美国国防部和美国空军于2006年提出的一个概念，用于描述一个有针对性的、隐蔽的、持续的攻击活动，该活动通常以获取目标网络敏感信息或破坏目标网络计算机系统为目的。木马技术具有隐蔽性高、可控性强、后台运行等特点，是APT攻击活动的主要手段之一。木马通信是指木马控制端与木马受控端之间的通信；传输木马通信数据的链路通常被称为木马命令控制信信道（Command and Control Channel，简称C&C信道），是木马技术的主要组成部分。根据拓扑结构的不同，木马通信模型可分为集中式模型和分布式模型，如图1.1所示。



**图1.1 木马通信模型**

集中式模型中，木马受控端直接与命令控制服务器（Command and Control Server，简称C&C服务器）进行通信，各个木马受控端之间相互独立、不发生通信行为。目前集中式木马通信模型的实现协议有IRC、HTTP、HTTPS、POP3等协议，通过将木马通信数据隐藏在海量的正常通信流量中可以使木马通信行为更加隐蔽。集中式木马通信模型中存在的最大威胁是单点失效的问题，一旦C&C服务器被外力关停，攻击者将失去控制权。分布式模型中，木马受控端与C&C服务器之间不直接进行通信，木马通信数据通过P2P协议随机选择路径从发送端传输到接收端。在P2P网络中任何节点既是客户端又是服务端，木马通信数据的传输通过对等节点来完成。基于P2P协议实现的分布式木马容易遭受Bootstrap、Index Poisoning、Sybil、Peer-list等攻击，造成P2P网络瘫痪，影响木马控制端与目标主机之间的通信。

2008年区块链的概念首次在论文《比特币：一种点对点式电子现金系统》中被提出后，区块链技术受到了全世界的高度关注。区块链具有去中心化、去信任、不可篡改、匿名、开放等特征，这些特性使得区块链上的数据不受监管，任何人都可以在区块链上存储和传播任意信息，同时任何组织或机构不能篡改区块链中的数据。当前主流木马命令控制信道存在的单点失效、认证不完备的问题，而这些问题能够通过区块链技术来解决。本文从攻击者的角度，还原一种恶意使用区块链传输木马通信数据的技术场景，为安全防御人员了解此类木马通信技术提供一个模型参考，有利于进一步研究相应的防御方法和对抗技术，而随着区块链技术的快速发展，此类恶意使用区块链实现木马通信的攻击活动将逐步发展成熟，提前做好相关的研究具有十分重要的现实意义。实现一款功能强大的木马系统并不是本文的目的，在本文中实现的木马模型仅仅是一个验证性概念（Proof of Concept，简称PoC），主要用于辅助验证本文的研究内容。

## 国内外研究现状

Roffel等人[7]首次提出了通过比特币网络来控制电脑蠕虫的想法。Ali等人[8]进一步提出通过比特币网络实现僵尸网络控制，并探讨了四种在比特币区块链中存储和隐藏控制命令的方法。Jonathan等人在文献中讨论了如何通过私有链来增强僵尸网络命令控制信道的弹性的问题。Pirozzi等人[11]提出将区块链作为新一代僵尸网络的命令控制中心，并在比特币网络中实现全双工的BITCHAIN，但是由于交易费用的问题极大的限制了其可扩展性。Unblockable Chains和BOTRACT分别基于以太坊网络实现僵尸网络命令控制基础设施，由于命令被公开记录在区块链上，它们均存在匿名性差的问题。Baden等人提出利用以太坊的Whisper协议实现僵尸网络命令控制的方案，但使用的主题有被列入黑名单的可能，存在单点失效的问题。Ahmet等人[15]实现的LNBot，在比特币闪电网络上通过交易的方式来传递C&C消息，与Botchain相比，该方案具有更高的隐蔽性、更快的传输速率以及更低的成本。Zhong等人[16]实现的DUSTBot，分别通过比特币主网和测试网来发布指令和回传结果，实现了一个隐蔽的、双工的、比较经济的命令控制信道，但是存在传输效率低的问题。李彦峰等人提出并证明了区块链环境下的新型网络隐蔽信道模型具有抗干扰、抗篡改、匿名、多路通信、线路无关等优良特性，能够有效克服当前网络隐蔽信道中存在的缺陷和弊端。

当前国内外的研究工作主要集中于区块链与僵尸网络的结合方案，通过区块链网络实现僵尸网络控制。木马是与僵尸程序最为接近的一种恶意代码，但两者之间也存在差异，主要体现在三个方面：

1. 控制方式。相比于僵尸程序，木马对目标主机的控制性更强，通常要求木马控制端与木马受控端之间能够进行交互。而僵尸网络中，僵尸程序只需要能够接收控制者的指令并完成相应的动作即可，不要求与僵尸控制者进行直接交互。
2. 协同性。僵尸主机之间通常需要协同完成任务，如命令中继和分布式拒绝服务攻击等。木马主机之间通常相互独立完成任务，不需要协同工作。
3. 功能。木马通常用于窃取用户隐私、获取敏感信息的目的，功能相比僵尸程序更加复杂。

## 主要研究内容与论文结构

### 主要研究内容

论文主要研究区块链环境中木马命令控制信道的构建方法，分析信道实现过程中存在的问题，针对这些问题研究并提出相应的解决方案。论文完成的主要工作如下。

1. 设计动态端到端加密方案，确保木马通信数据的机密性。区块链具有开放性特征，任何用户均可从区块链账本中读取数据，木马通信数据在区块链中传输时将被永久记录在区块链中，木马通信过程中存在机密信息泄露的风险，针对上述问题，综合对称加密技术和非对称加密技术的优点设计动态端到端加密方案对区块链上的木马通信数据进行保护，防止机密信息泄露，确保木马通信数据安全。
2. 设计数据分片还原算法，实现任意长度木马通信数据在区块链上传输。区块链中单笔交易信息中可嵌入的消息长度是有限制的，当需要传输的数据超过这个长度限制时，需要将数据拆分后进行传输并在接收端完成重组合并。针对此问题，基于哈希分类和校验技术设计数据分片还原算法对木马通信数据进行分片传输和重组合并，实现任意长度的木马通信数据在区块链上可靠传输，增强木马命令控制信道的鲁棒性。
3. 提出通过匿名网络技术，提高木马控制端的抗溯源能力。木马命令控制信道中，木马控制端是控制中心，其安全性直接关系到整个信道的安全性，将木马控制端隐藏在匿名网络之后，能有效隐藏木马控制端的真实地址信，提高木马控制端的隐蔽性和抗溯源能力。
4. 研究木马通信数据在区块链传输的方法，分别基于转账交易和智能合约两种方式构建木马命令控制信道。在利用转账交易实现的木马命令控制信道中，发送方将木马通信数据嵌入交易信息中，通过发起转账交易的方式将交易信息记录到区块链账本中，接收方从区块中读取该交易信息并解析得到通信内容。在利用智能合约实现的木马命令控制信道中，发送方通过调用智能合约接口将木马通信数据记录到合约状态中，接收方通过调用智能合约接口从合约状态中获取通信数据。与转账交易的方式相比，用户不需要了解区块链底层数据组织结构，通过合约代码动态获取木马通信数据，提高了信道的工作效率。
5. 实验部分，在以太坊Ropsten测试网中分别通过转账交易传输方式和智能合约传输方式实现木马命令控制信道，对信道中各功能进行实验测试并采集实验数据，通过信道的传输效率、误码率，通联关系分析其传输性能、可靠性和匿名性。

### 论文结构

本文主要包括六章，各章节的内容分别如下：

第一章，绪论。系统介绍本课题的目的与意义。概述当前国内外针对僵尸网络与区块链技术结合的研究现状，介绍当前使用区块链实现僵尸网络命令控制中存在的问题，阐述本课题研究的内容、目的和意义。

第二章，木马通信技术发展。介绍当前常被用于实现木马命令控制信道的几种协议，并分析各种协议实现木马命令控制信道的方法及其优缺点。

第三章，区块链技术。主要介绍区块链的发展、区块链的运行机制和智能合约相关的内容，从技术层面分析区块链能够实现去中心化、去信任、不可篡改、匿名、开放的原理。分析在公有链上构建木马命令控制信道的优势。

第四章，区块链上木马命令控制信道研究。分析在区块链中构建木马命令控制信道时面临的敏感数据暴露、单次传输数据量限制、木马控制端匿名性、数据的传输方式等问题，提出通过数据隐私保护、数据分片还原、匿名接入、数据传输四个过程及其关键技术来解决。详细阐述各过程中采用的关键技术和实现细节、分析各过程的健壮性。

第五章，区块链上木马命令控制信道实现。根据第四章中研究的内容，在以太坊Ropsten测试网中构建木马命令控制信道，对其功能和性能进行测试，分析实验数据。

第六章，结论与展望。对本文的内容进行总结。

# 木马通信技术发展

木马通信协议也称为木马命令控制信道（C&C），是木马系统的核心组成部分，主要用于木马控制端与木马受控端之间的信息交换，其实现方式决定了木马系统的传输效率以及健壮性。木马命令控制信道一旦被阻断，控制者将失去对目标主机的控制权。命令控制信道的实现协议多种多样，历史上使用广泛的实现协议包括互联网中继聊天协议（Internet Relay Chat，简称IRC）、超文本传输协议（Hyper Text Transfer Protocol，简称HTTP）、点对点传输协议（Point To Point，简称P2P），其中IRC和HTTP采用C/S架构，是中心化的，而纯P2P是分布式的，不存在中心节点。本章将分别介绍通过IRC协议、HTTP协议、P2P协议实现木马命令控制信道的方法及其运行机制。

## 基于IRC实现的木马命令控制信道

IRC协议是早期使用比较广泛的网络聊天协议，IRC用户通过客户端成功连接到IRC服务器后，即可与其他成功连接到该服务器并处于同一个聊天室（也称为频道）的IRC用户实时交流。IRC协议具有实现简单、速度快、应用广泛、功能全面等特点，早期的攻击者基于该协议来实现木马通信协议、构建木马命令控制信道。使用IRC协议构建木马命令控制信道具有两个优势：

1. 信道部署简单，控制者能够依托公共的IRC服务器来构建信道；
2. 速度快，IRC协议中数据是实时传输的，控制者可以实时对目标主机进行控制。

基于IRC协议的木马命令控制信道的实现原理如所示。



**图2.1 IRC木马实现原理**

1. 木马控制者在IRC服务器中创建一个聊天室，该聊天室主要用于木马控制端与木马受控端之间的信息交换；
2. 控制者登录到IRC服务器（C&C服务器）并进入特定聊天室后，向一台或多台目标主机发送指令；
3. 目标主机加入聊天室获取指令；
4. 目标主机执行指令并将执行结果通过聊天室发送给控制者；
5. 控制者从聊天室中获取指令执行结果。

通过以上(2)、(3)、(4)、(5)四个步骤，木马控制端与木马受控端之间完成了信息交换，从而实现了基于IRC协议的木马命令控制信道。木马控制者向目标主机发送指令的方式有二种：

1. TOPIC方式，控制者将指令发布到聊天室公告中，当目标主机登录并进入该聊天室时首先会接收到公告信息，通过该方式可以实现离线命令执行。
2. PRIVMSG方式，控制者登录进入聊天室中通过群聊或者私聊的方式向特定的并在线的目标主机发布指令。

IRC协议是明文协议，通信端口和通信内容具有显著特征，其流量在传输过程中容易被检测到，因此基于IRC协议实现的木马命令控制信道容易被检测、追踪和封锁。使用IRC协议作为通信协议的木马系统其拓扑结构是中心化的，存在单点失效的问题，一旦IRC服务器或聊天室被关闭，木马控制端和木马受控端之间将不能再进行信息交换，导致木马控制者失去对目标主机的控制权。

## 基于HTTP实现的木马命令控制信道

HTTP协议和HTTPS协议（HTTP协议的加密版本）是目前使用最为广泛的应用层协议之一，具有灵活、简单、快速、高效等特点，能够穿透IDS和防火墙等安全防护设备，是当前主流的木马命令控制信道实现协议。攻击者将木马通信流量混杂在大量的HTTP正常请求之中，使得其不容易被检测出来，特别是使用HTTPS协议进行通信的木马系统，其通信流量更难以被检测到。基于HTTP协议实现的木马命令控制信道，其命令控制流程如图2.2所示。具体步骤如下：

1. 木马控制者在WEB服务器上发布指令；
2. 目标主机通过周期轮询的方式访问WEB服务器获取指令内容；
3. 目标主机执行指令并将执行结果回传到WEB服务器上；
4. 木马控制者从WEB服务器上取回指令执行结果。



**图2.2 HTTP木马实现原理**

图2.2中的WEB服务器（C&C服务器）可以使用自建的，也可以使用能够发布自定义内容的公共WEB服务器，如博客、微博、QQ空间、Facebook、Twitter、Github，Pastbin等，使用此类流行的内容自定义网站作为C&C服务器能够进一步增加检测和封锁的难度，使木马命令控制信道更加隐蔽和健壮。与IRC协议一样，基于HTTP协议实现的木马命令控制信道也是中心化的，存在单点失效的问题。在使用HTTP协议实现C&C信道时，攻击者通常将C&C服务器的地址信息（如域名、IP地址、URL地址等信息）硬编码到木马程序中，木马受控端程序在目标主机上运行后解码得到C&C服务器的地址信息并与之通信。这种静态寻址的方式很容易暴露C&C服务器的地址信息，防御人员通过逆向分析木马程序可以获取C&C服务器的地址信息并对其进行封堵，从而阻断木马命令控制信道。为了解决静态寻址的脆弱性问题，提高命令控制信道的健壮性，攻击者探索了动态寻址的方法。典型的动态寻址有Fast-Flux和Domain-Flux。

Fast-Flux的基本思想是快速切换单个域名与多个IP之间的映射关系，其目的是隐藏C&C服务器的真实IP信息。这些映射的IP通常来自一个由部分目标主机IP地址组成的IP池，IP池中的目标主机充当代理节点，透明转发其他目标主机与C&C服务器之间的通信流量，如所示。Fast-Flux能有效对抗屏蔽IP地址的防御手段，但是无法阻止防御人员通过屏蔽域名的方式来阻断木马命令控制信道，Fast-Flux增强了信道的健壮性，但没有解决单点失效的问题。



**图2.3 Fast-Flux命令控制流程**

Domain-Flux的核心是域名生成算法（Domain Generation Algorithm，简称DGA），木马控制端和木马受控端使用同一套DGA，通过相同的输入生成大量的域名，控制者将其中的任意一个域名解析到C&C服务器上，木马控制端和木马受控端即可进行通信。域名生成的过程如所示，DGA算法中将字典、时间、热门话题等内容作为种子（Seeds）输入，计算生成域名前缀并添加顶级域（Top-level Domain，简称TLD），得到最终的域名。由于生成的域名数量多且不断变化，防御人员很难通过屏蔽IP地址或域名的方式来切断命令控制信道。Domain-Flux协议中会生成大量的域名，寻址效率低；在寻址过程中会产生大量的DNS请求，容易被检测出来；DGA算法一旦被攻破，防御人员通过提前抢注域名等手段可接管攻击者的控制权。



**图2.4 域名生成算法(DGA)原理**

## 基于P2P实现的木马命令控制信道

基于IRC协议和HTTP协议实现的木马命令控制信道中均使用了中心化的C&C服务器，存在单点失效的问题，如果C&C服务器被劫持或关闭，攻击者将失去对目标主机的控制权。在纯P2P协议中不存在中心节点，所有节点均是对等的，同时P2P协议具有健壮、灵活和可扩展等优良特性，经常被用于构建命令控制信道。基于P2P协议的木马系统中，所有目标主机组建成一个P2P网络，各目标主机就是一个对等节点，控制者将控制指令注入到其中的一个节点中，指令数据通过节点间的信息交换传输到特定的目标主机，如所示。控制者向节点中注入控制指令后，指令依次经过路径①、②、③、④到达特定目标主机，目标主机执行完指令后将执行结果经过路径⑤和⑥返回给控制者，数据从发送节点到接收节点的过程中传输路径是随机选择的。



