



(12) 发明专利申请

(10) 申请公布号 CN 114679282 A

(43) 申请公布日 2022. 06. 28

(21) 申请号 202210315822.3

(22) 申请日 2017.02.14

(30) 优先权数据

1603112.2 2016.02.23 GB

1603114.8 2016.02.23 GB

(62) 分案原申请数据

201780009343.6 2017.02.14

(71) 申请人 区块链控股有限公司

地址 安提瓜和巴布达圣约翰

(72) 发明人 克雷格·史蒂文·赖特

史蒂芬·萨凡纳

(74) 专利代理机构 隆天知识产权代理有限公司

72003

专利代理师 石海霞

(51) Int.Cl.

H04L 9/32 (2006.01)

H04L 9/06 (2006.01)

G06F 16/27 (2019.01)

G06Q 20/38 (2012.01)

权利要求书3页 说明书13页 附图5页

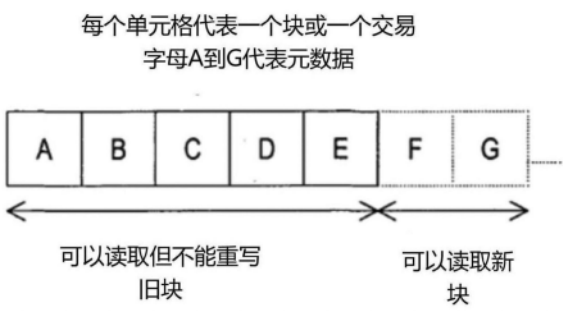
(54) 发明名称

用区块链实施的用于安全投票和分配的计数系统和方法

(57) 摘要

本发明主要涉及区块链实施方案,适用于但不限于与比特币区块链一起使用。本发明涉及用于管理投票、计数、选择和/或决策过程的技术方案,可用于实现自动化处理,如设备/系统控制、过程控制、分布式计算和存储等。本发明提供了一种事件检测、监视和/或计数机制。该事件可以是:例如由给定实体进行的投票、决策或选择。本发明提供一种计数解决方案,其中的计算资源与区块链同时并行运行,并管理基于循环的操作。计算资源持续监视区块链的状态以及任何其他区块链外的输入数据或源。循环的执行受到区块链状态的影响。由计算资源所执行循环的每次迭代都记录在写入区块链的交易中,且作为哈希存储在交易的元数据中。如果计算资源找到包含与循环相关的哈希交易,则它将访问代码的相关部分。循环包含一个条件语句,该条件语句使计算资源能够决定采取哪种操作。而条件可能取决于

区块链的状态或任何其他数据源。该操作可以是区块链上或链外任何类型的操作。因此,计算资源和区块链的结合提供了图灵完备解决方案。



1. 一种计算机实施的方法,用于计数、监控、检测、记录或处理事件,其中,所述事件是投票、选择、操作或决策,所述方法包括以下步骤:

将至少一个令牌分发或分配给至少一个实体,所述至少一个实体用于生成至少一个事件,其中:

- i) 所述至少一个令牌由加密公钥和私钥对表示;并且
 - ii) 通过将交易广播到区块链网络来执行分发或分配,
- 以及

执行、监督或管理至少一个计算资源上的循环,所述至少一个计算资源与区块链或区块链相关平台并行或一起运行,并用于维护由所述至少一个实体生成的所述至少一个事件的记录或计数。

2. 根据权利要求1所述的方法,其中,所述至少一个计算资源用于将所述区块链用作图灵机的不可擦除磁带。

3. 根据权利要求1或2所述的方法,其中,所述至少一个计算资源用于实施图灵机的控制堆栈。

4. 根据任一前述权利要求所述的方法,其中,由所述至少一个计算资源执行或管理的循环是基于图灵完备语言。

5. 根据任一前述权利要求所述的方法,其中,所述公钥和私钥对与以下各项相关联:
所述至少一个事件;或者

一个或多个选择或可能的操作方案;或者

代表所述至少一个事件的项目,优选地,其中,所述项目是加密货币量。

6. 根据任一前述权利要求所述的方法,其中,所述事件与由所述至少一个计算资源接收、检测、记录或生成的数据有关,或者与所述区块链的状态有关。

7. 根据任一前述权利要求所述的方法,包括以下步骤:

i) 使用所述至少一个计算资源来决定要广播到所述区块链网络并写入所述区块链的一组交易;或者

ii) 使用区块链堆栈作为数据存储空间。

8. 根据任一前述权利要求所述的方法,其中,所述至少一个计算资源:

i) 存储与所述循环有关的信息;和/或

ii) 是谕示或机器人。

9. 根据任一前述权利要求所述的方法,其中,所述区块链或区块链平台与以下各项相关联:

i) 功能有限的脚本语言,所述功能有限的脚本语言不能天然地支持将控制结构并入用所述语言编写的代码中,所述控制结构诸如为循环;和/或

ii) 区块链网络。

10. 根据任一前述权利要求所述的方法,包括以下步骤:

使用脚本实施所述循环;以及

优选地,生成所述脚本的加密哈希。

11. 根据任一前述权利要求所述的方法,还包括以下步骤:

i) 使用秘密交换协议将所述至少一个令牌分发或分配给所述至少一个实体;或者

ii) 生成与所述循环相关的代码的加密哈希,并且优选地,将所述加密哈希存储在所述区块链上的交易中。

12. 根据任一前述权利要求所述的方法,其中:

i) 与所述循环的至少一次迭代有关的信息存储在所述区块链上的交易中;优选地,其中,所述信息作为元数据存储在所述交易中;和/或

ii) 所述计算资源用于针对包括与所述循环相关的代码的加密哈希的交易来监控所述区块链的状态。

13. 根据任一前述权利要求所述的方法,还包括以下步骤:

针对所述循环的每次迭代:

评估条件并基于评估的结果执行至少一个操作,其中,所述至少一个操作包括:

引起至少一个交易被写入所述区块链;和/或

引起链外操作的执行。

14. 根据权利要求13所述的方法,其中,所述条件涉及:

由所述计算资源接收、检测或生成的数据;或者

所述区块链的状态。

15. 根据任一前述权利要求所述的方法,其中,所述计算资源用于监控:

所述区块链的状态;由所述计算资源生成或接收的值;和/或在所述区块链外提供的数据或信号源。

16. 根据任一前述权利要求所述的方法,包括以下步骤:

i) 使用所述区块链作为数据、指令或数据指针和/或指令指针的存储组件;和/或

ii) 使用计算资源作为图灵完备过程的控制流管理组件,所述计算资源用于执行循环机制。

17. 根据任一前述权利要求所述的方法,还包括以下步骤:

如果所述计算资源在所述区块链内的交易中找到部分代码的预定哈希,则在指定迭代处重新启动所述循环;

