区块链系统中智能合约技术综述

范吉立 李晓华 聂铁铮 于 戈
(东北大学计算机科学与工程学院 沈阳 110169)

摘 要 区块链是一个全球性的去中心化分布式数据库账本。智能合约作为一段由事件驱动的、具有状态的、运行于区块链系统之上的程序,能够保管、处理区块链账本上的数字资产;运行在通用平台上的智能合约还能够实现传统应用系统的部分功能。区块链技术的发展为智能合约提供了很好的运行基础,智能合约在区块链上能够发挥重要作用。随着比特币和以太坊等区块链平台的迅速发展,智能合约具备了良好的发展契机。然而,智能合约应用还处于早期发展阶段,相关研究相对较少,实际应用中智能合约的适用场景不够丰富。文中从智能合约编程语言、实现技术、发展现状和存在的挑战、未来前景等几个方面进行研究和探讨,重点描述了不同智能合约开发语言的特性,以及相互之间的对比;以智能合约运行环境为标准进行分类,研究了各种区块链系统中智能合约的开发、部署和运行机制,并探讨了不同类型智能合约平台的应用范围,对不同区块链系统在智能合约开发、社区支持以及相应的生态系统等方面进行了全面的比较;然后,介绍智能合约的研究现状和面临的挑战,从安全性、可扩展性、可维护性等方面进行深入分析;最后,分析了区块链和智能合约技术的发展趋势,探讨了未来的应用场景。

关键词 区块链,去中心化,智能合约,以太坊虚拟机
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/jsjkx.190300013

Survey on Smart Contract Based on Blockchain System

FAN Ji-li LI Xiao-hua NIE Tie-zheng YU Ge
(School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China)

Abstract Blockchain is a decentralized global distributed database leger. Smart contract is a piece of event-driven program with states that runs over blockchain systems, which can take custody over digital assets. Smart contracts running on a common platform can also implement parts of the functions of traditional applications. Development of the blockchain provides an appropriate platform for smart contract, and smart contract plays an important role on blockchain systems. With the rapid development of blockchain platforms such as Bitcoin and Ethereum, smart contracts have a good development opportunity. However, applications of smart contract are still in the early stage of development, and there are relatively few related studies. The application scenarios of smart contracts are not enough in practical application. This paper studied programming languages and implementation technologies of smart contract, discussed and explored the development status as well as challenges and future prospects. It described the characteristics of different development languages and took a comparison among them. Then, it classified blockchain systems based on the running environment of smart contract, and studied the development, deployment and running mechanism of smart contracts in various blockchain systems. Also, this paper explored the application scope of various smart contract platforms, and took a comprehensive comparison of different blockchain systems on smart contract development, community support and corresponding ecosystems. It introduced the status and challenges of smart contract research, and conducted analysis on security, scalability, and maintainability. Finally, it analyzed the development trend of blockchain and smart contract technology, and discussed the application scenarios in the future.

Keywords Blockchain, Smart contract, Decentralization, Ethereum virtual machine

到稿日期:2019-03-07 **返修日期**:2019-06-01 **本文受国家自然科学基金项目**(61672142,61433008,U1435216),**中央高校基本科研业务费项目**(N150408001-3,N150404013),**辽宁省科学技术基金**(20180550321)**资助**。

范吉立(1987-),**男**,**硕士**,**工程师**,**主要研究方向为区块链技术、云计算**;**李晓华**(1969-),**女**,**博士**,**讲师**,CCF **会员**,**主要研究方向为数据库系统、移动计算**;**聂铁铮**(1980-),**男**,**博士**,**副教授**,CCF **高级会员**,**主要研究方向为分布式数据库系统、数据集成**;**于 戈**(1962-),**男**,**博士**,**教授**,CCF **会士**,**主要研究方向为分布与并行系统、大数据管理**,E-mail:yuge@mail.neu.edu.cn(**通信作者**)。

1 引言

近年来，随着比特币[1]的快速发展与普及，区块链技术的研究与应用也逐渐被很多学者和科研人员重视，并呈现出爆发式增长态势。区块链的核心，是一种大规模的分布式数据库，这个技术平台是开放的、可编程的，它不仅能够记录金融交易，还可以与智能合约相结合，记录并运行、维护几乎所有有价值的东西[2]。

目前知名度较高的区块链系统有：比特币（Bitcoin）、以太坊[3]（Ethereum）、超级账本[4]（Hyperledger）、Zcash、EOS、Quorum、Parity、Litecoin、Corda 等。其中，比特币、Litecoin、Zcash 属于加密货币系统；以太坊、超级账本、EOS、Parity 等系统应用扩展到了数字资产与通用智能合约的范畴。

区块链的系统进化过程[5]可以归纳为 3 个阶段：

1）区块链 1.0——数字货币；

2）区块链 2.0——数字资产与智能合约；

3）区块链 3.0——从 DAO（区块链自治组织）、DAC（区块链自治公司）到区块链社会（科学、医疗、教育等）。

智能合约是由尼克萨博提出的理念[6]，发表在负熵学会（Extropy Institute）网站上[7]，几乎与互联网同龄。智能合约的定义为：一段由事件驱动的、具有状态的、运行在一个复制的且分享的账本之上的、能够保管账本上资产的程序[8]。由于技术发展及执行环境的限制，智能合约并没有得到良好的应用。自比特币系统获得巨大成功以来，越来越多的人意识到区块链技术以及智能合约的重要性，区块链技术的不断发展也让其成为智能合约运行的最佳底层支撑。相对于传统金融合约，智能合约具有较低的法律开销和交易开销，并且降低了用户使用的门槛[9]。智能合约已在许多区块链系统上成功实现，比较著名的系统有以太坊和超级账本。以太坊是一个开源的支持智能合约功能的公共区块链平台[10]，是 Vitalik Buterin 在 2013 年受比特币发展的启发而提出的。其由于支持用户开发智能合约的功能，被称为第二代区块链系统。利用智能合约，在以太坊上可以创建任何去中心化的应用[11]。超级账本（Hyperledger Fabric）是一个为部署商业应用许可区块链而设计的系统[12]，具有良好的灵活性和通用性，支持种类繁多的非确定性智能合约（链码）和可插拔的服务，可插拔组件使得 Fabric 具有灵活的可扩展性。

2 区块链中的智能合约语言

2.1 智能合约语言

广义上的智能合约，即能够让用户自己定义所需交易逻辑的代码程序，几乎存在于所有区块链系统，包括最广为人知的比特币，以及以太坊、超级账本、Parity、Zcash 等。从编程语言表现或者运行环境考虑，智能合约可以分为脚本型、图灵完备型、可验证合约型[13] 3 种。1）比特币系统可以允许通过编写基于堆栈的操作码（Opcode）来实现简单的交易逻辑，比如改变比特币花费的前提条件，这个系统称为比特币脚本系统[14]。2）以太坊提供一种基于图灵完备语言的智能合约平台，也是最早的图灵完备智能合约。以太坊系统提供以太坊虚拟机（Ethereum Virtual Machine，EVM）[15]，合约代码在 EVM 内部运行。以太坊用户使用特定语言编写智能合约代码，并编译成 EVM 字节码运行。超级账本提供另一种图灵完备智能合约，它在 Docker 容器环境中运行语言无关的智能合约，即智能合约代码可以使用任何编程语言进行编写，之后被编译器编译并打包进 Docker 镜像，以容器作为运行环境。3）正在开发中的 Kadena 项目提供一个可验证的智能合约系统，以保证商业应用逻辑的可靠性和高效性[16]，以及商业应用操作的快速、安全运行。但它使用的编程语言 Pact 是非图灵完备的。

