(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 112438035 B
(45)授权公告日 2024.06.28

(21)申请号 201980048402.X
(22)申请日 2019.07.15
(65)同一申请的已公布的文献号
申请公布号 CN 112438035 A
(43)申请公布日 2021.03.02
(30)优先权数据
1811773.9 2018.07.19 GB
(85)PCT国际申请进入国家阶段日
2021.01.19
(86)PCT国际申请的申请数据
PCT/IB2019/056014 2019.07.15
(87)PCT国际申请的公布数据
WO2020/016739 EN 2020.01.23

(73)专利权人 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 丹尼尔·约瑟夫
(74)专利代理机构 北京市竞天公诚律师事务所 11770
专利代理师 陈果

(51)Int.Cl.
H04L 9/32(2006.01)
H04L 9/08(2006.01)
(56)对比文件
CN 106503994 A,2017.03.15
CN 106850611 A,2017.06.13
审查员 刘星星

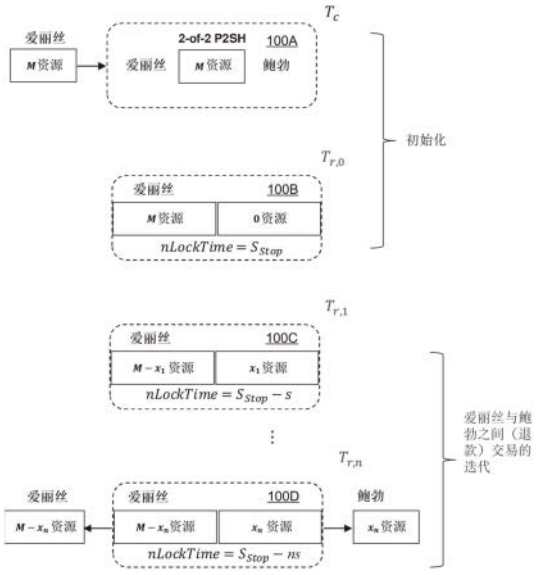
权利要求书2页 说明书21页 附图20页

(54)发明名称

用于控制分布式系统的处理步骤的计算机实现的系统和方法

(57)摘要

一种控制和协调分布式系统中的处理步骤的方法,该方法可以通过参与区块链网络的节点组成的循环有序集的发起者节点来实现。该方法包括:为该循环有序集的节点生成私钥及其加密份额,并对其进行分配。基于所述份额来确定锁定值,并且设置交易来传输对资源的控制,以响应于提供相应解锁值。一种在节点之间的交易环路,每个节点被设置成发送对资源的控制,以响应于提供锁定值对应的解锁值,其中所述锁定值是基于分配给两个相邻节点之一的第一节点的份额和从紧挨在其之前的另一节点接收到的值而确定的。发起者节点可以属于发起者节点的循环有序集。



CN 112438035 B

1.一种计算机实现的方法,包括:

由参与区块链网络的节点组成的循环有序的第一集合的第一发起者节点,生成私钥;

由所述第一发起者节点,为所述第一集合的每个节点生成私钥的加密份额,并将各自的加密份额分配给所述第一集合的其他节点;

由所述第一发起者节点,基于每个加密份额对应的公钥确定第一锁定值;

由所述第一发起者节点,准备交易,其中所述交易被设置成从与所述第一发起者节点相关联的源地址向在所述第一集合中紧挨在所述第一发起者节点之后的节点的接收地址发送对资源的控制,以响应于满足包括提供与所述第一锁定值对应的第一解锁值的执行条件;以及

由所述第一发起者节点,发起进一步交易的准备,以在所述第一集合中的每对相邻节点之间形成交易环路,其中每个进一步交易被设置成从与每对相邻节点中的第一节点相关联的地址向与该对相邻节点中的第二节点相关联的地址发送相应资源的控制,以响应于满足包括提供对应于相应锁定值的相应解锁值的执行条件,其中每对相邻节点中的所述第二节点在所述第一集合中紧挨在该对相邻节点中的所述第一节点之后,并且基于分配给给定对相邻节点中的第一节点的加密份额和从在所述第一集合中紧挨所述第一节点之前的节点接收到的值来确定相应的锁定值;

其中,所述资源不包括虚拟货币。

2.根据权利要求1所述的方法,还包括:

由所述第一发起者节点向发起者节点组成的循环有序集的主发起者节点,发送与所述私钥对应的公钥,其中发起者节点集的每个发起者节点是多个循环有序节点集中的相应一个节点集的发起者节点,所述多个循环有序节点集包括所述第一集合;以及

由所述第一发起者节点从第二发起者节点,接收第一值,所述第一值基于与包括从所述第一发起者节点到所述主发起者节点组成的集合中每个节点相关联的公钥,其中所述第二发起者节点在所述发起者节点集中紧挨在所述第一发起者节点之前。

3.根据权利要求2所述的方法,还包括:

由所述第一发起者节点,基于所述第一值和所述私钥对应的所述公钥确定第二锁定值;以及

由所述第一发起者节点,准备交易,其中所述交易被设置成从与所述第一发起者节点相关联的源地址向第三发起者节点的接收地址发送资源的控制,以响应于满足包括提供对应于所述第二锁定值的第二解锁值的执行条件,其中所述第三发起者节点在所述发起者节点集中紧挨在所述第一发起者节点之后。

4.根据权利要求3所述的方法,还包括:

由所述第一发起者节点,基于与从所述第三发起者节点到所述主发起者节点组成的发起者节点集的每个节点相关联的公钥对应的私钥来获得第二值。

5.根据权利要求4所述的方法,其中所述第二值由所述第一发起者节点从区块链获得。

6.根据权利要求4所述的方法,其中所述第二值由所述第一发起者节点从所述第三发起者节点获得。

7.根据权利要求4所述的方法,还包括:

由所述第一发起者节点,确定不应发起所述交易环路的执行;以及

响应于确定不应发起所述交易环路的执行：

由所述第一发起者节点，基于所述第二值和所述私钥确定第三解锁值；以及

由所述第一发起者节点使用所述第三解锁值，执行由所述第二发起者节点准备的交易，其中所述交易被设置成将资源的控制从与所述第二发起者节点相关联的源地址发送到所述第一发起者节点的接收地址，其中，响应于满足包括提供所述第三解锁值的执行条件来发送所述资源的控制。

8.根据权利要求4所述的方法，其中所述第一锁定值进一步基于所述第二值。

9.根据权利要求4所述的方法，还包括：

由所述第一发起者节点确定应发起所述交易环路的执行；

响应于确定应发起所述交易环路的执行：

由所述第一发起者节点，基于所述第二值和所述私钥的相应一个加密份额来确定第四解锁值；以及

由所述第一发起者节点使用所述第四解锁值，执行由在循环有序的所述第一集合中紧挨在所述第一发起者节点之前的节点准备的交易，其中所述交易被设置成将对资源的控制从与所述紧挨在所述第一发起者节点之前的节点相关联的源地址发送到所述第一发起者节点的接收地址，其中，响应于满足包括提供所述第四解锁值的执行条件来发送对所述资源的控制。

10.根据权利要求1所述的方法，其中所述的发起进一步交易的准备，以在所述第一集合中的每对相邻节点之间形成交易环路的步骤包括：由所述第一发起者节点，将所述第一锁定值发送到在所述第一集合中紧挨在所述第一发起者节点之后的所述节点。

11.根据权利要求1所述的方法，其中所述资源中的每一个资源与所述资源中的其他资源具有相同的量。

12.根据权利要求1所述的方法，其中每个公钥及其对应的私钥形成椭圆曲线加密公钥-私钥对。

13.根据权利要求1所述的方法，其中使用可公开验证的密码秘密共享算法来生成所述私钥的加密份额。

14.一种计算机实现的系统，包括：处理器和存储器，所述存储器包括可执行指令，所述指令被所述处理器执行时，使所述系统执行根据权利要求1至13中任一项所述的方法。

15.一种非暂时性计算机可读存储介质，其中所述介质用于存储指令，所述指令用于使计算机系统适于执行根据权利要求1至13中任一项所述的方法。

用于控制分布式系统的处理步骤的计算机实现的系统和方法

技术领域

[0001] 本公开大体涉及分布式计算系统,更具体地,涉及与分布式系统的节点相关联的处理步骤的控制和协调,包括使用区块链网络来指导和控制有序处理步骤。

背景技术

[0002] 在分布式计算系统中,充当分布式系统中的节点的各种计算设备可以通过网络进行通信。可以在这些节点之间进行消息交换。这类消息交换可以例如允许节点协作以执行计算任务。这类任务可能涉及分布在各个节点上的处理。这类分布式处理可能需要控制和协调与各个节点相关联的步骤。例如,可以执行特定的处理顺序。

[0003] 分布式系统的一个示例可能出现在飞机信息系统的背景下。例如,飞机信息系统可以包括各种子系统,其中每个子系统由子组件组成。每个子系统都可以负责执行特定的处理,例如执行飞行前检查。为了执行那些步骤,每个子系统可以要求子组件执行特定的处理步骤。

发明内容

[0004] 在这类分布式系统中,可能需要或期望能够执行处理中的步骤顺序和/或形成特定处理的步骤顺序。例如,子系统可能需要按顺序执行其各自的处理。另外,与子组件相关联的节点可能需要按顺序执行其各自的处理步骤。

[0005] 另外,可能期望提供一种防篡改的审计跟踪,用于记录处理步骤或特定处理的输出。这类审计跟踪可以在各种场景中具有应用。例如,以飞机为例,这可以例如允许诸如在事故发生之后对事故进行法证调查。在一个特定的例子中,这类系统可以允许将故障归因于飞机的特定子系统,例如,以确定根本原因。

[0006] 在本文中,术语“区块链”涵盖所有形式的基于计算机的电子分布式分类账。这些分类账包括基于共识的区块链和交易链技术、许可和非许可的分类账、共享分类账,及其变体。应注意,替代的区块链实施方案和协议也落入本公开范围内。

[0007] 区块链是一种点对点的电子分类账,其实现为基于计算机的去中心化的分布式系统,其中所述系统由区块组成,而区块又由交易(transaction)组成。每个交易都是一种数据结构,所述数据结构对所述区块链系统参与者之间的数字资产控制权的转移进行编码,并且包括至少一个输入和至少一个输出。每个区块都包含前一个区块的哈希值,因此区块被链接在一起,以创建自所述区块链创建以来写入其中的所有交易的永久性的不可更改的记录。交易包括嵌入到其输入和输出中的小程序,称为脚本,这些脚本指定如何以及由谁访问所述交易的输出。这些脚本是使用基于堆栈的脚本语言编写的。

[0008] 为了将交易写入所述区块链,必须对其进行“验证”。网络节点进行工作以确保每个交易均有效,而无效交易则被网络拒绝。安装在所述节点上的软件客户端通过执行其锁定和解锁脚本对未花费的交易输出执行此验证工作。如果所述锁定和解锁脚本的执行评估为真,则所述交易有效,将所述交易写入所述区块链。因此,为了将交易写入所述区块链,所

述交易必须:i)由接收所述交易的第一个节点进行验证—如果所述交易通过验证,则此节点将其中继到网络中的其他节点;ii)添加到新区块中;iii)已挖掘,即被添加到历史交易的公共分类账中。

[0009] 数字企业家已经开始探索如何利用加密安全系统和可以存储在区块链上的数据来实现新的系统。如果区块链可以用于自动任务和过程,则会非常有利。这类解决方案将能够发挥区块链的优势(例如,永久及防篡改的事件记录、分布式处理等),同时其应用将更加广泛。

[0010] 2018年4月20日提交的名称为“计算机实现的方法和系统”的共同拥有的第1806448.5号英国专利申请中公开了一种用于控制和协调分布式系统中的处理步骤的方法、系统和计算机可读存储介质,所述分布式系统可以由参与区块链网络的节点组成的循环有序集的节点来实现。该申请的主题包括协议的描述,该协议可以称为多因素相关决策(MDD)协议。该申请的主题可用于提供分布式系统的节点之间的步骤的排序。例如,它可以用于提供系统内的处理步骤的排序。

[0011] 在2017年4月18日和2018年2月8日提交的名称为“计算机实现的方法和系统”的共同拥有的第1706071.6和1802063.6号英国专利申请中分别公开了用于在由参与区块链网络的节点组成的循环有序集形成的分布式系统中提供功能的其他方法、系统和计算机可读存储介质。前者包括与名为群随机交换(Group Random Exchange,GRE)的协议有关的公开内容,该协议可允许资金从一个地址转移到另一个地址,从而模糊实体关联的地址之间的关联,同时消除了该实体资源被挪用的可能性。总的来说,GRE建立在一个群的概念上,该群同意一群节点的每个成员将x个计算资源的单元转移到另一个成员,以便每个成员都可以接收x个计算资源的单元。它还包括与GRE协议的变体(即群随机交换更改秘密(Group Random Exchange Change Secret,GRECS))有关的公开内容,GRECS更改GRE协议以便在为每条支付通道使用不同的秘密值时提供相同的功能。特别地,参与GRECS协议的每个节点都将各自的秘密值发送到发起者节点,然后由发起者节点使用这些值来发起转移。后者包括涉及对GRE协议的改进的公开内容,更具体地,涉及GRECS协议的公开内容。这类改进的协议,即群随机交换延迟显示(Group Random Exchange Late Reveal,GRELR),通过避免使用公共秘密的方式来改进GRECS。

[0012] 然而,值得注意的是,如上所述,可能期望不仅能够控制系统的组件之间(例如,子系统之间)的处理顺序,而且还期望能够控制组成特定子系统的子组件之间的处理顺序。另外,可能期望能够允许由特定子系统进行的处理可选地被跳过或省略。

[0013] 现在已设计出这类改进的解决方案。

[0014] 根据本公开,可以提供一种计算机实现的方法。该方法可以包括:由参与区块链网络的节点组成的循环有序第一集合的第一发起者节点生成私钥。该方法可以包括:第一发起者节点为该第一集合的每个节点生成私钥的加密份额,并且将各自的加密份额分配给第一集合的其他节点。该方法可以包括:第一发起者节点基于与每个加密份额对应的公钥来确定第一锁定值。该方法可以包括:第一发起者节点准备交易,该交易被设置成从与第一发起者节点相关联的源地址向在第一集合中紧挨在第一发起者节点之后的节点的接收地址发送对资源的控制,以响应于满足包括提供对应于第一锁定值的第一解锁值的执行条件。该方法可以包括:第一发起者节点发起进一步交易的准备,以在第一集合中的每对相邻节

点之间形成交易环路,每个进一步交易被设置成从与每对相邻节点中的第一节点相关联的地址向与该对相邻节点中的第二节点相关联的地址发送相应资源的控制,以响应于满足包括提供对应于相应锁定值的相应解锁值的执行条件。每对相邻节点中的第二节点在第一集合中可以紧挨在该对相邻节点中的第一节点之后。可以基于分配给给定对相邻节点中的第一节点的加密份额和从在第一集合中紧挨在该第一节点之前的节点接收到的值来确定相应的锁定值。

[0015] 在一些实现方式中,该方法可以包括:第一发起者节点向发起者节点组成的循环有序集的主发起者节点发送对应于私钥的公钥。发起者节点集的每个发起者节点可以是多个循环有序节点集中的相应一个节点集的发起者节点。多个循环有序节点集可以包括第一集合。该方法可以包括:第一发起者节点从第二发起者节点接收基于包括从第一发起者节点到主发起者节点集合中与每个节点相关联的公钥的第一值。第二发起者节点在发起者节点集中可以紧挨在第一发起者节点之前。

[0016] 在一些实现方式中,该方法可以包括:第一发起者节点基于第一值和与私钥对应的公钥来确定第二锁定值。该方法可以包括:第一发起者节点准备交易,该交易被设置成从与第一发起者节点相关联的源地址向第三发起者节点的接收地址发送对资源的控制,以响应于满足包括提供对应于第二锁定值的第二解锁值的执行条件。第三发起者节点在发起者节点集中可以紧挨在第一发起者节点之后。

[0017] 在一些实现方式中,该方法可以包括:第一发起者节点基于与从第三发起者节点到主发起者节点组成的发起者节点集的每个节点相关联的公钥对应的私钥来获得第二值。

[0018] 在一些实现方式中,第二值可以由第一发起者节点从区块链获得。

[0019] 在一些实现方式中,第二值可以由第一发起者节点从第三发起者节点获得。

[0020] 在一些实现方式中,该方法可以包括:第一发起者节点确定不应发起交易环路的执行。该方法可以包括:响应于确定不应发起交易环路的执行:第一发起者节点基于第二值和私钥确定第三解锁值;以及第一发起者节点使用第三解锁值执行由第二发起者节点准备的交易,其中所述交易被设置为将资源的控制从与第二发起者节点相关联的源地址发送到第一发起者节点的接收地址。可以发送对资源的控制,以响应于满足包括提供第三解锁值的执行条件。

[0021] 在一些实现方式中,第一锁定值可以进一步基于第二值。

[0022] 在一些实现方式中,该方法可以包括:第一发起者节点确定应发起交易环路的执行,并且响应于确定应发起交易环路的执行:第一发起者节点基于第二值和私钥的相应一个加密份额确定第四解锁值;以及第一发起者节点使用第四解锁值执行由循环有序的第一集合中紧挨在发起者节点之前的节点准备的交易,其中该交易被设置成将资源的控制从与紧挨在前的节点相关联的源地址发送到发起者节点的接收地址。可以发送对资源的控制,以响应于满足包括提供第四解锁值的执行条件。

[0023] 在一些实现方式中,发起准备进一步交易以形成第一集合中的每对相邻节点之间的交易环路可以包括:第一发起者节点向在第一集合中紧挨在第一发起者节点之后的节点发送第一锁定值。