优选地,其中,与所述迭代有关的信息是使用在所述交易内提供的或与所述交易相关联地提供的元数据来指定的。

18. 根据任一前述权利要求所述的方法,其中,所述循环的代码:

i) 硬编码到所述计算资源之中或之上;

ii) 存储在私人或公共可用文件中;和/或

iii) 作为条目存储在私人或公共哈希表文件中;和/或

iv) 是具有硬编码变量或至少一个参数的静态代码块;和/或

v) 与至少一个参数相关联或包括至少一个参数,所述至少一个参数:

使用任何数据格式的单个值来填充、初始化或实例化;

是部分代码;

从区块链交易中的元数据或从所述计算资源外部的源中获取;

从数据库、私人或公共文件或哈希表中获取;和/或

使用值填充,所述值使用至少一个数据源指针来访问,优选地,其中,所述至少一个指针作为元数据存储在所述区块链上的交易中。

19. 根据任一前述权利要求所述的方法, 其中, 所述计算资源包括注册表或与注册表通信, 所述注册表使得所述计算资源能够访问所述循环的预存储版本; 优选地, 其中, 所述注册表存储:

- i) 与所述循环相关的代码的加密哈希; 和/或
- ii) 指示能够从中访问所述代码的副本的位置的信息。

20. 根据任一前述权利要求所述的方法, 还包括以下步骤:

使用区块链交易更新所述循环的代码, 以便用新代码替换现有代码; 并且
优选地:

- i) 其中, 所述交易是多签名P2SH交易; 和/或
- ii) 所述方法包括存储所述现有代码的哈希和所述新代码的哈希的步骤。

21. 一种基于计算机的系统, 用于实施任一前述权利要求所述的方法, 包括:

区块链; 和

计算资源, 用于执行循环和/或监控所述区块链的状态。

用区块链实施的用于安全投票和分配的计数系统和方法

[0001] 本申请是申请日为2017年2月14日,申请号为201780009343.6,发明名称为“用区块链实施的用于安全投票和分配的计数系统和方法”的PCT国家阶段申请的分案申请。

技术领域

[0002] 本发明主要涉及区块链实施方式和技术。本发明特别适合但不限于与比特币区块链一起使用以及用于诸如设备/系统控制,过程控制,分布式计算和存储的应用。特别地,本发明涉及用于管理投票、计数、选择和/或决策过程的技术解决方案。本发明不限于在政治投票应用或场景中使用。

背景技术

[0003] 在本文档中,我们使用术语“区块链”包括基于共识的电子类、基于计算机的、以及分布式账簿的所有形式,其包括但不限于区块链和交易链技术,许可和未许可的账簿,共享账簿及其变体。尽管其他区块链实施方案已经被提出并开发,但是区块链技术最广为人知的应用是比特币账簿。虽然出于方便和说明的目的本文会提及比特币,但需要注意的是,本发明不限于与比特币区块链及其他区块链实施方案一起使用。此外,协议也属于本发明的范围。

[0004] 区块链是一种基于共识的电子账簿。电子账簿是一种基于计算机的分散式分布式系统,由块组成,而这些块又由交易组成。每个交易是编码区块链系统中参与者之间的数字资产控制转移的数据结构,并且包括至少一个输入和至少一个输出。每个块包含前一个块的哈希。这些块被链接在一起以创建一个永久的不可更改的记录—记录自区块链开始创建以来写入区块链的所有交易。交易包含嵌入其输入和输出中的称为脚本的小程序,这些程序指定了如何以及通过谁来访问交易的输出。在比特币平台上,这些脚本是使用基于堆栈的脚本语言编写的。

[0005] 为了将交易写入区块链,必须对其进行“验证”。网络节点(矿工,miner)执行该工作以确保每个交易都有效,并拒绝网络中的无效交易。安装在节点上的软件客户端通过执行其锁定和解锁脚本功能对未使用完的交易(Unspent transaction,UTXO)执行此验证工作。如果锁定和解锁脚本的执行评估为TRUE,则交易有效并且将被写入区块链。因此,为了将交易写入区块链,交易必须i)由接收交易的第一个节点验证-如果交易被验证,则节点将其传送到网络中的其他节点;ii)被添加至由矿工建造的新区块;以及iii)被挖矿,即被添加至过去交易的公共账簿中。

[0006] 尽管区块链技术因用于加密货币实施方案而广为人知,但数字企业家已经开始探索使用比特币所基于的安全加密系统以及可存储在区块链上的数据来实现新系统。这些包括但不限于:

- [0007] • 存储元数据
- [0008] • 实现数字令牌(digital tokens)
- [0009] • 建立使用数字签名签署的合同

[0010] 如果区块链可以用于不限于加密货币领域的自动化任务和过程,那将是非常有利的。这样的解决方案将能够利用区块链的优势(例如:事件的永久性防篡改记录、分布式处理、安全加密功能等),并在应用中更加通用。

[0011] 区块链社区当前感兴趣的一个领域是图灵完备性,特别是如何利用图灵完备性推动区块链技术的研发。出于安全方面的考虑,这些技术的研发旨在限制功能。

[0012] 比特币脚本语言是否是图灵完备的是有争议的,因为它本身不支持复杂的流控制功能,例如,即将发生的循环。这种限制的一个优点是程序具有可预测的执行次数(Execution times)。

[0013] 将比特币脚本限制为线性或树状决策任务的另一个重要优点是:避免了无限循环,这可以用作启动诸如拒绝服务(DoS或DDoS)攻击等漏洞利用的手段。由于这种限制,比特币脚本通常仅限用于线性任务而不用于更复杂的应用程序,如自动化任务的控制、设备管理等。

[0014] 以太坊区块链平台通过结合“内置”图灵完备语言Solidity来解决这个问题。这种语言是以太坊平台的原生语言,因此用Solidity编写的脚本可以包括控制流机制,如循环。但是,以太坊已遭受到了多次攻击和漏洞利用。

[0015] 鉴于上述安全问题,以及比特币脚本语言的广泛使用性和熟悉性,区块链社区的大部分人仍然希望保留与区块链技术相关的这些有限脚本语言的使用。

[0016] 因此,我们期望提供一种解决方案。该方案有助于将图灵完备功能例如循环机制和其他复杂控制结构与区块链脚本集成或结合,同时避免潜在安全弱点(例如无限循环)的破坏性影响。这样的解决方案将具有许多益处,包括:

- [0017] • 实现复杂区块链相关交易的自动化;
- [0018] • 控制记录在区块链上的元数据流;
- [0019] • 扩展区块链平台的功能和应用,这些平台不依赖或有目的地汇编图灵完备语言。

[0020] 现在已经设计出这种改进的解决方案。本发明提供了一种解决方案,其包括并行计算资源与区块链的新的结合,使得能够在典型的区块链脚本之外仿真、模拟和/或编入循环和其他图灵完备功能。反过来,这给与例如分布式数据存储、分布式计算和无人机控制或任何IoT(物联网)设备相关的自动化任务的许多应用提供了便利。此类应用程序可以包括使用区块链进行元数据存储、管理数字令牌和建立合同。另一个有用的应用是投票或选择计数过程的自动化。