所谓图灵完备，是指能用该编程语言模拟任何图灵机[17]。图灵完备的规则能够实现任何操作逻辑，例如，一种编程语言中包含条件控制语句 if，goto 等，并且能够维护任意数量的变量，则可以编写出符合任何逻辑的代码，因此这种编程语言具备图灵完备性。

下面分别介绍这 3 类语言，并加以对比和总结。

比特币系统有其嵌入式的脚本语言，以太坊等基于 EVM 的智能合约平台发布了智能合约专用开发语言，其他系统或平台大多采用通用编程语言。本文重点以比特币和以太坊为例，来介绍智能合约语言及其运行环境。

2.2 比特币脚本语言

比特币脚本语言是一种基于堆栈的逆波兰式[18]简单执行语言，它用于编写比特币交易中未花费交易输出（UTXO）的锁定脚本（Locking Script）和解锁脚本（Unlocking Script）。锁定脚本确定了花费输出所需要的条件，而解锁脚本用来满足 UTXO 上锁定脚本所确定的条件，解锁并支付。当一条交易被执行时，每个 UTXO 的解锁脚本和锁定脚本同时执行，根据执行结果（true/false）来判定该笔交易是否满足支付条件。

脚本语言被设计得非常简单，类似于嵌入式装置，仅可在有限的范围内执行，可做较简单的处理。脚本指令被称为操作码，分为常量、流程控制、栈操作、算术运算、位运算、密码学运算、保留字等。后文中提及的 OP_DUP 等属于栈操作指令。脚本是非图灵完备的语言，包含的操作码不具备循环和复杂的流控制功能，仅可执行有限的次数，避免了因编写疏忽等原因导致的无限循环或其他类型的逻辑炸弹。比特币脚本这种有限的执行环境和简单的执行逻辑，有利于对可编程货币的安全性进行验证，能够防止形成脚本漏洞而被恶意攻击者所利用。

比特币系统处理的大多数交易花费都是由“付款至公钥哈希（P2PKH）”脚本锁定的输出[19]，即锁定脚本中包含一个公钥的哈希值（比特币地址），解锁时通过包含公钥和对应私钥所创建的数字签名的脚本来验证。例如，用户 A 向用户 B 支付一笔交易，锁定脚本可以表示为：

OP_DUP OP_HASH160 〈B Public Key Hash〉 OP_EQUALVERIFY OP_CHECKSIG

其中，B Public Key Hash 为用户 B 的公钥的哈希。当用户 B 解锁该笔交易时，使用包含 B 的数字签名和公钥的解锁脚本：

〈B Signature〉〈B Public Key〉

比特币系统中的节点把解锁脚本与锁定脚本组合，形成验证脚本：

〈B Signature〉〈B Public Key〉 OP_DUP OP_HASH160 〈B Public Key Hash〉 OP_EQUALVERIFY OP_CHECKSIG

该验证脚本被放入堆栈中执行，输出结果决定着交易的有效性。

2.3　以太坊图灵完备型语言

由于比特币等脚本语言不具备图灵完备性，编写的智能合约交易模式非常有限，只能用于虚拟货币类应用，因此 Vitalik Buterin 推出了支持图灵完备语言的以太坊智能合约平台。以太坊提供了智能合约专用开发语言，其他系统或平台大多采用通用编程语言。目前，以太坊提供了 2 种编程语言：Serpent[20] 和 Solidity[21]。Solidity 在语法上类似于 JavaScript，也是以太坊官方推荐的智能合约编程语言，它具有详细的开发文档；Serpent 语言类似于 Python 语言，具备简洁的特性。以太坊曾经提供了 Mutan[22] 和 LLL 语言，Mutan 是类似于 C 语言的一种高级语言，但该语言已于两年前停止维护；LLL 语言已经废弃，官方代码库也已经无法访问。

2.3.1　Solidity 语言

Solidity 是一种“面向合约”（或面向对象）的高级编程语言，它是专门为编写运行在 EVM 上的智能合约而设计的。其语法接近 JavaScript，并且支持强类型、继承、库以及用户自定义类型。但 Solidity 也有其独特的语言特性：

1）特殊的数据类型——Address。运行在以太坊上的智能合约被当作一个特殊的账户——合约账户，类似于外部账户，合约账户也是由一个 20 字节的地址所定位。因此，Solidity 语言设计了用于定义合约地址的 Address。

2）灵活的变量声明。在作用范围内，状态变量的定义声明与调用没有绝对的顺序关系，定义声明可以在调用语句之后。

3）两种数据存储方式：Memory（内存型）和 Storage（持久型）。Memory 类似其他高级语言的变量存储方式，使用完被回收。默认的函数参数即为 Memory 类型；然而，在区块链上有非常多的状态需要永久记录下来，状态变量默认保存为 Storage 类型。在用户编程过程中，也可以使用关键字灵活地手工指定数据的存储方式。

4）数字货币支付属性。Payable 关键字使其在代码层支持以太币等数字货币的支付以及收款操作，使得合约可以接受交易并持有一定数量的货币。

5）支持回滚的异常机制。对于异常事件，不是让程序去捕获处理，而是触发回滚对应的代码自动处理，从而保证合约中状态数据的一致性以及合约执行的原子性。

6）严格控制可见性。函数和状态变量共有 4 种可见性定义，即 External，Internal，Public，Private，用来限制函数或状态变量在合约内外以及继承关系中的调用和访问权限。

Solidity 还支持一些针对智能合约特性的独特变量，它们作用于全局命名空间，主要用于获取区块链的相关信息，如表 1 所列。

表 1　智能合约编程语言中的特殊变量

Table 1　Special variables in smart contract programming languages

特殊变量	类型	描述
msg. sender	address	当前消息的发送者地址
msg. value	uint	随当前消息发送的以太币数量（wei）
block. coinbase	address	当前区块的矿工收益地址
block. gaslimit	uint	当前区块的 gas 限定额
…	…	…

2.3.2　Serpent 语言

从命名来看，Serpent 的设计非常类似于 Python，它是一种专门编写智能合约的高级语言，具备低级语言高效易用的编程风格以及针对智能合约的特性。最新版本的编译器由 C++ 语言编写，目的是能够更广泛地嵌入客户端程序。Serpent 与 Python 之间虽然相似，但也有诸多不同之处：

