



(12) 发明专利申请

(10) 申请公布号 CN 118044151 A

(43) 申请公布日 2024. 05. 14

(21) 申请号 202280065966.6

(22) 申请日 2022.08.30

(30) 优先权数据

2113977.9 2021.09.30 GB

(85) PCT国际申请进入国家阶段日

2024.03.28

(86) PCT国际申请的申请数据

PCT/EP2022/074013 2022.08.30

(87) PCT国际申请的公布数据

W02023/052019 EN 2023.04.06

(71) 申请人 区块链许可股份公司

地址 瑞士楚格

(72) 发明人 迈克尔拉·佩蒂特

史蒂文·帕特里克·考格兰

(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)

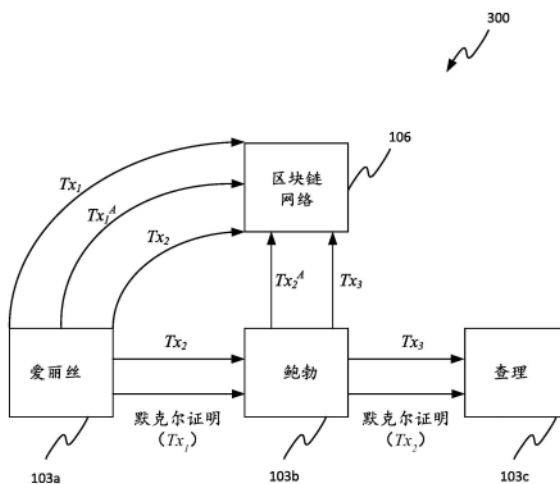
权利要求书4页 说明书34页 附图9页

(54) 发明名称

传播锁定脚本

(57) 摘要

一种强制锁定脚本在区块链事务链中传播的方法。所述方法包括生成包括主锁定脚本的主事务,所述主锁定脚本被配置为在由辅事务的解锁脚本执行时:验证所述解锁脚本包括基于所述辅事务生成的消息;以及基于所述消息来验证所述辅事务的输出包括辅锁定脚本。所述辅锁定脚本被配置为:验证第二主事务的解锁脚本包括基于所述第二主事务生成的消息;验证所述第二主事务的所述解锁脚本包括数据项,所述数据项包括所述主事务的一个或多个锁定脚本;以及基于所述消息和所述数据项来验证所述第二主事务的输出包括所述主锁定脚本。



1. 一种计算机实现的方法,用于强制锁定脚本在区块链事务链中传播,其中所述方法由第一方执行并且包括:

生成包括第一输出的第一主事务,其中所述第一输出包括主锁定脚本,其中所述主锁定脚本被配置为在由第一辅事务的第一解锁脚本执行时:i)验证所述第一解锁脚本包括基于所述第一辅事务而生成的第一消息,以及ii)基于所述第一消息来验证所述第一辅事务的第一输出包括辅锁定脚本;其中所述辅锁定脚本被配置为在由第二主事务的第一解锁脚本执行时:i)验证所述第二主事务的所述第一解锁脚本包括基于所述第二主事务而生成的第二消息,ii)验证所述第二主事务的所述第一解锁脚本包括第一数据项,所述第一数据项包括所述第一主事务的一个或多个锁定脚本,以及iii)基于所述第二消息和所述第一数据项来验证所述第二主事务的第一输出包括所述主锁定脚本。

2. 根据权利要求1所述的方法,所述方法包括:

生成所述第一辅事务,其中所述第一辅事务包括第一输入,所述第一辅事务的所述第一输入引用所述第一主事务的所述第一输出,并且包括所述第一消息,以及包含所述辅锁定脚本的第一输出。

3. 根据权利要求1或2所述的方法,其中所述主锁定脚本包括所述辅锁定脚本的哈希,其中所述第一消息包括候选辅锁定脚本,并且其中所述的验证所述第一辅事务的所述第一输出包括所述辅锁定脚本包括:所述主锁定脚本对所述候选辅锁定脚本进行哈希处理、并验证所述候选辅锁定脚本的哈希与所述辅锁定脚本的所述哈希匹配。

4. 根据前述任一项权利要求所述的方法,其中所述第一主事务包括第二输出,并且其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时:iii)验证所述第一主事务的所述第二输出由所述第二主事务的第二输入解锁。

5. 根据权利要求4所述的方法,其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时:iv)验证所述第二主事务的所述第一解锁脚本包括第二数据项,所述第二数据项包括所述第一辅事务;并且其中所述的验证所述第二输出由所述第二主事务的所述第二输入解锁包括:a)从所述第二消息中提取由所述第二主事务的所述第二输入解锁的所述第二输出的相应事务标识符,b)从所述第二数据项中提取由所述第一辅事务的所述第一输入解锁的所述第一输出的相应事务标识符,以及c)验证在步骤a)期间提取的所述相应事务标识符与在步骤b)期间提取的所述相应事务标识符匹配。

6. 根据前述任一项权利要求所述的方法,其中所述第一主事务具有通过对所述第一主事务应用哈希函数而生成的相应事务标识符,其中所述第二主事务的所述第一解锁脚本包括第三数据项,其中所述第三数据项包括对所述第一主事务应用的所述哈希函数的中间状态,并且其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时:iv)基于所述第一数据项和所述第三数据项来生成候选事务标识符,以及验证所述候选事务标识符与所述第一主事务标识符的所述相应事务标识符匹配。

7. 根据前述任一项权利要求所述的方法,其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时:v)验证所述第二主事务包括特定事务标识符。

8. 根据前述任一项权利要求所述的方法,所述方法包括:

生成所述第二主事务,其中所述第二主事务包括第一输入,所述第二主事务的所述第一输入引用所述第一辅事务的所述第一输出,并且包括所述第二消息和所述第一数据项,

并且其中所述第二主事务包括第一输出,所述第二主事务的所述第一输出包括所述主锁定脚本。

9.根据前述任一项权利要求所述的方法,所述方法包括:向第二方发送所述第一辅事务和所述第二主事务。

10.根据权利要求9所述的方法,所述方法包括:向所述第二方发送默克尔证明,用于证明所述第一主事务已经记录在区块链的区块中。

11.根据前述任一项权利要求所述的方法,其中所述主锁定脚本和/或辅锁定脚本包括与令牌的所有权相关的数据。

12.根据前述任一项权利要求所述的方法,其中所述第一主事务是所述区块链事务链中的初始主事务。

13.根据权利要求11和12所述的方法,其中所述第一方是所述令牌的发行者。

14.根据从属于权利要求7的权利要求12或13所述的方法,其中所述特定事务标识符是所述第一主事务的所述相应事务标识符。

15.根据前述任一项权利要求所述的方法,其中所述第一主事务包括第一计数器值,并且其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时:验证所述第二主事务包括第二计数器值,其中所述第二计数器值是所述计数器值的迭代。

16.根据权利要求1至11中任一项所述的方法,其中所述第一主事务不是所述区块链事务链中的初始事务,并且其中所述第一主事务包括第一输入,所述第一主事务的所述第一输入引用包含所述辅锁定脚本的先前辅事务的第一输出,其中所述先前辅事务包括第一输入,所述先前辅事务的所述第一输入引用包含所述主锁定脚本的先前主事务的第一输出,并且其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时:vi)验证所述第二主事务的所述第一解锁脚本包括第四数据项,所述第四数据项包括所述先前主事务的一个或多个锁定脚本,以及vii)基于所述第二消息和所述第四数据项来验证所述第二主事务的所述第一输出包括所述主锁定脚本。

17.根据权利要求16所述的方法,其中所述先前主事务具有通过对所述先前主事务应用哈希函数而生成的相应事务标识符,其中所述第二主事务的所述第一解锁脚本包括第五数据项,其中所述第五数据项包括对所述先前主事务应用的所述哈希函数的中间状态,并且其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时:viii)基于所述第四数据项和所述第五数据项来生成候选事务标识符,以及验证所述候选事务标识符与所述先前主事务标识符的所述相应事务标识符匹配。

18.根据权利要求16或17所述的方法,其中所述先前主事务是所述区块链事务链中的初始主事务,并且其中所述方法包括:验证所述初始主事务已经使用与预定公钥对应的签名进行签名。

19.根据前述任一项权利要求所述的方法,所述方法包括:向区块链网络提交所述第一主事务。

20.根据前述任一项权利要求所述的方法,所述方法包括:向区块链网络提交所述第一辅事务和/或所述第二主事务。

21.一种计算机实现的方法,用于验证锁定脚本已经在区块链事务链中被强制执行,其中所述方法由第二方执行并且包括:

获取第二主事务,其中所述第二主事务包括第一输入和第一输出,其中所述第一输入包括基于所述第二主事务而生成的第二消息、包含解锁第一主事务的第一输出的第一输入的第一辅事务、以及包含所述第一主事务的一个或多个锁定脚本的第一数据项,其中所述第一主事务的所述第一输出包括主锁定脚本,并且所述第一辅事务的第一输出包括辅锁定脚本,并且其中所述第二主事务的所述第一输出包括所述主锁定脚本;

验证第一主事务已经记录在区块链上;以及

对以下进行验证:所述第一主事务的所述主锁定脚本被配置为在由所述第一辅事务的第一解锁脚本执行时 i) 验证所述第一解锁脚本包括基于所述第一辅事务而生成的第一消息,和 ii) 基于所述第一消息来验证所述第一辅事务的第一输出包括辅锁定脚本;其中所述辅锁定脚本被配置为在由所述第二主事务的第一解锁脚本执行时 i) 验证所述第一解锁脚本包括所述第二消息, ii) 验证所述第一解锁脚本包括所述第一数据项,和 iii) 基于所述第二消息和所述第一数据项来验证所述第二主事务的所述第一输出包括所述主锁定脚本,从而验证所述主锁定脚本已经在所述第一主事务和所述第二主事务中强制执行。

22. 根据权利要求21所述的方法,其中所述主锁定脚本包括所述辅锁定脚本的哈希,其中所述第一消息包括候选辅锁定脚本,并且其中所述主锁定脚本被配置为在由所述第一辅事务的所述第一解锁脚本执行时:通过对所述候选辅锁定脚本进行哈希处理并验证所述候选辅锁定脚本的哈希与所述辅锁定脚本的所述哈希匹配,来验证所述第一辅事务的所述第一输出包括所述辅锁定脚本,并且其中所述的验证所述主锁定脚本被配置为执行步骤 i) 和 ii) 包括:验证所述主锁定脚本包括所述辅锁定脚本的所述哈希作为哈希谜题的一部分。

23. 根据权利要求21或22所述的方法,所述的验证所述第一主事务已经记录在所述区块链上包括:获取默克尔证明,用于证明所述第一主事务已经记录在所述区块链的区块中。

24. 根据权利要求23所述的方法,其中所述第一主事务具有通过对所述第一主事务应用哈希函数而生成的相应事务标识符,其中所述第二主事务的所述第一输入包括第三数据项,其中所述第三数据项包括对所述第一主事务应用的所述哈希函数的中间状态,并且其中所述的验证所述第一主事务已经记录在所述区块链上包括:基于所述第一数据项和所述第三数据项来生成候选事务标识符;以及验证所述默克尔证明对所述候选事务标识符有效。

25. 根据权利要求23或24所述的方法,所述方法包括:获取所述第一主事务,并且其中所述的验证所述第一主事务已经记录在所述区块链基于所述获取的默克尔证明和所述获取的第一主事务。

26. 根据权利要求19或其任何从属权利要求所述的方法,所述方法包括:验证所述第二主事务包括初始主事务的所述相应事务标识符。

27. 根据权利要求19或其任何从属权利要求所述的方法,所述方法包括:

生成第二辅事务,其中所述第二辅事务包括第一输入,所述第二辅事务的所述第一输入引用所述第二主事务的所述第一输出,并且包括基于所述第二辅事务而生成的第三消息,以及包含所述辅锁定脚本的第一输出;以及

生成第三主事务,其中所述第三主事务包括第一输入,所述第三主事务的所述第一输入引用所述第二辅事务的所述第一输出,并且包括基于所述第三主事务而生成的第四消息、所述第一数据项、所述第二辅事务、和包含所述第二主事务的一个或多个锁定脚本的第

六数据项,并且其中所述第三主事务包括第一输出,所述第三主事务的所述第一输出包括所述主锁定脚本。

28.根据权利要求27所述的方法,其中所述辅锁定脚本被配置为在由所述第三主事务的第一解锁脚本执行时:vi)验证所述第二主事务的所述第一解锁脚本包括所述第三数据项,以及vii)基于所述第三消息和所述第六数据项来验证所述第三主事务的所述第一输出包括所述主锁定脚本。

29.根据权利要求27或28所述的方法,所述方法包括:向第三方发送所述第二辅事务和所述第三主事务。

30.根据权利要求29所述的方法,所述方法包括:向所述第三方发送默克尔证明,用于证明所述第二主事务已经记录在所述区块链的区块中。

31.根据权利要求21或其任何从属权利要求所述的方法,其中所述主锁定脚本和/或辅锁定脚本包括与令牌的所有权相关的数据,并且其中所述第一主事务和/或所述第一辅锁定脚本已经由令牌发行者生成。

32.根据权利要求27或其任何从属权利要求所述的方法,所述方法包括:向所述区块链提交所述第二辅事务和所述第三主事务。

33.一种计算机设备,所述计算机设备包括:

存储器,所述存储器包括一个或多个存储器单元;以及

处理装置,所述处理装置包括一个或多个处理单元,其中所述存储器存储被设置在所述处理装置上运行的代码,所述代码被配置为当在所述处理装置上运行时,执行根据权利要求1至32中任一项所述的方法。

34.一种计算机程序,所述计算机程序包含在计算机可读存储器上并且被配置为当在一个或多个处理器上运行时,执行根据权利要求1至32中任一项所述的方法。

传播锁定脚本

技术领域

[0001] 本公开涉及一种强制(enforce)锁定脚本在区块链事务链中传播的方法,以及一种验证锁定脚本已经在区块链事务链中强制执行(enforce)的方法。

背景技术

[0002] 区块链是指一种分布式数据结构,其中在分布式对等(P2P)网络(以下称为“区块链网络”)中的多个节点中的每个节点处维护区块链的副本,并且广泛公开该副本。区块链包括一系列数据区块,其中每个区块包括一个或多个事务(transaction)。除所谓的“coinbase事务”外,每个事务都指向序列中的先前事务,该序列可以跨越一个或多个区块,回到一个或多个coinbase事务。coinbase事务将在下文进一步讨论。提交给区块链网络的事务包括在新区块中。新区块的创建过程通常称为“挖掘”,该过程涉及多个节点中的每个节点争相执行“工作证明”,即,基于等待被包括在区块链的新区块中的一组定义的有序且核实有效的未决事务的表示解决加密难题。应当注意的是,区块链可以在一些节点处被修剪(prune),并且区块的发布可以通过仅发布区块头来实现。

[0003] 区块链中的事务可用于以下目的中的一个或多个:传送数字资产(即,一定数量的数字令牌);对虚拟化分类账或注册表中的一组条目进行排序;接收和处理时间戳条目;和/或对索引指针按时间排序。也可利用区块链实现区块链上的层级附加功能。例如,区块链协议可允许在事务中存储附加的用户数据或数据索引。能够存储在单个事务中的最大数据容量没有预先指定的限制,因此可以并入越来越复杂的数据。例如,这可用于在区块链中存储电子文档、音频或视频数据。