[0024] 在一些实现方式中,每个资源可以与其他资源具有相同的量。

[0025] 在一些实现方式中,每个公钥及其对应的私钥可以形成椭圆曲线加密公钥-私钥

对。

[0026] 在一些实现方式中,可以使用可公开验证的密码秘密共享算法来生成私钥的加密份额。

[0027] 本公开还提供了一种系统,包括:处理器和存储器。存储器包括可执行指令,所述指令被处理器执行时,使系统执行本文描述的计算机实现的方法的任何实施例。

[0028] 本公开还提供一种其上存储有可执行指令的非暂时性计算机可读存储介质,其中所述指令被计算机系统的处理器执行时,使得计算机系统至少执行本文所述的计算机实现的方法的实施例。

[0029] 本文的资源不包括虚拟货币。

附图说明

[0030] 本公开的这些方面和其他方面将从本文所述的实施例中变得显而易见,并参考本文所述的实施例进行阐述。现将仅通过举例的方式并参考附图对本公开的实施例进行说明,其中:

[0031] 图1是示出在支付通道中使用的交易的图;

[0032] 图2是示出如何创建图1中的支付通道的流程图;

[0033] 图3是示出本申请的示例性操作环境的简化示意图;

[0034] 图4是示出本申请的另一示例操作环境的简化示意图;

[0035] 图5示出了示例性计算设备;

[0036] 图6是示出由节点构成的循环有序集的图;

[0037] 图7是一个简化示意图,示出了一个二维结构,其可以概念性地对应于多个循环有序集;

[0038] 图8是提供类似于图7所示的二维结构的更详细表示的图;

[0039] 图9是示出图8的循环有序集的一个典型集的图;

[0040] 图10是示出由图8的发起者节点形成的循环有序集的图;

[0041] 图11是示出用于构造在发起者节点之间形成环路的支付通道的操作的流程图;

[0042] 图12是示出用于构造在给定的循环有序节点集的节点之间形成环路的支付通道的操作的流程图;

[0043] 图13提供了图8的示意图的版本,其被注释以示出与节点之间的支付通道相关联的解锁值;

[0044] 图14是示出图11和图12的流程图中的操作的组合的流程图;

[0045] 图15提供支付通道的调拨通道版本的表示;

[0046] 图16示出了可以用作支付通道的调拨交易的示例性交易;

[0047] 图17示出了可以用作支付通道的退款交易的示例性交易;

[0048] 图18示出了可用作支付通道的支付交易的示例性交易;

[0049] 图19示出了可以如何将图19的调拨交易的各方面与图18的支付交易的各方面组合;以及

[0050] 图20是示出在控制和执行有序处理中由各种循环有序集的节点执行的操作的流程图。

[0051] 在附图中使用相同的附图标记来表示相同的元件和特征。

具体实施方式

[0052] 本文描述了体现本公开的协议,用于一群协作节点,以便每个协作节点以有序方式进行决策。它是基于以下概念而建立的:通过要求节点调拨(commit)计算资源单元,该计算资源单元只能通过利用从处理任务中先前步骤的输出中得出的累积秘密值来恢复。

[0053] 现在将通过背景技术来描述几个概念。

[0054] 本文描述的协议基于现有的支付通道,特别是结合了退款交易的用法。

[0055] 本文的退款交易、支付交易等交易均不涉及虚拟货币交易。

[0056] 支付通道

[0057] 如上所述,在下文的描述中涉及支付通道,因此为了方便读者,下面概述支付通道。

[0058] 支付通道是为各方设计的技术,用于进行多次资源交易,而不将所有交易调拨(commit)到区块链。在典型的支付通道实现方式中,可以进行几乎无限量的支付,但是仅需要向区块链添加两个交易。

[0059] 除了减少了增加到区块链的交易的数量和降低相关的成本之外,支付通道还提供了速度的优点,并且重要的是,如果事情不是按计划进行的,或者任何一个参与者决定不进行超过某组支付,则各方具有将单元退还的能力。下面概述支付通道的实现。

[0060] 考虑Alice(爱丽丝)需要将区块链资源转移到Bob(鲍勃)的情况。根据情况需要,这可能需要在一段时间内从Alice向Bob进行多次支付。为方便起见,在Alice和Bob之间建立支付通道,并如下操作。

[0061] 首先,Alice创建2-2(2-of-2)的多重签名支付到脚本哈希(P2SH)交易T_c,由Alice和Bob共同管理。此时,交易不被提交到网络。

[0062] 接下来,Alice创建单独的退款交易$T_{r,0}$,将多重签名控制的资金的所有单元返回给Alice。该交易包括100个块的nLockTime值。Bob对交易进行签名。如果Alice和Bob之间的交换出现问题,在nLockTime已经到期之后,则此退款交易允许Alice获得退款。

[0063] 接下来,Alice对原始交易T_c进行签名。

[0064] 此时,Alice和Bob可以继续创建新的退款交易,以反映从Alice到Bob进行的(区块链外)转移。这些退款交易将反映Alice在该时间点需要转移给Bob的资源的净数量。对于每个新的退款交易,假设他们都同意详细信息,则双方均对交易进行签名,但不必将交易提交到网络。

[0065] 应当注意,创建的每个连续退款交易的nLockTime均比前一个退款交易的nLockTime短,即$nLockTime(T_{r,i+1}) < nLockTime(T_{r,i})$。

[0066] 如果参与者拒绝对任何$T_{r,i}$进行签名,则受欺骗的参与者可以只提交$T_{r,i-1}$。在最坏情况下,Alice对$T_{r,0}$进行签名,并将其提交给收回所有她的单元的网络(在nLockTime到期之后)。

[0067] 所构造的最终退款交易表示从Alice转移给Bob的单元的净额。此交易被提交到网络。

[0068] 图1示出了在支付通道中使用的交易T_c 100A和$T_{r,n}$ 100D。M表示可以从Alice发送

给Bob的最大金额。x_i是Alice需要支付给Bob的单元的当前净额。S_{stop}是初始退款交易的nLockTime,n是在Alice和Bob之间正在进行的(区块外)转移中创建的退款交易的数量(不包括初始退款交易),s是指在一方冒着另一方提交先前的退款交易的风险之前,分配给双方参与者(Alice和Bob)同意退款交易的时间,从而有效地终止Alice和Bob之间的交易。

[0069] 应当注意:$t+n*s<S_{stop}$,其中t是当前时间,并且$(S_{stop}-n*s)\geq s$。

[0070] 图1中的交易T_c 100A、$T_{r,0}$ 100B、$T_{r,1}$ 100C和$T_{r,n}$ 100D是可能出现在区块链上的交易。

[0071] 在图2的流程图200中示出了用于在Alice和Bob之间构造支付通道的操作。操作210及后续操作由一个或多个计算设备中的一个或多个处理器执行,该一个或多个计算设备执行包括可以存储在计算机可读存储介质上的计算机可执行指令的软件。

[0072] 在操作210,与Alice相关联的计算设备的处理器以上述方式创建2-2多重签名支付到脚本哈希(P2SH)交易T_c。

[0073] 控制流程从操作210进行到操作212。在操作212,与Alice相关联的计算设备的处理器创建单独的退款交易$T_{r,0}$,以上述方式将多重签名控制的资金的所有单元返回到与Alice相关联的账户。

[0074] 控制流程从操作212进行到操作214。在操作214,与Alice相关联的计算设备的处理器对上述退款交易进行签名。

[0075] 控制流程从操作214进行到操作216。在操作214,与Bob相关联的计算设备的处理器也可以对上述退款交易进行签名。如果交易被如此签名,则控制流程进行到操作218。或者,如果交易未被如此签名,则中止支付通道的创建。

[0076] 在操作218,与Alice相关联的计算设备的处理器对T_c进行签名并将其提交到区块链。然后,控制流程进行到操作220。

[0077] 在操作220,将上述退款交易识别为第一退款交易,以便稍后可以协商从Alice到Bob的进一步转移。

[0078] 控制流程从操作220进行到操作222。在操作222,确定有充足的剩余时间来协商进一步的转移。如果剩余时间不足,则控制流程进行到操作224,在该操作中,最后退款交易被提交到区块链。或者,如果剩余时间充足,则控制流程进行到操作226。

[0079] 在操作226,协商Alice和Bob之间的进一步转移。控制流程从操作226进行到操作228,在该操作中,确定协商是否成功。如果该协商不成功,则控制进行到上述操作224。或者,成功的协商将导致控制流程进行到操作230。

[0080] 在操作230,以上述方式创建反映源于成功协商的协议的新退款交易。然后,控制流程进行到操作240,在该操作中,新退款交易被识别为当前退款交易。

[0081] 控制流程从操作240进行到操作242,在该操作中,与Alice和Bob相关联的计算设备的处理器可以对当前退款交易进行签名。如果这样,则控制流程返回到上述操作222。或者,如果交易没有被如此签名,则在操作244处,当前退款交易的识别返回到先前退款交易。控制流程从操作244返回到上述操作222。

[0082] 秘密共享

[0083] 如下文进一步描述的,在本申请的主题中采用阈值秘密共享方案。

[0084] 在一些示例中,阈值秘密共享协议可以由(t;n)阈值定义,其中n可以是参与节点

的数量,并且t+1可以是重建秘密所需的最小节点数量。秘密共享方案可以是阈值密码系统的示例,其中秘密可以被划分为n个节点中的部分,从而要求至少t+1个节点参与以重建秘密。对任何t个部分的了解可能不会使秘密泄露。

[0085] Shamir(沙米尔)秘密共享

[0086] Shamir,A.1979年在《ACM通讯》22(11)第612-613页(“Shamir方法”)发表的题为“如何共享秘密”一文对示例性阈值秘密共享解决方案进行了描述。Shamir方法可以基于多项式插值,并且在不失一般性的前提下,将秘密假定为有限域F的元素。Shamir方法可以包括交易商(dealer)方法或无交易商(dealerless)方法,并且可以包括一组n个节点$U_1,\cdots,U_n$和结构化的访问。参与者群可能能够重建秘密。利用Shamir方法,将任意随机秘密作为f(0)存储在t级多项式f(x)中,并且只有参与者i可以计算其份额$f(x_i)$。如果n个参与者中的t+1个参与者协作,则他们可以使用(密钥k的)份额$k_1,k_2,\cdots,k_n$(对应于使用拉格朗日(Lagrange)多项式插值法的$f(x_1),f(x_2),\cdots,f(x_n)$)来重建f(x)上的任意点。使用拉格朗日多项式插值法,可以用t+1点来重建t级函数f(x)。

[0087] $p=\{(x_1,f(x_1)),(x_2,f(x_2)),\cdots,(x_{t+1},f(x_{t+1}))\}$

[0088] $$f(x)=\sum_{i\in p}f(x_i)\prod_{j\in p,j\neq i}\frac{x-x_j}{x_i-x_j}=\sum_{i\in p}f(x_i)b_{i,p}(x)$$

[0089] 其中:

[0090] $$b_{i,p}(x)=\prod_{j\in p,j\neq i}\frac{x-x_j}{x_i-x_j}$$

[0091] 注意:$b_{i,p}(x_i)=1$,且$b_{i,p}(x_j)=0$。

[0092] 在存在交易商节点的情况下,交易商节点可以选择被假定为大小为p(p为素数)的有限域F的元素的秘密$a_o=k$,并且可以随机地选取t-1个正整数$a_1,\cdots,a_{t-1}$,这些正整数代表多项式$f(x)=a_o+a_1x+a_2x^2+\ldots$的系数。然后,交易商节点可以计算属于多项式的n个点$(x_i,f(x_i))$,并将他们分配给参与者。

[0093] 在Shamir无交易商节点的份额分配阶段:

[0094] 1.每个参与者U_i被分配x_i,其中x_i被每个参与者所知。每个x_i都必须是唯一的。

[0095] 2.每个参与者U_i生成具有t级的随机多项式$f_i(x)$。

[0096] 3.每个参与者U_i秘密地(用接收者的公钥加密)向其他参与者发送其在多项式$f_i(x_j)\bmod n$上的相应点。

[0097] 4.每个参与者U_i将其所有接收到的$f_1(x_i),f_2(x_i),\cdots f_p(x_i)$(所有mod n,其中n是由椭圆曲线上的点G生成的群的顺序)求和以形成$k_i=f(x_i)\bmod n$,这是多项式$f(x)\bmod n$上的份额。

[0098] 上述对Shamir方法的描述是阈值秘密共享解决方案的一个示例;然而,也可以考虑其他方法或阈值秘密共享解决方案。

[0099] 在一些实现方式中,阈值签名计算可以基于k×G的确定,其中k是密钥,并且G是椭圆曲线上的点。

[0100] 如果f(x)是t级多项式,则秘密k可以通过$k=\Sigma_{i\in\pi}b_{i,\pi}k_i$来插值,其中π可以是份额$k_a,k_b,\cdots,k_t,k_{t+1}$的尺寸t+1子集,并且b可以是插值因子,π可以是协作计算k×G而不公开其

份额k_i的t+1节点群。k可以是t级多项式上的x＝0点

[0101] -每个节点U_i可以计算一部分$b_{i,\pi}k_i \times G$。

[0102] -π中的所有节点可以将他们的部分加在一起(通过拉格朗日插值法重建秘密k)，得到：

[0103] $b_{a,\pi}k_a \times G+b_{b,\pi}k_b \times G+\cdots+b_{t+1,\pi}k_{t+1} \times G＝k \times G$

[0104] 上述计算Q＝kG的过程可以被称为秘密共享加入。

[0105] 可公开验证的秘密共享

[0106] 本申请的主题中涉及的节点可能期望能够验证其秘密份额的正确性。验证密钥份额时，可以采用可公开验证的秘密共享(PVSS)协议或算法。密钥份额验证可以允许节点验证他们自己的份额以及其他节点的份额。

[0107] Stadler,M在1996年5月出版的《密码技术理论和应用国际会议(International Conference on the Theory and Applications of Cryptographic Techniques)》(由总部设在柏林和海德堡的施普林格出版社出版)中的“可公开验证的秘密共享”一文(第190至199页)对示例性PVSS协议进行了描述。

[0108] 在示例PVSS协议中，每个节点U_i可以具有解密函数D_i，该解密函数能够访问用相应的公共加密函数E_i加密的信息。如此一来，交易商节点可以使用公共加密函数来分配密钥份额并以以下形式进行发布：

[0109] $K_i＝E_i(k_i)$，i＝1，…，n。

[0110] 加密的份额可以由感兴趣的节点进行公开验证。节点可以验证其密钥份额，并且可以校验其他节点是否接收到正确的密钥份额，即交易商节点是否诚实。

[0111] PVSS协议的主要(高级)组件包括：

[0112] (i)秘密共享：交易商节点可运行算法Share(k)＝$(k_1,\cdots,k_n)$来计算份额并将其分配给各参与节点。

[0113] (ii)重建：参与者节点可以通过运行Recover(恢复)算法来重建秘密，使得Recover($\{D_i(K_i)|i\in A\}$)＝k。

[0114] (iii)验证：可以使用算法PubVerify来验证加密的份额。如果一节点操作该算法PubVerify($\{K_i|i\in A\}$)＝1→Recover($\{D_i(K_i)|i\in A\}$)＝u，且u＝j，则概交易商节点可以被确定为诚实，并且密钥份额可以是一致的。

[0115] 在一些实施方式中，根据恢复阶段的需要，PVSS方案可以是交互式的或非交互式的。

[0116] 可以考虑基于几个密码系统的PVSS协议的各种实现。为了说明，以下重点介绍了由Schoenmakers,B.在1999年8月出版的《年度国际密码学会议(Annual International Cryptology Conference)》(由总部设在柏林和海德堡的施普林格出版社出版)中的“一个简单的可公开验证的秘密共享方案及其在电子投票中的应用”一文(第148至164页)中描述的协议。

[0117] 在初始化阶段，可以使用公共过程选择一个群G_q和两个独立选择的生成器G和g。每个节点，给定其私钥$x_i \in Z_q^*$，可以将$y_i = G^{x_i}$设置为公钥。

[0118] 然后，交易商节点可以选择至多具有t-1级的随机多项式$f(x) = \sum_{j=0}^{t-1} a_j x^j$，其

系数包括在Z_q和集a_0=k之中。可以将调拨(commitment)$C_j = g^{a_j}$,j=0,…,t和加密的份额f(i)(使用参与者的公钥)$Y_i = y_i^{f(x_i)}$,i=1,…,n发布。

[0119] 通过计算$X_i = \prod_{j=0}^{t-1}(C_j)^{ij}$,交易商节点可以显示经加密的份额是一致的。特别地,交易商节点可以通过出示$X_i = g^{f(x_i)},\ Y_i = y_i^{f(x_i)}$来产生f(i)的知识证明。

[0120] 可以使用上面引用的Stadler中的Fiat-Shamir密码技术来执行密钥份额的验证。非交互协议的主要步骤包括:

[0121] 证明者(每个节点)可以选择一个随机变量$w_i \in Z_q$,可以计算$a_{1i} = g^{w_i}, a_{2i} = y_i^{w_i}$并可以广播这些值。

[0122] -使用$c=H(X_i, Y_i, a_{1i}, a_{2i})$(其中H()是密码哈希函数),证明者(例如,节点)可以计算和广播$r_i = w_i - f(x_i)c$。

[0123] -给定r_i,c,验证者节点可以计算$a_{1i} = g^{r_i} X_i^c$, $a_{2i} = y_i^{r_i} Y_i^c$,

[0124] 并且可以证明X_i,Y_i,a_{1i},a_{2i},1≤i≤n的散列与c匹配。

[0125] 必要时,参与者节点可以重建秘密s,而不必了解关于其他节点的份额$f(x_i)$的任何内容。所需的信息可以包括$S_i = G^{f(x_i)}$,i=1,…,t。通过拉格朗日插值法计算秘密

