(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111247547 B
(45)授权公告日 2024.09.20

(21)申请号 201880034526.8
(22)申请日 2018.05.23
(65)同一申请的已公布的文献号
申请公布号 CN 111247547 A
(43)申请公布日 2020.06.05
(30)优先权数据
1708488.0 2017.05.26 GB
1708491.4 2017.05.26 GB
1708493.0 2017.05.26 GB
(85)PCT国际申请进入国家阶段日
2019.11.25
(86)PCT国际申请的申请数据
PCT/IB2018/053653 2018.05.23
(87)PCT国际申请的公布数据
WO2018/215949 EN 2018.11.29

(73)专利权人 区块链控股有限公司
地址 安提瓜和巴布达圣约翰
(72)发明人 莹·陈 迪安·克莱默
(74)专利代理机构 北京市竞天公诚律师事务所 11770
专利代理师 陈果

(51)Int.Cl.
G06Q 20/02(2006.01)
G06Q 20/06(2006.01)
(56)对比文件
KR 101701131 B1,2017.02.13
US 2017091750 A1,2017.03.30
审查员 朱艳君

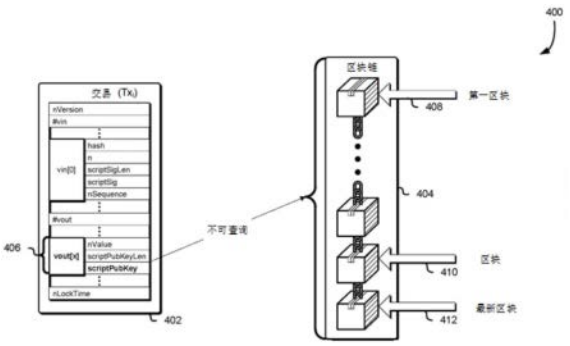
权利要求书2页 说明书43页 附图29页

(54)发明名称

基于脚本的区块链交互

(57)摘要

可提供一种计算机实现的方法,所述方法可使用区块链网络来实现。所述计算机实现的方法包括:i)在区块链网络中的节点处接收与数字资产相关的第一交易,所述第一交易至少指定:a)对第二交易的第一约束集合,所述第二交易用于转移对数字资产的控制,所述第一约束集合包括一个或多个约束,所述一个或多个约束使得所述第二交易包含来自区块链的数据集合;和b)对所述第二交易的第二约束集合,所述第二约束集合包括一个或多个约束,所述一个或多个约束与所述数据集合的数据项相关联;i i)验证所述第一约束集合和所述第二约束集合得以满足;和iii)至少部分基于所述验证来重新关联所述数字资产。



CN 111247547 B

1.一种计算机实现的方法,包括:

在区块链网络中的节点处接收与数字资产相关的第一交易,所述第一交易包括锁定脚本,所述锁定脚本至少包括:

对第二交易的第一约束集合,所述第二交易用于转移对数字资产的控制,所述第一约束集合包括一个或多个约束,所述一个或多个约束使得所述第二交易包含来自区块链的数据集合;其中所述第一约束集合包括这样的约束:所述数据集合包括所述区块链的区块头;和

对所述第二交易的第二约束集合,所述第二约束集合包括一个或多个约束,所述一个或多个约束与所述数据集合的数据项相关联;

验证所述第一约束集合和所述第二约束集合得以满足;和

至少部分基于所述验证来重新关联所述数字资产。

2.根据权利要求1中所述的计算机实现的方法,其中在所述第二交易的节点处接收所述数据集合。

3.根据权利要求1或2中所述的计算机实现的方法,还包括验证所述第二交易以作为所述验证的结果。

4.根据权利要求3中所述的计算机实现的方法,其中成功执行验证所述第二交易,而未验证创建了所述第二交易的实体有权访问秘密信息。

5.根据权利要求1所述的计算机实现的方法,其中所述第一约束集合包括这样的约束:所述数据集合包括来自所述区块链的区块的第三交易。

6.根据权利要求1所述的计算机实现的方法,其中所述第一约束集合包括这样的约束:所述数据集合包括区块头链,所述区块头链包括区块头的有序集合,所述区块头的有序集合包括多个区块头,所述区块头的有序集合指定了与所述多个区块头相关联的顺序。

7.根据权利要求1所述的计算机实现的方法,其中所述第二约束集合包括对所述数据集合的数据项的值的约束。

8.根据权利要求1所述的计算机实现的方法,其中所述第二约束集合包括从与所述数据集合的数据项相关联的一个或多个值得出的约束。

9.根据权利要求1所述的计算机实现的方法,其中所述第一约束集合包括这样的约束:从公共区块链接收所述数据集合。

10.根据权利要求9所述的计算机实现的方法,其中所述节点通过至少以下方式来确定所述从所述公共区块链接收所述数据集合的约束是否得以满足:

在所述第二交易中接收第一区块集合和第二区块集合,所述第一区块集合包括了从区块链网络接收的区块之前的区块,所述第二区块集合包括了从区块链网络的接收的区块之后的区块;

验证所述第一区块集合链接至从所述区块链网络接收的区块;

验证所述第二区块集合链接至从所述区块链网络接收的区块;

验证所述第一区块集合和所述第二区块集合有效;和

验证所述第一区块集合和所述第二区块集合中的每个区块都具有比预定值更大的难度值。

11.一种系统,包括:

处理器;和

存储器,包含可执行的指令,所述处理器执行所述指令使得所述系统执行权利要求1至10中的任意一项所述的计算机执行的方法。

12.一种存储了可执行指令的非暂时性计算机可读存储介质,计算机系统的处理器执行所述可执行指令,使得所述计算机系统至少执行权利要求1至10中任意一项所述的计算机实现的方法。

基于脚本的区块链交互

技术领域

[0001] 本发明主要涉及分布式账本技术,包括区块链交易(transaction),特别涉及使得将来自区块链、区块头、区块和区块链交易的字段注入至交易脚本中。本发明特别适合但不限于在基于区块链状态的交易中使用。

背景技术

[0002] 在本文中,我们使用“区块链”一词来涵盖所有形式的基于计算机的电子分布式账本,包括基于共识的区块链和交易链技术、允许和不允许的账本、共享账本及其变体。应注意的是,包括非商业应用在内的其它区块链应用和协议也落入本发明的范围内。

[0003] 如本文所使用的,“数字资产”是由区块链管理的资源的单位。数字资产是由区块链管理的资源的单位。可以考虑到,在实施例中,数字资产可以在其他情况下附加地或替代地使用。注意,本发明虽然适用于数字资产的控制,但是本质上是技术性的,并且可以在其他利用区块链数据结构的情况下使用,而不必涉及数字资产的转移。本发明中使用的“数字资产”可以指一种或多种数字资产。例如,一个交易可能有多个输入,并且这些输入中的每一个都可以代表不同的数字资产。在该示例中,控制权被转移的数字资产可以是多个数字资产的集合,该集合本身是数字资产。类似地,交易可以细分和/或组合那些多个输入以产生一个或多个输出,使得例如输入的数量和输出的数量不同。

发明内容

[0004] 本发明描述了一个或多个基于区块链的计算机程序的技术方面。基于区块链的计算机程序是记录在区块链交易中的机器可读和可执行程序。基于区块链的计算机程序可以包括可以处理输入以产生结果的规则,然后可以根据这些结果来执行动作。如果锁定脚本可以访问解锁和先前的交易,则可以利用区块链来启用高度灵活和复杂的基于区块链的计算机程序。当前研究的一个领域是使用基于区块链的计算机程序来实现“智能合约”。与以自然语言编写的传统合同不同,智能合约是旨在自动执行机器可读合约或协议条款的计算机程序。

[0005] 在实施例中,尽管可以在智能合约中的特定步骤编码与特定实体的交互,但是智能合约也可以自动执行和自我执行。在一些示例中,自动执行是指智能合约的执行,该合约成功执行以启用UTXO的传输。注意,在这样的示例中,能够使得UTXO转移的“实体”是指能够创建解锁脚本而无需证明知晓某些秘密的实体。也就是说,可以在不验证数据源(例如,创建解锁交易的实体)有权访问加密秘密(例如,私密非对称密钥,对称密钥等)的情况下,验证解锁交易。同样,在这样的示例中,自我执行是指使区块链网络的验证节点根据约束来执行解锁交易。在一些示例中,从技术意义上使用“解锁”UTXO,是指创建引用该UTXO并作为有效执行的解锁交易。解锁UTXO在本领域中也可以称为花费UTXO。

[0006] 区块链交易输出包括锁定脚本和有关数字资产所有权的信息。锁定脚本(也可以称为“负载”)通过指定转移UTXO所需满足的条件来“锁定”数字资产。例如,锁定脚本可能要

求在解锁脚本中提供某些数据以解锁关联的数字资产。锁定脚本也称为“脚本公钥”(scriptPubKey)。要求锁定方提供数据以解锁数字资产的技术涉及将数据的散列嵌入锁定脚本中。但是,如果在创建锁定脚本时未确定(例如,未知和未固定)数据,则会出现问题。

[0007] 另外,如果基于区块链本身的各个方面(例如,区块中的其他交易或区块头的内容)预测锁定脚本,则在创建锁定脚本时该数据不存在,并且在区块链中不存在用来查询区块链的状态以获得该数据的操作码。因此,锁定脚本不能要求特定的区块头,不能要求区块链的特定状态,也不能要求交易位于区块链的特定区块内。

[0008] 因此,期望提供在一个或多个这些方面中改进区块链技术的方法和系统。因此,根据本发明,提供了所附权利要求中所限定的方法。

[0009] 下文更详细地描述了计算机实现的方法和电子设备,用于对解锁脚本中的数据实施约束,以要求解锁脚本包括区块头,区块链或区块头链。通过对解锁脚本中的数据实施此类约束,并在运行时将此类数据注入至解锁脚本中,交易可以基于区块链的各个方面。

[0010] 因此,根据本发明,可以提供一种所附权利要求书中所限定的计算机实现的方法(和相应的系统)。该方法可以描述为区块链数据约束方法。该计算机实现的方法包括:i)在区块链网络中的节点处接收包括第一脚本的第一交易,该第一脚本指定了对第二交易的约束集合,约束集合包括这样的约束:节点获得的数据集合包括从与区块链网络相关联的区块链获得的信息;ii)获得第二交易,所述第二交易包括第二脚本,由于执行该第二脚本而导致节点获得上述数据集合;iii)通过执行第一脚本和第二脚本来验证第二交易。

[0011] 下文更详细地描述了计算机实现的方法和电子设备,用于对解锁脚本中的数据实施约束,以要求在可以使用解锁脚本来解锁锁定脚本和访问交易的数字资产之前区块链处于特定状态。通过在锁定脚本上实施约束以要求解锁脚本包括区块头、区块链或区块头链,并在运行时将此类数据注入至解锁脚本中,交易可以基于区块链的状态。

[0012] 约束集合可包括这样的约束:上述数据集合包括区块链的区块的区块头。

[0013] 通过至少验证区块头具有预定大小,节点可以确定是否满足这样的约束:数据集合包括区块链的区块的区块头。附加地或替代地,节点可以通过至少验证区块头包含大于或等于某难度值的难度值来确定是否满足数据集合包括区块链的区块的区块头的约束。附加地或替代地,节点可以通过至少验证区块头的散列小于或等于在区块头中包含的难度值计算出的目标值来确定这样的约束是否得以满足:数据集合包括区块链的区块的区块头。

[0014] 约束集合可以包括这样的约束:数据集合包括来自区块链的区块的第三交易。

[0015] 该数据集合可包括区块链的区块的区块头。附加地或可替代地,约束集合可包括这样的约束:第三交易包括在区块中。附加地或替代地,节点可以至少部分地基于区块链的区块的区块头来确定是否满足所述第三交易包括在区块中的约束。

[0016] 节点可至少部分地基于由区块头识别的区块中的交易的编码,通过至少计算第三交易的散列值来确定是否满足第三交易包括在区块中的约束。附加地或替代地,节点可以通过至少验证第三交易的散列值等于存储在区块头中的散列值来确定是否满足第三交易包括在区块中的约束。

[0017] 约束集合可包括这样的约束:数据集合包括区块头链,该区块头链包括区块头的有序集合,区块头的有序集合包括多个区块头,区块头的有序集合指定了与所述多个区块头相关联的顺序。

[0018] 节点可通过至少部分地基于与多个区块头相关联的顺序至少选择一对区块头来确定这样的约束是否得到满足:第二脚本包括区块头链。该对区块头包括:一对区块头中的第一区块头和一对区块头中的第二区块头。附加地或替代地,节点可以通过至少针对该对区块头来验证该对区块头中的第一区块头的散列是否等于存储在该对区块头中第二区块头中的散列值,来确定第二脚本包括区块头链的约束是否得到满足。

[0019] 约束集合可包括这样的约束:数据集合从区块链网络的公共区块链获得。

[0020] 可以在执行第一脚本和第二脚本之前,将区块链网络的一个或多个属性提供给节点。

[0021] 可以在不验证创建了第二交易的实体有权访问秘密信息的情况下成功地执行验证第二交易。

[0022] 第一脚本可以是第一交易的锁定脚本,第二脚本是第一脚本的解锁脚本。

[0023] 该计算机实现的方法可进一步包括:至少部分地基于验证的结果来转移数字资产。因此,根据本发明,可以提供一种在所附权利要求中所定义的计算机实现的方法(和(一个或多个)相应的系统)。该方法可以描述为区块链状态确认方法。该计算机实现的方法包括:i)在区块链网络中的节点接收至少指定以下内容的第一交易:a)对第二交易的第一约束集合,该第一约束集合包括一个或多个约束使第二交易包含来自区块链的数据集合;b)对第二交易的第二约束集合,第二约束集合包括与该数据集合中的数据项相关的一个或多个约束;ii)在验证满足了第一约束合和第二约束集合的情况下,将第二交易存储到区块链中。

[0024] 下文更详细地描述了计算机实现的方法和电子设备,用于对解锁脚本中的数据实施约束,以要求在可以使用解锁脚本来解锁锁定脚本之前区块链处于特定状态,并在这些状态下保持对交易的数字资产的访问。通过对锁定脚本实施约束以要求解锁脚本包括区块头、区块链或区块头链,并通过在运行时将此类数据注入至解锁脚本中,交易结果可以基于区块链的状态。