[0004] 区块链网络的节点(通常称为“矿工”)执行分布式事务注册和验证过程,这将后续更详细地描述。总之,在该过程中,节点核实事务并将这些事务插入到区块模板中,这些事务尝试为该区块模板标识有效的工作证明解。一旦找到有效的解,新区块便会被传播到网络的其它节点,从而使得每个节点能够在区块链上记录新区块。为了将事务记录在区块链中,用户(例如,区块链客户端应用程序)将该事务发送到网络中的节点中的一个节点进行传播。接收该事务的节点可以争相寻找将核实有效的事务并入新区块的工作证明解。每个节点被配置为执行相同的节点协议,该协议将包括用于确认事务有效的一个或多个条件。无效事务将不会传播或并入到区块中。假定事务已经核实有效,从而在区块链上被接受,则该事务(包括任何用户数据)将因此在区块链网络中的每个节点上作为不可改变的公共记录进行注册和索引。

[0005] 成功解决工作证明难题可创建最新区块的节点通常被奖励一个称为“coinbase事务”的新事务,该事务分发数字资产数额,即令牌数量。无效事务的检测和拒绝是通过竞争节点的行动来执行的,这些竞争节点充当网络的代理并且通过激励报告和阻止不正当行为。信息的广泛发布使得用户可以连续地审计节点的性能。仅发布区块头使得参与者可以确保区块链具有持续完整性。

[0006] 在“基于输出的”模型(有时称为基于UTX0的模型)中,给定事务的数据结构包括一

个或多个输入和一个或多个输出。任何可花费输出包括指定数字资产数额的元素,该元素可从进行中的事务序列导出。可花费输出有时称为UTX0(“未花费事务输出”)。输出还可以包括锁定脚本,该锁定脚本指定输出的未来赎回条件。锁定脚本是限定核实和传送数字令牌或资产所必需的条件的谓词。事务(除coinbase事务之外)的每个输入包括指向先前事务中的此类输出的指针(即引用),并且还可以包括解锁脚本,用于解锁指向输出的锁定脚本。因此,考虑一对事务,将其称为第一事务和第二事务(或“目标”事务)。第一事务包括指定数字资产数额的至少一个输出,并且包括定义解锁该输出的一个或多个条件的锁定脚本。第二目标事务包括至少一个输入和解锁脚本,该至少一个输入包括指向第一事务的输出的指针;该解锁脚本用于解锁第一事务的输出。

[0007] 在此类模型中,当第二目标事务被发送到区块链网络以在区块链中传播和记录时,在每个节点处应用的有效性条件之一将是解锁脚本满足在第一事务的锁定脚本中定义的一个或多个条件中的所有条件。另一条件将是第一事务的输出尚未被另一早期有效事务赎回。根据这些条件中的任何一个条件发现目标事务无效的任何节点都不会传播该事务(作为有效事务,但可能注册无效事务),也不将该事务包括在要记录在区块链中的新区块中。

[0008] 另一种事务模型是基于账户的模型。在这种情况下,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节点单独存储到区块链中,并不断更新。

发明内容

[0009] 众所周知,可以在花费事务(即,花费(解锁、分配、转移等)先前事务的输出的事务)的字段上强制执行条件。例如,所述先前事务可以包括对所述花费事务的一个或多个输出施加条件的锁定脚本。应当注意的是,在本领域中使用的术语“花费事务”是指解锁至少一个先前事务的至少一个输出的当前事务,并且不一定意味着所述当前事务与金融事务相关。

[0010] 想要对花费事务的字段强制执行条件的一个原因在于,确保所述花费事务具有包括与所述先前事务相同的锁定脚本的输出。通过这种方式,可以确保所述花费事务对后续花费事务强制执行相同的条件。也就是说,第n-1个事务包括锁定脚本,所述锁定脚本强制第n个事务包括相同的锁定脚本,从而强制第n+1个事务包括相同的锁定脚本。通过这种方式,创建事务链,由此每个事务包括相同的锁定脚本。这可以在数字令牌(token)的环境中使用,该数字令牌(例如)表示真实世界对象的、甚至虚拟世界中的对象的所有权。这是有利的,因为这意味着令牌的每次转移遵循相同的规则。

[0011] 强制事务链包括相同的锁定脚本的先前尝试至少遇到以下问题中的至少一个问题。首先,一些尝试要求验证者追溯到所述链中的第一个事务(例如,“发行事务”),以确保所述第一个事务(例如,由特定的授权机构,诸如令牌发行者)正确地创建,或者最近事务中包括的锁定脚本与所述第一个事务和所述链中的每个其他事务中包括的锁定脚本相同。这会消耗所述验证者的资源(在计算资源、时间和精力方面)来核实所述链中的最新事务。其次,一些尝试要求第三方参与创建和/或核实所述链中的所述后续事务(例如,转移令牌),以确保正确地创建所述后续事务。这引入了依赖第三方的安全弱点,并增加了必须相互交

互的各方数量,从而降低了效率。第三,强制事务链包括相同的锁定脚本的一些尝试要求所述链中的最新事务包括所述链中的每个先前事务,从而引入事务膨胀问题。这对于事务的传输和存储都是个问题。第四,一些尝试容易受到重放攻击,下面将对此进行讨论。

[0012] 因此,希望实现一种机制,所述机制强制事务链包括解决所识别问题中的一个、一些或全部问题的相同锁定脚本。

[0013] 根据本文公开的一个方面,提供了一种强制锁定脚本在区块链事务链中传播的计算机实现的方法。所述方法由第一方执行并且包括:生成包括第一输出的第一主事务,其中所述第一输出包括主锁定脚本,其中所述主锁定脚本被配置为在由第一辅事务的第一解锁脚本执行时:i)对于所述第一解锁脚本包括第一消息进行验证,所述第一消息是基于所述第一辅事务而生成的,以及ii)基于所述第一消息来验证所述第一辅事务的第一输出包括辅锁定脚本;其中所述辅锁定脚本被配置为在由第二主事务的第一解锁脚本执行时:i)对于所述第二主事务的所述第一解锁脚本包括第二消息进行验证,所述第二消息是基于所述第二主事务而生成的,ii)验证所述第二主事务的所述第一解锁脚本包括第一数据项,所述第一数据项包括所述第一主事务的一个或多个锁定脚本,以及iii)基于所述第二消息和所述第一数据项,来验证所述第二主事务的第一输出包括所述主锁定脚本。

[0014] 在本文中使用的术语“主事务”是指事务链中被强制包括相同锁定脚本的事务,所述锁定脚本称为“主锁定脚本”。例如,所述主锁定脚本可以包括与令牌相关的数据,例如财产契据。术语“辅事务”用于指代用于对所述主事务强制执行条件的事务,即所述主事务必须包括所述主锁定脚本。这些条件至少部分地由每个辅事务中包括的“辅锁定脚本”强制执行。

[0015] 所述辅事务充当连续主事务之间的桥梁。也就是说,第一辅事务花费(即,解锁)第一主事务的输出,并且第二主事务花费(即,解锁)所述第一辅事务的输出。所述主锁定脚本对后续辅事务施加条件。例如,所述主锁定脚本强制所述后续辅事务包括所述辅锁定脚本。所述辅锁定脚本对所述后续主事务施加条件。例如,所述辅锁定脚本强制所述后续主事务包括所述主锁定脚本。

[0016] 在一些示例中,所述第一主事务是所述链中的初始主事务,即包括所述主锁定脚本的第一个事务。例如,这可以由发行令牌的令牌发行者创建。在其他示例中,所述第一主事务不是所述初始主事务,即在包括所述主锁定脚本的所述事务链中存在至少一个其他先前主事务。例如,在该示例中,所述第一主事务可以由所述令牌的转让者创建。

[0017] 根据本文公开的另一方面,提供了一种验证锁定脚本已经在区块链事务链中被强制执行的计算机实现的方法。所述方法由第二方执行并且包括:获取第二主事务,其中所述第二主事务包括第一输入和第一输出,其中所述第一输入包括基于所述第二主事务生成的第二消息,第一辅事务,所述第一辅事务包括解锁第一主事务的第一输出的第一输入,以及第一数据项,所述第一数据项包括所述第一主事务的一个或多个锁定脚本,其中所述第一主事务的所述第一输出包括主锁定脚本,并且所述第一辅事务的第一输出包括辅锁定脚本,并且其中所述第二主事务的所述第一输出包括所述主锁定脚本。所述方法还包括:验证第一主事务已经记录在区块链上;以及,对以下进行验证:所述第一主事务的所述主锁定脚本被配置为在由所述第一辅事务的第一解锁脚本执行时i)验证所述第一解锁脚本包括基于所述第一辅事务而生成的第一消息,和ii)基于所述第一消息来验证所述第一辅事务的

第一输出包括辅锁定脚本；其中所述辅锁定脚本被配置为在由所述第二主事务的第一解锁脚本执行时 i) 验证所述第一解锁脚本包括所述第二消息, ii) 验证所述第一解锁脚本包括所述第一数据项, 和 iii) 基于所述第二消息和所述第一数据项来验证所述第二主事务的所述第一输出包括所述主锁定脚本, 从而验证所述主锁定脚本已经在所述第一主事务和所述第二主事务中强制执行。

[0018] 所述第二方获取(例如, 从所述第一方接收)所述第二主事务。所述第二主事务可以是所述事务链中第二个出现的主事务, 也可以是后续事务。也就是说, 所述第二事务通常可以是所述事务链中的第n个事务。所述第二主事务在输出中包括所述主锁定脚本。所述第二主事务具有花费第一辅事务的输出的输入, 并且因此满足所述辅锁定脚本的要求。所述第一辅事务花费第一主事务的输出, 并且因此满足所述主锁定脚本的要求。因此, 如果所述第一辅事务成功地解锁所述第一主事务的所述主锁定脚本, 并且所述第二主事务成功地解锁所述第一辅事务的所述辅锁定脚本, 则所述第二主事务必须包括所述主锁定脚本。这取决于所述事务链中包括的正确主锁定脚本。

[0019] 所述第二方验证所述第一主事务已经记录在所述区块链上, 并且因此是有效事务。所述第二方还验证所述主锁定脚本是正确的, 例如它强制所述辅事务包括所述辅锁定脚本, 所述辅锁定脚本然后强制所述后续主事务包括相同的主锁定脚本。例如, 这可以通过确保所述主锁定脚本包括基于所述辅锁定脚本的哈希的哈希谜题来实现。在这种情况下, 只有当所述辅事务包括所述辅锁定脚本时, 才能解决所述哈希谜题(并且因此解锁所述主锁定脚本)。

[0020] 然后, 所述第二方可以生成第二辅事务和第三主事务。需要所述第二辅事务来解锁所述第二主事务的所述主锁定脚本。所述第二辅事务还强制所述第三主事务包括所述主锁定脚本。所述链中的每个后续事务都被强制以相同的方式包括所述主锁定脚本。

[0021] 应当注意的是, 实际上会强制传播两个锁定脚本。所述主锁定脚本被强制包括在每个主事务的输出中, 并且所述辅锁定脚本被强制包括在每个辅事务的输出中。

附图说明

[0022] 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:

[0023] 图1是一种用于实现区块链的系统的示意性框图;

[0024] 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;

[0025] 图3示意性地示出了用于传播锁定脚本的示例性系统;

[0026] 图4是示例性令牌协议的事务结构的示意图;

[0027] 图5是具有一个输入和一个输出的事务链的示意图;

[0028] 图6示意性地示出了事务膨胀问题;

[0029] 图7是具有两个输入和两个输出的事务链的示意图;

[0030] 图8是示出关于如何根据示例性令牌协议来防止事务膨胀的示意图;

[0031] 图9示意性地示出了示例性发行事务;

[0032] 图10示意性地示出了示例性辅事务;

[0033] 图11示意性地示出了将令牌从爱丽丝转移到鲍勃的示例性主事务;

- [0034] 图12示意性地示出了另一辅事务的示例；
- [0035] 图13示意性地示出了将令牌从鲍勃转移到查理的主事务的示例；
- [0036] 图14示意性地示出了第n个主事务的示例性格式；
- [0037] 图15示意性地示出了第n个辅事务的示例性格式。

具体实施方式

[0038] 1. 示例性系统概述

[0039] 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。

[0040] 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。

[0041] 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。

[0042] 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。

[0043] 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。

[0044] 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”。花费或赎回并不一定意味着转移金融资产,尽管这肯定是一种常见的应用。更一般地,可以将花费描述为消耗输出,或者将其分配给另一个后续事务中的一个或多个输出。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。

[0045] 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。

[0046] 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花费(或“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。

[0047] 在基于输出的模型中,给定输出(例如,UTXO)是否分配(或“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。

[0048] 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。

[0049] 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花费或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。

[0050] 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。

[0051] 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。

[0052] 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。

[0053] 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。

[0054] 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。

[0055] 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。

[0056] 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。

[0057] 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。

[0058] 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。

[0059] 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。

[0060] 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,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。

[0061] 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。

[0062] 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。

[0063] 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。

[0064] 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)。

[0065] 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。

[0066] 2. 基于UTX0的模型

[0067] 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。

[0068] 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令牌。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。

[0069] 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“Tx_j”。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Tx_j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Tx_j指池154中的后续事务。Tx_j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。

[0070] 当爱丽丝创建其新事务Tx_j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Tx_j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Tx_j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。

[0071] 先前事务 Tx_0 的一个或多个输出203中的一个包括特定的UTXO,标记为 $UTXO_0$ 。每个UTXO包括指定UTXO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UTXO。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。

[0072] 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。

[0073] 因此在示出的示例中, Tx_0 的输出203中的 $UTXO_0$ 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 $UTXO_0$ (严格来说,是为了使试图赎回 $UTXO_0$ 的后续事务有效)。 $[Checksig P_A]$ 包含爱丽丝的公私密钥对中的公钥 P_A 的表示(即哈希)。 Tx_1 的输入202包括指向 Tx_1 的指针(例如,通过其事务ID($TxID_0$),其在实施例中的是整个事务 Tx_0 的哈希值)。 Tx_1 的输入202包括在 Tx_0 中标识 $UTXO_0$ 的索引,以在 Tx_0 的任何其他可能输出中对其进行标识。 Tx_1 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。

[0074] 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多个标准)。在实施例中,这涉及并置两个脚本:

[0075] <Sig PA >< PA >||[Checksig PA]

[0076] 其中“||”表示并置,“<...>”表示将数据放在堆栈上,“[...]”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)。

[0077] 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。

[0078] 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 添加到待订事务有序池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。

[0079] 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。

[0080] 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。

[0081] 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(或花费)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。

[0082] 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。

[0083] 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。

[0084] 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)。

[0085] 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”。

[0086] 3. 侧信道

[0087] 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。

[0088] 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。

[0089] 4. 强制传播锁定脚本

[0090] 图3示出了用于强制传播锁定脚本的示例性系统300,即强制事务链中的每个事务包括相同的锁定脚本。系统300包括第一方(例如,爱丽丝103a)和第三方(例如,鲍勃103b)。系统300还可以包括一个或多个附加方,例如查理103c。系统300还包括区块链网络106的一个或多个区块链节点104。尽管在下面的描述中将第一方和第三方称为爱丽丝103a和鲍勃103b,但是应当注意的是,这仅仅是为了方便。也就是说,第一方和第三方不必被配置为执行上面参考图1和图2描述的由爱丽丝103a和鲍勃103b执行的一些或全部动作,尽管这当然是一种选择。更一般地,第一方和第三方两者可以被配置为执行上面参考图1和图2描述的由爱丽丝103a和/或鲍勃103b执行的一些或全部动作。应当理解的是,被描述为由爱丽丝103a、鲍勃103b或查理103c执行的任何操作都由他们的相应计算设备执行。