**图2.5 P2P木马命令控制流程**

根据拓扑结构的不同P2P网络可分为结构化P2P网络和非结构化P2P网络。P2P网络中最大的难点是节点地址的管理，节点之间通过泛洪（Flooding）的方式查询和发现，会产生大量的网络流量，造成网络拥堵。结构化P2P网络采用分布式哈希表（Distributed Hash Table，简称DHT），通过哈希函数将不同节点的地址规范成固定长度，整个系统运转比较高效，每个节点只需存取少量其他节点的信息即可保证网络的正常运转，典型的案例有CAN、Choed、Pastry、SkipNet等。非结构化的P2P网络采用随机图的方式形成一个松散的网络[28]，节点可以自由加入和退出，通常使用泛洪（Flooding）技术实现节点发现和动态转发的机制，大量的广播信息会造成网络的拥堵，运行效率比较低，典型的案例有Gnutella、KaZaA、FreeNet。

基于P2P协议实现的命令控制信道虽然改善了单点失效的问题，但存在三个方面的缺陷。

1. Bootstrap脆弱性。Bootstrap是节点寻找并加入P2P网络的过程，该过程依赖于硬编码在木马程序中的Peer-list或Server Cache节点，如果Peer-list或Server Cache失效，新的节点将无法加入到网络中接收控制者的指令。
2. 节点间缺乏认证。新节点加入网络时缺乏有效的认证机制，防御人员通过逆向工程技术掌握通信协议后，可以加入大量的蜜罐节点对整个网络进行测量以及对目标主机实施欺骗，可进一步切断命令控制信道。
3. 命令认证不完善。理论上，网络中的任何节点均可充当服务端向网络中注入控制命令，如果防御人员发现命令认证机制中存在缺陷，则可以接管控制权或干扰信道的正常运行。

## 其他方式实现的木马命令控制信道

IRC协议、HTTP协议和P2P协议是攻击者用来构建木马命令控制信道的三种典型的协议，除此之外，能够用于实现木马命令控制信道的协议还有很多，如DNS协议、POP3协议、网盘协议、ICMP协议、ARP协议等，不同协议实现的C&C信道优缺点对比如所示。

**表2.1 不同协议实现C&C信道的优缺点**

**Table 2.1 Advantages and Disadvantages of C&C with Different Protocols Implementes**

|  |  |  |
| --- | --- | --- |
| **协议** | **优点** | **缺点** |
| DNS协议 | 穿透性强 | * 产生大量的DNS查询请求容易被检测出来 * 存在单点失效的问题 |
| POP协议 | 隐蔽性高 | * 通信时会快速产生大量的邮件，容易被当作垃圾邮件丢弃 * 存在单点失效的问题 |
| 网盘协议 | 隐蔽性高 | * 存在单点失效的问题 |
| ICMP协议 | 穿透性强  隐蔽性高 | * 存在单点失效的问题 |
| ARP协议 | 隐蔽性高 | * 只能在局域网中使用 |

使用不同协议构建木马命令控制信道，最终目的都是使信道更加健壮，即穿透性更强、隐蔽性更高、更不容易被阻断或接管。

## 本章小结

本章主要阐述了木马通信技术的发展演进历程，从简单到复杂、从静态寻址到动态寻址、从中心化架构到分布式架构发展，多种实现技术共存；重点介绍了使用IRC协议、HTTP协议、P2P协议构建木马命令控制信道的方法，并分析各种协议实现的命令控制信道存在的缺陷；最后比较了其他协议实现木马命令控制信道的优缺点。

# 区块链技术

区块链（Blockchain）技术也被称为分布式账本技术（Distributed Ledger Technology，简称DLT），是一种由多方参与并共同维护，借助密码学保证传输和访问安全，能够确保数据一致存储、不可以篡改、防止抵赖的记账技术。

## 区块链技术的发展

2008年10月31日学者中本聪（Satoshi Nakamoto）在其论文《Bitcoin: A peer-to-peer electronic cash system》中首次提出了区块链的概念，文中指出通过区块链技术可实现一种全新的、去中心化、去信任的点对点交易体系，是构建比特币电子现金系统的关键基础技术。由于具有分布式、去信任、不可篡改等优点，区块链技术受到各行业的高度关注，一度被认为是构建未来“信任互联网”、“价值互联网”的支撑技术。2013年Vitalik Buterin发表《以太坊白皮书》，提出基于区块链构建下一代加密货币与去中心化应用平台。白皮书中指出以太坊区块链平台的设计目标是创建一个永不停止、无需审查和自动维护的分布式世界计算机。它构建了类似于比特币的数字加密货币以太币，同时提供了图灵完备的编程环境以支持智能合约的运行，通过智能合约可实现去中心化应用程序（Decentralized Application，简称DApp）。目前区块链技术的发展经历了三个阶段，分别是技术起源、区块链1.0和区块链2.0，如图3.1所示。



**图3.1 区块链发展历程**

（1）技术起源：区块链技术是将现有的多种成熟技术结合在一起形成的新技术，这些成熟技术包括P2P网络技术、加密技术、数据库技术和电子现金相关技术。P2P网络技术是区块链系统的组网技术，将各对等节点连接组合在一起，是区块链技术架构中的核心技术；非对称加密技术中通过公私钥对维持区块链各节点之间的信任关系；数据库技术是软件业的基石，也是建设区块链系统的基础技术；电子现金是对现实货币的模拟，当前电子支付手段中的信任关系通常依托于第三方可信机构，而区块链技术改变了这种支付模式使得支付时不再需要中介。

（2）区块链1.0：以可编程虚拟货币为主要特征，典型代表是比特币。通常采用以区块为单位的链状数据块结构，各节点通过共识机制对交易进行打包，每次打包成功后均会生成一个新的区块并向全网广播，在新的区块中包含当前时间戳、一段时间内发生的多笔有效交易及其梅克尔树根值、上一个区块的哈希值等内容。由于各区块通过密码学证明的方式与其前续区块链接在一起，当区块达到一定高度后，要想修改历史区块中的交易信息就必须重构该区块之后所有新增的区块，而这个重构的难度非常巨大，从而能够有效防止历史区块被篡改。在区块链网络中，所有全节点均存储了完整的、一致的账本数据，同时全网各节点通过P2P协议互连，不存在中心服务器，因此单个节点受到攻击或账本数据被篡改不会影响到全网总账的安全。区块链中的账户体系通常由非对称加密体制中的公钥和私钥组成，只有通过私钥才能使用对应公钥中的资产。区块链1.0主要实现了数字货币的功能。

（3）区块链2.0：以智能合约为主要特征，典型代表是以太坊。区块链为智能合约提供可靠的运行环境，通过外部事件触发智能合约执行，自动完成合约规定的内容。通过智能合约可以实现复杂多样的去中心化应用而不仅限于数字货币，如分布式身份认证、分布式域名系统、分布式自治组织等。当前区块链技术的发展处于2.0阶段。

根据应用场景和设计的不同，区块链通常可以分为公有区块链、联盟区块链和私有区块链三类。

（1）公有区块链（以下简称：公有链）中任何节点均可自由加入或退出区块链，任何人均可获取区块链上全部账本数据的副本、发送交易和参与打包。公有链是去中心化的，不受任何组织或机构控制，依靠数学难题来保证其安全。通过共识机制和激励机制来提高用户的参与度，从而提高公有链的安全性和稳定性。

（2）联盟区块链（以下简称：联盟链）是用于解决多个联盟组织之间信任问题的区块链，因此联盟链是多中心化的。联盟链中的各机构运行一个或多个节点，节点在加入或者退出网络时需要通过授权，所有授权节点共同维护区块链的正常运转。

（3）私有区块链（以下简称：私有链）也被称为专有区块链，是指该区块链由一个组织或个人控制，只有通过许可的节点才能接入到区块链网络中参与和查看数据。虽然私有链失去了部分去中心化的特性，但其仍然具备区块链的通用结构，由多个节点共同维护，适用于机构内部数据的管理和审计。

以上三种不同类型的区块链的特点如表3.1所示。

**表3.1 区块链的类型与特性**

**Table 3.1 The Classification and Characteristics of Blockchain**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **公有链** | **联盟链** | **私有链** |
| 参与者 | 任何人 | 预置或满足条件加入的成员 | 由控制者决定参与成员 |
| 中心化程度 | 去中心化 | 半去中心化 | 中心化 |
| 激励机制 | 需要 | 可选 | 不需要 |
| 特点 | 1. 交易成本高 2. 交易速度慢 3. 数据公开 | 1. 低成本运行和维护 2. 交易速度快、扩展性好 3. 较好的保护隐私 | 1. 交易成本极低 2. 交易速度非常 3. 更好的保护隐私 |
| 典型代表 | 比特币、以太坊 | Fabric | Multichain |

区块链技术的快速发展也引起了我国政府的高度重视，2016年工信部指导编写的《中国区块链技术和应用发展白皮书（2016）》是我国第一个区块链官方指导文件，与此同时，区块链技术作为战略性前沿技术被列入《“十三五”国家信息化规划》。2017年编写的《软件和信息技术服务业“十三五”技术标准体系建设方案》，有效指导了国内区块链标准化的工作。2017年中国区块链技术和产业发展论坛发布《区块链参考框架》团体标准，对区块链系统架构的标准化发挥了重要作用，现已转化成为国家标准。2018年1月，ISO/TC 307正式启动了国际标准ISO 23257《信息技术区块链和分布式记账技术参考架构》项目，该项目在《区块链参考架构》团体标准的核心内容的基础上构建区块链系统架构的国际标准。2018年5月，IEEE P2418.2《区块链系统标准数据格式》国际标准获批立项。2019年10月24日，中央政治局就区块链技术发展现状和趋势进行第18次集体学习，习近平主席在主持学习时强调了区块链技术在新的技术革新和产业变革中的重要作用，提出加快推动区块链技术和产业创新发展的要求。因此，区块链技术必将在我国快速发展和应用，随着区块链技术及应用的逐渐普及，滥用区块链的情况将不断增长，区块链的安全问题非常值得深思。

## 区块链技术的原理

从狭义角度来讲，区块链是一种将数据区块按时间顺序组合相连而成的链式数据结构，是使用密码学方法保证的、可抗篡改和防伪造的分布式账本。从广义角度来讲，区块链是一种分布式基础架构和计算范式，其使用可验证的块链式数据结构来存储数据、通过共识机制来生成和更新区块、依靠密码学的知识来保证数据安全、可使用智能合约来操作数据。区块链中融合了密码学技术、P2P网络技术、共识机制，数据库技术等多种成熟技术，实现了区块链的四大特点：去中心化与集体维护、去信任、不可篡改与可追溯、匿名性与安全可靠。区块链的技术架构如所示，包括数据层、网络层、共识层、激励层和合约层。



**图3.2 区块链技术框架**

### 数据层

区块链系统中的矿工节点通常将一段时间内接收到的多笔交易信息进行打包，生成一个带有时间戳的数据区块，然后将该区块链接到当前主链上，形成新的主链。打包的过程涉及多个技术要素，包括数据区块、哈希函数、时间戳、Merkle树、数字签名、链式结构等。

典型的区块结构如所示，由区块头和区块体两个部分组成。区块头中包含了区块链的版本信息（Version）、前一区块的地址（Pre-block）、当前区块的目标哈希值（Target-Hash）、当前区块在工作量证明（Proof of Work，简称PoW）共识过程中的随机数解（Nonce），时间戳（Timestamp）以及Merkle根（Merkle-Root）等信息。区块体中记录了交易的数量以及交易信息，这些交易记录以Merkle树的形式进行组织。在区块链中使用的Merkle树是二叉树，如图3.3所示，每个叶子结点中记录了一笔交易信息，在构建Merkle树时将叶子结点两两相加并计算其和的哈希值，以此递归下去，最终生成的哈希值就是Merkle树的树根。哈希函数对输入非常敏感，不同的输入数据计算出来的哈希值不同，因此，如果叶子节点中的交易信息被篡改，则会引起Merkle根的哈希值发生变化，进而导致当前区块中区块头的哈希值发生改变。由于当前区块的下一个区块中的Pre-block保存了当前区块的哈希值，而这个哈希值的改变会引起下一个区块头部哈希值发生变化，以此类推会造成当前区块之后所有区块的哈希值都发生改变，也就是说想要篡改某一区块的数据，则需将该区块后续的区块全部篡改，而完成这项工作需要控制超过全网51%的算力，这几乎是不可能做到的，因此使用Merkle树作为存储数据的组织结构能够保证区块链账本中数据不能被篡改。



**图3.3 区块结构**

区块链通过链式结构将区块按时间顺序链接起来形成主链，主链中记录了完整的区块链数据，能够快速对历史交易进行定位和溯源。当前区块中包含了前一区块的哈希信息，通过当前区块可以快速验证前一区块的正确性。矿工在对区块进行打包时，在区块头部加盖时间戳来保证主链上各区块是按时间顺序依次链接的。时间戳标明了区块数据写入主链的时间，是对每一次交易记录的认证，为链上的交易数据增加了时间维度。

区块链中使用了密码学相关的安全技术来保障数据的安全，这些技术包括：哈希技术、公钥加密技术、数字签名技术等。

哈希函数可以将任意长度的数据作为输入生成一个固定长度的哈希值，哈希值能够用于验证区块数据的正确性，防止区块数据被篡改。哈希函数通过三个特性来保证数据的准确性。首先，哈希函数非常敏感，即使输入数据相差非常微小，生成的哈希值也千差万别；其次，哈希算法具有很强的抗碰撞能力，找到两个不同的输入数据使得输出的哈希值相同的可能性非常小；最后，哈希函数是单向的，无法通过哈希值计算出相应的输入数据。

公钥加密算法也被称为非对称加密算法，算法中包括公钥和私钥：公钥和私钥是成对的，其中公钥是公开的，用于加密数据；私钥只有用户拥有，用于解密密文。公钥加密解密的过程如所示。公钥加密算法的安全性通常依赖于数学难题，以当前掌握的知识和计算能力，不能通过公钥推测或计算出对应的私钥。目前应用广泛的公钥加密算法有RSA（Rivest Shamir Adleman，简称RSA）和椭圆曲线密码（Elliptic Curve Cryptography，简称ECC）等。区块链中通常使用ECC算法，因为ECC算法密钥长度更短，加解密效率相对更高，同时所有的用户可以使用同样的操作在同一个有限域上不同的椭圆曲线上完成域运算。区块链系统中用户的公钥和私钥是通过椭圆曲线函数以及哈希算法来生成的。在交易过程中，发送者使用接收者的公钥对交易信息进行加密并将密文发送给接收者，接收者使用私钥解密密文获取交易信息，有效保障了交易信息的机密性。同时，发送者使用其私钥对交易信息进行签名，并将生成的数字签名发送给接收者，接收者使用发送者的公钥对该数字签名进行验证，确保交易信息来源的可靠性和真实性。



**图3.4** **公钥加解密过程**

### 网络层

区块链系统中使用P2P网络技术进行组网，将区块链中各对等节点连接在一起形成区块链网络。由于采用P2P组网方式，区块链网络中各节点的地位是平等的，不存在中心节点以及层级结构，因此区块链系统是去中心化的，如所示。同时，各个节点独立参与系统，均会承担节点发现、路由、验证与传播交易信息等工作，因此部分节点宕机或遭受攻击时不会影响区块链系统的运行。



**图3.5 区块链中P2P网络结构**

区块链网络中会通过广播的方式公布交易信息。节点在发起交易时，将生成的交易信息广播给邻居节点，邻居节点验证通过后再进行广播，当全网有51%以上的节点接收到该交易信息，则可以认为该笔交易有效。验证过程中，节点一旦发现交易信息中存在错误，就会直接丢弃该交易信息并停止对其进行广播。矿工节点生成新区块时，在找到满足条件的随机数后也会进行广播，当验证通过以后确认新区块的记账权，并将新的区块添加到主链上。

### 共识层

区块链网络中各个节点的地位是平等的，均会承担区块数据的传输、验证、存储工作。共识机制是分布式系统的核心，主要用于解决分布式节点如何达成共识的问题。在区块链系统中共识机制主要用于各节点之间达成一致，确定交易的记账权和有效性。目前在区块链系统中常用的共识算法有四种，分别是：工作量证明机制（PoW）、权益证明机制（PoS）、股份授权证明机制（DPoS）和拜占庭容错机制（PBFT）。以上四种共识算法在去中心化程度、容错节点比例、性能效率、资源消耗、安全性、可监管性以及应用场景方面的对比如表3.2所示。