[0025] 该数据集合可在第二交易中的节点处接收。

[0026] 该计算机实现的方法可进一步包括:验证第二交易以作为验证的结果。

[0027] 可以在不验证创建第二交易的实体有权访问秘密信息的情况下成功地执行验证第二交易。

[0028] 第一约束集合可包括这样的约束:数据集合包括区块链的区块头。

[0029] 第一约束集合可包括这样的约束:数据集合包括来自区块链的区块的第三交易。

[0030] 第一约束集合可包括这样的约束:数据集合包括区块头链,该区块头链包括区块头的有序集合,区块头的有序集合包括多个区块头,区块头的有序集合指定与多个区块头相关联的顺序。

[0031] 第二约束集合可包括对数据集合的数据项的值的约束。

[0032] 第二约束集合可包括从与数据集合的数据项相关联的一个或多个值得出的约束。

[0033] 第一约束集合可包括在第一交易的锁定脚本中。

[0034] 第二约束集合可包括在第一交易的锁定脚本中。

[0035] 第一约束集合可包括这样的约束:数据集合从公共区块链接收。

[0036] 节点可通过至少在第二交易中接收第一区块集合和第二区块集合来确定从公共

区块链接收数据集合的约束是否得以满足,所述第一区块集合包括了从区块链网络接收的区块之前的区块,所述第二区块集合包括了从区块链网络接收的区块之后的区块。附加地或可替代地,节点可通过至少验证第一区块集合链接至从区块链网络接收的区块来确定从公共区块链接收数据集合的约束是否得以满足。附加地或可替代地,节点可通过至少验证第二区块集合链接至从区块链网络接收的区块来确定从公共区块链接收数据集合的约束是否得以满足。附加地或替代地,节点可通过至少验证第一区块集合和第二区块集合是否有效来确定从公共区块链接收数据集合的约束是否得以满足。附加地或可替代地,节点可通过至少验证第一区块集合和第二区块集合中的每个区块具有比预定值更大的难度值来确定从公共区块链接收数据集合的约束是否得以满足。

[0037] 因此,根据本发明,可以提供一种在所附权利要求中所定义的计算机实现的方法(和(一个或多个)相应的系统)。该方法可以描述为区块链状态感知方法。该计算机实现的方法包括:i)在区块链网络中的节点处接收至少指定以下内容的第一交易:a)对第二交易的第一约束集合,该第一约束集合包括一个或多个约束使第二交易包含来自区块链网络的数据集合;b)对第二交易的第二约束集合,第二约束集合包括该数据集合包括包括第一交易的区块的约束,所述区块包括在与区块链网络相关联的区块链中;ii)成功验证满足第一约束集合和第二约束集合,使得将第二交易记录到区块链网络中。

[0038] 该数据集合可以在第二交易中的节点处接收。

[0039] 第一约束集合可以包括这样的约束:数据集合包括区块链的区块头。

[0040] 第一约束集合可包括这样的约束:数据集合包括区块链的区块的第三交易。

[0041] 第一约束集合可包括这样的约束:数据集合包括区块头链,该区块头链包括区块头的有序集合,区块头的有序集合包括多个区块头,区块头的有序集合指定与多个区块头相关联的顺序。

[0042] 验证满足第一约束集合和第二约束集合,从而可以将区块链网络的一组属性提供给节点。

[0043] 所述区块链网络的该组属性可包括与区块链网络的区块链的每个区块相关联的对应时间戳。

[0044] 第一约束集合包括这样的约束:第二交易包括与区块相关的时间戳。附加地或可替代地,对第二交易包括与区块相关的时间戳的约束进行验证可至少部分基于区块链网络的该组属性。

[0045] 第一约束集合包括这样的约束:数据集合包括区块头。附加地或可替代地,第二约束集合可包括这样的约束:第二交易包括第一交易的标识符。附加地或可替代地,第二约束集合可包括这样的约束:第一交易的标识符与区块头的值相关联。

[0046] 第一约束集合和第二约束集合可包括在交易的锁定脚本中。

[0047] 第一交易可以在第二交易上指定第三约束集合,第三约束集合包括对数据集合的数据项的值的约束。

[0048] 第一交易可以在第二交易上指定第三约束集合,第三约束集合包括从与该数据集合的数据项相关联的一个或多个值得出的约束。

[0049] 该计算机实现的方法可进一步包括:由于验证了满足第一约束集合和第二约束集合而验证第二交易,因此成功验证了第二交易,而未验证创建了第二交易的实体有权访问

秘密信息。因此,根据本发明,可以提供一种在所附权利要求中所定义的计算机实现的方法(和(一个或多个)相应的系统)。该方法可以描述为区块链数据约束方法。该计算机实现的方法包括:i)在区块链网络中的节点处接收与数字资产相关联的第一交易,第一交易包括第一脚本,第一脚本对第二交易规定了约束集合,所述第二交易用来转移对数字资产的控制,约束集合包括的约束为节点获取的数据集合包括从与区块链网络相关联的区块链获得的信息;ii)获得第二交易,所述第二交易包括第二脚本,由于执行该第二脚本而使得节点获得数据集合;iii)通过执行第一脚本和第二脚本来验证第二交易。

[0050] 约束集合可包括这样的约束:数据集合包括区块链的区块的区块头。

[0051] 通过至少验证区块头具有预定大小,节点可以确定是否满足这样的约束:数据集合包括区块链的区块的区块头。附加地或替代地,节点可以通过至少验证区块头包含大于或等于某难度值的难度值来确定是否满足这样的约束:数据集合包括区块链的区块的区块头。附加地或替代地,节点可以通过至少验证区块头的散列小于或等于从区块头中包含的难度值计算出的目标值来确定是否满足数据集合包括区块链的区块的区块头的约束。

[0052] 约束集合可以包括这样的约束:数据集合包括来自区块链的区块的第三交易。

[0053] 该数据集可包括区块链的区块的区块头。附加地或可替代地,约束集合可包括这样的约束:第三交易包括在区块中。附加地或替代地,节点可以至少部分地基于区块链的区块的区块头来确定是否满足第三交易包括在区块中的约束。

[0054] 节点可至少部分地基于由区块头识别的区块中的交易的编码,通过至少计算第三交易的散列值来确定是否满足第三交易包括在区块中的约束。附加地或替代地,节点可以通过至少验证第三交易的散列值等于存储在区块头中的散列值来确定是否满足第三交易包括在块中的约束。

[0055] 约束集合可包括这样的约束:数据集合包括区块头链,该区块头链包括区块头的有序集合,区块头的有序集合包括多个区块头,区块头的有序集合指定了与所述多个区块头相关联的顺序。

[0056] 节点可通过至少部分地基于与多个区块头相关联的顺序至少选择一对区块头来确定第二脚本包括区块头链的约束是否得到满足,即区块头对,该区块头对包括:该区块头对中的第一区块头和该区块头对中的第二区块头。附加地或替代地,节点可以至少通过针对该区块头对来验证该区块头对中的第一区块头的散列是否等于存储在该区块头对中第二区块头中的散列值,来确定第二脚本包括区块头链的约束是否得到满足。

[0057] 约束集合可包括这样的约束:数据集合从区块链网络的公共区块链获得。

[0058] 可以在执行第一脚本和第二脚本之前,将区块链网络的一个或多个属性提供给节点。

[0059] 可以在不验证创建了第二交易的实体有权访问秘密信息的情况下成功地执行验证第二交易。

[0060] 第一脚本可以是第一交易的锁定脚本,第二脚本是第一脚本的解锁脚本。

[0061] 该计算机实现的方法可进一步包括:至少部分地基于验证的结果来转移数字资产。

[0062] 因此,根据本发明,可以提供一种在所附权利要求中所定义的计算机实现的方法(和(一个或多个)相应的系统)。该方法可以描述为区块链状态确认方法。该计算机实现的

方法包括:i)在区块链网络中的节点处接收与数字资产相关联的第一交易,第一交易至少指定:a)对第二交易的第一约束集合,所述第一约束集合用来转移对数字资产的控制,该第一约束集合包括一个或多个约束使第二交易包含来自区块链的数据集合;b)对第二交易的第二约束集合,第二约束集合包括与该数据集合中的数据项相关的一个或多个约束;ii)验证满足第一约束集合和第二约束集合;iii)至少部分基于验证来重新关联数字资产。

[0063] 该数据集合可以在第二交易中的节点处接收。

[0064] 该计算机实现的方法可以进一步包括:验证第二交易以作为验证的结果。

[0065] 成功地执行验证第二交易而未验证创建第二交易的实体有权访问秘密信息。

[0066] 第一约束集合可包括这样的约束:数据集合包括区块链的区块头。

[0067] 第一约束集合可包括这样的约束:数据集合包括来自区块链的区块的第三交易。

[0068] 第一约束集合可包括这样的约束:数据集合包括区块头链,该区块头链包括区块头的有序集合,区块头的有序集合包括多个区块头,区块头的有序集合指定与多个区块头相关联的顺序。

[0069] 第二约束集合可包括对数据集合的数据项的值的约束。

[0070] 第二约束集合可包括从与数据集合的数据项相关联的一个或多个值得出的约束。

[0071] 第一约束集合可包括在第一交易的锁定脚本中。

[0072] 第二约束集合可包括在第一交易的锁定脚本中。

[0073] 第一约束集合可包括这样的约束:数据集合从公共区块链接收。

[0074] 节点可通过至少在第二交易中接收第一区块集合和第二区块集合来确定从公共区块链接收数据集合的约束是否得以满足,所述第一区块集合包括了从区块链网络接收的区块之前的区块,所述第二区块集合包括了从区块链网络接收的区块之后的区块。附加地或可替代地,节点可通过至少验证第一区块集合链接至从区块链网络接收的区块来确定从公共区块链接收数据集合的约束是否得以满足。附加地或可替代地,节点可通过至少验证第二区块集合链接至从区块链网络接收的区块来确定从公共区块链接收数据集合的约束是否得以满足。附加地或替代地,节点可通过至少验证第一区块集合和第二区块集合是否有效来确定从公共区块链接收数据集合的约束是否得以满足。附加地或可替代地,节点可通过至少验证第一区块集合和第二区块集合中的每个区块具有比预定值更大的难度值来确定从公共区块链接收数据集合的约束是否得以满足。

[0075] 因此,根据本发明,可以提供一种在所附权利要求中所定义的计算机实现的方法(和(一个或多个)相应的系统)。该方法可以描述为区块链状态感知方法。该计算机实现的方法包括:i)在区块链网络中的节点处接收与数字资产相关联的第一交易,所述第一交易至少指定以下内容:a)对用于转移对数字资产的控制的第二交易的第一约束集合,该第一约束集合包括一个或多个约束使第二交易包含来自区块链网络的数据集合;b)对第二交易的第二约束集合,第二约束集合包括该数据集合包括包括第一交易的区块的约束,所述区块包括在与区块链网络相关联的区块链中;ii)验证满足第一约束集合和第二约束集合;以及iii)至少部分基于验证来转移对数字资产的控制。

[0076] 该数据集合可以在第二交易中的节点处接收。

[0077] 第一约束集合可以包括这样的约束:数据集合包括区块链的区块头。

[0078] 第一约束集合可以包括这样的约束:数据集合包括来自区块链的区块的第三交

易。

[0079] 第一约束集合可包括这样的约束:数据集合包括区块头链,该区块头链包括区块头的有序集合,区块头的有序集合包括多个区块头,区块头的有序集合指定与多个区块头相关联的顺序。

[0080] 验证满足第一约束集合和第二约束集合,从而可以将区块链网络的一组属性提供给节点。

[0081] 所述区块链网络的该组属性可包括与区块链网络的区块链的每个区块相关联的对应时间戳。

[0082] 第一约束集合包括这样的约束:第二交易包括与区块相关的时间戳。附加地或可替代地,对第二交易包括与区块相关的时间戳的约束进行验证可至少部分基于区块链网络的一组属性。

[0083] 第一约束集合包括这样的约束:数据集合包括区块头。附加地或可替代地,第二约束集合可包括这样的约束:第二交易包括第一交易的标识符。附加地或可替代地,第二约束集合可包括这样的约束:第一交易的标识符与区块头的值相关联。

[0084] 第一约束集合和第二约束集合可包括在交易的锁定脚本中。

[0085] 第一交易可以在第二交易上指定第三约束集合,第三约束集合包括对数据集合的数据项的值的约束。

[0086] 第一交易可以在第二交易上指定第三约束集合,第三约束集合包括从与该数据集合的数据项相关联的一个或多个值得出的约束。

[0087] 该计算机实现的方法可进一步包括:由于验证了满足第一约束集合和第二约束集合而验证第二交易,其中成功验证了第二交易,而未验证创建了第二交易的实体有权访问秘密信息。

[0088] 还期望提供一种系统,包括:处理器;和包含可执行指令的存储器,所述处理器执行所述指令使得所述系统执行权利要求的方法中的任意一项。

[0089] 还期望提供一种存储了可执行指令的非暂时性计算机可读存储介质,计算机系统的处理器执行所述可执行指令,使得所述计算机系统至少执行权利要求的方法中的任意一项。

[0090] 还期望提供一种系统,包括:处理器;和包含可执行指令的存储器,所述处理器执行所述指令使得所述系统执行权利要求的方法中的任意一项。

[0091] 根据本发明,可提供一种电子设备。该电子设备包括接口设备,耦合到该接口设备的处理器以及耦合到该处理器的存储器。存储器上存储了计算机可执行指令,该计算机可执行指令在被执行时使处理器用于执行本文所述的方法。

