

(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 孙磊 徐民

(51) Int.Cl.

H04L 9/00 (2022.01)

G06F 21/64 (2013.01)

PCT/EP2022/079830 2022.10.25

W02023/072959 EN 2023.05.04

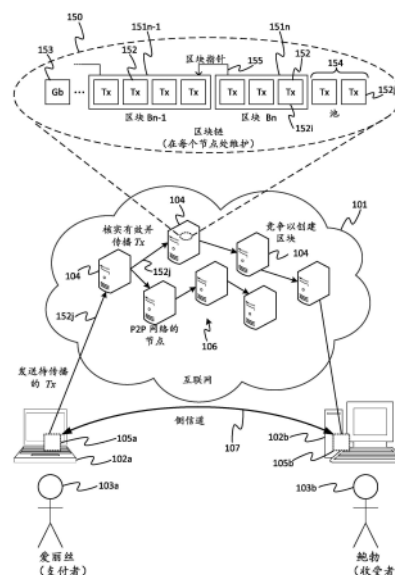
地址 瑞士楚格

(72)发明人 克雷格·史蒂文·赖特

权利要求书2页 说明书34页 附图9页

用于分布式区块链功能的方法和系统

本公开提供了用于对数据记录进行分布式和/或并行处理,并且具体地对区块链区块中的区块链事务进行核实的方法和系统。在优选实施例中,公开了一种分布式核实节点,其中将一个或多个事务分配给多个核实资源中的一个核实资源。所述一个或多个事务涉及用于所述区块的默克尔树的一部分,使得每个核实资源能够在验证所述区块的事务的子集时独立操作,每个子集基于所述默克尔树的段。本公开包括至少用于以下各项的有利技术:将树段分配给不同的核实资源,负载均衡,下载要核实的事务,分布式UTXO池,索引方案,防止双重花费事件。



1. 一种计算机实现的方法,所述方法包括以下步骤:
生成、存储和/或维护第一输出存储库,用于记录、搜索和/或处理多个未花费事务输出,每个未花费事务输出与区块链区块的多个区块链事务 (TxS) 中的事务 (Tx) 相关联;
其中:
所述多个区块链事务提供所述区块链区块的默克尔树的一部分、和/或由用于所述区块链区块的默克尔树的一部分表示。
2. 根据权利要求1所述的方法,所述方法包括:
生成、存储和/或维护至少一个其他输出存储库。
3. 根据权利要求1或2所述的方法,所述方法还包括:
创建和/或维护数据库日志,所述数据库日志包括与所述输出存储库相关的动作、更改和事件的历史记录。
4. 根据前述任一项权利要求所述的方法,其中:
所述第一输出存储库和/或所述至少一个其他输出存储库包括至少一个记录,所述至少一个记录关联于:
 - i) 未花费事务输出;和/或
 - ii) 标识符,所述标识符与a) 未花费事务输出和/或b) 所述多个区块链事务中的事务 (Tx) 相关联。
5. 根据权利要求4所述的方法,其中:
所述至少一个记录包括记录标识符,所述记录标识符具有:
 - i) 区块标识符 (block_ID),所述区块标识符与区块链区块相关联;和/或
 - ii) 事务标识符 (TxID),所述事务标识符与所述多个区块链事务中的事务 (Tx) 相关联。
6. 根据权利要求5所述的方法,其中:
 - i) 所述记录标识符包括所述区块标识符 (block_ID) 和所述事务标识符 (TxID) 的函数;和/或
 - ii) 所述记录标识符包括所述区块标识符 (block_ID) 和所述事务标识符 (TxID) 的级联;和/或
 - iii) 所述多个区块链事务中的所述事务与未花费事务输出相关联。
7. 根据权利要求5或6所述的方法,所述方法还包括以下步骤:
使用所述记录标识符来搜索、标识、访问或插入所述输出存储库中的至少一个记录。
8. 根据前述任一项权利要求所述的方法,其中:
所述多个未花费事务输出中的至少一个未花费事务输出与锁定标志相关联,所述锁定标志:
 - i) 指示所述未花费事务输出是否能够用于花费;和/或
 - ii) 能够在第一状态与第二状态之间配置,所述第一状态指示允许花费所述未花费事务输出,所述第二状态指示禁止花费所述未花费事务输出。
9. 根据权利要求8所述的方法,所述方法包括以下步骤:
 - i) 将所述未花费事务输出与所述锁定标志相关联;和/或
 - ii) 将所述锁定标志的状态从所述第一状态更改为所述第二状态,或者从所述第二状态更改为所述第一状态。

10. 根据权利要求8或9所述的方法,所述方法包括以下步骤:

从第一处理资源向至少一个其他处理资源发送通信,以使所述至少一个其他处理资源将与所述未花费事务输出相关联的所述锁定标志的所述状态从所述第一状态更改为所述第二状态、或者从所述第二状态更改为所述第一状态。

11. 根据权利要求10所述的方法,其中所述通信包括:

- i) 事务 (TX)、事务标识符 (TxID) 和/或事务 (Tx) 的哈希;以及
- ii) 一个或多个未花费事务输出的列表。

12. 根据权利要求10或11所述的方法,所述方法包括以下步骤:

在所述至少一个其他处理资源处接收所述通信;

将所述锁定标志的所述状态从所述第一状态更改为所述第二状态,或者从所述第二状态更改为所述第一状态。

13. 根据前述任一项权利要求所述的方法,其中:

i) 所述默克尔树的所述部分是用于所述区块链区块的所述默克尔树的子部分或段;和/或

ii) 所述多个区块链事务由所述默克尔树的内部节点表示。

14. 一种区块链核实系统,所述系统用于核实区块链区块的至少一部分,所述区块链区块包括多个区块链事务和用于所述区块的默克尔树的根;

其中所述系统包括多个核实资源,每个核实资源包括:

处理器;以及

存储器,包括可执行指令,所述处理器执行所述可执行指令时,使得所述系统执行根据前述任一项权利要求所述的计算机实现的方法。

15. 一种非暂时性计算机可读存储介质,其上存储有可执行指令,计算机系统的处理器执行所述可执行指令时,使得所述计算机系统执行根据权利要求1至13中任一项所述的计算机实现的方法。

用于分布式区块链功能的方法和系统

技术领域

[0001] 本公开大体上涉及用于处理相关或相关联数据记录的改进的方法和系统。本公开特别适合但不限于用于在区块链网络上实现或使用区块链网络实现的传输,例如区块链事务的挖掘前和/或挖掘后核实、SPV检查等。优点包括但不限于提高安全性、弹性和效率,降低速度和资源需求,以及实现现有技术布置不可能实现的新颖核实方法,从而实现先前不可能实现的区块链实现的布置。

背景技术

[0002] 虽然本文可能出于提供用于实现的说明性上下文的目的而提及比特币协议和网络,但是本公开不限于与比特币区块链一起使用,并且替代协议和实现方式(包括基于账户的协议和实现方式以及包括权益证明共识的那些协议和实现方式)也在本公开的范围内。此后,纯粹出于方便目的,术语“UTXO”可以用于指代事务输出,并且不应被解释为意味着本公开的实施例仅限于用于基于UTXO的区块链模型。

[0003] 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分布式系统,该系统由区块组成,而区块又由事务(transaction)组成。每个事务都是一种数据结构,该数据结构对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。每个区块都包含上一区块的哈希,因此区块被链接在一起,以创建自区块链创建以来写入其中的所有事务的永久性且不可更改的记录。

[0004] 为了将事务(Tx)写入区块链,必须对该事务进行核实。网络节点(矿工)进行工作以确保每个事务均有效,而无效事务则被网络拒绝。在一些协议中,安装在这些节点上的软件客户端通过执行其锁定脚本和解锁脚本对未花费事务(UTXO)执行该核实工作。如果锁定脚本和解锁脚本的执行评估为真,则该事务有效,将该事务写入区块链。因此,为了将事务写入区块链,该事务必须:i)由接收该事务的第一节点进行核实,如果该事务通过核实,则该节点将其中继到网络中的其他节点,即该事务被传播;ii)添加到由矿工建造的新区块中;iii)被挖掘,即添加到过去事务的公共分类账中。一旦该事务作为UTXO存储在区块链中,用户就可将相关联加密货币的控制权转移到与随后写入区块链的另一事务中的输入相关联的另一地址。这通常使用数字钱包来完成,该数字钱包存储与用户的加密货币相关联的公钥和私钥对。已知的加密货币钱包有多种形式,包括SPV钱包(简单支付验证)。SPV技术允许用户和商家节点仅基于与特定转移相关的部分信息来执行本地验证。SPV将在下面更详细地讨论。

[0005] 然而,已知尽管核实对于确保安全性、符合给定区块链的相关协议以及防止双重花费利用至关重要,但是应当认识到由于需要下载和存储区块、维护大型UTXO池以及执行用于验证的必要处理任务,此类核实任务可能需要大量资源和时间。许多用户要么无法满足此类要求,要么不愿意满足此类要求,因为他们可能不需要满足此类要求。因此,需要一种更快、更高效的验证模型,该验证模型至少可以应对这些挑战(和其他挑战),而不会损害安全性或需要适配现有协议。现在已设计出这种改进的解决方案。

发明内容

[0006] 本公开的实施例提供了改进的区块链相关方法、设备和系统。根据一种形式的措辞,这些实施例提供了用于核实区块链事务和/或部分或整个区块链区块的解决方案。根据附加或替代形式的措辞,这些实施例提供了用于控制、管理和/或提高区块链事务的已知处理方法的效率、资源需求、速度和/或弹性的安全解决方案。实施例还实现了区块链实现的解决方案的可扩展性,从而提供了用于电子传输数字资源的改进的方法和技术架构。

[0007] 本公开的实施例可以部分地或全部地由各种装置实现。这些装置可以是基于硬件和/或软件的装置,包括(但不限于)一个或多个虚拟机、服务器、基于GPU的计算资源或多处理器系统。附加地或替代地,实施例可以包括一个或多个数字钱包。然而,重要的是,实施例提供了用于对区块链相关核实任务进行分布式处理的机制。已知分布式过程的协调、管理和控制本质上是技术性的,因为这需要全面了解所涉及的硬件与软件组件之间的交互,并且此类分布式解决方案的实现超出了技术上的琐碎范围。

[0008] 实施例可以包括实现或促进跨多个处理资源分布核实任务的解决方案,为了方便起见,将这些处理资源称为“核实器”。核实器可以包括单个处理资源,也可以包括多个相关处理资源,这些相关处理资源可以被共同视为核实资源。

[0009] 当需要核实和/或下载事务区块时,所述区块的默克尔树可以分解为一个或多个较小段,每个段包括树结构,所述树结构具有其自身的根并且表示所述区块中的所述事务的子集。然后,可以将这些段分配给不同的核实器。每个核实器用于对已经分配给它的所述事务子集执行必要的处理任务。可以各种方式将树段分配给核实器,但是根据一个有利的实施例,可以使用二进制索引系统,其中使用随机生成的经过双重哈希处理的默克尔根的前导数位来将给定段分配给具有匹配的二进制标识符的核实器(或核实器组/集群)。这提供了一种用于跨多个核实器进行负载均衡的简单、高效且快速的机制。

[0010] 每个树段可以包括一个小二进制标记,所述小二进制标记使得能够在所述核实器完成其操作之后重建事务的整个默克尔树。段标记允许控制器组件将段重新组装为其原始形式,其中所述标记指示原始树内的位置。这具有以下优点:尽管多个树段跨许多不同的核实器分布,可能分布在全球任何地方,但是它们可以快速且容易地重新组装,以提供用于所述区块的完整默克尔树。

[0011] 在一个或多个实施例中,核实器可以包括或者可以访问存储库(repository),所述存储库记录与其已经执行的工作相关的数据和/或其已经处理的数据。在一个实施例中,这可以包括数据库,所述数据库包括已经分配给给定核实器用于处理的未花费事务输出(UTXO)。应当记住的是,在传统模型中,所述区块链上的所有UTXO都由称为UTXO池的数据库中的节点进行跟踪。每个完整节点都有自己的用于所述区块链的所述UTXO池的完整副本。然而,根据本公开,可以利用不同的方法,其中每个核实器具有其自身的UTXO池,所述UTXO池跟踪已经分配给它以供核实的事务的UTXO。此类分布式UTXO池的优点包括但不限于保证数据完整性,提高速度和效率,以及并入和支持SPV等各种核实技术。

附图说明

[0012] 为了帮助理解本公开的实施例并示出如何实施此类实施例,现将仅通过举例的方式参考附图进行说明,其中:

- [0013] 图1是一种用于实现区块链的系统的示意性框图。
- [0014] 图2示意性地示出了可记录在区块链中的事务的一些示例。
- [0015] 图3提供了本领域已知的一般默克尔树结构的图示。
- [0016] 图4示出了如本领域中已知的可以如何从区块链事务集中导出默克尔根。
- [0017] 图5提供了根据本公开的实施例可以如何将默克尔树划分为子集(或“段”)的示例,然后将该子集(或“段”)分配给相应核实资源。
- [0018] 图6示出了可以如何将默克尔树划分为逻辑段的图5的替代示例。
- [0019] 图7示出了根据本公开的说明性实施例的系统级分布式核实节点。
- [0020] 图8示出了本公开的说明性方法中涉及的步骤的简要流程图。
- [0021] 图9更详细地示出了图7的示例性系统的图示。

具体实施方式

- [0022] 出于说明目的(而非限制目的)并且参考附图,下面描述本公开的示例性实施例。
- [0023] 传统上,区块链网络中的节点维护区块链上所有事务的全局分类账。全局分类账是分布式分类账,并且每个节点可以存储全局分类账的完整或部分副本。影响全局分类账的节点的事务由其他节点验证,从而维护全局分类账的有效性和完整性。本领域普通技术人员应当理解实现和操作区块链网络(例如,使用比特币协议的区块链)的细节。
- [0024] 每个事务通常具有一个或多个输入以及一个或多个输出。嵌入到输入和输出中的脚本指定可以如何以及由谁访问事务的输出。事务的输出可以是由于该事务而将值的控制权转移到地址。然后,该值作为未花费事务输出(UTXO)与该输出地址相关联。然后,后续事务可以将该地址作为输入来引用,以便获得对该值的控制权或所有权。
- [0025] 如上所述,以比特币网络和协议为例,挖掘节点竞相在区块链中创建下一区块。为了组装区块,矿工将该区块构建为来自未确认事务池(“内存池”)的事务集。然后,该矿工尝试完成关于其已经组装的区块的工作证明(PoW)谜题。如果该矿工在接收到关于任何其他矿工已经成功生成其自身的区块并完成其PoW的通知之前设法完成PoW,则该矿工通过将其区块发送到网络上的对等节点来传播该区块。这些节点核实该区块,然后在网络中将其进一步发送到其他节点。如果该矿工在完成自己的PoW之前接收到关于另一区块已经完成的通知,则该矿工就会放弃努力并开始尝试构建下一区块。
- [0026] 因此,快速传播区块有助于避免代表矿工和核实节点浪费努力(和相关联的能量)。通过提供一种能够更快地核实并因此传播区块的解决方案,本发明提供了增强的网络性能。该解决方案减少了所需的计算时间和工作量,从而减少了网络所需的能量。该解决方案提供了一种在资源和时间方面更高效的网络。该解决方案最终提供了一种改进的(区块链)网络。
- [0027] 在区块链的当前实现方式(例如,比特币网络)中,接收到区块的每个节点首先核实该区块,然后再将该区块发送到其他节点。核实区块所需的时间会减慢该区块在网络中的传播速度。应当注意的是,区块链的一些实现方式(包括现有协议的演进)可以仅通过节点子集而不是网络中的每个节点进行区块核实;然而,大多数节点处的区块核实仍然可能是任何区块链实现方式的特征,以防止无效区块通过网络传播。
- [0028] 核实区块涉及确认该区块符合适用区块链协议设定的规定标准。适用于比特币协

议的示例性标准可以包括CheckBlock和CheckBlockHeader等函数。除了确认该区块本身符合规定的标准之外,还可以评估该区块内的每个事务是否符合事务级标准。例如,在比特币协议中应用的事务级标准可以包括AcceptToMemoryPool、CheckTransaction和CheckInputs函数。

[0029] 基于比特币协议的区块级标准的具体示例可以包括:

- [0030] • 区块数据结构在句法上是有效的。
- [0031] • 区块头哈希小于目标难度(强制执行工作证明)。
- [0032] • 区块时间戳在未来少于两小时(考虑到时间误差)。
- [0033] • 区块大小在可接受的范围内。
- [0034] • 第一事务(并且仅是第一事务)是Coinbase生成事务。
- [0035] • 区块内的所有事务都是有效的。

[0036] 基于比特币协议的事务级标准的具体示例可以包括:

- [0037] • 事务的语法和数据结构必须正确。
- [0038] • 输入列表和输出列表都不为空。
- [0039] • 每个输出值 x 以及所有输出的总和必须在 $0 < x < 21 \cdot 10^6$ 的范围内。
- [0040] • 所有输入都不具有空哈希。
- [0041] • $nLockTime$ 小于或等于 INT_MAX 。
- [0042] • 以字节为单位的事务大小大于或等于最小值且小于最大值。
- [0043] • 签名操作次数少于签名操作限制。
- [0044] • 解锁脚本scriptSig只能将数字推送到堆栈上,而锁定脚本scriptPubkey必须与isStandard形式匹配。
- [0045] • 对于每个输入,如果所引用的输出存在于池中的任何其他事务中,则必须拒绝该事务。
- [0046] • 对于每个输入,如果所引用的输出事务是Coinbase输出,则该输出事务必须具有至少COINBASE_MATURITY(100)个确认。
- [0047] • 对于每个输入,所引用的输出必须存在,并且不能已经花费。
- [0048] • 使用所引用的输出事务来获取输入值,检查每个输入值以及总和是否在值 x 的允许范围内,即 $0 < x < 21 \cdot 10^6$ 。
- [0049] • 池中或主分支中的区块中必须存在匹配的事务。
- [0050] • 输入值的总和必须等于或大于输出值的总和。
- [0051] • 事务费用必须足以获得对空区块的输入。
- [0052] • 每个输入的解锁脚本必须对照对应的输出锁定脚本进行核实。

[0053] 这些示例性标准是说明性的,并且不应被解释为对于所有实施例来说都是充分的或必要的,因为规定的标准在不同的协议中可能不同,并且如果对给定协议进行更改,则对于该协议可以随时间而更改。通常,事务级核实标准是指根据适用的区块链协议,事务必须被视为有效的规定特征。类似地,区块级核实标准是指根据适用的区块链协议,区块必须被视为有效的规定特征。

[0054] 根据本申请,描述了加速区块核实以便促进区块在网络中更快传播的方法和设备。更快、更高效的核实和传播有助于应对关于如何扩展区块链网络的技术挑战,并且因此