[0126] $$\prod_{i=1}^{t} S_i^{\lambda_i} = \prod_{i=1}^{t}(G^{f(x_i)})^{\lambda_i} = G^{\sum_{i=1}^{t} f(x_i)\lambda_i} = G^{f(0)} = G^k,$$

[0127] 其中,$b_{i,p}(x) = \prod_{j \in p, j \neq i} \frac{x - x_j}{x_i - x_j}$是拉格朗日系数。

[0128] 本申请的主题的以上示例和以下描述利用交易商节点来计算和分配密钥份额。然而,可以想到,可以如Fiat-Shamir启发式技术(例如,参见"Fiat-Shamir启发式",https://en.wikipedia.org/wiki/Fiat%E2%80%93Shamir_heuristic,于2018年6月13日访问)所描述的那样来实现秘密共享协议的无交易商实现方式。

[0129] 现在参见图3,其示出了本申请的示例性操作环境。如图所示,多个节点300通过计算机网络310进行通信。每个节点300是计算设备并且参与区块链,并且具有一个或多个相关联的地址,与之相关联的区块链反映了一定数量的单元,例如计算资源单元。

[0130] 节点300中的各个节点可以在分布式处理中执行处理步骤,其输出被组合以形成结果。

[0131] 本申请的主题可以在多种情况下应用。例如,图4示出了在系统尤其是飞机信息系统400的背景下本申请的特定示例操作环境。

[0132] 飞机信息系统400的分布式处理由控制计算机设备410控制,并且可以依赖于多个子系统。例如,分布式处理可以涉及各种节点,例如燃油子系统420、导航子系统430、除冰子系统440和起落架子系统450。这些子系统中的每一个可以包括子组件。例如,燃油子系统420可以包括油检查子部件422、温度子部件424、油压子部件426和燃油量子部件428。子系统可以作为飞行前检查的一部分以特定顺序进行检查,而给定子系统的子组件又可以按特定顺序进行检查。例如,可以按照图中箭头所示的顺序从燃油子系统420开始直到起落架子系统450进行子系统检查。在另一个示例中,燃油子系统420的子部件可以从油检查子部件

422开始顺时针进行检查,直到检查完燃油量子部件428。如下文进一步描述的,本申请的主题可以用于以有序的方式控制这类处理。另外,如果情况合理,则可能跳过子系统,例如在热带气候中可以跳过除冰子系统440。附加地或可替代地,如果一个子系统发生故障,则仍然可以控制其他子系统以按顺序完成其处理。

[0133] 回到图3,请记住,每个节点300是一个计算设备。图5是示例计算设备的高级操作图。示例性计算设备500可以是本文描述的一个或多个计算机系统的示例,包括例如一个或多个节点300。示例性计算设备500包括使其适于执行特定功能的软件。

[0134] 示例性计算设备500包括各种模块。例如,如图所示,示例性计算设备500可以包括处理器510、存储器520和网络接口530。如图所示,示例性计算设备500的上述组件通过总线540进行通信。

[0135] 处理器510是硬件处理器。处理器510例如可以是一个或多个ARM、Intel x86、PowerPC处理器等。

[0136] 存储器520允许存储和检索数据。存储器520可以包括例如随机存取存储器、只读存储器和永久存储器。永久存储器可以是例如闪存、固态硬盘等。只读存储器和永久存储器是非瞬时性计算机可读存储介质。计算机可读介质可以使用文件系统来组织,例如可以由管理示例性计算设备500的整体操作的操作系统来管理。

[0137] 网络接口530允许示例性计算设备500与其他计算设备和/或各种通信网络(例如计算机网络310(参见图3))进行通信。

[0138] 包括指令的软件由处理器510从计算机可读介质执行。例如,软件可以从存储器520的永久存储器加载到随机存取存储器中。附加地或可替代地,可由处理器510直接从存储器520的只读存储器执行指令。

[0139] 如下文进一步描述的,软件可以使示例性计算设备500的实例适于用作这里提到的各种计算机系统中的一个或多个,包括例如一个或多个节点300和/或一个或多个控制计算机设备410和/或子系统420-450和/或其子组件。

[0140] 参见图6,如下文进一步描述的,在本文描述的协议中,负责执行整个分布式计算任务的处理步骤的节点可以形成按与任务中的处理步骤的顺序对应的方式排序的循环有序集600。循环有序集可以被认为是环,其中每个节点610具有其他两个节点作为紧挨在该节点之后的节点和紧挨在该节点之前的节点。例如,定向环可以由节点610的循环有序集形成,如图6所示。

[0141] 如下文进一步描述的,在本申请的主题中,采用了一个以上这类循环有序节点集。

[0142] 例如,节点组成的循环有序集的节点可以对应于由给定子系统(类似于MDD协议)执行的处理步骤,其中采用多个这类循环有序集,每个集对应于特定子系统。此外,子系统之间的关系,特别是在子系统之间的处理的排序也可以使用另一个循环有序节点集(特别是其每个节点都是从对应于给定子系统的循环有序集的一个特定循环有序集中抽取的另一个循环有序节点集)来表示。

[0143] 作为背景技术,在上述共同拥有的第1706071.6、1802063.6和1806448.5号英国专利申请中讨论了类似的循环有序集,具体地,这些循环有序集与形成“环路”或“群交换环路”的一组支付通道相关联,并且成为上述MDD、GRECS和GRELR协议的一大特征。

[0144] 在另一个示例中,除了控制处理集之外,或者作为控制处理集的替代,可以将节点

组成的循环有序集的节点用于其他目的,例如,用于在GRECS或GRELR协议中的目的或与其相似的目的,这些集中的多个集被链接到另一循环有序节点集,所述另一循环有序节点集由分别从这些集之一提取的节点组成。

[0145] 在任何情况下,如图7所示,这类循环有序节点集的二维结构可以可视化为连接到整个环710中的一组环700。在图7的简化示意图中,未示出形成每个环700和整个环710的各个节点。然而,箭头指示在对应于环700的每个循环有序节点集内并且在对应于整个环710的循环有序节点集内的排序的可能方向。

[0146] 图8为这种二维结构的更详细的示意图。

[0147] 如图所示,二维结构包括诸如可以对应于环700的ω个循环有序集C_a,C_b,...C_i...C_ω。

[0148] 如图所示,每个循环有序集可以包括n_j个节点:发起者节点I_j和n-1个其他节点:$U_1^j,\ U_2^j,\ \ldots U_i^j \ldots U_{n_x-1}^j$。对于各种循环有序集$C_j$,$n_j$可以相同或不同。每个发起者节点$I_j$充当其相应的循环有序集的监督者。

[0149] 另一个循环有序集,即发起者节点的循环有序集800,由每个循环有序集C_a,C_b,...C_i...C_ω中的发起者节点I_a,I_b,...I_i...I_ω以及主发起者节点I_M组成。如下文进一步描述的,主发起者节点I_M充当整个协议的监督者。

[0150] 如下文进一步描述的,本申请的主题依赖于非对称密码技术。可以采用各种公钥密码系统。例如,可以采用椭圆曲线密码学(ECC)。在特定示例中,可以是采用椭圆曲线密码学并且采用各种私钥和相应的公钥,由此给定的私钥k及其对应的公钥P可以是椭圆曲线密码学公钥-私钥对,使得P=kG,其中G是椭圆曲线的基点,阶次为q:q×G=0,其中q是大素数,且0<k<q。在特定示例中,可以采用椭圆曲线secp256kl(定义于Certicom公司在2010年1月27日发布的“高效密码学标准2(SEC 2)”)。换言之,每个公钥及其对应的私钥可以形成椭圆曲线密码学公钥-私钥对。

[0151] 现在将描述链接每个循环有序节点集的节点的支付通道的结构。

[0152] 例如,考虑循环有序集C_a,C_b,...C_i...C_ω中给定的一个。仅作为示例,以下讨论将参考C_j,尽管其同样适用于环路C_a,C_b,...C_i...C_ω中的任何一个。示例性循环有序集C_j在图9的示意图900中示出。

[0153] 参考示意图900,Cj的节点——$U_1^j,\ U_2^j,\ \ldots U_{n_x-1}^j$和Ij——形成有序集$\{Ij,\ U_1^j,\ U_2^j,\ \ldots U_{n_x-1}^j\}$。更具体地,如图所示,节点被排序以形成如节点之间的箭头所示的循环或环。

[0154] 如图所示,密钥份额$k_\epsilon^j, k_1^j, k_2^j, \ldots, k_{n_j-2}^j, k_{n_j-1}^j$分别与节点$I_j$,$U_1^j,\ U_2^j,\ldots U_{n_x-1}^j$中的每一个相关联。特别地,如下文进一步描述的,发起者I_j生成上述密钥份额,然后将他们分配给有序集的其他节点。总的来说,发起者I_i可以生成私钥v_i,然后可以使用秘密共享协议来生成n个密钥份额(即$k_\epsilon^j, k_1^j, k_2^j, \ldots, k_{n_j-2}^j, k_{n_j-1}^j$),然后将这些份额$k_1^j, k_2^j, \ldots, k_{n_j-2}^j, k_{n_j-1}^j$(通过一些私有和安全通道)分别分配给其他节点,即$U_1^j,\ U_2^j,\ \ldots U_{n_x-1}^j$,同时保留密钥

份额中的剩余一份,即k_{ϵ}^{j},用于其自身。在一些实现方式中,可以使用可公开验证的密码秘密共享(PVSS)算法。

[0155] 值得注意的是,在使用椭圆曲线密码学(ECC)的情况下,私钥v_i可以选择为随机数,使得$0<v_j<q$。类似地,可以生成份额,使得它们也适于用作ECC私钥,包括$0 < k_i^j < q$。

[0156] 如下文进一步描述的,密钥份额用于在循环有序集C_j的节点之间建立支付通道的环路。特别地,在相邻节点之间建立支付通道,其中使用基于密钥份额的不同值来锁定和解锁那些通道。可以在由箭头连接的节点对之间创建支付通道。如下文进一步描述的,这些支付通道可以被引导以在与图9中的箭头所指示的方向相反的方向上提供资源转移。

[0157] 图10是示出图8的发起者节点的循环有序集800的图。如上所述,发起者节点的循环有序集800提供了对应于整个环710(图7)的第二维度。

[0158] 如图所示和如上所述,发起者节点的循环有序集800的发起者节点被排序以形成如发起者节点之间的箭头所示的循环或环。

[0159] 如上所述,每个发起者节点I_a,I_b,...I_i...I_ω将具有相关联的私钥v_j。另外,如图所示,主发起者I_M具有相关联的私钥v_x。如下文进一步描述的,这些相关联的私钥被用于在发起者节点的循环有序集800的节点之间建立支付通道的环路。特别地,在相邻节点之间建立支付通道,其中使用基于与发起者节点相关联的私钥的不同值来锁定和解锁那些通道。可以在由箭头连接的节点对之间创建支付通道。如下文进一步描述的,这些支付通道可以被引导以在与图10中的箭头所指示的方向相反的方向上提供资源转移。

[0160] 现在将描述各种支付通道的创建。

[0161] 首先,应当注意,使用支付通道需要将计算资源的单元调拨(commit)给该通道。例如,在支付通道的给定环路中,计算资源将由每条支付通道从节点转移到其直接支付通道。

[0162] 由于节点之间的值转移可能不是此处描述的协议的某些应用的要求,因此该数量可能是标称数量。然而,显而易见的是,每个节点都需要参与选择结果,否则其调拨到支付通道的值将被没收。因此,可能还需要节点向支付通道贡献足够多的计算资源,以阻止轻率的参与和/或违反本文描述的协议。被调拨到每条支付通道的计算资源的量可以是相同的量,或者换言之,被调拨到每条支付通道的资源在资源是可互换的情况下可以是相同的。或者,这些值可以变化。例如,在几个不同环路中的一个给定环路中,同量的计算资源可以被调拨到每条支付通道,但是那些值在那些不同环路之间并不相同。

[0163] 此外,为了解释清楚起见,采用以下符号来讨论在发起者节点组成的循环有序集的发起者节点之间形成环路的支付通道的创建以及在循环有序集C_a,C_b,...C_i...C_ω中的每一个的节点之间形成环路的支付通道的创建。

[0164] 首先,字母i被用作与发起者节点组成的循环有序集的元素(例如,发起者节点/其各自的循环有序集的每一个)相关的索引。

[0165] 其次,字母j被用作与循环有序集C_a,C_b,...C_i...C_ω中给定的C_i中的内部元素(例如,参与者)有关的索引。

[0166] 接下来,对于循环有序集C_i的使用下标和上标索引描述的元素,上标索引表示i值(集标识符),而下标索引表示集内部的j值(该循环有序集内部的顺序的组件标识符)。作为一个示例,k_4^2是循环有序集C_2的索引4处的私钥。

[0167] 接下来，n_i表示环路C_i中参与者的数量。值得注意的是，n_i包括发起者$I_i = U_0^i$。

[0168] 此外，对于给定的循环有序集C_i的索引j的任何引用应被认为是对n_i取模，这意味着U_j^i应被视为$U_{j\ mod\ n_i}^i$。例如，如果$n_i=4$，则$U_4^i = U_{4\ mod\ 4}^i = U_0^i$。

[0169] 此外，上面使用的涉及循环有序集C_i的发起者的密钥份额的符号k_ϵ^i也可以被呈现为k_0^i。这是与发起者I_i相关联的密钥份额(v_i)。

[0170] 接下来，应当注意，尽管存在ω个循环有序集(除了发起者节点组成的循环有序集之外)，但是由于存在主发起者I_M(其也可以被称为I_0)，在发起者节点组成的循环有序集中存在ω+1个节点。

[0171] 此外，对于发起者节点组成的循环有序集的索引i的任何引用应当被认为是对(ω+1)取模，这意味着，作为示例，$I_i = I_{i \bmod (\omega+1)}$。因此，如果ω=4，则$I_{4+1} = I_{5 \bmod 5} = I_0 = I_M$。

[0172] 接下来，应当注意，对发起者节点组成的循环有序集的任何引用可使用字母下标(a,b,c,d....)(例如，如图8和图10中所示)，或者，可等同地使用数字对应的索引(0,1,2,3....)。例如，后者可以用于指代发起者节点组成的循环有序集上的迭代。

[0173] 最后，要注意的是，在下面的描述中，使用了术语“紧挨在.....之前”和“紧挨在.....之后”，分别对应于图8、图9和图10所示的箭头的方向。例如，在图8中标记为I_a的节点将被认为是紧挨在节点I_b之前的节点，并且标记为I_b的节点将被认为是紧挨在节点I_a之后的节点。

[0174] 现在将参考图11、12和14所示的一系列流程图来讨论由本文描述的协议中的节点执行的示例性操作。

[0175] 首先，将结合图11讨论在发起者节点之间形成环路的支付通道的构造。图11示出了流程图1100，其示出了在准备创建支付通道的环路时执行由发起者节点组成的循环有序集的发起者节点执行的操作。由各个发起者节点中的每个节点的一个或多个处理器来执行操作1110及后续操作。这样，操作1110及后续操作由各种计算设备的一个或多个处理器执行。例如，示例性计算设备500的适当配置的实例的处理器510(图5)执行包括计算机可执行指令的软件，所述计算机可执行指令可以存储在计算机可读存储介质上，例如存储在存储器520上。

[0176] 在操作1110，主发起者节点I_M生成上述私钥v_x。

[0177] 值得注意的是，该值对于协议的安全性至关重要，特别是对于仅主发起者节点I_M知道的发起者节点组成的循环有序集的安全性至关重要。值v_x至少在每个循环有序集之间的所有支付通道形成环路之前一直保持秘密。

[0178] 另外，在操作1110，主发起者节点I_M生成对应于私钥v_x的公钥P_x^*。例如，在使用椭圆密钥密码学的情况下，主发起者节点I_M可以计算$P_x^* = v_x G$。

[0179] 接下来，在操作1120，其他发起者节点(I_a至I_ω)中的每一个生成其各自的私钥v_i。另外，每个其他发起者节点生成相应的公钥P_i^*，并将该值发送到主发起者节点I_M。例如，在采用椭圆密钥密码学的情况下，每个发起者节点可以为其各自的v_i计算$P_i^* = v_i G$。

[0180] 接下来，在操作1130，主发起者节点I_M将接收到的P_i^*值与其关联的P_x^*值组合在一

起以生成值Q_A^*。例如,在使用ECC的情况下,可以进行求和$Q_A^* = P_a^* + P_b^* + \cdots P_{\omega-1}^* + P_\omega^* + P_x^*$。

[0181] 如下所述,将在I_M和I_a之间为I_a生成一条支付通道,即$I_M \rightarrow I_a$,其中T_{pay}交易采用值Q_A^*进行锁定,这意味着必须为Q_A^*提供对应的私钥SV_A(而不是使用该密钥的签名),以解锁T_{Pay}。值得注意的是,由于定义了Q_A^*,解锁该T_{Pay}交易的解锁值SV_A将基于与所有发起者节点相关联的私钥。例如,在采用ECC的情况下,与Q_A^*对应的秘密值SV_A可以与Q_A^*相关,如下所述。首先,请记住,根据椭圆曲线加法的同态性质,E(m+n)=E(m)+E(n),其中E是函数E(x)=xG。这样,对应于Q_A^*的私钥SV_A是私钥$v_a+v_b+\ldots+v_{\omega-1}+v_\omega+v_x$的和。

[0182] 建立支付通道需要进行迭代。迭代从i=0处开始,因此在主发起者节点处开始(请回忆,$I_M=I_0$)。

[0183] 接下来,在操作1140,确定是否有充足的剩余时间来完成支付通道的建立。下文进一步描述确定是否有充足的剩余时间来完成支付通道的建立的方式。如果确定剩余时间不足,则建立终止(并且由于超时将发生所建立的任何支付通道的退款交易)。