[0021] 以下提供本发明技术领域的背景资料:US2016/027229A;W02016/022864A和US6061449A。以下在提交申请时也可获得的在线资源提供了技术背景:

[0022] <https://nxt.org/what-is-nxt/voting/>

[0023] <http://bitcoin-development.narkive.com/uRciVtAQ/standard-bip-draft-turing-pseudocompleteness>

[0024] <http://news.ycombinator.com/item?id=7287155>

[0025] <http://news.ycombinator.com/item?id=11372455>

[0026] <http://cryptonomics.org/2014/02/01/ethereum-turing-complete>

发明内容

[0027] 因此,根据本发明,提供了如附加权利要求中定义的解决方案。根据本发明,可以提供(过程)控制方法和相应的系统。本发明可以称为区块链实施的控制方法/系统。本发明可以控制自动化任务或过程。

[0028] 本发明可用于使用区块链来仿真/模拟图灵完备性。另外/或者,本发明可以使涉及图灵完备控制机制的应用程序能够在区块链平台上得以执行。

[0029] 另外/或者,本发明可被描述为用于使用区块链和/或一个或多个区块链交易来控制块外计算资源上执行的进程的方法或系统。因此,本发明包括一种设置,其中在功能和结构上彼此不同的各种计算部件交互作用以便提供新的技术结果。不同计算系统(计算资源和区块链)的交互作用产生了一个非常强大的控制解决方案。

[0030] 从计算资源的角度来看,本发明提供了程序执行的永久性防篡改记录的优势。从区块链的角度来看,本发明提供了改进的区块链实施方案,因为它能够通过使用区块链使得图灵完备性至少被部分模拟,这反过来使得功能上更为复杂的基于区块链的应用得以利用。虽然保留了区块链交易中有限脚本语言的使用,但是这一切也都得以实现。脚本语言可能是有限的(受限制的),因为其设计或实现防止或至少不能天然地支持将诸如循环之类的复杂控制流机制编入用该语言编写的代码中。语言的指令集,即程序员可以使用的“命令”或“操作码”,可以不包括用于复杂流控制机制的命令。

[0031] 区块链可以与包括一种有限语言(可能是一种脚本语言)的区块链协议相关联或一起使用。本发明可以使用区块链扩展有限脚本语言的功能以执行任务。

[0032] 本发明可以使用区块链的状态来执行基于循环的进程。而基于循环的进程可以在与区块链网络并行操作的计算资源上执行。计算资源可以与区块链网络不同(不是区块链网络的一部分)。计算资源可被称为“oracle”或“bot”。

[0033] 这使得区块链协议能够利用功能有限的脚本语言,同时允许诸如循环机制之类的控制流机制在区块链外实现。这种新的结合增强了区块链技术的多功能性,同时保留了其安全性。

[0034] 根据本发明的一个或多个实施例,本发明可以提供用计算机实施的事件处理、监视、检测和/或计数过程及相应系统。本发明的实施例可以用于由实体生成的事件。本发明可以监视/检测和/或计数投票、选择或决定或其他类型的事件。为了方便起见,术语“投票”或“决定”可用于指代由本发明监视、计数、记录和/或检测等的事件。

[0035] 本发明的一个或多个实施例可描述为控制方法及相应的系统。本发明可以描述为区块链实施的方法/系统。本发明可以用于控制过程。该过程可以是工业或非工业过程。事件例如决定可以由机器、基于电子/软件的实体、自然人、法人或其他做出。本发明的一个或多个实施例可以包括用于计数、监视、记录、检测和/或以其他方式处理事件的方法/系统。事件可以是决定或选择,例如由应用领域内的实体进行的投票。尽管它可以用于在政治背景下计算政治导向的选票或事件,但它并不只限于此目的。例如,它可以用于记录机器进行特定选择的次数,或者细胞变异的次数等。该决定可以是二元或非二元决策。它也可以计算或记录事件发生/未发生的次数。

[0036] 该方法可以包括以下步骤:

[0037] 将至少一个令牌分发或分配给至少一个预定实体,其中至少一个令牌由加密公钥

和私钥对表示；

[0038] 在计算资源上执行循环以维持由至少一个实体生成的一个或多个事件(例如,投票,选择,操作或决定)的计数。

[0039] 本发明可以提供用计算机实施的投票或决策计数过程,包括以下步骤:

[0040] 将至少一个令牌分发或分配给至少一个预定实体,其中至少一个令牌由公钥和私钥对表示;并在计算资源上执行循环以维持由至少一个实体做出的一个或多个投票或决定的计数。

[0041] 加密密钥对可被称为“区块链公钥和私钥对”。如本领域所公知的,它用于和区块链平台一起使用,以及用于区块链交易输入和输出之间的值的安全转移。密钥可以用作区块链地址。

[0042] 本发明可以提供一种用计算机实施的投票或决策计数方法,包括以下步骤:

[0043] 将至少一个令牌分发或分配给至少一个预定实体,其中至少一个令牌由区块链公钥和私钥对表示;在计算资源上执行循环,该循环使用脚本来实现,以维持一个或多个投票或决策的计数,该一个或多个投票或决策由至少一个实体生成或与至少一个实体相关联;并生成脚本的加密哈希。

[0044] 本发明可以提供一种用计算机实施的投票或决策计数方法,包括以下步骤:

[0045] 将至少一个令牌分发或分配给至少一个预定实体,其中至少一个令牌由区块链公钥和私钥对表示;并在计算资源上执行循环以维持由至少一个实体做出的一个或多个投票或决策的计数;

[0046] 其中,与循环的至少一次迭代有关的信息存储在区块链上的交易中。

[0047] 本发明可以提供一种用计算机实施的投票或决策计数方法,包括以下步骤:

[0048] 将至少一个令牌分发或分配给至少一个预定实体,其中至少一个令牌由区块链公钥和私钥对表示;在计算资源上执行循环以维持由至少一个实体做出的一个或多个投票或决策的计数;和

[0049] 生成与循环相关代码的加密哈希;

[0050] 以及优选包括将加密哈希存储在区块链的交易中的步骤。

[0051] 本发明可以提供一种用计算机实施的投票或决策计数方法,包括以下步骤:

[0052] 将至少一个令牌分发或分配给至少一个预定实体,其中至少一个令牌由区块链公钥和私钥对表示;并在计算资源上执行循环以维持由至少一个实体做出的一个或多个投票或决策的计数;

[0053] 其中,计算资源由于一笔交易对区块链的状态进行监视,该交易包括与循环相关的代码的加密哈希。

[0054] 至少一个预定/预选实体可以是机器,或基于电子/软件的实体。在其他实施例中,实体可以是自然人、法人或其他。区块链可以是比特币区块链或其他区块链平台。可以通过广播到区块链网络来执行分配。可能存在多个实体(例如选民或决策者)。可以将至少一个令牌分发和/或分配给多个实体中的每个相应实体。