[0092] 根据本发明,可提供一种计算机可读存储介质。该计算机可读存储介质包括计算机可执行指令,该计算机可执行指令在被执行时使处理器用于执行本文所述的方法。

附图说明

[0093] 参考本文描述的实施例,本发明的这些和其他方面将变得显而易见或得以阐释。现在将仅通过举例的方式并参考附图来描述本发明的实施例。

[0094] 图1示出了根据实施例的示例电子设备的示意图,其中所述示例电子设备可以用

作区块链网络中的节点；

[0095] 图2示出了与基于脚本的区块链交互相关联的交易的示例实施例的示意图；

[0096] 图3示出了由实施例解决的示例问题的示意图，其中所述示例问题与访问与基于脚本的区块链交互相关联的区块链有关；

[0097] 图4示出了由实施例解决的示例问题的示意图，其中所述示例问题与访问与基于脚本的区块链交互相关联的区块链中的区块有关；

[0098] 图5示出了根据实施例的示例环境的示意图，其中数据被验证为区块头，所述区块头用于基于脚本的区块链交互；

[0099] 图6示出了根据实施例的示例过程的流程图，所述示例过程用于验证数据是基于脚本的区块链交互中的区块头；

[0100] 图7示出了根据实施例的示例环境的示意图，其中数据验证为用于基于脚本的区块链交互的先前区块的区块头；

[0101] 图8示出了根据实施例的示例过程的流程图，所述示例过程用于验证数据是基于脚本的区块链交互中的先前区块的区块头；

[0102] 图9示出了根据实施例的示例环境的示意图，其中数据验证为用于基于脚本的区块链交互的区块头链；

[0103] 图10示出了根据实施例的示例过程的流程图，所述示例过程用于验证数据是基于脚本的区块链交互中的区块头链；

[0104] 图11示出了根据实施例的示例环境的示意图，该示例环境示出了可在基于脚本的区块链交互中使用的区块头的默克尔(Merkle)树；

[0105] 图12示出了根据实施例的示例环境的示意图，其中交易验证为包括在用于基于脚本的区块链交互的区块头中；

[0106] 图13示出了根据实施例的示例过程的流程图，所述示例过程用于验证交易包括在基于脚本的区块链交互的区块头中；

[0107] 图14示出了根据实施例的示例环境的示意图，该示例环境示出了如何使用本文所述的方法来解决访问与基于脚本的区块链交互相关联的区块链的问题。

[0108] 图15示出了根据实施例的示例环境的示意图，该示例环境示出了与基于脚本的区块链交互相关联的区块链的状态；

[0109] 图16示出了根据实施例的示例环境的示意图，在所述环境中，在基于脚本的区块链交互中确认了区块链的状态；

[0110] 图17示出了根据实施例的示例过程的流程图，所述示例过程用于在基于脚本的区块链交互中确认区块链的状态；

[0111] 图18示出了由实施例解决的在基于脚本的区块链交互中锁定脚本的示例问题的示意图；

[0112] 图19示出了根据实施例的在基于脚本的区块链交互中锁定脚本对数据访问的示例的示意图；

[0113] 图20示出了根据实施例的示例环境的示意图，在所述环境中，在基于脚本的区块链交互中，从串行化交易字段集合中生成签名；

[0114] 图21示出了根据实施例的示例过程的流程图，所述示例过程用于在基于脚本的区

块链交互中引起串行化解锁交易字段集合的注入；

[0115] 图22示出了由实施例解决的在基于脚本的区块链交互中锁定脚本的示例问题的示意图；

[0116] 图23示出了根据实施例的示例环境的示意图，在所述环境中，在基于脚本的区块链交互中引起先前的串行化交易的注入；

[0117] 图24示出了根据实施例的字段集合的示例的示意图，在基于脚本的区块链交互中，所述字段集合基于签名散列类型而可用；

[0118] 图25示出了根据实施例的在基于脚本的区块链交互中从串行化交易中提取交易标识符的示例的示意图；

[0119] 图26示出了根据实施例的示例过程的流程图，在基于脚本的区块链交互中，所述示例过程用于引起先前串行化交易的注入；

[0120] 图27示出了根据实施例的示例环境的示意图，在所述环境中，区块链的状态用于基于脚本的区块链交互；

[0121] 图28示出了根据实施例的示例过程的流程图，所述示例过程为在基于脚本的区块链交互中使用区块链的状态的示例过程；

[0122] 图29示出了根据实施例的在基于脚本的区块链交互中使用区块链的状态的示例实施方式的示意图。

具体实施方式

[0123] 运行区块链协议并且形成区块链网络的节点的电子设备可以是各种类型，包括例如计算机(如台式计算机、膝上型计算机、平板计算机、服务器)、移动设备(如智能手机)、可穿戴设备(如智能手表)，或其他电子设备。

[0124] 区块链网络的节点使用适当的通信技术彼此耦合，所述通信技术可包括有线和无线通信技术。这种通信遵循与区块链相关的协议。节点维护区块链上所有交易的全局账本。因此，全局账本是分布式账本。每个节点可以存储全局账本的完整副本或部分副本。节点影响全局账本的交易由其他节点验证，从而全局账本的有效性得以维护。

[0125] 区块链中的区块充当从一个地址到另一个地址的交易记录。要求将交易包含在区块中的一方通过使用与他们的公钥相对应的私钥对请求进行签名，来证明他们被授权而发起转移。如果请求已有效签名，则仅将转移添加到区块中。

[0126] 公钥和地址之间存在一一对应的关系。也就是说，每个公钥都与唯一一个地址相关联。因此，本文中对向公钥或自公钥转移数字资产(例如，转移至与公钥相关联的地址)以及向或自与公钥相关联的地址转移数字资产的任何引用均指通用操作。

[0127] 某些节点可能会充当验证节点。交易的验证可以涉及检查签名或在锁定脚本中指定的其他条件，确认对有效UTXO的引用等。

[0128] 图1以示意图的形式示出了示例电子设备200，该电子设备200可用作根据实施例的区块链网络中的节点。示例电子设备200可用作节点。在一个实施例中，区块链网络是点对点区块链网络。

[0129] 电子设备可以为各种形式，包括例如台式计算机、膝上型计算机、平板计算机、服务器、移动设备(如智能手机)、可穿戴设备(如智能手表)或其它形式。

[0130] 电子设备200包括处理器210、存储器220和接口设备230。这些组件可以直接或间接耦合到彼此,并且可以彼此通信。例如,处理器210、存储器220和接口设备230可以通过总线240彼此通信。存储器220存储计算机软件程序,该计算机软件程序包括机器可读指令和用于执行本文描述的功能的数据。例如,存储器可包括处理器可执行指令,该处理器可执行指令在由处理器210执行时使电子设备执行本文所述的方法。处理器可执行指令可包括在由处理器210执行时使电子设备实现与区块链网络相关联的协议的指令。

[0131] 存储器220可存储区块链网络的全局账本或其一部分。即,存储器220可存储区块链的所有块或块的一部分,例如最新的块,或一些块中的信息的一部分。

[0132] 虽然存储器220在图1中以单个框示出,实际上,电子设备200可包括多个存储器组件。所述存储器组件可以为各种类型,包括例如随机存取存储器(RAM)、硬盘驱动器(HDD)、固态驱动器(SSD)、闪存驱动器(Flash Drives)等。不同类型的存储器可适用于不同的目的。此外,尽管将存储器220与处理器210分开示出,但是处理器210可包括嵌入式存储器。

[0133] 如图1所示,处理器210可包括安全区域,例如可信执行环境(TEE)。TEE 250是隔离的执行环境,其向电子设备200提供附加的安全性,例如隔离的执行,可信应用的完整性和资产机密性。TEE 250提供了执行空间,该执行空间保证了在TEE 250内部加载的计算机指令和数据在机密性和完整性方面得到保护。TEE 250可用于保护重要资源(例如加密密钥)的完整性和机密性。TEE 250至少部分地在硬件级别上实现,从而保护TEE 250内执行的指令和数据免受来自电子设备200的其余部分以及电子设备所有者等外部各方的访问和操纵。TEE 250内的数据和计算受到保护,不被操作包括TEE 250的节点的一方访问和操纵。

[0134] TEE 250可操作以实例化安全执行环境(在本文中也称为“隔离区(enclave)”),然后一次添加一个存储器的页面,同时累积地散列所添加的存储器的页面。在一个实施例中,在远程机器(例如,开发者机器或另一台机器)上执行存储器页面的散列,使得远程机器确定并存储所期望的散列。因此,隔离区的内容可以由任何远程机器进行验证,以确保隔离区正在运行批准的算法。可以通过比较散列来执行该验证。隔离区完全建成后将被锁定。可以在TEE 250中运行代码并将秘密发送给代码,但是一旦隔离区被锁定,就无法更改代码。最终的散列可以由认证密钥签名,并且可以在数据所有者将任何秘密发送到隔离区之前供数据所有者使用以进行验证。

[0135] TEE 250可用于保护存储在TEE 250内的加密密钥的机密性和完整性。例如,TEE 250可用于生成和存储私钥份额。TEE 250的目的是确保没有成员能够直接获得TEE 250隔离区内的私钥份额,也不能够从成员间通信或隔离区域间通信中获得有关其他私钥份额的信息。该协议在折中隔离区阈值方面也很强大。此外,TEE 250可使得能够由节点使用的远程认证向其他节点证明TEE 250是真实的并且正在运行已批准的由区块链网络实现的协议的计算机可执行指令。远程认证可以由TEE250通过运行特定代码段并发送隔离区内部的由隔离区的内部认证密钥签名的节点的散列来提供。

[0136] TEE 250可以配备有安全的随机数发生器,该发生器位于TEE的隔离区内部,可以用来生成私钥、随机质询或其他随机数据。TEE 250还可用于从外部存储器读取数据,并且可用于将数据写入外部存储器。可以使用仅在隔离区内保存的秘密密钥来加密此类数据。

[0137] 可以使用各种平台来实现TEE 250,例如可信平台模块(TPM)或英特尔软件保护扩展(SGX)。例如,SGX支持远程认证,使隔离区能够从执行特定隔离区的处理器中获取签名语

句,该特定隔离区具有被称为引用(quote)的给定成员。第三方认证服务,例如英特尔认证服务(IAS),可以证明这些签名语句源自符合SGX规范的真实CPU。

[0138] 本发明可提供这样的方法(和相应的系统):该方法用于使用在另一交易的解锁脚本中提供的未确定的数据,来修改嵌入在区块链交易(Tx)的锁定脚本中的加密公钥。当与例如使用串行化的交易作为消息的签名检查操作码结合使用时,交易和数据都需要获得公钥所有者的批准或授权。这样可以保护交易和数据免受更改。

[0139] 本文所述的方法使用一个或多个数字签名方案来验证各种交易。该数字签名方案可以是椭圆曲线数字签名算法(ECDSA)方案。数字签名方案也可以是两方ECDSA方案。数字签名方案也可以是阈值ECDSA方案。该方案可用于构造有效签名,而不必重构私钥并且任何一方都不必向另一方透露其密钥份额。在两方ECDSA方案的示例中有两方,并且双方都需要重建私钥。

[0140] 该ECDSA方案包括可以由节点使用的各种机制来识别恶意或不合作的一方。例如,可验证的秘密共享(VSS)可用于共享沙米尔的秘密共享(SSS)所需的多项式。SSS是秘密共享的一种形式,其中将秘密分为多个部分,提供给每个参与者,每个参与者获得其自己独有的一部分。这些部分可用于重建秘密。如果向不同的节点提供了不一致的份额,或者如果将份额秘密发送到与广播给所有节点的盲份额不同的节点,则节点可以使用VSS来识别恶意节点或成员。任一节点可以识别出不一致的份额。可以通过包括辅助信息来使秘密的共享可验证,该辅助信息允许节点将其份额验证为一致。

[0141] 可以通过份额的预期接收者节点来识别向单个节点发送不正确的份额(即,与广播的盲份额不同的份额)。可以使用可公开验证的秘密共享(PVSS)技术公开验证对秘密发送到节点的不正确份额的识别。这样的技术可以避免在不使用PVSS,并且不正确的份额的接收者离线,或在发送不正确的份额时接收者从网络的大部分断开时,对欺骗发送者进行识别中可能发生的延迟。

[0142] 不当行为,例如向不同节点提供不一致的份额,可以由网络解决,以阻止恶意行为。例如,当一个节点被其他节点标识为恶意方时,许多节点可以合作以惩罚该恶意方。例如,节点可以对涉及由恶意方存放到区块链网络的数字资产(例如股份或价值)采取行动。例如,区块链网络可以通过将股份或价值转移到不可赎回的地址来毁掉股份或价值,或者区块链网络可以通过与其他拒绝节点达成共识以没收这种数字资产。不是异常行为节点的节点还可通过协作排除异常行为节点来阻止异常行为(例如,通过有效地使密钥份额无效;例如,通过将节点排除在参与大会协议之外,或者通过重新共享私钥,而不为行为异常的节点分配份额)。

[0143] 可以通过使用TEE来增强上述ECDSA技术。例如,基于Ibrahim等人的阈值ECDSA签名技术构想了一种强大的对手形式,在此称为拜占庭式对手。这种类型的对手可能会表现随意,例如,他们不仅拒绝参与签名过程或中途停止聚会,而且还可能假装诚实参与并发送不正确的信息。但是,通过使用TEE,并在存储密钥份额的TEE的隔离区内生成用于签名的数据,可以提供额外的安全性,因为隔离区极不可能被大量破坏。例如,如果为每个TEE分配的密钥份额不超过一个,假设n足够大,则可以合理预期可能受损的TEE的数量不会接近针对拜占庭式对手的强大阈值。如果相对于密钥份额总数而言,它只容忍一小部分恶意对手,则可以使该协议安全。

[0144] 例如,如果所有节点都具有TEE,则只有在不损害TEE的制造商的情况下,只有通过对节点的物理访问,并且只有付出很大的努力和费用,才能达到对存储在隔离区域中的秘密的获取。这种制造商级别的损害预计是可控的。例如,如果制造商错误地声称许多公钥对应于真正的TEE,则他们可以获得直接访问私钥份额的权利并可能发动攻击。但是,这样的攻击将需要足够数量的密钥份额,以使得制造商能够在没有其他节点帮助的情况下产生有效的签名。这将意味着累积总股份的很大一部分,这将是非常昂贵的。而且,通过进行攻击,很大一部分所持股份的价值将被破坏。

[0145] 使用TEE时,考虑协议对“损坏的节点”的强大性很有用。损坏的节点是这样的节点,即TEE外部的硬件已损坏,但TEE的完整性没有受到损害。损坏的节点可以控制了隔离区接收和不接收的信息。特别是,损坏的节点可以暂停,即避免参与协议。如果提供给协议的信息要求由秘密存放在隔离区(在认证过程中对相应的公钥进行了证实)中的私钥签署,则私钥与隔离区本身一样值得信赖。因此,损坏的节点无法向协议发送任意(证实的)信息,只能通过中止或尝试使隔离区误动作(例如通过向其提供过时的信息)来尝试进行干扰。因此,对于损坏的节点,成功的攻击将需要收集足够数量的部分签名以产生完整的签名。

[0146] 在一些实施例中,可以使用其他阈值方案,包括非ECDSA签名方案。

[0147] 图2以示意图形式示出了与基于脚本的区块链交互相关联的交易的示例实施例300。图2所示的示例实施例300描述了:对区块链的第一状态306进行编码的先前区块链交易302和对区块链的第二状态308进行编码的解锁区块链交易304。在示例实施例300中,处于第一时间的当前状态306表示为嵌入在先前交易302中的参数。在一个实施例中,解锁交易304包括表示下一状态308的参数。