**表3.2 不同共识算法的特点**

**Table 3.2 Characteristics of Different Consensus Algorithms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **共识算法** | **PoW** | **PoS** | **DPoS** | **PBFT** |
| 去中心化程度 | 完全 | 完全 | 完全 | 半中心化 |
| 容错节点比例 | 50% | 50% | 50% | 33% |
| 性能效率 | 低 | 较高 | 高 | 高 |
| 资源消耗 | 高 | 中 | 低 | 低 |
| 安全性 | 高 | 低 | 较高 | 高 |
| 可监管性 | 弱 | 弱 | 弱 | 强 |
| 应用场景 | 公有链 | 公有链 | 公有链 | 私有链/联盟链 |

### 合约层

区块链系统中的数据层、网络层和共识层是区块链的底层，分别承担数据表示、数据传播和数据验证的任务。合约层建立在区块链底层之上，封装了各类脚本代码、算法以及智能合约，是实现区块链可编程性和操作数据的基础。区块链通过合约层为其他各类应用提供接口，极大的扩展和丰富区块链的功能，是完成复杂的业务逻辑的基础。

## 智能合约

20世纪90年代学者Nick Szabo首次提出了智能合约的概念，指出智能合约是一套以数字形式定义的承诺，包括合约参与方以及在其上执行这些承诺的协议。智能合约首次被提出后并没有得到广泛的应用，主要原因是缺少一个可信的环境来支撑智能合约的可靠运行。区块链技术出现以后，人们发现区块链与智能合约天然契合：区块链具有去中心化、不可篡改、去信任、开放性等特征，可以为智能合约提供一个理想的可信执行环境；智能合约丰富了区块链的功能，使得区块链能够满足复杂的应用需求。

当前区块链是智能合约最主要的运行环境，所以当前的智能合约通常是指运行在区块链上的智能合约。文献中将运行在区块链上的智能合约称为狭义的智能合约并进行了定义：智能合约是运行在分布式账本上的计算机程序，该程序是由事件驱动的，具有状态的，能够根据预置的响应条件和规则自动完成信息交换、价值转移和资产管理等任务，可以封装、验证、执行分布式节点复杂行为。典型的智能合约模型如所示，从图中可以看出，智能合约在部署成功后会被记录到区块中，当接收到外部数据或有外部事件发生时，智能合约将根据预置的条件和规则进行响应，完成相应的动作并将结果记录到区块链中。



**图3.6 智能合约模型**

运行在区块链上的智能合约继承了区块链三个方面的特性：公开透明、不可篡改、永久运行。

1. 公开透明。智能合约部署成功后，其代码以及运行时产生的上链数据均会被记录到区块链账本中，而区块链对任何人都是开放的，因此任何人都可以查看区块链上运行的智能合约的代码和数据。
2. 不可篡改。区块链上的任何数据都是不可以篡改的，位于区块链上的智能合约代码和运行时产生的上链数据也是不可篡改的。
3. 永久运行。区块链由大量的对等节点共同维护，具有极高的容错性，部分节点失效不会影响区块链系统的正常运行，也不会造成智能合约停止运行。

当前主流区块链系统中，智能合约的实现技术不尽相同，但其本质是区块链上的可执行代码。智能合约的代码通常在沙箱中执行，通过沙箱技术能够隔离和限制合约所使用的资源。区块链中的沙箱按其实现方式的不同可以分成两种：虚拟机和容器。以太坊是第一个支持智能合约的区块链平台,其采用轻量级的虚拟机（Ethereum Virtual Machine，简称EVM）来执行智能合约代码，EVM支持的智能合约语言有Solidity、Serpent、LLL等，通过这些编程语言能够实现复杂多样的智能合约和去中心化应用。联盟链中使用较多的Hyperledger Fabric，其采用Docker容器来执行智能合约，代码执行效率较高。在Fabric中，智能合约也被称为链码（Chaincode，链上代码），可以使用Go语言和Java语言来开发链码。中从执行环境、开发语言、特点等方面对不同的区块链平台的智能合约进行对比。

**表3.3 不同区块链平台上智能合约对比**

**Table 3.3 Comparisons of Smart Contracts on Different Blockchain Platforms**

|  |  |  |  |
| --- | --- | --- | --- |
| **区块链平台** | **执行环境** | **编程语言** | **特点** |
| Ethereum | 虚拟机EVM | Solidity,Serpent,LLL | 图灵完备、占用资源少、代码执行效率低 |
| Fabric | Docker容器 | Go,Java | 资源消耗多、代码执行效率高 |
| Corda | 虚拟机JVM | Kotlin,Java | 有强大的基础设施、大量开发工具、成熟的集成能力 |
| QTUM | 虚拟机Qtum-x86 | C++,C,Go,Rust | 有标准的程序库、内存空间大、调用速度快 |
| RSK | 虚拟机RVM | Solidity | 图灵完备、兼容EVM |

## 公有链上构建木马命令控制信道的优势

从上述内容可知，区块链具有去中心化、去信任、不可篡改、匿名性、开放性等特征，这些特性使得区块链具备构建健壮、可靠的木马命令控制信道的优势，能够解决传统木马命令控制信道单点失效、命令认证不完备的问题。

（1）公有链是去中心化的，部分节点关闭或宕机不会影响区块链网络的正常运行；同时，公有链是开放的，任何节点均可自由加入区块链网络参与账本维护工作。在公有链上构建的木马命令控制信道中，区块链网络是C&C中心，木马受控端可以通过任何方式接入区块链网络中获取控制命令，能够有效解决单点失效的问题。

（2）公有链是去信任的，依托于数字签名技术，只有拥有私钥的用户才能操作对应账户中的资产、数据；同时，公有链通过哈希函数、时间戳等技术确保数据的安全，防止被篡改。在公有链上构建的木马命令控制信道中，木马受控端只会接收和执行特定账户发布的控制命令，能够准确判断命令的有效性，具有较强的认证能力；同时，区块链的不可篡改性能够确保信道中接收数据的原始性和完整性。

（3）任何用户均可在公有链中创建账户，账户不与用户的真实身份信息关联，能有效隐藏攻击者的身份信息。

在公有链上构建木马命令信道能够降低硬件成本，不需要维护庞大的网络基础设施，C&C信道寄生于公有链中，具备公有链的所有特征，其健壮性依赖于公有链的健壮性；同时，木马通信数据混杂于海量的交易信息之中，不容易被发现，具有较好的隐蔽性。

## 本章小结

本章从区块链的发展、区块链的原理和智能合约三个方面来详细介绍区块链技术。首先介绍了区块链发展演进的三个阶段以及每个阶段的具体特征；比较了公有链、私有链、联盟链三种不同类型的区块链的定义和特点。其次分别阐述了区块链中数据层、网络层、共识层、合约层的功能以及各层中的关键技术，从技术层面分析了区块链实现去中心化、去信任、不可篡改、匿名性、开放性的原理。然后阐述了区块链上的智能合约模型、智能合约的执行环境及实现方法。最后分析了在公有链上构建木马命令控制信道的优势。

# 区块链上木马命令控制信道研究

在第二章中介绍了多种协议实现木马命令控制信道的方法，分析了基于不同协议构建的木马命令控制信道中存在的缺陷：IRC协议、HTTP协议是C/S架构的，中心C&C服务器一旦失效，基于其实现的木马命令信道将随之关闭，造成木马控制端与受控端之间无法通信，控制者将丢失对目标主机的控制权；纯P2P协议虽然解决了单点失效的问题，但其认证机制不够完备，容易受到Bootstrap、Index Poisoning、Sybil、Peer-list等攻击，造成命令控制信道瘫痪而无法工作。在第三章中从技术层面介绍了区块链技术的底层技术原理，分析在公有链环境中构建木马命令控制信道的优势。本章主要研究在公有链上构建安全、稳定、可靠的木马命令控制信道的方法，分析信道实现过程中面临的问题并提出相应的解决方案。

## 木马命令控制信道模型设计

利用区块链进行通信的木马运行，依次经历木马植入、区块链网络接入和接受控制三个阶段。

（1）木马植入阶段攻击者通过漏洞攻击、邮件附件携带、网页挂马、捆绑伪装等方式将木马受控端程序植入目标主机。

（2）区块链网络接入是指木马受控端程序在目标主机上运行之后，连接到区块链网络中，并将目标主机的基本信息（例如：主机名、IP、MAC、系统类型）发送给木马控制端。控制端与受控端之间无连接，不直接进行通信，通信数据以交易的形式在区块链网络中传播。区块链网络作为控制端与受控端之间通信的中转站，能有效提高木马的隐蔽性。

（3）接受控制阶段是指木马控制端向目标主机发送控制指令，目标主机接收到指令后进行响应。木马的通信数据存放在区块链上，由于区块链是开放的，所有用户都可以加入区块链网络并读取区块链上的数据，所以需要对通信数据进行加密处理以消除木马控制端与木马受控端之间通信流量特征，增强木马的隐蔽性和通信数据的机密性。

如图4.1所示，本课题中设计的基于区块链通信的木马系统分为三层：木马层、交易层和网络层。



**图4.1 区块链木马总体设计**

（1）木马层主要负责控制端与目标主机之间的指令交互，控制端与目标主机之间通信时不直接建立连接，将通信数据嵌入交易信息中，通过区块链上交易的方式实现数据的传输。

（2）交易层中通过转账交易或智能合约的方式传输木马通信数据，完成木马控制，从而在区块链上实现木马命令控制信道。

（3）网络层主要是区块链节点之间通过P2P协议同步区块信息共同维护区块链的运行，在区块中记录了交易的详细信息。

本课题主要研究木马控制端与目标主机之间的通信数据在区块链上隐蔽、安全、可靠传输的问题，并在区块链上构建健壮的木马命令控制信道。由于本课题是在现有的公有链上开展的研究，因而研究的内容主要集中在木马层和交易层。通信数据在控制端与目标主机之间的传输过程如所示。



**图4.2 区块链木马通信模型**

从图4.2中可以看出，本课题中木马命令控制信道主要分为四个过程及其关键技术：数据隐私保护、数据分片还原、匿名接入和数据传输。

（1）数据隐私保护主要负责对木马通信数据进行加密处理。由于公有链开放性的特性，任何接入区块链的用户均可查看全局账本数据。木马控制端与目标主机之间的通信是通过区块链上的交易来实现的，所有的通信内容均被永久记录在区块链账本中，故必须对控制端与目标主机之间的通信内容进行加密保护，确保木马通信数据的机密性。

（2）数据分片还原负责对木马控制端与目标主机之间的通信数据（如指令数据，文件等）进行分片和重组还原。由于区块链中单笔交易中能够携带的信息是有限的，当需要传输的数据长度超过这个限制时，需要将数据进行切分，确保分片数据能够嵌入到交易信息中。数据发送端完成数据分片后，将不同的分片嵌入到不同的交易信息中；数据接收端从区块链上的交易信息中读取所有数据分片，并对分片进行重组、还原得到加密的通信数据。

（3）匿名接入主要负责对木马控制端的真实地址的保护。控制端通过匿名网络接入区块链网络节点中发布控制命令，能够有效隐藏木马控制端的真实地址信息防止被追踪溯源。

（4）数据传输负责将木马通信数据记录到区块链中。在区块链环境中传输木马通信数据时，数据发送方将木马通信数据嵌入交易信息中，并通过发起交易的方式记录到区块链账本中；数据接收方从区块链账本中读取木马通信数据，从而实现木马控制端与木马受控端之间的通信。

在区块链环境中构建木马命令控制信道的目的是通过区块链的特性来解决单点失效的问题，同时增强木马的认证能力，实现木马控制端与木马受控端之间的数据交换。通过数据隐私保护、数据分片还原、匿名接入、数据传输之间的协同工作，保证木马通信数据在区块链上隐蔽、安全、可靠传输。整个通信过程中控制端与受控端之间不直接通信，通信数据通过区块链网络进行传输；木马控制端使用匿名网络接入到区块链中发布指令，增强了木马控制端的隐蔽性和抗溯源能力。

## 数据隐私保护

由于本课题设计的木马命令控制信道运行于公有链上，任何节点均可以不受限制的加入到公有链中读取全局账本数据。从而为了确保木马通信数据的机密性，木马控制端与目标主机的通信数据在传输之前必须经过加密处理。

数据加密是指以明文数据作为输入通过加密算法和加密密钥将其转化成密文数据，而数据解密与此相反，是指以密文数据作为输入通过解密算法和解密密钥将其恢复成明文数据。根据加、解密密钥是否相同，可以将加密体制分为对称加密体制和非对称加密体制。对称加密算法中通常使用相同的密钥进行加密和解密，具有加解密速度快、效率高等优点，典型的对称加密算法有数据加密标准（Data Encryption Standard，简称DES）、高级加密标准（Advanced Encryption Standard，简称AES）等。非对称加密算法中使用一对公私钥进行加密和解密，其中公钥是公开的，用于加密；私钥只有解密者才持有，用于解密。以人类当前掌握的知识和计算能力，很难通过公钥推算或者猜解得到对应的私钥，非对称加密算法安全性高，但加解密速度慢，不适合用于对大量数据的加密操作，典型的非对称加密算法有RSA加密算法、ECC加密算法等。

### 加解密设计

对称加密体制中，密钥是保密通信的关键，对密钥的妥善管理关系到整个通信数据的安全，通信双方中的任何一方泄露了密钥均会对通信安全造成重大损失。非对称加密体制中加解密过程中分别使用不同的密钥，公钥用于加密，私钥用于解密，通过公钥无法直接得到私钥。与对称加密相比非对称加密更加安全，但非对称加密存在加解密速度慢、效率低的问题。在本课题中通信双方分别是木马控制端和目标主机，其中一个木马控制端可以控制多个目标主机，如所示。



**图4.3 木马一对多控制模型**

采用固定密钥的对称加密算法对木马控制端与目标主机之间的通信数据进行加密时，通过逆向木马程序样本即可还原出加解密算法和密钥，导致永久记录在区块链上的木马通信数据存在被解密的风险，故采用固定密钥的对称加密方案在本课题中达不到保护数据机密性的要求。如果采用非对称加密算法对木马通信数据进行加密，当木马控制端向目标主机传输数据时，木马控制端使用私钥加密数据而目标主机使用公钥解密数据；当目标主机向木马控制端传输数据时，目标主机使用公钥加密数据而木马控制端使用私钥解密数据。由于公钥是公开的，任何人均可以获取到，采用非对称加密方案仅能实现数据的单向加密，任何人都可以通过公钥解密出从木马控制端发送到目标主机的数据，故采用非对称加密方案在本课题中同样达不到完全保护木马通信数据机密性的要求。

以上两种方案在单独使用时都不能完全达到保护木马通信数据机密性的目的，综合对称加密和非对称加密的优点，本课题设计了基于动态密钥的对称加密方案对木马通信数据进行加密保护，动态密钥通过协商的方式产生，在密钥协商过程中涉及的敏感数据则通过非对称加密方案进行保护。在该方案中，动态密钥能有效解决固定密钥泄露造成数据暴露的问题，单次通信的安全性取决于对称加密体制的安全性，而控制端与目标主机之间整个通信的安全性取决于非对称加密体制的安全性。动态密钥是随机生成的，不同目标主机与控制端之间协商生成的密钥不同，从而实现一马一密；同一目标主机在不同时间与控制端协商得到的密钥不同，保证密钥的动态更新。该方案的实施步骤如下：

(1)双方通过协商达成密钥共识；

(2)使用(1)中得到密钥通过对称加解密算法处理木马通信数据；

(3)密钥有效期内，重复(2)；密钥失效后，重复(1)、(2)。直到木马受控端在目标主机上停止运行。

### 密钥协商

密钥协商的目的是通过协商的方式产生通信双方用于加密会话的密钥。协商过程中生成的密钥具有随机性、不可预见性，并且该密钥不能在非安全的信道中明文传输，密钥传输时需要加密处理。