[0055] 一个或多个投票或决定的计数可以存储在区块链中(例如,在交易中)和/或区块链的备用堆栈中。

[0056] 该方法可以包括使用脚本实现循环的步骤。该脚本可以用脚本语言编写,它可以

基于Forth脚本语言,而该语言可能是非图灵完备的。也可以是基于堆栈的语言。脚本语言可能是有限的(受限制的),因为其设计或实现防止或至少不能天然地支持将诸如循环之类的复杂控制流机制编入用该语言编写的代码中。语言的指令集,即程序员可以使用的“命令”或“操作码”,可以用于不包括用于复杂流控制机制的命令。

[0057] 该方法还可以包括生成脚本或其一部分的加密哈希的步骤。

[0058] 该方法还可以包括使用秘密交换协议将至少一个令牌分发和/或分配给至少一个实体的步骤。

[0059] 该方法还可以包括用表示一个或多个选择的项目加载每个公共密钥和地址的步骤,其中优选项目是货币量。货币可能与比特币有关。

[0060] 该方法还可以包括以下步骤:在计算资源上维护公钥列表或者维护与计算资源相关的公钥列表,公钥列表与每个授权实体和/或令牌相关联。该方法还可以包括销毁(和/或永不存储)私钥列表以及实体身份与其分配的令牌之间的映射/关联的步骤。

[0061] 该方法还可以包括维护地址列表的步骤。该列表可以是空列表或包括一个或多个地址的列表。至少一个地址可以与至少一个实体相关联。优选地,至少列表的一部分在区块链脚本中指定、表示或定义。

[0062] 该方法还可以包括将地址列表传送到(区块链)备用堆栈以便存储的步骤;优选地,一旦对该地址(或其他条件)的投票已被计数、检测和/或以其他方式处理,则该地址将从备用堆栈中移除。

[0063] 该方法还可以包括执行循环直到地址列表变空的步骤。

[0064] 该方法可以包括以下步骤:

[0065] 在计算资源(例如服务器或多个服务器)上执行循环;使用区块链的状态来影响循环的执行;和/或由于执行循环而影响区块链的状态。

[0066] 计算资源可以是任何基于处理器的设备或系统。例如,它可以是服务器或多个服务器;也可以是单机或分布式资源。区块链可以是比特币区块链或任何其他区块链相关平台,也可以是基于共识的电子账簿。

[0067] 与循环的至少一次迭代有关的信息可以存储在区块链上的交易中。该信息可以作为元数据存储存储在交易中。循环可以包含“If condition then action”(ICTA)指令。

[0068] 该方法还可以包括生成与循环有关代码的加密哈希的步骤,并且优选地,将加密哈希存储在区块链上的交易中。代码可以是包含控制流语句的代码块,例如“If condition then action”语句。

[0069] 计算资源可以由一笔交易对区块链的状态进行监视,该交易包括与循环有关的代码的加密哈希。

[0070] 该方法还可以包括以下步骤:

[0071] 对于循环的每次迭代:

[0072] 评估条件并基于评估结果执行至少一个操作,其中至少一个操作包括:

[0073] 触发至少一个交易写入区块链;和/或

[0074] 触发执行一次区块链外操作。

[0075] 该条件可用于监视无论在何处,如何或由谁生成的,以及区块链上或链外的任何值、信号或输入。该条件可以涉及由计算资源接收,检测或生成的数据;和/或区块链的状

态。该条件可以被描述为“触发器”，它可以是或涉及区块链的特定状态，或者在块外检测到的事件（例如，日期或温度读数等），或两者的组合。

[0076] 操作可以包括发送一个触发事件开始的信号，或广播一个新交易，或两者的组合。索引可以维护 (i) 在管理器块外，或者可以是 (ii) 存储在随后广播的交易中的值，(i) 以及 (ii) 表示维护控制数据的两种备选方式。

[0077] 计算资源可用于监视：

[0078] 区块链的状态；计算资源生成或接收的值；和/或从区块链外提供的数据或信号源。

[0079] 该方法可以包括以下步骤：

[0080] i) 使用区块链作为数据、指令或数据指针和/或指令指针的存储部件；和

[0081] ii) 使用计算资源作为图灵完备过程的控制流管理部件，同时计算资源用于执行循环机制。

[0082] 因此，区块链可以用作图灵机的只读磁带。计算资源可用于控制过程的执行流程，实现循环并扩展脚本语言的功能。

[0083] 该方法还可以包括在指定迭代中重启（重新生成）循环的步骤。如果计算资源在区块链内的交易中找到部分代码的预定哈希，则可重启循环。该部分代码可涉及循环体，也可能包括一个ICTA语句。

[0084] 计算资源可以在每次迭代中重新生成循环。这可以以各种方式执行。

[0085] 例如，循环代码块可以：

[0086] 硬编码到计算资源中；

[0087] 存储在私人或公共文件中；

[0088] 存储为私人或公共哈希表文件的条目；

[0089] 或上述方式的组合。

[0090] 代码块可以是具有硬编码变量的静态代码块，也可以是包含可填充参数的静态代码块，而参数可以是任何数据格式的单个值，也可以是小块代码，或者是上述的组合。参数可以通过直接从交易的元数据中（例如比特币交易）或从诸如内部数据库或私人/公共文件或哈希表或上述任何组合的外部源中提取来填充。参数值外部源的指针可以存储在交易的元数据中。

[0091] 可以使用元数据来指定与迭代有关的信息，该元数据在交易中提供或与交易相关。

[0092] 计算资源可以包括注册表或与注册表通信，该注册表使计算资源能够访问子程序的预存版本。或者，注册表可以描述为数据库、存储库或其他形式的存储设备。注册表可以存储：

[0093] i) 与循环相关代码的加密哈希；和

[0094] ii) 表示可以访问代码副本位置的信息；

[0095] 该方法还可以包括使用区块链交易来更新循环代码以便用新代码替换现有代码的步骤。优选地，交易是多签名P2SH交易。可以存储现有代码的哈希和新代码的哈希。

[0096] 本发明还提供了一种用于实现上述方法的任何实施例的系统。

[0097] 本发明可以提供基于计算机的系统。该系统可以描述为计算机实施的事件计数、

监视、检测和/或处理系统。事件可以是投票、决策、选择或任何其他类型的事件。该系统可以用于模拟或仿真图灵完备性。该系统可包括：

[0098] 区块链；和

[0099] 计算资源，其用于执行循环，使得循环的执行受到区块链状态的影响。

[0100] 与循环的至少一次迭代相关的信息存储在区块链上的交易中。优选地，信息作为元数据存储在交易中。

[0101] 优选地，计算资源用于生成与循环有关的代码的加密哈希。优选地，加密哈希存储在区块链上的交易中。另外/或者，计算资源由于一笔交易对区块链的状态进行监视，该交易包括有与循环相关的代码的加密哈希。