1）Serpent 的数值不能大于 2^{256}，否则会发生溢出；

2）Serpent 不支持 Decimal 数值类型；

3）Serpent 不支持 List，Dictionary 以及其他一些高级特性；

4）Serpent 没有第一类函数的概念，虽然合约中可以定义函数，也可以调用自己的函数，但是在调用过程中，变量（Storage 类型除外）不是永久存在的；

5）类似于 Solidity，Serpent 支持持久存储变量的概念，即 Storage 型变量；

6）类似于 Solidity，Serpent 可以使用 extern 语句来调用其他合约，或从其他合约中调用函数；

7）作为运行在区块链上的编程语言，Serpent 同样支持表 1 中的特殊变量。

2.4　可验证型语言 Pact

Pact 语言类似于 Haskell 语言，用于编写直接运行在 Kadena 区块链上的智能合约，主要应用于安全性和效率要求较高的商业交易场合。

Pact 智能合约由 3 部分构成：tables，keysets，module。它们分别负责合约的数据存储、合约授权验证以及合约代码 code。该语言的主要特点有：语言逻辑结构属于图灵非完备，不支持循环和递归；代码人工可读，并且嵌入式地运行于区块链上；支持组件化设计和导入；支持 key-row 和列式数据库模式；支持类型推断；支持秘钥轮换（key rotation）；支持与工业数据库进行集成。

Pact 语法设计类似于 LISP 语言，代码结构利于快速分析和执行语法树。下面给出一段计算平均值的函数代码：

```
(defun average (a b)
    "take the average of a and b"
    (/ (+ a b) 2))
```

定义了 average 函数，用于计算两个数的平均值。这种语法特点能够使计算机更加快速地执行代码。

2.5　超级账本智能合约语言

超级账本智能合约 Chaincode 一般由 Golang 编写，同时也支持其他编程语言，如 Java。Go 是由 Robert Griesemer，Rob Pike，Ken Thompson 从 2007 年末主持开发，并最终于 2009 年 11 月开源的语言，属于图灵完备型。每个 Go 程序都是由包构成的，并且总是从 main 包开始执行。Go 语言具有以下特点：

1)良好的并发机制,程序能够充分利用多核和联网机器。Go 语言引入了 goroutine 来实现并发机制,并使用消息传递来共享内存。

2)设计简洁。代码风格简洁,格式统一,阅读性和可维护性高。该语言只有 25 个关键字,但能够支持大部分其他编程语言支持的特性,如继承、重载、对象等。

3)内嵌 C 语言支持。该语言可以直接包含 C 语言代码,利用现有的丰富 C 程序库。

4)错误处理。Go 语言使用 3 个关键字来处理异常错误,与 Java 语言的 Try-Catch 模块不同,能够大大减少处理异常的代码量。

5)支持自动垃圾回收。Go 语言中不需要 delete 关键字,也不需要 free()方法来明确释放内存。

2.6 开发语言的对比

以上开发语言中,基于以太坊 EVM 的 Solidity 在开发活跃度和普及率上远超其他智能合约语言,类似于 JavaScript 的语法,能够让开发者易于掌握并快速创建应用核心代码。相比其他语言,Solidity 中增加了与以太坊和交易相关的属性,需要开发者进行熟悉,比如其中的 Storage 和 Payable 属性,为了安全起见,在函数应用中应多加注意。Solidity 支持通用计算,理论上能够实现任何应用场景的程序设计。

比特币脚本包含指令和数据两部分,支持的指令不超过 200 个,开发者很容易学习并精通脚本的编写,开发难度很小。比特币脚本利用栈空间对数据元素进行出栈和入栈操作,从而实现比特币的输入和输出。由于不具备循环、条件和跳转操作,其能够实现的逻辑非常有限,仅应用于基础的数字货币所有权的转移机制,相比于 Solidity,应用范围单一。

Pact 语言则介于两者之间,受比特币脚本语言启发,它的代码采用嵌入式方式直接运行在区块链中,具有 keyset 公钥验证模式,能够实现几乎所有的交易应用。为了避免智能合约编码缺陷造成的应用漏洞,该语言采用非图灵完备设计,没有循环结构和递归操作[23]。相比其他被编译成机器码后部署在区块链上的智能合约,Pact 智能合约明确地展示了区块链上运行的代码,利于人工验证和审核。在安全性方面,Pact 优于 Solidity 等图灵完备语言。

其他区块链系统的智能合约一般可以采用通用语言进行开发,如 Hyperledger Fabric 的智能合约基于 Docker 运行,可以使用 GO,Java 等语言进行编写;Corda 智能合约基于 Java 虚拟机 JVM 运行,可以使用 Java 等在 JVM 上运行的编程语言进行开发。GO 和 Java 属于图灵完备的高级语言,通用性高,能够实现任何应用逻辑,但对于开发者而言,学习难度稍大。表 2 对比了几种语言的特性。

表 2 智能合约语言的对比

Table 2 Comparison of smart contract languages

语言	运行平台	图灵完备性	开发难易程度	数据存储类型	应用复杂性	应用安全性
比特币脚本	Bitcoin	非图灵完备	操作码数量少,开发难度小	基于交易	简单	较高
Solidity/Serpent/Mutan/ LLL	Ethereum	图灵完备	易于掌握,开发难度较小	基于账户	复杂	一般
Pact	Kadena	非图灵完备	代码语法利于执行,但开发有难度	基于表	一般	较高
Go/ Java	Hyperledger	图灵完备	Java 体系庞大,Go 开发难度稍高	基于账户	复杂	一般
C/C++	EOS	图灵完备	低级语言,开发难度高	基于账户	复杂	一般

3 区块链中智能合约的实现技术

现有区块链系统中,智能合约的实现技术可以按照智能合约运行的环境进行划分,具体可分为 3 类:嵌入式运行、基于虚拟机运行和容器式运行。表 3 列举了现有的主流区块链系统及其智能合约的应用类型、运行环境、编程语言。其中,比特币、以太坊和超级账本是当前最为成熟和应用最为广泛的智能合约平台。

表 3 区块链系统的举例

Table 3 Examples of blockchain systems

区块链系统	应用类型	智能合约运行环境	智能合约语言
Ethereum	通用应用	EVM	Solidity,Serpent,Mutan
Hyperledger	通用应用	Docker	Golang,Java
Bitcoin	加密货币	嵌入式运行	Golang,C++
Zcash	加密货币	嵌入式运行	C++
Quorum	通用应用	EVM	Golang
Parity	通用应用	EVM	Solidity,Serpent,Mutan
Litecoin	加密货币	嵌入式运行	Golang,C++
Corda	数字资产	JVM	Kotlin,Java
Sawtooth	通用应用	嵌入式运行	Python

3.1 嵌入式运行

嵌入式运行环境下的智能合约直接嵌入在区块链核心代码中,与区块链本身的其他堆栈代码同时运行。比特币系统和 Kadena[24] 的智能合约均直接与区块链程序本身同时运行。

以比特币系统为例,其采用简单的、基于堆栈的、嵌入式运行的智能合约脚本,实现了基于数字签名的电子货币交易。在每一笔交易中,比特币脚本由 bitcoin core 生成并自动执行,交易由多个输入和输出组成,输入中包含未被花费的 UTXO 及其解锁脚本,输出包含币值和锁定脚本。以上文中用户 A 向用户 B 转账为例,其锁定脚本和解锁脚本如图 1 所示。

验证交易时,将两个脚本组合,栈空间中脚本的执行顺序为从左至右。首先,将解锁脚本两个操作数〈B Signature〉和〈B Public Key〉依次入栈,OP_DUP 为复制操作,堆栈顶元素创建副本,OP_HASH160 对栈顶元素执行 RIPEMD160 哈希运算;然后,将〈B Public Key Hash〉入栈,OP_EQUALVERIFY 验证栈顶两个操作数是否相等,OP_CHECKSIG 验证数字签名与公钥是否匹配,如匹配,则证明用户 B 合法拥有该笔资金。