[0184] 如果剩余时间充足,则在操作1140之后,确定是否已经在发起者节点之间创建了所有支付通道。如果是,则创建结束。或者,如果仍然需要创建支付通道以便在发起者节点组成的循环有序集的节点之间形成环路,则下一步操作是操作1150。

[0185] 在操作1150,节点I_i使用锁定值Q_{i+1}^*在锁定资源I_i和I_{i+1}之间创建支付通道。例如,在i=0的情况下,上述支付通道$I_M \rightarrow I_a$将被创建为被Q_A^*锁定。

[0186] 从操作1150开始,如果成功地创建了支付通道,则递增名义迭代器(notional iterator)i,然后进行操作1160。或者,如果存在故障,则下一步操作是操作1140,其中只要剩余时间充足,就可以尝试重建支付通道。

[0187] 在操作1160,节点I_i将基于用于锁定最后的支付通道的锁定值和与其私钥SV_i对应的公钥P_i^*来确定下一个锁定值。例如,在使用ECC的情况下,它可以计算$Q_{i+1}^* = Q_i^* - P_i^*$(其中,由于迭代器i的增加,$Q_i^*$现在指先前的锁定值)。

[0188] 从操作1160开始,下一步操作是操作1140,其中只要剩余时间充足,则将创建(根据需要)另一条支付通道(并且可以成功创建)。

[0189] 值得注意的是,在上述过程中,除主发起者节点I_M之外的每个发起者节点在其提供资源的支付通道被创建之前,都从有利于自己的支付通道的创建中获益。如此一来,发起者节点很方便地免于另一个发起者节点滥用资源的风险(无需信任主发起者节点)。值得注意的是,I_M不需要为其提供这类排序,因为它是拥有所有锁定值都基于的v_x的唯一节点。

[0190] 首先,将结合图12讨论在给定的循环有序节点集C_i的节点之间的形成环路的支付通道的构造。图12示出了流程图1200,其示出了在准备在该集中的每对相邻节点之间创建支付通道的环路时,由发起者节点组成的典型的循环有序集C_i的节点执行的操作。操作1210和后续操作由各种发起者节点中每一个的一个或多个处理器执行。这样,操作1210及后续操作由各种计算设备的一个或多个处理器执行,例如,示例性计算设备500的适当配置的实例的处理器510(图5)执行包括计算机可执行指令的软件,其中这些指令可以存储在计算机可读存储介质上,例如存储在存储器520。

[0191] 首先,在操作1210,循环有序节点集C_i的发起者节点I_i以如上所述的方式生成私钥v_i,并分配其密钥份额$k_\epsilon^i, k_1^i, k_2^i, \ldots, k_{n_i-2}^i, k_{n_i-1}^i$。

[0192] 接下来,在操作1220,发起者节点I_i获得锁定值Q_{i+1}^*。可以从发起者节点组成的循环有序集的下一个发起者节点获得Q_{i+1}^*。例如,可以从下一个循环有序节点集C_{i+1}的发起者节点接收值Q_{i+1}^*。

[0193] 接下来,在操作1230,发起者节点基于密钥份额和Q_{i+1}^*确定锁定值。特别地,当被视为私钥时,它确定与每个密钥份额对应的公钥。例如,针对k_ϵ^i时,在使用ECC的情况下,I_i可以计算$P_\epsilon^i = k_\epsilon^i G$。值得注意的是,循环有序节点集$C_i$的其他节点中的每一个也将能够以类似的方式确定与其密钥份额对应的公钥。

[0194] 然后,发起者节点I_i基于与密钥份额对应的公钥及Q_{i+1}^*来确定锁定值Q_1^i。例如,在使用ECC的情况下,可以进行求和$Q_1^i = P_1^i + P_2^i + \cdots + P_{n_i-2}^i + P_{n_i-1}^i + P_\epsilon^i + Q_{i+1}^*$。

[0195] 如下所述,将在I_j和U_1^i之间为U_1^i生成一条支付通道,即$U_0^i \rightarrow U_1^i$,其中T_{Pay}交易采用值Q_1^i进行锁定,这意味着必须为Q_1^i提供对应的私钥sv_1(而不是使用该密钥的签名),以解锁T_{Pay}。值得注意的是,由于定义了Q_1^i,解锁该T_{Pay}交易的解锁值sv_1将基于与C_i中的所有发起者节点相关联的密钥份额(并且基于Q_{i+1}^*)。例如,在采用ECC的情况下,与Q_1^i对应的秘密值sv_1^i可以通过上面解释的同态特性而与Q_1^i相关。这样,对应于Q_1^i的私钥sv_1^i是上述密钥份额和SV_{i+1}的和,即Q_{i+1}^*,即$k_1^i + k_2^i + \cdots + k_{n_i-2}^i + k_{n_i-1}^i + k_\epsilon^i + SV_{i+1}$。

[0196] 建立支付通道需要进行迭代。迭代从j=0开始,因此在发起者节点处开始(请记住$I_i = U_j^i$)。

[0197] 接下来,在操作1240,确定是否有充足的剩余时间来完成支付通道的建立。下文进一步描述确定是否有充足的剩余时间来完成支付通道的建立的方式。

[0198] 如果确定剩余时间不足,则建立终止(并且由于超时将发生所建立的任何支付通道的退款交易)。

[0199] 或者,如果剩余时间充足,则在操作1240之后,确定是否已经在C_i的节点之间创建了所有支付通道。如果是,则针对该循环有序集的支付通道的建立将随着完成而终止。或者,如果仍然需要创建支付通道以便在该循环有序集的节点之间形成环路,则下一步操作是操作1250。

[0200] 在操作1250,节点U_j^i使用锁定值Q_{j+1}^i在锁定资源U_i和U_{j+1}^i之间创建支付通道。例如,在j=0的情况下,上述支付通道$I_j \rightarrow U_1^i$将被创建为被Q_1^j锁定。

[0201] 从操作1250开始,如果成功地创建了支付通道,则递增名义迭代器j,然后进行操作1260。或者,如果存在故障,则下一步操作是操作1240,其中只要剩余时间充足,就可以尝

试重建支付通道。

[0202] 在操作1260,节点U_j^i将基于用于锁定最后支付通道的锁定值和与其密钥份额k_j^i对应的公钥P_j^i来确定下一个锁定值。例如,在使用ECC的情况下,它可以计算$Q_{j+1}^i = Q_j^i - P_j^i$(其中,由于迭代器j的增加,$Q_j^i$现在指先前的锁定值)。

[0203] 从操作1260开始,下一步操作是操作1240,其中只要剩余时间充足,则将创建(根据需要)另一条支付通道(并且可以成功创建)。

[0204] 图13提供图8的版本,其被注释为示出可以与根据前述建立的节点之间的支付通道相关联的解锁值。

[0205] 最后,应当注意,流程图1100和1200中所示的操作由于所采用和生成的各种操作的共同值而必然相互依赖。图14提供了流程图1400,其示出了流程图1100和流程图1200中的操作可以如何组合。值得注意的是,流程图1400中以点画边框示出的操作可以并行进行。另外,应当注意,以阴影填充示出的操作对应于流程图1200的各种操作。

[0206] 调拨通道

[0207] 如下文进一步描述的,本申请的主题不需要使用支付通道进行多于一次的支付迭代。实际上,通道的主要焦点不是支付本身。相反,支付通道充当托管机制以强制遵守本文所述的协议。这样,可以采用可能不同于上述现有支付通道的支付通道版本,因为这类不同版本的支付通道中的重点可能在于托管资源和提供时间锁定的退款交易,而不是提供多次退款-支付交易的迭代。支付通道的这类变体可以被称为调拨通道(commitment channel)。

[0208] 调拨通道是"单向"通道,因为资源的转移仅在一个方向上,类似于上述支付通道的示例实现。对于一对节点,使得U_A支付U_B的调拨通道可以通过符号$U_A \rightarrow U_B$来表示。

[0209] 图15提供$U_A \rightarrow U_B$调拨通道的表示1500。

[0210] 如图所示,调拨通道包括三个交易:T_c、T_{pay}和T_r。应当注意,符号$\sigma(U_x)$表示U_x的加密签名。

[0211] T_c交易表示支付通道的调拨组件。这里,通过交易,U_b发送/调拨指定数量的单元,以由2-2的多重签名($\mathbf{U_j^i}$和$\mathbf{U_{j+1}^i}$)或秘密值$\mathbf{sv_{j+1}^i}$和加密签名$\mathbf{U_{j+1}^i}$管理。

[0212] T_r交易代表将指定数量的单元(从调拨交易(commitment transaction)中)退还给U_b,并在指定时间到期后才有资格提交给区块链。这样,T_r被设置为在满足时间到期的退还条件时将资源的控制退还给特定节点U_b。为了成功地执行该交易,它需要U_A和U_B的加密签名。

[0213] T_{pay}交易是将指定数量的单元从U_b调拨的资金转移到U_a(即从与U_b相关联的源地址转移到与U_a相关联的目的地址)。为了成功地执行该交易,它需要秘密值$\mathbf{sv_{j+1}^i}$和U_B的加密签名。换言之,交易将仅响应于满足执行条件而执行,该执行条件包括提供所需要的值,包括解锁值。

[0214] 可以使用区块链来提供调拨通道的实现。在这类实现方式中,T_c、T_r和T_{pay}中的每一个都可以是交易(transaction)。相应的交易1600、1700和1800分别在图16、图17和图18中示出。值得注意的是,交易1600、1700和1800中的一个或多个可以采用非标准操作码OP_

ECPMULT。OP_ECPMULT取编码的椭圆曲线点和数字,并按标量执行椭圆曲线乘法,输出结果作为编码的椭圆曲线点。2017年4月10日提交的名称为“计算机实现的系统和方法”的共同拥有的第1705749.8号英国专利申请提供了OP_ECPMULT操作码的详细信息。

[0215] 图19提供了表格1900,该表格示出了如何使用操作码OP_ECPMULT来将调拨交易1600的<scriptPubKey>与支付交易1800的<scriptSig>组合以解锁所调拨的资源。

[0216] 剩余时间充足

[0217] 现在将讨论确定是否有足够的时间来创建支付通道的方式。

[0218] 考虑到支付通道是调拨通道的情况,这类“充足”或“充足时间”概念的核心是:当一对参与者创建支付通道$U_A \rightarrow U_B$时,他们必须确保考虑到:

[0219] 1.每条调拨通道都有一个nTimeLocked退款交易T_r,参与者U_A可以将其提交给区块链;如果U_A不满足特定条件,则将调拨通道的托管资金返还给U_A;

[0220] 2.必须为每个参与者U_A提供一个“公平(fair)”或“商定的(agreed-upon)”时间段,以构建通道$U_A \rightarrow U_B$并接收他们的资金;

[0221] 3.创建$U_A \rightarrow U_B$后,可能还需要创建其他调拨通道。这可以包括:

[0222] a)相同循环有序集C_i的环路中的调拨通道,

[0223] b)与其他循环有序集相关联的环路中的调拨通道,其中C_k:k>i,

[0224] c)与发起者的循环有序集相关联的环路中的调拨通道$U_e \rightarrow U_f$,其中C_k:e>A;以及

[0225] 4.当主发起者I_M将第一T_{pay}交易提交到区块链时(在公开$SV_x = v_x$的过程中),有一个预期的约定时间T_{2D}。

[0226] 还应当注意,如果在开始时间T_{2D}之前出现了错误(包括未创建所有必需的通道),则参与者U_b期望确保他们有充足的时间来:

[0227] a)从调拨通道$U_b \rightarrow U_c$提交退款T_r;或

[0228] b)从调拨通道$U_a \rightarrow U_b$提交支付T_{pay},假定U_c已经类似地从调拨通道$U_b \rightarrow U_c$提交了支付T_{pay}。

[0229] 考虑到由于计划的开始时间T_{2D}而导致的这些情况,确定是否剩余“充足时间”必须考虑是否有充足的时间让循环有序集C_i(或发起者节点组成的循环有序集)中的每个节点有机会创建自己的通道并接收退款。

[0230] 为此,可以如下定义值s_i和$S^*_{i \rightarrow i+1}$。

[0231] s_i表示各个U^i_j被给予的时间量,以进行:

[0232] 1.构建调拨通道$U^i_j \rightarrow U^i_{j+1}$,

[0233] 2.确定在区块链内找到的必要的秘密值,以及

[0234] 3.向区块链网络提交支付U^i_{j+1}的交易T_{Pay}。

[0235] s_i可以以诸如秒或块数等单位来表示。

[0236] $S^*_{i \rightarrow i+1}$表示给予给定发起者以创建调拨通道$I_i \rightarrow I_{i+1}$。$S^*_{i \rightarrow i+1}$的时间应足够多,以便:

[0237] 1.可以构建与循环有序集C_{i+1}相关联的环路的所有调拨通道$U^{i+1}_j \rightarrow U^{i+1}_{j+1}$,

[0238] 2.可以在区块链内找到与循环有序集C_{i+1}相关联的环路的调拨通道的所有必需的秘密值，

[0239] 3.可以将环路C_{i+1}的所有T_{Pay}交易提交到区块链，

[0240] 4.可以构建与发起者节点组成的循环有序集相关联的环路的调拨通道$I_i \rightarrow I_{i+1}$，

[0241] 5.可以在区块链内找到调拨通道$I_i \rightarrow I_{i+1}$的必要秘密值，以及

[0242] 6.可以将$I_{i-1} \rightarrow I_i$的T_{Pay}交易提交到区块链网络。

[0243] 值得注意的是，$S^*_{i \rightarrow i+1}$值的选择可以考虑到上述操作中的一个或多个可以并行发生。

[0244] $S^*_{i \rightarrow i+1}$可以用例如秒或块数等单位来表示。

[0245] 利用如上所定义的值s_i和$S^*_{i \rightarrow i+1}$，用于评估剩余时间是否充足的条件可以对应于T_{2D}-currenttime>f({si:i∈[a,ω]})或$T_{2D} - current\ time > f(\{S^*_{i \rightarrow i+1} : i \in [a, \omega]\})$，如果合适的话。

[0246] 上面解释了如何为各种循环有序集建立支付通道的环路。现在将结合图20描述这些环路在控制系统组件的排序处理中的使用(例如，在与节点的每个ω循环有序集C_a，C_b,...C_i...C_ω对应的子系统之间)，及在这些组件的子组件中的使用(例如，在与ω循环有序集C_a,C_b,...C_i...C_ω的每个节点对应的子组件之间)。

[0247] 图20示出了流程图2000，其示出了由各个循环有序集的节点执行的操作，该操作用于控制各个循环有序集的节点之间以及构成每个集的节点之间的处理的顺序。由各个节点中的每个节点的一个或多个处理器来执行操作2010及后续操作。这样，操作2010及后续操作由各种计算设备的一个或多个处理器执行，例如，示例性计算设备500的适当配置的实例的处理器510(图5)执行包括计算机可执行指令的软件，所述计算机可执行指令可以存储在计算机可读存储介质上，例如存储在存储器520上。

[0248] 图20所示的操作假定已经建立了支付通道以在每个循环有序节点集(包括发起者节点组成的循环有序集)的节点之间形成环路。

[0249] 在考虑图20时，对于读者来说，参考图8或图13可能是有帮助的。记住，在那些附图所示的示例中，除了发起者节点组成的循环有序集(即ω＝d＝4)之外，还存在四个循环有序集，并且这四个循环有序集的每一个都恰好包括四个节点(即$n_i = 4\ \forall\ i \in [1,d]$)。应当注意，根据图8和图13中的各个循环有序节点集的表示，那些节点之间的支付通道的执行以及因此由那些节点进行的处理的触发以逆时针方向进行。

[0250] 在操作2010，主发起者I_M为其自身($I_\omega \rightarrow I_M$)提交支付通道的交易T_{pay}。值得注意的是，由于支付通道的构造，该通道的T_{Pay}将被锁定值Q^*_x锁定，并被相应的解锁值SV_x解锁(锁定值是与解锁值的公钥对应的私钥)其中Q^*_x和SV_x都基于v_x。(请回忆，到现在为止，只有I_M才知道v_x。)值得注意的是，这具有在提交T_{Pay}时区块链上公开v_x的效果。

[0251] I_ω将需要SV_x的值来解锁T_{Pay}交易。在一些实现方式中，I_M可以将SV_x发送到I_ω。附加地或可替代地，I_ω可以从区块链中检索SV_x。

[0252] 需要迭代，以便处理可以在子系统之间进行(因此，通过与该子系统对应的循环有序集的节点，由该循环有序集的发起者触发)。迭代从i＝ω开始，因此在循环有序节点集C_ω

的发起者节点I_{ω}处开始。

[0253] 接下来,在操作2020,确定是否有充足的剩余时间来完成与还未完成处理的循环有序节点集(除了循环有序发起者节点集之外)相关联的环路的支付通道的提交,以及是否有充足的剩余时间来完成与循环有序发起者节点集相关联的环路的剩余支付通道的提交。

[0254] 如果确定剩余时间不足,则处理进行到操作2030,在该操作,发起者节点I_{i-1}向区块链提交它创建的支付通道的退款交易(即$T_r:I_{i-1}\rightarrow I_i$)。值得注意的是,这将具有触发链式反应的效果,由此,循环有序发起者节点集的所有其他发起者节点都将提交退款交易。这也将具有使尚未执行的循环有序集的节点提交其退款交易的效果。

[0255] 或者,如果在操作2020确定剩余时间充足,则处理进行到操作2040。

[0256] 在操作2040,发起者节点I_i确定与它所属的(非发起者)节点组成的循环有序集C_i相关联的处理是否应当进行。这类确定例如可以基于由节点整体或在特定子系统中进行的分布式处理的特定目的和应用的环境。例如,如结合图4所讨论的,可能存在这类情况,其中将由特定的循环有序节点集执行的处理可能是不必要的或无保证的。