[0102] 优选地，对于循环的每次迭代：评估条件并且基于评估结果执行至少一个操作；所述至少一个操作包括：触发至少一个交易写入区块链；和/或

[0103] 触发执行一次区块链外操作。

[0104] 该条件可以涉及由计算资源接收、检测或生成的数据；或区块链的状态。

[0105] 该计算资源可用于监视：

[0106] 区块链的状态；

[0107] 计算资源生成或接收的值；和/或

[0108] 区块链外提供的数据或信号源；

[0109] 区块链可以用作数据、指令或数据指针和/或指令指针的存储部件。计算资源可以用作图灵完备过程的控制流管理部件，同时计算资源用于执行循环机制。

[0110] 如果计算资源在链内的交易中找到部分代码的预定哈希，则可在指定迭代中重启循环。可以使用元数据来指定与迭代有关的信息，该元数据在交易中提供或与交易相关。

[0111] 计算资源可以包括存储设备或者与存储设备通信，该存储设备可称为注册表、数据库或存储库，并且可使计算资源能够访问子程序的预存版本。注册表可以存储：

[0112] i) 与循环相关代码的加密哈希；和

[0113] ii) 表示可以访问代码副本位置的信息。

[0114] 系统可以配置成使用区块链交易来更新循环代码，以使用新代码替换现有代码。优选地，交易是多签名P2SH交易。优选地，系统用于存储现有代码的哈希和新代码的哈希。

[0115] 所描述的与本发明一个方面或实施例相关的任何特征也可适用于任何其他方面或实施例。例如，所描述的与本发明相关的任何特征也可用于相关系统，反之亦然。

附图说明

[0116] 参考本文所述的实施例，本发明的这些和其他方面将变得显而易见并得以阐明。现将仅通过示例及参考附图来描述本发明的实施例，其中：

[0117] 图1示出了区块链作为图灵机只读磁带的说明性使用。

[0118] 图2阐明了一个子程序，管理器可以使用该子程序与区块链一起实现重复循环。

[0119] 图3示出了根据本发明使用ICTA (If Condition Then Action) 代码块的实例。

[0120] 图4示出了根据本发明的实施例允许用户将数据移入和移出备用堆栈的比特币命令。

[0121] 图5示出了根据本发明的实施例的管理器代码注册表。

- [0122] 图6示出了根据本发明的实施例中与管理器代码块相关联的元数据。
- [0123] 图7示出了根据本发明的实施例,在管理器循环的特定迭代中与输出相关联的元数据。
- [0124] 图8示出了本发明的实施例的交易脚本和元数据。
- [0125] 图9示出了说明性管理器软件补丁验证和审计跟踪。
- [0126] 图10示出了本发明的说明性用途,并示出了投票计数机器人在伪代码中重复循环的实施例。

具体实施方式

[0127] 下面描述使用比特币区块链的说明性实施例。然而,也可以使用其他区块链协议和实施方案。本发明在此方面并不受限。

[0128] 本发明解决了如何在操作受限的区块链平台(即,使用了不支持复杂控制机制的脚本语言的区块链平台)上实现图灵完备性的问题,并因此扩展了区块链的用途或应用。马文·明斯基(Marvin Minsky)(明斯基(Minsky)等人,《计算:有限和无限的机器》,普伦蒂斯·霍尔出版社(Prentice Hall Inc),1967)描述了如何使用只读磁带来实现图灵完备的机器,以及如何能够执行也可在通用图灵机上执行的任何算法。

[0129] 本发明包括一种计算资源,该计算资源与区块链一起操作,在图灵机的实现中将其用作只读磁带。该计算资源与区块链网络并行运行,监督和处理循环过程的执行。循环过程用于执行给定任务,例如,过程的自动化或设备或系统的控制。此并行资源监视区块链的状态,并可触发交易写入区块链。因此,为了便于参考,本文称其为“管理器”。

[0130] 本发明的特征和优点包括:

- [0131] • 启用区块链作为图灵机的只读磁带
- [0132] • 与区块链一起运行的基于计算机的监控和管理部件(管理器)的功能和实现
- [0133] • 使用管理器作为图灵机的指令表
- [0134] • 使用代码注册表管理管理器
- [0135] • 与管理器代码和循环重新生成相关的交易元数据
- [0136] • 使用数字签名实现管理器的软件更新
- [0137] • 使用备用区块链的管理器的特殊实现

[0138] 区块链作为图灵机的只读磁带

[0139] 参考图1,本发明利用区块链作为图灵机的只读磁带,具有以下定义和特征:

[0140] 1. 区块链作为图灵机的磁带。区块链中的每笔交易代表磁带上的一個单元格。该单元格可以包含来自有限字母表的符号。

[0141] 2. 磁头可以从已经写入区块链的块中读取信息。

[0142] 3. 磁头可以将包含许多交易的新块写入区块链的末尾,但是无法写入已存在的块;因此,区块链磁带是只读的。

[0143] 4. 每个交易的元数据可以存储为多签名付费到脚本哈希(P2SH)交易的一部分。

[0144] 管理器的一个重要功能是充当监视区块链当前状态的代理。它还可以接收来自任何块外源的信号或输入。

[0145] 根据区块链状态和/或接收到的输入,管理器可以执行某些操作。管理器决定要执

行哪些操作。这些可能涉及或可能不涉及“现实世界”中的操作(即,块外)和/或区块链上的操作(例如创建和广播新交易)。管理器采取的操作可能由区块链状态触发。管理器还可以决定向比特币网络广播的下一组交易,并随后决定将这组交易写入区块链。

[0146] 管理器的操作在比特币网同时并行运行。从某种意义上说,这扩展了行为受限的比特币脚本的功能。这种连续监视实现了‘循环’控制流结构,使得组合的管理器和区块链系统图灵完备。

[0147] 管理器作为图灵机的指令表

[0148] 根据本发明的一个实施例,图灵机包括两个堆栈:

[0149] • 数据堆栈:如上所述,由区块链表示。

[0150] • 控制堆栈:由管理器功能表示。存储关于重复控制流功能的信息。

[0151] 控制堆栈与数据堆栈的分离提供了防止在比特币核心发生无限循环的优点,反过来又减轻了对比特币系统的拒绝服务攻击。

[0152] 管理器管理和运行能够通过任何类型的循环结构(例如FOR-NEXT;REPEAT UNTIL;等)循环的子程序。本文描述的说明性实施例包括使用“重复”结构的一个实例的进程(参见图2)。用户指定索引(z)和限制(J),它们分别表示当前迭代次数(通常从0开始计数)和重复循环的总迭代次数。

[0153] 对于每次迭代:

[0154] 1.索引递增1.对于退出条件,在索引达到限制时迭代停止;