[0091] 本发明实施例可以用于确保锁定脚本在区块链事务链中传播。通常,协议依赖于两种类型的事务:主事务(primary transaction)和辅事务(auxiliary transaction)。每个主事务包括相应主锁定脚本,该相应主锁定脚本对尝试解锁相应主锁定脚本的相应辅事务强制执行条件。每个主锁定脚本都是相同的,即采用相同的格式,包含相同的数据,并实现相同的功能。辅事务各自包括相应辅锁定脚本,该相应辅锁定脚本对尝试解锁相应辅锁定脚本的相应主事务强制执行条件。每个辅锁定脚本都是相同的。

[0092] 第一方爱丽丝103a可以负责生成发行事务。发行事务是事务链中的初始主事务。发行事务也可以称为第一主事务。在主事务包含令牌相关数据的示例中,爱丽丝103a可以是令牌的发行者。在其他示例中,不同的一方可以发起令牌链。

[0093] 第一主事务包括一个或多个输入以及一个或多个输出。例如,主事务的输入可以用于解锁爱丽丝的先前事务中的一个先前事务的输出,并且因此可以包括执行此操作所需的数据,例如爱丽丝的签名和公钥。第一主事务的输出(例如,第一输出)包括主锁定脚本,即将在整个未来事务中传播的相同主锁定脚本的实例。第一主事务的另一输出(例如,第二输出)可以指定花费条件,例如它可以被锁定到由爱丽丝103a或不同方拥有的公钥。例如,第二输出可以包括支付到公钥哈希(P2PKH)输出。

[0094] 主锁定脚本被配置为对尝试解锁主锁定脚本的事务强制执行若干条件,在这种情况下,该事务是第一辅事务。对辅事务强制执行的条件中的一个条件是,第一辅事务的解锁脚本必须包含基于第一辅事务本身的第一消息。也就是说,第一消息基于第一辅事务的一个、一些或全部字段。一些字段可能采用它们的原始形式,而另一些可能经过混淆处理,例如经过哈希处理。一种用于强制解锁脚本包含此类消息的技术在本领域中称为PUSHTX,它是伪操作码,即被配置为执行特定功能的操作码的组合。PUSHTX的描述如下:<https://xiaohuiliu.medium.com/op-push-tx-3d3d279174c1>。在英国专利申请GB2112930.9中描述了另一个示例。强制解锁脚本包含第一消息允许对第一辅事务进行某些检查。该消息有时称为sighash原像,因为它是作为区块链签名协议的一部分在签名之前进行哈希处理的原像。换言之,强制解锁脚本包含第一消息允许对第一辅事务强制执行某些条件。强制执行的一个条件是,第一辅事务的输出(例如,第一输出)包含辅锁定脚本(如下所述)。

[0095] 作为关于如何强制执行该条件的示例,主锁定脚本可以包括被配置为基于辅锁定脚本的哈希来实现哈希谜题的子脚本。也就是说,哈希谜题包括辅锁定脚本的哈希,并且被配置为使得仅当提供该哈希的原像时才能被解锁,并且因此子脚本被解锁,由此该原像是辅锁定脚本。因此,主锁定脚本可以被配置为从第一消息中提取锁定脚本(例如,第一锁定脚本),对所提取的锁定脚本进行哈希处理,并验证它哈希到作为主锁定脚本的一部分包括的辅锁定脚本的哈希。

[0096] 第一辅事务可以由爱丽丝103a创建,也可以由不同方生成。如上所述,第一辅事务包括输入和输出,该输入包括第一消息,该输出包括辅锁定脚本。辅锁定脚本被配置为对尝试解锁辅锁定脚本的事务强制执行若干条件,在这种情况下,该事务是第二主事务。对第二主事务强制执行的条件中的一个条件是,第二主事务的解锁脚本(例如,第一解锁脚本)必须包含基于第二主事务的第二消息。也就是说,第二消息基于第二主事务的一个、一些或全部字段。一些字段可能采用它们的原始形式,而另一些可能经过混淆处理,例如经过哈希处理。对第二主事务强制执行的另一个条件是,解锁脚本还必须包含包括第一主事务的一个、

一些或每个锁定脚本的数据项,并且因此第一数据项必须包含主锁定脚本。也就是说,第一数据项必须包括至少主锁定脚本。在一些示例中,第一数据项还必须包括第一主事务的第二锁定脚本,这将在下面讨论。使用第二消息和第一数据项,辅锁定脚本强制第二主事务在输出(例如,第一输出)中包括主锁定脚本。因此,主锁定脚本在主事务之间传播。由于主锁定脚本强制辅事务包含辅锁定脚本,因此辅锁定脚本也在辅事务之间传播。因此,作为主锁定脚本或辅锁定脚本的一部分而包括的任何数据(例如,令牌相关数据)分别在主事务或辅事务之间传播。

[0097] 如上所述,第一主事务可以包括第二输出。辅锁定脚本可以被配置为强制第一主事务的第二输出由第二主事务的输入花费。也就是说,在这些示例中,第一主锁定脚本的第二输出和第一辅事务的第一输出必须由第二主事务的相应输入花费,而不是由单独事务花费。

[0098] 强制执行该条件的一种选择是,强制第二主事务的第一解锁脚本包括第一辅事务。也就是说,第二数据项包括在第二主事务的解锁脚本中,其中该第二数据项包括例如采用序列化形式的第一辅事务。辅锁定脚本被配置为从第二消息中提取由所述第二主事务的第一输入花费的输出的相应事务标识符。类似地,辅锁定脚本被配置为从第二数据项中提取由第一辅事务的第一输入花费的输出的相应事务标识符。辅事务被配置为强制所提取的事务标识符是相同的。

[0099] 在这些示例中,第二主事务包括第二输入,该第二输入被配置为解锁第一主事务的第二输出,从而创建链接的事务链。应当注意的是,可以以这种方式链接事务,而无需上述强制执行由相同的第二主事务花费第一主事务的输出和第一辅事务的输出。然而,在脚本中强制执行该条件也有好处(如下所述)。

[0100] 在一些示例中,可以由辅锁定脚本执行附加验证,以验证第一数据项(即,包括第一主事务的锁定脚本的数据项)是正确的。第一主事务具有事务标识符或至少与事务标识符相关联,该事务标识符是通过使用SHA256或SHA512等哈希函数对该事务进行哈希处理而生成的。可以使用任何合适的哈希函数。第二主事务的第一解锁脚本可以包括第三数据项,该第三数据项包括对第一主事务应用哈希函数的中间状态(midstate)。技术人员将熟悉中间状态本身。通常,当(例如,使用基于SHA的哈希函数)对数据进行哈希处理时,数据会被划分为区块。使用第一组区块来执行一系列计算以产生第一中间状态。使用第一中间状态和第二组区块来执行一系列计算以产生第二中间状态,依此类推,直到所有区块都已经被操作以给出哈希摘要。换句话说,哈希函数不涉及单个操作。相反,可以将哈希函数的输入分解为区块(例如,对于SHA256为64个字节),并按顺序处理这些区块。中间状态是处理这些区块中的一个或多个区块之后的内部状态。在处理每个区块之后产生中间状态。在处理第n个区块之后计算的中间状态依赖于所有先前区块,而不是后续区块。SHA256哈希函数的内部工作原理的示例可在<https://qvault.io/cryptography/how-sha-2-works-step-by-step-sha-256/>找到。

[0101] 本公开认识到,通过提供通过对第一主事务的第一部分(不包括锁定脚本)应用哈希函数而生成的中间状态,则可以使用第一数据项(即,锁定脚本)在脚本中继续执行计算,以产生第一主事务的候选事务标识符。如果候选事务标识符与第一主事务的实际事务标识符匹配,则第一数据项必须包括第一主事务的实际锁定脚本(并且因此包括主锁定脚本)。

[0102] 辅锁定脚本还可以被配置为强制第二主事务包括链中的初始主事务的事务标识符。例如,第二主事务可能需要在第二输出中包括事务标识符。如果第一主事务是链中的初始事务,则包括第二主事务的每个后续事务被强制包括第一主事务的事务标识符。在这些示例中,初始主事务(例如,第一主事务)将不包括其自己的事务标识符,因为它基于整个初始主事务。可以强制每个后续主事务包括初始主事务的事务标识符。

[0103] 第二主事务可以由爱丽丝103a生成,也可以由不同方生成。如上所述,第二主事务包括花费第一辅事务的第一输出的第一输入,并且可以包括花费第一主事务的第二输出的第二输入。第二主事务还包括第一输入,该第一输入包括主锁定脚本。第二主事务可以包括第二输出,例如将由后续主事务的第二输出解锁。第二输出可以锁定到第二方(鲍勃103b)的公钥。第二输出可以包括初始主事务的事务标识符。

[0104] 爱丽丝103a被配置为使得第一主事务被提交到一个或多个区块链节点104。例如,爱丽丝103a自己可以将第一主事务发送到区块链网络106,或者她可以经由中间方(例如,鲍勃103b)将第一主事务发送到区块链网络。类似地,爱丽丝103a可以被配置为使得第一辅事务和/或第二主事务被提交到一个或多个区块链节点104。

[0105] 在一些示例中,爱丽丝103a可以将第二主事务发送给鲍勃103b。爱丽丝103a还可以向鲍勃103b发送默克尔证明,其中默克尔证明可以用于验证第一主事务已经被记录在区块链150上。

[0106] 到目前为止,第一主事务主要被描述为初始主事务。然而,第一主事务通常可以是第n个主事务,在这种情况下,上述第二主事务将是第n+1个主事务。类似地,在该示例中,存在第n-1个主事务。在这种情况下,爱丽丝103a可以负责也可以不负责生成发行事务。

[0107] 继续该示例,第一主事务可以包括引用先前辅事务的第一输出的第一输入。与第一辅事务一样,先前辅事务的第一输出包含辅锁定脚本(即,完全相同的辅脚本的实例)。先前辅事务还具有引用先前主事务的第一输出的输入。先前主事务的第一输出包含主锁定脚本。在该示例中,先前主事务可以是初始主事务,也可以是主事务链中的不同主事务。

[0108] 先前辅事务的辅锁定脚本被配置为对第一主事务执行上述检查(即,强制执行相同的条件),如同第一辅事务的辅锁定脚本对第二主事务执行上述检查一样。例如,强制第一主事务包括主锁定脚本。

[0109] 在这些示例中,当第二主事务具有父主事务(即,第一主事务)和祖父主事务(即,先前主事务)时,第一主事务的辅锁定脚本被配置为强制第二主事务的解锁脚本包括第四数据项,该第四数据项包括先前主事务的一个、一些或每个锁定脚本。这将包括主锁定脚本。使用第二主事务的解锁脚本中包括的第二消息(即,基于第二主事务生成的消息)和第四数据项,辅锁定脚本验证第二主事务的第一输出中包括的主锁定脚本与先前主事务中包括的主锁定脚本相同。这验证:相同的主锁定脚本从一个主事务传递到后续主事务,并传递到后续主事务。换句话说,这证明先前主事务的主锁定脚本已经被执行。

[0110] 通过对先前主事务应用哈希函数(例如,SHA256)而生成的中间状态可以用于验证第四数据项是正确的。也就是说,辅锁定脚本可以执行附加验证,以验证第四数据项(即,包括先前主事务的锁定脚本的数据项)是正确的。先前主事务具有事务标识符或至少与事务标识符相关联,该事务标识符是通过使用SHA256等哈希函数对该事务进行哈希处理而生成的。第二主事务的第一解锁脚本可以包括第五数据项,该第五数据项包括对先前主事务应

用哈希函数的中间状态。辅锁定脚本可以使用第四数据项和第一数据项来生成先前主事务的候选事务标识符。如果候选事务标识符与先前主事务的实际事务标识符匹配,则第四数据项必须包括先前主事务的实际锁定脚本(并且因此包括主锁定脚本)。

[0111] 第一主事务可以包括花费先前主事务的第二输出的第二输入。该条件可以由先前辅锁定脚本中包括的辅锁定脚本强制执行。

[0112] 先前主事务和/或先前辅事务可以由爱丽丝103a生成,也可以由不同方(例如,鲍勃103b或查理103c)生成。类似地,虽然以上描述总体上是根据爱丽丝103a生成第一主事务、第一辅事务和第二主事务来描述的,但是这些事务中的任何一个或多个事务可以由鲍勃103b或查理103c生成。

[0113] 一方(例如,鲍勃103b)可以出于验证主事务已经被正确地生成的意图而获取主事务。例如,鲍勃103b可以从爱丽丝103a获取第二主事务。第二主事务已经在上面描述。简而言之,第二主事务具有包括主锁定脚本的第一输出、包括至少(基于第二主事务生成的)第二消息的第一输入(即,第一输入的解锁脚本)以及包括第一主事务的一个、一些或每个锁定脚本的第一数据项。这些锁定脚本中的一个锁定脚本包括主锁定脚本。在以下示例中,第二主事务的第一输入还包括第一辅事务。

[0114] 鲍勃103b验证第一主事务已经被记录在区块链150上,并且因此是有效事务。执行此操作的一种选择是获取默克尔证明(即,一组哈希),该默克尔证明可以与第一主事务的事务标识符一起用于计算候选区块头。如果候选区块头与区块链区块151的区块头匹配,则鲍勃103b可以确保该区块包含第一主事务。另一种选择是向区块链节点104请求确认。

[0115] 如上所述,第二主事务的解锁脚本可以包括第三数据项(对第一主事务的一部分应用的哈希函数的中间状态)。鲍勃103b可以使用第一数据项和第三数据项来生成第一主事务的候选事务标识符,并且使用候选事务标识符来执行默克尔证明。

[0116] 鲍勃103b还验证第一主事务中包括的主锁定脚本是正确的,这意味着主锁定脚本被配置为要求第一辅事务包含辅锁定脚本。鲍勃103b可以显式地验证主锁定脚本,即检查它是否包含正确的函数等。作为更高效的选择,如上所述,主锁定脚本可以包含辅锁定脚本的哈希作为哈希谜题的一部分。因此,为了解决哈希谜题,辅事务必须包括辅锁定脚本。因此,鲍勃103b可以验证主锁定脚本包括辅锁定脚本的哈希。例如,哈希可以预先由鲍勃103b所知,例如由爱丽丝103a或生成初始主事务的一方发送。

[0117] 作为选项检查,鲍勃103a可以验证第二主事务(例如,在第二输出中)包含主事务链中的初始事务的事务标识符,该初始事务可以是第一主事务。

[0118] 在执行验证之后,鲍勃103b可以生成第二辅事务和第三主事务。例如,这可以用于将令牌转移给查理103c。第二辅事务采用与第一辅事务相同的格式。第二辅事务具有花费第二主事务的第一输出(即,包含主锁定脚本的输出)的第一输入。为此,它包括基于第二辅事务的一些或全部字段生成的第三消息。第二辅事务具有包括辅锁定脚本的第一输入。鲍勃103b直接或经由不同方将第二辅事务发送到区块链网络106。

[0119] 鲍勃103b还生成第三主事务。第三主事务类似于第二主事务。它包括花费第二主事务的第一输出的第一输入。第三主事务的第一输入包括基于第三主事务的一些或全部字段生成的第三消息。在该示例中,第一输入包括第二辅事务、第一数据项(即,第一主事务的每个锁定脚本)以及第六数据项,该第六数据项包括第二主事务的一个、一些或每个锁定脚

