

智能合约安全综述

孟博, 刘加兵, 刘琴, 王潇潇, 郑旭睿, 王德军

(中南民族大学计算机科学学院, 湖北 武汉 430074)

摘要: 区块链为构建社会价值传递和信任机制提供了一种新的技术。区块链的快速发展促进了智能合约与人工智能、大数据、物联网等技术的深入融合, 其安全性受到重点关注。近几年, 区块链与智能合约安全研究取得了较大进展, 基于区块链智能合约, 对智能合约的运行机制、链上安全和链外安全的最新研究成果进行归类、分析、比较、总结和讨论, 并展望了智能合约安全性的研究方向和趋势。

关键词: 区块链; 智能合约; 链上安全; 链外安全; 安全性分析

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2020030

Survey of smart contract security

MENG Bo, LIU Jiabing, LIU Qin, WANG Xiaoxiao, ZHENG Xurui, WANG Dejun

School of Computer Science, South-Central University for Nationalities, Wuhan 430074, China

Abstract: Blockchain provides a new technology for building transmission and trust mechanism of social value. The rapid development of blockchain has promoted the deep integration of smart contract with artificial intelligence, big data and internet of things, so its security has attracted attention. In recent years, researches on security of blockchain and smart contract have made great progress. Thus, based on smart contract on the blockchain, the related works on security of operating mechanism, on-chain smart contract security and off-chain security were classified, analyzed, compared, summarized and discussed. The hot issues of smart contract security in the future were forecasted.

Key words: blockchain, smart contract, on-chain security, off-chain security, security analysis

收稿日期: 2019-09-03; 修回日期: 2019-12-19

通信作者: 孟博, mengscuec@gmail.com

基金项目: 国家自然科学基金(61272497); 湖北省自然科学基金(BZY18001); 中央高校攻关计划专项基金(CZT20013)

Foundation Items: The National Natural Science Foundation of China (61272497), The Natural Science Foundation of Hubei Province (BZY18001), The Key Research Funds for the Central Universities (CZT20013)

论文引用格式: 孟博, 刘加兵, 刘琴, 等. 智能合约安全综述[J]. 网络与信息安全学报, 2020, 6(3): 1-13.

MENG B, LIU J B, LIU Q, et al. Survey of smart contract security[J]. Chinese Journal of Network and Information Security, 2020, 6(3): 1-13.

1 引言

区块链应用了密码学、分布式存储、共识机制、智能合约等技术,建立了一种新型的可量化的信任和激励体系,大大提升了透明度,减少了信任风险,降低了成本,提高了效率,将改变整个社会价值传递的方式和信任机制,因而被认为是颠覆性的核心技术。区块链技术的发展为智能合约提供了可信的执行环境,允许在不依赖第三方的情况下进行可信、可追踪且不可逆的合约交易。智能合约作为区块链的重要组成部分,旨在以信息化方式传播、验证和执行合同的谈判或者履行计算机协议^[1]。智能合约最早在 1995 年由 SZABO 提出^[2],起初被定义为一套以数字形式定义的承诺,包括合约参与方在智能合约上执行承诺协议,其设计初衷是希望将智能合约内置到物理实体中来创造灵活可控的智能资产。随着区块链的进一步发展,应用在区块链上的智能合约被重新定义:智能合约是由事件驱动的、具有状态的、运行在一个可复制和共享的账本上且能够保管账本上资产的程序。智能合约的普及和大规模的应用,必须满足安全性、一致性、完整性、原子性等属性,其中,安全性是一个非常关键的属性。

智能合约安全分为链上安全和链外安全,其中,链上安全重点关注智能合约架构设计安全和代码安全等,链外安全重点关注智能合约与链外数据交互时的安全(在本文,跨链数据被统称作链外数据)。虽然文献[3-7]对区块链的发展状况做

了介绍和讨论,但是较少关注智能合约安全方面的研究成果,因而有必要对智能合约安全性研究成果进行总结、分析和讨论。本文首先介绍和分析了主流的 Ethereum、Hyperledger Fabric 和 EOS 平台上的智能合约运行机制,进而对智能合约链上安全和链外安全的相关研究成果进行归类、分析、比较、总结和讨论,并对未来智能合约的发展及其安全性的研究方向进行了展望。

2 智能合约运行机制

智能合约是区块链的核心构成要素(合约层),对区块链技术的应用和普及具有重要意义。一方面,智能合约是区块链的激活器,不仅为静态的底层区块链数据提供了灵活可编程机制和算法,而且为构建基于区块链 2.0 和 3.0 的应用奠定了基础;另一方面,智能合约的自动化和可编程特性使封装分布式区块链系统中各节点的复杂行为得以实现,这促进了区块链技术在各类分布式系统中的应用。

智能合约是运行在可复制的共享区块链数据账本上的计算机程序,通过交易转账到特定的合约地址以触发智能合约,接受、存储和发送数据,以及控制和管理链上智能资产。智能合约由事件驱动,具有状态和区块链数据的一般特征,如分布式记录、存储和验证,不可篡改和不可伪造等。智能合约运行机制^[8]如图 1 所示。

通常情况下,智能合约经各方认可后,首先以程序代码的形式附着在未经验证的区块上,然

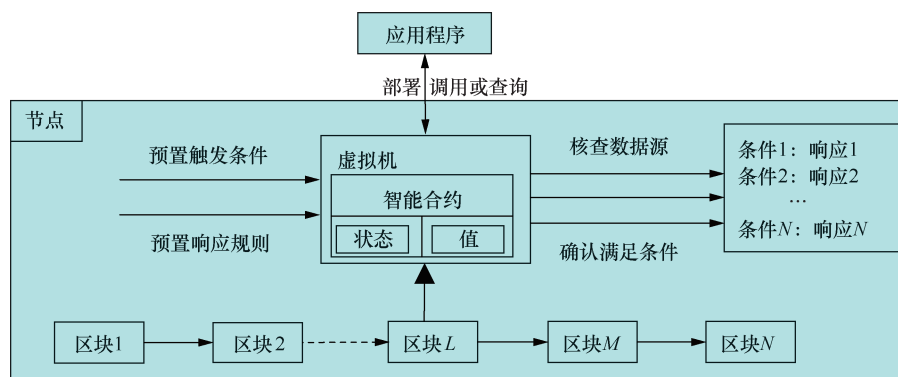


图 1 智能合约运行机制
Figure 1 Operating mechanism of smart contract

后, 经 P2P 网络传播和节点验证后添加到区块链上的特定区块中。智能合约封装了预定义的若干状态和转换规则、触发合约执行的情景(如到达特定时间或发生特定事件等)及相应的响应规则。区块链可实时监控智能合约的状态, 并在核查数据源、确认满足特定触发条件后激活并执行合约。目前, 主流的 3 个智能合约平台有 Ethereum、Hyperledger Fabric 和 EOS。对包含图灵完备智能合约的 Ethereum、针对机构用户进行多方授权交易的 Hyperledger Fabric, 以及面向高性能互联网应用的 EOS 的主要特点^[9-10]进行分析, 如表 1 所示。