[0148] 在示例实施例300中,交易302和304是字段值集合,其包含一个或多个输入和一个或多个输出。在一些实施例中,输入和输出反映了将数字资产的控制从至少一个实体转移到至少另一实体的意图。在示例实施例300中,先前交易302是包括在区块链中的最新的,确认过的交易。在示例实施例300中,解锁交易304是近期将要发生的交易,其尚未被确认并且尚未被包括在区块链中。解锁交易304的每个输入接收先前交易302的输出。

[0149] 一些区块链技术,使用椭圆曲线数字签名算法(ECDSA)作为数字签名的数学方案。在实施例中,ECDSA是用于确保UTXO仅被转移给其合法所有者的加密数字签名。椭圆曲线数字签名(ECDS)通常会在其末尾附加签名散列标记(SIGHASH类型)。然而,可以想到,本发明的技术可与不实现SIGHASH类型的区块链技术一起使用。在这样的区块链技术中,ECDS被构想为符合特定区块链技术的签名生成原理。

[0150] 在实施例中,SIGHASH类型是指在被串行化(例如规范化)和散列化之前从交易中提取的字段集合。例如,SIGHASH类型可以影响交易的哪些字段包含在签名中。在某些示例中,SIGHASH类型可以是SIGHASH_ALL,SIGHASH_NONE,SIGHASH_SINGLE或SIGHASH_ANYONECANPAY中的一种或多种。在一个实施例中,类型SIGHASH_ALL表征除了输入脚本之外,交易中的所有字段都将被散列和签名。在一个实施例中,类型SIGHASH_NONE表征不需要对输出进行签名,这可以允许其他人更新交易。在一个实施例中,类型SIGHASH_SINGLE表征输入被签名但是序列号被空白,因此其他人可以创建交易的新版本,但是唯一被签名的输出是与输入在相同位置的输出。在一个实施例中,类型SIGHASH_ANYONECANPAY与其他类型组合,并表征包括SIGHASH_ANYONECANPAY的输入是已经签名的,而其他输入则不需要签署。

SIGHASH类型可以由表征SIGHASH类型的值表示。例如,在一些实施中,SIGHASH_ALL由值为1的字节表示,SIGHASH_NONE由值为2的字节表示(例如,“00000010”),SIGHASH_SINGLE由值为3的字节表示(例如,“00000011”),SIGHASH_ANYONECANPAY由值为80的字节表示(例如“01010000”)。在一些实施中,通过将字节值相加在一起来组合SIGHASH类型。

[0151] 在一些示例中,由SIGHASH类型确定的交易字段集合指的是由SIGHASH类型确定的以字节编码的相应交易的子集。例如,在SIGHASH类型为SIGHASH_ANYONECANPAY的情况下,签名中仅包含一个交易输入。

[0152] 在实施例中,ECDS由以区别编码规则(DER)格式编码的一对256位数字(r,s)表示。然而,请注意,本发明的技术可与其他编码格式一起使用,例如基本编码规则(BER)或规范编码规则(CER)。ECDSA中使用的参数包括K(坐标为(x,y)的椭圆曲线点),k(256位数字;通常是随机的以保护私钥),G(顺序为n的椭圆曲线上的基点:$n\times G=0$,其中0表示同一性椭圆曲线点),n是用作椭圆曲线有限域中的参数的大质数,r(ECDS中的256位的数字之一),s(签名中的256位数字中的另一个,k^{-1}(k的模乘逆数,即$k^{-1}k\equiv 1 \bmod n$),m(正在签名的消息/数据;在实施例中,其由散列函数调整为256位),a(私钥;例如256位的数字)。在该实施例中,以如下方式生成ECDS:

[0153] 首先,通过将k与生成器$K=k\times G$相乘来确定椭圆曲线点K。然后,根据K确定点x,然后根据公式$r=x \bmod n$确定ECDS中的256位的数字中的第一个:r。然后,根据公式$s=k^{-1}(m+r\times a) \bmod n$确定ECDS中256位的数字中的第二个:s。最后,(r,s)以DER格式编码。在给定签名(r,s),消息/数据m和对应于私钥a的公钥A的情况下,该签名是可验证的。为了验证签名,计算$v=s^{-1}\times(m\times G+r\times y)$。如果$v=r$,则签名有效。

[0154] 在本发明中,引用了各种脚本操作码和关键字来执行各种操作。然而,可以想到,其他区块链技术可以实现不同的指令集,因此,本发明中描述的操作码被认为是由操作码而非脚本中的特定操作码执行的操作的说明。在一些实施例中,脚本系统是图灵不完整指令集(例如,不支持循环、递归、“goto”语句等)。在其他实施例中,脚本系统是图灵完整指令集。

[0155] 本发明的某些实施例在以下假设下操作:用于实现所描述的指令集的脚本系统或其他系统在单个脚本中允许超过200条指令(例如,操作码)。同样,本发明的某些实施例还假定由本发明所引用的操作码提供的功能在执行操作码脚本/指令集的系统中存在并启用。默认情况下可以禁用或限制这些操作码中的一个或多个。

[0156] 本发明中引用的操作码的示例包括:

[0157] • OP_ECPX:返回椭圆曲线点的x坐标

[0158] • OP_ADD:将堆栈中的前两个项目相加

[0159] • OP_BIGMOD:将堆栈中的前两个项目相除后返回余数

[0160] • OP_BIGMODADD:对堆栈的前两项进行模数相加之后再将模数相加的结果对第三项取模

[0161] • OP_BIGMODINVERSE:执行模数负指数运算

[0162] • OP_BIGMODMUL:对堆栈的前两项进行模数相乘之后再将模数相乘的结果对第三项取模

[0163] • OP_CAT:将堆栈中的前两个项目连接在一起

[0164] • OP_CHECKSIG:根据SIGHASH类型,从堆栈中弹出公钥和签名,并对交易字段的签名进行验证。如果签名有效,则返回1,否则返回0。

[0165] • OP_CHECKSIGVERIFY:其功能与OP_CHECKSIG相同,但之后执行OP_VERIFY

[0166] • OP_DERENCODE:以DER格式编码堆栈中的前两个项目

[0167] • OP_DUP:复制最上面的堆栈项

[0168] • OP_ECPMULT:对堆栈上的前两个项目执行椭圆曲线相乘

[0169] • OP_ELSE:如果未执行前面的OP_IF或OP_NOTIF或OP_ELSE,则将执行这些语句;否则,如果执行了前面的OP_IF或OP_NOTIF或OP_ELSE,则不执行这些语句

[0170] • OP_ENDIF:结束if/else块

[0171] • OP_EQUAL:如果输入完全相等,则返回1,否则返回0

[0172] • OP_EQUALVERIFY:与OP_EQUAL相同,但之后运行OP_VERIFY

[0173] • OP_FROMALTSTACK:将输入放置在主堆栈的顶部,并将其从备用堆栈中删除

[0174] • OP_HASH256:输入被散列两次:首先使用SHA-256,然后使用RIPEMD-160

[0175] • OP_IF:如果顶层堆栈的值不为False,则执行该语句并删除顶层堆栈的值

[0176] • OP_NOTIF:如果最高堆栈值为错误(False),则执行该语句并删除最高堆栈值

[0177] • OP_ROLL:将堆栈中n个项目深的项目移至顶部

[0178] • OP_SUBSTR:返回字符串的一部分

[0179] • OP_SWAP:交换堆栈中的前两个项目

[0180] • OP_TOALTSTACK:将输入放置在备用堆栈的顶部,并将其从主堆栈中删除

[0181] • OP_VERIFY:如果顶部堆栈值不为真,则将交易标记为无效

[0182] 在实施例中,可以使用操作码来创建ECDS生成脚本,该操作码可以通过将SIGHASH字节附加到末尾来扩展以创建签名生成脚本OP_GENSIG。

[0183] 表1

[0184]

类型	脚本
解锁脚本	<SIGHASH Byte> <m> <a> <k>
锁定脚本	*// Compute K = k x G* OP_DUP OP_TOALTSTACK <PubK G> OP_ECPMULT *// Compute r = x mod n* OP_ ECPX <N> OP_BIGMOD OP_DUP OP_TOALTSTACK *// Compute $k^{-1}(r \times d + m) \bmod n$* <N> OP_BIGMODMUL <N> OP_BIGMODADD OP_FROMALTSTACK OP_SWAP OP_FROMALTSTACK <N> OP_BIGMODINVERSE <N> OP_BIGMODMUL *// DER encode (r,s) and append SIGHASH Type* OP_DERENCODE OP_SWAP OP_CAT

[0185] 在表1所示的上述脚本中,<SIGHASH Byte>,消息<m>,私钥<a>和数字<k>依次(例如,在插入所述四项之后,数字<k>在堆栈的顶部,然后是私钥<a>,然后是消息<m>,最后是<

SIGHASH Byte>)输入到主(例如,后进先出)堆栈中。

[0186] 脚本操作“OP_DUP OP_TOALTSTACK<PubK G>OP_ECPMULT”的执行会使得将数字<k>复制到备用堆栈,其中<k>与椭圆曲线生成器<PubK G>相乘,以在主堆栈的顶部生成椭圆曲线点K。

[0187] 脚本操作“OP_ECPX<N>OP_BIGMOD OP_DUP OP_TOALTSTACK”的执行使r根据K mod n的x坐标进行计算。r的副本压入备用堆栈。

[0188] 脚本操作“<N>OP_BIGMODMUL<N>OP_BIGMODADD”,“OP_FROMALTSTACK OP_SWAP OP_FROMALTSTACK<N>”和“OP_BIGMODINVERSE<N>OP_BIGMODMUL”的执行通过$k^{-1}(m+r\times a) \bmod n$计算出s。最后,脚本操作“OP_DERENCODE OP_SWAP OP_CAT”的执行使得r和s以DER格式编码,并与<SIGHASH Byte>连接。

[0189] 在本发明中,该脚本称为OP_GENSIG。因此,在本发明的实施例中对OP_GENSIG的引用应视为上述脚本中执行的操作的简写。

[0190] 图3以示意图形式示出了由实施例解决的示例问题400,其中所述示例问题400与访问与基于脚本的区块链交互相关联的区块链有关。在图3中所示的示例问题400中,即交易402中的锁定脚本406(例如,结合图2描述的先前交易(例如先前交易302)或解锁交易304)不能访问区块链404。

[0191] 如所指出的,在一些实施例中,交易402是包括在区块链中的最新的,确认的交易。类似地,在一些实施例中,交易402是未来交易,其尚未被确认并且尚未包括在区块链中,并且表示对转移由先前交易控制的数字资产的至少一部分的控制的尝试。

[0192] 在一些实施例中,锁定脚本406是这样的脚本:通过指定为了传输输出而需要满足的条件来阻碍交易。更具体地,执行锁定脚本406用于:作为由区块链系统的验证节点执行的结果,从执行的解锁脚本接受数据,并基于该数据执行某些操作,并返回结果,该结果表征解锁脚本的执行是否成功“解锁”(即,满足其中设置的条件集合)锁定脚本。在一些实施例中,锁定脚本406定义了一个或多个数据约束,所述约束必须被满足(例如,通过经由解锁脚本提供的数据),以便成功进行交易的验证。例如,锁定脚本406可要求在解锁脚本中提供某些数据以解锁交易402的关联数字资产。

[0193] 如示例问题400中所示,交易402的锁定脚本406无法查询区块链404以例如确定块的数量。交易402的锁定脚本406也无法查询包括区块链404的第一区块408(也称为“初始区块”)的区块链404的区块、区块链404的内部区块(例如,区块410)中的任意一个、或区块链404的最新区块412。本文描述的方法使交易402的锁定脚本406能够查询包括第一区块408的区块链404的区块,任何内部区块(例如,区块410)、或区块链404的最新区块412。

[0194] 图4以示意图形式示出了由实施例解决的示例问题500,其中所述示例问题500与访问与基于脚本的区块链交互相关联的区块链中的区块有关。在图4中所示的示例性问题500中,即交易502(例如,先前的交易,例如结合图2描述的先前交易302;或解锁交易,例如结合图2描述的解锁交易304)中的锁定脚本506不能访问区块链中的区块,例如区块504(即,第一块、内部块、或最新块)。

[0195] 如所指出的,在一些实施例中,交易502是包括在区块链中的最新的、确认的交易。类似地,在一些实施例中,交易502是未来交易,其尚未被确认并且尚未被包括在区块链中,并且表示对转移对由先前交易控制的数字资产的至少一部分的控制的尝试。

[0196] 在一些实施例中,锁定脚本506是这样的脚本:通过指定如本文所述为了传输输出而需要满足的条件来阻碍交易。如示例问题500中所示,交易502的锁定脚本506无法查询区块链的区块504,从而例如确定交易的数量或访问特定交易。如示例问题500中所示,交易502的锁定脚本506无法查询区块链的区块504,无法查询区块504的区块头508,并且无法查询区块504的交易510。在区块504包括交易502(表示为区块504的交易510中的交易Tx_i 512)的实施例中,交易502的锁定脚本506也不能查询交易Tx_i 512(例如,无法查询包含锁定脚本506的交易)。本文所述的方法使交易502的锁定脚本506能够查询区块链的区块504,查询区块504的区块头508,查询区块504的交易510,以及查询区块504的交易510中的交易Tx_i 512(例如,使锁定脚本能够查询包含锁定脚本506的交易)。

[0197] 图5以示意图形式示出了根据实施例的示例环境600,其中数据被验证为区块头,所述区块头用于基于脚本的区块链交互。区块头是区块链的区块的一部分,所述区块链包含有关区块的数据。例如,区块头602是80字节的数据,前4个字节存储区块链实施的版本(“nVersion”),接下来的32个字节(或256位)存储区块链先前区块的散列(“HashPrevBlock”),接下来的32个字节存储该区块的交易的默克尔(Merkle)树的散列(“HashMerkleRoot”,本文对此进行了详细说明),接下来的4个字节存储该区块的创建时间(“nTime”),接下来的4个字节存储该区块的32位难度值(“nBits”),最后4个字节存储该区块的随机种子(“nNonce”)。

[0198] 可以观察到,如果区块头602是用作锁定脚本的输入的数据(例如,作为解锁脚本而提供),则通过将区块头602上的约束编码为锁定脚本,解锁脚本数据必须包含有效的区块头,否则其将不是有效的解锁脚本。对区块头例如区块头602的三个约束是:i)80字节长;ii)区块头的nBits字段必须大于或等于某个难度值;iii)区块头的双SHA256必须小于或等于难度值。这些约束中的每一个在本文中被示为单独的脚本,为了便于图示,省略了链接脚本的细节(例如,分支约束和维护堆栈)。