[0257] 如果确定应当跳过与循环有序节点集C_i相关联的处理,则处理继续进行操作2050。或者,如果确定与循环有序节点集C_i相关联的处理应该进行,则处理进行到操作2070。

[0258] 因为,如下文进一步所述,给定循环有序节点集C_i的每个节点进行处理会导致将交易调拨给区块链,因此区块链提供了有关发生了什么处理的审计跟踪。此外,考虑到区块链的特性,这类审计跟踪可以被认为是防篡改的。

[0259] 在操作2050,发起者节点I_i从区块链中检索sv_{i+1}。附加地或替代地,它可以从I_{i+1}接收该值。一旦获得sv_{i+1},该值就可以被用来确定sv_i,因为为该值基于v_i和sv_{i+1}所基于的组件,即对应于发起者节点组成的循环有序集的每个节点的私钥v_j,该发起者节点来自紧挨在发起者节点组成的循环有序集中I_i之后的发起者节点直到主发起者节点。例如,在使用ECC的情况下,可以对v_i和sv_{i+1}进行求和。

[0260] 处理从操作2050进行到操作2060。在操作2060,I_i将用于支付通道$I_{i-1}\rightarrow I_i$的支付交易T_{Pay}提交给区块链。值得注意的是,这具有在区块链上公开sv_i的效果,从而允许序列中的下一个发起者节点(即I_{i-1})继续其处理。

[0261] 在操作2060之后,名义迭代器i递减,并且假设存在剩余的发起者节点(即i>0),则处理在操作2020处继续。或者,如果所有发起者节点都有机会执行,则处理终止。

[0262] 请记住,I_i还可能已经确定不跳过与循环有序集C_i的节点关联的处理。在这类情况下,如上所述,处理进行到操作2070,在该操作,发起循环有序集C_i的处理。

[0263] 在操作2070,发起者节点识别sv_{ϵ}^{i}的值。(请回忆,$sv_{\epsilon}^{i}=k_{\epsilon}^{i}+SV_{i+1}$)。

[0264] 在操作2070之后的操作2080,发起者节点I_i向区块链提交T_{pay}交易,从而公开sv_{ϵ}^{i}的值。值得注意的是,如下文进一步描述的,这具有公开节点$U_{n_i-1}^{i}$计算向区块链提交T_{pay}交易所需的解锁值所需的值的效果。这样,操作2080具有由节点$U_{n_i-1}^{i}$触发处理的效果。

[0265] 需要迭代,以便处理可以在循环有序集C_i的节点上进行,即在相应子系统的子组件上进行。值得注意的是,当处理到达每个这类节点时,触发该处理以执行与该节点相关联

的任何处理。

[0266] 迭代从j＝n_i-1开始,因此,在循环有序节点集C_i的节点$U_{n_i-1}^i$处开始。

[0267] 处理从操作2080进行到操作2090。在操作2090,确定是否有充足的剩余时间来完成与循环有序节点集C_i相关联的环路的支付通道的提交。

[0268] 如果确定剩余时间不足,则处理进行到操作2100,在该操作,节点U_{j-1}^i向区块链提交它创建的支付通道的退款交易(即$T_r: U_{j-1}^i \rightarrow U_j^i$),从而发起$C_i$的其他节点依次提交相应的退款交易$T_r: U_k^i \rightarrow U_{k+1}^i$。

[0269] 或者,如果在操作2090确定剩余时间充足,则处理进行到操作2110。

[0270] 在操作2110,与循环有序节点集Ci相关联的环路中的下一个节点U_j^i从序列中的前一个节点提交到区块链的支付交易(即$T_{Pay}: U_j^i \rightarrow U_{j+1}^i$)中检索$sv_{j+1}^i$,并基于该值及其密钥份额$k_j^i$来确定$sv_j^i$。例如,在使用ECC的情况下,它可以计算$sv_j^i = k_j^i + sv_{j+1}^i$。

[0271] 处理从操作2110进行到操作2120。在操作2120,节点U_j^i将支付通道的支付交易(即$T_{Pay}: U_{j-1}^i \rightarrow U_j^i$)提交给区块链。值得注意的是,这具有在区块链上公开sv_j^i的效果。

[0272] 从操作2120开始,当处理进行到循环有序节点集C_i中的下一个节点时,名义迭代器j递减。如果该集中剩余的节点尚未完成处理,则处理进行到操作2090。或者,如果针对C_i中的所有节点的处理已经完成,则控制流程返回到操作2050。

[0273] 在操作2050,如上所述,下一个发起者节点(I_i)从区块链中检索SV_{i+1},以便它可以计算SV_i,然后,在操作2060,提交支付通道的T_{pay}交易,其使用支付通道的SV_i进行解锁,就像紧挨在发起者节点I_i之前的节点已经准备的那样。然而,值得注意的是,因为与循环有序集C_i的每个节点相关联的秘密值是v_i的密钥份额,并且那些值将通过提交循环有序集C_i的节点的支付交易而全部被公开在区块链上,即使I_i无法向区块链提交用于支付通道I_{i-1}→I_i的T_{pay}交易,仍然由下一个发起者节点I_{i-1}通过使用揭示的密钥份额重建v_i来发起循环有序集C_i的节点的处理。

[0274] 上述实施例实际上涉及循环有序集的一个循环有序集,前者是指循环有序集C_i,后者是指由发起者的循环有序集定义的它们之间的关系(除主发起者节点外,该循环有序集的每个成员还都是循环有序集C_i之一的成员)。换言之,上述实施例有效地涉及二维的循环有序集。在一些实施例中,上述实施例的主题可进一步概括以允许更多的维度。例如,可以定义发起者节点组成的循环有序集,这些节点中的每一个都是发起者节点组成的相应另一循环有序集的主发起者,然后这些发起者节点中的每一个都是相应循环有序集的发起者,从而提供三个维度。

[0275] 应当注意的是,上述实施例是说明性的,而不是限制本公开,并且本领域技术人员能够在不脱离本公开的范围的前提下设计出多种替代实施例。在权利要求书中,括号中的任何附图标记都不应解释为对权利要求的限制。词语“包括”等不排除任一项权利要求或说明书中整体列出的元件或步骤之外的元件或步骤的存在。在本说明书中,“包括”是指“包

含”或“由.....组成”。元件的单数形式并不排除此类元件的复数形式,反之亦然。本公开可以通过包括几个不同元件的硬件以及通过适当编程的计算机来实现。在列举几个装置的设备权利要求中,这些装置中的几个装置可以由同一硬件来体现。在互不相同的从属权利要求中引用某些措施的事实并不意味着不能有利地使用这些措施的组合。

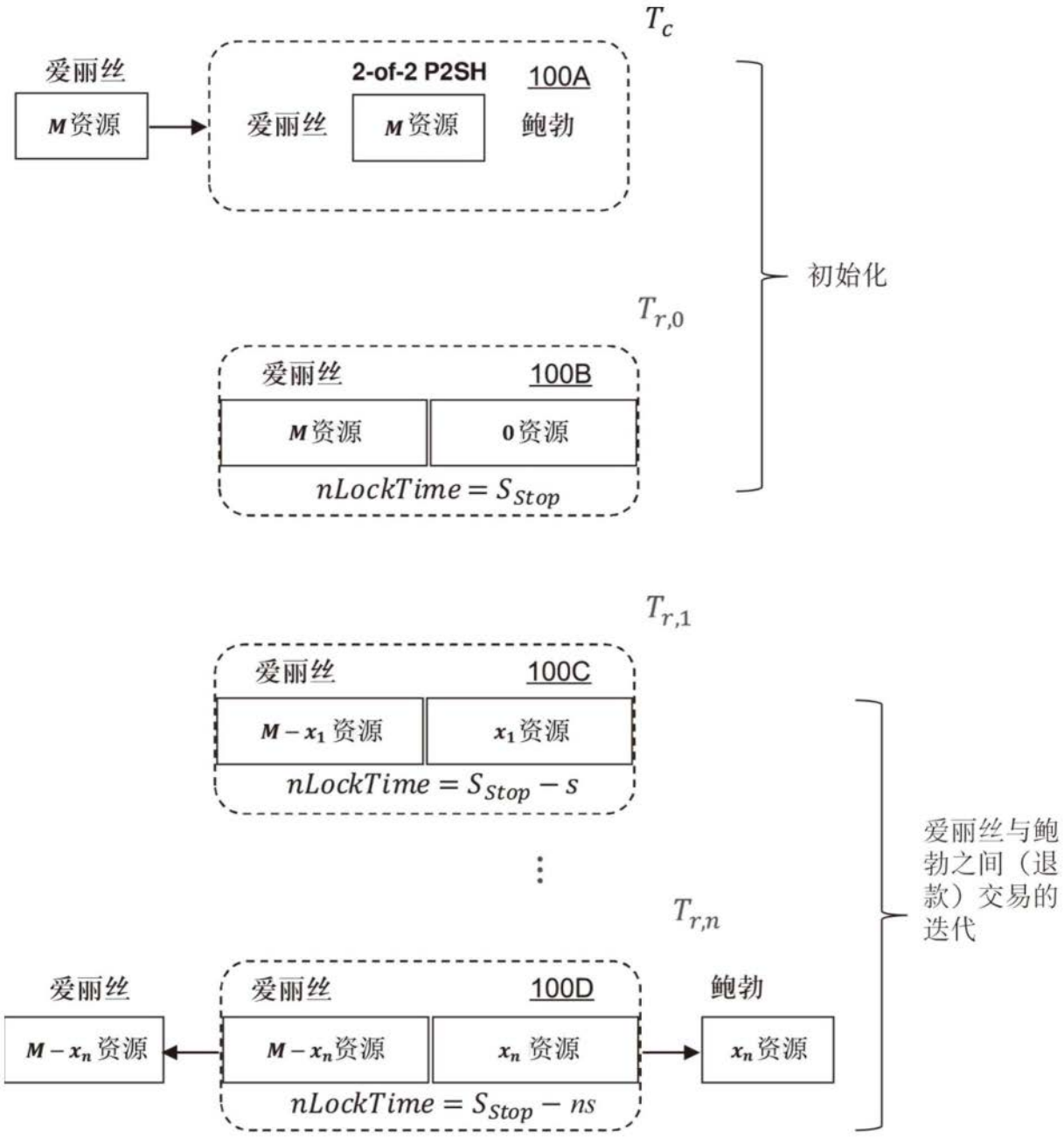

T_c
爱丽丝
M资源
2-of-2 P2SH
100A
爱丽丝
M资源
鲍勃
初始化
$T_{r,0}$
爱丽丝
100B
M资源
0资源
$nLockTime = S_{Stop}$
$T_{r,1}$
爱丽丝
100C
$M - x_1$ 资源
x_1资源
$nLockTime = S_{Stop} - s$
爱丽丝与鲍勃之间（退款）交易的迭代
$T_{r,n}$
爱丽丝
爱丽丝
100D
鲍勃
$M - x_n$ 资源
$M - x_n$资源
x_n 资源
x_n资源
$nLockTime = S_{Stop} - ns$

图1

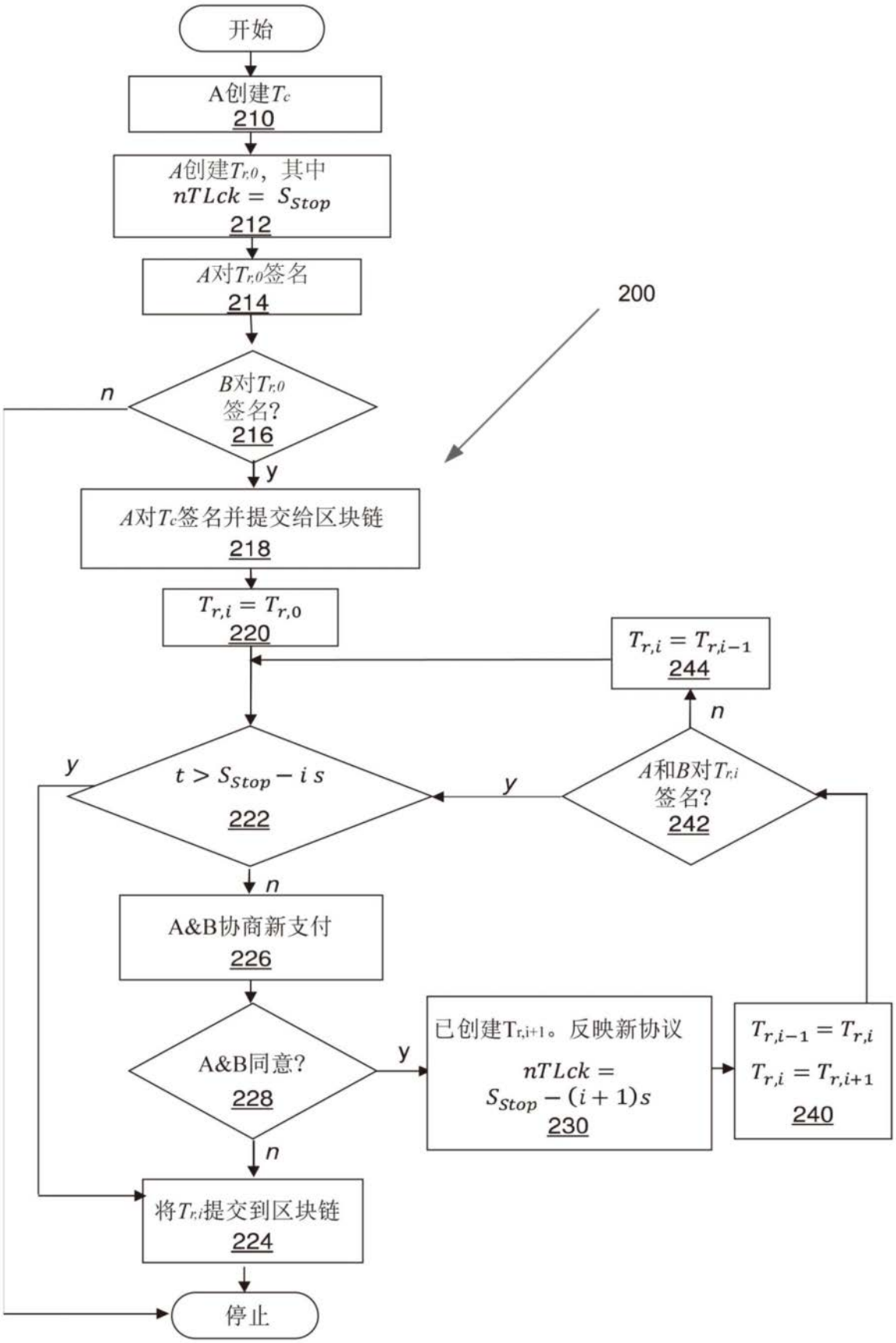

开始
A创建T_c
210
A创建$T_{r,0}$，其中
$nTLck = S_{Stop}$
212
A对$T_{r,0}$签名
214
200
B对$T_{r,0}$
签名?
216
n
y
A对T_c签名并提交给区块链
218
$T_{r,i} = T_{r,0}$
220
$T_{r,i} = T_{r,i-1}$
244
n
y
$t > S_{Stop} - i\,s$
222
y
A和B对$T_{r,i}$
签名?
242
n
A&B协商新支付
226
A&B同意?
228
y
已创建$T_{r,i+1}$。反映新协议
$nTLck =$
$S_{Stop} - (i+1)s$
230
$T_{r,i-1} = T_{r,i}$
$T_{r,i} = T_{r,i+1}$
240
n
将$T_{r,i}$提交到区块链
224
停止