在本课题中木马控制端与目标主机之间不建立网络连接，通信数据在区块链上以交易的方式捎带传输，而区块链上的交易从发起到完成要依次经过如所示的五个步骤。在公有链中对交易数据的打包工作通常是由矿工来完成，打包的过程是一个耗时的操作，需要完成大量的运算。由于每一次交易均会消耗大量时间，故木马控制端与目标主机协商会话密钥的过程中应当减少双方通信的次数。



**图4.4 区块链交易过程**

本课题中在保证密钥安全、完整、不可预测的前提下，密钥协商过程中将通信次数压缩到一次，其实现方法为：木马控制端与目标主机共享一对公私钥，其中控制端保存私钥，目标主机端仅保存公钥；目标主机生成随机密钥并计算其哈希值，使用公钥将生成的密钥及其哈希值一同加密并以区块链交易的方式发送给控制端；控制端通过私钥解密交易信息中的加密数据得到密钥及其哈希值，通过哈希值验证密钥的正确性；验证通过以后密钥协商完成，控制端与目标主机达成密钥共识，数据收发双方使用对称加密算法以及协商生成的密钥加密会话的内容，如图4.5所示。简单来说就是通过非对称加密算法确保密钥传输过程中的安全，利用哈希校验保证密钥的正确性，使用对称加密算法提高木马通信数据加密的速度。



**图4.5 密钥协商过程**

由于加密后的会话密钥记录在区块链账本中，目标主机能从账本中读取并对内容进行验证，同时木马控制端也能够从账本中读取并通过私钥对其进行解密后验证，所以目标主机与木马控制端之间能达成密钥共识。密钥协商完成以后，在一段时间内控制端与目标主机之间使用协商生成的密钥加密会话的内容，目标主机定期更新密钥并通知控制端，同时控制端可以在任何时候要求目标主机更新密钥。频繁更新密钥能够更好的保护双方通信的内容，减小会话密钥被破解后带来的影响。

## 数据分片还原

木马控制端与目标主机之间的通信数据包括指令数据和文件数据，两者均可看作是二进制数据，故可统一按二进制数据进行处理，这些处理包括数据分片和分片数据重组。通过将数据分片传输的方式实现任意长度的木马通信数据在区块链中传输。

数据分片是将数据按固定大小进行分片，所有分片中都附带一个头部信息。头部信息中包含了数据类型标识、数据大小、分片序号、原数据校验值、分片校验值等，其中数据类型标识表明该分片中的数据是指令数据或是文件数据；数据大小用来记录分片前数据的总长度，通过该值和分片的大小可计算总的分片数；分片序号用来标明该分片属于原数据中的第几个分片；原数据校验值是通过对原数据进行计算得到，可用于标明该分片的来源，具有相同原数据校验值的分片来自于同一份原始数据。

数据重组是读取数据分片，将头部信息中具有相同原数据校验值的分片按分片序号进行组合，待全部分片接收完毕后完成数据重组并将重组后的数据内容交由其他模块处理。

### 数据分片算法

数据分片时，将待传输的数据全部或分批读入内存，计算并构造头部信息。根据待传输数据的大小申请发送数组并将构造的头部信息填充到发送数组中每一个元素的开始位置，修改每个元素头部信息中的分片序号为该元素位于发送数组中的位置序号。按顺序从内存读取固定分片大小的原数据并依次填充到发送数组各元素中，紧跟在头部信息之后，计算分片数据的校验值并填充到分片头部中，如此反复，直到将所有的待传输数据填充到发送数组之中。每一个数组元素中的数据结构如所示。



**图4.6 分片数据结构**

发送数组填充完毕，每一个数组元素对应一个完整的数据分片，通过分片头部信息能够唯一确定该分片中数据的来源。头部信息中的数据类型字段标明该分片中的数据来源于文件或者指令内容；数据总长度标明原始数据的总长度，通过该字段值可计算分片的总数量；分片序号表示当前分片属于原始数据的第几个分片，用于分片合并时保证分片按正确的顺序进行组合；数据校验值字段用于区分不同原始数据的分片，数据校验值和数据总长度均相同的不同分片来源于同一份原始数据；分片校验值是当前分片中数据部分的校验值，用于验证分片中数据部分的正确性。

数据分片的算法步骤如下：

1. 将原始数据读入内存，由公式(4-1)计算分片数量*Count*；

(4-1)

式中：*Count*——分片数量；*L*——原始数据总长度；*P*——分片结构数据部分长度。

1. 计算并构造分片的头部信息(不包含分片校验值)；
2. 申请包含*Count*个元素的数组，每个元素的长度为*E*，*E*由公式(4-2)计算得到。

(4-2)

式中：*E*——分片结构总长度；*H*——分片结构头部长度；*P*——分片结构数据部分长度。

1. 从原始数据中读取长度为*P*的数据，计算其校验值并修改头部信息中的分片校验值为该值，将头部信息和读取的数据填充到对应序号的数组元素中；
2. 重复第(4)步操作，直到处理完所有原始数据。

数据分片算法包括三个部分：数据读取、分片信息获取，分片写入。各部分的关系如图4.7所示。



**图4.7 数据分片过程**

### 数据重组算法

数据重组是数据分片的反过程，其目的是将分散的数据分片按序组合在一起构成正确完整的原始数据。从区块链账本中读取特定交易信息并提取数据分片，根据数据头部判断是否所有数据分片已经全部上链。待所有数据分片上链后，提取数据部分并按序重组得到完整的数据。

数据重组的过程依赖于分片数据的头部信息。通过数据校验值和数据总长度可以判断数据是否来源于同一份原始数据；通过分片序号能够确定该分片在原始数据中的位置；通过数据总长度和预置的固定分片长度可以计算出分片的总数量；通过分片校验值可验证分片数据的正确性；通过数据校验值和数据总长度可验证还原后数据的正确性。

数据重组算法步骤如下：

1. 读取分片数据，并按头部信息进行分类，将具有相同数据校验值和数据总长度的分片归成一类；
2. 通过分片校验值验证分片数据的正确性；
3. 计算每一类数据的分片总数；
4. 扫描各类数据分片是否已经完整接收；
5. 对于数据分片完整的数据分类，将分片按序读入内存组合得到原始数据；
6. 根据原数据校验值和数据总大小验证原始数据的正确性；
7. 重复步骤(4)、(5)、(6)，直到把所有能够合并的数据都合并完成。

数据重组算法包括三个部分：分片数据分类、分片数据信息获取、分片数据重组。各部分的关系如图4.8所示。



**图4.8 数据重组过程**

数据分片算法与数据重组算法互为反过程：数据分片算法是将原始数据进行分片并添加分片头部信息的过程，而数据重组算法则是根据分片头部信息，将分片组合还原得到原始数据的过程，如所示。



**图4.9 数据分片与重组关系**

## 匿名接入

区块链通常具有匿名性的特点，用户只与账户地址相对应，而不与用户的真实身份相关联，也就是说用户在使用区块链完成交易时不会暴露用户的真实身份，但是用户通过节点发起交易时可能会暴露节点的网络地址信息。探针节点也属于区块链网络中的节点，其通过主动探测或被动监听的方式搜集区块链网络中节点信息，包括节点IP、节点间的拓扑关系等信息。祝烈煌等人在比特币主网中部署探针节点后，在一周的时间内收集到比特币服务器节点IP 9713个、客户端节点IP 36513个，还搜集到大量额外的身份信息，如表4.1所示[52]。

**表4.1 比特币服务器节点信息**

**Table 4.1 Bitcoin Server Node Information**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IP地址** | **城市** | **国家** | **AS号码** | **机构** |
| 82.XX.XX.68 | 巴伦 | 荷兰 | AS9143 | Ziggo |
| 118.XX.XX.128 | 杭州 | 中国 | AS37963 | 杭州阿里巴巴公司 |
| 122.XX.XX.20 | 杭州 | 中国 | AS4134 | Chinanet |

分析节点间的拓扑关系和区块链上信息的传播路径能够确定信息的始发节点，Koshy等人[53]分析比特币上交易的传播规律后发现可以利用特殊的交易模式找到始发节点。Kaminsky[54]提出：“第一个通知你交易的节点可能就是交易的始发节点”，这种方法的准确性依赖于探针节点的数量，探针节点越多准确率越高。Biryukov等人提出利用邻居节点转发信息的时间排序来定位始发节点，因为邻居节点是第二批转发由始发节点发起的交易的节点，该方法是对Kaminsky提出的算法的优化，进一步提高了准确率，配合其他手段准确率可达60%。Florian等人[56]研究了针对匿名交易系统的侧信道攻击，通过用户P2P节点在收到交易后泄露的时间侧信道和通信模式脆弱性来攻击系统的匿名性和不可关联性，并在Zcash和Monero中进行了验证。

通过以上分析可知，区块链交易的始发节点是可以进行追溯的，因此木马控制端用于发起控制命令的区块链节点也是容易被发现的，该节点一旦暴露，与之直接相连的木马控制端也将随之暴露。木马控制端是木马命令控制信道的控制中心，其安全性决定了整个木马系统的安全性，因此保护木马控制端的真实网络信息防止被追踪溯源是非常必要的。匿名通信是一种隐藏通信实体和内容的隐私保护技术，通常采用数据转发、内容加密、流量混淆等多种技术手段来实现。匿名通信系统本质上是一种提供匿名通信服务的覆盖网络，具有节点发现难、服务定位难、用户监控难、通信关系确认难等特点，通常被用于为用户提供网络匿名接入服务，隐藏用户真实网络地址信息。目前应用比较广泛的匿名通信系统有Tor、I2P等。

本文通过匿名通信网络隐藏木马控制端的真实网络信息，实现在用于发布木马控制命令的区块链始发节点暴露的情况下，无法根据暴露节点的网络通联关系定位出木马控制端的真实网络地址信息，从而提高木马控制端的抗溯源能力，增强木马命令控制信道的隐蔽性，如所示。



**图4.10 控制端通过匿名网络接入区块链**

## 数据传输

### 基于转账交易传输

区块链中的交易信息通常包含有如下四个字段：From、To、Value和Data。From字段用于记录交易的发送方，To字段用于记录交易的接收方，Value字段用于记录转账的货币数量，Data字段用于记录交易时附带的信息，其中From、To和Value字段是必须的，而Data字段通常是可选的，并且用户可定义其内容。图4.11中包含了一笔交易信息的具体内容。



**图4.11 交易信息结构**

由于Data字段内容是用户可自行定义的，故可以将木马通信数据写入到Data字段中，通过转账交易的方式进行传输；同时分别利用From和To字段来定义木马通信数据的发送方和接收方。在所示的交易信息中，通过From字段可以唯一确定交易的发送方，目标主机通过该字段能够判断出该指令的发布者是否为木马控制者；通过To字段可确定交易的接收方，目标主机通过该字段能够判断该指令是否有效，从而判断自身是否需要执行该指令；在发起交易时Value字段可以自行设置；Data字段包含了交易中的附加信息，交易发起方可进行自定义，中该字段解码后的内容为：dir %USERPROFILE%\Desktop /s /o-d。

通过以上分析可知，在交易信息中嵌入木马通信内容时，交易中各字段可进行如中的定义：



**图4.12 木马通信交易一般形式**

当木马通信数据通过交易由木马控制端发送到目标主机时，From字段是木马控制者的账户地址，To字段是目标主机的账户地址，Value字段可自行设置，Data字段中包含指令的具体内容。当木马通信数据通过交易由目标主机发送到木马控制端时，From字段是目标主机的账户地址，To字段是木马控制者的账户地址，Value字段可自行设置，Data字段中则包含了上一条命令执行的结果。综上所述，可以通过区块链上的转账交易来传输木马通信数据，从而实现在区块链上构建木马命令控制信道。

### 基于智能合约传输

运行在区块链上的智能合约具备区块链去中心化、去信任的特性，同时具有可编程、不可篡改的特性。智能合约中可灵活嵌入各类数据，能够安全的完成信息交换、价值转移和资产管理等功能。如所示，区块链中的智能合约通常包括代码、状态和值。智能合约一旦在区块链上部署其代码将写入区块链并且不可以再修改，而状态在交易的驱动下可以进行状态转移。



**图4.13 智能合约的组成**

智能合约中的状态保存于区块链上，支持多种数据类型，能够通过智能合约代码对其进行管理，故可以将木马通信数据保存到智能合约的状态中并通过智能合约代码对其进行读写，从而实现在区块链上传输木马通信数据。

由于任何能够接入到区块链上的用户均可查看和调用区块链上的智能合约，为了确保合约数据的安全，需要对智能合约的调用进行权限控制，阻止未授权的非法调用破坏合约数据，影响木马命令控制信道的正常运行。根据不同的用途，本课题中将用户类型分为管理用户、控制用户和目标用户三类，各类用户的用途如所示。

**表4.2 不同用户的用途**

**Table 4.2 The Function of Different Users**

|  |  |
| --- | --- |
| **用户** | **用途** |
| 管理用户 | 1. 管理控制用户 |
| 控制用户 | 1. 管理目标用户 2. 向目标主机下发指令 3. 获取指令执行结果 |
| 目标用户 | 1. 获取指令 2. 上传指令执行结果 |

从表4.2中可以看出各类用户的用途划分明确，每一类用户能够抽象成一个角色，而每一个角色的操作权限相对固定，所以本课题中采用基于角色的访问控制（Role-Based Access Control，简称RBAC）模型[59]来管理不同用户对智能合约不同接口的访问权限，对用户的权限进行约束。RBAC中对用户权限的授予和取消的方式是通过在角色中添加或删除用户来实现的，首先根据需求定义角色并为其分配权限，然后指派用户为相应的角色。整个访问控制过程分成两部分：权限与角色相关联、角色与用户相关联。用户与权限通过角色实现逻辑分离，简化了权限的管理。课题中角色分为管理角色、控制角色，目标角色，为各角色分配的权限如所示。



**图4.14 不同用户角色权限分配**

## 木马命令控制信道健壮性分析

传统的木马通信时通常采用中心化的C/S架构，木马控制端和受控端直接建立连接进行通信，网络连接特征非常明显很容易暴露。基于P2P网络进行通信的木马在控制端和受控端之间不直接通信具有较高的隐蔽性,但是容易遭受Bootstrap、Index Poisoning、Sybil、Peer-list等攻击[60]。通过区块链网络构建木马命令控制信道时，木马控制端与受控端之间不直接通信，木马通信数据隐藏于海量的区块链交易信息中，由于交易信息在区块链中同步时传输路径是动态变化的，很难追踪网络地址信息，提高了木马的隐蔽性，增强了木马命令控制信道的健壮性。本文在数据隐私保护模块部分采用端到端动态加密方案对木马通信数据进行动态加密，即使安全研究人员从区块链中还原出完整的木马通信数据，也会因为无法解密而获取不到明文数据，增强了通信数据的安全性。在区块链交易信息中可嵌入的数据量是有限的，设计数据分片还原机制对木马通信数据进行分片和重组，保证木马通信数据传输前后的一致性。采用匿名接入技术隐藏木马控制端的真实地址信息，提高了木马的隐蔽性。分别采用普通转账交易和智能合约的方式在区块链上实现木马命令控制信道，丰富了信道实现的多样性。

### 数据隐私保护机制

通过端到端动态加密的方式实现数据隐私保护，任何记录在区块链上的木马通信数据均是以密文的形式存在，即使获取到完整的木马通信数据，也会因为缺少解密密钥而无法还原得到明文数据，从而有效保证了木马通信数据的机密性。

木马控制端与受控端通信前进行密钥协商，木马受控端将随机生成的密钥及其哈希值通过公钥加密后发送给木马控制端，控制端接收到加密的密钥后通过私钥对其解密得到通信密钥及其哈希值，然后使用哈希值对通信密钥进行校验确保密钥的正确性，如果校验不正确，木马控制端将发送指令要求重新协商通信密钥。密钥协商完成后，在一段时间内木马控制端与受控端之间通信的数据均通过协商出来的密钥进行加密。密钥会定期进行更换，并且木马控制端可以在任何时候要求重新协商通信密钥，从而实现动态密钥更新。在不同时间段，木马控制端与不同受控端所使用的通信密钥均不相同，并且所有通信密钥均受公私钥加密体制的保护，有效保证了木马通信数据的机密性，增强木马命令控制信道的安全性。