Ethereum 秉持开放的理念, 采用公有链形式。区块链系统中每一个节点都可读取、任何节点都能发送交易(transaction)且交易能够获得有效确认, 任何节点都能参与其中的共识过程, 这导致采取公有链形式的 Ethereum 智能合约平台在一定程度上性能较低。EOS 虽然同样采取公有链形式, 但是 EOS 依赖已经在压力测试中展现出每秒 1 万至 10 万笔交易处理能力的石墨烯技术, 且 EOS 使用并行化拓展网络, 因而 EOS 的吞吐量在 Ethereum 的基础上有了很大提升。但公有链也有积极意义: 规则可信, 规则公开透明可预见, 保护用户免受开发者的影响; 访问门槛低, 成为系统节点的任何用户都可以访问, 而在联盟链中, 节点需要得到许可搭建应用。Hyperledger Fabric 采取了联盟链的形式, 只有获得特定许可的节点才能接入网络, 也只有特定的节点才能从事记账参与共识算法, 为交易处理的低延迟和吞吐量大幅增长提供了可能。下面分别对 Ethereum、Hyperledger Fabric 和 EOS 智能合约运行机制进行分析和讨论。

2.1 Ethereum 智能合约

Ethereum 智能合约是区块链 2.0 中的智能合

约的典型代表之一。Ethereum 智能合约存在两种账户: 外部账户和合约账户。外部账户既可以通过创建和使用其私人密钥签署一项交易, 也可以向其他外部账户或合约账户发送消息。两个外部账户之间的消息只是一种价值转移, 但从一个外部账户到合约账户的消息会激活合约的代码, 使它能够执行各种操作(如转移代币、写入内存、生成代币、执行计算、创建合约等)。与外部账户不同, 合约账户不能自行启动新的交易。相反, 合约账户只能根据它们收到的其他交易(从外部账户或从其他合约账户)进行交易。因而, 在 Ethereum 区块链上发生的任何操作都是由外部控制账户的交易引起的。Ethereum 智能合约运行机制^[11-12]如图 2 所示。

交易是连接外部世界与以太坊内部状态的桥梁。交易(包括调用和合约创建)总是由外部账户启动并提交给区块链的, 不同账户之间发生的交易使 Ethereum 智能合约从一个状态转移到另一个状态。首先, 外部账户(或者 Ethereum-Wallet)将编译好的智能合约通过 RPC 接口部署到 Geth 节点上。然后, 当外部账户发起交易时, 外部账户通过 RPC 接口将要写入区块中的信息发送到 Geth 节点(即区块链节点), Geth 节点会将数据发送到本地虚拟机(EVM)中进行运算, 得到运算结果后进行相互验证(区块链共识), 验证成功的结果将写入区块链中。

2.2 Hyperledger Fabric 智能合约

Hyperledger Fabric 智能合约被称为 Chaincode(链上代码)。它有 6 个状态, 分别是 Install、Instantiate、Invocable、Upgrade、Deinstantiate 和 Uninstall。Install 表示链上代码部署到区块链上。Instantiate 表示智能合约进行初始化。Invocable 表示经过初始化后的智能合约进入被调用的状态。联盟链跨多家企业、地区, 合约

表 1 3 个智能合约平台分析
Table 1 Analysis of smart contracts on three platforms

智能合约平台名称	链类型	吞吐量/笔交易·秒 ⁻¹	交易延迟/s	合约语言	隐私保护
Ethereum	公有链	约 100	约 15	Solidity	不支持
Hyperledger Fabric	联盟链	约 100	约 1	Go/Java/Node.js	支持
EOS	公有链	约 10 000	0.5	C++	不支持

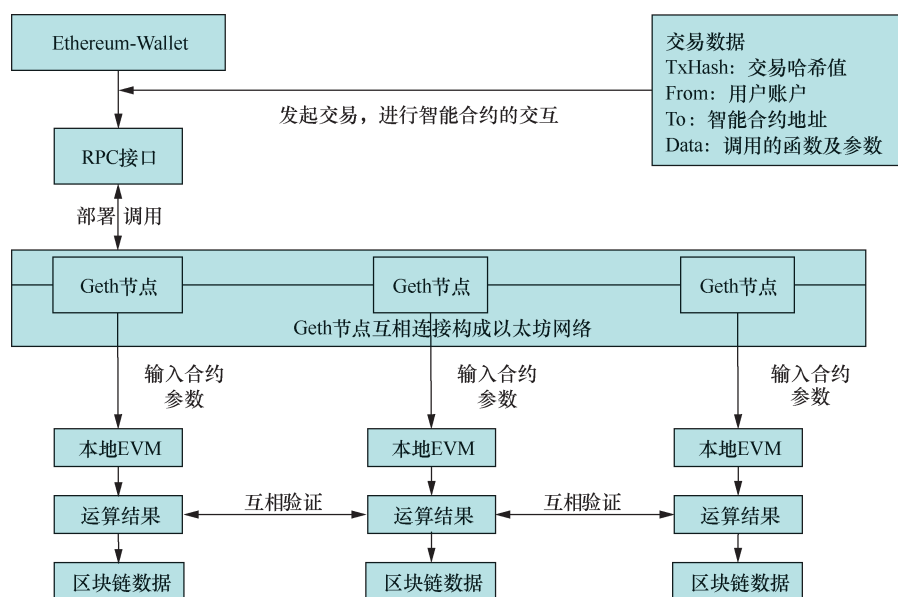


图2 Ethereum 智能合约运行机制
Figure 2 Operating mechanism of Ethereum smart contract

很难保持一致的版本,因而每个合约都有版本号, Upgrade 表示版本升级。Deinstantiate 和 Uninstall 对应智能合约离链。6个状态之间的关系是 Install→Instantiate→Invocable→Upgrade→Deinstantiate→Uninstall。Hyperledger Fabric 中智能合约运行机制^[13-14]如图3所示。

Hyperledger Fabric 智能合约运行机制包含如下7个步骤。

1) 应用程序或者命令行通过 RPC 请求向背书节点发起链上代码的调用请求,背书节点再转发给链上代码执行,应用程序不能直接与链上代码通信。调用之前,应用程序将编写和编译完的智能合约打包上传、部署到背书节点。

2) 背书节点检查对应的链上代码是否启动。

检查的方法是查看本地维护的映射表中是否有指定的链上代码名称和版本的记录。如果没有相关的记录,则通过 Docker 的 API 发起创建或者启动容器的命令。

3) Docker 服务器根据 API 的命令启动链上代码容器,并建立和背书节点的 RPC 接口。如果背书节点接收到请求以后检查链码容器已经启动,则直接跳过第2)步和第3)步。