提供基于此类区块链平台构建的改进的应用程序和系统。

[0055] 在一个方面,本申请描述了一种节点,该节点被构造为通过并行和/或以分布式方式至少执行各个事务的事务级核实来核实区块。然而,可能不会并行评估某些事务级标准。例如,可以在串行的基础上评估UTXO的唯一性。在这种情况下,本公开的分布式核实节点可以被构造为或设置为在将多个事务集分配在两个或更多个并行处理器的集合中以供核实其余事务级标准之前,确认事务的所引用的输入(UTXO)的唯一性。

[0056] 具体地,本公开的实施例提供了用于处理存储在树结构中的相关或相关联数据记录的改进的验证和安全解决方案。该树可以是二叉树或网状(mesh)结构。如本领域已知的,树结构可以分解为较小的树(在本文中可称为树“段”、“子集”或“部分”),其中每个段包括整个树中的数据记录的子集,并且具有其自身的根。有利地,本公开的实施例利用该特征来提供用于跨多个处理资源分布和并行化相关数据记录的处理的方法和系统。

[0057] 在本公开的示例性实施例中,多个数据记录包括区块链事务,这些区块链事务由于形成默克尔树内的节点而相关。该默克尔树具有已经或可以包括在根据区块链协议的事务的区块的区块头中的根,使得该根提供可以跟随到树内的每个叶(即,事务ID(TxID))的路径。尽管在本公开的一些示例中区块链协议是比特币协议、或者从比特币协议导出,但是其他协议也在本公开的范围之内。

[0058] 在本公开示例中,处理多个事务包括核实区块链区块的至少一部分,该区块链区块包括多个区块链事务和用于该区块的默克尔树的根。这些示例是非限制性的,并且本文公开的技术可以用于非区块链相关数据和/或除了核实之外的其他过程。例如,实施例可以用于存储、构造、搜索和/或维护可以在默克尔树中表示的任何类型的数据记录。可以利用数据库和其他已知的存储资源来代替区块链分类账,或者也可以利用区块链分类账。

[0059] 在另一示例性实施例中,处理多个事务包括下载区块链区块的至少一部分,该区块链区块包括多个区块链事务和用于该区块的默克尔树的根。

[0060] 为了完整性起见,并且参考图3和图4,下面讨论默克尔树及其在表示区块链事务的区块中的使用。

[0061] 默克尔树

[0062] 参阅图3,默克尔树是分层数据结构,能够对数据集合进行安全验证。在默克尔树中,树中的每个节点均提供有索引对(i, j),并且表示为N(i, j)。索引i、j只是与树中的特定位置相关的数字标签。

[0063] 默克尔树的一个特征是,其每个节点的构造由以下等式控制:

$$[0064] \quad N(i, j) = \begin{cases} H(D_i) & i = j \\ H(N(i, k) \parallel N(k + 1, j)) & i \neq j \end{cases}$$

[0065] 其中H是加密哈希函数。

[0066] 图3中示出了根据这些等式构建的二进制默克尔树的一个示例。从图中可以看出, $i = j$ 情况对应于叶节点,该叶节点仅仅是数据 D_i 的对应第i分组的哈希。 $i \neq j$ 情况对应于内部节点或父节点,其通过递归地哈希和级联子节点直到找到一个父节点(默克尔根)而生成。

[0067] 例如,节点N(0, 3)由四个数据包 $D_0, \dots, D_3$ 构造为

[0068] $N(0, 3) = H(N(0, 1) \parallel N(2, 3))$

[0069] $= [H(N(0,0) || N(1,1)) || H(N(2,2) || N(3,3))]$

[0070] $= [H(H(D_0) || H(D_1)) || H(H(D_2) || H(D_3))]$ 。

[0071] 树的深度M被定义为树中节点的最低级别,节点的深度m是节点所在的层级。例如, $m_{\text{root}}=0$ 且 $m_{\text{leaf}}=M$,其中在图3中, $M=3$ 。

[0072] 对于比特币和一些其他区块链中的默克尔树,哈希函数是双重SHA256,即应用标准哈希函数SHA-256两次: $H(x) = \text{SHA256}(\text{SHA256}(x))$ 。

[0073] 默克尔树的主要功能是验证某个数据包 D_i 是N个数据包 $\mathcal{D} \in \{D_0, \dots, D_{N-1}\}$ 的列表或集合的成员。该验证机制称为默克尔证明,涉及为给定的数据包 D_i 和默克尔根R获取一组称为默克尔路径的哈希。数据包的默克尔证明只是通过反复哈希和级联的方式重建根R所需的最小哈希列表,通常称为“认证证明”。

[0074] 如果证明者知道所有分组 $D_0, \dots, D_{N-1}$ 及其顺序,则可以简单地执行存在证明。然而,这确实需要比默克尔证明大得多的存储开销,并且需要整个数据集可供证明者使用。

[0075] 使用默克尔证明和使用整个列表的比较如下表所示,其中使用了二进制默克尔树,并假设数据块的数量N正好等于2的整数幂。

[0076] 下表示出了默克尔树中叶节点的数量与默克尔证明所需的哈希数量(默克尔证明)之间的关系。

[0077]						默克尔树
	数据包的数量	32	256	1024	1048576	$N = 2^M$
	存在证明所需的哈希数量	5	8	10	20	$M = \log_2 N$

[0078] 在这个简化场景中(其中数据包的数量等于叶节点的数量),已发现,计算默克尔证明所需的哈希值的数量以对数方式缩放。显然,计算涉及 $\log_2 N$ 个哈希的默克尔证明比存储N个数据哈希并计算显示证明要高效和实用得多。

[0079] 如果给定默克尔根R,则希望证明数据块 D_0 属于由R表示的有序列表 $\mathcal{D} \in \{D_0, \dots, D_{N-1}\}$,可以按如下所述执行默克尔证明:

[0080] i. 从可信源获取默克尔根R。

[0081] ii. 从源获取默克尔证明 Γ 。在这种情况下, Γ 是哈希的集合:

[0082] $\Gamma = \{N(1,1), N(2,3), N(4,7)\}$ 。

[0083] iii. 使用 D_1 和 Γ 按如下所述计算默克尔证明:

[0084] a. 对数据块进行哈希处理以得到:

[0085] $N(0,0) = H(D_0)$ 。

[0086] b. 与 $N(1,1)$ 级联并进行哈希处理以得到:

[0087] $N(0,1) = H(N(0,0) || N(1,1))$ 。

[0088] c. 与 $N(2,3)$ 级联并进行哈希处理以得到:

[0089] $N(0,3) = H(N(0,1) || N(2,3))$ 。

[0090] d. 与 $N(4,7)$ 级联并进行哈希处理以获取根:

[0091] $N(0,7) = H(N(0,3) || N(4,7))$,

[0092] $R' = N(0,7)$ 。

[0093] e. 将计算得出的根 R' 与(i)中获取的根 R 进行比较:

[0094] 1. 如果 $R'=R$, 则确认树中存在 D_0 , 从而确认数据集 $\mathcal{D}$ 。

[0095] 2. 如果 $R' \neq R$, 则证明失败, 无法确认 D_0 是 $\mathcal{D}$ 的成员。

[0096] 这是一种有效机制, 可为作为由默克尔树及其根表示的数据集的一部分的某些数据提供存在证明。例如, 如果数据 D_0 对应于区块链事务, 并且根 R 作为区块头的一部分公开可用, 则可以快速证明该事务包含在该区块中。

[0097] SPV

[0098] 简单支付验证 (SPV) 利用默克尔树的这些特征, 如在中本聪2008年的白皮书“比特币: 一种点对点的电子现金系统”的第8节中首次提出的。在爱丽丝和鲍勃之间的基于SPV的加密货币交换中, 双方使用相同类型的SPV钱包。SPV钱包存储用户的私钥和公钥、未花费的事务和区块头, 这些区块头唯一地标识区块, 以便它们可以位于区块链上。如所解释的, 区块头包括提供整个区块的内容的唯一摘要或指纹的数据字段, 以及提供用于该区块的默克尔根的字段。默克尔根是通过对区块中的成对事务ID (TxID) 一起重复进行哈希处理直到最终产生单个哈希来生成的。默克尔根提供一种高效且安全的机制来验证事务是否是区块的一部分, 因为它允许钱包等用户和商家节点在本地验证特定事务而无需下载整个区块链。这对于不需要或不希望运行完整节点而只需要执行本地化检查以确定特定事务是否在特定区块中的用户 (例如, 希望在他们之间执行转移的商家和客户等各方) 来说是有利的。总之, SPV使得此类用户能够搜索具有给定根的默克尔树, 以检查 (即, 验证) 特定事务是否包括在特定区块链区块中, 而不必下载和存储整个区块链。

[0099] 因此, SPV钱包至少具有以下优点, 即手机和笔记本电脑等电力和存储受限设备能够在比特币生态系统中运行, 因为它只需要确认事务是否已通过验证 (因此称为“简单支付验证”), 而不是按照其他形式的钱包对区块链进行全面检查。由于SPV钱包仅下载区块头而不包括任何事务, 因此这显著地减少了验证所需的存储空间、能量和处理资源。由于下面解释的原因, SPV钱包特别适合与本公开的实施例一起使用, 并且在本文中使用术语“验证”来包括SPV检查。

[0100] 事务的区块

[0101] 图4示意性地示出了区块链区块的示例。每个区块都包含区块头和事务集。其中, 该区块头包括上一区块头的哈希, 即构建当前区块所基于的区块的区块头的哈希。该区块头还包括使用该事务集构建的默克尔树的默克尔根。首先对每个事务进行哈希处理 (例如, 双重哈希处理), 以生成该事务的事务标识符 (TxID)。然后, 将事务标识符用作默克尔树的叶节点。然后, 对成对事务标识符进行级联和哈希处理, 以形成默克尔树的第一内部层级的相应内部节点。然后, 对第一内部层级的成对内部节点进行级联和哈希处理, 以形成默克尔树的第二内部层级的相应内部节点。重复执行对成对内部节点进行级联和哈希处理的过程, 直到仅保留单个哈希: 默克尔根。该默克尔根有时称为区块默克尔根。

[0102] 下面转到本公开的实施例, 具体参考图5、图6和图7。

[0103] 标识区块的默克尔树的段

[0104] 假设特定方 (例如, 爱丽丝) 希望核实一些事务。根据本公开的实施例, 标识至少一个事务子集, 其中该子集形成用于区块的整个默克尔树的段, 并且/或者由用于区块的整个默克尔树的段表示。因此, 可以基于区块的默克尔树在逻辑上将事务区块分割为多个段, 每

个段包括区块的事务的子集,并且每个段具有其自身的根节点(或“根哈希”)。该公共根哈希有时在下文中称为“段哈希”,以将其与整个区块的根哈希区分开来。树段内同一层级(即,最低层级,有时称为“叶层级”或“叶层”)上的事务是同级(siblings)。给定段中的所有事务共享该段的公共根节点。该公共根节点可以属于默克尔树的相邻层级,即紧邻最低层级之上的层级。或者,该公共根节点可以属于较高层级。通常,该公共根节点可以属于默克尔树的最低层级与默克尔根之间的任何层级。

[0105] 基于区块的默克尔树将区块分解为较小的部分提供了显著的技术优点,包括能够快速且高效地在多个核实器之间分配事务。例如,由于比特币协议使用二叉树,因此可以跨多台机器实现二进制分配。通过使用小二进制标记作为段的索引系统,可以快速计算每个段在整个默克尔树中的位置,使得段能够在核实已经完成之后重新组合在一起,从而重建用于区块的完整默克尔树。该二进制索引方法将在下面更详细地讨论。

[0106] 可以使用各种技术来标识段,但是根据一种方法,段的数量可以由系统中可用核实器的数量来确定。例如,在具有四个核实器的系统中,默克尔树可以分割为四个段;如果有八个核实器,则默克尔树可以分解为八个段,依此类推。如图7的控制器702所示,对给定默克尔树的段的标识可以由控制实体执行或影响。

[0107] 参考图5和图6进一步示出了上面解释的要点,其中图5示出了如何将默克尔树划分为要分配给核实器的单独部分502的示例。在图5的示例中,每个箭头表示相应事务,该相应事务被哈希处理以形成相应事务标识符,该相应事务标识符在默克尔树的相应叶节点处使用。默克尔树的顶部是区块默克尔根。在该示例中,由默克尔树表示的事务区块包含32个事务。然而,应当理解的是,这仅仅是说明性示例,并且通常根据区块中事务的数量,默克尔树可以包含任何数量的事务。如图所示,默克尔树被划分为由虚线框指示的四个部分502a至502d。每个部分502由默克尔树的相应公共内部节点(内部哈希)504链接,该相应公共内部节点(内部哈希)由实线圆圈表示。每个部分502表示八个事务。在该示例中,公共内部节点504属于默克尔树的第四层级。根据本文所述的实施例,每个相应部分502(或者更确切地说,形成和/或表示一部分的事务)被分配给相应核实器以供处理,例如,以供核实属于相应部分502的事务。

[0108] 图6示出了如何将默克尔树划分为部分602的另一示例。图6中的默克尔树与图5中的默克尔树相同。现在,在该示例中,将默克尔树划分为八个部分602a至602h,每个部分602表示四个事务。在该示例中,公共内部节点604属于默克尔树的第三层级。相反,可以将图5和图6的默克尔树划分为更多(例如,16个)或更少(例如,2个)部分502、602。通常,可以将由区块的事务集形成的默克尔树划分为任意数量的部分502、602,其中每个部分包括至少两个事务。

[0109] 将段分配给相应核实资源

[0110] 在标识之后,跨多个核实资源分布事务子集,为了便于参考,该多个核实资源也可以称为“核实器”。在图7和图9中,多个核实器被示为资源A至D(704a至704d)。分配过程可由图9所示的组件904等专用单元来引导或影响,但不限于此。

[0111] 每个核实器(704a至704d)可以包括一个或多个处理资源。因此,多个核实器(704a至704d)中的至少一个核实器可以是或包括以下各项中的至少一项:一个或多个虚拟机、一个或多个服务器、一个或多个基于GPU的计算资源、一个或多个线程和/或一个或多个多处

理器系统等。本质上,多个核实器中的任何一个核实器可以由任何一种或多种类型的处理资源或处理资源的组合组成,每个核实器能够核实通过区块的默克尔树的段彼此关联的一个或多个事务。多个核实器(704a至704d)和其他系统组件形成集体资源或实体700,该集体资源或实体将称为“(分布式)核实节点”。

[0112] 优选地,分布包括将每个段分配给多个核实器中的相应核实器。可以至少将核实器设置为:

[0113] • 对一个或多个事务进行操作,该一个或多个事务构成已经分配给该一个或多个事务的一个或多个段;

[0114] • 核实一个或多个事务以验证该一个或多个事务是否符合区块链协议;和/或

[0115] • 核实它们是否可以在区块链分类账或已知、已注册或已花费事务的数据库等现有存储库中标识。

[0116] 核实器的活动以及将子集分配给不同的核实器可以由控制器指导。图7示出了控制器702将相应树段的事务A至D的子集分别分配给核实器704a至704d。系统级控制器702协调分布式核实节点内的系统或设备704a至704d的活动,并且可以控制或影响任务,例如利用区块的默克尔树来标识树段,将所标识的段分配给相应核实器,将所核实的树段重新排序为用于区块的完整默克尔树,和/或对已重建区块内的事务进行排序。

[0117] 这些核实器中的一个或多个核实器可以包括至少一个协调实体,该至少一个协调实体被设置为充当核实器级控制器。因此,核实器704a至704d中的任何一个或全部核实器可以包括其自身的至少一个控制器组件。该较低级控制器可以影响或指导操作,例如将任务或子任务分配给核实器内的一个或多个处理资源,针对给定段重建默克尔树,或者与其他系统组件(例如,其他核实器或较高级控制器、UTXO池、钱包等)交互。反过来,处理资源本身可以进一步分解为较小系统,其中一个或多个较小系统可以包括控制器及其自身的一个或多个处理资源。以这种方式,系统可以包括分层架构,其中通过核实实体来执行段核实,这些核实实体包括用于执行核实任务的一个或多个处理资源,以及用于协调处理器活动和执行组件间通信的一个或多个控制器。

[0118] 在核实器包括多个处理资源的实施例,该核实器可以将其分配的段分割为较小段。然后,该核实器的控制器可以在其控制下的处理器之间分布子段。以这种方式,核实过程可以分层和分布式方式实现。

[0119] 这种分层分解还可以扩展到事务级,使得核实可以进一步分解为每个事务的子过程或任务,而不是在树段级。通过这种方法,将对一个或多个单独事务的核实分解为跨不同机器分布的子任务,或者在相同或不同机器上运行的不同线程。这些过程可以排队,使得当一个线程变得可用时,向该线程分配另一事务或任务。

[0120] 因此,本公开使得能够同时处理许多事务,唯一的限制是可用于形成分布式核实节点的硬件的数量,而不是如传统技术那样成为瓶颈的可用处理速度的数量。这使得区块链处理系统能够在不需要改变区块链网络的底层协议的情况下进行水平扩展。

[0121] 因此,本公开与传统核实方法存在显著偏差,该传统核实方法在下面标题为“用于实现本公开的说明性实施例的示例性技术环境”的部分并且参考图1和图2更详细地描述。如所解释的,传统方法涉及将一个区块作为整个实体进行核实,而将核实节点(参见图1的104)的传统视图作为单个计算单元。相反,本公开的实施例将默克尔树分解为多个段,这些

段被提供给不同的核实器(图7和图9中的704a至704d),每个段和核实器能够进一步分解以提高所涉及的分布程度。

[0122] 此外,通过基于每个区块的默克尔树将每个区块分解为段,本公开的实施例使得核实器能够访问、下载和处理区块的小部分,而不是整个区块。应当记住的是,每个段中的事务(成对地)哈希到单个根值。这意味着可以仅使用必要的相关事务来核实该段,而不是整体下载、存储和处理的整个区块。由于比特币SV等协议允许扩展区块大小,并在分类账中包含较大区块,因此下载整个区块的传统模型成为瓶颈。本公开的实施例通过使得各个核实器能够仅接收和处理与其相关的(较小)部分来克服对区块链可扩展性的这一挑战。这会产生更快的总体核实时间、改进的区块链网络以及在区块链上运行的改进的应用程序。

[0123] 此外,实施例支持并促进SPV过程和资源的使用,因为此类SPV仅涉及对给定方感兴趣的默克尔树的部分进行本地核实。因此,SPV技术的树修剪性质非常适合与本公开的实施例结合使用。在SPV上下文中,可以仅向核实器提供其需要的区块数据的部分,即区块头或段根节点和相关事务。