本。第三主事务可以包括锁定到查理103c的公钥的第二输出。第三主事务可以包括花费第二主事务的第二输出的第二输入。鲍勃103b直接或经由不同方将第二主事务发送到区块链网络106。

[0120] 5. 区块链节点核实令牌

[0121] 本节描述了上述实施例的详细示例。应当理解的是,本节中介绍的一些特征是可选的。该示例侧重于事务是令牌事务的用例,即这些事务包含令牌相关数据。这只是一个说明性用例,并且通常可以用于需要可证明事务链的其他应用程序,而无需验证所有事务返回发行。

[0122] 本节描述了满足四个要求的令牌协议。

[0123] 1. 令牌事务的验证者不需要一直追溯到令牌的发行事务,从而提高了效率。

[0124] 2. 令牌发行之后,令牌在两个用户之间传输不涉及可信第三方,从而提高了安全性和可用性。

[0125] 3. 令牌事务的大小是恒定的。它独立于历史事务的数量,因此改进了存储。

[0126] 4. 该方案不易受到重放攻击,即令牌无法复制,从而提高了安全性。

[0127] 在脚本中强制执行而不需要验证者或第三方执行任何令牌规则的令牌称为“矿工核实”令牌或“区块链节点核实”令牌。据申请人所知,本申请公开了区块链150上的第一矿工核实令牌协议。

[0128] 该协议使用以下构建块:

[0129] ●验证相同的操作码出现在事务链中的祖父事务的两个锁定脚本中的每个锁定脚本中,并且这些锁定脚本已经被执行的脚本片段;

[0130] ●强制执行事务链中的后续子锁脚本包含相同操作码的脚本片段;

[0131] ●实现部分SHA256的脚本片段;

[0132] ●将验证数据从一个令牌事务传递到其花费事务(以避免自引用)的辅事务;以及

[0133] ●非可重放的唯一令牌ID(UTID)。为了简单起见,假设这是发行事务ID。

[0134] 下面分步骤介绍该协议,一次描述该方案的一个部分。该方案的每次迭代都会产生一个问题,该问题将在该方案的下次迭代时得到解决,最终获得一个完整的方案。以花费事务 $T_{x_{n+1}}$ 为视角,解释如何将 $T_{x_{n+1}}$ 确定为有效的令牌事务,而无需一直追溯到发行事务。使用以下术语来指代与 $T_{x_{n+1}}$ 相关的事务:

[0135] ●父事务 T_{x_n} :由 $T_{x_{n+1}}$ 的第二输入引用的事务;

[0136] ●祖父事务 $T_{x_{n-1}}$:由 T_{x_n} 的第二输入引用的事务;以及

[0137] ●辅事务 $T_{x_n}^A$:在其输入中引用 T_{x_n} 的第一输出并由 $T_{x_{n+1}}$ 的第一输入引用的事务。

[0138] 这些事务及其关系如图4所示。假设祖父事务、父事务和辅事务都已经被矿工接受。花费事务 $T_{x_{n+1}}$ 是最后看到的事务。

[0139] 5.1 事务结构

[0140] 该协议考虑令牌链中最新的三个事务。对于每个事务,都有一个辅事务。辅事务是避免事务膨胀和自引用所必需的。图4示出了事务结构的示意图,其中事务链被定义为第二输入和输出链。这将在下面更详细地解释。

[0141] 在事务的序列化中,首先出现所有输入,然后出现所有输出。输入和输出按其索引顺序出现。下面的讨论从事务编码中抽象出来,仅侧重于解锁脚本和锁定脚本。应当注意的

是,当提到解锁脚本时,也可能是指所引用的与该解锁脚本对应的输出点。这是为了避免与函数的“输入”混淆。

[0142] 考虑单个事务链,例如非同质化令牌(NFT)的表示。假设不存在拆分和组合令牌,从而创建事务有向无环图(DAG)。此外,还假设不存在其他可以添加的输入和输出。

[0143] 5.2表示法

[0144] 使用OP前缀OP_CODE以及不带前缀的预定义操作码(例如,CODE)的组合来表示预定义操作码。依赖于变量(例如,公钥)的操作码或操作码的组合将用括号CODE(PK)表示。将操作码的组合的组成表示为CODE1+CODE2。

[0145] 5.3推送事务及其衍生物

[0146] 在给定事务 Tx_{n+1} 的sighash原像 m_{n+1} 作为输入的情况下, Tx_n 的锁定脚本中标记为PUSHTX的操作码的组合的结果为TRUE。操作码秘密地构建并验证sighash原像 m_{n+1} 上的签名。如果输入了具有对应公钥的签名,则最终操作码OP_CHECKSIG的结果为TRUE,其中签名位于从事务本身提取的事务消息 m_{n+1} 之上。

[0147] 事务消息 m_{n+1} 是从 Tx_{n+1} 和 Tx_n 中的各个字段构建的。具体地,感兴趣的字段如下:

[0148] 1. Tx_{n+1} 的锁定脚本;

[0149] 2. 输出点 $TxID_n$ 、 $TxID_n^A$;以及

[0150] 3. Tx_n 的锁定脚本。

[0151] 上标A表示该事务是辅事务。

[0152] 包括PUSHTX的含义是,当前事务 Tx_{n+1} 的sighash原像可证明地位于堆栈上。这意味着可以对这些字段中的任何字段强制执行条件,例如包含操作码或序列号的某些组合。如果不满足这些条件,则花费事务无效。

[0153] 如果 $dSHA256(Tx_n) = TxID_n$,其中dSHA256是双重SHA256哈希函数,则在给定 Tx_n 和 $TxID_n$ (取自 m_{n+1})作为输入的情况下, Tx_n 的锁定脚本中的操作码PUSHTXPARENT的组合的结果为TRUE。操作码秘密地计算序列化事务 Tx_n 的哈希,并将结果与 $TxID_n$ 进行比较,其中事务ID已经从输入到PUSHTX的sighash原像中的输出点中提取。如果这两个值相同,则结果为TRUE。为了确保推送到堆栈的确实是当前事务的父事务,PUSHTXPARENT包含PUSHTX以从中提取输出点。由于PUSHTX包含所有输出点,因此任何父事务都可以被推送,即使任何父事务与它出现在其中的输入的输出点不对应。

[0154] 序列化事务消息 Tx_n 是从各种字段构建的。特别感兴趣的字段如下:

[0155] 1. Tx_n 的锁定脚本;以及

[0156] 2. 输出点 $TxID_{n-1}$ 。

[0157] 如上所述,包含PUSHTXPARENT和PUSHTX的含义是,可以对序列化父事务中的任何字段强制执行条件。

[0158] 类似地,如果 $dSHA256(Tx_{n-1}) = TxID_{n-1}$,在给定 Tx_{n-1} 和 $TxID_{n-1}$ (来自 Tx_n)作为输入的情况下, Tx_n 的锁定脚本中的操作码PUSHTXGRANDPARENT的组合的结果为TRUE。与PUSHTXPARENT和PUSHTX一样, Tx_n 是提供给PUSHTXPARENT以确保该事务确实是祖父事务的输入。只要通过PUSHTXPARENT、PUSHTXGRANDPARENT等迭代地建立链接,就可以重复此操作来推送任何祖先事务。

[0159] 5.4SHA256哈希函数

[0160] 哈希函数将任意大小 m 的数据映射到固定大小的数据。当输入数据长于447位时, SHA256哈希函数的执行通过使用默克尔-达姆加德结构的压缩函数 $CF(\blacksquare, \blacksquare)$ 来迭代地处理数据。压缩函数用于通过以下方式计算SHA256(m)。

[0161] 1. 计算填充的原像 $PI = m || \text{pad}$, 其中 pad 是消息上的填充, 确保长度是512位的倍数。

[0162] 2. 将 PI 拆分成大小为512位的1个区块, 其中 $PI = m_1 || m_2 || \dots || m_l$ 。

[0163] 3. 计算 $\text{mid}_1 = CF(m_1, IV)$, 其中 IV 是预定义的初始化向量。

[0164] 4. 计算 $\text{mid}_2 = CF(m_2, \text{mid}_1)$ 。

[0165] 5. 继续处理大小为512位的每第 j 个区块, $\text{mid}_j = CF(m_j, \text{mid}_{j-1})$, 直到处理完整个原像。

[0166] 6. 输出 $CF(m_l, \text{mid}_{l-1}) = \text{SHA256}(m)$ 。

[0167] 可以执行上述步骤的计算直到第 j 次迭代, 并停止执行。可以存储其余原像 $PPI = m_{j+1} || \dots || m_l$ 和中间状态 $\text{mid} = CF(m_j, \text{mid}_{j-1})$, 并在稍后继续执行计算。进一步的“部分SHA”计算将表示为 $h = \text{partialSHA256}(PPI, \text{mid})$, 输出与SHA256(m) 相同的结果。

[0168] 5.5操作码中的部分SHA256

[0169] 前 j 次迭代的计算将在链下执行。最后 $(l - j)$ 次迭代将在脚本中执行, 需要在操作码中实现计算。如果 $h = \text{partialSHA256}(PPI, \text{mid})$, 在给定 h 和其余原像 PPI 以及中间状态 mid 作为输入的情况下, 这些操作码PARTIALSHA256将返回TRUE。在给定 PPI 和 mid 作为输入的情况下, 操作码秘密地计算SHA256哈希函数的其余迭代。将这一结果与 h 进行比较。如果两者相同, 则该组操作码返回TRUE。

[0170] 在当前情况下, 计算序列化事务的部分SHA256, 该序列化事务经过哈希处理并与给定TxID进行比较。如前所述, 输入首先出现在序列化事务中。这意味着可以使用部分SHA256计算来编辑解锁脚本中序列化事务的第一输入, 该第一输入是PUSHTXPARENT或PUSHTXGRANDPARENT所需的输入。

[0171] 利用部分SHA256来编辑序列化事务的第一解锁脚本。将滥用上述表示法, 写为 $CF(us_{n,0}, IV)$ 意味着计算SHA256哈希函数的前 j 次迭代, 直到第一输入结束。换句话说, 以这种方式写入并不意味着 $us_{n,0}$ 只有512位 (以及SHA256哈希计算的第一次迭代)。这很可能会持续更长时间。

[0172] 5.6. 问题1: 执行和传播证明

[0173] 考虑的第一个问题是所谓的“返回创始”问题。当令牌的核实者想要证明他们接收的令牌事务是返回发行事务的事务链的一部分时, 就会发生这种情况。

[0174] 单独使用PUSHTX使得发行者能够强制任何子事务采用某种格式, 包括其任何子事务、孙事务等。例如, 他们可能要求在每个未来事务中包含序列号。然而, 核实者仍然必须显式地核实返回发行事务的每个事务。这会增加每次传输的证明长度和核实时间。

[0175] 为了免于验证每个事务返回创始事务, 可以由可信令牌处理器来见证每次传输。这可解决回归创始问题, 但会引入性能瓶颈和安全风险, 从而降低可用性。此外, 这还会引入通信和基础设施成本。此外, 这违背了针对矿工核实令牌的目的。可信令牌处理器也可以处理整个传输, 从而消除脚本复杂性和PUSHTX成本。

[0176] 考虑包含一个输入和一个输出的简单事务链, 如图5所示。

[0177] 定义PROOFOFEXECUTION,如果 $T_{x_{n-1}}$ 和 T_{x_n} 的所有锁定脚本(在该用例中是一个)相同,即 T_{x_n} 中的第一锁定脚本与 $T_{x_{n-1}}$ 中的第一锁定脚本相同,在给定sighash原像 m_{n+1} 作为输入的情况下,该PROOFOFEXECUTION的结果为TRUE。将该执行证明标记为希望脚本片段确保在 $1s_n$ 和 $1s_{n-1}$ 中执行操作码中的相同函数。如果锁定脚本包含变量(例如,公钥),则它不会保持不变,并且会被忽略。操作码PROOFOFEXECUTION包括以下内容:

[0178] 1.PUSHTX,如果 m_{n+1} 是事务 $T_{x_{n+1}}$ 的sighash原像,该PUSHTX将 m_{n+1} 作为输入并输出TRUE。

[0179] 2.PUSHTXPARENT,如果 T_{x_n} 是 T_{xID_n} 的哈希原像,该PUSHTXPARENT从 m_{n+1} 和 T_{x_n} 获取输出点 T_{xID_n} 并输出TRUE。

[0180] 3.PUSHTXGRANDPARENT,如果 $T_{x_{n-1}}$ 是 $T_{xID_{n-1}}$ 的哈希原像,该PUSHTXGRANDPARENT从 T_{x_n} 获取 $T_{x_{n-1}}$ 和 $T_{xID_{n-1}}$ 并输出TRUE。

[0181] 4.操作码,如果 $T_{x_{n-1}}$ 和 T_{x_n} 中的锁定脚本包含相同的操作码,该操作码将来自 $T_{x_{n-1}}$ 的 $1s_{n-1}$ 中的操作码与来自 T_{x_n} 的 $1s_n$ 中的操作码进行比较并输出TRUE。

[0182] 5.操作码,该操作码确保父事务和祖父事务中仅存在一个输入和一个输出。

[0183] 因此,由 $(m_{n+1}, T_{x_n}, T_{x_{n-1}})$ 解锁包含PROOFOFEXECUTION的脚本。

[0184] 定义PROOFOFPROPAGATION,如果 T_{x_n} 和 $T_{x_{n+1}}$ 的所有锁定脚本(在该用例中是一个)相同,即 T_{x_n} 中的第一锁定脚本与 $T_{x_{n+1}}$ 中的第一锁定脚本相同等等,在给定sighash原像 m_{n+1} 作为输入的情况下,该PROOFOFPROPAGATION的结果为TRUE。将该传播证明标记为希望脚本片段确保在 $1s_n$ 中执行操作码中的相同函数,并且将在 $1s_{n+1}$ 中执行该函数。与PROOFOFEXECUTION一样,如果锁定脚本包含变量(例如,公钥),它会被忽略。操作码PROOFOFPROPAGATION包括以下内容:

[0185] 1.PUSHTX,如果 m_{n+1} 是事务 $T_{x_{n+1}}$ 的sighash原像,该PUSHTX将 m_{n+1} 作为输入并输出TRUE。

[0186] 2.PUSHTXPARENT,如果 T_{x_n} 是 T_{xID_n} 的哈希原像,该PUSHTXPARENT从 m_{n+1} 和 T_{x_n} 获取输出点 T_{xID_n} 并输出TRUE。

[0187] 3.操作码,如果第一锁定脚本中的操作码相同,该操作码将来自 T_{x_n} 的 $1s_n$ 中的操作码与来自 m_{n+1} 的 $1s_{n+1}$ 中的操作码进行比较并输出TRUE。

[0188] 4.操作码,该操作码确保子事务中仅存在一个输入和一个输出。

[0189] 因此,由 (m_{n+1}, T_{x_n}) 解锁包含PROOFOFPROPAGATION的脚本。

[0190] 我们定义一组操作码 δ_1 ,如

[0191]

PROOFOFPROPAGATION
OP_IF
PROOFOFEXECUTION
OP_ENDIF

[0192] 其中,如果 $n \geq 3$,则执行PROOFOFEXECUTION。下面详细说明了如何捕获该用例。

[0193] 如果对于所有 $n \geq 3$ 来说 T_{x_n} 中的锁定脚本 $1s_n$ 是 δ_1 ,则解锁脚本是 $us_n = (m_{n+1}, T_{x_n},$

$T_{x_{n-1}}$), 其中:

[0194] $m_{n+1} = T_{x_{n+1}}$ 的 sighash 原像;