4) 通过链上代码和背书节点建立的 RPC 连接,转发应用程序调用的请求,链上代码在执行过程中和背书节点有多次的交互。

5) 链上代码执行完以后,调用背书节点的 ESCC 对模拟执行的结果进行背书。

6) ESCC 对模拟执行进行签名,返回背书结果。

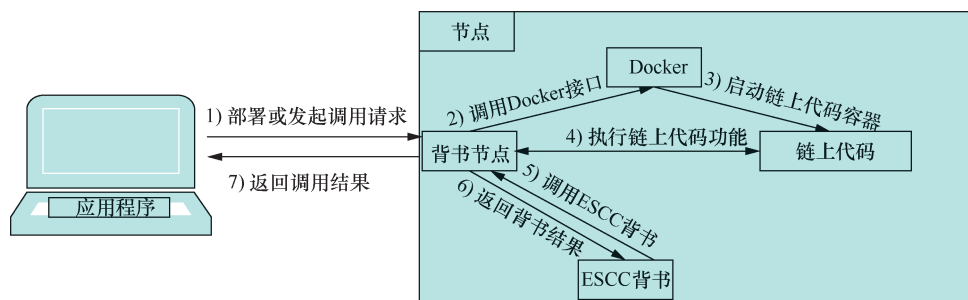


图3 Hyperledger Fabric 智能合约运行机制
Figure 3 Operating mechanism of Hyperledger Fabric smart contract

7) 背书节点返回包含背书节点的背书结果给应用程序, 不再转发给其他背书节点。

如果涉及交易, 在整个交易流程中, 链上代码只参与业务逻辑模拟执行的过程, 后续的交易排序和验证分别由排序服务和记账节点完成。应用程序自由选择背书节点发起请求, 只要最终生成的交易满足背书策略即可。

2.3 EOS 智能合约

EOS 智能合约由一系列 Action 组成。每个 Action 代表一项合约条款, 实现该条款中的具体规则。在智能合约部署阶段, 编译好的智能合约代码通过客户端 (Cleos) 发送到服务器, 由服务器部署在区块链上, 随后其他用户调用、执行该智能合约。在智能合约调用阶段, 由客户端通过 Cleos 命令发送 Action 请求给服务器。每个服务器都有一个 Action 处理函数集合副本, 当客户端发起 Action 请求后, 服务器根据 Action 请求信息, 在区块链上找到对应的智能合约代码, 并将代码加载到内存中, 然后执行。服务器会在本地运行 Action 处理函数, 并相互校验结果, 最后将执行结果返回给客户端。EOS 智能合约运行机制^[15-16]如图 4 所示。

EOS 智能合约运行机制包含如下 4 个步骤。

1) Cleos 将编写好的智能合约代码上传到区块链上。

2) 客户端请求部署智能合约, 解析智能合约名称和 Action 名称。

3) Cleos 请求调用智能合约, 服务器根据 Action 请求信息在 Action 处理函数集合副本中找到相对应的智能合约代码, 并将其加载到内存中执行。

4) 服务器在本地执行 Action 处理函数并相互校验结果后, 将执行得到的结果返回给客户端。

Cleos 将一组 Action 封装成 Transaction 数据包发送给服务器。一个 Transaction 代表一个事务, 一个 Transaction 中包含一个或多个 Action。为保证事务的原子性, 事务内的 Action 要么全部执行, 要么都不执行。

3 智能合约链上安全

智能合约链上安全主要关注智能合约及其与区块链内部要素交互过程中的安全性, 如智能合约架构设计安全、代码安全、运行安全等。下面从智能合约链上代码的架构设计安全 (主要开发框架、设计模式)、代码安全 (代码自身安全性) 以及安全性分析 3 个方面进行总结归纳。

3.1 架构设计安全

智能合约架构设计安全分为开发框架安全和设计模式安全。智能合约在开发过程中采用安全的开发框架^[17-20]以及设计模式^[21], 使其可以按序、

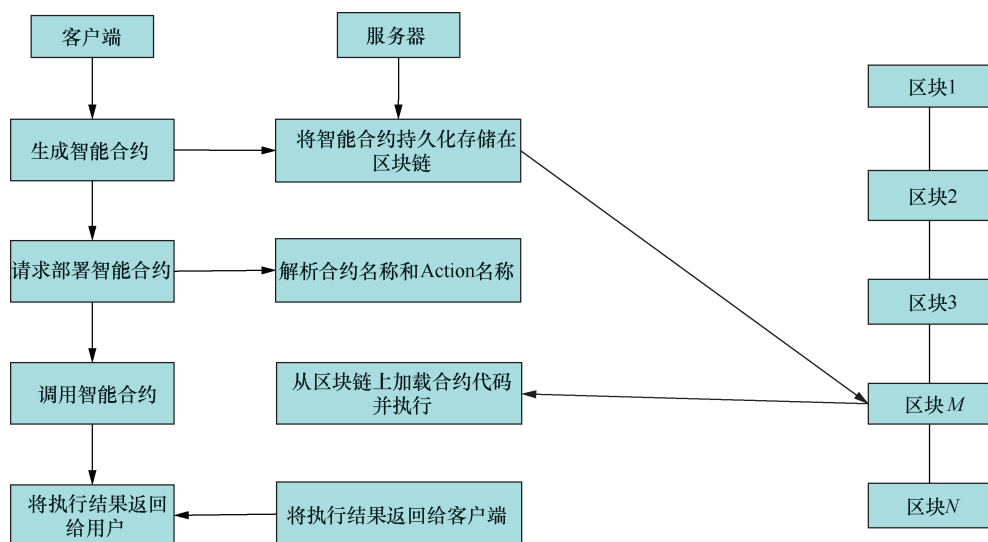


图 4 EOS 智能合约运行机制
Figure 4 Operating mechanism of EOS smart contract

安全、可验证地实施特定的流程,为智能合约正确性和安全性提供支持和保障,防止受到拒绝服务攻击、重入攻击等,并且可以获得代码可重用性和可扩充性,缩短开发周期,提高开发质量。

针对开发框架安全,2018年,Kalra等^[17]设计和实现了一种基于抽象解释和符号执行的智能合约自动形式化验证框架——Zeus。Zeus首先将高级语言编写的智能合约作为输入,并由用户生成模板的正确性和公平性标准。将合约和策略规范转化为低级中间表示形式,编码执行语义正确解释合约的行为。然后,对中间表示形式进行静态分析,确定需要由断言验证的地方。最后,将断言验证后的中间表示形式提供给验证引擎,确保智能合约的安全性。同年,Sánchez等^[18]提出Raziel框架,将安全多方计算和携带证明的代码相结合,实现了智能合约的隐私性、正确性和可验证性。Raziel提供一个编程框架,用来开发实现具有隐私性的智能合约并且进行形式化验证隐私性,分别通过安全多方计算和携带证明的代码来实现和验证隐私性。但是,此方法既增加了智能合约代码量,又增加了计算和存储方面的开销,从而对整体性能和可伸缩性产生影响。同年,Dargaye等^[19]提出基于Coq实现的Cyberlogic信任管理框架。此框架不仅可以生成自然语言生成形式化语言规则并验证法律合约,而且可以将法律合约转化为智能合约Solidity代码。面向交易,2016年,Kosba等^[20]提出Hawk——一种分布式的智能合约系统框架。Hawk中的编译器使用加密原语(如零知识证明)将私有智能合约编译为区块链和用户之间安全的加密协议,进而实现交易隐私性。