[0124] 当每个核实器已经执行其检查并确认其已经处理的段的有效性时,由于用于生成树的哈希机制,可以保证区块是有效的。

[0125] 跨多个核实器进行负载均衡

[0126] 负载均衡技术和系统是本领域已知的,其目的是跨多个资源均匀地分配任务,以便提高效率。这样做的目的是最大限度地降低一些处理资源处于空闲状态,而其他处理资源过载,从而导致性能下降甚至故障的风险。因此,负载均衡在确保整个系统的弹性以及其性能和效率方面变得至关重要。本公开的实施例可以利用任何已知的负载均衡技术,例如静态或动态负载均衡。附加地或替代地,可以有利地使用本文公开的负载均衡方法。

[0127] 如上所述,本公开的实施例可以在将区块段分配给相应核实器时使用索引系统。优选地,该索引系统是二进制索引系统。在该优选系统中,每个核实器被指定为二进制标签或标识符。假设每个标识符的长度为4位,第一核实器被标识为0000,下一核实器被标识为0001,后一核实器被标识为0010,依此类推。显然,4位标识符允许256个核实器ID,最后一个核实器被标识为1111(即,以十进制表示的核实器编号255)。

[0128] 当需要将树段分配给核实器时,其双重哈希(即,该树段的段哈希)的前4位可以用于确定哪个核实器将处理该段。应当记住的是,默克尔根是通过以下方式生成的,即对区块中的成对事务ID(TxID)一起进行哈希处理以生成默克尔树的相应内部节点(或内部哈希),然后对相邻内部哈希重复进行哈希处理直到最终产生单个哈希。该经过双重哈希处理的默克尔根提供了一种高效、快速且安全的验证机制。在本公开上下文中,这还具有双重哈希生成随机二进制数的优点。每个内部哈希(包括每个段哈希)本身都是双重哈希。因此,可以将段哈希的前x个前导数位作为分配索引。具有四个前导零的哈希将导致树段被分配给ID为0000的核实器,而具有前导数位0001的哈希将导致分配给ID为0001的核实器,依此类推。随机生成双重哈希确保将树段随机分布给核实器。

[0129] 尽管在生成默克尔树时通常使用双重哈希,但是并不是在所有示例中都必须使用双重哈希,而是可以仅使用单一哈希。事实上,任何数量的哈希操作都将产生随机二进制数。负载均衡任务可以由专用系统组件执行,如图9中的905所示,或者可以在系统700内的其他地方提供,或者与系统700相关联并进行通信。

[0130] 分布式接收/下载区块

[0131] 根据一些实施例,将区块默克尔树的段分配给不同核实器可以用于提供更快、更高效的过程,用于下载或以其他方式接收事务区块的部分或全部。

[0132] 在一些实施例中,一个或多个核实器可以发送到核实器订阅的多播地址的一个或多个数据包的形式接收数据。该多播地址可以是本领域已知的IPv6多播地址。这些核实器中的一个、一些或全部核实器可以订阅该多播地址。在一些实施例中,核实器的子集可以订阅不同的相应多播地址,使得可以将用于不同段的事务分配给共享公共多播地址的多组核实器。

[0133] 为每个核实器分配默克尔树的段,例如,基于上述分配索引来为每个核实器分配默克尔树的段。然后,任何给定的核实器用于下载事务集,该事务集形成所分配的树段。这可能涉及从区块链本身(例如,从区块链节点)或从不同的资源或实体(例如,第三方服务提供商)下载事务集。可以将事务集下载到核实器的内部存储器,也可以将事务集下载到共享存储位置,例如云中的共享驱动器。

[0134] 分布式节点可能需要完整区块,即形成区块的整个事务集。在这种情况下,分配有树段的每个核实器接收(例如,下载)形成该段的事务子集。在其他情况下,分布式节点可能只需要区块的某些部分。在这种情况下,只有一些核实器可能需要下载其相应事务子集,以便获得所需事务。

[0135] 以这种方式下载区块(或部分区块)会提高整体下载速度,因为每个核实器仅处理形成该区块的整个事务集的事务子集。这与传统区块下载形成对比,在传统区块下载中,给定实体(例如,完整节点)必须接收或下载整个区块,例如,通过按照每个事务在区块中的出现顺序下载每个事务。现在,该区块由多个核实器并行下载。区块可以包含成千上万个事务,也许会多出几个数量级。单个实体下载该数量的事务将消耗大量资源并花费大量时间。现在,计算负担分布在核实器之间,使得每个单独的核实器消耗一小部分处理资源。类似地,这会减少下载区块的总时间。

[0136] 如所讨论的,每个核实器可以下载事务子集。然后,可以组合这些子集,以便在单个存储位置中重建区块。“单个存储位置”是指作为独立实体的存储资源或形成集体实体的多个相关联的存储资源。为此,各个核实器可以将其相应子集传输到分布式节点的中央控制器,该中央控制器被配置为以正确的顺序排列事务。段哈希(即,链接树段的哈希)可以用于该目的。例如,可以维护段哈希到其在默克尔树中的位置的映射,例如从左到右(当段哈希出现在默克尔树中时)。然后,可以基于对应的段哈希来按顺序(例如,从第一个到最后一个)放置事务子集。

[0137] 在一些实施例中,各个核实器(或作为整体的分布式节点)可以通过重建默克尔树来确认已经下载正确的事务(或已经正确下载事务)。在下载事务子集之后,核实器可基于这些事务来生成候选段哈希。通过对成对TxID进行哈希处理以生成相应内部哈希,并对成对内部哈希重复进行哈希处理直到产生候选段哈希,来构建候选段哈希。候选段哈希所属的默克尔树的层级将取决于默克尔树被划分成的树段的数量。该核实器可以验证候选段哈希是默克尔树的哈希。如果哈希不匹配,则说明下载过程中发生了错误。在一些示例中,每个核实器可以生成候选段哈希,并将其发送到控制器以执行验证。又如,可以基于下载的整个事务集来生成候选区块默克尔根。同样,如果区块已经正确下载,则候选默克尔根应该与

实际区块默克尔根(即存储在区块中的默克尔根)匹配。

[0138] 在一些情况下,核实器可以使用上述技术来核实所下载的事务。也就是说,每个核实器分配有树段,下载对应的事务子集,并核实这些事务。在其他情况下,核实器可能不一定核实事务,并且可以简单地下载事务以供稍后使用,例如,用于发送给第三方。

[0139] 分布式UTXO池

[0140] 优选地,形成分布式核实节点的一部分的每个核实器704具有其自身的存储库(池),用于生成、存储和/或维护未花费事务输出(UTXO)。该存储库(池)充当UTXO池,该UTXO池提供与区块链事务相关联的未消费(即,未花费)输出的记录。因此,每个核实器的UTXO池基于控制器相对于默克尔树段分配给它的事务,并且根据这些事务构建。在一个实施例中,这可以是(图)数据库,其包括与已经分配给给定核实器以供处理的事务的未花费UTXO相关的数据。将新的默克尔树段分配给核实器时,为核实器意识到的每个UTXO在数据库中创建记录。因此,从分布式核实节点的角度来看,UTXO池不是单个池,而是由多个不同的UTXO池组成,每个不同的UTXO池在不同的核实器处或之上提供并且包括不同的UTXO集。因此,用于该节点的UTXO池在数据以及存储和/或处理数据的资源方面都是分布式的。

[0141] 这与传统UTXO模型有很大不同,在传统UTXO模型中,网络中的每个完整节点具有UTXO池的副本,该UTXO池跟踪区块链上的所有UTXO。相比之下,本公开跨多个核实资源分布UTXO池,每个核实资源具有作为区块链的整个UTXO集的子集的UTXO池。每个核实器的UTXO池包括事务的UTXO,这些事务构成负责核实的默克尔树子部分。

[0142] 根据这种方法,每次需要核实新区块时,可以类似于SQL事务日志的方式实现该新区块,因为与数据库相关的每个命令、事件和项目都记录在该日志中。本文将使用术语“数据库日志”,以避免因使用术语“事务”而引起的混淆,如关于区块链已知的,但是使用术语“数据库日志”来包括“事务处理日志”、“事务日志”等术语。本质上,数据库日志可以被解释为由数据库管理系统执行的动作的历史记录,从而提供关于数据库状态已经发生的所有更改的记录,如基于计算机的数据库领域已知的(参见https://en.wikipedia.org/wiki/Transaction_log)。

[0143] 使用有序的历史数据库日志意味着可以通过按日志的原始顺序执行日志的历史记录来构建整个UTXO池。有利地,这确保在需要时总是可以(重新)生成数据库的副本,并且不需要存储数据的单独副本。这确保数据完整性,并且需要更少的存储资源。每个UTXO池可以单独存储、维护和处理。同样有利的是,如果SPV技术在默克尔树的修剪部分上操作,则SPV技术可以促进为每个核实器创建单独的UTXO数据库。

[0144] 数据库内的事务(TX)可以各种方式来构造,尽管特别有利的方法是根据包括区块ID和事务ID的级联(block_ID || TxID)的标识符来构造事务。区块ID和事务ID都是256位哈希,从而产生安全且无冲突的512位级联字段结构。

[0145] 以这种方式构造事务提供了一种快速、高效的查找机制。可以按block_ID对事务进行排序,使得具有相同block_ID的所有事务一起位于数据库中。因此,当核实器需要事务时(例如,以检查该事务的UTXO是否已经花费),该核实器可以首先通过搜索对应的block_ID,然后搜索对应的TxID来定位数据库中的事务。这样做的效果是搜索仅限于数据库的相关部分。这种效率减少了搜索操作所需的时间、处理资源和能量,从而实现与现有技术相比的显著改进。

[0146] 标志或标记与核实器池中的每个UTX0相关联,并指示UTX0是锁定还是解锁。为了方便起见,可以将该标志或标记称为“锁定标志”。当UTX0被标记为“锁定”时,这向组中的核实器(即,分布式核实节点中的其他地方)指示该UTX0不可用于花费。相反,当UTX0被标记为“解锁”时,这向核实器指示该UTX0可以花费。因此,它的作用是使得已被分配来验证事务花费UTX0的核实器能够向其对等方发信号通知该事务花费UTX0,假设该事务被证明是有效的,该事务已被赎回并且因此不再是可花费的。“锁定”状态意味着允许花费,而“解锁”状态意味着禁止花费。

[0147] 该锁定/解锁标志可以是简单的小二进制标记,例如0表示“锁定”,而“1”表示解锁。标记机制由分布式节点系统中的核实器在内部使用,在与区块链交互之前从事务中移除标记,使得事务符合协议规则。

[0148] 在使用中,核实器检查控制器分配给它的每个新事务中的输出。任何未花费输出(UTX0)被添加到核实器的UTX0池,即被记录为UTX0数据库中的条目。在每个新UTX0的相关数据库记录中,锁定标志被设置为“解锁”。

[0149] 当核实器发现新分配的事务正在花费UTX0时,该核实器会向多个核实器中的每个其他核实器发送消息,以向这些核实器通知该UTX0也应该锁定在其相应池中。本质上,该核实器向其对等方发送通信,指示该核实器在特定时间已经发现涉及具有特定哈希ID的事务的花费。其他核实器不需要接收整个事务的完整数据,因为事务哈希和其花费的UTX0列表足以让它们标识相关事务并将其标记为锁定在其自身的数据库中。在接收到消息时,每个接收核实器检查相关UTX0是否在其UTX0池中。如果是,则锁定标志的状态被更改为“锁定”。因此,该锁定防止核实器允许在后续事务中花费相同的UTX0。在新事务确实尝试花费相同UTX0的情况下,检查锁定标志将指示忽略第二花费尝试。如果发送消息的核实器确定核实失败,因此UTX0尚未花费,则可以向核实器对等方发送另一消息以达到这种效果,从而指示应该将UTX0的锁定标志更改为“解锁”状态。一旦已经完成有效花费,就可以发送消息以达到这种效果,并且可以从相关UTX0池中删除锁定的UTX0。

[0150] 在上述实施例中,每个核实器具有单个UTX0池,该UTX0池包括已经分配给它的所有树段中的所有事务的UTX0。然而,通过替代方法,由每个核实器维护的UTX0池可以划分/分割/分隔为多个子池/由多个子池组成,每个区块一个子池。通过这种方式,可以将单个UTX0池组织成逻辑层级结构。通过另一种方法,可以与相应多个UTX0池相关联地设置一个或多个核实器,该多个UTX0池中的每个UTX0池与用于一个或多个树段的集合的UTX0相关。因此,在一些实施例中,一个或多个核实器可以针对不同的单个树段或根据诸如树段类型或在给定范围内的树段等的某些预定义标准将UTX0组织成单独的UTX0池。在此类实施例中,标识符可以包括可以用于将搜索范围缩小到相关UTX0池的区块ID,然后该搜索可以在该池内进行,以(尝试)通过其TxID来标识相关事务。本领域技术人员应当理解的是,在一些实施例中,可以使用这些方法的混合,即分布式节点内的一个或多个核实器可以使用单个UTX0池方法,而其他一个或多个核实器被设置为使用多个单独的UTX0池和/或被组织成子池的UTX0池,或者它们的任意组合。

[0151] 这可以防止“双重花费”情况,即一方尝试花费同一UTX0两次。这提供了一种简单且安全的锁定机制,无论系统中核实器的数量或位置如何,该锁定机制可以高效、快速地运行,并保持在区块链上实现的传输的安全性和完整性。

[0152] 可能实施例的说明性系统

[0153] 图7和图9示出了用于实现所描述的实施例中的至少一些实施例的示例性系统700。图8示出了可以在本公开的方法(的简要视图)中采取的示例性步骤的流程图。

[0154] 系统700可以是封闭系统,因为该系统可以与组织相关联并且形成更大专有系统的一部分。在这种情况下,可以从该组织的较广泛系统内的其他组件接收其数据(例如,事务),并将其结果和输出发送到内部目的地。附加地或替代地,系统700可以被设置为与各种实体对接,其中这些实体中的一些或全部实体可以位于组织外部。在这种情况下,系统700可以被设置为提供核实功能作为服务。例如,系统700可以被设置为与区块链网络交互,以获得其需要的数据。附加地或替代地,该系统可以与希望使用其核实服务的实体交互。因此,系统700的活动可以仅仅是关于特定组织或实体的内部活动,也可以开放与外部实体的交互以向其他方提供核实服务,或者这两者的组合。其他内部或外部实体之间的通信可以由一个或多个接口或通信组件协调,如图9中的902所示。

[0155] 如图7和图9所示,系统700包括控制实体702(或简称为“控制器”)和多个核实资源704(在本文中还可简称为“核实器”)。图7仅示出了四个核实器701a至701d,但是系统700通常可以包括任意数量的核实器。此外,控制器702在图7和图9中示为与核实器704不同,但是不排除控制器702可以包括核实器704中的一个核实器,也可以由核实器704中的一个核实器组成。如上所述,每个核实器可以包括一个或多个处理资源,并且可以包括其自身的用于协调其自身的内部活动的控制器。可以这种方式实现的分层层级没有技术或逻辑限制。然而,为了简单起见,同时为了便于理解,图7仅示出了这种层级结构的一个(顶层)层级。

[0156] 如图7所示,控制器702获得事务集。可以通过电子信道或网络从发送资源接收事务。发送者可以是希望如上所述在系统组织内部或外部执行某种核实检查的任何实体。例如,这可以是区块链网络上的完整节点,例如图1中的节点104,或者数字钱包,或者希望执行与各方之间进行的区块链实现的转移相关的本地检查的商家/SPV节点。一个或多个接口902可以促进系统700与该系统外部源之间的数据传输。

[0157] 事务形成或者可以形成事务区块。事务可以从单个资源(例如,从区块链的区块)获得,也可以从不同资源(例如,一个或多个用户、一个或多个区块链节点等)获得。事务可以在它们已经在区块链上发布之前(即,在记录在区块中之前)获得。或者,事务可以在已经记录在区块链上之后获得。

[0158] 控制器702向每个核实器704分配相应事务子集,如本文所述。每个事务子集形成基于整个事务集生成的默克尔树的相应部分的至少一部分,并且由默克尔树的相应公共内部节点链接。在图7的示例中,将事务子集A分配给核实器A,将事务子集B分配给核实器B,将事务子集C分配给核实器C,并且将事务子集D分配给核实器D。在已经分配事务子集之后,核实器704随后处理其相应子集。在一些实施例中,这涉及每个核实器704核实其相应事务子集。为此,控制器702可以将相关事务传输到相应核实器704。核实器704可以传达回控制器704,以指示其相应事务子集中的每个事务都是有效的,或者至少一个事务是无效的。

[0159] 核实器704a至704d中的至少一个核实器,但优选地一些或全部核实器可以访问其自身的UTXO池,这些池在图9中示为901a至901d。该池包括上述数据库等存储设施,并且可能具有包括区块ID和事务ID的级联的有利索引结构。在图9中,这些池被示为包括在相应核实器内,但是本领域技术人员将容易理解,这些池也可以/替代地通过以下方式提供,即在

核实器外部但与核实器通信。

[0160] 在一个或多个实施例中,所公开的过程可以包括区块级核实阶段,在该阶段期间对照区块级标准测试传入的新区块。示例性区块级标准如上所述,并且通常涉及适用于区块本身(而不是区块内的事务)的规定格式要求和特征或限制。示例包括区块大小、区块头结构或内容以及类似标准。此类操作可以由控制器或控制器的组件执行,也可以由另一系统组件执行。

[0161] 在一些实施例中,该方法还可以包括UTXO唯一性确认模块,该模块用于评估对新区块中的事务的每个输入(即每个UTXO)是否是唯一的。如果同一UTXO在新区块中作为输入多次出现,这表明潜在的双重花费问题,并违反UTXO唯一性标准。如果UTXO唯一性确认模块标识在新区块中的事务输入中被多次引用的UTXO,则它可以输出错误信号或其他中断以指示该区块将被拒绝。

[0162] 假设新区块未被拒绝,即所有UTXO输入都是唯一的,则可以标识默克尔树段,并且在核实器集中分配这些默克尔树段的相关联事务。标识过程可以由如图9所示的段标识单元903等组件执行。分配过程可以由段分配单元(在图9中示为904)执行。分配单元904可以采用多种可能的分配方案中的任何一种分配方案来在各个核实器之间分布区块段,但是在有利的方法中,该分配方案可以如上所述的负载均衡为目标。分配单元904可以包括负载均衡单元905(或与该负载均衡单元通信)。尽管这在图9中示为系统的单独的相关联组件,但是在其他实施例中,负载均衡单元可以是分配单元904的一部分,或者相对于控制器702可以是分离的。可以容易地采用组件的任何组合。