此外,比特币脚本还可以实现条件稍加复杂的交易,如多重签名、付款至脚本哈希(P2SH)、输出数据记录(RETURN 操作)、时间锁、条件控制等复杂逻辑。然而,脚本不便于自定义,要实现比特币交易以外的复杂应用,嵌入式脚本的表达能力还远远不够。

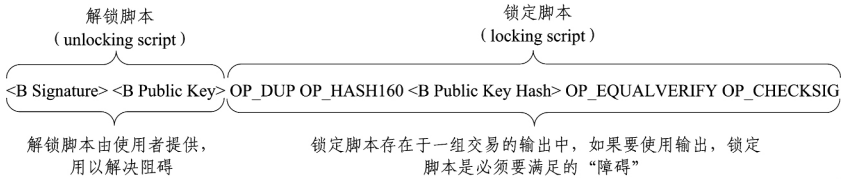

图 1　比特币解锁脚本和锁定脚本

Fig. 1　Bitcoin unlocking and locking scripts

3.2　虚拟机运行

以太坊包含一个以太坊虚拟机 EVM[9]，它是一个完全独立的沙盒，合约代码在 EVM 内部运行并且对外隔离。下面介绍 EVM 的主要机制。

3.2.1　gas 计费机制

在以太坊系统中，为了防止区块链网络资源滥用或由图灵完备引起的无限循环故障，任何可编程的计算都受计费限制。该计费机制以 gas 为单位，创建智能合约、调用消息、访问账户存储的数据，并且虚拟机上的运行操作都对应一定的 gas 计费标准。gas 计费的引入为智能合约的运行提供了机制上的安全保障，一旦 gas 超过计费限制 gasLimit，整个交易将会被回滚，以保证数据的完整性和安全性。

3.2.2　EVM 虚拟机

EVM 虚拟机被部署在执行智能合约操作码的各个节点之上，负责对智能合约进行指令解码，并按照堆栈完全顺序执行代码。其结构不同于标准的冯诺依曼模型，程序代码并非保存在通用内存和永久存储，而是被置于特殊的交互式虚拟 ROM 中。其内存模型和存储模型分别为基于简单字地址的字节数组和字数组，并有可变和不可变之分。虚拟机提供简单栈式结构，为了与 Keccak-256 哈希算法和椭圆曲线算法相匹配，栈的元素大小被设计为 256 位。

EVM 本身运行一个状态函数，也称状态机，用于持续监测状态的变化。当新的进程触发时，EVM 运行代码并将一定数据写入内存或永久存储，每一个新状态都是基于上一个状态进行改变[25]。

3.2.3　合约的创建与运行过程

以太坊系统中，创建合约可看作一种特殊的交易过程，合约创建函数利用一系列固定参数实现新合约的创建，并产生一组新的状态。过程如下：

$$(\sigma', g', A) \equiv \Lambda(\sigma, s, o, g, p, v, i, e)$$

其中，σ 为系统状态，s 为交易发送者，o 为交易源账户主体，g 为可用 gas，p 为 gas 价格，v 为账户金额，i 为初始化 EVM 代码，e 为创建合约栈的深度；σ' 为系统新状态，g' 为剩余 gas，A 为子状态。

最终，通过执行初始化 EVM 代码，创建新的合约账户，产生账户地址、存储空间以及账户的主体代码。该过程中，除去发生交易所消耗的 gas，代码创建的 gas 消耗量与所创建合约的代码量成正比。然而，一旦 gas 剩余量小于代码创建所需 gas，则会产生 gas 异常(Out of Gas，OOG)，并且 gas 剩余量将被置为零，也不再创建新的合约。

合约运行模型则描述了在接收一系列字节码和环境数据元组之后，系统状态的转变方式。在实际运行中，该模型由全系统状态和虚拟机状态的迭代过程构成。迭代器不停地运行迭代函数，直到虚拟机出现状态异常(如 OOG)而暂停或产生正常结果数据而暂停。

智能合约部署的流程如图 2 所示。

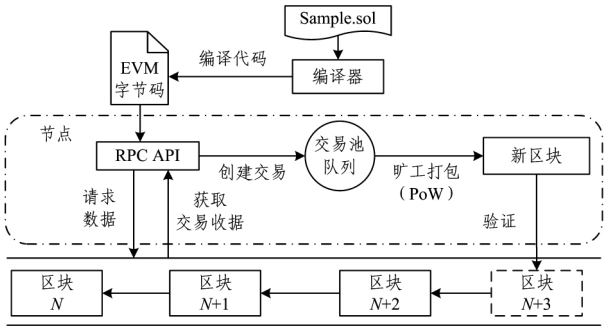

图 2　智能合约部署流程

Fig. 2　Deployment of smart contract

智能合约的部署流程为：1)编写智能合约代码，形成合约代码文件(如 SampleContract. sol)；2)通过智能合约编译器对代码文件进行编译，将其转换成可以在 EVM 虚拟机执行的字节码；3)向区块链节点 RPC API 发送创建交易(部署合约)请求，交易被验证合法后，识别为合约创建交易，检查输入数据，进入交易池；4)矿工打包该交易，生成新的区块，并广播到 P2P 网络；5)节点接收到区块后对交易进行验证和处理，为合约创建 EVM 环境，生成智能合约账户地址，并将区块入链；6)API 获取智能合约创建交易的收据，得到智能合约账户地址，部署完成。

智能合约调用流程与部署流程类似，也是通过 RPC API 创建交易，并由验证节点对交易进行处理，调用 EVM 实例，进行状态变更。

3.3　容器式运行

Hyperledger Fabric 区块链系统中的智能合约称为 Chaincode[26]。Chaincode 分为特殊的系统 Chaincode 和交易 Chaincode，前者负责区块链的管理，后者负责保存状态和账本数据，并执行交易。

3.3.1　Chaincode

Chaincode 运行在一个受保护的 Docker 容器中，是强制执行查询或修改超级账本数据状态的一段程序，可通过应用提交的交易对账本状态初始化并改变当前数据状态(Current State)，执行结果最终提交到网络中的每个账本节点。

3.3.2　Chaincode 生命周期

Chaincode 生命周期分为 5 个阶段：打包(package)、安装(install)、实例化(instantiate)、升级(upgrade)、删除(delete)。Chaincode 通过 API 与区块链网络中的各种节点进行交互，

同时也可以通过 API 对 Chaincode 的生命周期进行管理。

打包过程包括创建包和包的签名，源码被按照部署规范(Chaincode Deployment Spec，CDS)格式打包，签名主要用于检查和确认 Chaincode 所有者，可以在创建包的同时进行签名，一次签名的包用于执行 install 交易，多次签名包为多个所有者依次签名。一个签名包的结构如图 3 所示。

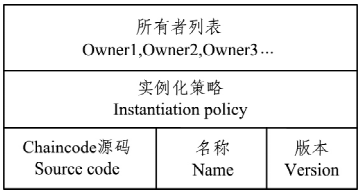

图 3 Chaincode 签名包的结构

Fig. 3 Structure of signature packet

安装过程须指定 CDS 包的路径，发送一条 SignedProposal 消息给 peer 节点的生命周期系统(Lifecycle System Chaincode，LSCC)，节点调用 LSCC 上的 install 方法完成 Chaincode 安装。