[0199] 当这三个脚本与省略的链接脚本详细信息结合在一起时,会形成一个单独的脚本,这是一个OP_CHECKBLOCKVERIFY脚本的示例,用于验证解锁脚本中的<Data>是有效的区块头。在所示的示例中,三个脚本返回真(true)或假(false),但是作为单个脚本链接在一起以生成OP_CHECKBLOCKVERIFY脚本时,通常会将OP_VERIFY添加到每个脚本的末尾。可以想到,OP_CHECKBLOCKVERIFY脚本的其他实施可以实现,并且可以视为在本发明的范围内。

[0200] 在步骤604中,提供所谓的区块头作为解锁脚本,表示为<Data>。

[0201] 在步骤606,执行第一脚本,该脚本验证<Data>的大小是80字节(例如,第一约束)。表2示出了验证<Data>的大小为80字节的第一个脚本。

[0202] 表2

[0203]

类型	脚本
解锁脚本	<Data>
锁定脚本	*// Check size is 80 bytes* OP_SIZE <80> OP_EQUAL

[0204] 在步骤608中,执行第二脚本,其验证区块头的“nBits”字段大于或等于区块链难

度值。表3示出了验证区块头的“nBits”字段大于或等于区块链难度值的第二个脚本。在实施例中,难度值用作解锁脚本的数据项。在实施例中,较高的难度值(例如,大于0x1D00FFF的值)被用作目标难度。在实施例中,较低的难度值(例如,小于0x1D00FFF的值)被用作目标难度。

[0205] 表3

类型	脚本
解锁脚本	<Data>
锁定脚本	*// Extract nBits (4 bytes, starting at byte 72)* <72> <4> OP_SUBSTR *// Check Difficulty* <0x1D00FFFF> OP_GREATERTHANOREQUAL

[0206]

[0207] 在步骤610中,执行第三脚本,该脚本验证区块头的双SHA256(如上所述,由操作码OP_HASH256实现)小于或等于表征的难度值。表4示出了第三个脚本,该脚本验证了区块头的双SHA256小于或等于表征的难度值。表4中示出的第三个脚本引入了两个附加操作码:

[0208] • OP_LBYTESHIFT:对任意大小的数据执行左字节移位

[0209] ●OP_BIGLESSTHANOREQUAL:将任意大小的数据解释为数字,并比较两个数字以确定第一个数字是否小于或等于第二个数字

[0210] 表4

类型	脚本
解锁脚本	<Data>
锁定脚本	*// Extract nBits* OP_DUP <72> <4> OP_SUBSTR *// Extract Last 3 Bytes* OP_DUP <1> <3> OP_SUBSTR *// Extract First Byte* OP_SWAP <0> <1> OP_SUBSTR *// Calculate Target* <3> OP_SUB OP_LBYTESHIFT *// Check against Block Header HASH256* OP_OVER OP_HASH256 OP_BIGLESSTHANOREQUAL

[0211]

[0212] 图6以流程图形式示出了根据实施例的示例过程700,所述示例过程用于验证数据是基于脚本的区块链交互中的区块头。过程700中的部分或全部可以在配置有可执行指令和/或其他数据的一个或多个计算机系统的控制下执行,并且可以实现为在一个或多个处

理器上共同执行的可执行指令。可执行指令和/或其他数据可以存储在非暂时性计算机可读存储介质上(例如,计算机程序永久存储在磁、光、或闪存介质上)。示例过程700可以由区块链网络的节点执行。也就是说,节点可执行示例过程700,用于验证数据是结合图6描述的基于脚本的区块链交互中的区块头。这样的节点可以由任何合适的计算设备组成(例如,由数据中心中的服务器组成,由客户端计算设备组成,由计算资源服务提供商的分布式系统中的多个计算设备组成,或由任何合适的电子客户端设备例如结合图1描述的计算设备200组成)。

[0213] 在示例过程700的步骤702中,系统接收<Data>。在示例过程700的步骤704中,系统如上所述结合图5开始OP_CHECKBLOCKVERIFY脚本。在示例过程700的步骤706中,系统通过执行例如上文表2中所示的脚本来验证<Data>的大小为80字节。在示例过程700的步骤708中,确定先前的脚本(例如,以上表2中所示的脚本)是否通过。

[0214] 如果在示例过程700的步骤708中确定先前的脚本没有通过,则在示例过程700的步骤718中,<Data>不是有效的区块头,并且锁定脚本解锁失败。

[0215] 如果在示例过程700的步骤708中确定先前的脚本确实通过了,则在示例过程700的步骤710中,系统通过执行例如上文表3中所示的脚本来验证<Data>的“nBits”字段大于或等于区块链难度。在示例过程700的步骤712中,确定先前的脚本(例如,上文表3中所示的脚本)是否通过。

[0216] 如果在示例过程700的步骤712中确定先前的脚本没有通过,则在示例过程700的步骤718中,<Data>不是有效的区块头,并且锁定脚本解锁失败。

[0217] 如果在示例过程700的步骤712中确定先前的脚本确实通过了,则在示例过程700的步骤714中,系统通过执行例如上文表4中所示的脚本来验证<Data>的SHA256小于或等于所需目标。在示例过程700的步骤716中,确定先前脚本(例如,上文表4中所示的脚本)是否通过。

[0218] 如果在示例过程700的步骤716中确定先前的脚本未通过,则在示例过程700的步骤718中,<Data>不是有效的区块头,并且锁定脚本解锁失败。

[0219] 如果在示例过程700的步骤716中确定先前的脚本确实通过了,则在示例过程700的步骤720中,<Data>是有效区块头,并且锁定脚本解锁成功。

[0220] 注意,在图6中示出的示例过程700中执行的一个或多个操作可以以各种顺序(包括并行)和组合来执行。

[0221] 图7以示意图形式示出了根据实施例的示例环境800,其中数据验证为用于基于脚本的区块链交互的先前区块的区块头。如本文所述,区块头是区块链的区块的一部分,该区块链包含有关该区块的数据。区块头的一部分是先前区块头(“HashPrevBlock”)的散列。如果将第一区块头作为输入提供给锁定脚本(例如,作为解锁脚本的一部分提供),并且所谓针对区块链中下一区块的第二区块头也作为输入提供给锁定脚本(例如,作为解锁脚本的一部分而提供),那么通过对第一区块头的散列是解锁脚本中的第二区块的HashPrevBlock的约束进行编码,如果第一区块头的散列不是第二区块的HashPrevBlock,那么两个区块头将不是有效的解锁脚本。

[0222] 在步骤804中,提供第一区块头802作为解锁脚本的数据(表示为<Data1>),并且在步骤810中,提供第二区块头808作为如下所示用于解锁脚本的数据(表示为<Data 2>)。在

步骤806中,脚本(例如以上结合图5和图6所描述的OP_CHECKBLOCKVERIFY脚本)用于验证<Data 1>是区块头。在步骤812中,脚本(例如以上结合图5和图6所描述的OP_CHECKBLOCKVERIFY脚本)用于验证<Data 2>是区块头。

[0223] 在步骤814中,执行脚本以验证第一区块头<Data 1>的双SHA256等于第二块头<Data 2>的HashPrevBlock字段。表5中示出了一个脚本的示例,该脚本验证了第一区块头<Data 1>的双SHA256等于第二区块头<Data 2>的HashPrevBlock字段。表5中所示的脚本是OP_CHECKCHAINVERIFY脚本的示例,用以验证<Data 1>是紧接在<Data 2>之前的有效区块。可以想到,OP_CHECKCHAINVERIFY脚本的其他实施方式可以实现,并且可以将其视为在本发明的范围内。应注意,表5中示出的OP_CHECKCHAINVERIFY脚本的最终操作码是OP_EQUAL,因为执行后“true”值应该在堆栈上保留。在将OP_CHECKCHAINVERIFY脚本与其他脚本结合使用的实施例中,该脚本的最终OP_EQUAL操作码更改为OP_EQUALVERIFY操作码。

[0224] 表5

[0225]

类型	脚本
解锁脚本	<Data 1> <Data 2>
锁定脚本	*// Validate <Data 1> and <Data 2> are Block Headers* OP_DUP OP_CHECKBLOCKVERIFY OP_OVER OP_CHECKBLOCKVERIFY *// Extract Previous Block Header from <Data 2>* OP_DUP <4> <32> OP_SUBSTR *// Compute HASH256 of Block Header from <Data 1>* OP_SWAP OP_HASH256 *// Compare* *// Replace with OP_EQUALVERIFY when used with other scripts* OP_EQUAL

[0226] 图8以流程图形式示出了根据实施例的示例过程900,所述示例过程用于验证数据是基于脚本的区块链交互中的先前区块的区块头。过程900的部分或全部可以在配置有可执行指令和/或其他数据的一个或多个计算机系统的控制下执行,并且可以实现为在一个或多个处理器上共同执行的可执行指令。可执行指令和/或其他数据可以存储在非暂时性计算机可读存储介质上(例如,计算机程序永久存储在磁、光或闪存介质上)。示例过程900可以由节点执行。也就是说,节点可执行示例过程900,所述示例过程用于验证数据是结合图8所述的基于脚本的区块链交互中的先前块的区块头。这样的节点可以由任何合适的计算设备组成(例如,由数据中心中的服务器组成,由客户端计算设备组成,由计算资源服务提供商的分布式系统中的多个计算设备组成,或由任何合适的电子客户端设备(例如结合图1描述的计算设备200)组成)。

[0227] 在示例过程900的步骤902中,系统接收<Data 1>和<Data 2>,这是两个所谓的区块头。在示例过程900的步骤904中,系统如上述结合图7开始对<Data 1>和<Data 2>进行

OP_CHECKCHAINVERIFY过程。在示例过程900的步骤906中,系统如上文结合图5和图6通过对<Data 1>执行OP_CHECKCHAINVERIFY脚本来验证<Data 1>是区块头。

[0228] 在示例过程900的步骤908中,确定先前的脚本(例如,<Data 1>上的OP_CHECKBLOCKVERIFY脚本)是否通过(即,OP_CHECKBLOCKVERIFY脚本返回了真(true))。如果在示例过程900的步骤908中确定先前的脚本没有通过,则在示例过程900的步骤910中,<Data 1>不是有效的区块头,并且在示例过程900的步骤922中,<Data 1>不是<Data 2>的先前区块,因此锁定脚本的解锁失败。

[0229] 如果在示例过程900的步骤908中确定先前脚本确实通过了,则在示例过程900的步骤912中,系统如上所述结合图5和图6通过执行OP_CHECKBLOCKVERIFY脚本来验证<Data 2>是区块头。

[0230] 在示例过程900的步骤914中,确定先前的脚本(例如,<Data 2>上的OP_CHECKBLOCKVERIFY脚本)是否通过(即,OP_CHECKBLOCKVERIFY脚本返回了真(true))。如果在示例过程900的步骤914中确定先前的脚本未通过,则在示例过程900的步骤916中,<Data 2>不是有效的区块头,并且在示例过程900的步骤922中,<Data 1>不是<Data 2>的先前区块,因此锁定脚本的解锁失败。

[0231] 如果在示例过程900的步骤914中确定先前的脚本确实通过了,则在示例过程900的步骤918中,系统验证<Data 1>的双SHA256等于<Data 2>的“HashPrevBlock”字段。

[0232] 在示例过程900的步骤920中,确定<Data 1>的双SHA256是否等于<Data2>的“HashPrevBlock”字段。如果在示例过程900的步骤920中确定<Data 1>的双SHA256不等于<Data 2>的“HashPrevBlock”字段,则在示例过程900的步骤922中,<Data 1>不是<Data 2>的先前区块,因此锁定脚本解锁失败。

[0233] 如果在示例过程900的步骤920中确定<Data 1>的双SHA256等于<Data 2>的“HashPrevBlock”字段,则在示例过程900的步骤924中,<Data 1>是<Data2>的先前区块,因此锁定脚本的解锁成功。

[0234] 注意,在图8中示出的示例过程900中执行的一个或多个操作可以以各种顺序(包括并行)和组合执行。

[0235] 图9以示意图的形式示出了根据实施例的示例环境1000,其中数据验证为用于基于脚本的区块链交互的区块头链。图9示出的示例环境1000是结合图7和图8描述的操作的扩展。在图9中示出的示例环境1000中,将区块头链1010(即,区块头的链)的第一区块头1002的双SHA256与区块头链1010的第二区块头1004的HashPrevBlock进行比较,将区块头链1010的第二区块头1004的双SHA256与区块头链1010的第三区块头1006的HashPrevBlock进行比较,依此类推,直到将区块头链1010的倒数第二个区块(未示出)的双SHA256与区块头链1010的最后一个区块头1008的HashPrevBlock进行比较。

[0236] 示例环境1000示出了通过对每对区块重复表5中所示的脚本,可以将表5中所示的OP_CHECKCHAINVERIFY脚本概括为附加的块。这样的脚本在本文中称为OP_CHECKBLOCKCHAINVERIFY。

[0237] 例如,OP_CHECKBLOCKCHAINVERIFY脚本可以将区块头链中的多个区块头和区块头链的区块头作为输入参数,并重复执行OP_CHECKCHAINVERIFY。表6示出了一个示例,说明如何使用此脚本来验证六个区块头的链。

[0238] 表6

[0239]

类型	脚本
解锁脚本	<Data 1> <Data 2> <Data 3> <Data 4> <Data 5> <Data 6>
锁定脚本	*// Perform OP_CHECKCHAINVERIFY for the five pairs of Block Headers* <5> OP_CHECKBLOCKCHAINVERIFY

[0240] 表7示出了OP_CHECKBLOCKCHAINVERIFY的实施示例,该实施对大量区块重复OP_CHECKCHAINVERIFY的操作。在表7中所示的OP_CHECKBLOCKCHAINVERIFY的示例实施中,重复该部分以覆盖<d-1>范围的值,该部分在区块头中循环(例如,<d>区块头循环<d-1>次)且对每对区块头执行OP_CHECKCHAINVERIFY操作,通过使用alt堆栈来保存和维护顺序。

[0241] 表7

[0242]

类型	脚本
解锁脚本	<Data 1> <Data 2> … <Data d-1> <Data d>
锁定脚本	*// d Block Headers, d-1 Chains* <d-1> ----Repeat the following section a large number of times to cover all possible values of <d>---- *// Save a copy of current d to alt stack* OP_DUP OP_TOALTSTACK *// If d is not 0 do branch 1, else do branch 2* OP_IF *// Check chain on first 2 items. Duplicate 2nd item to maintain a copy for the next iteration* OP_OVER OP_SWAP OP_CHECKCHAINVERIFY *// Get d from alt stack and decrement* OP_FROMALTSTACK OP_1SUB OP_ELSE *// Get d from alt stack and do nothing* OP_FROMALTSTACK OP_ENDIF ----------end section----------