[0195] T_{x_n} = 序列化父事务; 以及

[0196] $T_{x_{n-1}}$ = 序列化祖父事务。

[0197] 对于 $n=2$, 解锁脚本是 $us_n = (m_2, T_{x_1})$ 。由于 T_{x_1} 是发行事务, 因此 T_{x_0} 中没有相关的锁定脚本。 T_{x_1} 是出现该锁定脚本的第一事务。

[0198] 应当注意的是, ls_n 中的操作码 $\mathcal{s}_1$ 的组合证明锁定脚本 ls_n 中的操作码 $\mathcal{s}_1$ 也包含在 ls_{n-1} 和 ls_{n+1} 中。

[0199] PROOF OF EXECUTION 和 PROOF OF PROPAGATION 两者均查看是否存在同时包含它们的 $\mathcal{s}_1$ 。换句话说, 包含这两组操作码的锁定脚本是这两组操作码的输入的一部分。

[0200] 通过归纳法证明, 如果发行事务 T_{x_1} 包含上述操作码, 则 $T_{x_{n+1}}$ 必须链接到该发行事务。

[0201] 主张:

[0202] 考虑从 T_{x_1} 到 T_{x_n} 的一组事务, 每个事务至少具有一个输入和一个输出。给定 T_{x_n} 满足以下条件:

[0203] ● T_{x_n} 的输入中的一个输入花费来自 $T_{x_{n-1}}$ 的输出, $T_{x_{n-1}}$ 在如上所述的锁定脚本 ls_n 中包含 $\mathcal{s}_1$; 以及

[0204] ● 矿工接受 T_{x_n} ;

[0205] 然后, 将 T_{x_n} 链接到发行事务 T_{x_1} 。

[0206] 证明:

[0207] 通过归纳法证明这一主张。当 $n=2$ 时, 很明显的是, 如果矿工接受 T_{x_n} , 则它会通过 T_{x_2} 的输出点链接到 T_{x_1} 。

[0208] 基础用例 $n=3$:

[0209] T_{x_2} 包含锁定脚本 $ls_2 = \mathcal{s}_1$, 则解锁脚本 $us_2 = (m_3, T_{x_2}, T_{x_1})$ 在 T_{x_3} 中。如果矿工接受 T_{x_3} , 则根据 $\mathcal{s}_1$ 的花费条件, 它必须是 T_{x_1} 的孙事务。因此, 它链接到发行事务。

[0210] 第 n 个用例:

[0211] 假设第 n 个用例为 True。

[0212] 第 $n+1$ 个用例:

[0213] 如果矿工接受 $T_{x_{n+1}}$, 在给定具有包含锁定脚本 $ls_n = \mathcal{s}_1$ 的输出的事务 T_{x_n} 的情况下, 可以证明花费包含 ls_n 的输出的事务 $T_{x_{n+1}}$ 通过事务链链接到发行事务 T_{x_1} 。

[0214] 如果满足以下三个条件:

[0215] ● $ls_n = \mathcal{s}_1$;

[0216] ● $T_{x_{n+1}}$ 花费 T_{x_n} 的输出; 以及

[0217] ● 矿工接受 $T_{x_{n+1}}$;

[0218] 则必须满足 $us_n = (m_{n+1}, T_{x_n}, T_{x_{n-1}})$, 才能解锁输出。由于 $ls_n = \mathcal{s}_1$, 因此由于 PROOF OF EXECUTION 而必须满足 $ls_{n-1} = \mathcal{s}_1$, 并且由于 PROOF OF PROPAGATION 而必须满足 $ls_{n+1} = \mathcal{s}_1$ 。因此, $T_{x_{n-1}}$:

[0219] ●包含锁定脚本 $ls_{n-1} = s_1$;

[0220] ●由 Tx_n 花费(对于 us_n ,需要该链接才能花费 ls_n);以及

[0221] ●矿工接受 Tx_n (如果是 Tx_{n+1})。

[0222] 因此, Tx_n 链接到 Tx_1 。此外, Tx_{n+1} 是 Tx_n 的子事务,并且因此链接到 Tx_n 。因此, Tx_{n+1} 链接到 Tx_1 。

[0223] 可以证明,链中的每个事务包含所有先前事务,如图6所示,从而导致事务膨胀问题。

[0224] 主张:

[0225] 考虑从 Tx_1 到 Tx_n 的一组事务,该组事务满足上述主张。第 n 个事务 Tx_n 包含返回发行 Tx_1 的所有事务。

[0226] 证明:

[0227] 事务 Tx_n 的解锁脚本包含其父事务和祖父事务,这是 Tx_{n-1} 的锁定脚本中的操作码 s_1 所要求的。对于所有 $n \geq 3$,这都为True。因此,对于给定 Tx_n ,该事务的父事务和祖父事务也包含它们相应的父事务和祖父事务。对于这些事务中的每个事务,这都为True,因此 Tx_n 包含返回发行事务的所有事务。

[0228] 5.7.问题2:部分SHA256哈希函数

[0229] 在解决问题1时,只需要来自父事务和祖父事务的 us_n 的输入中的 Tx_n 的 ls_n 、 Tx_n 的输出点的事务ID $TxID_{n-1}$ 以及 Tx_{n-1} 的 ls_{n-1} 来执行PROOFOFPROPAGATION和PROOFOFEXECUTION。不需要父事务和祖父事务的解锁脚本。可以考虑如何编辑事务的这些部分。

[0230] 考虑具有两个输入和两个输出的事务链,如图7所示。附加输入和输出对应于P2PKH(PK)脚本。引入第二输入和输出,使得第一解锁脚本能够编辑,并且PROOFOFEXECUTION仍然能够使用第二输出点来利用祖父事务。在描述以下方案之后,将明确说明第 n 个事务不再包含返回发行事务的所有事务的原因。在较高级别,这是因为解锁脚本数据不会传播,该解锁脚本数据是事务中携带有关先前事务的信息的部分。可以通过在链下计算事务的哈希(以与 $TxID$ 进行比较)来防止数据传播。

[0231] 定义PUSHTXPARTIALPARENT,如果 $TxID_n = \text{SHA256}(\text{partialSHA256}(PPI_n, \text{mid}_n))$,在给定来自 Tx_{n+1} 中的第二输出点的 $TxID_n$ 和 PPI_n 以及 mid_n 作为输入的情况下,该PUSHTXPARTIALPARENT的结果为TRUE。

[0232] 定义PUSHTXPARTIALGRANDPARENT,如果 $TxID_{n-1} = \text{SHA256}(\text{partialSHA256}(PPI_{n-1}, \text{mid}_{n-1}))$,在给定来自 Tx_n 中的第二输出点的 $TxID_{n-1}$ 、 PPI_{n-1} 、 mid_{n-1} 作为输入的情况下,该PUSHTXPARTIALGRANDPARENT的结果为TRUE。

[0233] 如上所述,部分SHA256计算使得分别能够编辑这两个操作码中的父事务和祖父事务的第一输入。因此,输入 $us_{n-1,0}$ 和 $us_{n-2,0}$ 不会出现在 $us_{n,0}$ 中。然后, PPI_n 包含父事务的第二解锁脚本和两个锁定脚本。 PPI_{n-1} 包含祖父事务的两个锁定脚本。父事务的第二解锁脚本使得能够提取祖父事务ID。将提取锁定脚本作为执行PROOFOFPROPAGATION和PROOFOFEXECUTION的输入。

[0234] 重新定义PROOFOFPROPAGATION和PROOFOFEXECUTION,以分别包含PUSHTXPARTIALPARENT和PUSHTXPARTIALGRANDPARENT,而不是PUSHTXPARENT和PUSHTXGRANDPARENT。重新定义意味着脚本现在从 PPI_n 或 PPI_{n-1} 而不是如前所述从 Tx_n 和 Tx_{n-1}

中提取所需锁定脚本中的操作码。因此,包含PROOFOFEXECUTION的脚本现在由 $(m_{n+1}, PPI_n, mid_n, PPI_{n-1}, mid_{n-1})$ 解锁,并且包含PROOFOFPROPAGATION的脚本由 (m_{n+1}, PPI_n, mid_n) 解锁。

[0235] 此外,重新定义PROOFOFPROPAGATION和PROOFOFEXECUTION,以包含操作码,这些操作码可以比较多组锁定脚本中的操作码,而不仅仅是一组锁定脚本中的操作码。如前所述,这分别将 Tx_n 中的第一锁定脚本中的操作码与 Tx_{n+1} 或 Tx_{n-1} 的第一锁定脚本中的操作码进行比较。现在,这还比较这些事务中的一组第二锁定脚本。如果第一组锁定脚本包含相同的操作码,并且第二组锁定脚本包含相同的操作码,则这些操作码返回TRUE。这还意味着,它们现在确保每个事务中仅存在两个输入和输出,而不是检查每个事务中的一个输入和输出。这些操作码的任何进一步引用是指这些新定义。

[0236] 定义一组操作码 $\mathcal{S}_2$,该组操作码具有重新定义的操作码

[0237]

```
PROOFOFEXECUTION +
OP_IF
    PROOFOFPROPAGATION
OP_ENDIF
```

[0238] 对于所有 n ,定义要使用 $us_{n,0} = (m_{n+1}, mid_n, PPI_n, mid_{n-1}, PPI_{n-1})$ 解锁的锁定脚本 $ls_{n,0} = \mathcal{S}_2$,其中

[0239] $m_{n+1} = Tx_{n+1}$ 的sighash原像;

[0240] $mid_n = CF(us_{n-1,0}, IV)$;

[0241] $PPI_n = us_{n-1,1} || ls_{n,0} || ls_{n,1} || pad_n$;

[0242] $mid_{n-1} = CF(us_{n-2,0} || us_{n-2,1}, IV)$;

[0243] $PPI_{n-1} = ls_{n-1,0} || ls_{n-1,1} || pad_{n-1}$ 。

[0244] 每个解锁脚本只包括这五个组成部分,它们将被证明具有固定的长度。会导致膨胀问题的解锁脚本 $us_{n-1,0}$ 、 $us_{n-2,0}$ 已经通过部分SHA256计算进行编辑。

[0245] 主张:

[0246] 假设 Tx_n 是来自上面定义的事务链的具有两个输入和两个输出的事务,则对于所有 n , Tx_n 的大小小于常量,即 $len(Tx_n) \leq const$ 。

[0247] 证明:

[0248] 可以简单地证明 $len(Tx_n)$ 并不取决于 n 。在序列化事务和sighash原像中,大多数字段的大小都有明确定义的界限。

[0249] sighash原像:

[0250] 不包括锁定脚本和解锁脚本,上限为

[0251] $4+32+32+32+4+9+8+4+32+4+4=165$ 个字节。

[0252] 应当注意的是,由于需要检查方案中的锁定脚本,因此最后32个字节(哈希函数的输出)将需要替换为其原像。这意味着不包括锁定脚本的上限为

[0253] $4+32+32+32+4+9+8+4+8vout+4+4=149$ 个字节。

[0254] 在该用例中, $vout$ 具有值2。9是VarInt,因此可以介于1个字节到9个字节之间。已

经排除了正在花费的锁定脚本(父锁定脚本 ls_{n-1})和当前事务 $ls_{n,vout}$ 中的锁定脚本。

[0255] 明确地说,解锁脚本和锁定脚本的大小如下表所示。

[0256]

脚本段	子脚本段	脚本段大小 (以字节为单位)	总脚本大小 (上限)
$us_{n-1,0}$ $= \begin{pmatrix} m_n, mid_{n-1}, \\ PPI_{n-1}, mid_{n-2}, \\ PPI_{n-2} \end{pmatrix}$	m_n	$149 + len(ls_{n-1,0})$ $+ len(ls_{n,0})$ $+ len(ls_{n,1})$	392 $+ 4len(s_2)$ $+ len(mid_{n-1})$ $+ len(mid_{n-1})$
	mid_{n-1}	$len(mid_{n-1})$	
	PPI_{n-1}	$32 + 4 + 9 + 4$ $+ len(us_{n-1,1})$ $+ 2(8 + 9)$ $+ len(ls_{n-1,0})$ $+ len(ls_{n-1,1})$	
	mid_{n-2}	$len(mid_{n-2})$	
	PPI_{n-2}	$2(8 + 9)$ $+ len(ls_{n-2,0})$ $+ len(ls_{n-2,1})$	
$us_{n-1,1}$ $= (sig_{n-1}, PK_{n-1})$	sig_{n-1}	71或72	137
	PK_{n-1}	33或65	
$ls_{n,0} = s_2$		$len(s_2)$	$len(s_2)$
$ls_{n,1} = P2PKH(PK_n)$		24	24

[0257] 在序列化事务中,不包括锁定和解锁脚本的上限为 $(4+9+9+4+vin(32+4+9+4)+vout(8+9))=158$ 个字节,

[0258] 其中vin和vout分别是输入和输出的数量,每个具有值2。应当注意的是,这9个字节中的每个字节都表示一个VarInt字段。这些值可以介于1个字节到9个字节之间,并且对于每个输入或输出可以是不同的。

[0259] 包括锁定脚本和解锁脚本,使用上表,可以知道序列化事务的大小为 $len(Tx_n) = 711 + 5len(s_2) + len(mid_{n-1}) + len(mid_{n-2})$ 个字节。

[0260] 事务 Tx_n 中的所有字段取决于脚本 s_2 的长度和父事务和祖父事务的中间状态的长度,这两个事务的长度都是恒定的。因此,事务大小的上限独立于n。

[0261] 备注:解锁脚本中存在一些重复,其中 s_2 出现四次。可以通过将其作为输入并复制数据来减轻这种情况。这将导致序列化事务的大小为

$(711 + 2len(s_2) + len(mid_{n-1}) + len(mid_{n-2}))$ 个字节。

[0262] 通过从当前事务中编辑父事务和祖父事务的第一解锁脚本,第n个事务 Tx_n 不再包含返回发行事务 Tx_1 的所有事务。第一输入仅包含第二解锁脚本和两个先前事务的两个输出,其中不包含关于先前事务的任何信息,如图8所示。

[0263] 应当注意的是,对应于 $us_{n-1,0}$ 和 $us_{n-1,1}$ 的输出点不能交换,因为这会重新引入事务

膨胀问题。包含先前事务的解锁脚本 $us_{n-1,0}$ 必须首先出现,才能在中 $us_{n,0}$ 进行编辑。

[0264] 虽然已经解决了膨胀问题,但是引入了以下自引用问题。每轮压缩函数处理64个字节,并且必须写入操作码。根据现有操作码,函数 $CF()$ 需要超过64个字节。

[0265] 锁定脚本 $ls_{n,0}$ 包括多轮压缩函数,被定义为 $partialSHA256()$,取 mid_n 和 PPI_n 作为输入。这意味着操作码中的 $partialSHA256()$ 必须处理(在操作码中)包含其自身的 PPI_n 。考虑最佳情况,其中它可以在65个字节中实现,以说明自引用问题。假设 PPI_n 最初仅包含一轮 $SHA256()$,使得长度为65个字节。

[0266] 为了处理 PPI_n 的65个字节, PPI_n 中的操作码中必须存在两轮 $SHA256()$ 。因此, PPI_n 的长度增加到130个字节。现在,操作码中必须存在3轮 $SHA256()$ 才能处理 PPI_n ,这意味着 PPI_n 实际上是195个字节。很快就会发现自引用问题。

[0267] 这不是PUSHTXPARENT的问题,因为它的长度不会随原像长度而改变。操作码 $OP_HASH256$ 可以处理任何长度的原像,并且仍然可以使用单个操作码来计算。