实例化过程调用 LSCC，在 channel[26](Ledger 上包含特定 peer 节点的私有链)上启动一个 Chaincode 容器，实现 Chaincode 与 channel 的绑定。实例化交易执行过程中，验证 Chaincode 的实例化策略，以确保实例化交易执行的合法性。实例化成功后，Chaincode 即处于激活状态，时刻监听并接收交易请求。

升级过程类似于实例化过程，即修改新的 Chaincode 版本并与 channel 绑定。为保证升级该 Chaincode 的合法性，该过程须验证当前旧版本的实例化策略。

删除过程只须删除对应的容器，同时删除每个安装 Chaincode 的背书节点上的 SignedCDS。

编写 Chaincode 智能合约时需要实现 Chiancode 接口，以响应传来的交易消息，Init 和 Invoke 是 2 个必需接口，分别实现智能合约的部署(实例化、升级等)和交易调用。

Chaincode 在超级账本上安装部署及运行的流程如图 4 所示，具体过程如下：1)Chaincode 源码以 CDS 规范签名生成 CDS 包；2)通过生命周期系统 Chaincode，将 CDS 包安装在同一通道内的背书节点上，生成运行在节点上的 Chaincode；3)应用程序通过 SDK 发送请求到背书节点；4)节点通过 Chaincode 执行交易并将执行结果返回给应用程序；5)应用程序收集结果，将结果发送给 Oder 共识服务节点；6)共识节点执行共识过程并生成区块验证结果；7)背书节点各自验证交易并提交到超级账本区块中。

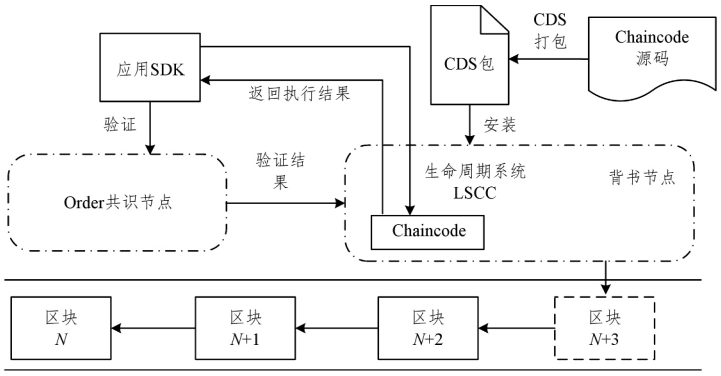

图 4 Chaincode 的部署和执行流程

Fig. 4 Deployment and processing of Chaincode

3.3.3 以太坊和超级账本智能合约的对比

以太坊和超级账本是两个应用最广泛的智能合约平台，但两者在架构设计上面向不同的应用领域。前者定位为完全独立于任何特定应用领域的通用平台，具有对应的账户和代币功能，允许智能合约作为特殊的账户部署在区块链节点上，应用程序通过 API 调用节点上的智能合约来产生交易，变更区块状态。而后者为模块化和可扩展的架构，不具有自身的代币，共识机制采用 PBFT 而非以太坊的 PoW，具有较高的共识效率，而且共识服务从背书节点分离，独立形成可插拔模块，可扩展性强。其主要作为联盟链，面向银行、医疗保健和供应链等行业。因此，基于两个平台的智能合约也具有各自的特点。不同智能合约的特性对比如表 4 所列。

表 4 以太坊和超级账本智能合约的对比

Table 4 Comparison of smart contract in Ethereum and Hyperledger

智能合约特性	以太坊平台	超级账本平台
执行环境	以太坊虚拟机(EVM)	Docker
编写语言	Solidity，Serpent，Mutan	Golang，Java
合约部署	作为交易广播到所有节点，通过矿工挖矿完成部署	直接在所有节点部署
合约升级	无法升级	v1.0 版本可以升级
合约间调用	可调用	许可后可调用
合约终止方式	计步/计价，引入 gas 消耗，对每一个执行指令都消耗 gas，消耗完毕强行终止	计时，以运行时间为标准来判定程序是否进入无限循环，超时后强行终止
库函数	少量对整数、字符串、JSON 等封装的库	Golang 库
加密货币	内置加密货币 Ether，可以利用合约交易加密货币或创建 Token	无内置加密货币，但可利用 Chaincode 创建 Token

3.4　智能合约生态

3.4.1　社区和支持者

嵌入式方式运行智能合约的比特币社区是最早的区块链社区，具有大量中英文文档支持，拥有最权威的区块链技术专家和广泛的支持者。然而由于比特币脚本不能独立创建应用，只有有限的操作模式，且依赖于区块链本身，比特币脚本创建的智能合约仅限于类似于比特币的一类加密货币项目。Kadena 项目也只在官方网站上有 Pact 和 Chainweb 的英文文档，目前支持者相对较少。

Ethereum 社区是目前智能合约开发最活跃的社区，Ethereum 也是智能合约开发者最多的区块链平台，在 ethercasts[27] 上已经有 1 330 个 DApp 发布在公有链。根据 Meetup[28] 数据统计，Ethereum 话题成员数达到 875 211。Ethereum 在 GitHub 上的官方地址也发布了大量技术文档，包括 Solidity 语言、Truffle 框架和 Web3.js 等相关配套工具的英文详细说明，内容非常丰富，中文社区和文档也如雨后春笋般涌现。

Hyperledger Fabric 平台在 GitHub 上开源并提供详细的英文文档，并有一定数量的中英文社区。据 Meetup 数据统计，其话题成员数为 122 871，与 Ethereum 有较大差距。

3.4.2　技术和工具支持

嵌入式智能合约平台的合约代码一般直接运行在区块链上，与交易同时执行，且与区块链系统本身同时开发，合约开发并不需要单独的开发工具。

Ethereum 具有 Mist 和 MetaMask 等前端客户端钱包，合约开发、部署、编译工具 Truffle，以及 JSON RPC 接口，通过该接口可以使前端 Web3 与智能合约进行通信。

Hyperledger 官方提供 SDK 开发包，编译、打包、部署工具，以及每个模块对应的 API 接口，可扩展性非常灵活。

3.4.3　合约的应用范围

由于运行平台和脚本语言表达能力的限制，嵌入式智能合约难以支持复杂的合约开发。但对于加密货币而言，这种设计既能够满足其功能性，又能够保证安全性。因此，嵌入式智能合约应用范围单一，目前多用于构建加密货币的交易条件。

基于 EVM 的智能合约平台，利用以太坊虚拟机，可以运行具有任何交易方式、任何资产类型、任何权限分配等级的去中心化应用，这使得基于 EVM 的智能合约相比于脚本智能合约有更强大的普适性，不仅在加密货币领域，在数字资产、股权、征信、物联网、医疗等其他领域也能发挥巨大作用。

基于 Docker 容器的 Hyperledger 平台同样可以运行具有通用功能的智能合约代码，而且 Hyperledger 引入了多通道(Multichannel)[29] 设计，使得不同通道互联节点之间的数据被隔离，相比 EVM 平台具有更好的隐私性，适用于数据敏感型商业应用。