基于开发设计模式安全,2018年,Wöhrer等^[21]首先介绍了Ethereum和Solidity及其相关安全性,然后提出了检查效果交互(CEI, checks-effect-interaction)、紧急停止(emergency stop)、减速带(speed bump)、速率限制(rate limit)、互斥(mutex)和余额限制(balance limit)6种设计模式,可以处理开发智能合约过程中的典型安全问题和漏洞。检查效果交互模式通过一定的代码顺序,在最后一步调用外部智能合约,阻止外部智能合约发动重复调用攻击,解决恶意

代码劫持控制流的漏洞。紧急停止模式将紧急停止功能集成到智能合约代码中,由认证方触发以禁用某些敏感功能。减速带模式通过延长执行敏感任务的智能合约的完成时间,解决短时间内某项任务请求执行频率过高的问题。速率限制模式通过降低智能合约一段时间内的执行速率,缓解任务请求繁忙状况,实现智能合约正常运行。互斥模式利用互斥死锁阻止外部调用重新输入调用方函数,防止重入攻击。余额限制模式通过限制智能合约中风险资金的最高金额,降低智能合约受到攻击后造成的金融风险。

3.2 代码安全

智能合约代码安全是智能合约安全的核心组成部分。智能合约代码安全主要涉及类型安全、漏洞挖掘、语言安全、代码静态分析和法律合约与代码一致性。下面从漏洞挖掘、语言安全和法律合约与代码一致性3个方面进行总结归纳。

在漏洞挖掘方面,2019年,付梦琳等^[22]对区块链领域的知名开源软件Dogecoin、Ripple、Litecoin、Dash、Ethereum-Wallet等,通过扫描工具和人工审计,在代码层面发现高危安全漏洞746个,中危漏洞3497个。同年,Natoli等^[23]引入区块链异常(the blockchain anomaly)概念。区块链异常不能中止提交的交易,可能出现Gas双花等漏洞,对智能合约使用者造成严重后果。2017年,Atzei等^[24]首先介绍了Ethereum平台及Solidity语言的安全漏洞,进而将这些漏洞分为3个级别:Solidity、EVM和blockchain。Solidity级别的漏洞主要涵盖调用不明确(call to unknown)、没有足够的Gas发送(gasless send)、异常障碍(exception disorder)、类型转换(type cast)、重入攻击(reentrancy)和保密(keeping secret);EVM级别的漏洞主要包括immutable等类型的Bug、传输过程中网络数据丢失(ether lost in transfer)和堆栈容量限制(stack size limit)等;Blockchain级别的漏洞主要涉及不可预测的状态(unpredictable state)、产生随机性(generating randomness)和时间限制(time constraint)。

在语言安全方面,2018年,Tonelli等^[25]研究了12000多个部署在Ethereum上的智能合约,

介绍了智能合约语言 Solidity 开发合约时代码的 4 种典型特征: 合约声明、地址声明和映射、所有者数据管理和实现块链地址之间的合约和交易的特定代码的函数。基于以太坊, 2018 年, Grishchenko 等^[26]提出了 Ethereum 智能合约典型安全属性的语义特征。智能合约的典型安全属性主要包含调用完整性 (call integrity)、原子性 (atomicity)、可变账户状态的独立性 (independence of mutable account state) 以及交易环境独立性 (independence of transaction environment)。智能合约调用完整性可以防止重入攻击和调用不明确等 Bug; 当智能合约不满足原子性时, 将出现不可预知异常 (mishandled exceptions) 等类型的 Bug; 当智能合约不满足可变账户状态的独立性时, 将出现交易顺序依赖 (transaction order dependency) 和不可预测状态等类型的 Bug; 当智能合约不满足交易环境独立性时, 将出现时间戳依赖 (timestamp dependency)、时间限制 (time constraints) 和产生随机性等类型的 Bug。

在法律合约与代码一致性方面, 2016 年, Frantz 等^[27]首先使用 Adico 语言建模自然语言描述的制度规则、法规和法律条文的语义; 然后, 将其转换成 Solidity 语言代码。同年, Clack 等^[28]提出智能合约模板的概念, 将智能合约定义为可自动执行的协议。智能合约模板包含法律条款和参数, 其中, 每个参数包含一个标识符、一个类型和一个可选的值。通过制定的法律条款和参数来实例化模板产生李嘉图合约三元组^[29], 然后生成智能合约代码。2017 年, 周学峰等^[30]提出对代码规则进行法律控制, 从而适用数字虚拟社会, 主要存在两种法律控制方法: 一种方法是直接对代码规则的设计提出要求, 确保在数字虚拟社会中运行的规则与现实世界中适用的法律规则一致; 另一种方法是不对代码规则本身进行直接规制, 而是通过对程序及其相关机构或主体进行规制, 从而实现对代码规则间接规制。

3.3 安全性分析

当前的智能合约研究处于初级阶段, 应用比较简单, 甚至是不够智能的, 智能合约还面临可信和安全问题。已经开发好的智能合约也有可能

会出现意想不到的错误, 因而有必要对智能合约

进行验证和分析。对智能合约链上安全性进行分析可以从携带证明的智能合约、开发软件验证工具、形式化证明和零知识证明 4 方面进行研究。

在代码中添加证明。1997 年, Necula 等^[31]提出通过由不受信任的代码生产者产生的携带证明的代码 (PCC, proof-carrying code), 主机验证代码与已定义的安全策略的一致性, 不需要使用加密技术和咨询外部代理。基于 Necula 的工作, 2018 年, Thomas 等^[32]提出携带证明的智能合约 (PCSC, proof-carrying smart contract)。PCSC 将智能合约的形式化正确性证明存储在链上, 从而验证者可以检查智能合约的正确性。但 PCSC 会占用链上存储空间, 增加计算开销。

开发软件验证工具。2016 年, Luu 等^[33]开发了 Oyente 软件工具。Oyente 可以检测智能合约 EVM 字节码中的安全漏洞, 但功能上具有局限性。同年, Bhargavan 等^[34]提出了一个分析和形式化验证智能合约的框架, 介绍了基于语言的验证方法来验证智能合约, 提出了两种基于 F* 语言的原型工具 Solidity*、EVM*。Solidity* 将 Solidity 语言编译为 F* 语言, 以此来验证功能正确性和检测运行期间的 Bug; EVM* 将 EVM 字节码解码成更简洁的 F* 语言, 以此来分析低级属性, 如完成调用或交易所需要的 Gas 范围。但是这两个工具不支持 Solidity 许多语法特征。

形式化建模与验证。2018 年, Bai 等^[35]使用 Promela 建模语言和形式化验证工具 (SPIN) 验证智能购物合约 (SSC, smart shopping contract) 的资金安全性和交易状态可达性。由于 SSC 是有限状态模型, 通过对 SSC 建模和验证, SPIN 随机生成合约的状态, 进而验证 SSC 执行结果与商务购物合约预期结果的一致性。