[0243] 图10以流程图形式示出了根据实施例的示例过程1100,所述示例过程用于验证数据是基于脚本的区块链交互中的区块头链。过程1100的部分或全部可以在配置有可执行指

令和/或其他数据的一个或多个计算机系统的控制下执行,并且可以实现为在一个或多个处理器上共同执行的可执行指令。可执行指令和/或其他数据可以存储在非暂时性计算机可读存储介质上(例如,计算机程序永久存储在磁、光、或闪存介质上)。示例过程1100可以由节点执行。也就是说,节点可执行示例过程1100,用于验证数据是结合图10描述的基于脚本的区块链交互中的区块头链。这样的节点可以由任何合适的计算设备组成(例如,由数据中心中的服务器组成,由客户端计算设备组成,由计算资源服务提供商的分布式系统中的多个计算设备组成,或由任何合适的电子客户端设备例如结合图1描述的计算设备200组成)。

[0244] 在示例过程1100的步骤1102中,系统接收有序数据集合,即<Data 1>至<Data n>。在示例过程1100的步骤1104中,系统如上文结合图9所描述的,对所述有序数据集合开始OP_CHECKBLOCKCHAINVERIFY脚本。

[0245] 在示例过程1100的步骤1106中,系统至少部分地基于有序数据集合的顺序,从有序数据集合中选择第一数据项(例如,<Data 1>)。在示例过程1100的步骤1108中,系统至少部分地基于有序数据集合的顺序,从有序数据集合中选择第二数据项(例如,<Data 2>)。在实施例中,系统使用例如上文结合图5和图6描述的OP_CHECKBLOCKVERIFY脚本来验证第一数据项和第二数据项是有效区块头。

[0246] 在示例过程1100的步骤1110中,系统通过对第一数据项和第二数据项执行OP_CHECKCHAINVERIFY来验证第一数据项是第二数据项的先前区块,如表5中结合图7至图8所示。

[0247] 在示例过程1100的步骤1112中,确定第一数据项和第二数据项上的OP_CHECKCHAINVERIFY脚本是否通过(即,第一数据项是否是第二数据项中的区块头的先前区块头)。

[0248] 如果在示例过程1100的步骤1112中确定第一数据项和第二数据项上的OP_CHECKCHAINVERIFY脚本没有通过,则在示例过程1100的步骤1114中,第一数据项不是第二数据项中的区块头的先前区块头,并且在示例过程1100的步骤1116中,数据的有序集合不是有效的区块头链。

[0249] 如果在示例过程1100的步骤1112中确定第一数据项和第二数据项上的OP_CHECKCHAINVERIFY脚本确实通过了,则在示例过程1100的步骤1118中,确定在有序数据集合中是否存在更多数据。

[0250] 如果在示例过程1100的步骤1118中确定有序数据集合中存在更多数据,则在示例过程1100的步骤1120中,第二数据项成为示例过程1100中的第一数据项,在示例过程1100的步骤1122中,至少部分地基于数据顺序从有序数据集合中选择新的第二数据项,并且在示例过程1100的步骤1110中,如表5中所示和结合图7和图8所描述的,系统通过对新的第一数据项和新的第二数据项执行OP_CHECKCHAINVERIFY来验证新的第一数据项是新的第二数据项的先前区块。

[0251] 如果在示例过程1100的步骤1118中确定在有序数据集合中没有更多数据,则在示例过程1100的步骤1124中,有序数据集合是有效的区块头链。

[0252] 注意,在图10中示出的示例过程1100中执行的一个或多个操作可以以各种顺序(包括并行地)和组合来执行。

[0253] 图11以示意图形式示出了根据实施例的示例环境1200,该示例环境1200示出了可在基于脚本的区块链交互中使用的区块头的默克尔树(Merkle tree)。默克尔树(也称为“二进制散列树”)是一种数据结构,用于验证区块中的交易集合。默克尔树是包含交易的加密散列的二叉树。但是,通过对数据进行重复散列和连接,可以将默克尔树存储为单个散列值(例如,单个32字节或256位值)。

[0254] 在示例环境1200中,通过以下方式生成默克尔树1204:i)计算区块的每个交易的散列;ii)选择一对散列并将它们连接在一起;iii)计算连接值的散列;iv)在树的连续更高级别上重复该过程,直到产生最终的散列值(“默克尔根”)为止。然后,将该默克尔根作为“HashMerkleRoot”值插入到区块头中。

[0255] 例如,从交易1206开始,计算该交易的散列1208。如本文所使用的,交易1206将称为T_A,交易1206的散列1208将称为$H(T_A)$。类似地,交易1210(T_B)具有散列1212$H(T_B)$。为了产生散列1216,在散列1214之前,将左值与右值连接起来,因此散列1216是$H(H(T_A)+H(T_B))$,其中“+”运算符在这种情况下是超载的,表示连接(例如,如果$H(T_A)$为“1234”,$H(T_B)$为“5678”,则$H(T_A)+H(T_B)$为“12345678”,$H(H(T_A)+H(T_B))$是“12345678”的散列。

[0256] 示例环境1200还示出了默克尔分支,该默克尔分支是默克尔树中的节点集合,其与默克尔根组合,可用于验证用于特定叶节点(即,特定交易)的数据在默克尔树中。在给定默克尔根和默克尔分支的情况下,可以以下方式验证默克尔树中给定交易(例如,交易1218)的存在。首先,计算交易的散列1220。然后,计算默克尔树中下一级的散列1224。然而,如上所述,在默克尔树的下一级的散列1224需要另一交易(例如,作为默克尔树中的交易1218的伙伴交易)的散列1222。该散列1222是默克尔分支中的第一节点。

[0257] 在给定散列1224的情况下,计算在默克尔树的下一级的散列1226。该计算需要默克尔树的其他交易(例如,另一分支)的散列1216。该散列1216是默克尔分支中的第二节点。可以想到,在默克尔树的每个级别上,默克尔分支需要在树的“其它”分支中的交易组合的散列。因此,假设散列1230的左分支是包含交易1218(可以计算为上述所述的)的分支,则要计算散列1230,需要右分支的散列1228(默克尔分支中的另一个节点)。最后,为了计算默克尔根,需要默克尔树的整个右侧的散列1232。可以用来验证交易1218在默克尔树1204中的默克尔分支是散列1230,散列1228,散列1216和散列1222。由于树的深度为4,因此四个节点位于默克树中。应注意,默克尔树的深度取决于区块中的交易数,并且是交易数的$\log_2$(即,对于512个交易,树的深度为9,用于验证默克尔分支的节点有9个节点,对于1024个交易,树的深度为10,验证节点的默克尔分支具有10个节点,对于2048个交易,树的深度为11,验证节点的默克尔分支具有为11节点,等等)。

[0258] 图12以示意图形式示出了根据实施例的示例环境1300,其中交易验证为包括在用于基于脚本的区块链交互的区块头中。如本文所述,区块头是区块链的区块的一部分,该区块链包含有关该区块的数据。区块头的一部分是默克尔树的默克尔根(“HashMerkleRoot”),所述默克尔树可用于验证区块中的交易。如果提供区块头作为锁定脚本的输入(例如,提供作为解锁脚本),以及用于交易和该交易的默克尔分支,则可以编码约束(所述约束为:区块头的HashMerkleRoot与从默克尔分支和交易计算出的默克尔根相同),如果区块头的HashMerkleRoot与计算出的默克尔根不同,则区块头、默克尔分支和交易将不是有效的解锁脚本。

[0259] 在步骤1304中,提供第一区块头1302作为用于解锁脚本的数据(表示为<Data 1>)。在步骤1306中,脚本例如上文结合图5和图6所述的OP_CHECKBLOCKVERIFY脚本用于验证<Data 1>是区块头。在步骤1308,从有效区块头中提取HashMerkleRoot。

[0260] 在步骤1312中,提供默克尔分支1310作为用于解锁脚本的数据(表示为<Data 2>)。默克尔分支1310可以是大约10、11或12个节点,以编码格式提供,该编码格式表征特定节点是如本文所述的连接操作的左输入还是连接操作的右输入,尽管本文未示出。

[0261] 表8示出了OP_CALCMERKLEROOT的示例实施,其中以编码格式提供默克尔分支1310,该编码格式包括对$<hash_1>$,$<x_1>$,$<hash_2>$,$<x_2>$,...,$<hash_d>$,$<x_d>$和值<d>,其中<d>表示对集合中有多少对,每个$<hash_i>$是默克尔树中某个值的散列,每个$<x_i>$是一个值(例如0或1),该值表征是左连接还是右连接。在表8所示的OP_CALCMERKLEROOT的示例实施中,$<x_i>$中的值1表征左侧连接,而$<x_i>$中的值0表征右侧连接。

[0262] 表8

[0263]

类型	脚本
解锁脚本	<Merkle Branch> <Tx ID>
锁定脚本	*// Swap current Merkle hash and d* OP_SWAP ----Repeat the following section a large number of times to cover all possible values of <d>---- *// Duplicate current d and save a copy to alt stack* OP_DUP OP_TOALTSTACK *//If d is not 0, calculate the next Merkle hash* OP_IF *// Swap x to the top and check if it is 1, if so swap to put our current hash on the right side* OP_SWAP OP_IF OP_SWAP OP_ENDIF *// Concatenate and Hash* OP_CONCAT OP_HASH256 *// Decrement d* OP_FROMALTSTACK OP_1SUB OP_ELSE *// Grab d from alt stack and do nothing* OP_FROMALTSTACK OP_ENDIF ------------------end section-------------

[0264] 在步骤1316中,提供交易1314作为用于解锁脚本的数据(表示为<Data3>)。在步骤1318中,使用OP_CALCMERKLEROOT脚本(在此未示出)从默克尔根<Data 2>和交易<Data 3>计算默克尔根。

[0265] 在步骤1320中,执行脚本以验证从<Data 1>提取的HashMerkleRoot与对<Data 2>和<Data 3>执行的由OP_CALCMERKLEROOT脚本计算出的默克尔根相等。表9显示了一个脚本示例,该脚本验证了从<Data 1>提取的HashMerkleRoot与由<Data 2>和<Data 3>计算出的默克尔根相同。表9中显示的脚本是OP_CHECKBLOCKTXVERIFY脚本的一个示例,使用<Data 2>的默克尔分支验证区块头<Data 1>的区块包含<Data 3>的交易。可以想到,OP_CHECKBLOCKTXVERIFY脚本的其他实施可以实现,并且可以视为在本发明的范围内。在表9所示的OP_CHECKBLOCKTXVERIFY脚本中,为了便于说明,省略了OP_CHECKBLOCKVERIFY操作。

[0266] 表9

[0267]

类型	脚本
解锁脚本	<Data 1> <Data 2> <Data 3>
锁定脚本	*// Calculate Merkle Root* OP_HASH256 OP_ CALCMERKLEROOT *// Extract Merkle Root from Block Header in <Data 1> (32 bytes starting at byte 36)* OP_SWAP <36> <32> OP_SUBSTR *// Compare* OP_EQUALVERIFY

[0268] 图13以流程图形式示出了根据实施例的示例过程,所述示例过程用于验证交易包括在基于脚本的区块链交互的区块头中。处理1400的部分或全部可以在配置有可执行指令和/或其他数据的一个或多个计算机系统的控制下执行,并且可以实现为在一个或多个处理器上共同执行的可执行指令。可执行指令和/或其他数据可以存储在非暂时性计算机可读存储介质上(例如,计算机程序永久存储在磁、光、或闪存介质上)。示例过程1400可以由节点执行。也就是说,节点可执行示例过程1400,用于验证结合图13描述的基于脚本的区块链交互中的区块头中包括交易。这样的节点可以由任何合适的计算设备组成(例如,由数据中心中的服务器组成,由客户端计算设备组成,由计算资源服务提供商的分布式系统中的多个计算设备组成,或由任何合适的电子客户端设备例如结合图1描述的计算设备200组成)。

[0269] 在示例过程1400的步骤1402中,如上文结合图12所描述,系统接收<Data 1>,<Data 2>和<Data 3>。在示例过程1400的步骤1402中,也如上文结合图12所述,系统对<Data 1>,<Data 2>和<Data 3>执行OP_CHECKBLOCKTXVERIFY脚本。

[0270] 在示例过程1400的步骤1406中,如上文结合图5和图6所述,系统使用例如OP_CHECKBLOCKVERIFY脚本来验证<Data 1>是有效区块头。

[0271] 在示例过程1400的步骤1408中,确定<Data 1>上的OP_CHECKBLOCKVERIFY脚本是否通过。如果在示例过程1400的步骤1408中确定<Data 1>上的OP_CHECKBLOCKVERIFY脚本未通过,则在示例过程1400的步骤1410中,<Data 1>不是有效的区块头,并且在示例过程1400的步骤1412中,<Data 3>的交易被验证不在具有区块头<Data 1>的区块中。

[0272] 如果在示例过程1400的步骤1408中确定<Data 1>上的OP_CHECKBLOCKVERIFY脚本确实通过了,则在示例过程1400的步骤1414中,如上文结合图12所述,系统从<Data 1>中提取HashMerkleRoot字段。在示例过程1400的步骤1416中,也如上文结合同13所述,系统从<Data 2>和<Data 3>计算默克尔根。

[0273] 在示例过程1400的步骤1418中,也如上文结合图12所述,系统验证从<Data 1>提取的HashMerkleRoot等于从<Data 2>和<Data 3>计算出的默克尔根。

[0274] 在示例过程1400的步骤1420中,确定从<Data 1>提取的HashMerkleRoot是否等于从<Data 2>和<Data 3>计算出的默克尔根。在示例过程1400的步骤1420中,如果确定从<

Data 1>提取的HashMerkleRoot不等于从<Data 2>和<Data3>计算的默克尔根,则在示例过程1400的步骤1412,<Data 3>的交易被验证不在带有区块头<Data 1>的区块中。

[0275] 在示例过程1400的步骤1420中,如果确定从<Data 1>提取的HashMerkleRoot确实等于从<Data 2>和<Data 3>计算出的默克尔根,则在示例过程1400的步骤1422中,<Data 3>的交易被验证在带有区块头<Data 1>的区块中。