### 数据分片还原算法

设计数据分片还原算法，通过将木马通信数据进行分片传输，确保单个数据分片的长度不超过交易信息中可嵌入数据的长度限制；在接收端对分片数据进行重组还原得到原始数据，通过哈希校验确保原始数据的正确性。

数据分片头部信息中包含分片重组还原过程中所必须的关键信息，其中包含了当前分片的来源标识、当前分片在原数据中的位置、当前分片的校验值以及原数据的校验值等信息。数据接收端在接收到分片数据之后，首先通过分片的校验值对分片中的数据进行校验，确保分片数据的正确性和完整性；其次按照分片序号和原数据校验值将属于同一份原数据的分片进行重组，重组完成后根据原数据大小和原数据校验值进一步对数据进行校验，确保重组后数据的正确性和完整性。理论上，通过数据分片还原算法可以实现任意长度的木马通信数据在区块链中传输，提高了木马命令控制信道的鲁棒性。

### 匿名接入机制

木马控制端是木马命令控制信道的控制中心，其安全关系到整个木马系统的安全，对木马控制端的保护非常重要。本文中木马控制端与受控端之间不直接相连，在传输木马通信数据时发送端将数据通过交易的方式发送到区块链上，接收端从区块链上读取交易信息中数据。木马控制端通过匿名网络接入区块链中进一步加大了溯源的难度，增强了木马控制端的安全，提高了整个木马系统的安全性和匿名性。

区块链网络中的每一个全节点均包含整个交易数据的完整备份，节点之间的数据同步基于P2P协议。在P2P协议中任意两个节点之间的数据传输的路径是动态变化的，通过接收节点很难直接判断出发送节点。由于公有链是开放的，任何人都可以接到区块链中，防御人员可以通过在区块链中部署探针节点查找木马控制端用于发送控制命令的节点，而始发节点可能会暴露木马控制端的真实地址信息。木马控制端在接入区块链发送控制命令之前，首先接入到匿名网络之中，通过匿名网络提供的动态链路接入区块链网络中，能够有效的隐藏木马控制端的地址信息。防御人员在发现始发节点的情况下，也不能通过始发节点获取到木马控制端的真实地址信息。木马控制端通过匿名网络接入区块链节点中发布指令，增加了其抗溯源的能力，有效保证木马命令控制信道的安全运行。

### 数据传输机制

在区块链环境中构建的木马命令控制信道中，区块链网络是木马通信数据传输的中心，木马控制端和受控端一旦接入到区块链网络中即可发送和接收数据。区块链具有去中心化、开放性、匿名性、不可篡改等特性，任何组织和个人不能阻止区块链的运行或随意篡改区块链账本中的数据，因此木马通信数据可以在区块链上稳定、连续、可靠的传输而不会受到任何监管、控制、封锁。

通过两种方式在区块链上传输木马通信数据，基于转账交易实现的传输方式通用性强，适用于大部分的区块链平台，如比特币、以太坊；基于智能合约实现的传输方式比较灵活，能够进行权限管理和控制。由于所有的木马通信数据均会永久记录到区块链账本中，发送方将通信数据推送到区块链账本中，接收方从区块链账本中读取数据，发送方和接收方不需要同时在线，因此木马控制端与受控端之间可以离线进行交互。在区块链环境中构建的木马命令控制信道具备区块链去中心化、匿名、开放、不可篡改的特性，使得该信道不会因为受到任何组织或个人的监管、控制、封锁而引起木马控制端和受控端之间的通信中断，目标主机可通过任何方式接入区块链网络中接收木马控制端的控制，因此在区块链环境中构建的木马命令控制信道能够显著提高木马通信的健壮性。

## 本章小结

本章研究了区块链环境中构建木马命令控制信道的方法，将信道实现分成数据隐私保护、数据分片还原、匿名接入和数据传输四个过程及关键技术，解决了木马通信数据在区块链环境中隐蔽、安全、可靠传输问题。（1）在数据隐私保护过程中，提出了动态端到端加密方案保护永久记录在区块链账本中的木马通信数据机密性，该方案包括密钥协商和通信数据加密两个关键过程。设计了密钥协商方案，通过一次通信即可达成密钥共识，提高了区块链环境中密钥协商的效率。木马控制端与木马受控端之间不断地通过密钥协商过程达成密钥共识、更新会话密钥，利用会话密钥加密通信数据。结合对称加密技术和非对称加密技术保证数据安全的前提下提高了木马通信数据加密效率。（2）在数据分片还原过程中，基于哈希分类及检验技术设计了数据分片还原算法实现任意长度的木马通信数据在区块链中可靠传输。数据分片算法中为每一个分片添加一个能唯一标识该分片来源的头部信息，数据重组算法中根据头部信息将分片数据进行分类并按序组合还原为原始数据。数据分片算法和数据还原算法互为逆过程，能确保木马通信数据传输前后的一致性。（3）在匿名接入过程中，针对当前的区块链匿名性对抗技术，提出了利用匿名网络技术增强木马控制端的抗溯源能力。在木马控制端接入区块链节点发送控制命令过程中，利用匿名网络隔离木马控制端与区块链节点之间的直接通信，隐藏了木马控制端的真实网络地址，提高了木马控制端的匿名性和抗溯源能力。（4）在数据传输过程中，分别设计了转账交易和智能合约两种方式实现木马通信数据在区块链中传输。在基于转账交易传输木马通信数据的方式中，数据发送方将通信数据嵌入交易信息中并通过发起交易的方式将其记录到区块链账本中，数据接收方从区块链账本中读取通信数据，该方式通用性强，能在大部分的区块链系统中运行。在基于智能合约传输木马通信数据的方式中，数据发送方将通信数据植入到智能合约状态中，数据接收方从智能合约状态中获取通信数据，利用智能合约代码实现对通信数据的动态管理和权限控制，提高了信道工作效率和安全性。

# 区块链上木马命令控制信道的实现

第四章中研究了在区块链上构建木马命令控制信道的方法，通过数据隐私保护、数据分片还原、匿名接入和数据传输四个过程来保证信道的隐蔽性、安全性、可靠性、健壮性。以太坊是目前应用最广泛、最知名的智能合约框架，本章主要对以上四个过程进行实现，并在以太坊Ropsten测试网络中完成实验验证以及实验结果分析。

## 实验环境部署

本文中涉及的实验环境包括开发环境和测试环境。在开发环境中完成中四个过程的开发工作，并在测试环境中完成功能测试，分析实验结果。

本文的开发工作主要包括木马命令控制信道的实现以及智能合约的开发。使用python语言编程实现木马命令控制信道的各功能，使用solidity语言编写智能合约，开发过程中使用ganache工具来模拟区块链网络以提高开发测试的效率。所有功能开发完成后，在以太坊Ropsten测试网络中完成进一步实验测试，分析实验结果。下文将详细介绍开发环境和测试环境中的软硬件配置。

### 开发环境

**表5.1 开发环境软硬件配置**

**Table 5.1 Development Environment Hardware and Software Configuration**

|  |  |
| --- | --- |
| **硬件配置** | **软件配置** |
| * CPU：Intel Core i5 1.60 GHz * 内存：8GB 2400 MHz DDR4 * 网卡：100/1000M 自适应无线以太网卡 * 显卡：NVIDIA GeForce MX150 1983 MB | * 操作系统：Microsoft Windows 10 Home * Visual Studio Code：v1.40.1 * Node.js：v10.15.3 * Truffle：v5.0.19 * Solc：v0.5.0 * Ganache：v2.1.2 * Python：v3.7.14 * Web3.js：v1.0.0-beta. |

Node.js是一个基于Chrome V8引擎的JavaScript运行环境。

Truffle是针对基于以太坊的Solidity语言的一套开发框架，其本身基于JavaScript。

Solc是Solidity语言的编译器，其本身基于JavaScript。

Ganache是一款以太坊节点仿真软件，能够帮助开发者快速进行以太坊DApp的开发与测试。

Python是一种跨平台的计算机程序设计语言，本课题中用来实现木马系统。

Web3.js是一个用于与以太坊节点进行交互的库。

### 测试环境

**表5.2 测试环境软硬件配置**

**Table 5.2 Test Environment Hardware and Software Configuration**

|  |  |  |
| --- | --- | --- |
| **主机** | **硬件配置** | **软件配置** |
| 主机1  （Ropsten节点） | * CPU：2核 * 内存：8 GB * 网卡：1000M | * 操作系统：ubuntu 18.04 x64 * Docker：v18.09.7 * Geth：v1.9.11 * Ropsten测试网络 |
| 主机2  （Ropsten节点） | * CPU：2核 * 内存：8 GB * 网卡：1000M | * 操作系统：ubuntu 18.04 x64 * Docker：v18.09.7 * Geth：v1.9.11 * Ropsten测试网络 |
| 主机3  （木马控制端） | * CPU：2核 * 内存：8GB * 网卡：1000M | * 操作系统：ubuntu 18.04 x64 * Tor * 木马控制端程序 |
| 主机4  （目标主机） | * CPU：2核 * 内存：8GB * 网卡：1000M | * 操作系统：ubuntu 18.04 x64 * 木马受控端程序 |

## 数据安全传输

本节主要实现木马通信数据在区块链中安全、可靠传输，这些数据包括指令数据和文件数据。数据从发送端到达接收端的传输过程中，会被永久记录在区块中，由于区块链是公开访问的，任何人都可以读取区块中的数据，因此需要对数据进行加密保护。在区块链中，单笔交易信息中能够嵌入的外部数据是有限的，当传输数据的长度超过区块链的限制时，需要采用分片传输的方式来进行传输，并通过分片重组算法对接收到的数据分片进行重组还原。木马通信数据在传输过程中的形态变化如所示。



**图5.1 木马通信数据在传输过程中的形态变化**

### 动态加解密

数据隐私保护模块通过端到端动态加密的方式确保数据传输过程中的机密性，其实现的关键是密钥协商算法。密钥协商过程中会生成随机的、不可预测的密钥，协商完成后木马控制端与木马受控端之间通过该密钥来加密会话内容，该过程如所示。



**图5.2 端到端动态加密实现**

在图5.2中木马控制端与木马受控端共享一对公私钥对，其中木马控制端中保存私钥，木马受控端中仅保存公钥。在协商过程中，木马受控端将随机生成的密钥及其哈希值通过RSA算法和公钥进行加密后发送到区块链账本中。木马控制端从区块链账本中读取加密后的密钥，通过RSA算法和私钥解密得到明文密钥及其哈希值，通过比对哈希值的方式对密钥的正确性进行校验。如果验证不通过则向木马受控端发送明文指令要求重新协商密钥。如果验证通过则双方达成密钥共识，之后木马控制端和木马受控端通过协商产生的密钥加密会话内容。

协商生成的密钥在一段时间内有效，超过密钥有效期时需要重新进行密钥协商；在密钥有效期内控制端与受控端可以不受限制的通过中的③、④、⑤和⑥进行交互，实现木马控制；木马控制端可以在任何时间要求重新协商密钥。

### 数据分片还原

本部分内容包括两个部分：数据分片和分片数据重组。分片数据重组的过程依赖于分片头部，根据分片头部信息将不同分片数据分类后按序进行组合，并对重组得到的数据的正确性进行校验。本课题中设计的分片结构如所示：



**图5.3 分片数据结构设计**

分片结构中各字段的信息如所示。

**表5.3 分片头部结构**

**Table 5.3 Segment Head Structure**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **长度(byte)** | **说明** |
| DataType(数据类型) | Byte | 1 | 0：指令数据  1：文件数据 |
| EncryptType(加密方式) | Byte | 1 | 0：AES加密  1：RSA加密  2：明文 |
| TotalLength(总长度) | DWORD | 4 | 原始数据的总长度 |
| SequenceNumber(分片序号) | WORD | 2 | 分片的序号 |
| OriginalDataHash(原始数据校验值) | Byte[32] | 32 | 原始数据的MD5哈希值 |
| SeqDataCrc32(分片校验值) | DWORD | 4 | 分片数据的CRC32校验码 |

#### 5.2.2.1 数据分片算法

数据分片的过程主要是将数据切割后存放到发送数组中，其算法实现如所示。

**表5.4 数据分片算法代码实现**

**Table 5.4 Data Shard Algorithm Code Implementation**

|  |
| --- |
| **算法1 数据分片算法** |
| 输入: data原始数据，len分片长度  输出: 分片数组   1. **function** *SplitData(data, len)* 2. *Array ← []* 3. *Totalength ← length(data))* 4. *count ← Totalength/len + 1* 5. *OriginalDataHash ← MD5(data)* 6. *ii ← 0* 7. **while** *ii < count* **do** 8. *SequenceNumber ← ii* 9. *SeqDataCrc32 ← CRC32(data[ii, ii + len])* 10. *Array[ii] ← (DataType, EncryptType, Totalength, SequenceNumber,* 11. *OriginalDataHash, SeqDataCrc32, data[ii, ii + len])* 12. *ii + +* 13. **end while** 14. **return** *Array* 15. **end function** |

#### 5.2.2.2 分片重组算法

分片重组还原的过程是将数据分片进行分类并按序重组还原得到原始数据的过程，其算法实现如所示。

**表5.5 分片重组算法代码实现**

**Table 5.5 Shard Reorganization Algorithm Code Implementation**

|  |
| --- |
| **算法2 分片重组算法** |
| 输入: piece 分片，len 分片长度  输出: 原始数据   1. **function** *RecombinedData(piece, len)* 2. *totalength ← piece.head.Totalength* 3. *sequenceNumber ← piece.head.SequenceNumber* 4. *buf[totalength] ← []* 5. *count ← tolalength/len + 1* 6. *flag[count] ← [0]* 7. *crc32 ← CRC32(piece.bady.data)* |

续表

|  |
| --- |
| **算法2 分片重组算法** |
| 1. **if***crc32 == piece.head.SeqDataCrc32* **then** 2. *buf + sequenceNumber ∗ len ← piece.bady.data* 3. *flag[sequenceNumber] ← 1* 4. **else** 5. **return** *[]* 6. **end if** 7. **while***true***do** 8. *newpiece ← read(piece.head.OriginalDataHash)* 9. *sequenceNumber ← newpiece.head.SequenceNumber* 10. *crc32 ← CRC32(newpiece.bady.data)* 11. **if***crc32 == newpiece.head.SeqDataCrc32* **then** 12. *buf + sequenceNumber ∗ len ← newpiece.bady.data* 13. *flag[sequenceNumber] ← 1* 14. **else** 15. *continue* 16. **end if** 17. *ii ← 0* 18. **while** *ii < count* **do** 19. *if flag[ii] == 0 then* 20. *break* 21. **end if** 22. *ii + +* 23. **end while** 24. **if***ii == count* **then** 25. *break* 26. **end if** 27. **end while** 28. **return** *buf* 29. **end function** |

#### 5.2.2.3 分片长度选取

选择一个合适的分片长度需要综合考虑时间消耗、费用成本以及传输成功率。分片长度设置的太短会降低传输效率、增加传输成本，设置的过长又会增加传输失败的概率。分片长度选取的原则是在保证传输成功的前提下，尽可能减少时间花费和gas消耗。

（1）转账交易传输方式中的分片长度选取

本部分采用不同的分片长度将1024\*1024字节大小的数据通过交易记录到以太坊Ropsten测试网中，通过比较总的时间花费和gas消耗，确定一个较优的分片长度。首先通过粗粒度的步长筛选出最优区间，然后在最优区间中进一步筛选出一个较优的长度值，并以该值作为分片长度。为了使测量结果更加准确，分别在10个不同的时间段进行实验，并将10次记录的实验结果的期望值作为最终的实验结果。

首先，将10240字节作为起始分片长度并以10240作为步长逐渐增加分片长度。然后，分别记录在各个分片长度下发送1024\*1024字节的数据花费的时间和gas。实验结果如所示。