[0163] 各个核实器对照事务级核实标准来核实与它们接收到的一个或多个段相关的事务。核实器之间不需要同步范例,因为它们各自独立地验证分配给它们的事务是否有效。每个核实器输出确认其分配的事务的有效性的结果。将结果相加或累加,以确认段中的所有事务都是有效的。如果这些核实器中的一个核实器标识不合规事务(即无效事务),则该核实器可以发出输出,例如中断或其他信号,以指示存在无效事务。该中断或信号可以被发送到其他核实器或者控制器或另一系统组件,使得它们可以立即停止测试其相应事务,并且不会在核实区块内要被拒绝的事务时浪费更多资源。

[0164] 在一些示例中,该系统可以被设置为检查区块级标准。这可以在将段分配给核实器之前执行,尽管应当理解,区块级核实阶段可以在核实器的事务级核实测试之后发生,或者在某些情况下与事务级核实测试并行发生。

[0165] 现在将参考图8,该附图以流程图的形式示出了核实区块的方法的一个示例。该区块包含多个事务,每个事务引用一个或多个输入,并且每个输入是UTXO(除了Coinbase生成事务的情况)。该方法使用区块链网络上节点内的适当硬件和处理器可执行指令来实现。

[0166] 在操作中,分布式核实节点700在步骤S801中接收新区块数据。这可以是整个区块,或者在SPV相关核实的情况下,这可以仅包括执行SPV检查所需的部分数据。为了方便起见,将该数据称为“区块”。要核实的新区块可以从区块链网络上生成该新区块并完成工作证明的挖掘节点接收,可以从希望执行(SPV)检查的商家节点接收,也可以从SPV钱包等钱包接收。该新区块可以从网络中的另一(非挖掘)节点接收。在一些示例中,分布式核实节点700在将区块转发到网络中的任何其他节点之前核实该区块。如上所述,核实新区块可以包括确认该区块满足某些基于协议的标准,和/或可以在给定实现方式内指定和要求的其他

标准。

[0167] 在步骤S802中,系统700标识区块的默克尔树的块。在S803中,将段分布到多个核实器;并且在S804中,核实器基本上并行且彼此独立地处理其相应事务子集。在S805中,核实器向控制器发信号通知核实是成功还是失败。

[0168] 应当注意的是,当结合本文中的并行处理器的描述使用时,术语“处理器”不一定表示物理上不同的微处理器,并且可以包括任何硬件或软件实现方式,该硬件或软件实现方式使得并行处理资源能够独立且并行地执行处理器功能。并行处理器可以包括一个具有多个核的处理器。在一些情况下,并行处理器可以包括多个单独的处理单元。并行处理器可以共享也可以不共享物理存储器。无论如何实现,每个并行处理器都具有用于发信号的软件或硬件机制,以便响应于标识无效事务而输出信号。并行处理器的实现方式还包括在软件和/或硬件中提供必要的数据传输机制,以将所分配的事务数据路由到各个处理器以供本地处理。

[0169] 列举条款

[0170] 出于说明目的但不限于说明目的,在以下列举条款中提供了本公开实施例。

[0171] 下面关于一组列举条款或本公开的各方面提及的特征并不旨在受限于此方面,并且关于一组条款提及的任何一个或多个特征可以并入其他组条款中的一组或多组条款。

[0172] 条款集1:

[0173] 条款1.1.一种计算机实现的方法,用于处理(例如,核实)区块链区块的至少一部分,所述区块链区块包括多个区块链事务和用于所述区块的默克尔树的根。

[0174] 所述方法可以包括以下一个或多个步骤:

[0175] 将所述区块链事务的相应子集分配给多个处理(例如,核实)资源,其中每个相应子集提供所述默克尔树的相应部分并由所述默克尔树的相应内部节点表示;以及/或者

[0176] 使用所述多个处理(例如,核实)资源来处理(例如,核实)其相应区块链事务子集。

[0177] 附加地或替代地,所述方法可以包括:

[0178] 发送或接收区块链区块的至少一部分,所述区块链区块包括多个区块链事务和用于所述区块的默克尔树的根。所述区块链区块的所述部分可以从发送资源发送到接收资源。所述区块链区块的所述部分可以通过网络(例如,互联网)来发送。所述方法可以包括以下步骤:在所述接收资源处对所述区块链区块的所述部分进行处理。所述接收资源可以称为处理或核实资源。所述区块链区块的所述部分可以使用IPv6多播传输来发送。所述多个处理资源中的至少一个、一些或全部处理资源可以是IPv6多播组的成员。所述发送资源可以包括网络设备。所述网络设备可以用于将IPv6多播通信发送到多播地址。可以在所述网络设备处启用MLD窥探。这在所述网络上的流量方面提供了效率,因为所述发送资源可以选择性地将所述区块链区块的部分发送到特定接收资源(例如,所述多个处理资源内的资源)。通过将数据包发送到所述网络上的所有资源,包括不需要或不希望接收数据的那些资源,可以减少网络流量,并且不会浪费能量和处理资源。这还提高了安全性,因为这避免了拒绝服务(DOS)攻击的可能性,并且这还通过允许由于网络拥塞程度较低而导致的较快事务吞吐量来促进所述区块链网络的可扩展性。

[0179] 术语“核实器”可以与“核实资源”互换使用。所述多个核实资源可以形成或提供分布式核实节点。所述多个核实资源中的至少一个核实资源可以包括一个或多个处理资源。

附加地或替代地,所述多个核实资源中的一个或多个核实资源可以包括基本上如本文所述的核实器控制器组件。

[0180] 所述相应内部节点可以是段根。换句话说,“内部节点”是默克尔树中的节点,所述节点既不是整个树的根节点也不是叶节点。每个子集中的事务可以共享相应公共内部节点。

[0181] 在一种替代措辞中,每个子集可以包括与所述默克尔树中的公共节点相关联的至少两个事务,使得所述子集提供所述默克尔树的(子)部分,并且/或者由所述默克尔树的(子)部分表示。所述公共(内部)节点可以是基本上如本文所述的段节点或根。所述默克尔树的所述部分可以是基本上如本文所述的“段”。所述分布式核实节点内的所述核实资源中的至少一个或一些核实资源可以共享(即,订阅)公共IPv6多播地址。

[0182] 条款1.2.根据条款1.1所述的方法,其中核实所述区块链区块和/或区块链事务子集包括:

[0183] i) 核实和/或验证至少一个区块链事务;以及/或者

[0184] ii) 执行简单支付验证 (SPV) 过程;以及/或者

[0185] iii) 确认给定区块链事务 (Tx) 是否包含在所述区块链区块内;以及/或者

[0186] iii) 生成所述区块链事务中的至少一个区块链事务的哈希,使用所述哈希来构建默克尔路径,以及/或者检查所述哈希是否与所述区块链区块的头部中的事务标识符 (TxID) 匹配。

[0187] 条款1.3.根据条款1.1或1.2所述的方法,其中:

[0188] 所述区块链事务子集中的至少一个子集包括与所述子集相关联、标识所述子集以及/或者表示所述子集的标识符。

[0189] 条款1.4.根据条款1.3所述的方法,其中:

[0190] 所述标识符有利于计算所述至少一个子集在所述默克尔树内的位置。

[0191] 条款1.5.根据条款1.3或1.4所述的方法,其中:

[0192] 所述标识符包括所述至少一个区块链事务子集内的区块链事务的哈希的一部分。

[0193] 条款1.6.根据前述任一项条款所述的方法,其中:

[0194] 将所述相应区块链事务子集分配给所述多个核实资源的步骤包括:基于与所述事务子集相关联的相应标识符,将所述相应子集与相应核实资源匹配。

[0195] 条款1.7.根据前述任一项条款所述的方法,所述方法还包括以下步骤:

[0196] i) 将至少一个区块链事务子集下载到所述多个核实资源中的至少一个核实资源;以及/或者

[0197] ii) 将至少一个区块链事务子集发送到所述多个核实资源中的至少一个核实资源。

[0198] 条款1.8.根据前述任一项条款所述的方法,其中:

[0199] 所述默克尔树包括所述多个区块链事务的哈希的二叉树或网格 (mesh)。

[0200] 条款1.9.根据前述任一项条款所述的方法,所述方法还包括以下步骤:

[0201] 标识和/或确定所述多个区块链事务内的所述区块链事务子集。

[0202] 条款1.10.根据前述任一项条款所述的方法,其中:

[0203] 所述多个核实资源中的至少一个核实资源是或包括以下各项中的一项或多项:

- [0204] 虚拟机、服务器、基于GPU的计算资源、处理线程和/或多处理器系统。
- [0205] 条款1.11.根据前述任一项条款所述的方法,其中:
- [0206] i) 所述至少两个事务是所述默克尔树中的同级;以及/或者
- [0207] ii) 所述公共节点是所述至少两个事务的父辈或祖先。
- [0208] 条款1.12.一种区块链核实系统,所述系统用于核实区块链区块的至少一部分,所述区块链区块包括多个区块链事务和用于所述区块的默克尔树的根;
- [0209] 其中所述系统包括多个核实资源,每个核实资源包括:
- [0210] 处理器;以及
- [0211] 存储器,包括可执行指令,所述处理器执行所述可执行指令时,使得所述系统执行根据前述任一项条款所述的计算机实现的方法。
- [0212] 条款1.13.一种非暂时性计算机可读存储介质,其上存储有可执行指令,计算机系统的处理器执行所述可执行指令时,使得所述计算机系统执行根据条款1.1至1.11中任一项所述的计算机实现的方法。
- [0213] 根据本公开的另一方面,提供了一种计算机实现的系统,所述系统用于执行在本文中所描述或要求保护的任何方法步骤或方法步骤组合。
- [0214] 此外,还提供了一种区块链系统(网络),所述区块链系统(网络)包括多个计算机实现的节点,其中所述区块链网络中的每个节点包括:
- [0215] 处理器;以及
- [0216] 存储器,包括可执行指令,所述处理器执行所述可执行指令时,使得所述系统执行本文要求保护的或描述的计算机实现的方法的任何变体。
- [0217] 所述网络可以用于使用如本文所述的区块链协议来操作。
- [0218] 附加地或替代地,本公开可以包括一种下载区块链区块的至少一部分的计算机实现的方法。所述区块可以包括多个区块链事务和用于所述区块的默克尔树的根。所述方法可以包括以下条款中的一个或多个条款中所述的步骤:
- [0219] 条款集2:
- [0220] 条款2.1.一种计算机实现的方法,用于接收(例如,下载)区块链区块的至少一部分,所述区块链区块包括多个区块链事务和用于所述区块的默克尔树的根;所述方法包括:
- [0221] 将所述区块链事务的相应子集分配给多个处理资源,其中每个相应子集提供所述默克尔树的相应部分并由所述默克尔树的相应内部节点表示;以及
- [0222] 使用所述多个处理资源中的一个、一些或全部处理资源来接收或以其他方式下载其相应区块链事务子集。
- [0223] 根据一个或多个实施例,所述处理资源中的所述一个、一些或全部处理资源可以从提供(发送)资源接收其相应区块链事务子集。所述区块链区块的所述部分可以通过网络(例如,互联网)来发送。所述相应区块链事务子集可以使用IPv6多播传输来发送。所述多个处理资源中的至少一个、一些或全部处理资源可以是IPv6多播组的成员。所述发送资源可以包括网络设备。所述网络设备可以用于将IPv6多播通信发送到多播地址。可以在所述网络设备处启用MLD窥探。这在所述网络上的流量方面提供了效率,因为所述发送资源可以选择性地将一个或多个区块链事务子集发送到特定接收资源(例如,所述多个处理资源内的资源)。通过将数据包发送到所述网络上的所有资源,包括不需要或不希望接收数据的那些

资源,可以减少网络流量,并且不会浪费能量和处理资源。这还提高了安全性,因为这避免了拒绝服务(DOS)攻击的可能性,并且这还通过允许由于网络拥塞程度较低而导致的较快事务吞吐量来促进所述区块链网络的可扩展性。

[0224] 每个相应子集可以由所述相应内部节点表示,在这个意义上,所述相应内部节点可以对所述相应子集进行编码。也就是说,所述相应内部节点可以基于(即,根据)所述相应子集来生成。相应子集中的每个事务可以通过一个或多个哈希操作链接到所述相应内部节点。

[0225] 条款2.2.根据条款2.1所述的方法,所述方法包括:

[0226] 所述多个处理资源中的一个、一些或全部处理资源将其相应区块链事务子集发送到中央存储处。

[0227] 条款2.3.根据条款2.2所述的方法,其中:

[0228] 所述默克尔树的所述相应内部节点在所述默克尔树中具有相应位置,并且其中所述方法包括:

[0229] 基于所述默克尔树的所述相应内部节点的所述相应位置来排列所述相应区块链事务子集。

[0230] 条款2.4.根据前述任一项条款所述的方法,所述方法包括:

[0231] 所述处理资源中的一个、一些或全部处理资源基于所下载的相应区块链事务子集来生成所述默克尔树的相应候选内部节点;并且所述方法还包括以下各项中的至少一项:

[0232] 验证所述相应候选内部节点与所述默克尔树的所述相应内部节点匹配;和/或

[0233] 基于所述默克尔树的所述根通过执行默克尔证明来验证所述相应候选内部节点是所述默克尔树的节点;和/或

[0234] 将所述默克尔树的所述相应候选内部节点发送到一个或多个其他处理资源。

[0235] 条款2.5.根据前述任一项条款所述的方法,所述方法包括:

[0236] 使用所述多个处理资源中的一个、一些或全部处理资源来核实其相应区块链事务子集。

[0237] 条款2.6.根据条款2.1所述的方法,其中:

[0238] 核实所述相应区块链事务子集包括:

[0239] i) 核实和/或验证至少一个区块链事务;和/或

[0240] ii) 执行简单支付验证过程;和/或

[0241] iii) 确认给定区块链事务是否包含在所述区块链区块内;和/或

[0242] iii) 生成所述区块链事务中的至少一个区块链事务的哈希,使用所述哈希来构建默克尔路径,和/或检查所述哈希是否与所述区块链区块的区块头中的事务标识符匹配。

[0243] 条款2.7.根据前述任一项条款所述的方法,其中:

[0244] 所述相应区块链事务子集中的至少一个相应子集包括相应标识符,所述相应标识符与所述相应子集相关联、标识所述相应子集、和/或表示所述相应子集。

[0245] 条款2.8.根据条款2.7所述的方法,其中:

[0246] 所述相应标识符有利于计算所述至少一个相应子集在所述默克尔树内的相应位置。

[0247] 条款2.9.根据条款2.7或2.8所述的方法,其中:

- [0248] 所述相应标识符基于所述默克尔树的所述相应内部节点。
- [0249] 10. 根据条款9所述的方法, 其中:
- [0250] 所述相应标识符包括所述默克尔树的所述相应内部节点的一部分。
- [0251] 条款2.11. 根据前述任一项条款所述的方法, 其中:
- [0252] 将所述相应区块链事务子集分配给所述多个相应处理资源的步骤包括: 基于与所述相应事务子集相关联的相应标识符来将所述相应子集与相应处理资源进行匹配。
- [0253] 条款2.12. 根据前述任一项条款所述的方法, 其中:
- [0254] 所述默克尔树包括所述多个区块链事务的哈希的二叉树或网状结构。
- [0255] 条款2.13. 根据前述任一项条款所述的方法, 所述方法包括:
- [0256] 标识和/或确定所述多个区块链事务内的所述区块链事务子集。
- [0257] 条款2.14. 根据前述任一项条款所述的方法, 其中:
- [0258] 所述多个处理资源中的至少一个处理资源是或包括: 虚拟机、服务器、基于GPU的计算资源、或多处理器系统。
- [0259] 条款2.15. 一种区块链处理系统, 所述系统用于下载区块链区块的至少一部分, 所述区块链区块包括多个区块链事务和用于所述区块的默克尔树的根; 其中所述系统包括多个处理资源, 每个处理资源包括:
- [0260] 处理器; 以及
- [0261] 存储器, 包括可执行指令, 所述处理器执行所述可执行指令时, 导致所述系统执行、或使得所述系统能够执行根据前述任一项条款所述的计算机实现的方法。
- [0262] 条款2.16. 一种非暂时性计算机可读存储介质, 其上存储有可执行指令, 计算机系统的处理器执行所述可执行指令时, 导致所述计算机系统执行、或使得所述计算机系统能够执行根据条款2.1至2.14中任一项所述的计算机实现的方法。
- [0263] 根据另一方面, 提供了一种非暂时性计算机可读存储介质, 其上存储有可执行指令, 计算机系统的处理器执行所述可执行指令时, 使得所述计算机系统执行本文要求保护的或描述的计算机实现的方法的任何版本。
- [0264] 条款集3:
- [0265] 条款3.1. 一种计算机实现的方法, 所述方法包括以下步骤:
- [0266] 生成、存储和/或维护第一输出存储库, 用于记录、搜索和/或处理多个未花费事务输出, 每个未花费事务输出与区块链区块的多个区块链事务 (TXs) 中的事务 (Tx) 相关联;
- [0267] 其中:
- [0268] 所述多个区块链事务提供所述区块链区块的默克尔树的一部分, 并且/或者由用于所述区块链区块的默克尔树的一部分表示。
- [0269] 在一些实施例中, 所述第一输出存储库可以称为第一UTXO输出存储库。所述未花费事务输出可以称为UTXO。
- [0270] 条款3.2. 根据条款3.1所述的方法, 所述方法包括:
- [0271] 生成、存储和/或维护至少一个其他输出存储库。
- [0272] 条款3.3. 根据条款3.1或3.2所述的方法, 所述方法还包括:
- [0273] 创建和/或维护数据库日志, 所述数据库日志包括与所述输出存储库相关的动作、更改和事件的历史记录。