[0268] 5.8. 方案:辅事务

[0269] 引入辅事务,该辅事务使用 Tx_{n-1} 的第一输出,然后由 Tx_n 的第一输入引用,如图4所示。辅事务中的锁定脚本在操作码中包含部分 $SHA256$ 哈希函数,而不是 Tx_n 。包括该事务的结果是,部分 $SHA256$ 哈希函数永远不会用作其自身的输入。

[0270] 如果 $dSHA256(Tx_n^A) = TxID_n^A$,其中 $TxID_n^A$ 是从 m_{n+1} 的第一输出点中提取的,在给定 m_{n+1} 和 Tx_n^A 作为输入的情况下,该组操作码PUSHTXAUXILIARY的结果为TRUE。

[0271] 操作码PUSHTXAUXILIARY包括以下内容:

[0272] 1. PUSHTX,如果 m_{n+1} 是事务 Tx_{n+1} 的sighash原像,该PUSHTX将 m_{n+1} 作为输入并输出TRUE

[0273] 2. PUSHTXPARENT,如果 Tx_n^A 是 $TxID_n^A$ 的哈希原像,该PUSHTXPARENT从 m_{n+1} 获取 Tx_n^A 和 $TxID_n^A$ 并输出TRUE

[0274] 因此,由 (m_{n+1}, Tx_n^A) 解锁包含PUSHTXAUXILIARY的脚本。

[0275] 定义脚本 $\mathcal{S}_3$,其为

[0276]

PUSHTXAUXILIARY + PROOFOFPROPAGATION OP_IF PROOFOFEXECUTION OP_ENDIF
--

[0277] 如图1所示的锁定脚本(其中 $ls_n^A = \mathcal{S}_3$)由解锁 $us_{n,0} = (m_{n+1}, Tx_n^A, mid_n, PPI_n, mid_{n-1}, PPI_{n-1})$,其中

[0278] $m_{n+1} = Tx_{n+1}$ 的sighash原像;

[0279] $Tx_n^A =$ 序列化辅事务;

[0280] $mid_n = partialSHA256(us_{n-1,0}, IV)$;

[0281] $PPI_n = us_{n-1,1} || ls_{n,0} || ls_{n,1} || pad_n$;

[0282] $mid_{n-1} = \text{partialSHA256}(us_{n-2,0} || us_{n-2,1}, IV)$; 以及

[0283] $PPI_{n-1} = ls_{n-1,0} || ls_{n-1,1} || pad_{n-1}$ 。

[0284] 此外,还定义了一组操作码HASHPUZZLE,如果 $hash(ls_n^A) = hash(s_3)$,其中 $hash(s_3)$ 是HASHPUZZLE中的固定常数,在给定 ls_n^A 作为输入的情况下,该组操作码HASHPUZZLE的结果为TRUE。

[0285] 将脚本 s_A 定义为:

[0286] $s_A = \text{PUSHTX} + \text{HASHPUZZLE}$

[0287] 对于所有n,将锁定脚本 $ls_{n,0}$ 重新定义为 s_A ,然后解锁脚本是 $us_n^A = m_n^A$,其中 m_n^A 是 Tx_n^A 的sighash原像。

[0288] 该锁定脚本强制 ls_n^A 包含上面定义的操作码。通过强制 s_3 解锁 $ls_{n,0}$ 中的哈希谜题,不需要更改PROOFOFEXECUTION和PROOFOFPROPAGATION的定义,并且辅事务在每次传输中都具有正确的格式,而无需进行显式检查。系统会自动强制执行所有辅事务。

[0289] 最后,如前所述,锁定脚本 $ls_{n,1}$ 是P2PKH(PK_n)。

[0290] 部分SHA256哈希函数永远不会出现在部分原像PPI中。它包含在 ls_n^A 中,后者输入到OP_HASH256。已经解决了自引用问题,而没有出现新的问题。

[0291] 5.9. 启动协议

[0292] 发行事务不具有祖父事务。这会引入需要修改 s_3 的特殊用例。发行事务可以通过至少两种方式来标识:

[0293] 1. 根据发行者拥有的公钥来核实事务上的签名。

[0294] 2. 为每个令牌提供唯一的令牌ID(UTID)。

[0295] 为了避免引入特定发行者,可以选择第二种方法,其中UTID被定义为发行事务 $TxID_1$ 的TxID。定义操作码UTIDPRESENT,以便当 Tx_{n+1} 中的UTID存在于 Tx_n 中时,在给定 m_{n+1} 作为输入的情况下,该UTIDPRESENT的结果为TRUE。此外,还定义操作码DEFINEUTID,以便当 Tx_n 中的UTID和从第二输出点到 $TxID_n$ 的 $TxID_{n-1}$ 相等时,在给定 m_{n+1} 作为输入的情况下,该DEFINEUTID的结果为TRUE。应当注意的是,只有在由于 $UTID = TxID_1$ 而使得 $n=2$ 的情况下才会出现这种情况。这意味着UTID不存在于 Tx_1 中。相反,对于所有 $n \geq 2$,它被强制存在于 Tx_n 的锁定脚本中。

[0296] 此外,还定义SAMEOUTPOINT,在给定 $us_{n,1}$ 的输出点和 us_n^A 的输出点(如果它们包含相同的TxID)的情况下,该SAMEOUTPOINT的结果为TRUE。

[0297] 将脚本 s 定义为

[0298]

```

PUSHTXAUXILIARY + PROOFOFPROPAGATION + SAMEOUTPOINT +
UTIDPRESENT

OP_IF
    PROOFOFEXECUTION
OP_ELSE
    DEFINEUTID
OP_ENDIF

```

[0299] 如果UTIDPRESENT被执行并输出1 (即, 当 $n > 2$ 时), 则执行包含PROOFOFPROPAGATION的“if”语句。如果UTIDPRESENT被执行并输出0 (即, 当 $n = 2$ 时), 则执行包含DEFINEUTID的“else”语句。

[0300] 使用 $ls_n^A = s$ 优化锁定脚本, 如图1所示, 该锁定脚本通过解锁脚本 $us_{n,0} = (m_{n+1}, Tx_n^A, mid_n, PPI_n, mid_{n-1}, PPI_{n-1})$ 解锁。这和其他解锁脚本一起与先前解锁脚本保持不变。锁定脚本的唯一更改如下:

[0301] ● $ls_{n,0}$ 现在包含 $hash(s)$ 而不是 $hash(s_3)$ 作为哈希谜题中的固定数据; 并且

[0302] ● $ls_{n,1}$ 还在末尾包含OP_RETURN<UTID>。

[0303] 应当注意的是, 每组操作码中包含的操作码都有一些重叠, 例如PUSHTX和PUSHTXPARTIALPARENT都在PROOFOFPROPAGATION和PROOFOFEXECUTION中。这些操作码可以利用相同的函数并复制任何所需的信息以提高效率。

[0304] 强制父事务的两个输出 $TxID_{n-1} || 0, TxID_{n-1} || 1$ 分别在辅事务和当前事务中花费。如果不强制以这种方式花费输出, 则可能使事务链分叉, 并且看起来仍然像是正确地遵守了协议。这可能会引入攻击向量, 攻击者可以在没有第二输出的情况下轻松地花费第一输出。

[0305] 就目前而言, 可以通过在没有第一输出的情况下花费第二输出来烧毁令牌, 因为这将锁定 Tx_n^A 的输出, 该输出必须与 Tx_n 的第二输出一起花费。如果不重新引入自引用问题, 则无法在脚本中阻止这种情况, 因为辅事务 (在操作码中包含部分SHA256哈希函数) 将是第二解锁脚本的一部分。因此, 部分SHA256哈希函数必须在操作码中自行处理, 这就是自引用问题。由于需要签名来花费 Tx_n 的第二输出, 因此在以这种方式燃烧令牌的情况下签名将证明这一点。

[0306] 此外, 还可以通过在第二输出中花费辅事务来烧毁令牌, 因为这会重新引入事务大小膨胀和部分SHA256的自引用问题。然而, 本公开的方案通过要求PROOFOFEXECUTION仅从 m_{n+1} 中的第二输出点中提取输出点来防止这种情况, 使得仅当第二输出点不是辅事务时事务才有效。

[0307] 如前所述, 通过要求PROOFOFEXECUTION从 m_{n+1} 中的第二输出点中提取输出点, 强制 Tx_n^A 的输出点必须与 Tx_n 的第二输出点一起花费。这是因为 Tx_n^A 的锁定脚本中的操作码仅当 Tx_n 和 Tx_{n-1} 中的锁定脚本包含多组相同操作码和相同UTID (已经被定义为发行事务的

$TxID$)时才返回TRUE。仅当第二输出点是 Tx_n 的第二输出时才会出现这种情况—任何其他输出点都不会包含相同的操作码和UTID。因此, Tx_n^A 的输出必须与 Tx_n 的第二输出一起花费, 否则将无效。

[0308] 脚本 $\mathcal{S}$ 是上述辅锁定脚本的示例。脚本 $\mathcal{S}_A$ 是上述主锁定脚本的示例。子脚本A表示主锁定脚本对辅事务强制执行条件。

[0309] 5.10. 令牌传输协议

[0310] 考虑由发行者I向爱丽丝103a发行令牌, 然后传输令牌两次, 首先向鲍勃103b传输令牌, 鲍勃向查理103c传输令牌。

[0311] 1. 发行者生成 Tx_1 , 其中 $ls_{1,0} = \mathcal{S}$ 和 $ls_{1,1} = P2PKH(PK_A)$ 在第3.3节中定义, 以将令牌支付到爱丽丝的公钥。

[0312] 2. 爱丽丝生成第3.4节中定义的辅事务 Tx_1^A 和第3.3节中定义的 Tx_2 , 并将这些事务发送给鲍勃。

[0313] 3. 鲍勃使用以下步骤核实这些事务。

[0314] 4. 鲍勃生成 Tx_2^A 和 Tx_3 , 并将这些事务发送给查理。

[0315] 5. 查理使用以下步骤核实这些事务。

[0316] 进一步的传输将以与向查理传输相同的方式执行。

[0317] 5.11. 第n次传输的核实者步骤

[0318] 1. 接收 Tx_{n-1} 的默克尔证明和事务 Tx_n 。使用 Tx_n 中的数据计算 $TxID_{n-1}$, 并确认默克尔证明有效。

[0319] 2. 从 Tx_{n-1} 中的哈希谜题中提取辅锁定脚本的哈希, 并验证该哈希对应于该辅锁定脚本的已知哈希。

[0320] 3. 提取UTID, 并确认它已经使用来自发行者的有效UTID列表或发行事务上的签名进行签名。

[0321] 应当注意的是, 虽然核实者需要执行一些步骤, 但是这类似于任何秘密现金交换, 例如仅当票据具有正确的标记时才接受该票据, 或者仅当区块链事务包含正确公钥的P2PKH时才接受该区块链事务。这是矿工核实令牌, 因为在给定正确发行的事务的情况下, 任何传输都必须始终产生另一个有效令牌, 而无需核实者投入任何精力。

[0322] 5.12. 详细的示例性协议

[0323] 考虑由发行者I向爱丽丝103a发行令牌, 然后传输令牌两次, 首先向鲍勃103b传输令牌, 鲍勃向查理103c传输令牌。进一步的传输将以与向查理103c传输相同的方式执行。

[0324] 发行

[0325] 1. 发行者生成花费需要其签名的任意输出的事务, 该输出表示如图9所示的令牌, 并将其发送给爱丽丝。该令牌的UTID是 $TxID_1$ 。该事务包含P2PKH输出, 要求爱丽丝103a解锁该输出。另一个输出强制辅事务在输出中包含正确的函数。

[0326] 令牌传输

[0327] 2. 如图10所示, 爱丽丝103a生成花费 Tx_1 的第一输出的辅事务。该事务将强制 $\mathcal{S}_A$ 和UTID出现在后续事务中, 并且仅在与来自 Tx_1 的P2PKH输出相同的事务中花费。

[0328] 3.为了将该令牌传输给鲍勃103b,爱丽丝103a生成并发送图11中给出的事务。如果通过验证(如上详细描述),则鲍勃103b接受该令牌。

[0329] 4.如图12所示,鲍勃生成花费 Tx_2 的第二输出的辅事务。

[0330] 5.为了将该令牌传输给查理103c,鲍勃103b生成图13中给出的事务并将其发送给查理103c。如果通过验证(如上详细描述),则查理103c接受该令牌。

[0331] 进一步的传输采用与图12和图13中的事务相同的格式以相同的方式执行。

[0332] 第n次令牌传输的一般格式如图14所示。辅事务 Tx_{n-1}^A 如图15所示。

[0333] 6.进一步评论

[0334] 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。

[0335] 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。

[0336] 在本发明的优选实施例中,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、传播和存储中的所有所述功能。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)。

[0337] 在本发明的其他实施例中,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。

[0338] 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。

[0339] 应当理解的是,上述实施例仅通过示例的方式进行描述。更通俗地说,可根据下述任何一个或多个语句提供一种方法、装置或程序。

[0340] 语句1.一种计算机实现的方法,用于强制锁定脚本在区块链事务链中传播,其中所述方法由第一方执行并且包括:

[0341] 生成包括第一输出的第一主事务,其中所述第一输出包括主锁定脚本,其中所述主锁定脚本被配置为在由第一辅事务的第一解锁脚本执行时:i)验证所述第一解锁脚本包括基于所述第一辅事务而生成的第一消息,以及ii)基于所述第一消息来验证所述第一辅事务的第一输出包括辅锁定脚本;其中所述辅锁定脚本被配置为在由第二主事务的第一解锁脚本执行时:i)验证所述第二主事务的所述第一解锁脚本包括基于所述第二主事务而生

成的第二消息, ii) 验证所述第二主事务的所述第一解锁脚本包括第一数据项, 所述第一数据项包括所述第一主事务的一个或多个锁定脚本, 以及 iii) 基于所述第二消息和所述第一数据项来验证所述第二主事务的第一输出包括所述主锁定脚本。

[0342] 语句2. 根据语句1所述的方法, 所述方法包括:

[0343] 生成所述第一辅事务, 其中所述第一辅事务包括第一输入, 所述第一辅事务的所述第一输入引用所述第一主事务的所述第一输出, 并且包括所述第一消息, 以及包含所述辅锁定脚本的第一输出。

[0344] 语句3. 根据语句1或2所述的方法, 其中所述主锁定脚本包括所述辅锁定脚本的哈希, 其中所述第一消息包括候选辅锁定脚本, 并且其中所述的验证所述第一辅事务的所述第一输出包括所述辅锁定脚本包括: 所述主锁定脚本对所述候选辅锁定脚本进行哈希处理、并验证所述候选辅锁定脚本的哈希与所述辅锁定脚本的所述哈希匹配。

[0345] 语句4. 根据前述任一项语句所述的方法, 其中所述第一主事务包括第二输出, 并且其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时 iii) 验证: 所述第一主事务的所述第二输出由所述第二主事务的第二输入解锁。

[0346] 语句5. 根据语句4所述的方法, 其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时: iv) 验证所述第一主事务的所述第一解锁脚本包括包含所述第一辅事务的第二数据项; 并且其中所述的验证所述第二输出由所述第二主事务的所述第二输入解锁包括: a) 从所述第二消息中提取由所述第二主事务的所述第二输入解锁的所述第二输出的相应事务标识符, b) 从所述第二数据项中提取由所述第一辅事务的所述第一输入解锁的所述第一输出的相应事务标识符, 以及 c) 验证在步骤 a) 期间提取的所述相应事务标识符与在步骤 b) 期间提取的所述相应事务标识符匹配。