上述 3 类平台在社区支持、技术文档和开发工具、合约应用范围方面的综合对比如表 5 所列。

表 5　各类型智能合约平台的对比

Table 5　Comparison of various smart contract platforms

平台	社区支持	技术文档	开发工具	合约的应用范围
嵌入式合约平台	除比特币外，其他社区支持较少	比特币文档丰富，Kadena 只有英文文档	Btcdeb 调试工具 Pact Tool	一般为加密货币
虚拟机合约平台	社区活跃，支持者最多	中英文文档均丰富	Mist MetaMask Truffle Remix	通用合约，应用广泛
容器式合约平台	社区活跃，支持者多	中英文文档较丰富	Vscode go Docker	通用合约，偏隐私性

3.4.4　智能合约生态系统

类似比特币的嵌入式智能合约平台的社区项目种类较少，主要针对主链钱包进行开发。Bitcoind[30] 是一个实现比特币协议的程序，用于远程过程调用(RPC)，是比特币网络的第二个客户端。Bitcoind 是一个多线程 C ++ 程序，旨在跨 Windows，Mac 和 Linux 系统移植。BitCoinJ[31] 是一个使用比特币协议的库，它可以维护钱包，发送/接收交易而无需 Bitcoin Core 的本地副本，具有高度优化的轻量级简化支付验证(SPV)模式，只下载一小部分区块链便可验证，使比特币适合在智能手机或廉价虚拟专用服务器等受限设备上使用。通过 BitCoinJ API 可以构建与比特币网络交互的 Java 应用程序。Bitcoin Wallet 和 Multibit[32] 均基于 BitCoinJ 开发。

以太坊社区逐渐形成了自己完善的生态系统。以太坊项目不仅为智能合约开发者提供了完整的开发工具，而且具有一系列与智能合约相关的衍生项目。Swarm[33] 是一个去中心化的内容存储和分发服务。作为以太坊 Web3 的基础层服务，Swarm 采用点对点网络的形式，使节点间相互交换存储资源和消息转发、支付处理服务。Swarm 能够为智能合约以及去中心化应用(DApps)提供文件数据的分布式存储。与 IPFS[34] 相比，Swarm 旨在成为以太坊区块链的全面内置解决方案。Whisper[35] 是去中心化应用相互通信的通信协议，适用于去中心化应用之间相互传送少量信息并需要持续相当长一段时间的场景，具有低带宽、通信加密、接口限定等特点。Truffle[36] 和 Embark[37] 是用于开发以太坊 DApps 的两个最常用的框架，它们抽象出在区块链上编译和部署智能合约的许多复杂步骤。WeiFund[38] 是一个基于以太坊区块链的众筹活动项目，基于该项目可以发布新的 Token。WeiFund 智能合约实现了众筹捐赠的核心机制——退款，如果活动未到达活动目标，将触发退款机制。用户可根据需要，使用项目提供的标准智能合约或使用可定制的用户界面启动一个众筹活动。Hyperledger 社区的 Hyperledger Burrow[39] 为模块化区块链客户端提供了一个许可智能合约解释器，该解释器部分基于以太坊虚拟机的规范开发，在 PoS 共识引擎的基础上提供高吞吐量的交易处理。Hyperledger Composer[40] 是一套用于构建区块链业务网络的协作工具，使业务所有者和开发人员能够简单、快速地创建智能合约和区块链应用程序来解决业务问题。Hyperledger Sawtooth[41] 是一个高度模块化的企业区块链平台，用于构建分布式账本应用和网络。其将保持分类账本的分布式和智能合约的安全性作为设计理念目标。

Sawtooth 通过将核心系统与应用程序分离来简化区块链应用程序开发。开发人员无须了解核心系统的基础设计，便可以使用自己的编程语言编写智能合约应用逻辑。

4 研究现状和存在的挑战

4.1 智能合约的研究现状

区块链技术研究从 2015 年下半年开始迅速走红，凭借其去中心化特点在全球引起广泛关注[42]，并在 2016 年呈爆发状态。2017 年第一季度，全球加密货币市场总值达到 250 亿美元，创历史新高，并且跃进资产市值回升前十[43]。智能合约作为产生于 20 世纪 90 年代的名词，以区块链技术为契机，逐渐得到全球业界的重视。以太坊项目的发布，为智能合约的研究和关注注入了新的活力。根据 Google Trend[44] 记录，自 2015 年 7 月开始，智能合约的搜索热度逐步增长，并在 2016 年 12 月急剧攀升。

目前热度比较高的智能合约研究项目有 Ethereum，Hyperledger，Hawk，Kadena 等。这些项目提供了一个开源的智能合约平台，并且支持对应的合约编程语言。其中，有众多智能合约项目面向专业的金融资产方向，如 R3 联盟发布的 Corda 分布式账本平台[45]，用于记录和处理金融协议，以保证平台上的金融合约建立于法律之上，具有执行效力。

国内也出现了大批针对区块链和智能合约技术的研究机构，如万向区块链实验室推出区块链即服务（BaaS）平台[46]；蚂蚁金服于 2016 年推出蚂蚁区块链平台，其结合蚂蚁中间件和研发运维体系，实现了基于集群的区块链公益项目；平安集团于 2016 年 5 月加入 R3 联盟，为金融市场设计和实现分布式共享分类账技术；2017 年 10 月，百度加入 Linux 基金会领导的超级账本区块链联盟，旨在利用区块链技术，根据用户需求更好地调整其搜索结果[47]。

4.2 智能合约面临的挑战

智能合约的优点显而易见，它去除了中间人环节，降低了信任成本，加快了合约验证和执行的进程。智能合约基于程序代码实现，一旦部署到区块链，不允许也不能更改，排除了人为干预的可能性。智能合约运行于分布式区块链系统之上，也具有其他一切分布式系统的优势，数据安全和高可用性得到了很好的保证。

然而，目前智能合约在技术和实现上仍有一定的局限性，特别是面临稳定性和安全性问题。

4.2.1 安全性

目前专门应用于智能合约的开发语言成熟度不够，例如基于以太坊平台的开发者对开发语言 Solidity 语义的理解偏差和对图灵完备的灵活性利用不当，并且对变量和操作符的越界缺乏相应的安全限制，导致合约产生了一些安全漏洞[48]。Etherdice 项目曾因 gas 消耗超过预期，导致智能合约执行异常而不得不停止维护[49]。2016 年 6 月，The DAO 项目被攻击[50]，黑客利用 DAO 智能合约中的 splitDAO 函数漏洞盗取了价值 6000 万美元的以太币，致使 The DAO 不得不采用硬分叉的方式来消除影响。由于以太坊智能合约一旦部署便不可修改，因此解决这些问题变得更加困难。如果智能合约实现其预期的功能，那么其运行时行为也将是预期的行为，这是由共识协议确保的。如果智能合约包含错误，则没有直接修补它的方法，因此程序员必须在实现智能合约时设计修改或终止合约的方法[51]，或者事件发生后采用硬分叉的对策。但这与以太坊“代码即是法律”的原则相悖。同时，这种解决方案也不会得到整个以太坊社区的同意，部分矿工拒绝分叉，形成另一条区块链。