**表5.6 转账交易方式中不同分片长度对比表（粗粒度）**

**Table 5.6 Comparison Table of Different Shard Lengths in Transfer Transaction Model**

**(Coarse granularity)**

|  |  |  |  |
| --- | --- | --- | --- |
| **分片长度(字节)** | **时间(秒)** | **gas消耗(万)** | **分片数量** |
| 10240 | 3772.299498 | 1903.8520  1813.1360  1793.8200  **1758.5360**  1764.4200  1807.2720  失败 | 103 |
| 20480 | 1830.432504 | 52 |
| 30720 | 1271.99532 | 35 |
| 40960 | 874.713563 | 26 |
| 51200 | 646.612376 | 21 |
| 61440 | **433.582857** | 18 |
| 71680 | 失败 | 15 |

从表中可知当分片大小为40960字节时gas消耗最少，当分片大小为61440字节时消耗的时间最短；随着分片大小的增加，总的时间消耗呈减少趋势；而总的gas花费比较平稳，呈先减少后增加，如图5.4所示。当分片长度到达71680字节时，由于单笔交易消耗的gas超过了Ropsten中的gas限制（8000029），从而造成交易失败。

**图5.4 转账交易方式中不同分片长度实验对比（粗粒度）**

通过以上步骤确定出分片大小的最优区间为40960至71680。由于单次最大发送长度在61440与71680之间，通过二分法逐步减小范围，最终求得Ropsten网络中单笔交易最多包含65433字节的附加数据，但是由于在不同运行状态下gas的消耗有小范围的波动，分片长度不能太过于接近极限值，否则容易出现发送失败的情况，故取最优区间为40960至64512。我们将40960作为起始分片长度并以1024作为步长逐渐增加分片长度，完成同上一步一样的实验测试，测试结果如所示。

**表5.7 转账交易方式中不同分片长度对比表（细粒度）**

**Table 5.7 Comparison Table of Different Shard Lengths in Transfer Transaction Model**

**(Fine Granularity)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **分片长度(字节)** | **时间(秒)** | **Gas消耗(万)** | **分片数量** | **单片平均传输时间(秒)** |
| 40960 | 844.713563 | 1758.5360 | 26 | 32.48884615 |
| 41984 | 807.299161 | 1731.8600 | 25 | 32.29196644 |
| 43008 | 759.7907066 | 1772.8200 | 25 | 30.39162827 |
| 44032 | 780.1763591 | 1741.2288 | 24 | 32.5073483 |
| 45056 | 717.5129313 | 1773.8058 | 24 | 29.89637214 |
| 46080 | 724.2786521 | 1744.0440 | 23 | 31.49037618 |
| 47104 | 754.4332196 | 1774.6848 | 23 | 32.80144433 |
| 48128 | 698.4113084 | 1740.3056 | 22 | 31.74596856 |
| 49152 | 766.3220268 | 1739.6489 | 22 | 34.8328194 |
| 50176 | 676.6015364 | 1722.5243 | 21 | 32.21912078 |
| 51200 | 656.5528077 | 1756.7818 | 21 | 31.26441942 |
| 52224 | 728.6840847 | 1767.6778 | 21 | 34.69924213 |
| 53248 | 697.6363743 | **1706.2556** | 20 | 34.88181871 |
| 54272 | 642.7907159 | 1762.5339 | 20 | 32.1395358 |
| 55296 | 620.3688495 | 1720.8984 | 19 | 32.65099208 |
| 56320 | 608.1175286 | 1752.0280 | 19 | 32.00618572 |
| 57344 | 573.6400688 | 1783.1576 | 19 | 30.19158257 |
| 58368 | 515.9207397 | 1710.1176 | 18 | 28.66226332 |
| 59392 | 584.0495313 | 1721.8003 | 18 | 32.44719618 |
| 60416 | 565.5025452 | 1732.8873 | 18 | 31.41680807 |
| 61440 | 464.9855602 | 1798.1443 | 18 | 25.83253112 |
| 62464 | 481.3522435 | 1716.1676 | 17 | 28.31483785 |
| 63488 | 452.4093164 | 1762.5736 | 17 | 26.61231273 |
| 64512 | **444.2920661** | 1790.4264 | 17 | 26.13482742 |

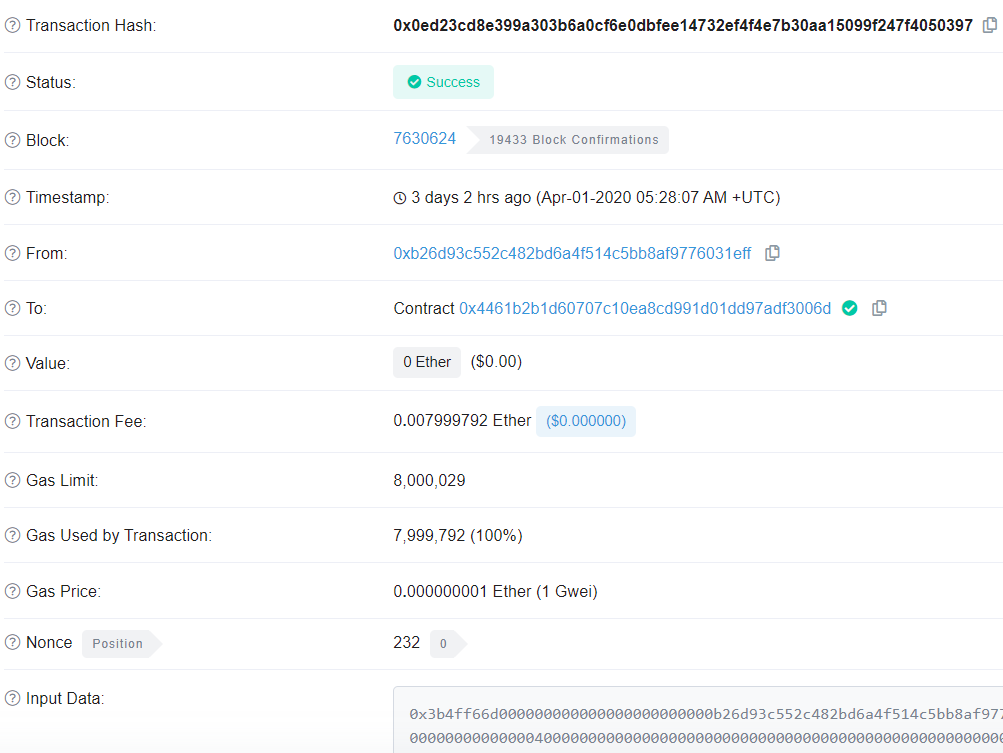
从表中可以看出当分片大小为64512字节时，发送1024\*1024字节大小的数据所花费的总时间最少，分片大小为53248时总gas消耗最少。从图5.5可以看出，在不同分片长度情况下，总的gas消耗基本平稳，没有太大的波动；而随着分片大小为增加，总的时间花费逐渐减少。故本课题中选取64512字节作为长数据传输时的分片大小。

**图5.5 转账交易方式中不同分片长度实验对比（细粒度）**

（2）智能合约传输方式中的分片长度选取

与通过转账交易传输方式相比，使用智能合约传输数据时，不仅要消耗区块链中的存储资源，还要消耗计算资源，所以在消耗相同gas的情况下，通过智能合约能够传输的数据量将少于通过转账交易方式能够传输的数据量。本部分将采用与（1）中相同的方式来选取分片长度。

实验过程中，使用智能合约在传输数据时，当发送的数据长度为12500字节时，其gas消耗几乎达到了Ropsten的上限，如图5.6所示，此时由于gas消耗的波动很容易造成传输失败，故选取500至12000作为分片长度测试区间。通过不同分片长度发送100\*1024字节的数据，从500字节开始测试，以500作为步长逐渐增长到12000，实验结果如表5.8所示。



**图5.6 通过智能合约传输12500字节时的生成交易信息**

**表5.8 智能合约方式中不同分片长度对比**

**Table 5.8 Comparison of Different Shard Lengths in Smart Contracts Method**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **分片长度(字节)** | **时间(秒)** | **Gas消耗(万)** | **分片数量** | **单片平均传输时间(秒)** |
| 500 | 5574.398807 | 7827.1460 | 205 | 27.19219 |
| 1000 | 2730.227166 | 7324.1034 | 103 | 26.50706 |
| 1500 | 1983.071003 | 7038.9729 | 69 | 28.74016 |
| 2000 | 1521.819314 | 7151.8677 | 53 | 28.71357 |
| 2500 | 1217.021855 | 6882.5798 | 41 | 29.68346 |
| 3000 | 1062.147294 | 6957.1005 | 35 | 30.34707 |
| 3500 | 978.419172 | 6951.0420 | 30 | 32.61397 |
| 4000 | 752.4219629 | 6827.8106 | 26 | 28.93931 |
| 4500 | 738.7452708 | 6797.3142 | 23 | 32.11936 |
| 5000 | 664.6827607 | 6897.7167 | 21 | 31.65156 |
| 5500 | 515.3783788 | 6828.0243 | 19 | 27.12518 |
| 6000 | 533.2797792 | 7061.3514 | 18 | 29.62665 |
| 6500 | 494.1841771 | 6803.6001 | 16 | 30.88651 |
| 7000 | 420.1836835 | 6841.9830 | 15 | 28.01225 |
| 7500 | 437.6215121 | 6846.8414 | 14 | 31.25868 |

续表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **分片长度(字节)** | **时间(秒)** | **Gas消耗(万)** | **分片数量** | **单片平均传输时间(秒)** |
| 8000 | 383.8686757 | 6759.5788 | 13 | 29.52836 |
| 8500 | 397.0589133 | 7187.6428 | 13 | 30.54299 |
| 9000 | 375.3212696 | 7029.8856 | 12 | 31.27677 |
| 9500 | 339.9037008 | 6784.0465 | 11 | 30.90034 |
| 10000 | 352.4547697 | 7146.2578 | 11 | 32.04134 |
| 10500 | 330.2784283 | 6825.8830 | 10 | 33.02784 |
| 11000 | 348.7969277 | 7134.9630 | 10 | 34.87969 |
| 11500 | **330.389262** | **6717.8241** | 9 | 36.70992 |
| 12000 | 343.9168791 | 6995.9979 | 9 | 38.21299 |

从图5.7可以看出采用不同分片长度传输100\*1024字节的数据时，消耗的总gas在7000万上下浮动；分片长度为11500字节时，消耗的总gas最少，并且花费的总时间最短。综合考虑gas消耗和传输性能，课题中选取11500字节作为智能合约的分片长度。

**图5.7 智能合约方式中不同分片长度对比**

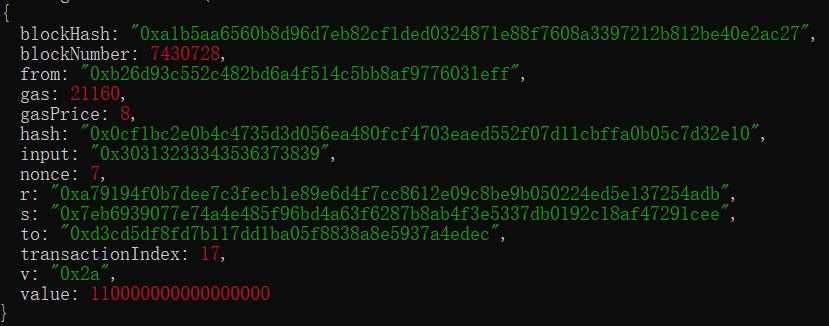
综合以上两种方式来看，与通过转账交易传输数据相比，使用智能合约传输数据的成本是非常昂贵的，在传输相同数据的情况下，其成本可达转账交易的40倍。

### 通信数据上链

木马控制端与木马受控端通信时，发送方将通信数据发送到区块链中，接收方从区块链中读取通信数据。本节通过两种方式将木马通信数据发送并记录到区块链账本中。

#### 5.2.3.1 转账交易的方式

转账交易的方式主要是利用交易结构中的可选字段，在发送交易前将数据嵌入到可选字段中，交易确认时可选字段的内容也将一并记录到区块链账本中。在以太坊中，交易信息的可选字段为input字段。图5.8是一笔真实交易信息，在input字段中嵌入的消息为：0123456789。



**图5.8 交易信息**

如图4.3所示，本课题中采用的是一对多的木马控制模型，即一个木马控制端可以控制多个目标主机。各目标主机使用独立的区块链外部账户来发送数据，并通过轮询遍历的方式从区块链中获取该账户收到的交易信息，然后从交易信息中提取通信数据。由于区块链中的区块是按照时间先后顺序组合相连而成的，因此每次遍历完成以后，下一次只需要遍历新增的区块即可。木马控制端与木马受控端的运行机制如所示。



**图5.9 木马运行机制**

木马控制端通过本地数据库维护所有目标主机的信息，包括外部账户地址、系统类型、IP地址、MAC地址、通信密钥等。其中外部账户可用于标识目标主机，通信密钥是通过协商过程产生的，用于加密会话内容。在目标主机上运行的木马受控端程序中硬编码了公钥和木马控制端所使用的外部账号地址（以下称为：控制账户），并且只会接收从该地址发送过来的指令数据。

木马受控端的运行机制如下：

1. 目标主机上第一次运行木马受控端程序时，首先将随机生成的密钥通过公钥加密后发送给控制账户；然后将目标主机的系统类型、IP地址、MAC地址等信息通过密钥加密后发送到控制账户。
2. 目标主机轮询读取新增区块中的交易记录，提取从控制账户发送给自己的交易内容，读取input字段中的分片数据，记录当前处理的区块高度；待所有分片接收完毕，重组还原数据并检验数据的正确性；通过密钥解密数据得到指令内容。
3. 根据指令的内容完成相应的操作，如更新密钥、执行命令、上传文件等；指令执行完毕后得到执行结果，将结果数据加密后分片发送给控制账户。

木马控制端的运行机制如下：

1. 控制端轮询读取新增区块中的交易记录，提取发送到控制账户的交易内容，读取input字段中的分片数据，记录当前处理的区块高度；根据分片的头部信息将分片进行分类，对分片接收完毕的分类进行重组和还原，检验数据的正确性。
2. 根据分片头中的加密类型确定数据是否是新的密钥，如果是则通过私钥解密数据得到新的密钥，将新的密钥记录到数据库中；否则根据发送账户地址从数据库中提取相应的密钥，通过密钥解密得到明文数据；判断明文数据是否是注册信息，如果是则将目标主机的信息记录到发送账户地址对应的数据库记录中；否则将明文数据的内容输出。
3. 选择需要进行控制的目标主机，将指令内容通过协商的密钥加密后分片发送给目标主机对应的外部账户。

#### 5.2.3.2 智能合约的方式

相对于转账交易的方式，通过智能合约来传输木马通信数据在实现方法上更加灵活，程序不需要处理底层的区块数据，即不需要对区块进行遍历和提取交易中的数据。在此方式中，通信数据存储在智能合约的状态中，用户通过调用智能合约接口读取或写入状态，状态变量是永久存储在合约中的值。本节根据具体的业务逻辑，结合Solidity语言的特性，将用户角色、目标信息、数据类型等内容定义成合约中的结构体，并分别通过状态变量和功能函数来引用和处理，依靠合约事件实现合约消息主动推送。在以太坊中，事件是合约内外沟通的桥梁，合约运行时触发事件产生的结果会记录在EVM日志中，合约外部可以通过Web3监听事件的内容，并判断下一步需要完成的操作。

本文中的智能合约主要用于在区块链上构建木马命令控制信道，实现木马控制端与木马受控端之间全双工的通信。该智能合约的主要功能是通过RBAC模式对不同类型用户的权限进行控制，为授权用户提供向信道注入命令或接收命令的接口服务，阻止未授权用户的非法调用，增加信道实现的灵活性和安全性。下文将介绍智能合约中关键的变量声明、函数定义、事件消息。

1. 变量声明

智能合约中变量的声明也叫状态的声明，本文中通过合约状态来记录信道中的通信实体和通信内容，将不同用户信息存入不同的结构中，便于用户权限的管理。本文中声明的全局变量有Owner、ControlRole和ClientRole，其数据类型和说明如表5.9所示。