[0276] 注意,在图13中示出的示例过程1400中执行的一个或多个操作可以以各种顺序(包括并行地)和组合来执行。

[0277] 图14以示意图形式示出了根据实施例的示例环境1500,示出了如何使用本文所述的方法来解决访问与基于脚本的区块链交互相关联的区块链的问题。在示例环境1500中,交易1502(例如结合图2所述的先前交易,如先前交易302或解锁交易304)中的锁定脚本1506能够使用一个或多个结合图5-13所述的方法来查询区块链1504。这与结合图3所述的示例问题400相反。类似地,在示例环境1500中,交易1502中的锁定脚本1506能够使用结合图5至图13所述的一种或多种方法来查询区块链1504的区块1508。这与结合图4所述的示例问题500相反。在实施例中,区块1508的交易1510与交易1502相同(例如,为交易Tx_i)。

[0278] 图15以示意图形式示出了根据实施例的示例环境1600,示出了与基于脚本的区块链交互相关联的区块链的状态。区块链包含自区块链产生以来的所有区块。在示例环境1600中,区块链1602的第一区块1604的区块头是最小nTime(例如,最早或时间最前)的区块链1602的状态1606的记录。使用本文中描述的方法,交易的锁定脚本可以查询区块链1602,可以查询第一区块1604,并且因此可以在最低nTime查询区块链1602的状态1606。使用本文中描述的方法,交易的锁定脚本可以查询区块链1602,可以查询区块链1602的区块1608(可以是包含交易的区块),因此可以查询在区块的nTime的区块链1602的状态1610。使用本文描述的方法,交易的锁定脚本可以查询区块链1602,可以查询区块链1602的最新区块1612,并且因此可以在最新区块的nTime查询区块链1602的状态1614(例如,可以查询区块链的当前状态)。这种状态查询的使用在图16和图17中进行了详细描述。

[0279] 图16以示意图形式示出了根据实施例的示例环境1700,在所述环境中,在基于脚本的区块链交互中确认了区块链的状态。如本文所述,区块头是区块链的区块的一部分,该区块链包含有关该区块的数据。区块头的一部分是默克尔树的默克尔根(“HashMerkleRoot”),所述默克尔树可用于验证区块中的交易。如果提供区块头作为锁定脚本的输入(例如,提供作为解锁脚本),以及用于交易和该交易的默克尔分支,则可以编码约束(所述约束为:区块头的HashMerkleRoot与从默克尔分支和交易计算出的默克尔根相同),如果区块头的HashMerkleRoot与计算出的默克尔根不同,则区块头、默克尔分支和交易将不是有效的解锁脚本。

[0280] 在实施例中,检查区块链的状态和/或确认区块链的状态包括检查区块链的其他方面,包括但不限于,检查区块链的一个或多个区块的区块头,检查区块链的一个或多个区块之间的关系,检查区块链的区块数量,检查区块链的单个区块中的一个或多个交易,或检查区块链的多个区块中的多个交易。

[0281] 在步骤1708中,提供区块头1706作为用于解锁脚本的数据(表示为<数据1>)。在一个实施例中,脚本例如以上结合图5和图6描述的OP_CHECKBLOCKVERIFY脚本用于验证<Data 1>是一个区块头。在步骤1712中,如以上结合图12至图13所述,提供默克尔分支1710作为用

于解锁脚本的数据(表示为<Data 2>)。在步骤1716中,提供交易1714作为用于解锁脚本的数据(表示为<数据3>)。在一个实施例中,如上文结合图12和13所述,使用OP_CALCMERKLEROOT脚本从默克尔根<Data 2>和交易<Data 3>计算默克尔根。

[0282] 在步骤1718中,执行OP_CHECKBLOCKTXVERIFY脚本以验证从<Data 1>提取的HashMerkleRoot与从上文表9所示对<Data 2>和<Data 3>执行的OP_CALCMERKLEROOT脚本计算的默克尔根相同。

[0283] 应注意,根据实施例,步骤1718示出了可以在例如示例环境1700之类的环境中执行的脚本类型的示例,其中在基于脚本的区块链交互中确认了区块链的状态。步骤1718示出了第一约束集合,其使得使用操作码将来自区块链的数据集合注入到交易中。在实施例中,如本文所述,OP_CHECKBLOCKVERIFY用于验证数据集合包括区块头。在实施例中,OP_CHECKBLOCKTXVERIFY(即,如步骤1718所示)用于验证该数据集合包括交易,并且该交易包括在区块头中。可以想到,在本文描述的第一约束集合中的约束的示例是说明性示例,第一约束集合中的其他类型的约束可以视为在本发明的范围内。下文关于步骤1722对第二约束集合进行描述。

[0284] 在示例环境1700中,在步骤1704中,脚本用于验证区块头<Data 1>是作为公共区块链的一部分的区块的区块头,而不是如上所述通过使用验证脚本(例如OP_CHECKBLOCKCHAINVERIFY)专门为模拟该公共区块链的区块而建立的区块。在实施例中,脚本用于验证区块头<Data 1>是作为公共区块链的一部分的区块的区块头,可以将区块链1702或如上文所述完整或部分区块头链提供给所述公共区块链。

[0285] 在示例环境1700中,在步骤1722中,使用脚本来验证交易的其他方面。在步骤1720中,在步骤1722中将交易1714提供为用于解锁脚本的数据(表示为<Data 3>)。应注意,在一些实施例中,步骤1720中的交易<数据3>是步骤1716中的交易<Data 3>的副本,此时,例如,步骤1718中的OP_CHECKBLOCKTXVERIFY脚本执行步骤1716中的交易<Data 3>。

[0286] 与上述步骤1718所示,应该注意的是,步骤1722示出了可以在例如示例环境1700之类的环境中执行的脚本类型的示例,在该环境中,根据实施例,在基于脚本的区块链交互中确认了区块链的状态。如上文结合步骤1718所述,步骤1722示出了第二约束集合,其与来自区块链的数据集合相关联,所述区块链被使用操作码注入到交易中。在实施例中,第二约束集合包括关于该数据集合的数据项的值的约束(即,包括该数据集合的数据项例如小于、等于或大于该值的约束)。在一个实施例中,第二约束集合包括根据数据确定的约束(例如,如果与区块头相关联的区块中的交易数大于100,则执行“A”,否则,执行“B”)。可以想到,本文描述的第二约束集合中的约束的示例是说明性示例,可以认为第二约束集合中的其他类型的约束在本发明的范围内。

[0287] 如果在步骤1704中用于验证区块头<Data 1>是区块(所述区块是公共区块链的一部分,而不是专门用来模拟公共区块链的区块的区块)的区块头的脚本通过了,则在步骤1718中,OP_CHECKBLOCKTXVERIFY脚本验证从<Data 1>提取的HashMerkleRoot与从对<Data 2>和<Data 3>执行的OP_CALCMERKLEROOT脚本计算的默克尔根相同,并且在步骤1722中,用于验证交易的其他方面的脚本通过了,然后在步骤1724中锁定脚本成功。在实施例中,步骤1704说明第三约束集合,这些约束与来自使用操作码注入到交易中的区块链的数据集合相关联(例如,约束是:该区块是公共区块链的区块)。

[0288] 应注意,如上所述,在步骤1722中,用于验证交易的其他方面的脚本包括第二约束集合的约束,该第二约束集合与使用操作码来自注入到交易中的区块链的数据集合相关联。这样,在实施例中,第二约束集合包括交易的多个方面的验证,包括但不限于受让方、转让方、交易的数量或交易的其他方面。另外,应注意,在步骤1722中,用于验证交易的其他方面的脚本可包括区块链的各方面,区块头链的各方面,区块的各方面或区块链环境的其他方面。例如,在步骤1722中,用于验证交易的其他方面并且包括第二约束集合的约束的脚本可包括以下用途的约束:验证区块的nTime,验证区块中的多个交易,确定两个区块之间经过的时间量,确定最小区块高度,或确定区块链环境的其他此类方面。还应注意,通过在脚本的末尾不包括“验证”操作码(该脚本用于验证步骤1722的交易的其他方面),可以引入分支条件(例如,如果区块中的交易有1000多个,爱丽丝(Alice)可以解锁;如果区块中的交易不超过1000个,则鲍勃(Bob)可以解锁)。分支条件的示例在下文结合图29更详细地描述。

[0289] 与步骤1704(所述脚本用于验证区块头<Data 1>是作为公共区块链的一部分的区块的区块头而不是专门为模拟该公共区块链的区块而建立的区块)相关的一个问题是,在一些实施例中,从计算上来说,创建一个专门模拟公共区块链的区块的区块可以相当容易。考虑例如表10的锁定脚本,该锁定脚本要求解锁脚本中的数据包括区块头(例如,解锁脚本可以不仅仅包括区块头<Data1>)。

[0290] 表10

[0291]

类型	脚本
解锁脚本	<Data 1>
锁定脚本	*// Validate that <Data 1> is a block header* OP_ CHECKBLOCKVERIFY

[0292] 如果将难度(nBits)设置为最小值,则数据可以是公共区块链中一个区块的区块头,但是也可以是完全不相关的区块,因为该数据不需要大量计算能力以最小的难度生成区块。为了防止这种做法,锁定脚本创建者应使对锁定脚本的约束足够困难,以使创建模仿公共区块链的替代区块链所需的计算能力的成本远大于可以解锁的价值。这可以通过增加难度(例如,在表3中所示并结合图5和6所描述的脚本的难度)或通过要求较长的区块头链(例如,在表6中所示的和结合图9和10所描述的脚本中的区块头链)来实现。

[0293] 表11示出了脚本的示例,该脚本用于验证步骤1724中的交易的其他方面,(即,确定是否满足第二约束集合和/或第一约束集合的方面)。在用于验证表11中所示交易的其他方面的示例脚本中,数据<Data 1>被约束为包含交易<Data 3>的区块头,并且该交易已支付给鲍勃的公钥。应当注意,出于说明目的,这里省略了“<Script to Check Output of Data 3is to P2KH of Bob>”脚本。

[0294] 表11

[0295]

类型	脚本
解锁脚本	*// <Data 1> is the block header, <Data 2> is the Merkle branch, and <Data 3> is the transaction* <Data 1> <Data 2> <Data 3>
锁定脚本	*// Duplicate <Data 3> - the transaction* OP_DUP OP_TOALTSTACK *// Duplicate <Data 1> - the block header and verify that it is a block header* <2> OP_PICK OP_ CHECKBLOCKVERIFY *// Check that <Data 3> combined with <Data 2> - the Merkle branch is the Merkle root of <Data 1>* OP_ CHECKBLOCKTXVERIFY *// Check that the transaction is paid to Bob* OP_ FROMALTSTACK <Script to Check Output of Data 3 is to P2KH of Bob>

[0296] 图17以流程图形式示出了根据实施例的示例过程1800,所述示例过程用于在基于脚本的区块链交互中确认区块链的状态。过程1800的部分或全部可以在配置有可执行指令和/或其他数据的一个或多个计算机系统的控制下执行,并且可以实现为在一个或多个处理器上共同执行的可执行指令。可执行指令和/或其他数据可以存储在非暂时性计算机可读存储介质上(例如,计算机程序永久存储在磁、光、或闪存介质上)。示例过程1800可以由节点执行。也就是说,节点可执行示例过程1800,用于确认在结合图17描述的基于脚本的区块链交互中的区块链的状态。这样的节点可以由任何合适的计算设备组成(例如,由数据中心中的服务器组成,由客户端计算设备组成,由计算资源服务提供商的分布式系统中的多个计算设备组成,或由任何合适的电子客户端设备例如结合图1描述的计算设备200组成)。

[0297] 在示例过程1800的步骤1802中,系统接收上文结合图16所描述的<Data1>、<Data 2>和<Data 3>。

[0298] 在示例过程1800的步骤1804中,如以上文结合图16所述,使用OP_CHECKBLOCKCHAINVERIFY来验证<Data 1>是公共区块链的区块。如上所述,OP_CHECKBLOCKCHAINVERIFY是可用于确定是否满足对数据集合(例如,上述第三约束集合)的约束的脚本的示例。

[0299] 在示例过程1800的步骤1806中,确定<Data 1>是否是公共区块链的区块。如果在示例过程1800的步骤1806中确定<Data 1>不是公共区块链的区块,则在示例过程1800的步骤1808中,<Data 3>的交易被验证不在区块头为<Data1>的区块中,并且在示例过程1800的步骤1810中,锁定脚本解锁失败。

[0300] 如果在示例过程1800的步骤1806中确定<Data 1>是公共区块链的区块,则在示例过程1800的步骤1812中,如表9中所示的示例,系统使用OP_CHECKBLOCKTXVERIFY脚本验证<

Data 3>的交易在区块头为<Data 1>的区块中。如上所述,OP_CHECKBLOCKTXVERIFY是脚本的示例,该脚本可用于确定是否满足对数据集合(例如,上述第一约束集合)的约束。在实施例中,OP_CHECKBLOCKTXVERIFY用于验证数据集合包括交易,并验证该交易包括在由区块头标识的区块中。在一个实施例中,使用OP_CHECKBLOCKVERIFY来验证该数据集合包括区块头。

[0301] 在示例过程1800的步骤1814中,确定OP_CHECKBLOCKTXVERIFY是否通过。如果在示例过程1800的步骤1814中确定OP_CHECKBLOCKTXVERIFY没有通过,则在示例过程1800的步骤1816中,<Data 3>的交易不在区块头为<Data 1>的区块中,并且在示例过程1800的步骤1810中,锁定脚本解锁失败。

[0302] 如果在示例过程1800的步骤1814中确定OP_CHECKBLOCKTXVERIFY通过,则在示例过程1800的步骤1818中,系统如本文所述验证交易的一个或多个其他方面(即,验证与数据集合相关联的第二约束集合的约束得以满足。

[0303] 在示例过程1800的步骤1820中,确定交易的一个或多个其他方面是否被验证。如果在示例过程1800的步骤1820中确定交易的一个或多个其他方面未被验证,则在示例过程1800的步骤1810中,锁定脚本解锁失败。

[0304] 如果在示例过程1800的步骤1820中确定交易的一个或多个其他方面被验证,则在示例过程1800的步骤1822中,锁定脚本解锁成功。

[0305] 注意,在图17中示出的示例过程1800中执行的一个或多个操作可以以各种顺序(包括并行地)和组合来执行。

[0306] 图18以示意图形式示出了由实施例解决的在基于脚本的区块链交互中锁定脚本的示例问题1900。在图18中所示的示例问题1900中,先前交易1902中的锁定脚本1906不能直接读取解锁交易1904的字段。

[0307] 如所指出的,在一些实施例中,先前交易1902是包括在区块链中的最新的、确认的交易。同样地,在一些实施例中,解锁交易1904是未来交易,其尚未被确认并且尚未包括在区块链中,该区块链表示尝试转移对由先前交易1902控制的数字资产的至少一部分的控制。