随着智能合约语言的发展及编译器等构建工具版本的迭代更新，语法、编译 bug 等与安全相关的问题会逐步得到解决。例如，为了解决 Solidity 数学运算越界溢出问题而发布的 SafeMath 库，为整数运算溢出和下溢提供了检测机制，保证了交易中代币余额的正确性。已发布智能合约的不可修改，容易导致不可修复的安全漏洞，因此发布之前进行智能合约代码审计是保证合约安全性的必要手段，目前有些合约代码审计机构已经出现，未来有望形成智能合约代码审计的标准。同时，类似于 Imandra Contracts[54] 的形式化验证项目，可以让智能合约开发者更有效地发现漏洞。以上方法都能在一定程度上提高智能合约的安全性。

4.2.2 并发性和可扩展性

受制于区块链应用平台，智能合约的执行性能和数据处理能力不高。以太坊、Hyperledger 等平台目前均采用全网节点共享单链的方案，每个节点需要处理全部交易数据，导致整个区块链平台的数据处理能力不大于单节点的处理能力[52]。虽然以太坊紫皮书和 Hyperledger 分别提出了分片（Sharding）[53] 和多通道（Multichannel）的解决方案，有望在以后的版本中实现，但目前能够应用的区块链平台还是以单链为主。

另外，区块存储也不具备可扩展性。区块链的结构决定数据记录只能追加，不能删除和修改。当记录有错误时，区块链的解决办法是在后面追加正确的记录，这样完整的修订历史能够被保存，节点只需对全网的完整交易历史进行验证即可。这种机制有利于去中心化系统运行，但严重限制了存储的可扩展性，因为随着交易数量的增加，区块数据最终会占满整个磁盘空间。

针对区块链平台的交易处理速率（Transaction per Second，TPS）较低、交易串行化导致交易规模受限的问题，以太坊推出分片方案，将整个系统节点划分为 100 个分片，每个分片作为子区块链独立验证交易，并通过验证人管理合约来管理分片验证人和跨分片通信等。分片方案应用之后，以太坊的交易并发将得到百倍提升。对于可扩展性，Hyperledger 的架构设计具有明显优势，从 v0.6 到 v1.0，随着版本的更新，Hyperledger 实现了共识服务从 Peer 节点的分离，使得共识服务具有可插拔特性，支持多种共识服务接入；同时，实现了多通道结构，可以将不同业务应用分区隔离，既增强了扩展性和灵活性，也提升了系统的安全性。以上两者均属于以多链方案来解决区块链系统并行性和可扩展性问题。另一种可行方案为侧链，侧链作为单独的区块链系统，与主链之间进行区块头部信息的交互，通过获取主链状态信息来实现虚拟货币的兑换，如 BlockStream 推出的侧链[55]。不过，该方案仅限于

虚拟货币交易的扩展，不适用于其他应用场景。关于区块链系统并行性和扩展性的提升，未来需要更多研究。

4.2.3　可维护性

一些区块链系统的智能合约代码（如比特币和以太坊）一旦部署到网络便不可更改，导致可维护性差，给整个区块链系统带来严重隐患。基于以太坊开发的合约，如果代码中包含漏洞，将无法对其进行升级修补，只能通过重新部署一套智能合约来替换，造成区块链上存储空间的浪费。

智能合约维护困难的底层根源在于区块链系统的不可篡改性，该特性正是区块链系统优于传统架构之所在。为提高智能合约的可维护性，应当专注于智能合约在自身升级迭代和销毁等接口方面的研究，利用合约代码实现多样的管理。

5　智能合约未来应用展望

区块链在各领域的应用是智能合约的发展方向，从人类文明及社会发展的角度看，区块链的终极应用形式为可编程的社会，或者分散式的自治社会[56]。智能合约的自动化和可编程特点为其带来了广阔的应用场景。

自动化执行合同代码在商业领域的合同执行中尤为重要，合同的执行必须是由某个特定事件触发，并由代码自动完成，排除人为干预，利用去中心化信任机制，以低运行成本实现点与点之间的数据交换，在金融和证券领域将首先得到广泛的实际应用，并可为共享经济提供新一轮的发展动力。

可编程性让智能合约基于区块链的新型技术平台，有望实现传统系统平台上的几乎所有业务逻辑。智能合约使得区块链技术的应用不局限于加密数字货币，未来在数据库、文件存储等计算机基础支撑系统方面将有更加成熟且稳定的方案出现。除金融和计算机领域之外，在法律、证券、征信、教育、医疗等各个应用场景都将看到智能合约的身影，其作为新技术平台下的基础编程模式，将助力智能社会的应用革命，为互联网在社会各领域的全面改革提供新的技术基础。

结束语　逻辑简单的脚本类智能合约，在有限的执行环境中能够保证加密数字货币的安全性，能够防止形成脚本漏洞而被恶意攻击者所利用。基于目前已经得到广泛应用的以太坊等区块链平台，可实现比较复杂和灵活的智能合约逻辑，使得区块链能够支持宏观金融和社会系统的诸多应用。可验证型智能合约，能够为商业领域提供安全、高效的交易保障。归纳而言，智能合约为静态的底层区块链数据赋予了灵活可编程的机制和算法，并为构建区块链 2.0 和 3.0 时代的可编程金融系统与社会系统奠定了基础。本文结合不同的智能合约平台，探讨了智能合约原理以及研究现状，分析不同智能合约平台适用的应用场景，对智能合约应用面临的挑战和可行的解决方案进行总结和展望。随着区块链和智能合约技术的发展，有更多智能合约应用问题值得深入研究。

参 考 文 献

[1] NAKAMOTO S. Bitcoin：A peer-to-peer electronic cash system [EB/OL]. https://bitcoin.org/bitcoin.pdf.

[2] ZHANG B. The application and related enlightenment of blockchain technology abroad[J]. Financial Technology Time，2016(5)：7. (in Chinese)
张波. 国外区块链技术的运用情况及相关启示[J]. 金融科技时代，2016(5)：7.

[3] ETHEREUM. Ethereum blockchain app platform[EB/OL]. https://www.ethereum.org/.

[4] HYPERLEDGER. About Hyperledger[EB/OL]. https://www.hyperledger.org/about.

[5] SWAN M. Blockchain：Blueprint for a new economy[M]. O'Reilly Media，Inc.，2015.

[6] NICK S. Smart Contracts：Building Blocks for Digital Markets [EB/OL]. http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html.

[7] NICK S. Exploding Onto The Web[EB/OL]. https://archive.is/zWbhL#selection-607.411-607.470.

[8] BROWN R G. A simple model for smart contracts [EB/OL]. https://gendal.me/2015/02/10/a-simple-model-for-smart-contracts/.

[9] DELMOLINO K，ARNETT M，KOSBA A，et al. Step by step towards creating a safe smart contract：Lessons and insights from a cryptocurrency lab[C]// International Conference on Financial Cryptography and Data Security. Springer Berlin Heidelberg，2016：79-94.

[10] WIKIPEDIA. Ethereum[EB/OL]. https://zh.wikipedia.org/wiki/%E4%BB%A5%E5%A4%AA%E5%9D%8A.

[11] DINH T T A，WANG J，CHEN G，et al. BLOCKBENCH：A Framework for Analyzing Private Blockchains[C]// Proceedings of the 2017 ACM International Conference on Management of Data. ACM，2017：1085-1100.