- [0274] 条款3.4.根据前述任一项条款所述的方法,其中:
- [0275] 所述第一输出存储库和/或所述其他输出存储库包括至少一个记录,所述至少一个记录与以下各项相关联:
- [0276] i) 未花费事务输出;以及/或者
- [0277] ii) 标识符,所述标识符与a) 未花费事务输出和/或b) 所述多个区块链事务中的事务 (Tx) 相关联。
- [0278] 条款3.5.根据条款3.4所述的方法,其中:
- [0279] 所述至少一个记录包括记录标识符,所述记录标识符具有:
- [0280] i) 区块标识符 (block_ID),所述区块标识符与区块链区块相关联;以及/或者
- [0281] ii) 事务标识符 (TxID),所述事务标识符与所述多个区块链事务中的事务 (Tx) 相关联。
- [0282] 条款3.6.根据条款3.5所述的方法,其中:
- [0283] i) 所述记录标识符包括所述区块标识符 (block_ID) 和所述事务标识符 (TxID) 的函数;以及/或者
- [0284] ii) 所述区块标识符 (block_ID) 和所述事务标识符 (TxID) 的级联;以及/或者
- [0285] iii) 所述多个区块链事务中的所述事务与未花费事务输出 (UTX0) 相关联。
- [0286] 条款3.7.根据条款3.5或3.6所述的方法,所述方法还包括以下步骤:
- [0287] 使用所述记录标识符来搜索、标识、访问或插入所述输出存储库中的至少一个记录。
- [0288] 条款3.8.根据前述任一项条款所述的方法,其中:
- [0289] 所述多个未花费事务输出 (UTX0) 中的至少一个UTX0在所述输出存储库中与锁定标志相关联,所述锁定标志:
- [0290] i) 指示所述未花费事务输出 (UTX0) 是否可用于花费;以及/或者
- [0291] ii) 可在第一状态与第二状态之间配置,所述第一状态指示允许花费所述未花费事务输出,所述第二状态指示禁止花费所述未花费事务输出。
- [0292] 条款3.9.根据条款3.8所述的方法,所述方法包括以下步骤:
- [0293] i) 将所述未花费事务输出 (UTX0) 与所述锁定标志相关联;以及/或者
- [0294] ii) 将所述锁定标志的状态从所述第一状态更改为所述第二状态,或者从所述第二状态更改为所述第一状态。
- [0295] 条款3.10.根据条款3.8或3.9所述的方法,所述方法包括以下步骤:
- [0296] 从第一处理资源向至少一个其他处理资源发送通信,以使所述至少一个其他处理资源将与所述未花费事务输出相关联的所述锁定标志的所述状态从所述第一状态更改为所述第二状态,或者从所述第二状态更改为所述第一状态。
- [0297] 可以使用多播传输将所述通信发送到与所述至少一个其他处理资源相关联的IPv6地址。
- [0298] 条款3.11.根据条款3.10所述的方法,其中所述通信包括:
- [0299] i) 事务 (TX)、事务标识符 (TxID) 和/或事务 (Tx) 的哈希;以及
- [0300] ii) 一个或多个未花费事务输出 (UTX0) 的列表。
- [0301] 条款3.12.根据条款3.10或3.11所述的方法,所述方法包括以下步骤:

- [0302] 在所述至少一个其他处理资源处接收所述通信；
- [0303] 将所述锁定标志的所述状态从所述第一状态更改为所述第二状态,或者从所述第二状态更改为所述第一状态。
- [0304] 条款3.13.根据前述任一项条款所述的方法,其中:
- [0305] i) 所述默克尔树的所述部分是用于所述区块链区块的所述默克尔树的子部分或段;以及/或者
- [0306] ii) 所述多个区块链事务由所述默克尔树的内部节点表示。
- [0307] 条款3.14.一种区块链实现的系统,所述系统包括多个处理资源,每个处理资源包括:
- [0308] 处理器;以及
- [0309] 存储器,包括可执行指令,所述处理器执行所述可执行指令时,导致所述系统执行、或使得所述系统能够执行根据前述任一项条款所述的计算机实现的方法。
- [0310] 15.一种非暂时性计算机可读存储介质,其上存储有可执行指令,计算机系统的处理器执行所述可执行指令时,导致所述系统执行、或使得所述系统能够执行根据条款3.1至3.13中任一项所述的计算机实现的方法。
- [0311] 条款集4:
- [0312] 在条款集4中的任何条款或条款的组合中定义的任何实施例可以用于实现条款集1至3中的任何一个或多个条款,或者与条款集1至3中的任何一个或多个条款组合。
- [0313] 条款4.1.一种系统,所述系统用于核实区块链区块的至少一部分,所述区块链区块包括多个区块链事务和用于所述区块的默克尔树的根,所述系统包括:
- [0314] 多个核实资源,每个核实资源包括:
- [0315] 至少一个处理器,所述至少一个处理器与存储可执行指令的存储器的至少一部分相关联,所述至少一个处理器执行所述可执行指令时,使得所述核实资源:
- [0316] 核实所述多个区块链事务的至少一个子集,其中所述至少一个子集提供所述默克尔树的一部分,并且由所述默克尔树的内部节点表示。
- [0317] 条款4.2.根据条款4.1所述的系统,其中所述系统还包括:
- [0318] i) 负载均衡组件,用于促进在所述多个核实资源之间均衡所述多个区块链事务的多个子集的分布;以及/或者
- [0319] ii) 段标识组件,用于促进标识所述多个区块链事务的所述至少一个子集;以及/或者
- [0320] iii) 分配单元;以及/或者
- [0321] iv) 一个或多个接口,用于在所述系统与一个或多个数据源或目的地之间发送或接收通信。
- [0322] 条款4.3.根据条款4.1或4.2所述的系统,其中:
- [0323] 所述系统包括至少一个控制器组件,所述控制器组件用于影响和/或控制以下各项中的至少一项的操作:
- [0324] 至少一个核实资源;
- [0325] 所述至少一个核实资源的至少一个处理器;
- [0326] 一个或多个接口;

- [0327] 一个或多个负载均衡组件;以及/或者
- [0328] 一个或多个段标识组件,用于促进标识所述多个区块链事务的所述至少一个子集。
- [0329] 条款4.4.根据前述任一项条款所述的系统,其中:
- [0330] i) 所述多个区块链事务中的至少两个事务是所述默克尔树中的同级;以及/或者
- [0331] ii) 所述内部节点是所述区块链事务子集的父辈或祖先。
- [0332] 条款4.5.根据前述任一项条款所述的系统,所述系统还包括:
- [0333] 多个UTX0存储库,所述多个存储库中的每个存储库与相应核实资源相关联,并且用于促进记录、搜索和/或处理多个未花费事务输出(UTX0);优选地,其中所述多个未花费事务输出中的每个未花费事务输出与所述多个区块链事务中的至少一个事务(Tx)相关联。
- [0334] 条款4.6.根据条款4.5所述的系统,其中所述系统用于:
- [0335] 创建和/或维护数据库日志,所述数据库日志包括与所述多个UTX0存储库中的至少一个UTX0存储库相关的动作、更改和事件的历史记录。
- [0336] 条款4.7.根据前述任一项条款所述的系统,其中:
- [0337] 所述多个UTX0存储库中的至少一个UTX0存储库包括至少一个记录,所述至少一个记录与以下各项相关联:
- [0338] i) 未花费事务输出(UTX0);以及/或者
- [0339] ii) 标识符,所述标识符与a) 未花费事务输出和/或b) 所述多个区块链事务中的事务(Tx)相关联。
- [0340] 条款4.8.根据条款4.7所述的系统,其中:
- [0341] 所述至少一个记录包括记录标识符,所述记录标识符具有:
- [0342] i) 区块标识符(block_ID),所述区块标识符与所述区块链区块相关联;以及/或者
- [0343] ii) 事务标识符(TxID),所述事务标识符与所述多个区块链事务中的事务(Tx)相关联。
- [0344] 条款4.9.根据条款4.7或4.8所述的系统,其中所述记录标识符包括:
- [0345] i) 所述区块标识符(block_ID)和所述事务标识符(TxID)的函数;以及/或者
- [0346] ii) 所述区块标识符(block_ID)和所述事务标识符(TxID)的级联。
- [0347] 10.根据条款8或9所述的系统,其中所述系统用于:
- [0348] 使用所述记录标识符来搜索、标识、访问或插入所述多个UTX0存储库中的至少一个UTX0存储库中的所述至少一个记录。
- [0349] 条款4.11.根据条款4.5至4.10中任一项所述的系统,其中:
- [0350] 所述多个未花费事务输出(UTX0)中的至少一个未花费事务输出(UTX0)与锁定标志相关联,所述锁定标志:
- [0351] i) 指示所述未花费事务输出(UTX0)是否可用于花费;以及/或者
- [0352] ii) 可在第一状态与第二状态之间配置,所述第一状态指示允许花费所述未花费事务输出,所述第二状态指示禁止花费所述未花费事务输出。
- [0353] 条款4.12.根据条款4.11所述的系统,所述系统用于:
- [0354] 将所述锁定标志的状态从所述第一状态更改为所述第二状态,或者从所述第二状态更改为所述第一状态。

[0355] 条款4.13.根据前述任一项条款所述的系统,其中所述系统用于:

[0356] i) 将所述区块链事务的相应子集分配给所述多个核实资源;以及

[0357] ii) 使用所述多个核实资源中的一个、一些或全部核实资源来下载和/或接收相应区块链事务子集。

[0358] 条款4.14.根据前述任一项条款所述的系统,所述系统用于:

[0359] 使用所述核实资源中的一个、一些或全部核实资源,基于所述下载的相应区块链事务子集来生成所述默克尔树的相应候选内部节点;并且所述系统还用于执行以下各项中的至少一项:

[0360] 验证所述相应候选内部节点与所述默克尔树的所述相应内部节点匹配;以及/或者

[0361] 通过基于所述默克尔树的所述根执行默克尔证明来验证所述相应候选内部节点是所述默克尔树的节点;以及/或者

[0362] 将所述默克尔树的所述相应候选内部节点发送到一个或多个其他处理资源。

[0363] 条款4.15.根据前述任一项条款所述的系统,其中所述系统用于:

[0364] i) 核实和/或验证至少一个区块链事务;以及/或者

[0365] ii) 执行简单支付验证过程;以及/或者

[0366] iii) 确认给定区块链事务是否包含在所述区块链区块内;以及/或者

[0367] iii) 生成所述区块链事务中的至少一个区块链事务的哈希,使用所述哈希来构建默克尔路径,以及/或者检查所述哈希是否与所述区块链区块的区块头中的事务标识符匹配。

[0368] 条款4.16.根据前述任一项条款所述的系统,其中:

[0369] 所述多个核实资源中的至少一个核实资源是或包括以下各项中的至少一项:虚拟机、服务器、基于GPU的计算资源、线程和/或多处理器系统。

[0370] 用于实现本公开的说明性实施例的示例性技术环境

[0371] 现在描述可以在其中实践本公开的一个或多个实施例的计算环境的概述。然而,如上所述,该上下文并非旨在进行限制,并且实施例可以用于处理并非经由区块链实现的数据记录和结构。例如,可以使用数据库而不是分布式分类账来设计非区块链实施例。

[0372] 图1示出了一种用于实现区块链150的示例性系统100。系统100可以包括分组交换网络101,通常是诸如互联网的广域互联网。分组交换网络101包括多个区块链节点104,该多个区块链节点可以被设置成在分组交换网络101内形成对等(P2P)网络106。虽然未示出,但是区块链节点104可以被设置为近完全图。因此,每个区块链节点104高度连接到其它区块链节点104。

[0373] 每个区块链节点104包括对等体的计算机设备,不同的节点104属于不同的对等体。每个区块链节点104包括处理装置,该处理装置包括一个或多个处理器,例如一个或多个中央处理单元(CPU)、加速器处理器、专用处理器和/或现场可编程门阵列(FPGA),以及其它设备,例如专用集成电路(ASIC)。每个节点还包括存储器,即采用非暂时性计算机可读介质形式的计算机可读存储器。存储器可包括一个或多个存储器单元,其采用一个或多个存储器介质,例如诸如硬盘等磁介质、诸如固态硬盘(SSD)、闪存或电可擦可编程只读存储器(EEPROM)等电子媒介和/或诸如光盘驱动器等光学介质。

[0374] 区块链150包括一系列数据区块151,其中在分布式或区块链网络106中的多个区块链节点104中的每个节点处维护区块链150的相应副本。如上所述,维护区块链150的副本不一定意味着完全存储区块链150。相反,只要每个区块链节点150存储每个区块151的区块头(下面讨论),区块链150就可以进行数据修剪。区块链中的每个区块151均包括一个或多个事务152,其中该上下文中的事务是指一种数据结构。数据结构的性质将取决于用作事务模型或计划的一部分的事务协议类型。给定的区块链全程使用一个特定的事务协议。在一种常见的事务协议中,每个事务152的数据结构至少包括一个输入和至少一个输出。每个输出指定将数字资产的数量表示为财产的数额,其一个示例是输出被密码锁定到的用户103(需要该用户的签名或其它解进行解锁,从而进行赎回或花费)。每个输入指向先前事务152的输出,从而链接这些事务。

[0375] 每个区块151还包括区块指针155,其指向区块链中先前创建的区块151,以定义区块151的顺序。每个事务152(除coinbase事务之外)包括指向先前事务的指针,以定义事务序列的顺序(注:事务152的序列可进行分支)。区块151的区块链一直追溯到创始区块(Gb)153,该创始区块是区块链中的第一区块。区块链150中早期的一个或多个原始事务152指向创始区块153,而非先前事务。

[0376] 每个区块链节点104被配置为将事务152转发到其它区块链节点104,从而使得事务152在整个网络106中传播。每个区块链节点104被配置为创建区块151,并将相同区块链150的相应副本存储在其相应的存储器中。每个区块链节点104还维护等待并入到区块151中的事务152的有序集(或“池”)154。有序池154通常称为“内存池”。在本文中,该术语并不意在限制于任何特定的区块链、协议或模型。该术语是指节点104已接受为有效的有序事务集,并且对于该有序事务集,强制节点104不接受试图花费相同输出的任何其它事务。

[0377] 在给定的当前事务152j中,输入(或每个输入)包括指针,该指针引用事务序列中先前事务152i的输出,指定该输出将在当前事务152j中被赎回或“花费”。通常,先前事务可以是有序集154或任何区块151中的任何事务。尽管为了确保当前事务有效,将需要存在先前事务152i并核实其有效,但是在创建当前事务152j甚至向网络106发送当前事务152j时,不必存在先前事务152i。因此,在本文中,“先前”是指由指针链接的逻辑序列中的前任,而不一定是时间序列中的创建时间或发送时间,因此,不一定排除无序创建或发送事务152i、152j的情况(参见下面关于孤立事务的讨论)。先前事务152i同样可以称为先行事务或前任事务。

[0378] 当前事务152j的输入还包括输入授权,例如先前事务152i的输出被锁定到的用户103a的签名。反过来,当前事务152j的输出可以加密锁定到新用户或实体103b。因此,当前事务152j可将先前事务152i的输入中定义的数额转移到当前事务152j的输出中定义的新用户或实体103b。在某些情况下,事务152可具有多个输出,以在多个用户或实体间分割输入数额(其中一个可以是原始用户或实体103a,以便进行变更)。在某些情况下,事务还可以具有多个输入,将一个或多个先前事务的多个输出中的数额汇总在一起,并重新分配到当前事务的一个或多个输出。

[0379] 根据基于输出的事务协议,例如比特币,当诸如个体用户或组织这类的一方103希望颁布新的事务152j时(由该方采用的自动程序或人为地),该颁布方将该新事务从其计算机终端102发送到接收者。颁布方或接收者将最终向网络106的一个或多个区块链节点104

(现在通常是服务器或数据中心,但原则上也可以是其它用户终端)发送该事务。另外还不排除颁布新事务152j的一方103可以将事务直接发送到一个或多个区块链节点104,并且在一些示例中,可以不将事务发送到接收者。接收事务的区块链节点104根据在每个区块链节点104处应用的区块链节点协议来检查事务是否有效。区块链节点协议通常要求区块链节点104检查新事务152j中的加密签名是否与预期签名相匹配,这取决于事务152的有序序列中的先前事务152i。在这种基于输出的事务协议中,这可以包括检查新事务152j的输入中包括的一方103的密码签名或其它授权是否与新事务分配的先前事务152i的输出中定义的条件匹配,其中该条件通常包括至少检查新事务152j的输入中的密码签名或其它授权是否解锁新事务的输入所链接到的先前事务152i的输出。条件可以至少部分地由包括在先前事务152i的输出中的脚本来定义。或者,这可仅由区块链节点协议单独确定,或可通过其组合确定。无论采用哪种方式,如果新事务152j有效,区块链节点104会将其转发到区块链网络106中的一个或多个其它区块链节点104。这些其它区块链节点104根据相同的区块链节点协议应用相同的测试,并因此将新事务152j转发到一个或多个其它节点104等等。通过这种方式,新事务在区块链节点104的整个网络中进行传播。

[0380] 在基于输出的模型中,给定输出(例如,UTX0)是否分配(例如,花费)的定义是,根据区块链节点协议,其是否通过另一个随后事务152j的输入有效赎回。事务有效的另一个条件是其试图赎回的先前事务152i的输出尚未被另一个事务赎回。同样,如果无效,则事务152j将不会在区块链150中传播(除非被标记为无效并且被传播用于提醒)或记录。这可防止重复花费,即事务处理者对同一个事务的输出分配超过一次。另一方面,基于账户的模型通过保持账户余额防止重复花费。因为同样存在定义的事务顺序,账户余额在任何时候均具有单一定义的状态。

[0381] 除了核实事务有效之外,区块链节点104还争相成为在通常称为挖掘的过程中创建事务区块的第一个节点,而该过程由“工作证明”支持。在区块链节点104处,新事务被添加到尚未出现在记录在区块链150上的区块151中的有效事务的有序池154。然后,区块链节点争相通过尝试解决加密难题以组装有序事务集154中事务152的新有效事务区块151。通常情况下,这包括搜索“随机数”值,从而当随机数与未决事务有序池154的表示并置且进行哈希处理时,哈希值的输出满足预定条件。例如,预定条件可以是哈希值的输出具有某个预定义的前导零数。注意,这仅仅是一种特定类型的工作证明难题,并且不排除其它类型。哈希函数的特性是,相对于其输入,其具有不可预测的输出。因此,该搜索只能通过强力执行,从而在试图解决难题的每个区块链节点104处消耗大量的处理资源。

[0382] 解决难题的第一区块链节点104在网络106上宣布难题解决,提供解决方案作为证明,然后网络中的其它区块链节点104则可以轻松检查该解决方案(一旦给出哈希值的解决方案,就可以直接检查该解决方案是否使哈希值的输出满足条件)。第一区块链节点104将一个区块传播到接受该区块的其它节点达成阈值共识,从而执行协议规则。然后,有序事务集154被每个区块链节点104记录为区块链150中的新区块151。区块指针155还分配给指向该区块链中先前创建的区块151n-1的新区块151n。创建工作证明解所需的大量工作(例如采用哈希的形式)发出信号通知第一节点104的意图以遵循区块链协议。这些规则包括如果它分配与先前核实有效的事务相同的输出,则不接受事务为有效,否则称之为重复花费。一旦创建,区块151就不能修改,因为它在区块链网络106中的每个区块链节点104处进行标识

和维护。区块指针155还向区块151施加顺序。由于事务152记录在网络106中每个区块链节点104处的有序区块中,因此提供了事务的不可改变公共分类账。

[0383] 应当注意的是,在任何给定时间争相解决难题的不同区块链节点104可以基于在任何给定时间尚未发布的事务的池154的不同快照来这样做,具体取决于它们何时开始搜索解或接收事务的顺序。解决相应难题的人员首先定义新区块151n中包括的事务152及其顺序,并且更新当前的未发布事务池154。然后,区块链节点104继续争相从新定义的未发布事务有序池154中创建区块,等等。此外,还存在解决可能出现的任何“分叉”的协议,其中两个区块链节点104彼此在很短的时间内解决难题,从而在节点104之间传播区块链的冲突视图。简言之,分叉方向最长的成为最终区块链150。应当注意的是,这不会影响网络的用户或代理,因为同一事务将出现在两个分叉中。