零知识证明。2018 年, Sánchez 等^[18]使用零知识证明使代码使用者验证代码证明的正确性, 无须泄露证据本身和智能合约代码的实际信息。Zcash^[36]是首个使用零知识证明的区块链“加密货币”项目, 目前最大的智能合约平台 Ethereum 也在 2017 年的“拜占庭”分叉过程中引入了使用同态加密的零知识证明技术 (zk-SNARK, zero knowledge succinct non-interactive arguments of

knowledge) 来保证智能合约交易双方和交易金额的匿名性。

4 智能合约链外安全

当前区块链的存储空间有限, 不能满足所有链外数据存储在链上的要求。一般情况下, 将部分关键数据和计算结果存储在链上, 非关键数据和数据计算放在链外^[37], 从而降低链上数据存储量。智能合约和链外数据交互过程中的安全与智能合约的快速发展关系密切, 因为智能合约需要访问跨链数据和链外数据。智能合约与链外数据交互过程中的安全主要涉及链外数据的可信性^[38-40]、不可否认性等安全属性。

由于智能合约访问链外数据异常复杂, 目前主流的智能合约都不支持直接访问链外数据, 主要有两种解决思路: Oracle^[41]和 RealityKeys^[42]。其中, 应用较为广泛的是 Oracle。Oracle 是第三方服务, 其主要任务是使智能合约访问来自其他区块链或者万维网的数据, 供智能合约使用。Oracle 代理机制^[43]如图 5 所示。

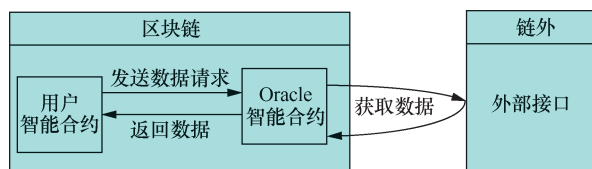


图 5 Oracle 代理机制
Figure 5 Proxy mechanism of Oracle

2018 年, Xu 等^[44]根据使用类型将 Oracle 分为 5 类: 软件 Oracle、硬件 Oracle、入境 Oracle、出境 Oracle 和基于共识的 Oracle。软件 Oracle 提取所需要的网络信息并将其提供给智能合约, 同时提供网络信息的真实性证明; 硬件 Oracle 可以提供通过传感器数据的加密证据和防篡改机制, 使设备在遭受破坏时无法操作, 从而直接获取来自物理世界的信息。硬件 Oracle 最大的挑战是在不牺牲数据安全性的条件下传递数据; 入境 Oracle 将外部世界的数据提供给智能合约; 出境 Oracle 将智能合约数据发送到外部世界; 只使用一个数据来源是不安全和不可靠的, 因而为了避免出现被操纵现象, 需要提供更进一步的安全性, 基于共识的 Oracle 通过使用

不同的 Oracle 组合提供共识机制, 保障数据的可信性和安全性。

2017 年, Eberhardt 等^[45]提出了 5 种可行的链外模式 (可单独使用也可组合使用), 目的在于将计算和数据放在链外, 共有 5 种模式, 分别是挑战响应模式 (challenge response pattern)、链外签名模式 (off-chain signatures pattern)、可寻址内容存储模式 (content-addressable storage pattern)、委托计算模式 (delegated computation pattern)、低合约足迹模式 (low contract footprint pattern)。挑战响应模式在客户端检查区块链智能合约状态是否是 Final 态, 客户端可以在达到 Final 态时通知智能合约, 这样只需将计算结果存放在链上, 而计算在链外进行。挑战响应模式增加了链上交易的总量, 同时需要各参与方配合。链外签名模式由两个参与者在链外对请求改变状态的消息进行签名, 发送到智能合约后, 智能合约验证签名, 之后智能合约更新状态。链外签名模式无须引入系统信任, 可以减轻系统负载, 并且一旦发现恶意参与者, 可以拒绝签名并冻结资金。可寻址内容存储模式将链外数据脱机存储在内容可寻址存储系统中, 并将其引用存储在智能合约中。使用智能合约的客户端可以检索引用, 并在此基础上检索数据。然后, 通过计算数据本身的地址, 将其与存储在智能合约中的引用进行比较, 从而验证数据的正确性。可寻址内容存储模式可以降低链上存储量, 但是引入了第三方, 因而其安全性依赖于第三方。委托计算模式将计算外包给不可信的链外第三方, 生成执行计算的证据, 在链上验证执行计算的证明。委托计算模式提高了区块链的吞吐量, 但是需要简单易用的链上验证工具。低合约足迹模式减少链上交易的数量和规模。本地节点先执行条件检查, 只有在满足条件的情况下才触发链上检查, 从而最小化写入和存储信息。

本文主要从基于硬件 Oracle、基于软件 Oracle 和基于共识 Oracle 这 3 个方面来分析智能合约链外安全。

4.1 基于硬件 Oracle

在硬件 Oracle 方面, 获取链外数据通常采用将硬件作为可信任的第三方中介来实现链外数据传输的方案。但此方案引入了第三方硬件, 对硬

件安全性和可靠性要求比较高。Zhang 等^[46]和 Alder 等^[47]分别通过离链基础设施 (Core) 以及适配器 (Adapter) 和可信硬件 (Relay) 作为中介来进行链外数据传输。

2016 年, Zhang 等^[46]提出了一个面向以太坊的智能合约数据认证系统——TC (town crier)。TC 作为智能合约和 Web 站点之间的桥梁, 通过结合区块链前端与 Inter 可信硬件 SGX 后端, 将经过源身份验证的链外数据提供给智能合约。TC 包含 4 个部分: TC 智能合约、用户 User 智能合约、Enclave 和中继器 Relay。TC 智能合约部署在区块链上, Enclave 和中继器 Relay 驻留在 TC 服务器上。TC 智能合约为 User 智能合约提供一个 API。Enclave 只能通过 Relay 进行网络连接访问数据。Relay 可以提供给 Enclave 3 个链外数据源: 区块链 (主要是以太坊系统)、客户端和 Web 数据。TC 的工作流程分为 5 步: 第 1 步, 用户提交 User 智能合约, 通过 User 智能合约向 TC 智能合约发送获取链外数据请求; 第 2 步, TC 智能合约将请求发送到 Relay, 通过 Relay 将请求进一步转发到 SGXEnclave; 第 3 步, Enclave 通过 Relay 查询链外数据源并获取链外数据; 第 4 步, Enclave 将获得的数据转化为基于数据报的区块链数字签名消息, 并通过 Relay 发送给 TC 智能合约; 第 5 步, TC 智能合约把数据发送给 User 智能合约。TC 系统的数据容易进行解析, 但是, 它要求 Relay 是一个受信任的第三方, 因为它不受 SGX 的完整性保护。

2018 年, Adler^[47]给出面向 Ethereum 的 Chainlink 解决方案, 使智能合约可以利用新创建的 ChainlinkOracle 与 Off-chain 系统进行安全通信。Chainlink 应用分散的离链 Oracle 将链外数据导入链上智能合约。Chainlink 解决方案包含 4 个部分: 链上用户 User 智能合约、ChainlinkOracle 智能合约、离链基础设施和适配器。Chainlink 解决方案的工作流程分为 6 步: 第 1 步, User 智能合约向 ChainlinkOracle 智能合约发出链上请求, 请求获取链外数据, 请求信息包含 Oracle 版本、所需资源数和其他重要属性; 第 2 步, ChainlinkOracle 智能合约记录这个事件并将此请求发送给 Chainlink 节点上的 Core; 第 3 步,