**表5.9 全局变量定义**

**Table 5.8 Global Variable Definition**

|  |  |  |
| --- | --- | --- |
| **变量名** | **数据类型** | **说明** |
| Owner | address | 记录部署合约的用户，即合约的拥有者或管理用户。 |
| ControlRole | mapping(address => bool) | 控制用户角色，记录控制用户地址及其有效性。 |
| ClientRole | mapping(address => CCData) | 目标用户角色，记录目标主机的相关信息。 |
| ClientAddr | mapping(uint=>address) | 记录目标主机的账户地址，主要用于协助遍历ClientRole中的信息。 |

mapping(key => value)，是由键值对组成的哈希表，初始化时每一个key对应的value值中的每一个字节均为0，所以当key不存在时其对应的value值为0，故在合约中未通过授权的用户默认是不具有任何一个用户角色对应的权限的。在使用时不需要预先申请分配存储空间，理论上其存储的元素数目可达2256个。

CCData是自定义的数据结构，用于存储目标主机的信息，包括其与木马控制端之间的通信数据，如所示如下：



**图5.10 目标主机信息数据结构**

每一个CCData结构中维护一个目标主机信息，通过目标主机的账户地址进行索引；CCData中使用mapping类型的变量Command和Result记录信道中的木马通信数据，单个mapping类型最多可拥有2256个元素，几乎可以不受限制的使用，简化了存储的管理；Cstart和Ccounts中分别记录特定目标主机当前可接收的分片位置和分片数；Rstart和Rcounts分别记录特定目标主机已经回传的数据分片的位置和分片数；key记录了特定目标主机与木马控制端之间会话密钥的密文信息。

1. 函数定义

本文中智能合约的功能是为木马命令控制信道中的数据提供读写操作的接口服务，同时限制未授权用户对接口的使用。本文在智能合约中关键的功能函数如所示。

**表5.10 函数定义**

**Table 5.9 Function Definitions**

|  |  |  |
| --- | --- | --- |
| **函数名** | **参数** | **说明** |
| SetControlRole | address newControlUser;//需要修改权限的控制账户地址  bool isBlock;//授予或取消权限 | 只允许管理用户修改控制用户的权限 |
| SetClientRole | address newClientUser;//需要修改权限的目标账户地址  bool isBlock;//授予或取消权限 | 只允许控制用户修改目标用户的权限 |
| SetCommand | address clientAddr;//命令接收者  string command;//命令内容 | 只允许控制用户向特定目标主机发送命令 |
| GetCommand | 无 | 目标用户只能获取发送给自己的命令 |
| SetResult | string result;//命令执行结果 | 目标用户上传命令执行结果 |
| GetResult | address clientAddr;//目标主机账户地址 | 只允许控制用户调用，获取目标账户地址返回的命令执行结果 |
| ClearCommand | address clientAddr;//目标账户地址 | 清空合约中特定目标主机尚未执行的命令 |
| RegisterInfo | CCData clientinfo;//目标主机信息 | 目标用户注册，也可用于更新密钥和主机信息 |

Solidity语言中通过内置特殊变量msg.sender能够获取当前调用者的账户地址，本文中的智能合约通过该地址即可判断该账户对应的权限。在SetCommand函数的实现中，通过该值来判断当前调用者是否有权限发布指令，其关键实现代码如所示。其他函数中的权限管理代码类似，主要判断调用者账户是否属于特定的用户角色。



**图5.11 智能合约中发布命令的函数**

1. 事件消息

通过监听合约事件能够了解合约的运行情况，本文在智能合约中定义多个事件，如，木马控制端监听特定事件能够实时掌握信道的运行状况，及时获取信道中的数据。

**表5.11 事件定义**

**Table 5.10 Event Definition**

|  |  |  |
| --- | --- | --- |
| **事件名** | **参数** | **说明** |
| ERegister | address clientinfo //目标主机的账户地址 | 定义目标主机注册事件，实时掌握目标主机的状态 |
| ECommand | address clientinfo //目标主机的账户地址  address senduser //发布指令的控制用户 | 定义发送指令的事件，实时了解命令发送情况 |
| EResult | address id //目标主机的账户地址 | 定义结果回传事件，实时掌握命令执行情况 |

智能合约中通过关键字event来定义事件，如: event EventName(address bidder, uint amount)；事件定义后在合约函数中通过关键字emit来触发事件，事件触发后会在EVM日志中记录事件的内容，通过Web3监听事件可及时获取事件的触发情况。

### 匿名接入

本文在4.4节中介绍了针对区块链网络匿名性的对抗技术，通过这些技术防御人员能够发现特定交易的始发节点。在本文实现的木马命令控制信道中木马控制端通过RPC的方式连接到区块链节点中向目标主机发布控制命令，如果木马控制端与区块链节点之间直接相连，防御人员对木马命令控制信道进行溯源时，容易定位出发送木马控制命令的始发节点，进一步结合网络溯源技术追溯到木马控制端的真实IP地址。针对上述问题，本部分通过Tor匿名通信技术隐藏木马控制端的网络地址，提高信道的抗追踪溯源的能力。实现的过程如下：

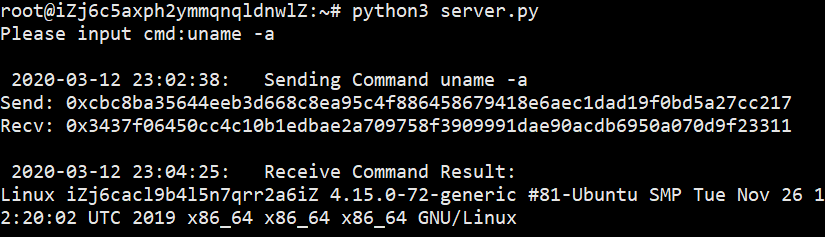
1. 在木马控制端本地运行Tor网络代理程序。
2. 木马控制端通过Tor提供的代理服务连接区块链节点。Tor网络动态转发木马控制端的请求内容，并通过频繁更换出口地址实现对木马控制端地址信息的隐藏。
3. 木马控制端通过区块链节点发送指令或接收数据。

## 功能测试及分析

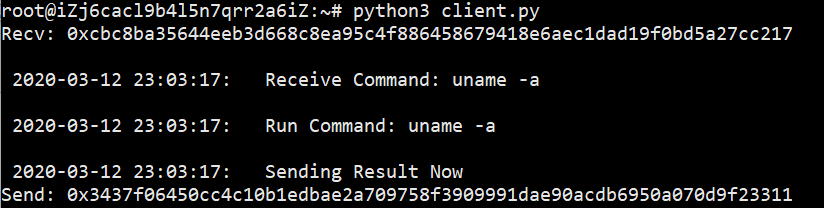
### 信道基本功能测试

#### 5.3.1.1 数据传输功能测试

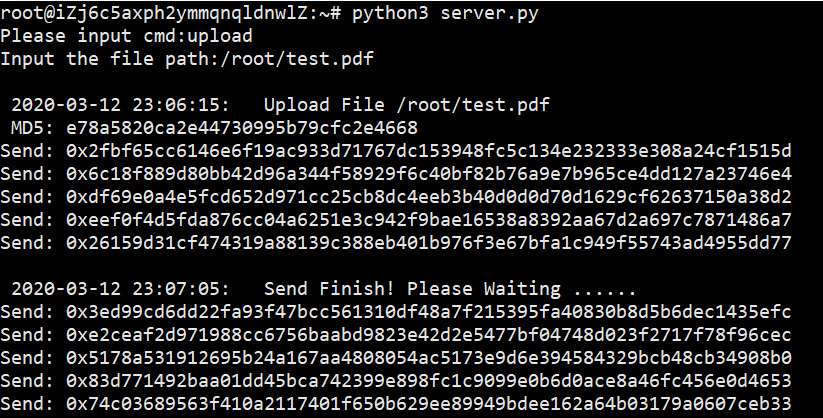
本文中实现的木马命令控制信道的基本功能是传输木马控制端与木马受控端之间的通信数据，这些数据包括指令交互数据和文件数据。和图5.13是一次完整的指令交互过程，和图5.15是一次完整的文件传输过程。在各图中Send字段之后的内容为发送数据时产生的交易的哈希值，通过这些哈希值能够在网站https://ropsten.etherscan.io/中查询得到具体的交易信息。实验结果表明，在区块链环境中传输木马通信数据，实现木马命令控制信道的方案是可行的。



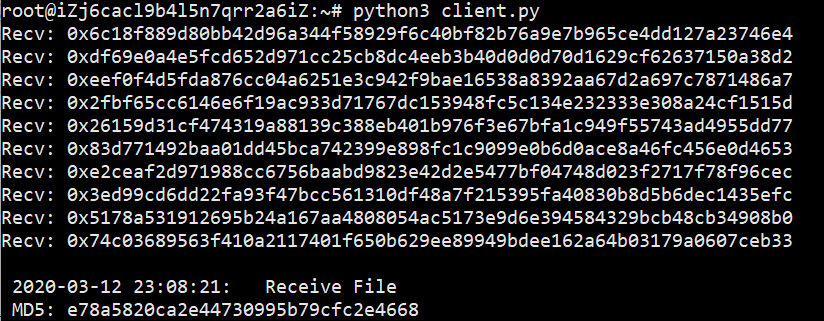
**图5.12 木马控制端发送指令**



**图5.13 木马受控端接收执行指令**



**图5.14 木马控制端上传文件**



**图5.15 木马受控端接收文件**

#### 5.3.1.2 智能合约测试

本文用于实现木马命令控制信道的智能合约中，采用RBAC模型限制不同类型用户对合约不同接口的操作权限，防止未授权用户向信道中注入指令数据。本部分实验主要是对不同类型用户的权限进行测试，验证智能合约中权限控制的有效性，测试方法如下：

第一步，在智能合约部署前，定义事件EPrivilegeTest，如图5.16所示；在合约各函数的各个分支结束前调用该事件，通知外部程序当前函数运行状态。



**图5.16 事件定义**

第二步，部署智能合约。通过合约管理用户A（即合约创建者）将用户B添加到控制用户角色中，用户C调用RegisterInfo接口注册成为目标用户，用户D不属于合约中的任何用户角色。

第三步，监听EPrivilegeTest事件，实时掌握合约函数的调用情况。

第四步，依次通过用户A、用户B、用户C和用户D调用智能合约的不同功能接口，根据监听到的事件判断函数是否调用成功。

实验结果如表5.12所示，该结果表明本文中实现的智能合约中的权限控制机制是有效的，能够有效阻止未授权的调用，防止信道中的木马通信数据被非法操作。

**表5.12 不同用户权限测试**

**Table 5.16 Permission Tests with Different User**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **权限内容** | **管理用户A** | **控制用户B** | **目标用户C** | **其他用户D** |
| 管理控制用户 | 是 | 否 | 否 | 否 |
| 发布指令 | 否 | 是 | 否 | 否 |
| 获取执行结果 | 否 | 是 | 否 | 否 |
| 管理目标用户 | 否 | 是 | 否 | 否 |
| 清空指令 | 否 | 是 | 是 | 否 |
| 获取指令 | 否 | 否 | 是 | 否 |
| 上传执行结果 | 否 | 否 | 是 | 否 |

### 信道指标测试

#### 5.3.2.1 信道传输效率测试

Wu等人[63]将信道传输效率定义成最大可能的无错信息速率，其衡量单位为bit/s（bps，比特每秒），计算方法如公式(5-1)，*C*越大则表明信道的传输效率越高。

(5-1)

式中：*C* ——信道传输效率；*N(t)* ——时间*t*内传输的信息总量；*t* ——时间。

实验过程中，采用串行发送队列发送数据时，转账交易传输方式和智能合约传输方式的数据传输效率*C*分别为18880.84 bit/s和2479.68 bit/s，基本能满足木马控制端与木马受控端之间的交互需求。当采用并行发送队列发送数据时，能够显著增加数据传输效率，是在不同并行数的条件下，分别传输数据时的数据传输效率*C*。

**表5.13 传输效率比较**

**Table 5.15 Transmission Efficiency Comparison**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **并行数** | | **1** | **2** | **4** | **8** | **16** | **32** |
| 传输效率C(bit/s) | 转账交易方式 | 18880.84 | 31517.30 | 56469.80 | **87297.71** | 85551.76 | 83477.86 |
| 智能合约方式 | 2479.68 | 4376.76 | 5162.29 | **5613.92** | 5589.37 | 5419.21 |

从表5.13中可以看出，在转账交易传输方式中，当并行数为8时传输效率C最大，达到87297.71 bit/s；当并行数大于8时，传输效率变化很小。在智能合约传输方式中，其传输效率的变化基本与转账交易传输方式相同，最大传输效率达到5613.92 bit/s。综上来看，以太坊Ropsten中，使用转账交易传输木马通信数据时传输效率较高，是智能合约的15倍。

#### 5.3.2.2 信道可靠性测试

Houmansadr等人[64]提出将比特误码率（bit error rate，简称BER）作为衡量信道可靠性的测量指标，误码率越低则信道的可靠性越高。误码率可通过公式(5-2)和(5-3)计算得到。

(5-2)

(5-3)

式(5-2)中：*BER* ——比特误码率；*k* ——传输信息总量；*m(i)* ——原始信息；*m’(i)* ——解码信息。

式(5-3)中：*x* ——原始信息；*y* ——解密信息。

本部分通过在木马命令控制信道中传输文件来测试信道的误码率*BER*。为了提高测试的准确性，实验过程中采用并行数分别为2、4、8、16的发送队列来传输文件，同时在不同并行条件下发起的交易数量均不低于100笔。通过二进制逐位对比的方式比较发送的数据和接收的数据是否匹配，统计不一致的位数，计算出错的比例得到误码率。实验结果显示不同并行条件下信道的误码率均为0。分析区块链环境下木马命令控制信道的误码率低的原因是：区块链网络是由所有节点共同维护的，节点在收到交易信息时，首先通过数字签名、哈希校验、时间戳等技术对交易信息的正确性和合法性进行校验，验证通过后才会广播给邻居节点，否则丢弃该交易信息并终止该交易信息的传播，所以最终能够记录到区块链上的数据都是正确的。

#### 5.3.2.3 控制端匿名性测试

区块链具有匿名性，但不能完全保证匿名，在4.4节中介绍了几种国内外研究的用于攻击区块链匿名性的方法，这些方法的主要思想是通过在区块链网络中部署探针节点，根据交易的传播规律定位发现交易的始发节点。本课题中木马控制端通过RPC的方式连接到区块链节点中发送木马控制数据，如果木马控制端直接与区块链节点建立连接，两者之间存在通联关系，防御人员在对木马命令控制信道进行溯源时，一旦定位到发送木马控制命令的始发节点，则木马控制端也将被追溯到。为了在始发节点已经暴露的情况下，防御人员无法根据始发节点的网络通联关系追溯到木马控制端的真实网络地址信息，5.2.4节中使用Tor匿名网络技术来保护木马控制端的真实地址信息，本节将对该方法的有效性进行测试。

以太坊Ropsten测试网中节点提供的RPC服务基于标准的http协议，调用RPC时请求的内容为json格式，如所示。在实验环境中Ropsten节点的IP为47.56.174.200，RPC服务端口为8545，木马控制端主机的IP地址为47.75.153.211。



**图5.17 RPC请求格式**

实验测试过程如下：

第一步，在以太坊Ropsten节点上运行网络数据包嗅探程序tcpdump，捕获所有发送到地址47.56.174.200:8545的数据包。

第二步，在将要运行木马控制端的主机上开启Tor匿名网络代理服务，木马控制端通过Tor代理服务接入Ropsten节点持续发送木马控制命令。

第三步，采集一段时间内的Ropsten节点8545端口接收到的通信数据，过滤出满足中条件的数据包，提取这些数据包的源IP地址。



**图5.18 数据包过滤条件**

将实验中提取的所有源IP地址与47.75.153.211进行比对，未发现与47.75.153.211相同的IP。实验结果表明，木马控制端通过Tor代理服务接入Ropsten节点发送木马通信数据时，其真实IP地址没有暴露，该方法能够有效隐藏木马控制端的真实网络地址，增强信道的安全性。