[0155] 2.执行包含“if condition then action”(ICTA)语句的代码块;该操作可以是区块链上或链外的任何操作;

[0156] 3.计算该子程序的加密哈希。这可以作为交易的一部分存储在区块链中。由于哈希对每个代码都是唯一的,因此可以验证已使用的代码。

[0157] 循环体包括一个代码块。每个代码块都包含一个“If condition then action”(ICTA)语句(参见图3),监视区块链的当前状态,以查找与以下内容匹配的交易:

[0158] • 启动或触发条件(例如,当特定比特币地址达到10BTC时)。

[0159] • 重复条件(即与先前迭代相关联的元数据或哈希)。

[0160] • 停止条件(即循环的最后一次迭代)。

[0161] ICTA语句使管理器能够基于区块链的当前状态决定下一个要进行的交易。进行下一个交易涉及将交易广播到比特币网,并将新交易写入区块链。这充当了已执行此迭代的记录。一旦交易写入区块链,管理器随后将发现上一次迭代已执行并写入区块链,并将执行下一次迭代。后者继续,直到索引i)达到代码块中指定的限制(J)时重复循环退出。

[0162] 每个交易都以可重用的方式保存在区块链中。在比特币实施方案中,交易中的每个签名都附加一个SIGHASH标志。该标志可以采用不同的值,每个值表示在没有该签名的所有者参与的情况下是否可以修改交易的其他部分。可重复使用的交易在其中一个交易输入中具有SIGHASH标志‘SigHash_AnyoneCanPay’,这允许任何人为交易的输入做出贡献。此参数使管理器的ICTA功能可以使用不同的输入多次重复执行。该功能的使用可以限于授权方-例如,通过可重复使用交易的版权。

[0163] ICTA代码块的“If条件”部分可以监视任何类型的条件。这与其他编程语言(例如C,C++,Java)类似,并且不限于存储在区块链上的信息。下面列出了一些示例条件:

- [0164] • 监控日期和时间 (即达到特定日期和时间时)。
- [0165] • 监测天气 (即当温度低于10℃且正在下雨时)。
- [0166] • 监控社交媒体 (即我收到朋友发来的消息时)。
- [0167] • 监控合同或信托的条件 (即公司A购买公司B时)。
- [0168] • 监控新闻和事件 (即足球队A赢得比赛时)。
- [0169] • 监控来自物联网的信息 (即灯泡需要更换时)。
- [0170] • 监控来自移动/可穿戴设备的数据 (即,当可穿戴式步进跟踪设备计数达10000步时)。
- [0171] • 监控云计算的结果 (即计算完成并收到结果时)。
- [0172] • 监控远程数据存储 (即文件是否仍然远程存在)。
- [0173] ICTA代码块的“Then action”部分可以执行多个操作。本发明不限于可采取操作的数量或类型。尽管包含与操作相关元数据的交易可以写在区块链上,但是操作不限于区块链上交易。
- [0174] 元数据可以是管理器指定的任何形式。然而,根据本发明的一个实施例,元数据可以将超链接存储到包含更多数据或指令的文件中。元数据既可以将超链接存储到哈希表中,该哈希表包含与操作相关的更多数据或指令,也可以存储到用作哈希表的循环键的操作的哈希中。实施例可以使用类似于比特流磁力链接格式的链接。
- [0175] 以下列出一系列示例操作:
- [0176] • 比特币交易 (即将比特币发送到特定地址)。
- [0177] • 社交媒体 (即向朋友发送消息)。
- [0178] • 交易 (即出售X股)。
- [0179] • 物联网 (即关掉灯泡)。
- [0180] • 商业 (即在线购买商品)。
- [0181] • 在线服务 (即按月支付费用或使用比特币支付所需服务)。
- [0182] 由于本发明不限于所执行操作的性质、类型或数量,因此它提供了一种高度通用的解决方案,该解决方案在广泛的应用中具有很大的优势。
- [0183] 管理器的控制堆栈可以通过多种方式实现,这些方式对每个用户的需求来说是特定的。例如,控制堆栈的重复循环可以基于任何图灵完备语言。一种可能的语言选择是基于堆栈的Forth风格语言。使用这种语言的一个优点在于它可以用广为人知和应用的比特币脚本使控制堆栈与编程风格保持一致。
- [0184] 使用比特币脚本的备用堆栈作为数据存储空间
- [0185] 比特币脚本包含命令,也称为操作码,使用户能够将数据移到备用堆栈,既“alt堆栈”。
- [0186] 操作码是:
- [0187] • OP_TOALTSTACK-将数据从主堆栈的顶部移到alt堆栈的顶部。
- [0188] • OP_FROMALTSTACK-将数据从alt堆栈的顶部移到主堆栈的顶部 (参见图4)。
- [0189] 这使得来自计算中间步骤的数据能够存储在alt堆栈中,类似于允许将数据存储在计算器上的“存储器”功能。根据本发明的一个说明性实施例,alt堆栈用于配置比特币脚本以解决小计算任务并返回计算中的结果。

[0190] 使用代码寄存器管理管理器

[0191] 管理器还管理其拥有和运行的所有代码的注册表。此注册表的结构类似于将特定键映射到特定值的查找表或字典(请参见图5)。密钥和值对由代码块的哈希(Hi)和存储代码的IPv6地址分别表示。要使用密钥Hi检索代码块,查找表则用于检索关联值(存储代码的位置)并相应地检索源代码。

[0192] 代码注册表的实现可以有所不同。例如,可以使用本地管理列表或P2P分布式哈希表来实现查找表。源代码可以本地、远程或使用分散文件存储系统存储。这可以使用磁力链接格式或使用共享零知识加密的任何链接格式来实现。

[0193] 管理器代码的交易元数据,以及循环的重新生成

[0194] 在特定迭代中重新生成管理器循环所需的信息将作为元数据存储区块链记录的交易中(参见图6和图7)。

[0195] 通过这种方式,区块链上的交易存储在管理器上执行循环的给定迭代的信息或提供对该信息的获取。该信息可以包括与循环相关联的任何变量的值,例如索引i,以及任何其他必要信息,例如代码块中使用的参数值或指定可以访问其他所需信息的位置的相关数据。

[0196] 元数据本身作为交易中多签名付费到脚本哈希脚本(P2SH)的一部分存储。有关脚本的格式,请参见图8。通过交易记录的元数据还能够记录过去如何执行代码的审计跟踪。

[0197] 管理器可以通过多种方式在每次迭代中重新生成重复循环代码块。代码块可以硬编码到管理器中,或者可以存储在私人或公共可用文件中,或者存储为私人或公共哈希表文件的条目,或者上述的组合中。代码块可以是具有硬编码变量的静态代码块,也可以是包含可填充参数的静态代码块。参数可以是任何数据格式的单个值,也可以是小块代码,或者是上述的组合。参数可以直接从交易的元数据中(例如比特币交易)或从诸如内部数据库或私人/公共文件或哈希表或上述任何组合的外部源中提取来填充。参数值外部源的指针可以存储在交易的元数据中。