[0384] 根据比特币区块链(和大多数其它区块链),成功构造新区块104的节点被授予在分配附加限定数量数字资产的新特殊类型事务中新分配附加的、接受的数额的数字资产的能力(与代理间或用户间事务相反,该事务将一定数量的数字资产从一个代理或用户转移到另一个代理或用户)。这种特殊类型的事务通常称为“coinbase事务”,但是也可以称为“启动事务”或“产生事务”。它通常形成新区块151n的第一事务。工作证明发出信号通知构造新区块的节点的意图以遵循协议规则,从而允许稍后赎回该特定事务。在可以赎回该特殊事务之前,区块链协议规则可能需要成熟期,例如100个区块。通常,常规(非生成)事务152还将在其输出中的一个输出中指定附加事务费用,以进一步奖励创建其中发布该事务的区块151n的区块链节点104。该费用通常称为“事务费用”,并在下文中讨论。

[0385] 由于事务核实和发布中涉及的资源,通常至少每个区块链节点104采用包括一个或多个物理服务器单元的服务器形式,或者甚至整个数据中心。但是,原则上来说,任何给定区块链节点104均可采用一个用户终端或联网在一起的一组用户终端的形式。

[0386] 每个区块链节点104的存储器均存储被配置为在区块链节点104的处理装置上运行的软件,以根据区块链节点协议执行其相应的角色并处理事务152。应当理解的是,在本文中归因于区块链节点104的任何动作均可通过在相应计算机设备的处理装置上运行的软件执行。节点软件可以在应用层或诸如操作系统层或协议层的较低层或这些层任意组合的一个或多个应用中实现。

[0387] 扮演消费用户角色的多方103中的每一方的计算机设备102也连接到网络101。这些用户可以与区块链网络106交互,但不参与核实事务或构造区块。其中一些用户或代理103可以充当事务中的发送者和接收者。其它用户可以与区块链150交互,而不必充当发送者或接收者。例如,一些当事方可以充当存储区块链150的副本(例如,已经从区块链节点104获得区块链的副本)的存储实体。

[0388] 各方103中的一些或所有当事方可以作为不同网络的一部分连接,例如覆盖在区块链网络106之上的网络。区块链网络的用户(经常称为“客户端”)可以被称为是包含区块链网络106的系统的一部分;然而,这些用户不是区块链节点104,因为它们不执行区块链节点所需的角色。相反,每一方103可以与区块链网络106交互,从而通过连接到区块链节点106(即,与区块链节点106通信)来利用区块链150。出于说明目的,示出了双方103及其相应的设备102:第一方103a及其相应的计算机设备102a,以及第二方103b及其相应的计算机设备102b。应当理解的是,更多此类当事方103及其相应的计算机设备102可能存在并参与系

统100,但为了方便起见,未进行说明。每一方103均可以是个人或组织。仅出于说明目的,在本文中,第一方103a称为爱丽丝,第二方103b称为鲍勃,但应当理解的是,这并不仅限于爱丽丝或鲍勃,且本文对爱丽丝或鲍勃的任何引用均可分别用“第一方”和“第二方”替换。

[0389] 每一方103的计算机设备102包括相应的处理装置,其包括一个或更多个处理器,例如一个或更多个CPU、图形处理单元(GPU)、其他加速器处理器、特定应用程序处理器和/或FPGA。每一方103的计算机设备102还包括存储器,即采用非暂时性计算机可读介质形式的计算机可读存储器。该存储器可包括一个或更多个存储器单元,其采用一个或更多个存储器介质,例如诸如硬盘等磁介质、诸如SSD、闪存或EEPROM等电子媒介和/或诸如光盘驱动器等的的光学介质。每一方103的计算机设备102上的存储器存储软件,其包括被设置为在处理装置上运行的至少一个客户端应用程序105的相应实例。应当理解的是,在本文中归因于给定方103的任何行动均可通过在相应计算机设备102的处理装置上运行的软件执行。每一方103的计算机设备102包括至少一个用户终端,例如台式或笔记本电脑、平板电脑、智能手机或诸如智能手表等的可穿戴设备。给定方103的计算机设备102还可包括一个或更多个其他网络资源,诸如通过用户终端访问的云计算资源。

[0390] 客户端应用程序105最初可通过例如从服务器下载的适当计算机可读存储介质,或通过诸如可移动SSD、闪存密钥、可移动EEPROM、可移动磁盘驱动器、软盘或磁带等的可移动存储设备、诸如CD或DVD ROM等的光盘或可移动光驱等提供至任何给定方103的计算机设备102。

[0391] 客户端应用程序105至少包括“钱包”功能。这有两个主要功能。其中一个功能是使相应方103能够创建、授权(例如签名)事务152并将其发送到一个或多个比特币节点104,然后在区块链节点104的网络中传播,从而包括在区块链150中。另一个功能是向相应方汇报其目前拥有的数字资产数额。在基于输出的系统中,该第二功能包括整理分散在区块链150中属于相关方的各种事务152的输出中定义的数额。

[0392] 注意:虽然各种客户端功能可以描述为集成到给定客户端应用程序105中,但这不一定是限制性的,相反,在本文中所描述的任何客户端功能可以在由两个或更多个不同应用程序组成的套件中实现,例如经由API进行接口连接或一个应用程序作为另一个应用程序的插件。更通俗地说,客户端功能可以在应用层或诸如操作系统的较低层或这些层的任意组合实现。下面将根据客户端应用程序105进行描述,但应当理解的是,这不是限制性的。

[0393] 每个计算机设备102上的客户端应用程序或软件105的实例可操作地耦合到网络106的区块链节点104中的至少一个。这可以启用客户端105的钱包功能,以将事务152发送至网络106。客户端105还可联络区块链节点104,以在区块链150中查询相应方103作为接收者的任何事务(或实际上在区块链150中检查其它方的事务,因为在实施例中,区块链150是在某种程度上通过其公开可见性提供事务信任的公共设施)。每个计算机设备102上的钱包功能被配置为根据事务协议制定和发送事务152。如上所述,每个区块链节点104运行软件,该软件被配置为根据区块链节点协议核实事务152并转发事务152以便在区块链网络106中传播。事务协议和节点协议相互对应,给定事务协议和给定节点协议一起实现给定的事务模型。相同的事务协议用于区块链150中的所有事务152。网络106中的所有节点104使用相同的节点协议。

[0394] 当给定方103(比方说爱丽丝)希望发送拟包含在区块链150中的新事务152j时,她

将根据相关事务协议(使用其客户端应用程序105中的钱包功能)制定新事务。然后,她将事务152从客户端应用程序105发送到她所连接的一个或多个区块链节点104。例如,这可能是与爱丽丝的计算机102最佳连接的区块链节点104。当任何给定区块链节点104接收新事务152j时,其将根据区块链节点协议及其相应的角色进行处理。这包括首先检查新接收的事务152j是否满足变为“有效”的特定条件,具体示例稍后将详细讨论。在一些事务协议中,有效条件可通过事务152中包含的脚本在每个事务的基础上进行配置。或者,条件可仅仅是节点协议的内置功能,或通过组合脚本和节点协议进行定义。

[0395] 如果新接收的事务152j通过有效性测试(即:“有效”的条件下),接收事务152j的任何区块链节点104将向在区块链节点104处维护的有序事务集154中添加新的核实有效事务152。进一步地,接收事务152j的任何区块链节点104随后将核实有效事务152传播至网络106中的一个或多个其它区块链节点104。由于每个区块链节点104应用相同的协议,因此假定事务152j有效,这意味着事务很快将在整个网络106中传播。

[0396] 一旦进入在给定区块链节点104处维护的未决事务有序池154,该区块链节点104将开始争相解决其各自的包含新事务152的池154的最新版本上的工作证明难题(请记住,其它区块链节点104可以尝试基于不同的事务池154来解决难题。但是,首先解决难题的人将定义包括在最新区块151中的事务集合。最终,区块链节点104将解决有序池154的一部分的难题,该有序集154包括爱丽丝的事务152j)。一旦包括新事务152j的池154完成工作证明,其将不可变地成为区块链150中区块151中的一个区块的一部分。每个事务152包括指向早前事务的指针,因此事务的顺序也被不可变地记录下来。

[0397] 不同的区块链节点104可以首先接收给定事务的不同实例,并且因此在一个实例被发布到新区块151中之前具有关于哪个实例“有效”的冲突视图,此时所有区块链节点104同意所发布的实例是唯一的有效实例。如果区块链节点104将一个实例接受为有效实例,然后发现第二实例已记录在区块链150中,则区块链节点104必须接受这一点,并将丢弃(即,视为无效)其最初接受的实例(即,在区块151中尚未公布的实例)。

[0398] 作为基于账户的事务模型的一部分,由一些区块链网络操作的另一种类型的事务协议可称为“基于账户的”协议。在基于账户的情况下,每个事务均不通过参考过去事务序列中先前事务的UTX0来定义转移的数额,而是通过参考绝对账户余额进行定义。所有账户的当前状态由网络的节点单独存储到区块链中,并不断更新。在此类系统中,事务使用账户的运行事务记录(也称为“头寸”)进行排序。该值由发送者签名作为其加密签名的一部分,并作为事务引用计算的一部分进行哈希处理。此外,可选的数据字段也可以在事务中签名。例如,如果数据字段中包含先前事务的ID,该数据字段可指向先前事务。

[0399] 基于UTX0的模型

[0400] 图2示出了示例性事务协议。这是基于UTX0的协议的示例。事务152(简称“Tx”)是区块链150的基本数据结构(每个区块151包括一个或多个事务152)。下面将通过参考基于输出或基于“UTX0”的协议进行描述。但这并不限于所有可能的实施例。应当注意的是,虽然参考比特币描述了示例性基于UTX0的协议,但是它同样可以在其它示例区块链网络上实现。

[0401] 在基于UTX0的模型中,每个事务(“Tx”)152包括数据结构,其包括一个或多个输入202和一个或多个输出203。每个输出203可包括未花费事务输出(UTX0),其可用作另一新事

务的输入202的来源(如果UTX0尚未赎回)。UTX0包括指定数字资产数额的值。这表示分布式分类账上的一组通证。UTX0还可包含其来源事务的事务ID以及其它信息。事务数据结构还可包括标头201,其可包括输入字段202和输出字段203的大小指示符。标头201还可包括事务的ID。在实施例中,事务ID是事务数据(不含事务ID本身)的哈希值,且存储在提交至节点104的原始事务152的标头201中。

[0402] 比方说爱丽丝103a希望创建转移相关数字资产数额至鲍勃103b的事务152j。在图2中,爱丽丝的新事务152j标记为“Tx₁”。该新事务获取在序列中先前事务152i的输出203中锁定至爱丽丝的数字资产数额,并至少将此类数额中的一部分转移至鲍勃。在图2中,先前事务152i标记为“Tx₀”。Tx₀和Tx₁只是任意的标记,其不一定意味着Tx₀指区块链151中的第一事务且Tx₁指池154中的后续事务。Tx₁可指向仍具有锁定至爱丽丝的未花费输出203的任何先前(即先行)事务。

[0403] 当爱丽丝创建其新事务Tx₁时,或至少在她将该新事务发送至网络106时,先前事务Tx₀可能已经有效并包括在区块链150的区块151中。该事务此时可能已包括在区块151中的一个区块中,或者可能仍在有序集154中等待,在这种情况下,该事务将很快包括在新区块151中。或者,Tx₀和Tx₁可以创建并一起发送至网络106;或者,如果节点协议允许缓冲“孤立”事务,Tx₀甚至可以在Tx₁之后发送。本文事务序列上下文中使用的“先前”和“后续”一词是指由事务中指定的事务指针定义的序列中的事务顺序(哪个事务指向哪个其他事务等等)。它们同样可以替换为“前任”和“继任”、“先行”和“后代”或“父项”和“子项”等。这不一定指其创建、发送至网络106或到达任何给定区块链节点104的顺序。然而,指向先前事务(先行事务或“父事务”)的后续事务(后代事务或“子事务”)不会有效除非父事务有效。在父事务之前到达区块链节点104的子事务被视为孤立事务。根据节点协议和/或节点行为,其可被丢弃或缓冲一段时间,以等待父事务。

[0404] 先前事务Tx₀的一个或更多个输出203中的一个包括特定的UTX0,标记为UTX0₀。每个UTX0包括指定UTX0表示的数字资产数额的值以及锁定脚本,该锁定脚本定义后续事务的输入202中的解锁脚本必须满足的条件,以使后续事务有效,从而成功赎回UTX0。通常情况下,锁定脚本将数额锁定至特定方(该数额的事务的受益人)。即,锁定脚本定义解锁条件,该解锁条件通常包括以下条件:后续事务的输入中的解锁脚本包括先前事务被锁定到的一方的加密签名。

[0405] 锁定脚本(亦称scriptPubKey)是节点协议识别的域特定语言中写入的一段代码。此类语言的特定示例称为“脚本(Script)”(S大写),其可由区块链网络所使用。锁定脚本指定花费事务输出203所需的信息,例如爱丽丝签名的要求。解锁脚本出现在事务的输出中。解锁脚本(亦称scriptSig)是提供满足锁定脚本标准所需信息的域特定语言中写入的一段代码。例如,其可包含鲍勃的签名。解锁脚本出现在事务的输入202中。

[0406] 因此在示出的示例中,Tx₀的输出203中的UTX0₀包括锁定脚本[Checksig P_A],该锁定脚本需要爱丽丝的签名Sig P_A,以赎回UTX0₀(严格来说,是为了使试图赎回UTX0₀的后续事务有效)。_A包含爱丽丝的公私密钥对中的公钥P_A的表示(即哈希)。Tx₁的输入202包括指向Tx₁的指针(例如,通过其事务ID(TxID₀),其在实施例中是整个事务Tx₀的哈希值)。Tx₁的输入202包括在Tx₀中标识UTX0₀的索引,以在Tx₀的任何其他可能输出中对其进行标识。Tx₁的输入202进一步包括解锁脚本<Sig P_A>,该解锁脚本包括爱丽丝的加密签名,

该签名由爱丽丝通过将其密钥对中的私钥应用于预定的部分数据(有时在密码学中称为“消息”)创建。爱丽丝需要签名以提供有效签名的数据(或“消息”)可通过锁定脚本、节点协议或其组合进行定义。

[0407] 当新事务 Tx_1 到达区块链节点104时,该节点应用节点协议。这包括一起运行锁定脚本和解锁脚本,以检查解锁脚本是否满足锁定脚本中定义的条件(其中该条件可包括一个或更多个标准)。在实施例中,这涉及并置两个脚本:

[0408] $\langle \text{Sig PA} \rangle \langle \text{PA} \rangle || [\text{Checksigs PA}]$

[0409] 其中“||”表示并置,“ $\langle \dots \rangle$ ”表示将数据放在堆栈上,“ $[\dots]$ ”表示由锁定脚本组成的函数(在该示例中指基于堆栈的语言)。同样,脚本可以使用公共堆栈一个接一个地运行,而不是并置脚本。无论采用哪种方式,当一起运行时,脚本使用爱丽丝的公钥 P_A (包括在 Tx_0 的输出的锁定脚本中),以认证 Tx_1 的输入中的解锁脚本是否包含爱丽丝签名预期部分的数据时的签名。也需要包括预期的部分数据本身(“消息”),以便执行此认证。在实施例中,签名的数据包括整个 Tx_1 (因此不需要包括一个单独的元素来明文指定签名的部分数据,因为其本身便已存在)。

[0410] 本领域技术人员将熟悉通过公私密码进行验证的细节。基本上而言,如果爱丽丝已使用其私钥加密签署消息,则给定爱丽丝的公钥和明文中的消息,诸如节点104等其它实体可验证消息必须已经由爱丽丝签名。签署通常包括对消息进行哈希,签署哈希值和将此标记到消息作为签名,从而使公钥的任何持有者能够验证签名。因此,应当注意的是,在实施例中,在本文中对签名特定数据片段或事务部分等的任何引用可以意味着对该数据片段或事务部分的哈希值进行签名。

[0411] 如果 Tx_1 中的解锁脚本满足 Tx_0 的锁定脚本中指定的一个或多个条件(因此,在所示示例中,如果在 Tx_1 中提供了爱丽丝的签名并进行验证),则区块链节点104认为 Tx_1 有效。这意味着区块链节点104会将 Tx_1 添加到待处理事务有序池154。区块链节点104还会将事务 Tx_1 转发到网络106中的一个或多个其它区块链节点104,以便其会在整个网络106中传播。一旦 Tx_1 有效并包括在区块链150中,这会将 $UTXO_0$ 从 Tx_0 定义为已花费。应当注意的是, Tx_1 仅在花费未花费事务输出203时才有效。如果其试图花费另一事务152已经花费的输出,则即使满足所有其它条件, Tx_1 也将无效。因此,区块链节点104还需要检查先前事务 Tx_0 中引用的 $UTXO$ 是否已经花费(即,其是否已经形成另一有效事务的有效输入)。这是为何区块链150对事务152施加定义的顺序很重要的原因之一。在实践中,给定区块链节点104可维护单独的数据库,标记已花费事务152的 $UTXO$ 203,但最终定义 $UTXO$ 是否已花费取决于是否在区块链150中形成了另一有效事务的有效输入。

[0412] 如果给定事务152的所有输出203中指定的总数额大于其所有输入202所指向的总数额,则这是大多数事务模型中的另一失效依据。因此,此类事务不会传播或包括在区块151中。

[0413] 请注意,在基于 $UTXO$ 的事务模型中,给定 $UTXO$ 需要作为一个整体使用。不能“留下” $UTXO$ 中定义为已花费的一部分数额,而同时又花费另一部分。但 $UTXO$ 的数额可以在后续事务的多个输出之间分割。例如, Tx_0 的 $UTXO_0$ 中定义的数额可以在 Tx_1 中的多个 $UTXO$ 之间分割。因此,如果爱丽丝不想将 $UTXO_0$ 中定义的所有数额都给鲍勃,她可以使用剩余部分在 Tx_1 的第二输出中自己找零,或者支付给另一方。

[0414] 在实践中,爱丽丝通常还需要包括用于比特币节点104的费用,该比特币节点104在区块151中成功包含爱丽丝的事务104。如果爱丽丝未包括此类费用,则 Tx_0 可能会被区块链节点104拒绝,并且因此尽管在技术上有效,但可能不会传播并且包括在区块链150中(如果区块链节点104不希望接受事务152,节点协议不强迫区块链节点104接受)。在一些协议中,事务费用不需要其自身的单独输出203(即不需要单独的UTXO)。相反,输入202指向的总数额与给定事务152的输出203指定的总数额之间的任何差额都将自动提供给发布事务的区块链节点104。例如,假设指向 $UTXO_0$ 的指针是 Tx_1 的唯一输入,并且 Tx_1 仅具有一个输出 $UTXO_1$ 。如果在 $UTXO_0$ 中指定的数字资产数额大于在 $UTXO_1$ 中指定的数额,则可以由赢得工作证明竞赛以创建包含 $UTXO_1$ 的区块的节点104分配该差值。替代地或附加地,这不一定排除可以在其自身事务152的其中一个UTXO 203中明确指定事务费用。