图2

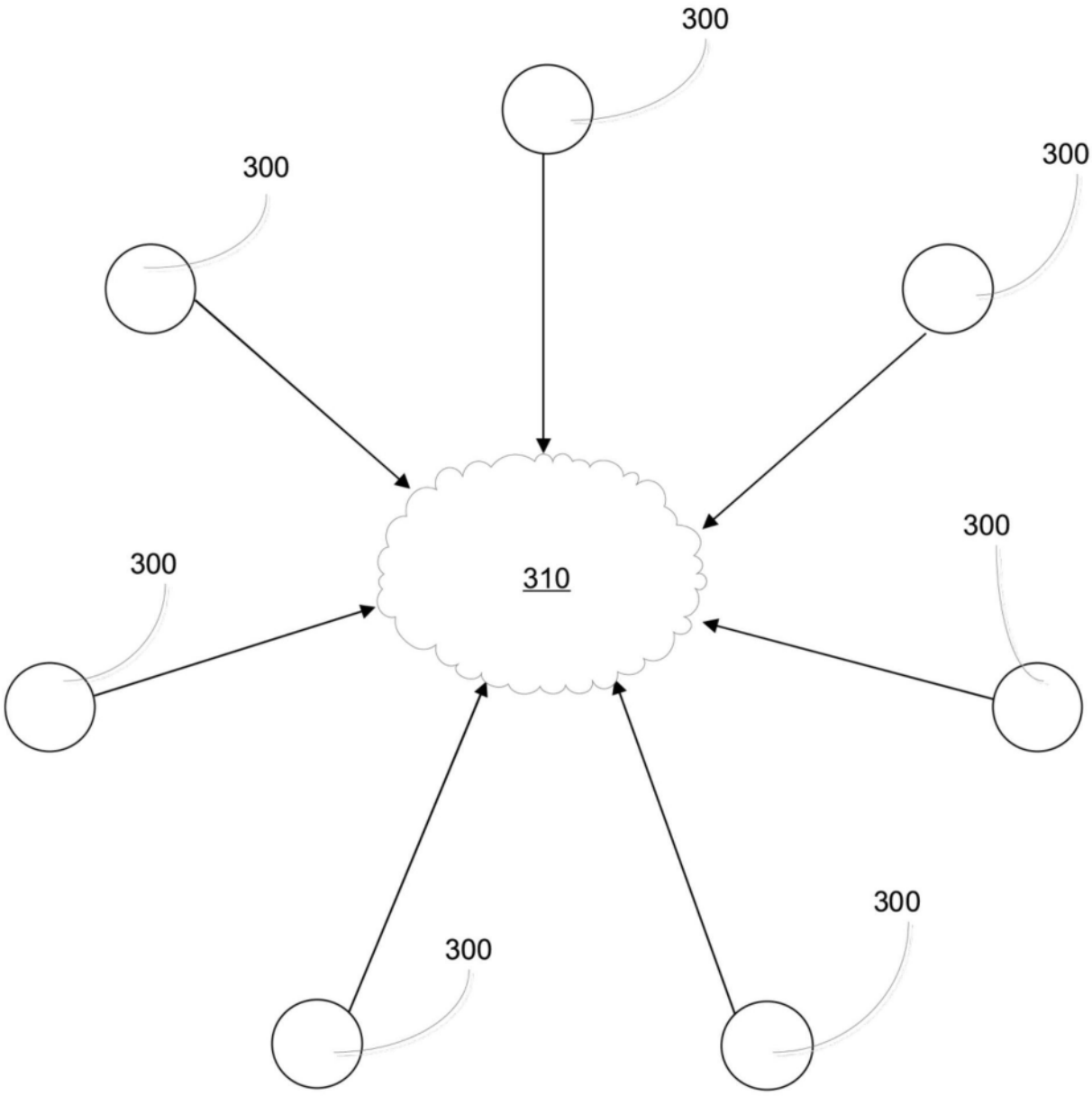


图3

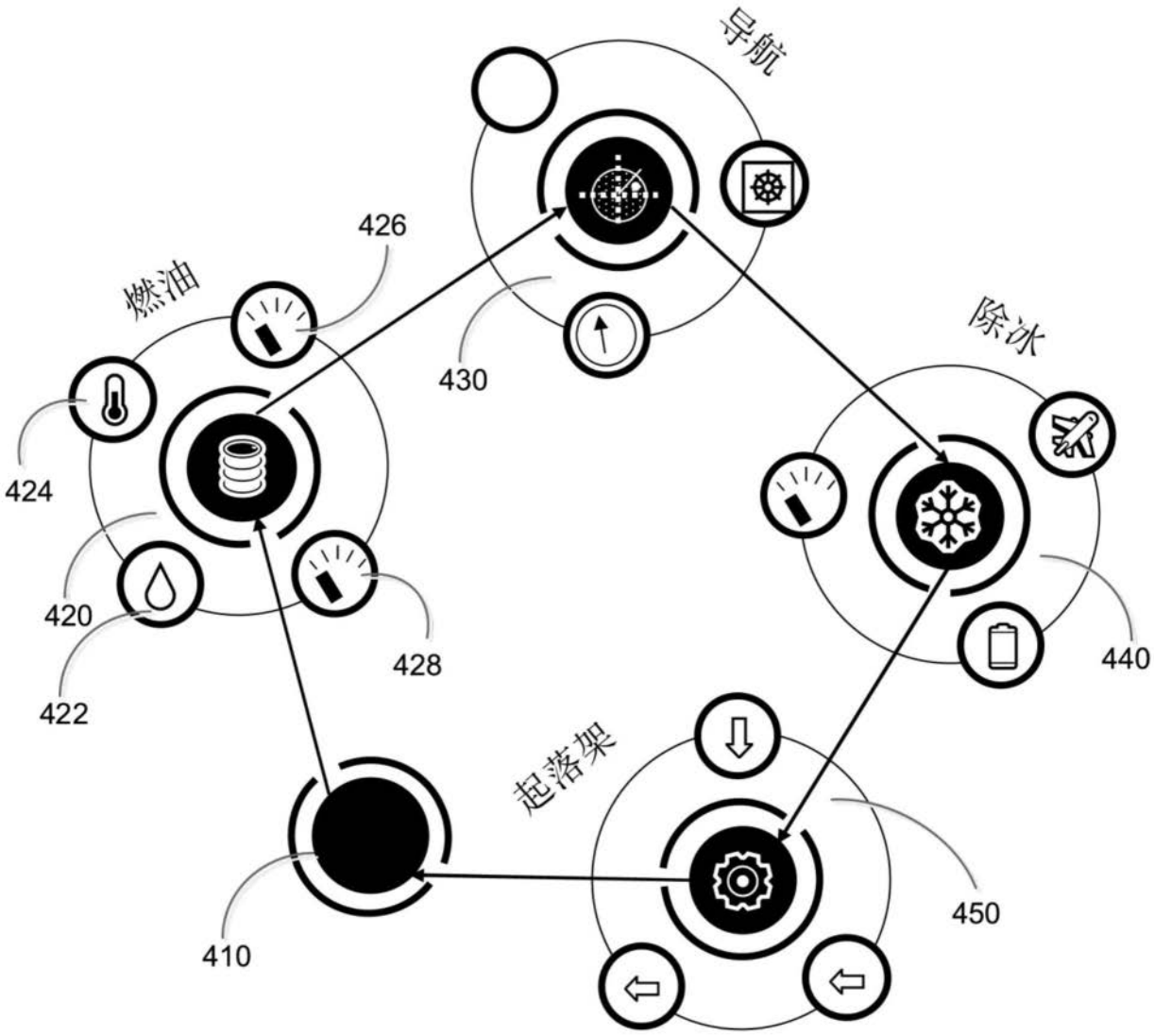


图4

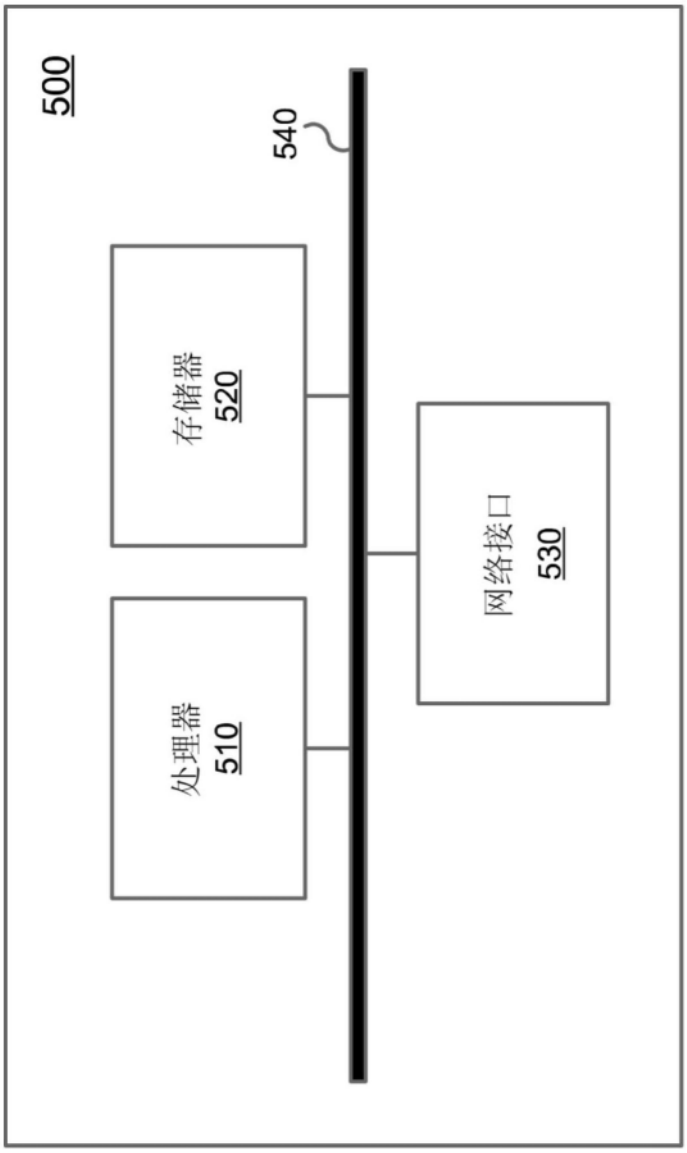


图5

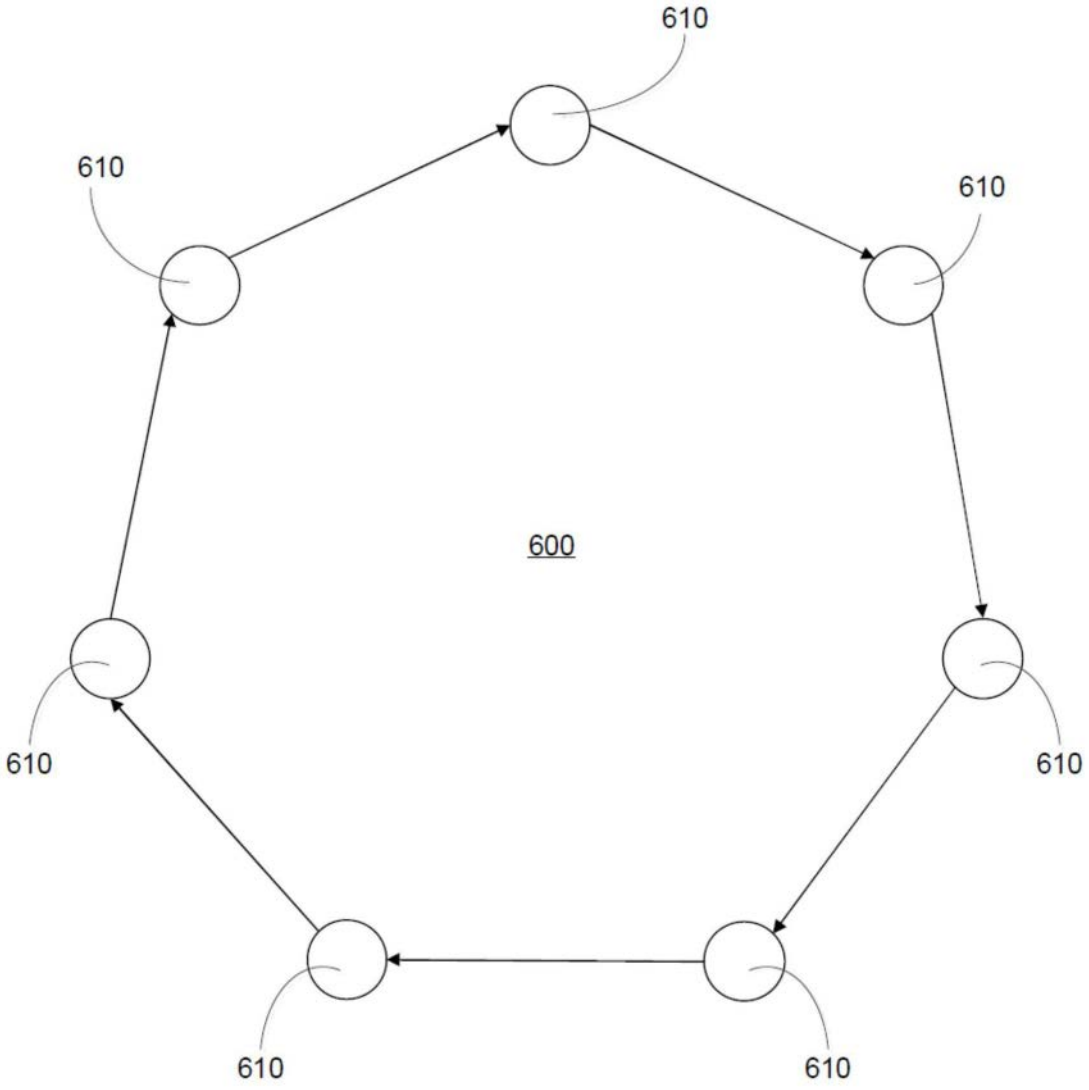


图6

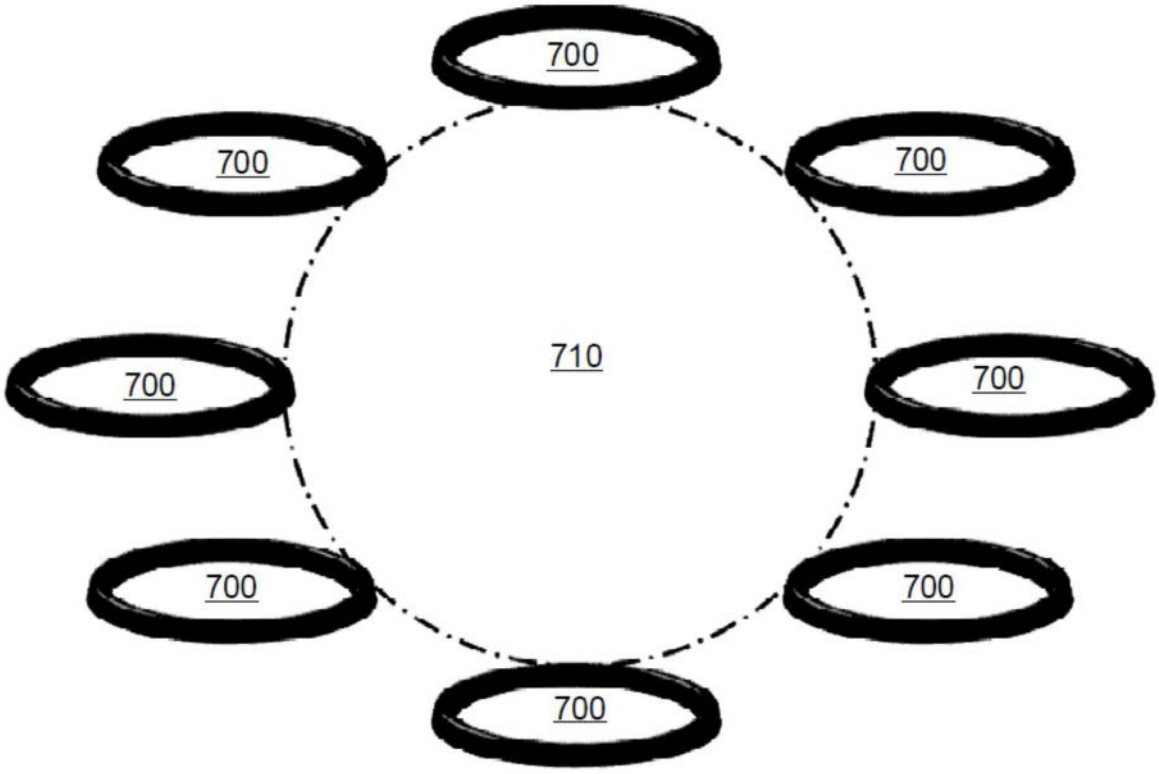


图7

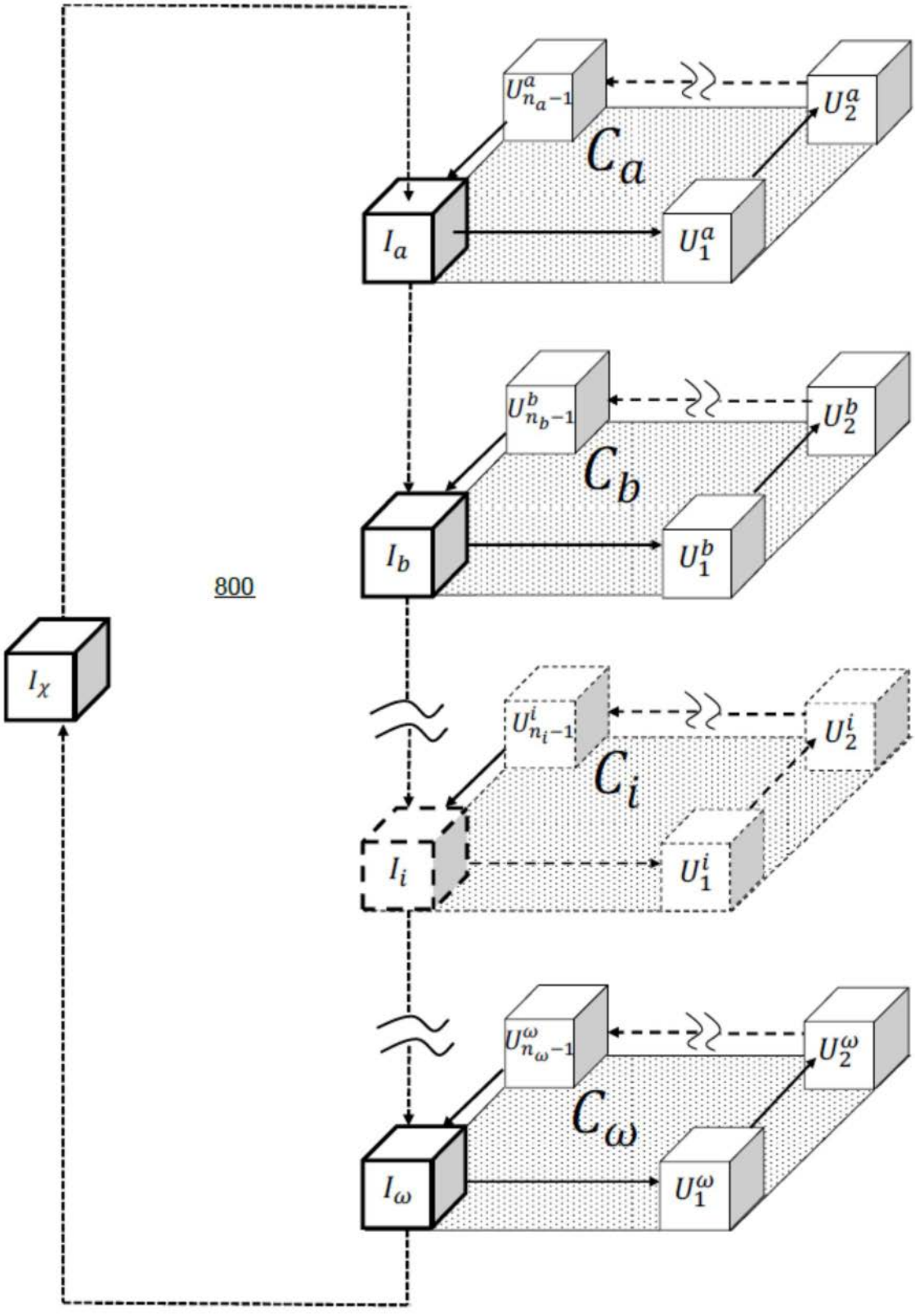

800
I_χ
I_a
C_a
$U^a_{n_a-1}$
U^a_2
U^a_1
I_b
C_b
$U^b_{n_b-1}$
U^b_2
U^b_1
I_i
C_i
$U^i_{n_i-1}$
U^i_2
U^i_1
I_ω
C_ω
$U^\omega_{n_\omega-1}$
U^ω_2
U^ω_1