[0198] 以下步骤提供了管理器如何在第i次迭代中重新生成重复循环代码块的一个示例。在此示例中,代码注册表是一个哈希表,其中哈希值充当表的查找键,并存储在交易的元数据中。

[0199] 1. 管理器监视区块链以查找包含与代码注册表中条目相匹配的代码块哈希的交易。

[0200] 2. 管理器查找包含相应哈希(Hi)的交易。

[0201] 3. 管理器读取'Metadata-CodeHash',获取CodeHash字段以获取Hi并用它来检索代码(Ci)。如果RIPEMD-160(SHA256(Ci))等于Hi,则代码未更改,可以安全地继续下一步。

[0202] 4. 管理器读取存储索引的'Metadata-CodeHash',并在第i次迭代中重新生成代码。换句话说,循环在适当的迭代中被“重新加载”。

[0203] 5. 用户签名包含在P2SH命令中以验证元数据的来源。

[0204] 6. 如果循环的这次迭代需要这些数据,则管理器读'Metadata-OutputHash'和'Metadata-OutputPointer'(参见图6)以检索前面步骤的输出。

[0205] 可能需要多个签名来解锁交易(例如,用户、操作系统、软件开发者和软件供应商)。这就让管理权利的数字版权管理(DRM)系统能够操作涉及P2SH交易各方的代码。

[0206] 更新管理器代码

[0207] 与管理器相关代码块的软件更新和补丁经多签名P2SH交易安全授权(参见图8)。多签名交易记录旧代码块和新代码块的元数据,如图5所示。这记录了旧代码到新代码的转换,从而提供了软件更新的审计跟踪。管理器需要存储新旧源代码块的所有哈希。新旧源代码块的哈希可用于验证代码文件的完整性。

[0208] 根据本发明的实施例,需要多个签名来解锁交易(例如用户、操作系统、软件开发者和软件供应商)。这提供了DRM系统,用于管理管理器所用代码的软件更新和补丁。

[0209] 与大多数不允许软件在运行时更新的软件不同,本发明的一个优点是软件更新可以在执行循环的过程中进行。本发明提供了一个动态响应解决方案,该解决方案可实时重新配置并且对本发明控制的过程破坏最小。

[0210] 在区块链上捕获的信息(参见图8和图9)可用于在循环中更新至新代码,并用旧代码前一次迭代的输出元数据开始下一个迭代步骤。

[0211] 投票计数发明

[0212] 当前的比特币脚本语言不允许循环发生。这可以防止使用比特币支付触发连续和自动操作,除非有外部干预;但是,由于管理器会持续监控区块链上信息,允许根据区块链上最新信息执行自动操作。

[0213] 以下阐明如何用管理器的控制堆栈实现在线自动计票机器人方面的进程自动化。

[0214] 本发明中的计票机器人旨在促进公平和伪匿名投票,由区块链记录安全且不可变的投票计数过程审计跟踪。计票机器人通过管理器的控制堆栈和重复循环(参见图10)实现自动化。以下场景说明了这方面的实施例可以如何操作。

[0215] 假设有100名选民。如果在2016年1月1日之前收到57张唯一“Yes”投票,则付款将发放给主席Jason。投票过程分为两部分:

[0216] • 令牌分发

[0217] • 计数

[0218] 对于令牌分发,向每个授权选民分发一个投票令牌,共100个。每个令牌由(例如)一个比特币公钥和私钥对表示,通过秘密交换协议分发给每个选民。密钥交换协议在本领域中是已知的。每个比特币公钥和地址都加载(关联)代表一票的少量比特币。机器人保留与每个授权令牌相关联的公钥列表,并在投票开始之前公开该列表。为了确保投票不能操纵并且投票是匿名的,私钥列表以及投票者身份和其令牌之间的映射被破坏(即从未存储)。

[0219] 拥有匿名和预授权地址列表可提供其他重要好处。这能确保只有获得授权的人才能投有效票;还可以促进排除源自特定地址(例如垃圾邮件发送者,不合格选民)的无用投票而不损害选民身份。为了实现计数过程,管理器运行重复循环。地址列表可保存在比特币脚本中,并被转移到备用堆栈以存储数据。一旦地址被计数,它就会被从备用堆栈中删除,不再被添加到下一个交易中。当地址列表变空时,重复循环停止。

[0220] 不是用整数索引*i*来跟踪循环当前所处的位置,而是投票机器人的管理器用它来存储投票计数的中间值,这样能够确保投票计数的中间值存储在区块链中。这提供了审计跟踪并表明投票计数过程是公平的。

[0221] 如果收到的唯一“Yes”票数达到57,则约定的比特币金额将支付至Jason的账户。

投票计数脚本的加密哈希以及存储此脚本的IPv6地址将向公众发布。这就意味着公众有足够的信息来进行重新计票并确保计票过程公平、正确。

[0222] 需要注意的是,上述实施例说明并非限制本发明,并且本领域技术人员在不脱离附加权利要求范围的前提下,将能设计许多替换实施例。在权利要求中,括号内的任何参考标记不应解释为限制权利要求。动词“包括(comprising)和(comprises)”等不排除权利要求或说明书中作为整体列出的那些之外的元件和步骤存在。在本说明书中,“包括(comprises)”意指“(includes或者consists of)包括或由……组成”,其词形变化“(comprising)”意指“(including or consisting of)包括或由……组成”。元件的单数引用不排除这些元件的复数引用,反之亦然。本发明可以通过包括若干不同元件的硬件以及通过适当编程的计算机来实现。在列举了若干装置的设备权利要求中,这些装置中的若干个可以通过同一个硬件项体现。在互相不同的从属权利要求中叙述某些步骤的仅有事实不表明这些步骤的组合不能用于获益。

每个单元格代表一个块或一个交易
字母A到G代表元数据

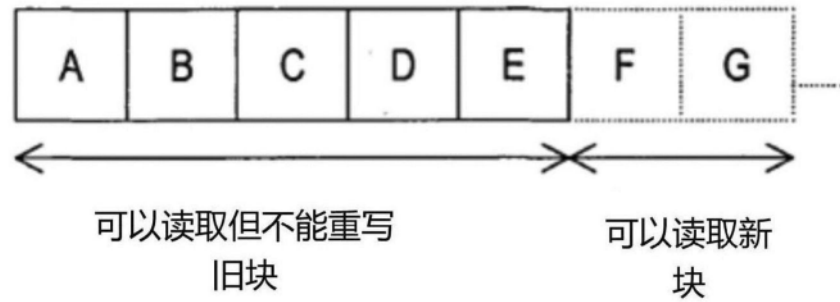


图1

```
i = 0  
repeat {  
  # If Condition Then Action  
    i = i + 1;  
} until (i = J)
```