[0347] 语句6. 根据前述任一项语句所述的方法, 其中所述第一主事务具有通过对所述第一主事务应用哈希函数而生成的相应事务标识符, 其中所述第二主事务的所述第一解锁脚本包括第三数据项, 其中所述第三数据项包括对所述第一主事务应用的所述哈希函数的中间状态 (midstate), 并且其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时: iv) 基于所述第一数据项和所述第三数据项来生成候选事务标识符, 以及验证所述候选事务标识符与所述第一主事务标识符的所述相应事务标识符匹配。

[0348] 语句7. 根据前述任一项语句所述的方法, 其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时: v) 验证所述第二主事务包括特定事务标识符。

[0349] 语句8. 根据前述任一项语句所述的方法, 所述方法包括:

[0350] 生成所述第二主事务, 其中所述第二主事务包括第一输入, 所述第二主事务的所述第一输入引用所述第一辅事务的所述第一输出, 并且包括所述第二消息和所述第一数据项, 并且其中所述第二主事务包括第一输出, 所述第二主事务的所述第一输出包括所述主锁定脚本。

[0351] 语句9. 根据前述任一项语句所述的方法, 所述方法包括: 向第三方发送所述第一辅事务和所述第二主事务。

[0352] 语句10. 根据语句9所述的方法, 所述方法包括: 向所述第三方发送默克尔证明, 用于证明所述第一主事务已经被记录在区块链的区块中。

[0353] 语句11. 根据前述任一项语句所述的方法, 其中所述主锁定脚本和/或辅锁定脚本

包括与令牌的所有权相关的数据。

[0354] 语句12.根据前述任一项语句所述的方法,其中所述第一主事务是所述区块链事务链中的初始主事务。

[0355] 语句13.根据语句11和12所述的方法,其中所述第一方是所述令牌的发行者。

[0356] 语句14.根据从属于语句7的语句12或13所述的方法,其中所述特定事务标识符是所述第一主事务的所述相应事务标识符。

[0357] 语句15.根据前述任一项语句所述的方法,其中所述第一主事务包括第一计数器值(counter value),并且其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时:验证所述第二主事务包括第二计数器值,其中所述第二计数器值是所述计数器值的迭代。

[0358] 语句16.根据语句1至11中任一项所述的方法,其中所述第一主事务不是所述区块链事务链中的初始事务,并且其中所述第一主事务包括引用包含所述辅锁定脚本的先前辅事务的第一输出的第一输入,其中所述先前辅事务包括引用包含所述主锁定脚本的先前主事务的第一输出的第一输入,并且其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时,vi)验证:所述第二主事务的所述第一解锁脚本包括包含所述先前主事务的一个或多个锁定脚本的第四数据项,以及vii)基于所述第二消息和所述第四数据项来验证:所述第二主事务的所述第一输出包括所述主锁定脚本。

[0359] 语句17.根据语句16所述的方法,其中所述先前主事务具有通过对所述先前主事务应用哈希函数而生成的相应事务标识符,其中所述第二主事务的所述第一解锁脚本包括第五数据项,其中所述第五数据项包括对所述先前主事务应用的所述哈希函数的中间状态,并且其中所述辅锁定脚本被配置为在由所述第二主事务的所述第一解锁脚本执行时:viii)基于所述第四数据项和所述第五数据项来生成候选事务标识符,以及验证所述候选事务标识符与所述先前主事务标识符的所述相应事务标识符匹配。

[0360] 语句18.根据语句16或17所述的方法,其中所述先前主事务是所述区块链事务链中的初始主事务,并且其中所述方法包括:验证所述初始主事务已经使用与预定公钥对应的签名进行签名。

[0361] 语句19.根据前述任一项语句所述的方法,所述方法包括:向区块链网络提交所述第一主事务。

[0362] 语句20.根据前述任一项语句所述的方法,所述方法包括:向区块链网络提交所述第一辅事务和/或所述第二主事务。

[0363] 语句21.一种计算机实现的方法,用于验证锁定脚本已经在区块链事务链中被强制执行,其中所述方法由第二方执行并且包括:

[0364] 获取第二主事务,其中所述第二主事务包括第一输入和第一输出,其中所述第一输入包括基于所述第二主事务而生成的第二消息、包含解锁第一主事务的第一输出的第一输入的第一辅事务、以及包含所述第一主事务的一个或多个锁定脚本的第一数据项,其中所述第一主事务的所述第一输出包括主锁定脚本,并且所述第一辅事务的第一输出包括辅锁定脚本,并且其中所述第二主事务的所述第一输出包括所述主锁定脚本;

[0365] 验证第一主事务已经记录在区块链上;以及,

[0366] 对以下进行验证:所述第一主事务的所述主锁定脚本被配置为在由所述第一辅事

务的第一解锁脚本执行时 i) 验证所述第一解锁脚本包括基于所述第一辅事务而生成的第一消息, 和 ii) 基于所述第一消息来验证所述第一辅事务的第一输出包括辅锁定脚本; 其中所述辅锁定脚本被配置为在由所述第二主事务的第一解锁脚本执行时 i) 验证所述第一解锁脚本包括所述第二消息, ii) 验证所述第一解锁脚本包括所述第一数据项, 和 iii) 基于所述第二消息和所述第一数据项来验证所述第二主事务的所述第一输出包括所述主锁定脚本, 从而验证: 所述主锁定脚本已经在所述第一主事务和所述第二主事务中强制执行。

[0367] 语句22. 根据语句21所述的方法, 其中所述主锁定脚本包括所述辅锁定脚本的哈希, 其中所述第一消息包括候选辅锁定脚本, 并且其中所述主锁定脚本被配置为在由所述第一辅事务的所述第一解锁脚本执行时: 通过对所述候选辅锁定脚本进行哈希处理并验证所述候选辅锁定脚本的哈希与所述辅锁定脚本的所述哈希匹配, 来验证所述第一辅事务的所述第一输出包括所述辅锁定脚本, 并且其中所述的验证所述主锁定脚本被配置为执行步骤 i) 和 ii) 包括: 验证所述主锁定脚本包括所述辅锁定脚本的所述哈希作为哈希谜题的一部分。

[0368] 语句23. 根据语句21或22所述的方法, 所述的验证所述第一主事务已经记录在所述区块链上包括: 获取默克尔证明, 用于证明所述第一主事务已经记录在所述区块链的区块中。

[0369] 语句24. 根据语句23所述的方法, 其中所述第一主事务具有通过对所述第一主事务应用哈希函数而生成的相应事务标识符, 其中所述第二主事务的所述第一输入包括第三数据项, 其中所述第三数据项包括对所述第一主事务应用的所述哈希函数的中间状态, 并且其中所述的验证所述第一主事务已经记录在所述区块链上包括: 基于所述第一数据项和所述第三数据项来生成候选事务标识符; 以及验证所述默克尔证明对所述候选事务标识符有效。

[0370] 语句25. 根据语句23或24所述的方法, 所述方法包括: 获取所述第一主事务, 并且其中所述的验证所述第一主事务已经记录在所述区块链是基于所述获取的默克尔证明和所述获取的第一主事务而进行的。

[0371] 语句26. 根据语句19或其任何从属语句所述的方法, 所述方法包括: 验证所述第二主事务包括初始主事务的所述相应事务标识符。

[0372] 语句27. 根据语句19或其任何从属语句所述的方法, 所述方法包括:

[0373] 生成第二辅事务, 其中所述第二辅事务包括第一输入和第一输出, 所述第二辅事务的所述第一输入引用所述第二主事务的所述第一输出, 并且包括基于所述第二辅事务而生成的第三消息, 所述第一输出包含所述辅锁定脚本; 以及

[0374] 生成第三主事务, 其中所述第三主事务包括第一输入, 所述第三主事务的所述第一输入引用所述第二辅事务的所述第一输出, 并且包括基于所述第三主事务而生成的第四消息、所述第一数据项、所述第二辅事务、和包含所述第二主事务的一个或多个锁定脚本的第六数据项, 并且其中所述第三主事务包括第一输出, 所述第三主事务的所述第一输出包括所述主锁定脚本。

[0375] 语句28. 根据语句27所述的方法, 其中所述辅锁定脚本被配置为在由所述第三主事务的第一解锁脚本执行时: vi) 验证所述第二主事务的所述第一解锁脚本包括所述第三数据项, 以及 vii) 基于所述第三消息和所述第六数据项来验证所述第三主事务的所述第一

输出包括所述主锁定脚本。

[0376] 语句29.根据语句27或28所述的方法,所述方法包括:向第三方发送所述第二辅事务和所述第三主事务。

[0377] 语句30.根据语句29所述的方法,所述方法包括:向所述第三方发送默克尔证明,用于证明所述第二主事务已经记录在所述区块链的区块中。

[0378] 语句31.根据语句21或其任何从属语句所述的方法,其中所述主锁定脚本和/或辅锁定脚本包括与令牌的所有权相关的数据,并且其中所述第一主事务和/或所述第一辅锁定脚本已经由令牌发行者生成。

[0379] 语句32.根据语句27或其任何从属语句所述的方法,所述方法包括:向所述区块链提交所述第二辅事务和所述第三主事务。

[0380] 语句33.一种计算机设备,所述计算机设备包括:

[0381] 存储器,所述存储器包括一个或多个存储器单元;以及,

[0382] 处理装置,所述处理装置包括一个或多个处理单元,其中所述存储器存储被设置在所述处理装置上运行的代码,所述代码被配置为当在所述处理装置上运行时,执行根据语句1至29中任一项所述的方法。

[0383] 语句34.一种计算机程序,所述计算机程序包含在计算机可读存储器上并且被配置为当在一个或多个处理器上运行时,执行根据语句1至32中任一项所述的方法。

[0384] 根据本文公开的另一方面,可以提供一种方法,所述方法包括所述第一方和所述第二方的动作。根据本文公开的另一方面,可以提供一种包括区块链节点的动作的方法,所述区块链节点被配置为至少处理所述第一主事务和所述第一辅事务。

[0385] 根据本文公开的另一方面,可以提供一种系统,所述系统包括所述第一方和所述第二方的计算机设备。根据本文公开的另一方面,可以提供一种区块链节点,所述区块链节点被配置为至少处理所述第一主事务和所述第一辅事务。

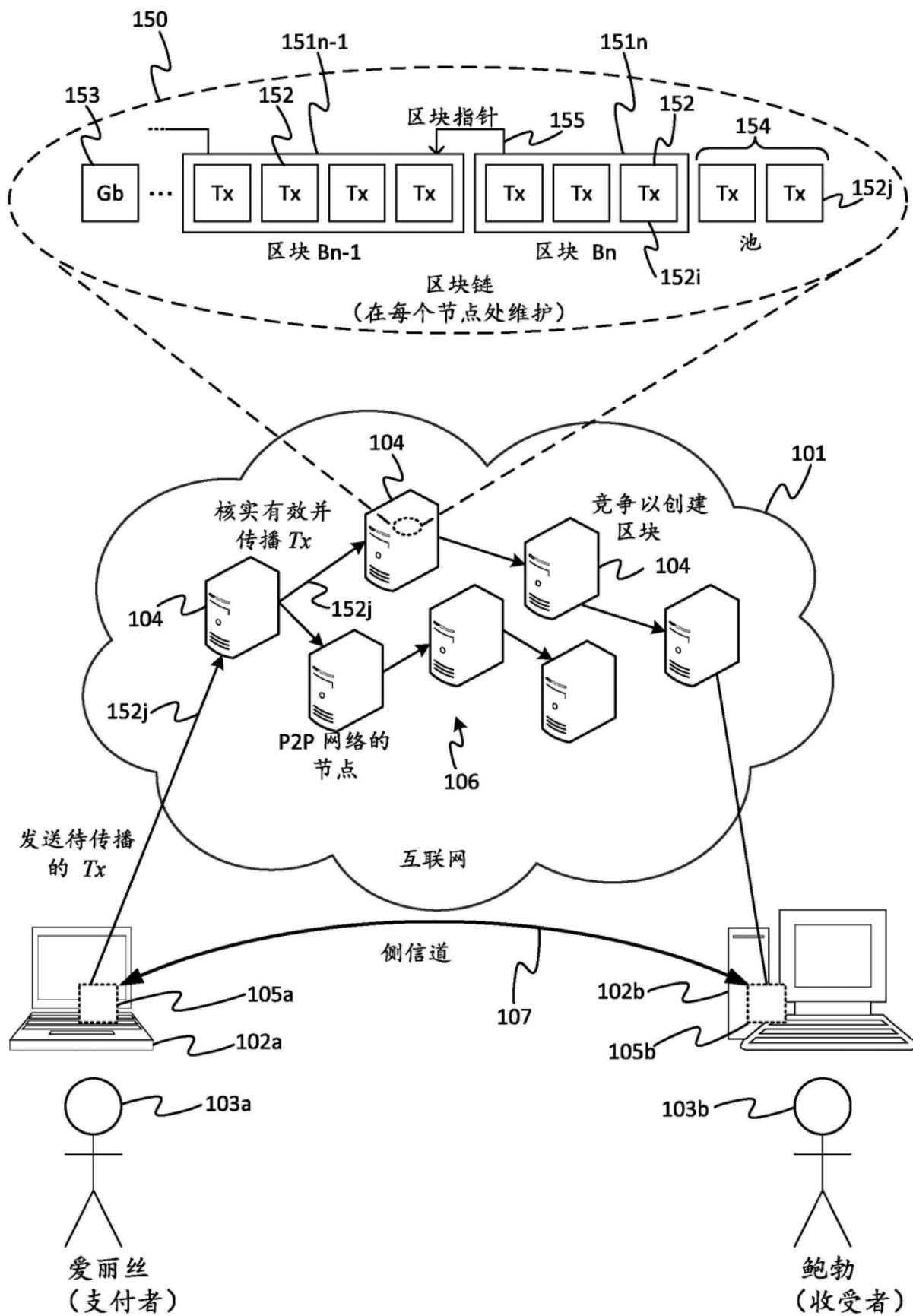
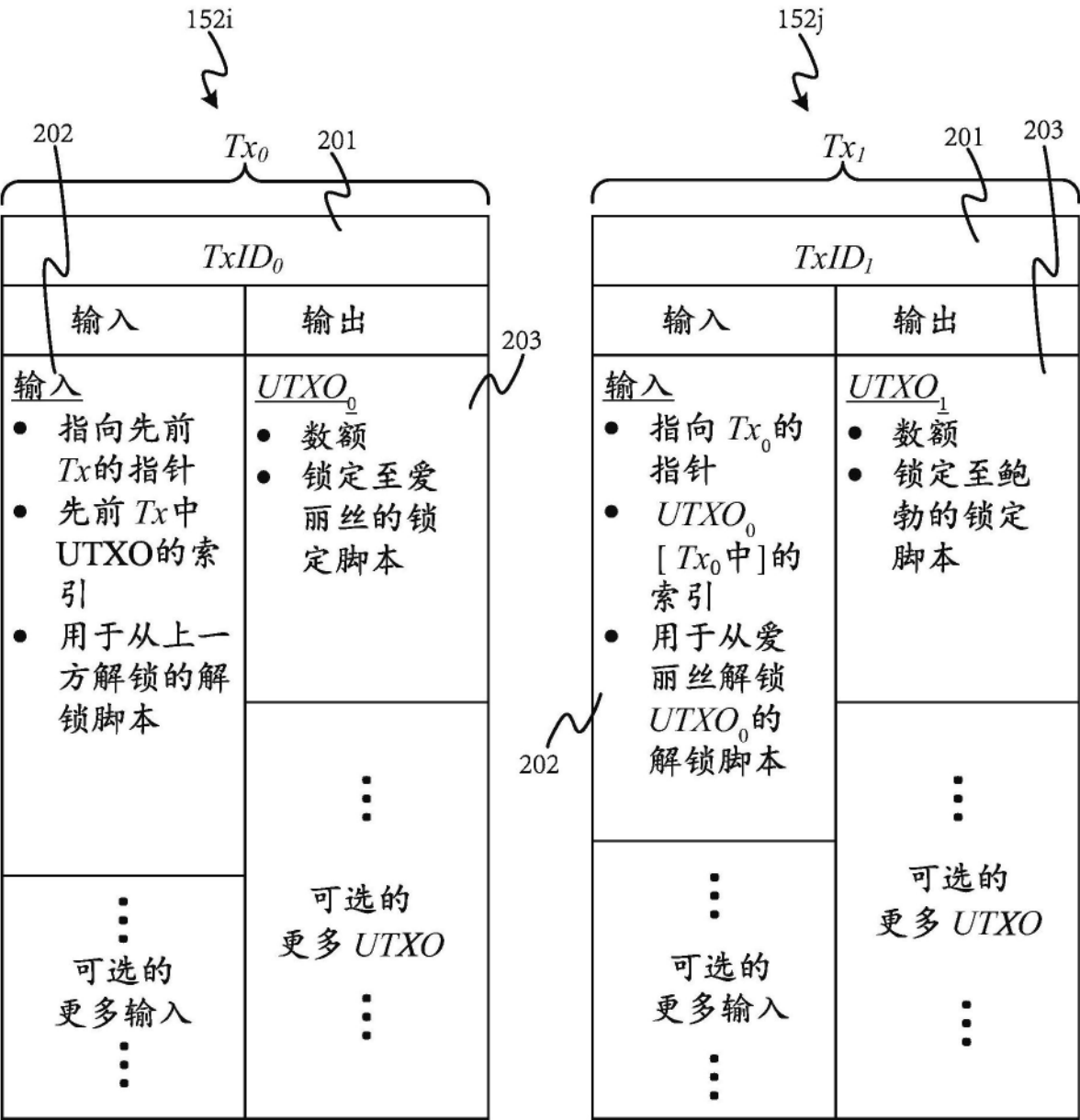