Chainlink 节点上的 Core 接收到请求后为 Adapter 配置路由; 第 4 步, 配置完路由后, Adapter 执行获取链外数据的请求, 获得链外数据并将数据返回到 Core; 第 5 步, Core 将数据发送到 ChainlinkOracle 智能合约; 第 6 步, ChainlinkOracle 智能合约会把所有响应的数据进行处理并合并为一个响应数据包发送到 User 智能合约。此方案安全性依赖于 Chainlink 节点上的 Core 和 Adapter, 虽然通过形成内部网络来分散应用程序, 以检测和处理数据的不一致性, 但是引入了 Core 和 Adapter, 安全性依赖于第三方。

4.2 基于软件 Oracle

在软件 Oracle 方面, 获取链外数据无须可信第三方, 但是通常需要对通信协议进行更改, 提供数据交互的证据来进行真实性验证, 因而在一定程度上增加了数据计算。同时, 对协议进行更改将导致数据难以进行解析。文献[48-51]通过修改 TLS 协议和对 TLS 协议进行扩展, 以证明数据的不可否认性和真实性。Guarnizo 等^[52]通过采用 TDS 数据结构构建日志系统, 以提供链外数据的真实性证明。

2014 年, Voyiatzis 等^[48]对 TLS 协议进行更改, 使用 TLSNotary 提供证明客户端与服务器之间发生网络流量交换的方法。该方法包括 3 个部分: Client (Auditee)、Server 和 Auditor。TLSNotary 有两个前提, 分别是 Auditor 拥有 ServerMACKey, Auditee 拥有 ClientEncryptionKey、Server EncryptionKey 和 ServerMACKey。Client 和 Server 进行 TLS 握手时, Client 将 Server 返回的 Response 作为证据发送到 Auditor, Auditor 保留一部分 Secret 数据以防 Client 伪造 Server 返回的 Response。在 Client 对 Server 返回的加密内容 Response 做出承诺后, Auditor 向 Client 提供其保留的 Secret 数据, Client 得以构造 ServerMACKey。至此, Client 可以进行 TLS 协议的解密和认证。但是, TLSNotary 证明有局限性: TLSNotary 仅支持 TLS1.0 和 TLS1.1; TLSNotary 指定使用弱哈希函数, 如使用哈希函数 MD5 和 SHA-1; TLSNotary 只支持 RSA 密钥交换, 不提供转发保密功能。

在 Casey 的工作基础上, 2017 年, Ritzdorf 等^[49]提出基于 TLS 扩展的 TLS-N 的方法。该方

法首先由服务器生成关于 TLS 会话内容的证据,然后在客户端生成关于 TLS 会话内容的非交互式证明,接着将会话内容和证明发送到区块链上的第三方或者智能合约进行验证。该方法通过一个分散的区块链 Oracle,将 Web 上的内容安全传输到区块链上,无须区块链外的可信第三方,同时支持 TLS1.3,是目前较为有效的获取链外数据的方法。但是,此方法在证据生成过程中应用了 MerkleTree^[50]和 SaltTree^[51],所以数据不容易被智能合约解析,同时 TLS 协议需要进行较大更改。

2018 年,Guarnizo 等^[52]设计了一个实用的智能合约数据馈送(data feeds)服务系统——PDFS,能够填补 Oracle 解决方案和传输层身份验证的空白,为智能合约提供通过验证的数据。该系统允许内容提供商将其 Web 实体与块链实体无缝链接起来而不破坏 TLS 信任链或 TLS 堆栈,因此不需新的受信任方进行数据验证。同时,内容提供者可以定义需要的数据格式,便于智能合约解析和使用。PDFS 包含 4 个部分:链外的 Content Provider、链上由 Content Provider 创建的 Content 智能合约、Contract Parties 和 Contract Parties 部署的 Relying 智能合约。Content Provider 产生防篡改数据结构(TDS, tamper-evident data structure),用来存储 Content Provider 可以向外界提供的数据 Entry。TDS 更新时,Content Provider 重新计算数据结构并将新的 DataEntry 添加到 TDS 中,然后将 TDS 根发送到 Content 智能合约,因而 Content 智能合约不需要存储任何数据,仅仅保存 TDS 根。Content 智能合约把 TDS 根和 Membership 证明存储在链上。Content 智能合约检查 TDS 的更新,同时验证更新仅仅是添加操作而不存在修改或删除操作。PDFS 工作流程:当 Contract Parties 调用获取链外数据的方法,这个方法需要使用 Content Provider 的数据。Contract Parties 先获得由 Content Provider 产生的数据 Entry 和 Membership 证明,并将数据 Entry 和 Membership 证明作为调用方法的参数;然后,根据参数,Contract Parties 验证 Content Provider 是否产生过这两个参数,同时 Relying 智能合约调用 Content 智能合约的 Membership 验证方法以验

证身份。当数据 Entry 验证通过后,Relying 智能合约通过 Content 智能合约提供的 TDS 根获取 Content Provider 提供的链外数据。此方法无须可信第三方,数据更容易被智能合约解析。

4.3 基于共识 Oracle

基于共识的 Oracle,Xu 等^[53]提出多重模型 Oracle。一个可靠的多重模型 Oracle,遵循博弈原理,有经济激励和惩罚措施,越多节点参与,其真实性越高。当数据发送到智能合约时,需要保证参与者节点无法知晓其他参与者的数据,智能合约从众多数据中选择最接近中位数的数据,如果是二元数据,则统计得票数据最多的数据。最后对提供正确数据的节点进行奖励。除此之外,利用预测市场也可以获取链外数据^[54],但是此方法责任方不明确,并且数据输入依赖人工,数据质量不高。

5 结束语

区块链技术的发展为智能合约提供了可信的执行环境,允许在不依赖第三方的情况下进行可信、可追踪且不可逆的合约交易。智能合约的普及和大规模的应用,必须满足安全性、一致性、完整性、原子性等属性,其中,安全性受到重点关注。本文总结和分析了主流的 Ethereum、Hyperledger Fabric 和 EOS 平台上的智能合约运行机制以及智能合约链上安全性和链外安全性的最新相关研究成果。针对智能合约链上安全性,主要总结和讨论了架构设计安全和代码安全。在架构设计安全方面,从平台安全和模式安全两个方面出发,介绍了开发过程中的智能合约开发框架安全和设计模式安全。在代码安全方面,从漏洞挖掘、语言安全和法律合约与代码一致性 3 个方面出发,概述了智能合约开发时的智能合约漏洞挖掘、开发语言安全以及法律合约与代码一致性。