[0415] 爱丽丝和鲍勃的数字资产由区块链150中任何位置的任何事务152中的锁定至他们的UTXO组成。因此,通常情况下,给定方103的资产分散在整个区块链150的各种事务152的UTXO中。区块链150中的任何位置均未存储定义给定方103的总余额的一个数字。客户端应用程序105的钱包功能的作用是将锁定至相应方且在其它随后事务中尚未花费的各种UTXO值整理在一起。为实现这一点,其可以查询存储在任何一个比特币节点104处的区块链150的副本。

[0416] 应当注意的是,脚本代码通常用示意图表示(即使用非精确语言)。例如,可以使用操作码(opcode)来表示特定功能。“OP...”是指脚本语言的特定操作码。举例来说,OP_RETURN是脚本语言操作码,当在锁定脚本的开始处在操作码前加上OP_FALSE时,操作码创建事务的不可花费输出,该输出可以在事务内存储数据,从而将数据不可改变地记录在区块链150中。例如,数据可包括需存储在区块链中的文件。

[0417] 通常,事务的输入包含对应于公钥PA的数字签名。在实施例中,这基于使用椭圆曲线secp256k1的ECDSA。数字签名对特定的数据段进行签名。在实施例中,对于给定事务,签名将对部分事务输入以及部分或全部事务输出进行签名。对输出的特定部分进行签名取决于SIGHASH标志。SIGHASH标志通常是包含在签名末尾的4字节代码,用于选择签名的输出(并因此在签名时固定)。

[0418] 锁定脚本有时称为“scriptPubKey”,指其通常包括相应事务被锁定到的当事方的公钥。解锁脚本有时称为“scriptSig”,指其通常提供相应的签名。但是更通俗地说,在区块链150的所有应用中,UTXO赎回的条件并不一定包括对签名进行验证。更通俗地说,脚本语言可用于定义任何一个或多个条件。因此,可以优选更为通用的术语“锁定脚本”和“解锁脚本”。

[0419] 侧信道

[0420] 如图1所示,爱丽丝和鲍勃的计算机设备102a、120b中的每个计算机设备上的客户端应用程序都可以包括附加通信功能。此附加功能可使爱丽丝103a建立与鲍勃103b的单独侧信道107(在任何一方或第三方的鼓动下)。侧信道107使得能够脱离区块链网络交换数据。此类通信有时称为“链下”通信。例如,这可用于在爱丽丝与鲍勃之间交换事务152,而不将该事务(尚未)注册到区块链网络106上或将其发布到链150上,直到其中一方选择将其广播到网络106上。以这种方式共享事务有时称为共享“事务模板”。事务模板可能缺少形成完整事务所需的一个或多个输入和/或输出。替代地或附加地,侧信道107可用于交换任何其

它事务相关数据,例如密钥、议付数额或条款、数据内容等。

[0421] 通过与区块链网络106相同的分组交换网络101可建立侧信道107。替代地或附加地,侧信道301可以经由诸如移动蜂窝网络的不同网络或者诸如无线局域网的局域网建立,甚至经由爱丽丝和鲍勃的设备102a、102b之间的直接有线或无线链路建立。通常,在本文中任何地方所指的侧信道107可以包括经由一项或多项联网技术或通信介质的任何一条或多条链路,这些链路用于“链下”交换数据,即脱离区块链网络106交换数据。在使用多条链路的情况下,链下链路束或集合整体上可以称为侧信道107。因此,应当注意的是,如果说爱丽丝和鲍勃通过侧信道107交换某些信息或数据等,则这不一定意味着所有这些数据都必须通过完全相同的链路或甚至相同类型的网络发送。

[0422] 进一步评论

[0423] 一旦给出本文的公开内容,所公开技术的其它变体或用例对于本领域技术人员可能变得显而易见。本公开的范围不受所描述的实施例限制,而仅受随附权利要求限制。例如,上面的一些实施例已经根据比特币网络106、比特币区块链150和比特币节点104进行了描述。然而,应当理解的是,比特币区块链是区块链150的一个特定示例,并且上述描述通常可以应用于任何区块链。也就是说,本发明决不限于比特币区块链。更一般地,以上对比特币网络106、比特币区块链150和比特币节点104的任何引用可以分别参考区块链网络106、区块链150和区块链节点104来替换。区块链、区块链网络和/或区块链节点可以共享如上所述的比特币区块链150、比特币网络106和比特币节点104的部分或全部所述特性。

[0424] 在本公开的优选实施例中,区块链网络106是比特币网络,并且比特币节点104至少执行对区块链150的区块151进行创建、发布、传播和存储中的一些或所有的所述功能。不排除可能存在仅执行这些功能中的一个或部分功能但不是全部功能的其它网络实体(或网络元件)。也就是说,网络实体可以执行传播和/或存储区块的功能,而不创建和发布区块(请记住,这些实体不被认为是优选的比特币网络106的节点)。

[0425] 在本公开的其他实施例中,区块链网络106可以不是比特币网络。在这些实施例中,不排除节点可以执行对区块链150的区块151进行创建、发布、传播和存储中的至少一个或部分功能但不是所有功能。例如,在这些其它区块链网络上,“节点”可用于指被配置为创建和发布区块151但不存储和/或传播这些区块151到其它节点的网络实体。

[0426] 甚至更通俗地说,上面对术语“比特币节点”104的任何引用可以用术语“网络实体”或“网络元件”代替,其中这样的实体/元件被配置为执行对区块进行创建、发布、传播和存储中的一些或全部角色。这种网络实体/元件的功能可以在硬件中实现,方法与上面参照区块链节点104所述的方式相同。

[0427] 本文使用的术语“用户”可以包括人类和基于机器的实体。

[0428] 上述实施例说明而不是限制本公开,本领域技术人员能够在不脱离所附权利要求限定的本公开范围的前提下设计出许多替代实施例。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”。贯穿本说明书使用的词语“包括”或变体(诸如“包含”或“包括”)将被理解为意味着包含规定的元件、整数、步骤或元件、整数或步骤组,但不排除任何其他元件、整数或步骤或元件、整数或步骤组。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。

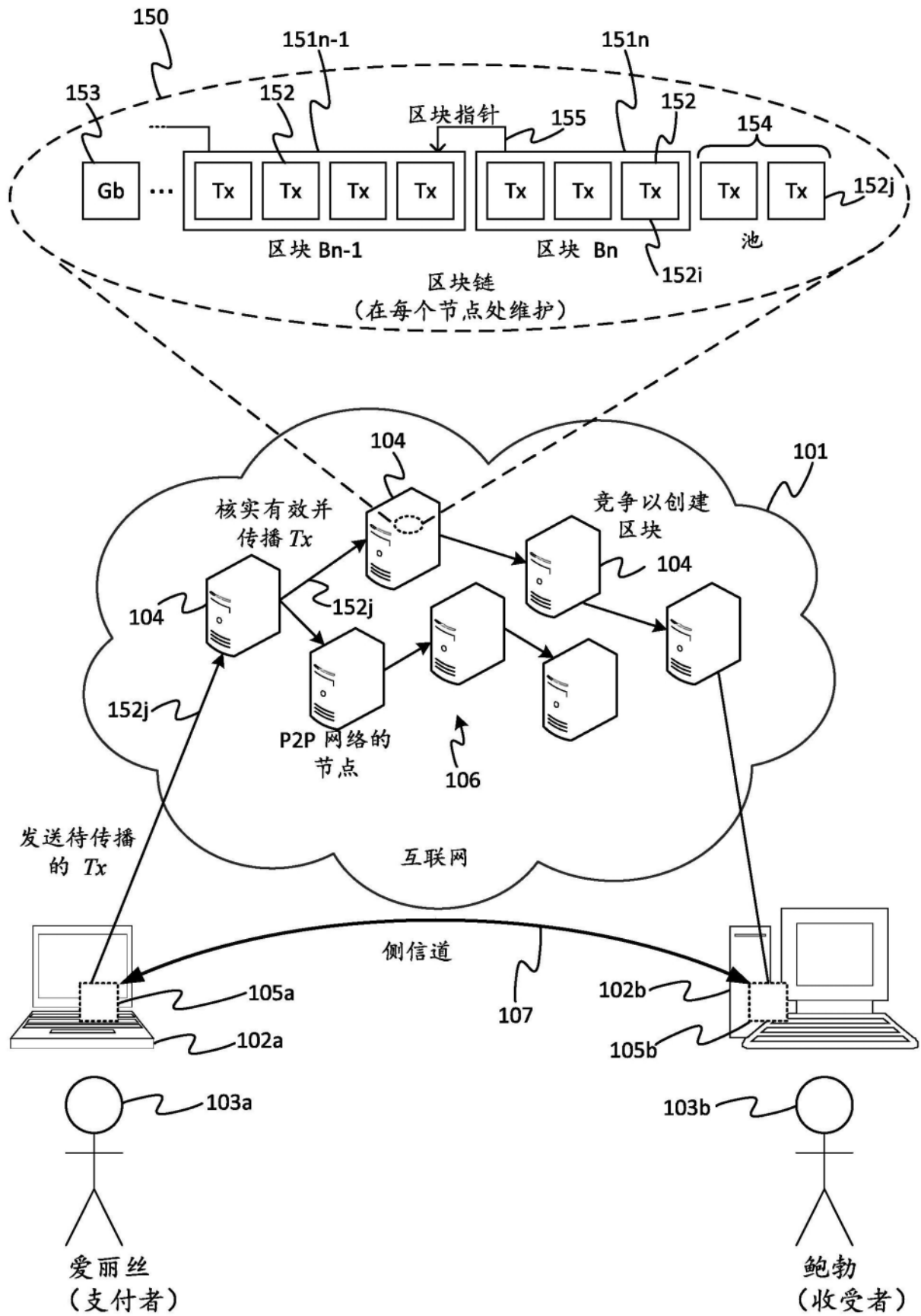
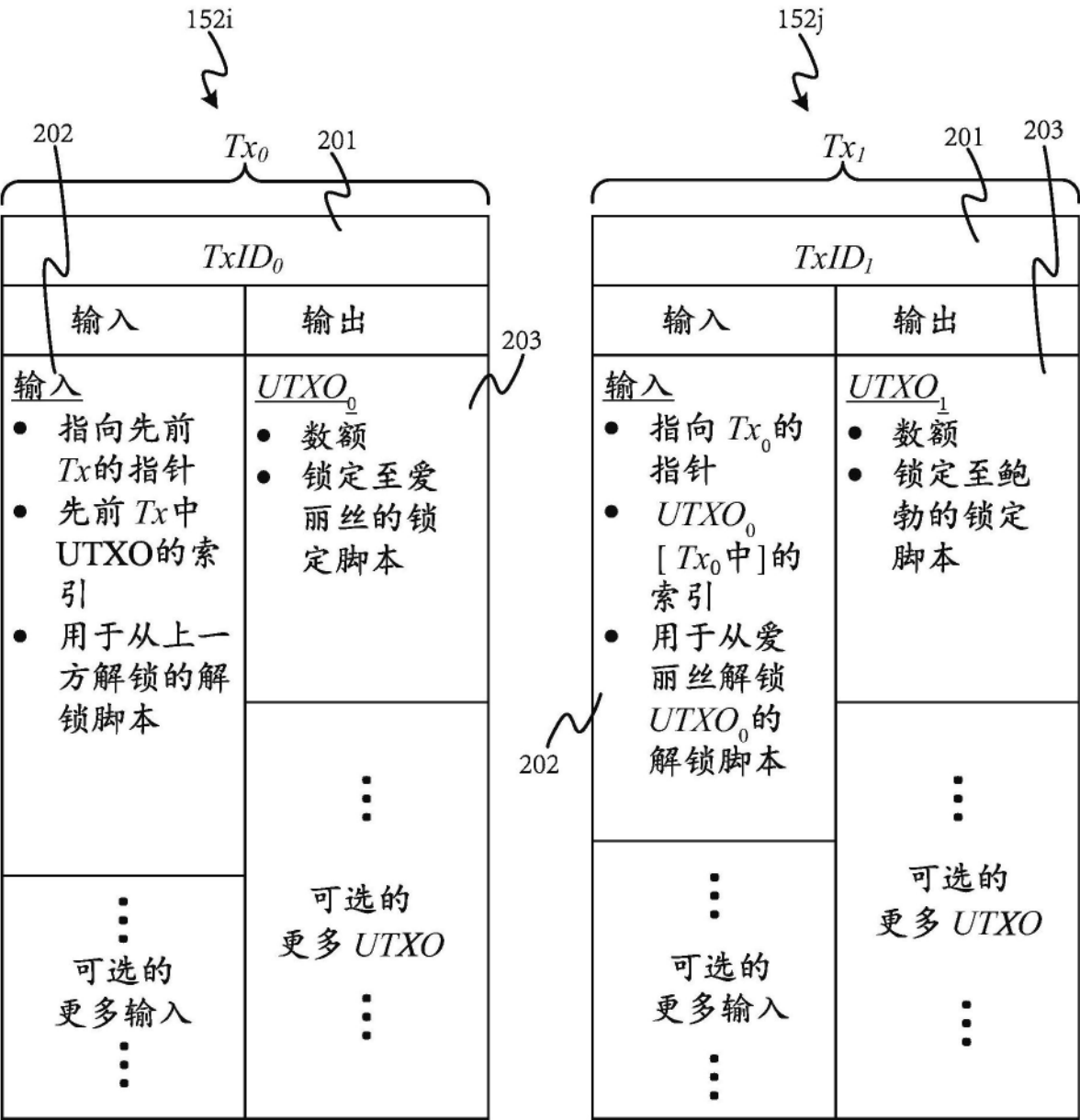


图1



通过运行：来自 Tx_0 的输出 203 的锁定脚本、和来自 Tx_1 的输入 202 的爱丽丝的解锁脚本来验证。此操作检查 Tx_1 中的爱丽丝的解锁脚本是否满足先前事务 Tx_0 中的锁定脚本中定义的条件。