获取上述代码的加密哈希 (H_1)

图2

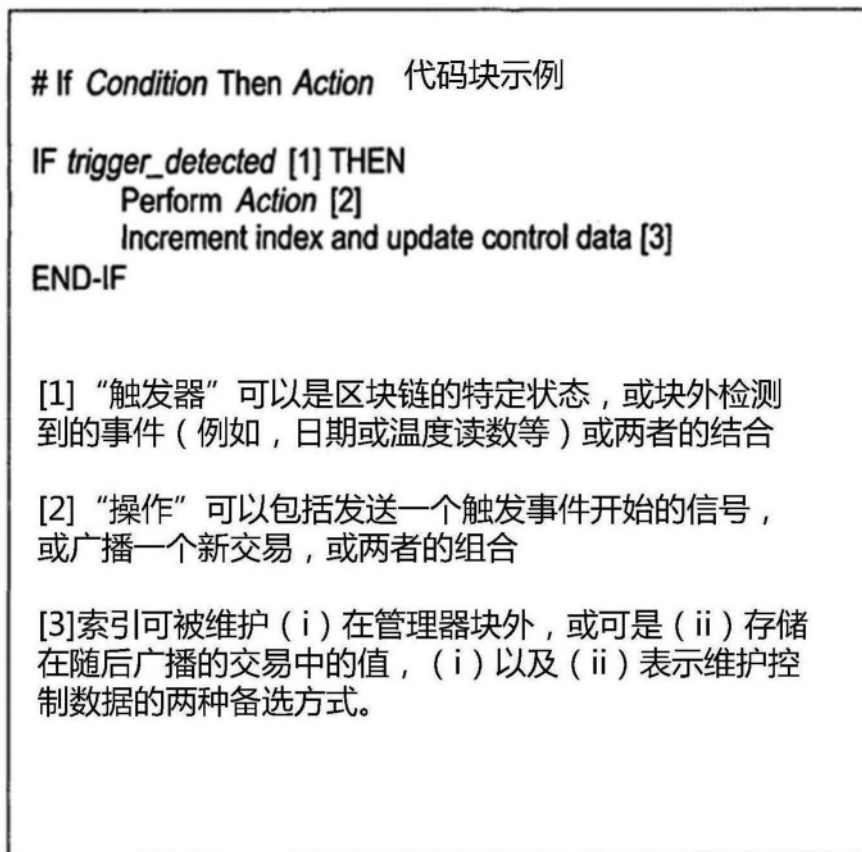


图3

字	操作码	十六进制	输入	输出	描述
OP_TOALSTACK	107	0x6b	x1	(alt)x1	将输入放到 alt 堆栈的顶部。将其从主堆栈中删除
OP-FROMALSTACK	108	0x6c	(alt)x1	x1	将输入放在主堆栈的顶部。将其从 alt 堆栈中删除。

图4

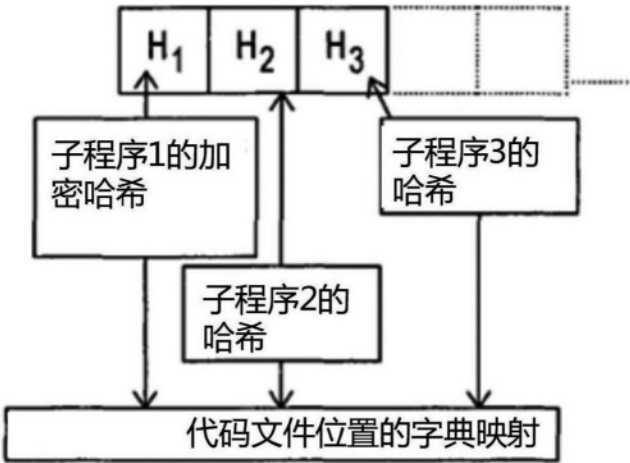


图5

字段	子字段	字节	值	描述
元数据-代码 哈希	代码哈希	20		RIPEND-160(SHA256(代码指针寻址的实际代码文件
	索引	12		整数索引/指示循环所在迭代
元数据-代码 指针	代码类型	4	0x0000001	表示它是执行循环的管理器代码
	代码指针	16		实际代码文件位置的 IPv6 地址
	填充	12	0x000000...	备用字节

图6

字段	子字段	字节	值	描述
Metadata-OutputHash 元数据-输出哈希	输出哈希	20		RIOEMD-160(SHA56(输出指针寻址的输出元数据文件))
	填充	12	0x00000...	备用字节
Metadata-OutputPointer 元数据-输出指针	输出指针	16		代码输出相关元数据的 IPv6 地址
	填充	16	0x00000...	备用字节

图7

赎回脚本： 4 Metadata-OldCodeHash Metadata-NewCodeHash Metadata- OldCodePointer Metadata-NewCodePointer PK-User PK-OS PK-Developer PK-Vendor 8 OP-CHECKMULTSIG	图例：
锁定脚本： OP_HASH160 < 20-byte hash of redeem script > OP_EQUAL	哈希：代码块的哈希 元数据：与交易相关的元数据 OS：操作系统 PK：公钥 Sig:签名
解锁脚本： Sig-User Sig-OS Sig- Developer Sig-Vendor Redeem Script	

图8

赎回脚本： 4 Metadata-OldCodeHash Metadata-NewCodeHash Metadata- OldCodePointer Metadata-NewCodePointer PK-User PK-OS PK-Developer PK-Vendor 8 OP-CHECKMULTSIG	图例：
锁定脚本： OP_HASH160 < 20-byte hash of redeem script > OP_EQUAL	哈希：代码块的哈希 元数据：与交易相关的元数据 OS：操作系统 PK：公钥 Sig:签名
解锁脚本： Sig-User Sig-OS Sig- Developer Sig-Vendor Redeem Script	

图9

#投票计数机器人示例重复循环

#投票条件

假设我们有100人，如果截止到2016年1月1日前我们有57张 “Yes” 投票，则我们将向Jason发放付款。

#第1部分:运行发送投票令牌机器人

i. 生成100个比特币公钥并用少量比特币填充每个公钥

ii. 每个地址代表一个必须用于投票的授权令牌

iii.安全地将令牌和相应私钥发送给每个有权投票的人

iv. 机器人保留与每个授权令牌相关联的公钥列表，并在投票前公开该列表

v. 为确保投票不能被操纵，销毁私钥列表

vi. 为确保投票是匿名的，销毁任何与选民身份相关的令牌关联使得他们匿名

#第二部分：运行投票计数机器人

```
Array_of_tokens = [list of 100 authorized tokens];
Yes_count = 0;
repeat {
    this_token = pop from Array_of_tokens;
    if ( signature received from this_token and
        current date < 1st January 2016 )
        Increment Yes_count;
    }
} until (Array_of_tokens is empty)

if ( Yes_count >= 57 ) {
    Pay to Jason's Account.
}
```

图10