图8

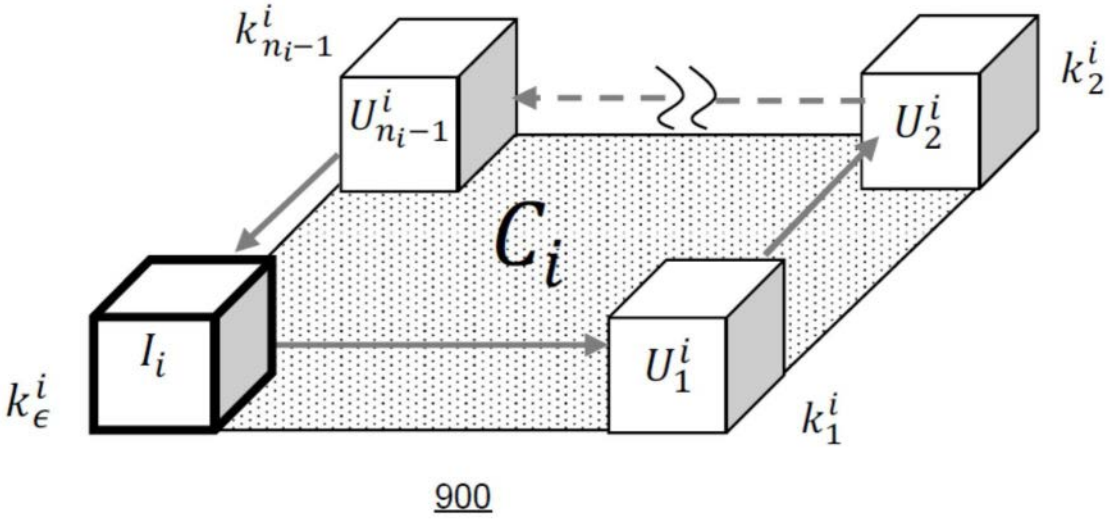


图9

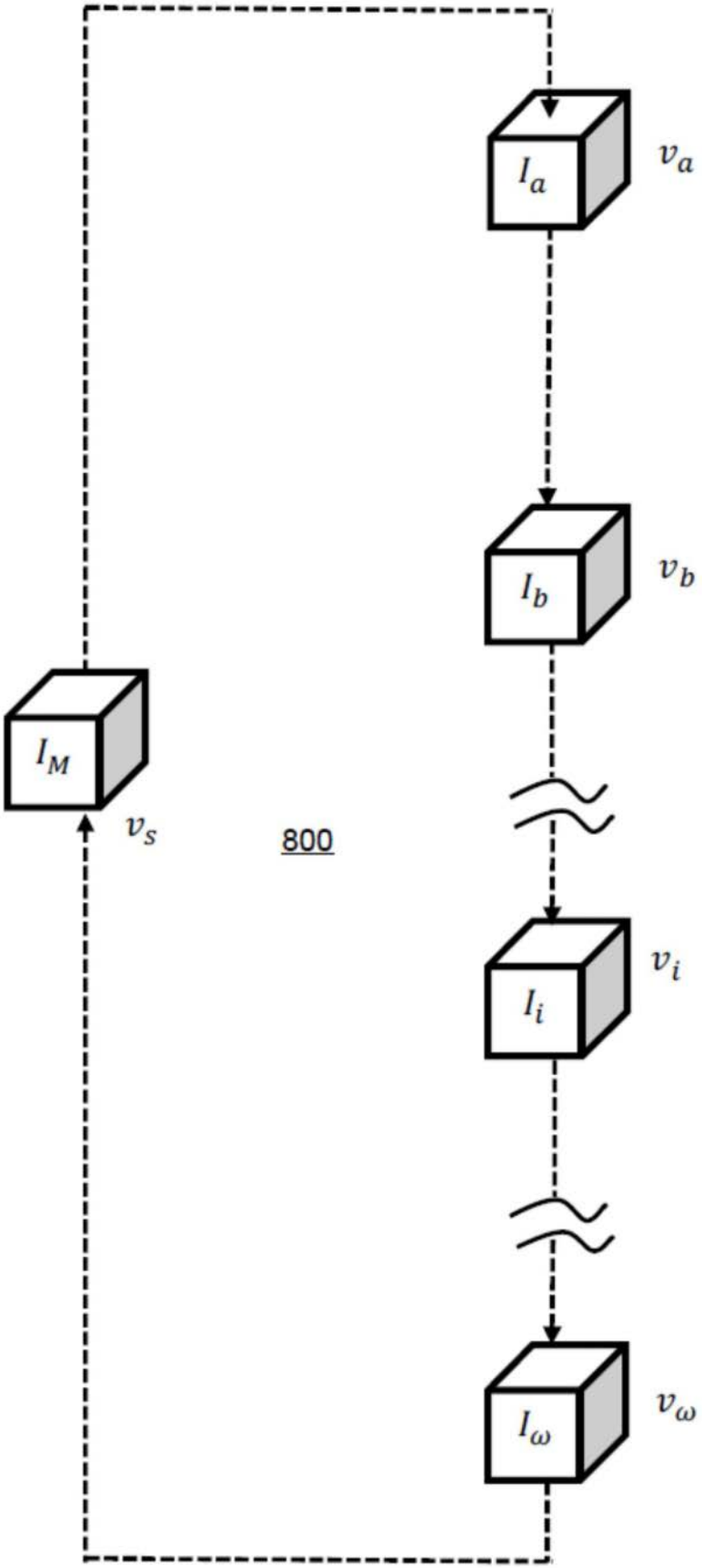


图10

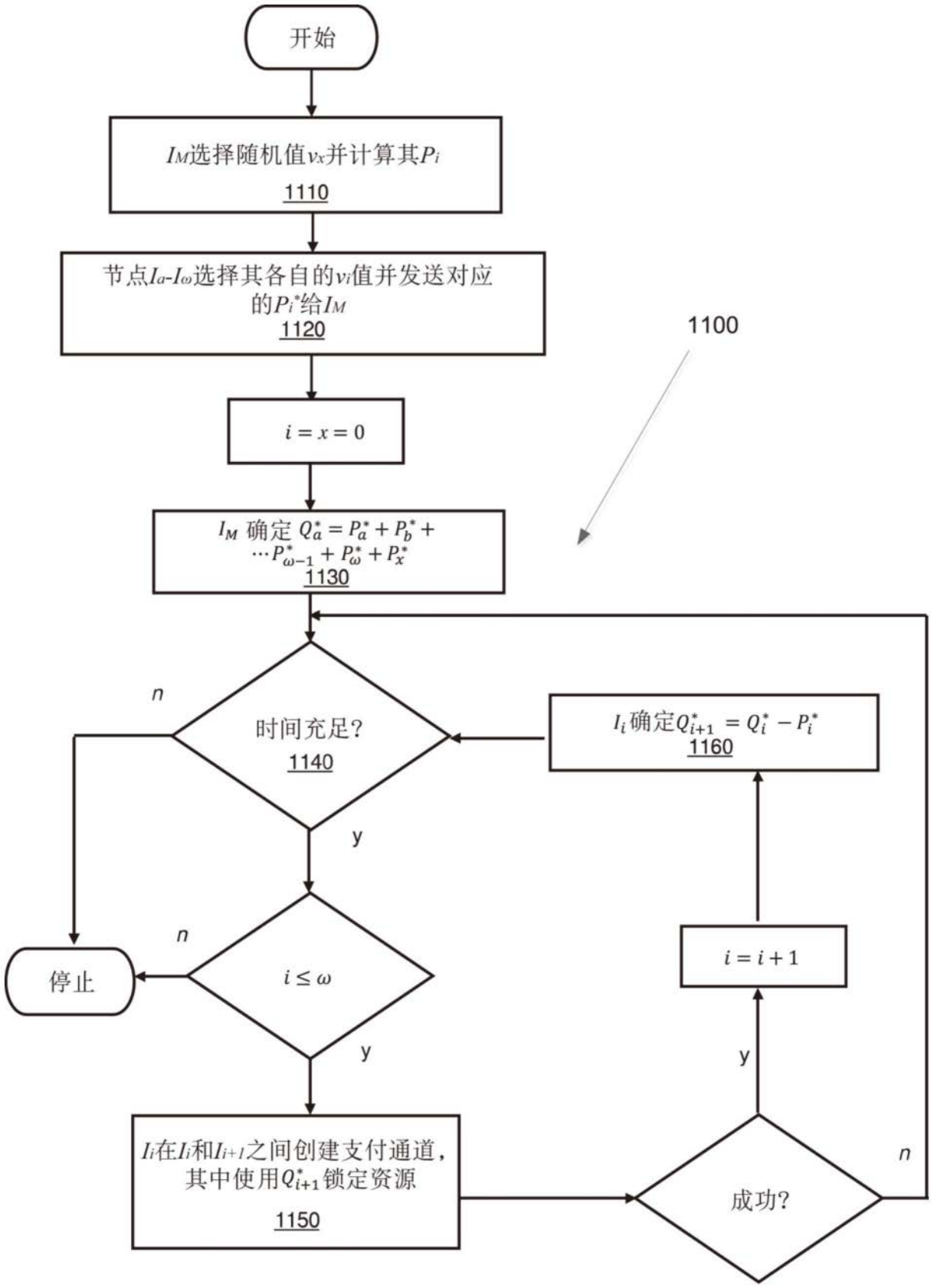

开始
I_M选择随机值v_x并计算其P_i
1110
节点I_a-I_ω选择其各自的v_i值并发送对应的P_i^*给I_M
1120
1100
$i = x = 0$
I_M 确定 $Q_a^* = P_a^* + P_b^* + \cdots P_{\omega-1}^* + P_\omega^* + P_x^*$
1130
时间充足?
1140
n
y
I_i确定$Q_{i+1}^* = Q_i^* - P_i^*$
1160
$i \leq \omega$
n
y
停止
$i = i + 1$
y
I_i在I_i和I_{i+1}之间创建支付通道，其中使用Q_{i+1}^*锁定资源
1150
成功?
n

图11

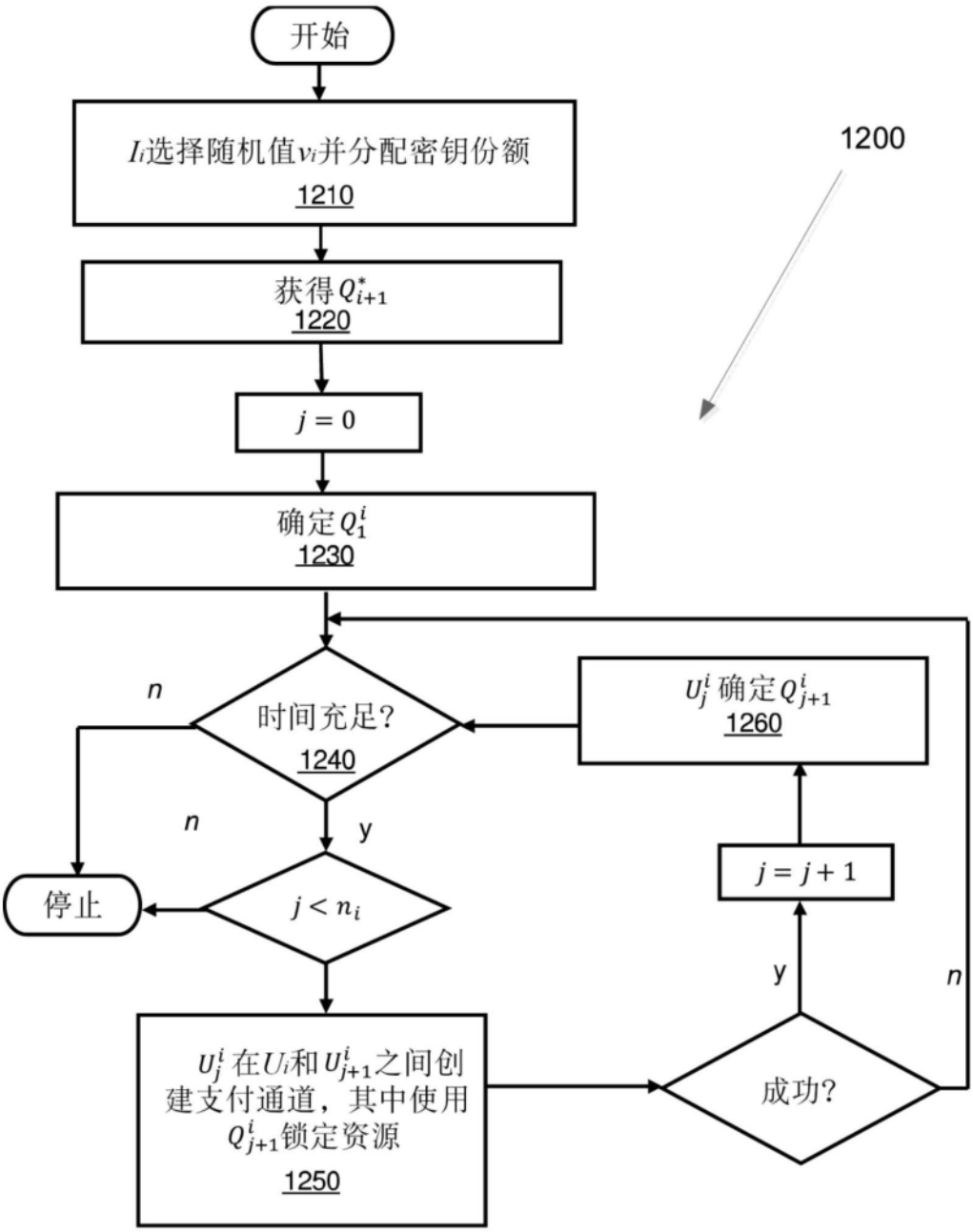

开始
I_i选择随机值v_i并分配密钥份额
1210
获得Q_{i+1}^*
1220
$j=0$
确定Q_1^i
1230
时间充足?
1240
n
y
n
停止
$j<n_i$
U_j^i在U_i和U_{j+1}^i之间创建支付通道，其中使用Q_{j+1}^i锁定资源
1250
成功?
y
n
$j=j+1$
U_j^i确定Q_{j+1}^i
1260
1200