[12] ANDROULAKI E，BARGER A，BORTNIKOV V，et al. Hyperledger fabric：a distributed operating system for permissioned blockchains[C]// Proceedings of the Thirteenth EuroSys Conference. ACM，2018：30.

[13] DINH T T A，LIU R，ZHANG M，et al. Untangling Blockchain：A Data Processing View of Blockchain Systems[J]. arXiv：1708.05665，2017.

[14] BITCOINWIKI. Script[EB/OL]. https://en.bitcoin.it/wiki/Script.

[15] WOOD G. Ethereum：A secure decentralised generalised transaction ledger[EB/OL]. https://github.com/ethereum/wiki/wiki/White-Paper.

[16] POPEJOY S. The Pact smart contract language[EB/OL]. http://kadena.io/docs/Kadena-PactWhitepaper.pdf.

[17] WIKIPEDIA. Turing_machine[EB/OL]. https://en.wikipedia.org/wiki/Turing_machine.

[18] MÜLLER P，BERGSTRÄβER S，RIZK A，et al. The Bitcoin Universe：An Architectural Overview of the Bitcoin Blockchain [C]// DFN-Forum Kommunikationstechnologien. Bonn：Gesellschaft für Informatik eV，2018：1-20.

[19] ANTONOPOULOS A M. Mastering Bitcoin：unlocking digital cryptocurrencies[M]. O'Reilly Media，Inc.，2014.

[20] JEREMYVINFOLIO. Serpent[EB/OL]. https://github.com/

ethereum/wiki/wiki/Serpent.

[21] ETHEREUM. Solidity[EB/OL]. http://solidity.readthedocs.io/en/latest/.

[22] ETHEREUM. Mutan [EB/OL]. https://github.com/ethere-um/go-ethereum/wiki/Mutan-0.2.

[23] POPEJOY S. The Pact smart contract language[EB/OL]. http://kadena.io/docs/Kadena-PactWhitepaper.pdf.

[24] KADENA LLC. Kadena[EB/OL]. http://kadena.io/.

[25] DANNEN C. The EVM[M]//Introducing Ethereum and Solidity. Apress,2017:47-67.

[26] HYPERLEDGER. Chaincode Tutorials[EB/OL]. http://hyperledger-fabric.readthedocs.io/en/latest/chaincode.html.

[27] STATE OF THE DAPPS. Featured Dapp Collections[EB/OL]. https://dapps.ethercasts.com.

[28] MEETUP. Ethereum Meetups[EB/OL]. https://www.meetup.com/topics/ethereum/.

[29] NGUYEN B,CACHIN C,YELLICK J,et al. Multichannel consensus[EB/OL]. https://lists.hyperledger.org/pipermail/hyperledger-fabric/2016-October/000389.html.

[30] LUKE-JR,SMTP,JONATHANCROSS. bitcoind[EB/OL]. https://en.bitcoin.it/wiki/Bitcoind.

[31] BITCOINJ. What is bitcoinj[EB/OL]. https://bitcoinj.github.io/#community.

[32] KEEPKEY LLC. Learn Bitcoin [EB/OL]. https://multibit.org/learn-bitcoin/faq.html.

[33] SWARM. Swarm Introduction[EB/OL]. https://swarm-guide.readthedocs.io/en/latest/introduction.html.

[34] BENET J. IPFS-content addressed,versioned,P2P file system [J]. arXiv:1407.3561,2014.

[35] JAMES R. Whisper[EB/OL]. https://github.com/ethereum/wiki/wiki/Whisper.

[36] TRUFFLE SUITE. Truffle Overview[EB/OL]. https://truffleframework.com/docs/truffle/overview.

[37] STATUS IM. Build Powerful DApps Easily[EB/OL]. https://embark.status.im/.

[38] WEIFUND. Weifund is Decentralized Secure Open-sourced Interoperable Crowd-funding[EB/OL]. http://weifund.io/.

[39] LINUX FOUNDATION. Hyperledger Burrow[EB/OL]. https://www.hyperledger.org/projects/hyperledger-burrow.

[40] LINUX FOUNDATION. Hyperledger Composer[EB/OL]. https://www.hyperledger.org/projects/composer.

[41] LINUX FOUNDATION. Hyperledger Sawtooth[EB/OL]. https://sawtooth.hyperledger.org.

[42] CHENG H,YANG Y Z. Research on the Development Trend of Blockchain and the Coping Strategies of Commercial Banks[J]. Financial Regulation Research,2016(6):73-91. (in Chinese)
程华,杨云志. 区块链发展趋势与商业银行应对策略研究[J]. 金融监管研究,2016(6):73-91.

[43] COINDESK'S. State of blockchain Q1 2017[EB/OL]. https://www.coindesk.com/research/state-blockchain-q1-2017/.

[44] GOOGLE. Trends[EB/OL]. https://trends.google.com.hk/trends/.

[45] BROWN R G,CARLYLE J,GRIGG I,et al. Corda:An Introduction[EB/OL]. https://docs.corda.net/_static/corda-introductory-whitepaper.pdf.

[46] XIAO F,LUO R G. WXBLOCKCHAIN[EB/OL]. http://www.wxblockchain.com/.

[47] SUJHA S. Chinese-search-giant-baidu-joins-hyperledger-blockchain-consortium[EB/OL]. https://www.coindesk.com/chinese-search-giant-baidu-joins-hyperledger-blockchain-consortium/.

[48] ATZEI N,BARTOLETTI M,CIMOLI T. A Survey of Attacks on Ethereum Smart Contracts (SoK)[C]// International Conference on Principles of Security and Trust. Berlin:Springer,2017:164-186.

[49] VNOVAK. EtherDice smart contract is down for maintenance [EB/OL]. https://www:reddit:com/r/ethereum/comments/47f028/etherdice is down for maintenance we are having/.

[50] DAVID S. Understanding the DAO attack[EB/OL]. https://-www.coindesk.com/understanding-dao-hack-journalists/.

[51] MARINO B,JUELS A. Setting standards for altering and undoing smart contracts[C]// International Symposium on Rules and Rule Markup Languages for the Semantic Web. Cham:Springer,2016:151-166.

[52] SHAO Q F,JIN C Q,ZHANG Z,et al. Blockchain:Architecture and Research Progress[J]. Chinese Journal of Computers,2018,41(5):969-988. (in Chinese)
邵奇峰,金澈清,张召,等. 区块链技术:架构及进展[J]. 计算机学报,2018,41(5):969-988.

[53] BUTERIN V. Ethereum 2.0 mauve paper[EB/OL]. https://cdn.hackaday.io/files/10879465447136/Mauve%20Paper%20Vitalik.pdf.

[54] GRANT P,KOSTYA K. Imandra Contracts[EB/OL]. https://github.com/AestheticIntegration/contracts.

[55] BACK A,CORALLO M,DASHJRL,et al. Enabling blockchain innovations with pegged sidechains[OL]. tinyurl.com/mj656p7.

[56] ZHANG L J. Blockchain Applications will be more Developed Towards Smart Contracts in the Future[J]. Cards World,2016(8):20-21. (in Chinese)
张立钧. 未来区块链应用更多向智能合约发展[J]. 金卡工程,2016(8):20-21.