图1



从爱丽丝到鲍勃
的事务



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

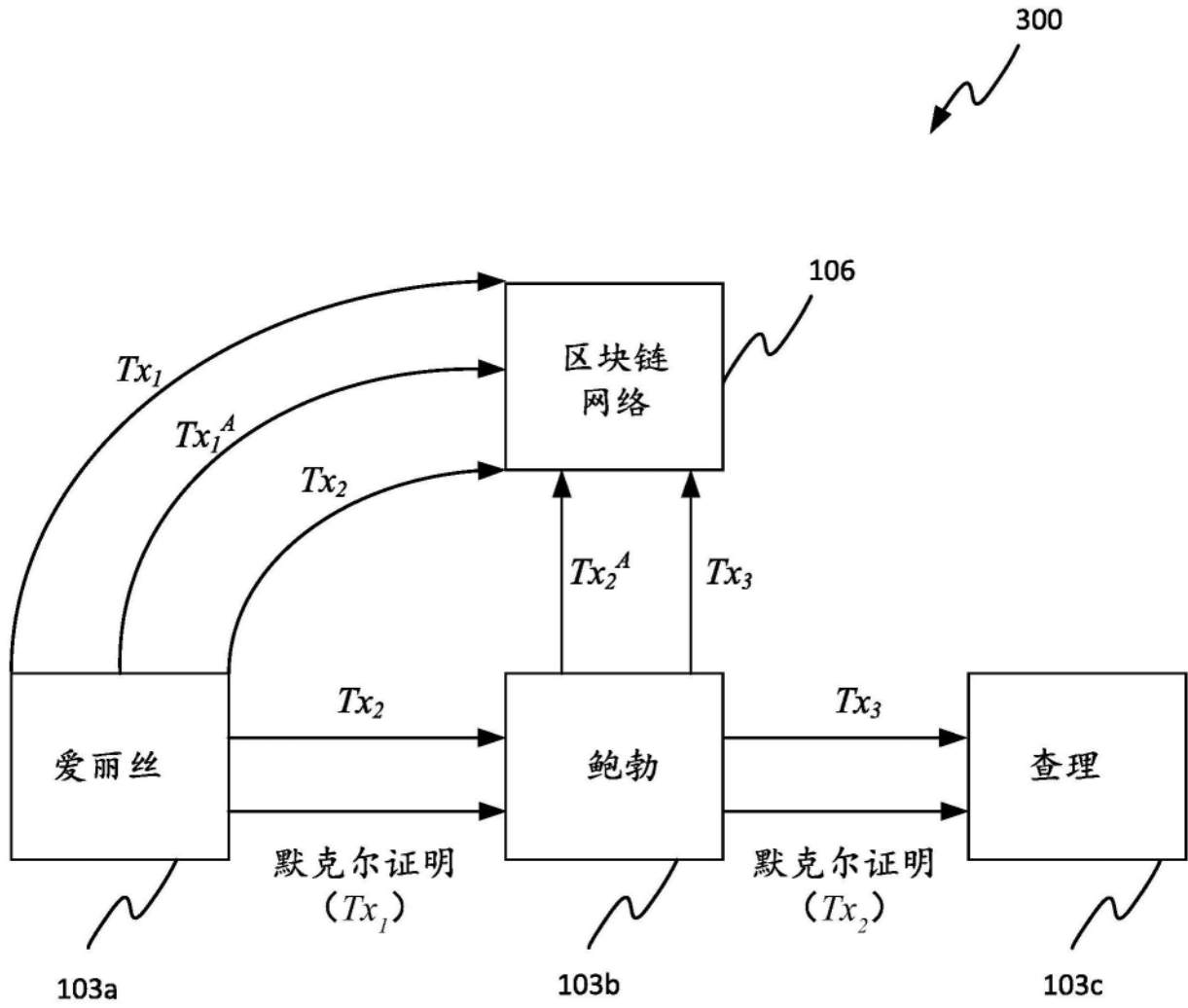


图3

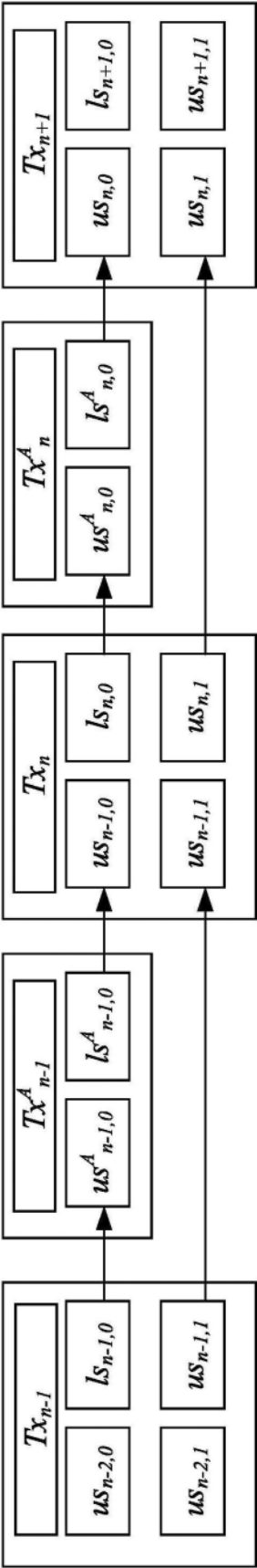


图4

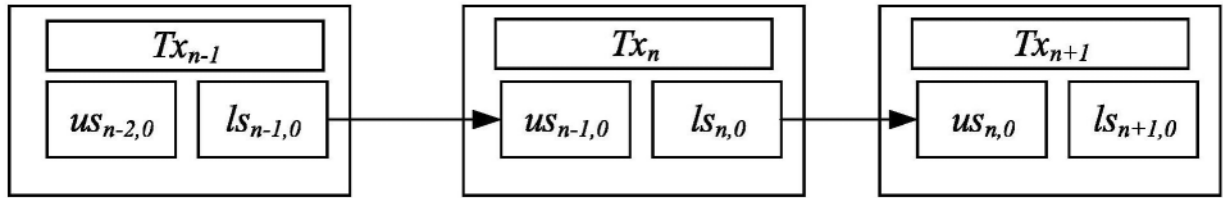


图5

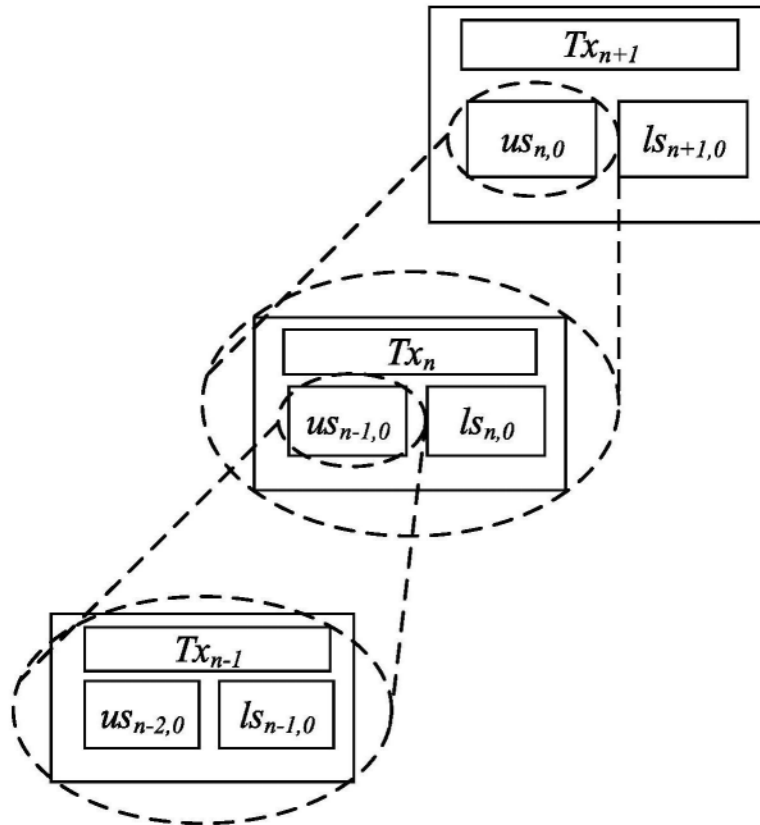


图6

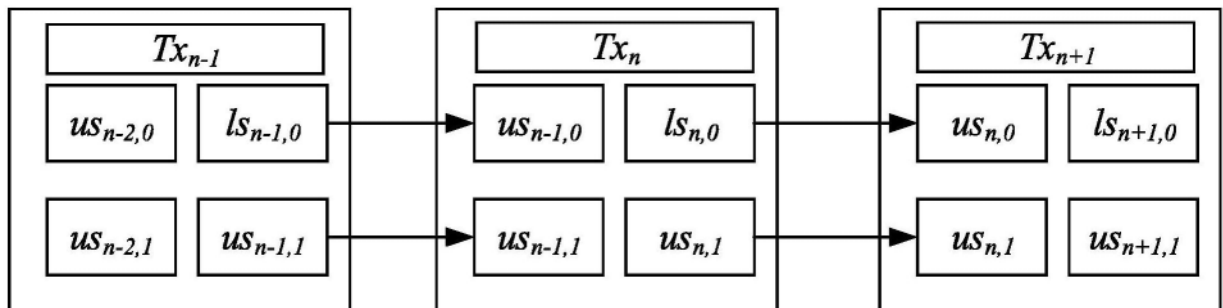


图7

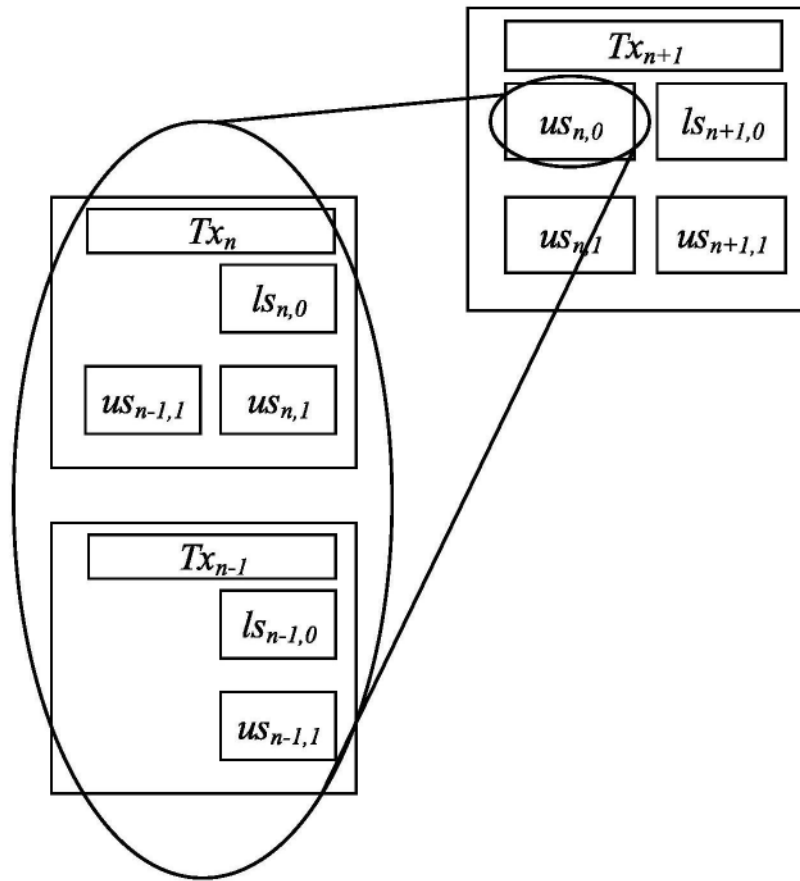


图8

Tx_1		
输入		输出
$TxID_0 i$	sig_{P_I}, P_I	$\mathcal{S}_A$
		P2PKH(P_A)

图9

Tx_1^A		
输入		输出
$TxID_1 0$	m_1^A	$\mathcal{S}$

图10

Tx_2		
输入		输出
$TxID_1^A 0$	$mid_1 + PPI_1,$ Tx_1^A, m_2	$\mathcal{S}_A$
$TxID_1 1$	sig_{P_A}, P_A	$P2PKH(P_B)$ $OP_RETURN <UTID>$

图11

Tx_2^A		
输入		输出
$TxID_2 0$	m_2^A	$\mathcal{S}$

图12

Tx_3		
输入		输出
$TxID_2^A 0$	$mid_1 + PPI_1,$ $mid_2 + PPI_2, Tx_2^A,$ m_3	$\mathcal{S}_A$
$TxID_2 1$	sig_{P_B}, P_B	P2PKH(P_C) OP_RETURN $\langle UTID \rangle$

图13

Tx_n		
输入		输出
$TxID_{n-1}^A 0$	$mid_{n-2} + PPI_{n-2},$ $mid_{n-1} + PPI_{n-1},$ Tx_{n-1}^A, m_n	$\mathcal{S}_A$
$TxID_{n-1} 1$	$sig_{P_{N-1}}, P_{N-1}$	P2PKH(P_N) OP_RETURN $\langle UTID \rangle$

图14

Tx_{n-1}^A		
输入		输出
$TxID_n 0$	m_{n-1}^A	s

图15