针对智能合约链外安全性,主要涉及智能合约与链外数据交互时的安全。Oracle 充当代理机制与链外数据进行交互,重点关注了基于软件 Oracle 的 TLSNotary 方法、TLS 扩展的 TLS-N 方法和 PDFS 系统等,总结了基于硬件 Oracle 的 TC 系统和 Chainlink 解决方案。另外,分析了基于共

识的多重模型 Oracle 以及 Oracle 分类。

综上所述, 针对智能合约链上和链外安全性的方案, 不同研究团体对于其安全性形成了各自不同的特色, 并且取得了许多重要成果。从智能合约发展趋势来看, 今后一段时间内的安全性研究将呈现以下特点。

1) 智能合约动态安全。目前, 智能合约的安全性主要关注其静态安全, 较少关注动态安全(如智能合约执行过程中的安全性)。智能合约安全应该涵盖静态安全和动态安全。因而, 智能合约动态安全是重要研究方向之一。

2) 法律合约与智能合约代码的一致性。将法律合约与智能合约结合, 必将改变传统的庭审模式。但是, 有时会出现某项法律规则的法律含义与其实际适用含义不一致的现象, 这给法律规则的代码化带来了困难。目前法律合约与智能合约的研究主要涉及: 如何由法律合约生成智能合约代码并保证其一致性; 如何验证法律合约和智能合约代码的一致性。这两个问题将成为智能合约未来研究热点。

3) 区块链链上存储空间有限。智能合约与链外数据交互验证数据安全性时往往会导致链上负载增加, 进一步压缩了链上存储空间, 从而造成智能合约平台吞吐量降低, 交易延迟提高, 不利于智能合约的发展。因而, 如何在验证链外数据安全性的同时不增加链上负载可作为今后研究工作的一个重点。

4) 智能合约开发技术、安全监管标准不统一。智能合约平台和监管合约的标准主要由分散的区块链应用联盟搭建, 如 2015 年摩根大通、巴克莱等全球 9 大银行联手制定了区块链支付标准和协议; 2016 年 5 月区块链技术提供商 Chain 和第一资本、花旗银行等金融机构发布了区块链方面的开放标准^[55], 在智能合约框架方面实现了突破。虽然分散的标准都在建立, 但是在全球层面缺乏一个统一的技术开发标准, 智能合约使用的兼容性不高, 对于智能合约安全性的监管受到极大挑战。

5) 目前智能合约应用逻辑是根据已有场景的“if-then”类型的条件响应预定义规则, 可以满足当前交易自动化和数据处理的需求。但是未

来智能合约应当做到根据未知场景做“what-if-then”推理演算、计算实验和一定程度的自主决策, 实现由“自动化合约”向“智能合约”的转变。而在此过程中, 智能合约推理自身代码、开发安全性, 以及针对推理做出相应的智能决策应当受到重点关注。

随着研究方法的改善和研究技术的成熟, 以上 5 点研究将会得到更好发展, 必将对智能合约的发展产生深远影响。

参考文献:

- [1] WIKIPEDIA. Smart contract[EB].
- [2] SZABO N. Formalizing and securing relationships on public networks[J]. First Monday, 1997, 2(9).
- [3] OUADDAH A, ABOU E A, AIT O A. Fair-access: a new blockchain-based access control framework for the Internet of things[J]. Security and Communication Networks, 2016, 9(18): 5943-5964.
- [4] ZHENG Z, XIE S, DAI H, et al. An overview of blockchain technology: architecture, consensus, and future trends[C]//2017 IEEE International Congress on Big Data. 2017: 557-564.
- [5] LI X, JIANG P, CHEN T, et al. A survey on the security of blockchain systems[J]. Future Generation Computer Systems, 2017, 10(8): 274-287.
- [6] YLI-HUOMO J, KO D, CHOI S, et al. Where is current research on blockchain technology —a systematic review[J]. PloS one, 2016, 11(10).
- [7] ZHENG Z, XIE S, DAI H N, et al. Blockchain challenges and opportunities: a survey[J]. International Journal of Web and Grid Services, 2018, 14(4): 352-375.
- [8] DINH T A, WANG J, CHEN G, et al. Blockbench: a framework for analyzing private blockchains[C]//The 2017 ACM International Conference on Management of Data. New York: ACM, 2017: 1085-1100.
- [9] BARTOLETTI M, POMPIANU L. An empirical analysis of smart contracts: platforms, applications, and design patterns[C]// International Conference on Financial Cryptography and Data Security. Berlin: Springer, 2017: 494-509.
- [10] ZHENG Z, XIE S, DAI H N, et al. Blockchain challenges and opportunities: a survey[J]. International Journal of Web and Grid Services, 2018, 14(4): 352-375.
- [11] WOOD G. Ethereum: a secure decentralised generalised transaction ledger[J]. Ethereum Project Yellow Paper, 2014, 151: 1-32.
- [12] BOGNER A, CHANSON M, MEEUW A. A decentralised sharing app running a smart contract on the ethereum block-chain[C]//The 6th International Conference on the Internet of Things. New York: ACM, 2016: 177-178.
- [13] ANDROULAKI E, BARGER A, BORTNIKOV V, et al. Hyperledger Fabric: a distributed operating system for permissioned block-chains[C]//The Thirteenth EuroSys Conference. New York:

- ACM, 2018: 30.
- [14] CACHIN C. Architecture of the hyperledger blockchain fabric[C]//Workshop on Distributed Cryptocurrencies and ConsensusLed-gers. 2016, 310.
- [15] WANG S, YUAN Y, WANG X, et al. An overview of smart contract: architecture, applications, and future trends[C]//2018 IEEE Intelligent Vehicles Symposium (IV). 2018: 108-113.
- [16] LI J, TANG J, ZHANG J, et al. Eos: expertise oriented search using social networks[C]//The 16th international conference on World Wide Web. 2007: 1271-1272.
- [17] KALRA S, GOEL S, DHAWAN M, et al. Zeus: analyzing safety of smart contracts[C]//The 25th Annual Network and Distributed System Security Symposium. 2018: 18-21.
- [18] SÁNCHEZ D C. Raziol: private and verifiable smart contracts on blockchains[J]. arXiv preprint, arXiv: 1807.09484, 2018.
- [19] DARGAYE Z, KIRCHNER F, TUCCI-PIERGIOVANNI S, et al. Towards secure and trusted-by-design smart contracts[C]//The 29th Francophone Days of Application Languages. 2018: 7-18.
- [20] KOSBA A, MILLER A, SHI E, et al. Hawk: the blockchain model of cryptography and privacy-preserving smart contracts[C]//2016 IEEE symposium on security and privacy (SP). 2016: 839-858.
- [21] WÖHRER M, ZDUN U. Smart contracts: security patterns in the Ethereum ecosystem and solidity[C]//2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE). 2018: 2-8.
- [22] 付梦琳, 吴礼发, 洪征, 等. 智能合约安全漏洞挖掘技术研究[J]. 计算机应用, 2019, 39(7): 1959-1966.
- FU M L, WU L F, HONG Z, et al. Research on smart contract security vulnerability mining technology[J]. Journal of Computer Applications, 2019, 39 (7): 1959-1966.
- [23] NATOLI C, GRAMOLI V. The blockchain anomaly[J]. arXiv preprint, arXiv:1605.05438, 2016.
- [24] ATZEI N, BARTOLETTI M, CIMOLI T. A survey of attacks on Ethereum smart contracts (SOK)[M]//Principles of Security and Trust. 2017: 164-186.
- [25] TONELLI R, DESTEFANIS G, MARCHESI M, et al. Smart contracts software metrics: a first study[J]. arXiv preprint, arXiv:1802.01517, 2018.
- [26] GRISHCHENKO I, MAFFEI M, SCHNEIDEWIND C. A semantic framework for the security analysis of Ethereum smart contracts[C]//International Conference on Principles of Security and Trust. Berlin: Springer, 2018: 243-269.
- [27] FRANTZ C K, NOWOSTAWSKI M. From institutions to code: towards automated generation of smart contracts[C]//2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W). 2016: 210-215.
- [28] CLACK C D, BAKSHI V A, BRAINE L. Smart contract templates: foundations, design landscape and research directions[J]. arXiv pre-print, arXiv:1608.00771, 2016.
- [29] 沈鑫, 裴庆祺, 刘雪峰. 区块链技术综述[J]. 网络与信息安全学报, 2016, 2(11): 11-20.
- SHEN X, PEI Q Q, LIU X F. Survey of block chain[J]. Chinese Journal of Network and Information Security, 2016, 2(11): 11-20.
- [30] 周学峰, 赵梓皓. 解析计算法律学[J]. 北京: 中国计算机学会通讯, 2017: 43-51.
- ZHOU X F, ZHAO Z H. Analysis of computational law[J]. Beijing: Communications of the CCF, 2017: 43-51.
- [31] NECULA G. Proof-carrying code[J]. Encyclopedia of Cryptography & Security, 1996, 141(1):106-119.
- [32] THOMAS D, PAUL G, MAURICE H, et al. Proof-carrying smart contracts[J]. Stevens Institute of Technology, 2018, 325-338.
- [33] LUU L, CHU D H, OLICKEL H, et al. Making smart contracts smarter[C]//The 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 254-269.
- [34] BHARGAVAN K, DELIGNAT-LAUDAUD A, FOURNET C, et al. Formal verification of smart contracts: short paper[C]//The 2016 ACM Workshop on Programming Languages and Analysis for Security. New York: ACM, 2016: 91-96.
- [35] BAI X, CHENG Z, DUAN Z, et al. Formal modeling and verification of smart contracts[C]//The 7th International Conference on Software and Computer Applications. Washington: ACM, 2018: 322-326.
- [36] 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, 4(4): 22-29.
- ZHANG F, SHI B X, JIANG W B. Review of key technology and its application of blockchain[J]. Chinese Journal of Network and Information Security, 2018, 4(4): 22-29.
- [37] 薛锐, 吴迎, 刘牧华, 等. 可验证计算研究进展[J]. 中国科学: 信息科学, 2015, 45(11): 1370-1388.
- XUE R, WU Y, LIU M H, et al. Progress in verifiable computing[J]. China Science: Information Science, 2015, 45(11): 1370-1388.
- [38] HARZ D. Trust and verifiable computation for smart contracts in permissionless blockchains[D]. KTH, School of Information and Communication Technology. 2017.
- [39] TEUTSCH J, REITWIEBNER C. A scalable verification solution for blockchains[EB].
- [40] ZYSKIND G. Efficient secure computation enabled by blockchain technology[D]. Massachusetts: Massachusetts Institute of Technology, 2016.
- [41] AS S. Enabling data markets using smart contracts and multi-party computation[C]//Business Information Systems Workshops: BIS 2018 International Workshops. Berlin: Springer, 2019: 258.
- [42] NEIDHARDT N, KOHLER C, NUTTGENS M. Cloud service billing and service level agreement monitoring based on blockchain[C]//EMISA. 2018: 65-69.
- [43] MOLINA-JIMENEZ C, SOLAIMAN E, SFYRAKIS I, et al. On and off-blockchain enforcement of smart contracts[C]//European Conference on Parallel Processing. 2018: 342-354.
- [44] XU X, PAUTASSO C, ZHU L, et al. The blockchain as a software connector[C]//2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA). 2016: 182-191.
- [45] EBERHARDT J, TAI S. On or off the blockchain? Insights on off-chaining computation and data[C]//European Conference on Service-Oriented and Cloud Computing. 2017: 3-15.
- [46] ZHANG F, CECCHETTI E, CROMAN K, et al. Town crier: an authenticated data feed for smart contracts[C]//The 2016 ACM

SIGSAC Conference on Computer and Communications Security. 2016: 270-282.

- [47] ADLER J, BERRYHILL R, VENERIS A, et al. Astraea: a decentralized blockchain oracle[C]//2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). 2018: 1145-1152.
- [48] VOYIATZIS A G, WEIPPL E. Whom you gonna trust? a longitudinal study on TLS notary services[C]//Data and Applications Security and Privacy XXX: 30th Annual IFIP WG 11.3 Conference. 2016: 18-20.
- [49] RITZDORF H, WÜST K, GERVAIS A, et al. TLS-N: non-repudiation over TLS enabling ubiquitous content signing for disintermediation[J]. IACR ePrint Report, 2017, 578.
- [50] JAKOBSSON M, LEIGHTON T, MICALI S, et al. Fractal Merkle tree representation and traversal[C]//Cryptographers' Track at the RSA Conference. 2003: 314-326.
- [51] XUE J, XU C, ZHANG Y, et al. DStore: a distributed cloud storage system based on smart contracts and blockchain[C]//International Conference on Algorithms and Architectures for Parallel Processing. 2018: 385-401.
- [52] GUARNIZO J, SZALACHOWSKI P. PDFS: practical data feed service for smart contracts[J]. arXiv preprint, arXiv:1808.06641, 2018.
- [53] XU X, PAUTASSO C, ZHU L, et al. The blockchain as a software connector[C]//2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA). 2016: 182-191.
- [54] BANERJEE A. Blockchain technology: supply chain insights from ERP[M]//Advances in Computers. 2018: 69-98.
- [55] FABIANO N. The internet of things ecosystem: the blockchain and privacy issues the challenge for a global privacy standard[C]//2017 International Conference on Internet of Things for the Global Community (IoTGC). 2017: 1-7.

[作者简介]



孟博（1974—），男，河北石家庄人，博士，中南民族大学教授，主要研究方向为区块链、安全协议和形式化方法。



刘加兵（1994—），男，湖北黄石人，中南民族大学硕士生，主要研究方向为区块链智能合约、协议分析。



刘琴（1990—），女，湖北黄冈人，中南民族大学硕士生，主要研究方向为智能合约代码与法律合约的一致性。



王潇潇（1996—），女，安徽宿州人，中南民族大学硕士生，主要研究方向为区块链共识机制。



郑旭睿（1996—），男，湖北武汉人，中南民族大学硕士生，主要研究方向为区块链自我主权认证。



王德军（1974—），男，湖北荆门人，博士，中南民族大学副教授，主要研究方向为安全协议和形式化方法。