## 本章小结

本章是对第四章研究内容的实现，在以太坊Ropsten网络中分别通过转账交易传输方式和智能合约传输方式构建了安全、隐蔽、可靠的木马命令控制信道，完成了对信道功能和性能方面的测试及结果分析。首先，详细介绍了信道各部分功能的实现细节，并给出关键的伪代码和数据结构。在动态端到端加密方案中，RSA算法有效保护了木马控制端与目标主机密钥协商过程中密钥传输的安全，AES算法提高了木马通信数据加密的效率。在数据分片还原算法中，将原始数据的MD5值作为分片数据的来源标识，是数据还原时对分片数据分类的依据以及数据还原后对原始数据正确性校验的关键。在转账交易传输方式，将木马通信数据嵌入交易信息input字段中进行传输，目标主机可通过收发双方账户地址判断木马命令的有效性。在智能合约传输方式中，将木马通信数据记录到智能合约状态中，通过智能合约代码进行动态管理，提高了信道工作的效率。利用Tor匿名网络隐藏真实网络地址，提高了木马控制端的抗溯源能力。其次，分别完成了对两种不同方式实现的木马命令控制信道的测试。在基于转账交易实现的木马命令控制信道中，综合传输成功率、手续费消耗、时间花费方面的因素选取64512字节作为数据传输时的分片长度，在此条件下信道的最大传输效率为87297.71比特/秒，误码率为0。在基于智能合约实现的木马命令控制信道中选取11500字节作为数据传输时的分片长度，在此条件下信道的最大传输效率为5613.92比特/秒，误码率为0。综合来看，两种方式都能高可靠的传输木马通信数据，其中转账交易传输方式具有更强的传输能力，其最高传输效率达到智能合约传输方式的15倍。

# 结论与展望

## 结论

区块链技术给人们生活带来便利的同时，也出现了恶意使用区块链的情况。本论文从攻击者的角度将区块链技术应用于木马通信技术中，通过在区块链上构建木马命令控制信道还原出一种在木马攻击活动中恶意使用区块链传输控制命令的技术场景，旨在指导信息安全研究人员特别重视此类攻击，并研究相应技术对策，提前做好防御预案，对区块链技术的安全应用具有较高的研究价值和前瞻性。本论文主要研究了在区块链环境中构建木马命令控制信道的方法和关键技术，并针对构建过程中存在的问题提出相应的技术解决方案。本文的主要研究工作内容及成果如下：

（1）提出了动态端到端加密方案，用于保护木马命令控制信道中的数据安全。在基于非对称加密技术的密钥协商过程中，木马控制端与木马受控端之间通过一次通信即可达成密钥共识，有效地提高了高延迟的区块链环境中密钥协商效率；基于对称加密技术的数据加密过程中，利用协商生成的随机密钥和预置的对称加密算法对木马通信数据进行加密保护，能有效地提高数据加密效率；密钥的动态更新增强了永久记录在区块链账本中的木马通信数据机密性。该方案能有效地保护区块链账本中存放的木马通信数据。

（2）设计了数据分片还原算法，保证任意长度的木马通信数据在信道中可靠传输。发送方传输数据时，通过数据分片算法将数据进行分片，为每一个分片添加一个能够唯一标识该分片来源的头部信息；接收方接收到数据分片后，根据分片头部信息通过分片重组算法将分片数据进行分类并按序组合，还原得到原始数据。该算法基于哈希分类及检验技术，能确保木马通信数据传输前后的一致性。

（3）利用匿名网络技术，增强了木马控制端的抗溯源能力。木马控制端通过匿名网路接入区块链节点中并发送控制命令，利用匿名网络提供的动态转发功能实现对真实网络地址的隐藏，有效地提高了木马控制端的匿名性和抗溯源能力。

（4）通过转账交易和智能合约两种方式实现了木马通信数据在区块链中传输。将木马通信数据嵌入交易信息中，通过构造交易信息的方式实现了木马通信数据在区块链上隐蔽传输，该方式通用性强、传输效率高；将木马通信数据植入智能合约状态中，通过状态转移的方式实现了木马通信数据传输，该方式可通过智能合约代码动态管理木马通信数据，简化了数据处理工作，有效提高了信道的工作效率及安全性。

本论文最后在以太坊Ropsten测试网络中分别通过转账交易和智能合约构建了木马命令控制信道，并对信道的功能和性能进行了测试及结果分析。实验结果表明在区块链上构建木马命令控制信道的方案是可行的，两种不同方式实现的信道最大传输效率分别为87297.71 bits/s和5613.92 bits/s，能较好地满足木马控制端与木马受控端之间的通信需求；经过多次反复测试，信道的误码率均为0，具有极高的可靠性；通过Tor匿名网络技术能有效地隐藏木马控制端的真实地址信息，增强了木马命令控制信道的匿名性和抗溯源能力。

## 展望

本文研究了如何在区块链上构建安全、可靠、匿名的木马命令控制信道的方法及关键技术，并在以太坊Ropsten测试测网中进行了实现。随着区块技术的快速发展和广泛应用，使用区块链平台的成本和难度必然会越来越低，此类恶意使用区块链平台的活动可能会长期存在。针对以上情况，未来的研究工作主要集中在以下几个方面：

（1）实验过程中使用的是以太坊Ropsten测试网络，如果将木马命令控制信道放到以太坊主网中运行，信道运行的成本将会非常昂贵。为了降低数据传输时所产生的费用、提高信道的传输效率，下一步可以研究其他运行成本更低的木马命令控制信道实现方式，如寻找运行成本更低的区块链平台或通过链上链下相结合的方式来实现。同时本文在以太坊Ropsten中选取分片长度时，采用了简单的统计模型，实验结果普适性不够，下一步可以研究动态选取分片长度的方案，根据区块链的不同运行状态选取与之适配的分片长度值，提高信道的传输效率。

（2）本文从攻击者角度开展研究，还原一种恶意使用区块链平台传输木马控制命令的应用场景，但并未针对此类木马攻击活动提出具体的防御方法。下一步可以从防御的角度开展研究，分析此类恶意使用区块链的行为，研究有效的检测方法和对抗措施。

# 参考文献

1. 李可, 方滨兴, 崔翔, 等. 僵尸网络发展研究[J]. 计算机研究与发展, 2016, 53(10): 2189-2206.
2. Li F, Lai A, Ddl D. Evidence of advanced persistent threat: A case study of malware for political espionage[C]//2011 6th International Conference on Malicious and Unwanted Software. IEEE, 2011: 102-109.
3. 江健, 诸葛建伟, 段海新, 等. 僵尸网络机理与防御技术[J]. 软件学报, 2012 (2012 年 01): 82-96.
4. Liang J, Naoumov N, Ross K W. The Index Poisoning Attack in P2P File Sharing Systems[C]// INFOCOM. 2006: 1-12.
5. Wang P, Wu L, Aslam B, et al. A systematic study on peer-to-peer botnets[C]//2009 Proceedings of 18th International Conference on Computer Communications and Networks. IEEE, 2009: 1-8.
6. Davis C R, Fernandez J M, Neville S, et al. Sybil attacks as a mitigation strategy against the storm botnet[C]//2008 3rd International Conference on Malicious and Unwanted Software (MALWARE). IEEE, 2008: 32-40.
7. D. Roffel and C. Garrett. (2014) A novel approach for computer worm control using decentralized data structures.
8. Ali S T, McCorry P, Lee P H J, et al. ZombieCoin: powering next-generation botnets with bitcoin[C]// International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2015: 34-48.
9. Ali S T, McCorry P, Lee P H J, et al. ZombieCoin 2.0: managing next-generation botnets using Bitcoin[J]. International Journal of Information Security, 2018, 17(4): 411-422.
10. J. Sweeny. (2017) Botnet resiliency via private blockchains. [Online]. Available: https://www.sans.org/ reading-room/whitepapers/ covert/botnet-resiliency-private-blockchains-38050
11. A. Pirozzi and P. Paganini. (2018) Experts presented botchain, the ﬁrst fully functional botnet built upon the bitcoin protocol. [Online]. Available: https://securityaffairs.co/wordpress/77395/malware/ botchain-botnet-bitcoin-protocol.html
12. O. Zohar. (2018) Unblockable chains. [Online]. Available: https://github.com/platdrag/ UnblockableChains
13. M. Malaika. (2017) Botract. [Online]. Available: https://sector.ca/ wp-content/uploads/presentations17/ Majid-Malaika-Botract SecTor.pdf
14. Baden M, Torres C F, Pontiveros B B F, et al. Whispering Botnet Command and Control Instructions[C]//2019 Crypto Valley Conference on Blockchain Technology (CVCBT). IEEE, 2019: 77-81.
15. Kurt A, Erdin E, Cebe M, et al. LNBot: A Covert Hybrid Botnet on Bitcoin Lightning Network[J]. arXiv preprint arXiv:1912.10617, 2019.
16. Zhong Yi, Zhou Anmin, Zhang Lei, et al. DUSTBot: A duplex and stealthy P2P-based botnet in the Bitcoin network[J]. PloS one, 2019, 14(12).
17. 李彦峰, 丁丽萍, 吴敬征, et al. 区块链环境下的新型网络隐蔽信道模型研究[J]. 通信学报, 40(05):71-82.
18. 诸葛建伟, 韩心慧, 周勇林, et al. 僵尸网络研究[J]. 软件学报(3):230-243.
19. Kartaltepe E J , Morales J A , Xu S , et al. Social Network-Based Botnet Command-and-Control: Emerging Threats and Countermeasures[C]//Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings. DBLP, 2010
20. Riden J．Know your Enemy:Fast -Flux service networks ［R/OL］．Ann Arbor,MI:The Honeynet Pro j ect,2008 ［2016-06-01］．http://www.honeynet.org/book/export/html/130
21. Nazario J, Holz T. As the net churns: Fast-flux botnet observations[C]// International Conference on Malicious & Unwanted Software. 2008.
22. 武岳, 李军祥. 区块链P2P网络协议演进过程[J]. 计算机应用研究, 2019(10).
23. Jain S, Mahajan R, Wetherall D. A Study of the Performance Potential of DHT-based Overlays[C]// USENIX Symposium on Internet Technologies and Systems. 2003.
24. Ratnasamy S , Francis P , Handley M , et al. A Scalable Content-Addressable Network[J]. ACM SIGCOMM Computer Communication Review, 2001, 31(4).
25. Stoica I, Morris R, Karger D, et al. A scalable peer-to-peer lookup service for internet applications[J]. Lab. Comput. Sci., Massachusetts Inst. Technol., Tech. Rep. TR-819, 2001.
26. Antony Rowstron. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems[C]//IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing. ACM, 2002.
27. Nicholas J. A. Harvey, Michael B. Jones, Stefan Saroiu,等. SkipNet: A Scalable Overlay Network with Practical Locality Properties[C]//Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems - Volume 4. 2003.
28. 郭静. 基于P2P的Botnet研究与实现[D]. 北京邮电大学, 2010.
29. Thangavel K , Pethalakshmi A . Dimensionality reduction based on rough set theory: A review[J]. Applied Soft Computing, 2009, 9(1):1-12.
30. Leibowitz N, Ripeanu M, Wierzbicki A. Deconstructing the kazaa network[C]//Proceedings the Third IEEE Workshop on Internet Applications. WIAPP 2003. IEEE, 2003: 112-120.
31. Clarke I, Sandberg O, Wiley B, et al. Freenet: A distributed anonymous information storage and retrieval system[C]//Designing privacy enhancing technologies. Springer, Berlin, Heidelberg, 2001: 46-66.
32. 李可. 基于行为分析的僵尸网络对抗技术研究[D]. 北京邮电大学, 2017.
33. 中国信息通信研究院和可信区块链推进计划.区块链白皮书(2019年)
34. S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system," 2008.
35. 孙毅, 范灵俊, 洪学海. 区块链技术发展及应用:现状与挑战[J]. 中国工程科学, 2018, v.20(02):35-40.
36. Buterin V. A Next Generation Smart Contract & Decentralized Application Platform (2013) Whitepaper[J]. Ethereum Foundation.
37. 中华人民共和国工业和信息化部. 中国区块链技术和应用发展白皮书(2016) [R]. 北京: 中华人民共和国工业和信息化部, 2016.
38. 李燕,马海英,王占君.区块链关键技术的研究进展[J].计算机工程与应用,2019,55(20):13-23+100.
39. 王子岳. 面向供应链溯源的区块链系统研究与优化[D]. 2019.
40. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends[C]// 6th IEEE International Congress on Big Data. IEEE, 2017.
41. Proof of Stake[EB/OL].[2019-01-08].https://delegated-proof-of-stake-consensus/.
42. Larimer D.Delegated proof of stake consensus[EB/OL].[2018-03-22]. https://bitshares.org/technology/ delegated-proof-of-stake-consensus/.
43. Castro M, Liskov B. Practical Byzantine fault tolerance[C]//OSDI. 1999, 99(1999): 173-186.
44. 章峰, 史博轩, 蒋文保. 区块链关键技术及应用研究综述[J]. 网络与信息安全学报, 2018, v.4；No.29(04):26-33.
45. 黄凯峰, 张胜利, 金石. 区块链智能合约安全研究[J]. 信息安全研究, 2019, 5(03):10-24.
46. Szabo N. Smart contracts: building blocks for digital markets[J]. EXTROPY: The Journal of Transhumanist Thought,(16), 1996, 18: 2.
47. 欧阳丽炜, 王帅, 袁勇, et al. 智能合约:架构及进展[J]. 自动化学报, 2019, 45(3).
48. 李赫, 孙继飞, 杨泳, et al. 基于区块链2.0的以太坊初探[J]. 中国金融电脑, 2017(6).
49. 贺海武, 延安, 陈泽华. 基于区块链的智能合约技术与应用综述[J]. 计算机研究与发展, 2018, 55(11):112-126.
50. The Linux Foundation Smart contract engine[EB/OL].[2018-11-19].https:// www.hyperledger.org/ projects/ fabric.
51. 高峰, 毛洪亮, 吴震, 等. 轻量级比特币交易溯源机制[J]. 计算机学报, 2018, 41(5): 989-1004.
52. 祝烈煌, 高峰, 沈蒙, 等. 区块链隐私保护研究综述[J]. 计算机研究与发展, 2017: 2170-2186.
53. Koshy P , Koshy D , Mcdaniel P . An Analysis of Anonymity in Bitcoin Using P2P Network Traffic[C]// International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2014.
54. Kaminsky D . Black Ops of TCP/IP 2011[J]. 2011. https://dankaminsky.com/2011/08/05/bo2k11/
55. Biryukov A , Khovratovich D , Pustogarov I . Deanonymisation of Clients in Bitcoin P2P Network[C]//Acm Sigsac Conference on Computer & Communications Security. ACM, 2014.
56. Tramer F, Boneh D, Paterson K G. Remote Side-Channel Attacks on Anonymous Transactions[J].
57. 罗军舟, 杨明, 凌振, 等. 匿名通信与暗网研究综述[J]. 计算机研究与发展, 2019, 56(1): 103-130.
58. Omohundro, Steve. Cryptocurrencies, smart contracts, and artificial intelligence[J]. AI Matters, 1(2):19-21.
59. 陈麒.基于角色与权限继承的访问控制模型解决方案[J].电脑知识与技术,2011,7(04):838-841.
60. 尹捷, 崔翔, 方滨兴, et al. 一种抗污染的混合P2P僵尸网络[J]. 信息安全学报, 2017, 3(1).
61. Atzei N, Bartoletti M, Cimoli T. A survey of attacks on ethereum smart contracts (sok)[C]// International conference on principles of security and trust. Springer, Berlin, Heidelberg, 2017: 164-186.
62. 郭雪梅. 基于区块链智能合约的网络威胁情报共享机制及实现[D]. 北京邮电大学, 2019.
63. Wu J , Wang Y , Ding L , et al. Improving performance of network covert timing channel through Huffman coding[J]. Mathematical and Computer Modelling, 2012, 55(1-2):69-79.
64. Houmansadr A , Borisov N . CoCo: Coding-Based Covert Timing Channels for Network Flows[C]// International Workshop on Information Hiding. Springer Berlin Heidelberg, 2011.

# 攻读学位期间取得的研究成果

**参与的科研项目：**

[1] 人工智能应用区块链基础平台关键技术研究，四川省新一代人工智能重大科技专项项目，项目编号：2019YFG0400，2019.6-2021.6

**发表的论文：**

[1] 以第一作者身份已在《现代计算机》期刊上正式发表《\*\*\*\*\*\*\*\*\*》。