图2

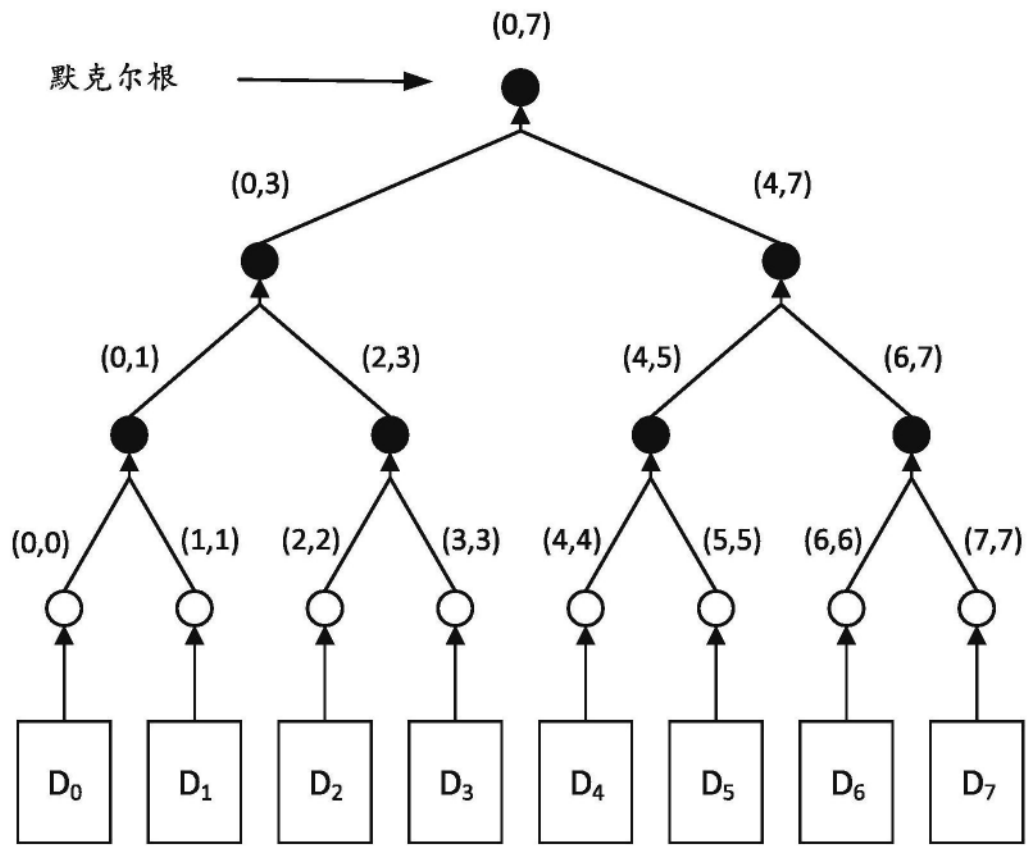


图3

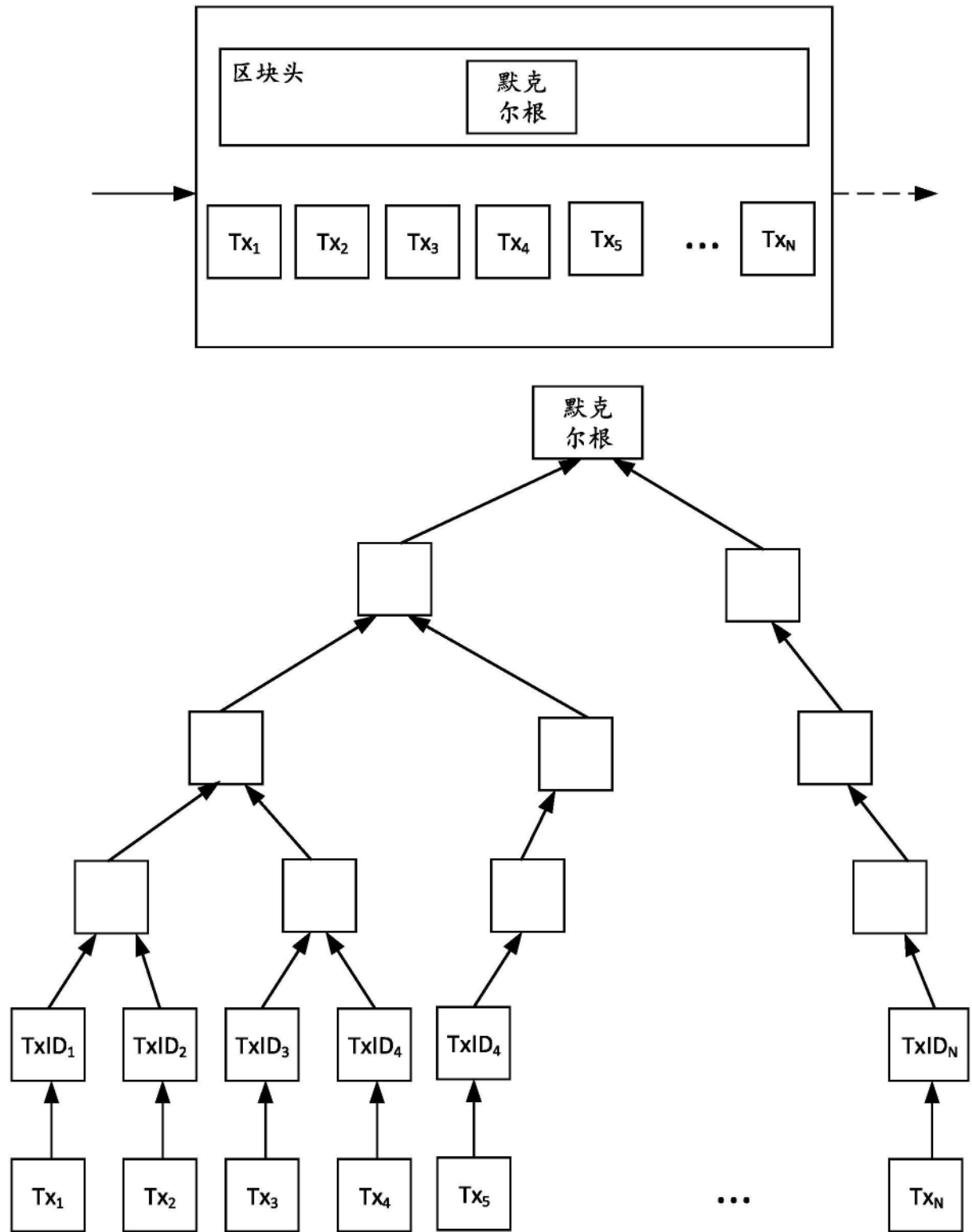


图4

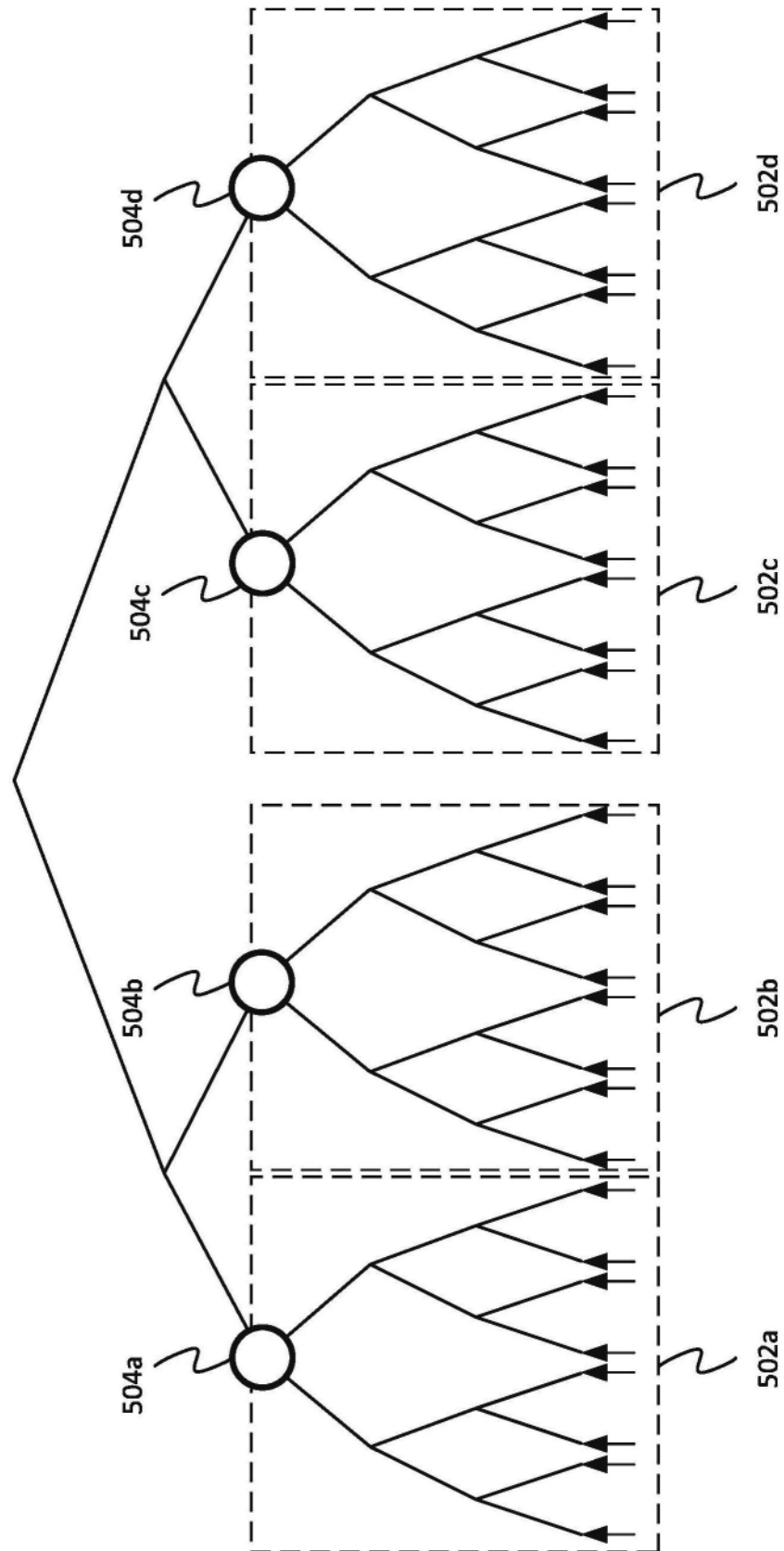


图5

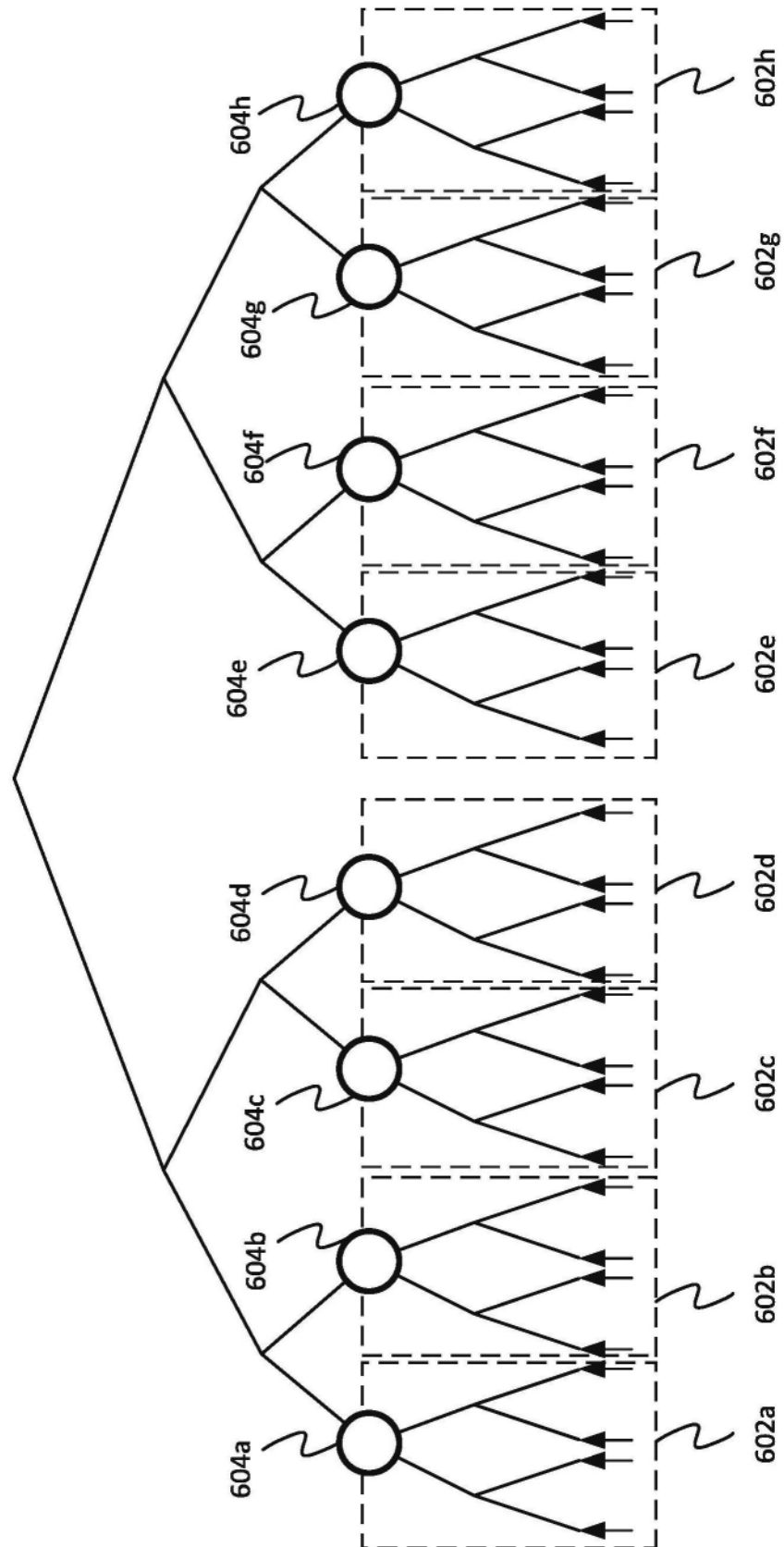


图6

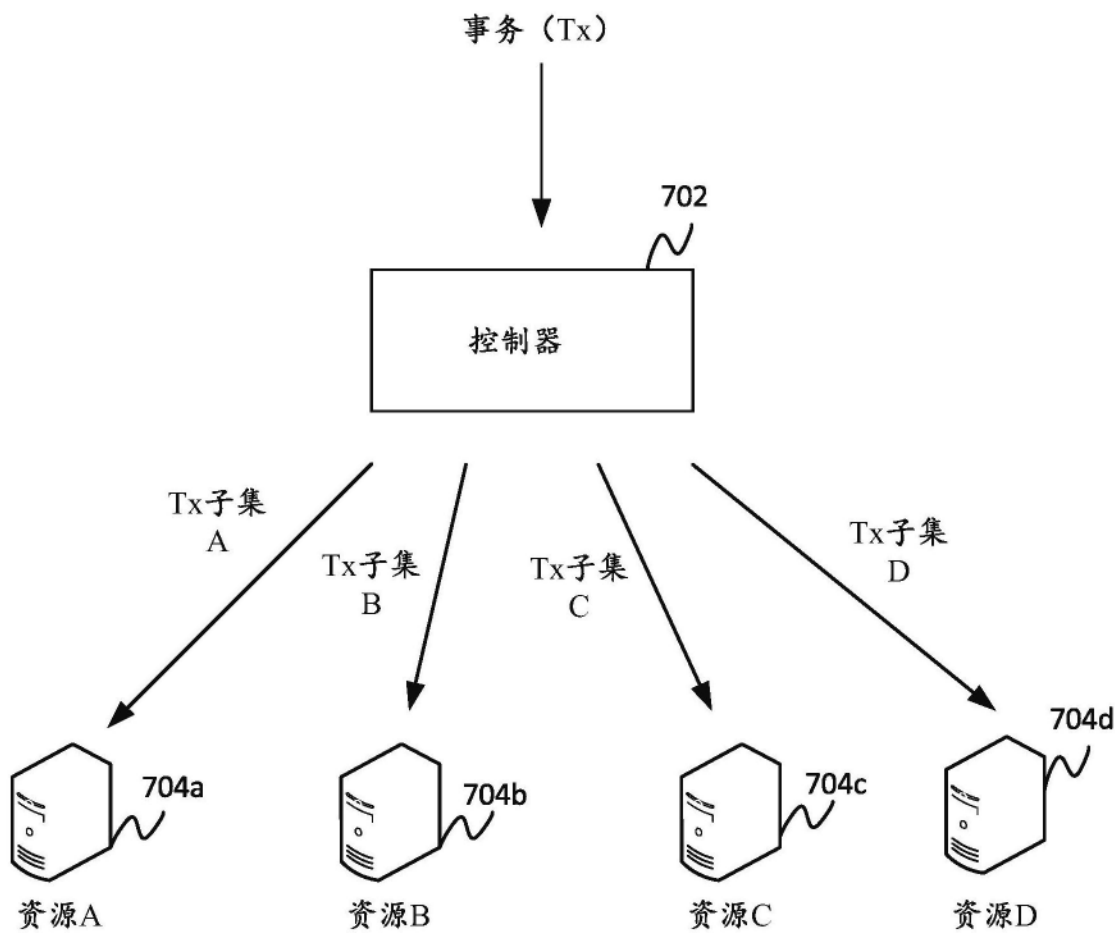


图7

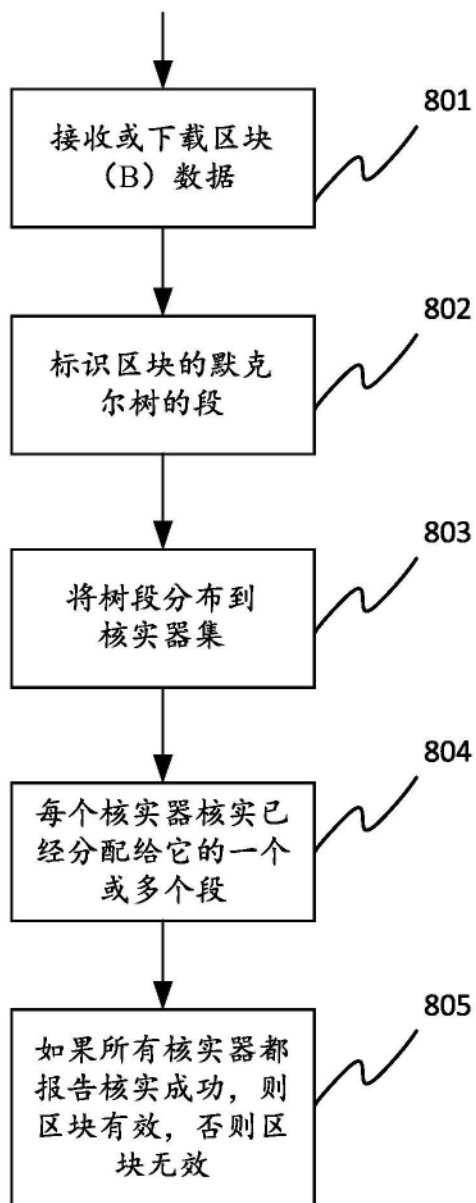


图8

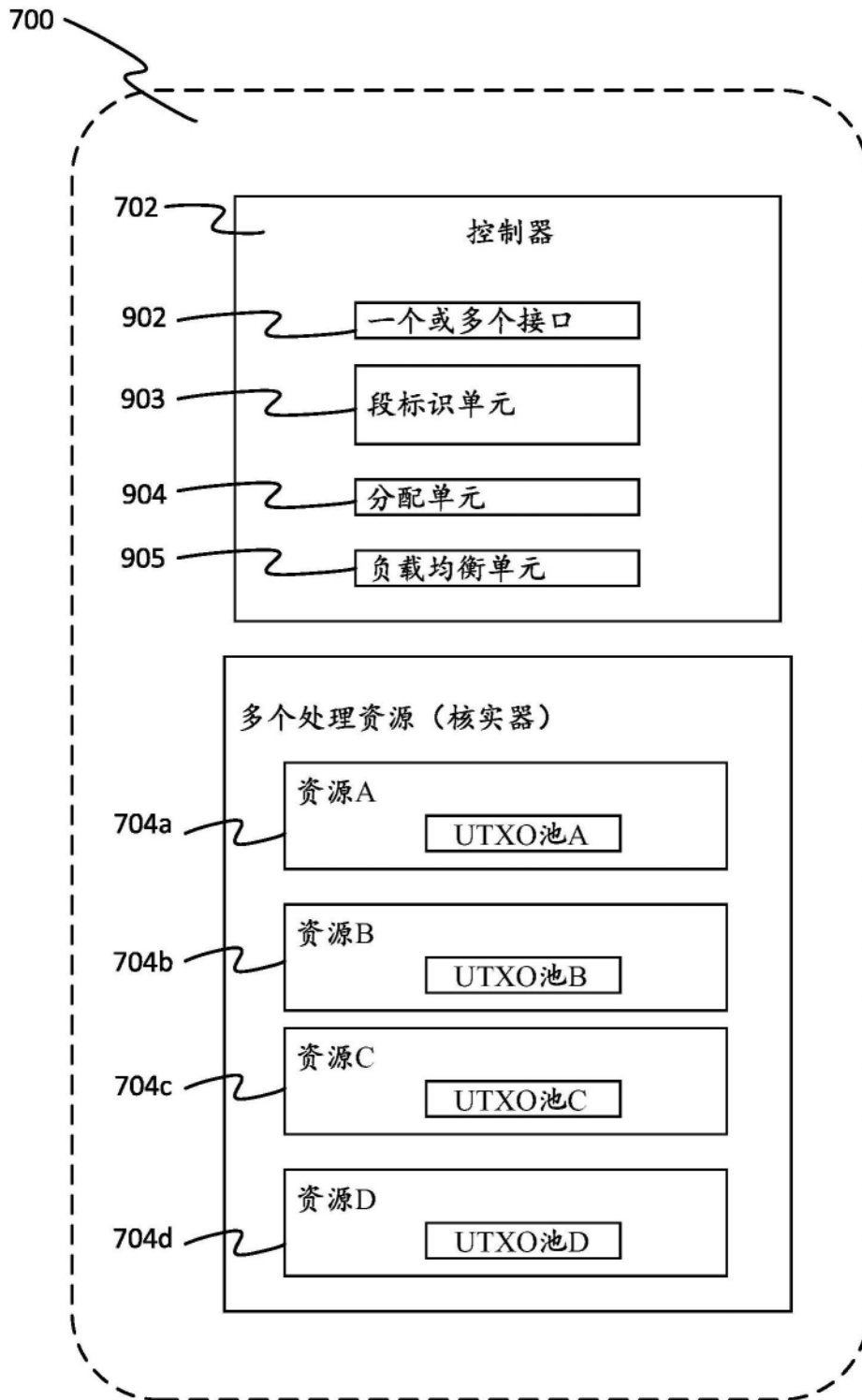


图9