图12

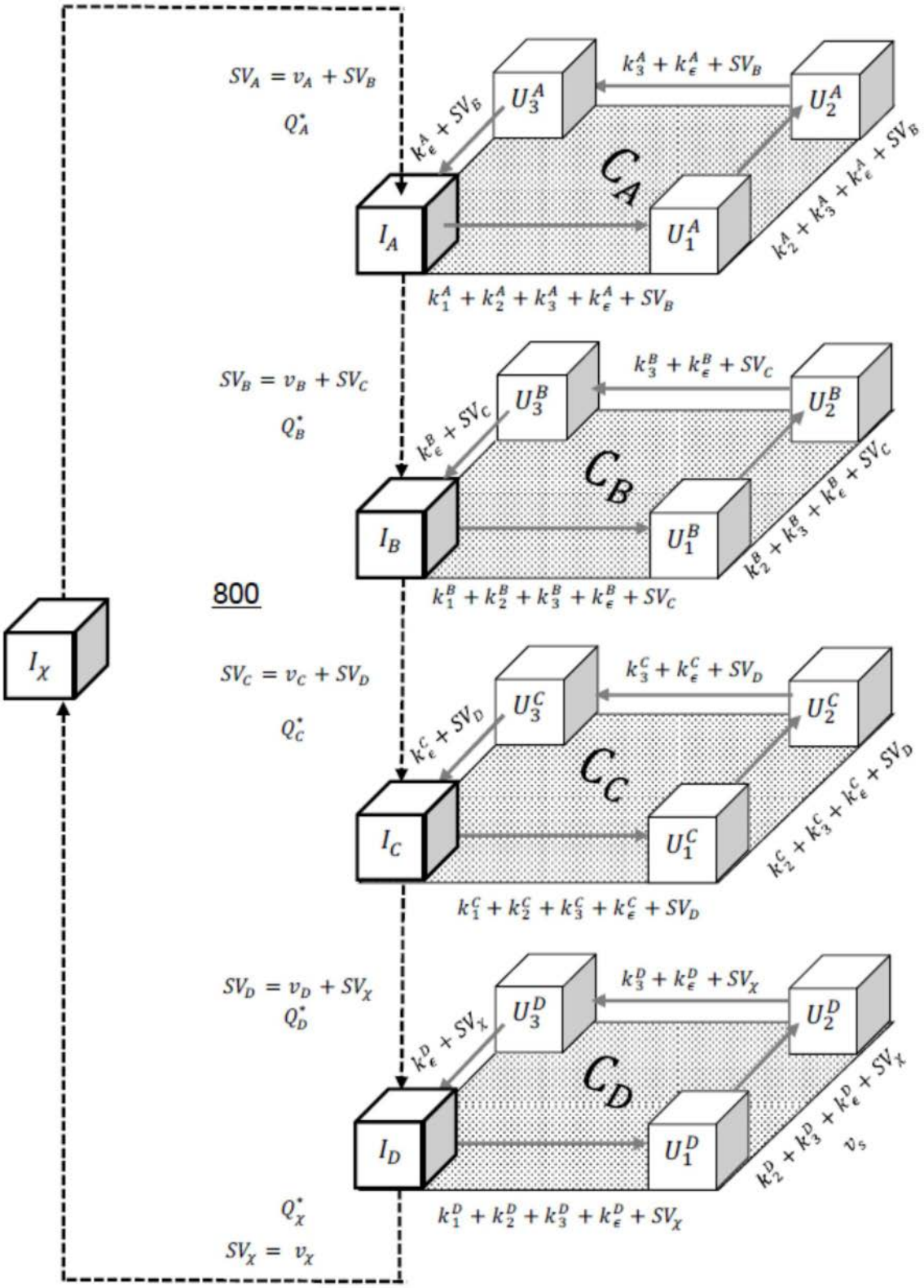

$SV_A = v_A + SV_B$
Q_A^*
$k_\epsilon^A + SV_B$
U_3^A
$k_3^A + k_\epsilon^A + SV_B$
U_2^A
C_A
I_A
U_1^A
$k_2^A + k_3^A + k_\epsilon^A + SV_B$
$k_1^A + k_2^A + k_3^A + k_\epsilon^A + SV_B$
$SV_B = v_B + SV_C$
Q_B^*
$k_\epsilon^B + SV_C$
U_3^B
$k_3^B + k_\epsilon^B + SV_C$
U_2^B
C_B
I_B
U_1^B
$k_2^B + k_3^B + k_\epsilon^B + SV_C$
$k_1^B + k_2^B + k_3^B + k_\epsilon^B + SV_C$
800
I_χ
$SV_C = v_C + SV_D$
Q_C^*
$k_\epsilon^C + SV_D$
U_3^C
$k_3^C + k_\epsilon^C + SV_D$
U_2^C
C_C
I_C
U_1^C
$k_2^C + k_3^C + k_\epsilon^C + SV_D$
$k_1^C + k_2^C + k_3^C + k_\epsilon^C + SV_D$
$SV_D = v_D + SV_\chi$
Q_D^*
$k_\epsilon^D + SV_\chi$
U_3^D
$k_3^D + k_\epsilon^D + SV_\chi$
U_2^D
C_D
I_D
U_1^D
$k_2^D + k_3^D + k_\epsilon^D + SV_\chi$
v_s
Q_χ^*
$k_1^D + k_2^D + k_3^D + k_\epsilon^D + SV_\chi$
$SV_\chi = v_\chi$

图13

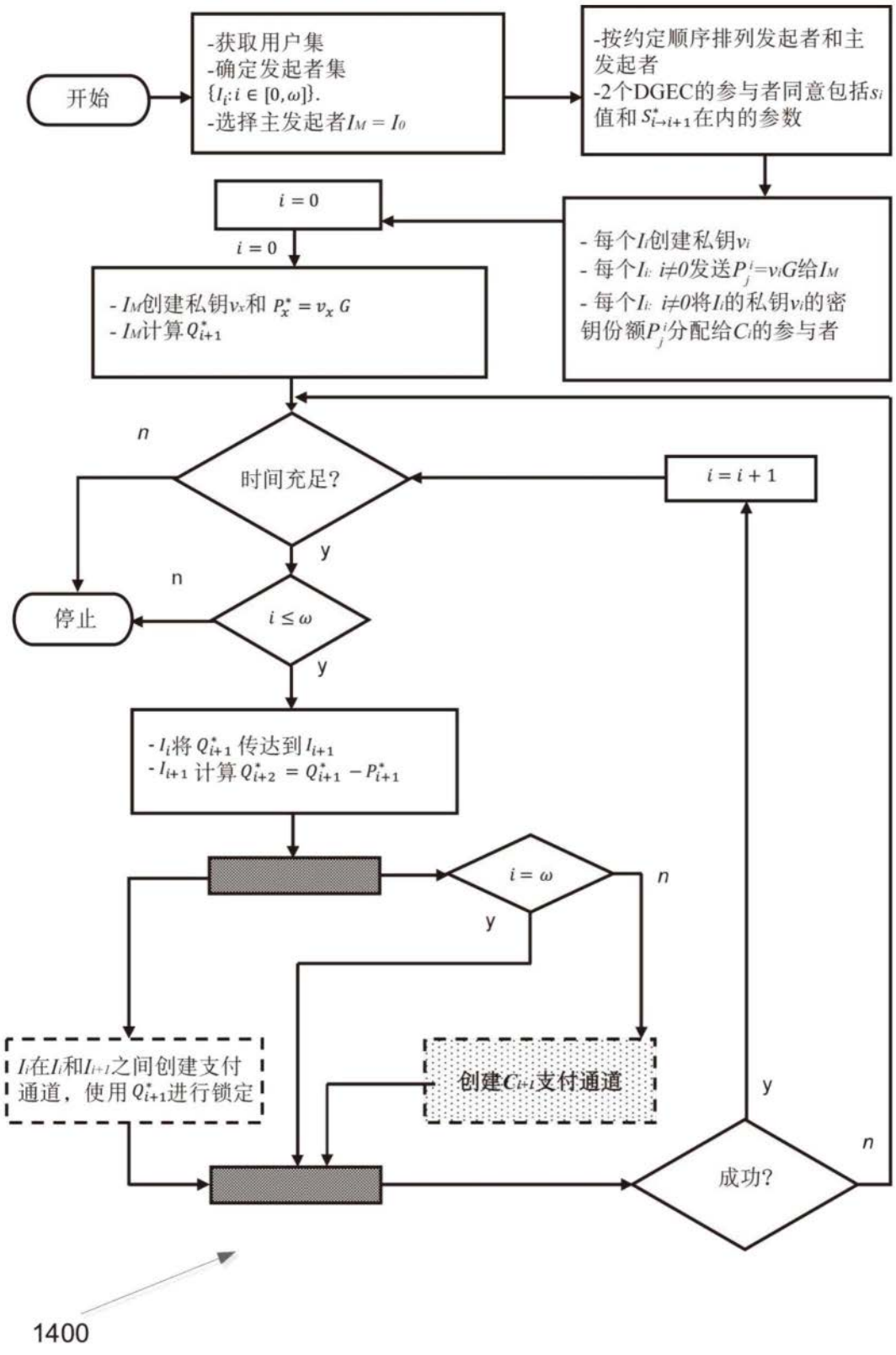

开始
-获取用户集
-确定发起者集
$\{I_i: i \in [0,\omega]\}$.
-选择主发起者$I_M = I_0$
-按约定顺序排列发起者和主发起者
-2个DGEC的参与者同意包括s_i值和$S^*_{i\to i+1}$在内的参数
$i = 0$
$i = 0$
- 每个I_i创建私钥v_i
- 每个I_i: $i\neq 0$发送$P^i_j = v_i G$给I_M
- 每个I_i: $i\neq 0$将I_i的私钥v_i的密钥份额P^i_j分配给C_i的参与者
- I_M创建私钥v_x和 $P^*_x = v_x G$
- I_M计算Q^*_{i+1}
n
时间充足?
$i = i + 1$
y
n
停止
$i \le \omega$
y
- I_i将 Q^*_{i+1} 传达到 I_{i+1}
- I_{i+1} 计算 $Q^*_{i+2} = Q^*_{i+1} - P^*_{i+1}$
$i = \omega$
n
y
I_i在I_i和I_{i+1}之间创建支付通道，使用Q^*_{i+1}进行锁定
创建C_{i+1}支付通道
y
n
成功?
1400

图14

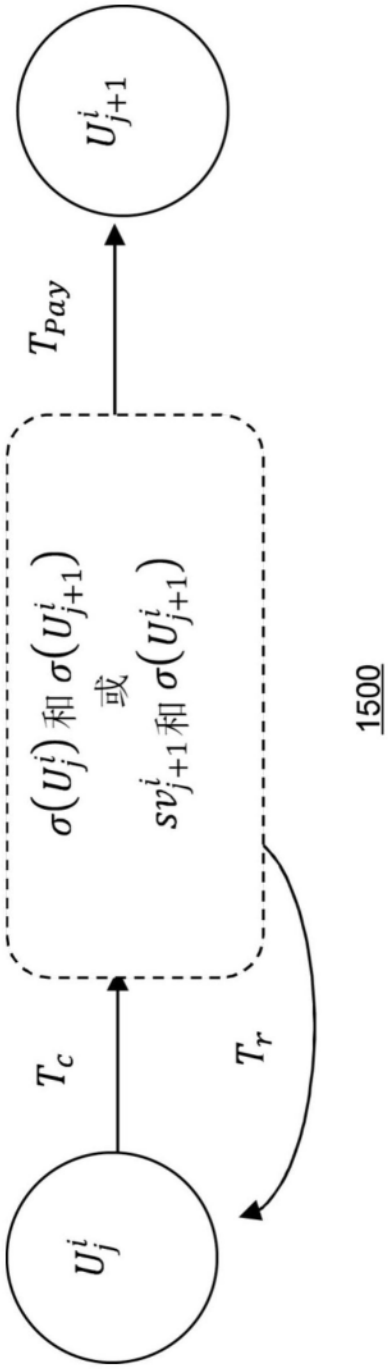

U_{j+1}^{i}
T_{Pay}
$\sigma(U_j^i)$ 和 $\sigma(U_{j+1}^i)$
或
sv_{j+1}^i 和 $\sigma(U_{j+1}^i)$
1500
T_c
T_r
U_j^i

图15

1600	
100	交易ID
Version Number	版本号
<#>	输入数量
<Ref>	前交易输出
<#>	前交易输出索引
Script length	脚本长度
<Script>	脚本签名
<ScriptSignatureFlags>	脚本签名
Sequence number	序号
<#>	输出数量
x XXX	输出值
Output Script Length	输出脚本长度
redeemScript: OP_IF OP_2 <pubkey U_A> <pubkey U_B> OP_2 OP_CHECKMULTISIG OP_ELSE <basepoint G> OP_ECPMULT <Q_B> OP_EQUALVERIFY <pubkey U_B> OP_CHECKSIG OP_ENDIF **scriptPupkey:** OP_HASH160 <$H(redeemScript)$ > OP_EQUAL	输出脚本
LockTime	锁定时间

图16

1700	
<ID>	交易ID
Version Number	版本号
<#>	输入数量
100	前交易ID
0	前交易输出索引
Script length	脚本长度
OP_0 <sig U_A> <sig U_B> OP_TRUE <T_c redeemScript>	脚本签名
<ScriptSignatureFlags>	脚本签名
Sequence number	序号
<#>	输出数量
x XXX	输出值
Output Script Length	输出脚本长度
OP_DUP OP_HASH160 $< H(\text{pubkey } U_A) >$ OP_EQUALVERIFY OP_CHECKSIG	输出脚本
$f(\{s_i : i \in [a, \omega]\})$	锁定时间

图17

1800	
<ID>	交易ID
Version Number	版本号
<#>	输入数量
100	前交易ID
0	前交易输出索引
Script length	脚本长度
<sig U_B> <sv_B> <T_c redeemScript>	脚本签名
<ScriptSignatureFlags>	脚本签名
Sequence number	序号
<#>	输出数量
x XXX	输出值
Output Script Length	输出脚本长度
OP_DUP OP_HASH160 < H(pubkey U_B) > OP_EQUALVERIFY OP_CHECKSIG	输出脚本
LockTime	锁定时间

图18

<scriptSig> T_{Pay}	<scriptPubkey> T_c	Combined (r_hash) **STACKED**	Combined (r_pure) **STACKED**
<sig U_B> <sv_B> <T_c redeemScript>	Redeem Pure <basepoint G> OP_ECPMULT <Q_B> OP_EQUALVERIFY <pubkey U_B> OP_CHECKSIG	OP_EQUAL <$H(redeemScript)$ > OP_HASH160 <T_c redeemScript> <sv_B> <sig U_B>	OP_CHECKSIG <pubkey U_B> OP_EQUALVERIFY <Q_B> OP_ECPMULT <basepoint G> <sv_B> <sig U_B>
	Redeem Hash OP_HASH160 <$H(redeemScript)$ > OP_EQUAL		

1900

图19

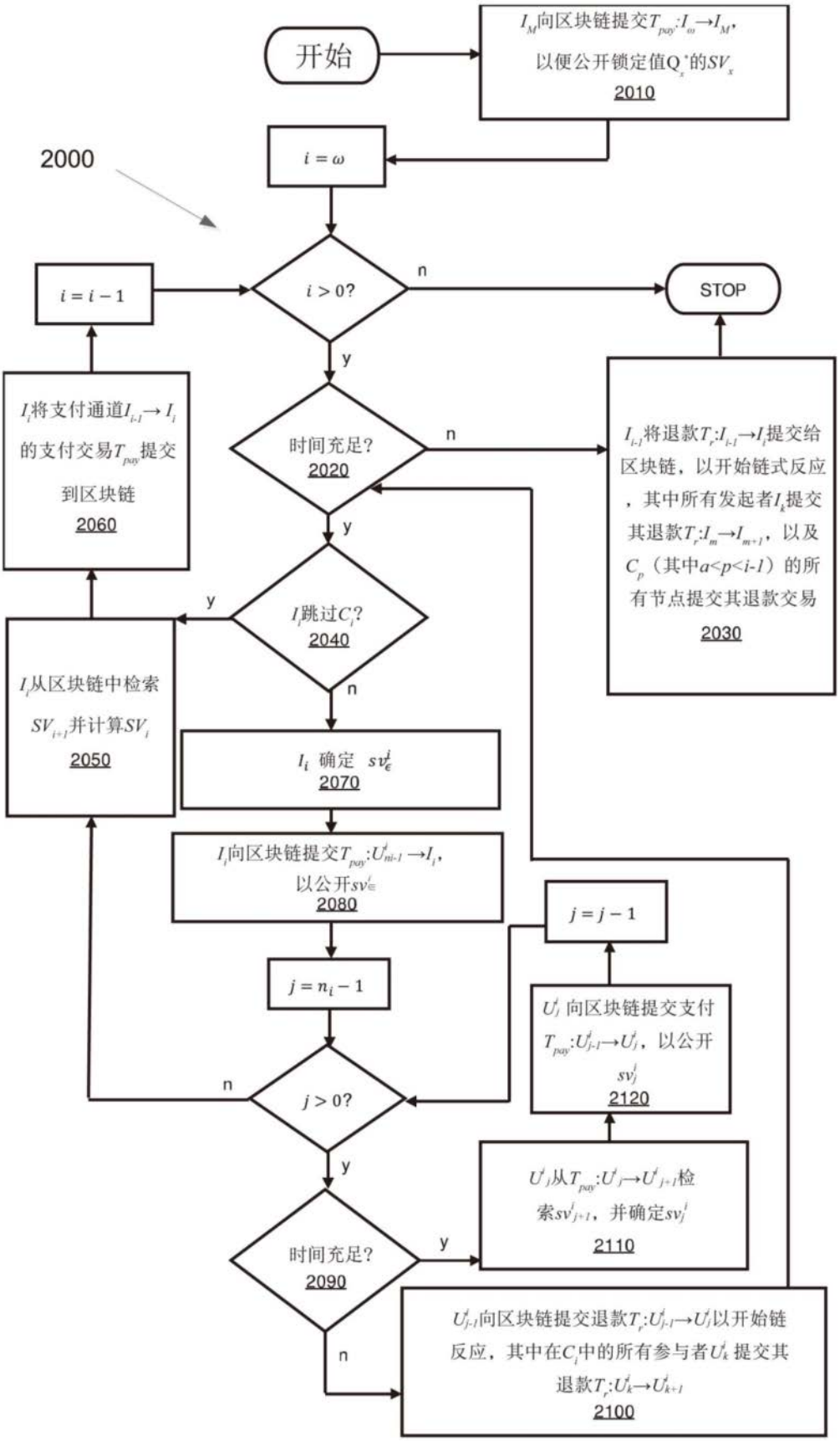

开始
I_M向区块链提交T_{pay}:$I_\omega \rightarrow I_M$，以便公开锁定值Q_x*的SV_x
2010
2000
$i=\omega$
$i>0$?
n
STOP
y
$i=i-1$
I_i将支付通道$I_{i-1} \rightarrow I_i$的支付交易T_{pay}提交到区块链
2060
时间充足?
2020
n
I_{i-1}将退款T_r:$I_{i-1} \rightarrow I_i$提交给区块链，以开始链式反应，其中所有发起者I_k提交其退款T_r:$I_m \rightarrow I_{m+1}$，以及C_p（其中$a<p<i-1$）的所有节点提交其退款交易
2030
y
I_i跳过C_i?
2040
y
I_i从区块链中检索SV_{i+1}并计算SV_i
2050
n
I_i 确定 sv_ϵ^i
2070
I_i向区块链提交T_{pay}:$U_{ni-1}^i \rightarrow I_i$，以公开$sv_\epsilon^i$
2080
$j=j-1$
$j=n_i-1$
U_j^i 向区块链提交支付T_{pay}:$U_{j-1}^i \rightarrow U_j^i$，以公开$sv_j^i$
2120
n
$j>0$?
y
U_j^i从T_{pay}:$U_j^i \rightarrow U_{j+1}^i$检索$sv_{j+1}^i$，并确定$sv_j^i$
2110
时间充足?
2090
y
n
U_{j-1}^i向区块链提交退款T_r:$U_{j-1}^i \rightarrow U_j^i$以开始链反应，其中在$C_i$中的所有参与者$U_k^i$提交其退款$T_r$:$U_k^i \rightarrow U_{k+1}^i$
2100

图20