[0308] 如本文所述,在一些实施例中,锁定脚本1906是为了转移输出而通过指定需要满足的条件来阻碍交易的脚本。更具体地,由于区块链系统的验证节点的执行,执行锁定脚本1906用于从执行的解锁脚本接受数据,并基于该数据执行某些操作,并返回结果,该结果表征是否解锁脚本成功执行“解锁”(即,满足其中设置的条件集合)了锁定脚本。在一些实施例中,锁定脚本1906定义了一个或多个数据约束,这些约束必须被满足(例如,被由解锁脚本提供的数据满足),以便成功进行交易的验证。例如,锁定脚本1906可以要求在解锁脚本中提供某些数据以解锁先前交易1902的相关联的数字资产。

[0309] 图19以示意图形式示出了根据实施例的示例环境2000,该示例环境示出了在基于脚本的区块链交互中锁定脚本对数据的访问。图19描述了示例环境2000,该示例环境2000示出了在实施例中,与先前交易2002的锁定脚本2006不同,本发明的解锁脚本2008有权访问解锁交易2004的字段(例如,根据SIGHASH类型确定的解锁交易字段的集合)。在实施例中,先前交易2002与图18的先前交易1902类似。在实施例中,解锁交易2004与图18的解锁交易1904类似。在一些实施例中,锁定脚本2006具有与图18的锁定脚本1906相似的特性。

[0310] 在一些实施例中,解锁脚本2008是写在交易的输入上的可执行脚本,所述交易上的输入试图满足通过锁定脚本写在交易的输出上的一组条件。解锁脚本也称为“scriptSig”。如上所述,解锁脚本2008被设计为提供根据SIGHASH类型确定的解锁交易字段集合作为锁定脚本的输入,从而使锁定脚本有权访问解锁交易的字段。关于串行化交易的内容和结构的更多细节在图20的描述中示出。

[0311] 图20以示意图的形式示出了根据实施例的示例环境2100,在所述示例环境中,在基于脚本的区块链交互中,从串行化交易字段集合中生成签名。如图20所示,串行交易2110(即,以串行化的特定格式的字节表示的交易)包括交易的字段值集合。在2112中,签名者选择SIGHASH类型和数字k。数字k通常是随机数或伪随机数,以掩蔽/保护私钥,因此在本发明中有时被称为“掩蔽数(mask number)”。交易2114的修改后的副本是串行化交易2110的字段集合的示例,所述串行化交易2110的字段集合是根据指定的SIGHASH类型(例如,在示例环境2100中为SIGHASH_NONE+ANYONECANPAY)选择的。签名人对交易2114的修改后的副本进行散列(例如执行OP_HASH256),从而产生消息m2116。签名人然后例如如上文结合图2所描述的使用SIGHASH类型、消息m、签名人的私钥(例如a)和数字k创建签名。

[0312] 表12示出了典型的锁定和解锁脚本的示例,其中使用OP_CHECKSIG操作码根据实体A的公钥来检查解锁脚本中指定的实体A的假定签名。

[0313] 表12

[0314]

类型	脚本
解锁脚本	<Sig A>
锁定脚本	<PubK A>OP_CHECKSIG

[0315] 因此,在理解了图20的签名生成的情况下,表13示出了如果示例环境2100中示出的过程的一部分被移动到解锁脚本内,则解锁和锁定脚本的示例。

[0316] 表13

[0317]

类型	脚本
解锁脚本锁定脚本	<SIGHASH Type><m><a><k>OP_GENSIG<PubK A>OP_CHECKSIG

[0318] 可以进一步修改解锁脚本,以包括计算解锁脚本中的消息m的步骤,如表14所示。

[0319] 表14

[0320]

类型	脚本
解锁脚本	<SIGHASH Type> < Serialized Set of Unlocking Tx Fields> OP_HASH256 <a> <k> OP_GENSIG
锁定脚本	<PubK A> OP_CHECKSIG

[0321] 但是请注意,除了SIGHASH类型和根据SIGHASH类型确定的解锁交易字段集合外,该过程的其他操作可以移至锁定脚本,如表15所示。

[0322] 表15

[0323]

类型	脚本
解锁脚本	<SIGHASH Type><Serialized Set of Unlocking Tx Fields>

锁定脚本	OP_HASH256<a><k>OP_GENSIG<PubK A>OP_CHECKSIG

[0324] 因此,通过在锁定脚本内移动“OP_HASH256<a><k>OP_GENSIG”的操作,使得解锁脚本包含SIGHASH类型和根据SIGHASH类型确定的解锁交易字段集合,从而使解锁脚本有效。

[0325] 这样就使得串行化的先前交易被注入到锁定脚本中。表16说明了锁定和解锁脚本如何执行示例环境2100的每个步骤过程并验证交易。

[0326] 表16

[0327]

类型	脚本
解锁脚本	<SIGHASH Type> <Serialized Set of Unlocking Tx Fields>
锁定脚本	*// Compute HASH256 the Message* OP_HASH256 *// Generate a Signature using SIGHASH Type, Message, Private Key and k* <a> <k> OP_GENSIG *// Perform Signature Verification* <PubK A> OP_CHECKSIG

[0328] 但是请注意,任何提供SIGHASH类型和根据SIGHASH类型确定的解锁交易字段集合的实体都可以接收交易输出。这可能成为本文所述方法的有用特征。例如,智能合约的实施可以允许将合约实施为区块链交易,从而释放合约的创建者和/或合约的任何其他方,以确保履行合约而无需要求各方信任彼此,且无需要求各方管理或监视合约。在实施例中,需要与合约进行一些交互。在实施例中,在不与合约交互的情况下自动执行合约。

[0329] 如上所述,任何提供SIGHASH类型和根据SIGHASH类型确定的解锁交易字段集合的实体都可以接收交易输出,所以智能合约可以自动执行(或部分自动执行)。在实施例中,该属性可以通过对解锁交易施加附加约束来限制,使得只有提供SIGHASH类型和根据SIGHASH类型确定的解锁交易字段集合的实体可以接收交易输出。另外,可以对智能合约进行编码,以使得提供SIGHASH类型和根据SIGHASH类型确定的解锁交易字段集合的实体只能以特定方式获得交易输出。

[0330] 在实施例中,智能合约包括用作计算机系统可检测到的信号的值,以使其他计算机系统验证交易。也就是说,通过提供数字资产的至少一部分(该数字资产依据执行与智能合约相关联的交易的验证的情况获得),可以促使验证实体对交易进行验证。例如,可以创建与1.0的数字资产相关联的智能合约,以使0.9的数字资产到达指定的接收者,而其余0.1的数字资产可以由验证实体声明。以这种方式执行智能合约。可以想到,影响实体以获得交易输出和/或执行智能合约的其他方法可以视为是在本发明的范围内。

[0331] 如本文中所使用的,智能合约具有自动执行属性和自我执行属性。智能合约的自动执行属性是两个要素的组合。智能合约的自动执行属性的第一要素是任何实体都可以有效地解锁锁定脚本。当任何实体可以有效地解锁锁定脚本时,任何实体都可以创建引用UTXO和/或锁定脚本的有效解锁交易。这是因为没有约束条件要求实体证明自己知道某些

秘密(例如,私钥或将产生特定散列的值)。但是,应注意,智能合约的自动执行属性的第一要素限于本文所述。智能合约的自动执行属性的第二要素是智能合约是自动的,因为可能有诱因促使其他方解锁锁定脚本。智能合约的自我执行属性也是两个要素的组合。智能合约的自我执行属性的第一要素是可以将约束写在串行化的解锁交易字段上,然后将这些约束反映在解锁交易中。智能合约的自我执行属性的第二要素是锁定脚本中的约束由区块链网络执行。

[0332] 图21以流程图的形式示出了根据实施例的示例过程2200,所述示例过程用于在基于脚本的区块链交互中引起串行化解锁交易字段集合的注入。示例过程2200中的部分或全部可以在配置有可执行指令和/或其他数据的一个或多个计算机系统的控制下执行,并且可以被实现为在一个或多个处理器上共同执行的可执行指令。可执行指令和/或其他数据可以存储在非暂时性计算机可读存储介质上(例如,永久存储在磁、光或闪存介质上的计算机程序)。示例过程2200可以由节点执行。也就是说,节点可执行示例过程2200,用于在结合图21描述的基于脚本的区块链交互中引起串行化的解锁交易字段的注入。这样的节点可以由任何合适的计算设备组成(例如,由数据中心中的服务器组成,由客户端计算设备组成,由计算资源服务提供商的分布式系统中的多个计算设备组成,或由任何合适的电子客户端设备例如结合图2描述的计算设备200组成)。

[0333] 示例过程2200包括串行化的操作,其中,执行示例过程2200的系统结合图20中描述的方式执行未验证交易的解锁脚本和锁定脚本,因此而获得SIGHASH类型和交易字段值集合,生成签名,并验证签名。

[0334] 在示例过程2200的步骤2202中,系统从寻求数字资产控制权转移的实体获得未验证的交易。未验证的交易包括锁定脚本和解锁脚本。解锁脚本由系统在执行锁定脚本之前执行。解锁脚本可能类似于上文表15和表16中所示的解锁脚本,并且表征SIGHASH类型和未验证交易的串行化字段。在示例过程2200的步骤2204中,执行示例过程2200的系统获得所表征的SIGHASH类型,并且在示例过程2200的步骤2206中获得未验证交易的串行化交易字段值。在实施例中,在成功完成解锁脚本的执行时,系统开始使用主堆栈的状态(以及在一些实施方式中为替代堆栈的状态)执行锁定脚本。锁定脚本可与上文表15和表16中所示的锁定脚本类似。

[0335] 在示例过程2200的步骤2208中,根据锁定脚本,系统至少使用SIGHASH类型和由于执行解锁脚本而写在主堆栈上的交易字段值集合和与公私钥对关联的私钥来生成签名。在示例过程2200的步骤2210中,根据锁定脚本,系统根据密钥对的公钥成功地验证了签名。这样,通过解锁脚本使交易字段集被注入到锁定脚本中。

[0336] 注意,在图21中示出的示例过程2200中执行的一个或多个操作可以以各种顺序(包括并行地)和组合执行。

[0337] 图22以示意图形式示出了由实施例解决的在基于脚本的区块链交互中锁定脚本的示例问题。在图22中示出的示例问题2300中,锁定脚本2306不能读取其所嵌入的交易的输入,因此,不能直接读取先前交易的字段。

[0338] 在一些实施例中,交易2304与本文所述的解锁交易类似,其表示尝试转移对由先前交易控制的数字资产的至少一部分的控制。如上文所述,在一些实施例中,与本文所述的锁定脚本类似,锁定脚本2306是为了转移输出而通过指定需要满足的条件来阻碍交易的脚

本。

[0339] 图23以示意图的形式示出了根据实施例的示例环境2400,在所述环境中,在基于脚本的区块链交互中引起先前的串行化交易的注入。图23中所示的示例环境2400示出了具有解锁脚本2408的解锁交易2404,解锁脚本2408利用了来自先前交易2402A-02B的输出。可以看出,解锁脚本使已串行化的先前交易被检索。在一些示例中,串行化的先前交易是指先前交易的字段值集合的未修改版本。

[0340] 图23所示的实施例可以通过观察到交易ID是串行化交易的双SHA256并且交易ID与交易一对一映射来理解。因此,通过应用表17中所示的约束,可以使交易注入到解锁脚本中。

[0341] 表17

[0342]

类型	脚本
解锁脚本	<Serialized Tx>
锁定脚本	OP_HASH256<Tx ID>OP_EQUAL

[0343] 本发明的实施例通过不仅引起任何串行化交易的注入,而且还通过引起在解锁交易2404的输入中引用的一个或多个串行化的先前交易(例如,先前交易2402A-02B)的注入,进一步改善了这种注入。如上文参考图20所述,可以使根据SIGHASH类型确定的解锁交易字段集合经由解锁脚本2406注入到锁定脚本中。图24示出了根据指定的SIGHASH类型在串行化交易中包括哪些字段。

[0344] 图24以示意图的形式示出了根据实施例的示例环境2500,其中,在基于脚本的区块链交互中,所述字段集合基于签名散列类型而可用。但是,请注意,图24的目的在于说明,在各种实施例中有更多比图24中描述的SIGHASH类型。如图24所示,在具有不同的SIGHASH类型的情况下,在根据SIGHASH类型确定的解锁交易字段集合中包括不同的先前交易ID集合(注意,“散列”字段以高位编址格式表示交易ID)。在一些实施例中,不管指定了哪种SIGHASH类型,嵌入锁定脚本的先前交易的交易ID总是可用的。

[0345] 因此,可以使某些字段在解锁交易字段集合中显示,所述解锁交易字段集合通过以下方式约束SIGHASH类型来确定:首先,复制SIGHASH类型。然后,将SIGHASH类型压入堆栈(例如SIGHASH_ALL)。最后,调用OP_EQUALVERIFY。从图24中可以看出,SIGHASH_ALL(如所示解锁交易2504A)包括先前交易“y”和“z”的散列,而且如所表示的,代表交易ID。相反,SIGHASH_ALL|ANYONECANPAY(如所示解锁交易2504B)仅包括紧邻的先前交易“y”的散列。在一些实施例中,解锁交易2504A-04B与图18的解锁交易1904类似。图18表示尝试转移对由先前交易(例如,交易y和/或x)控制的数字资产的至少一部分的控制。可以通过使用子串操作码解析串行化的交易直到得到期望的字段来完成提取交易ID的操作,如图25所示。

[0346] 图25以示意图形式示出了根据实施例的示例环境2600,其中,在基于脚本的区块链交互中从串行化交易中提取交易标识符。在图25中所示的示例环境2600中,由高位编址格式的交易ID组成的散列2620是串行化交易2610的子串,串行化交易2610的子串可以通过使用子串操作码在脚本中提取。在一些实施例中,首先解析串行化交易1020来在可以提取交易ID前识别散列2620的位置,因为一些字段(例如,#vin,scriptSigLen,scriptSig,scriptPubKeyLen和scriptPubKey)的字节长度是可变的。然而,可以想到,在字段具有固定长度的实施中,解析可能不是必需的。

[0347] 因此,在一些实施例中,通过构建锁定脚本,该锁定脚本以参照图21描述的方式在解锁交易中强制注入引用的特定串行化的先前交易,可以使先前的交易ID可访问。例如,首先确定SIGHASH类型的约束,提取交易ID,复制串行化的先前交易,并按照从解锁交易串行化字段集合(根据SIGHASH类型)提取的交易ID对串行化的先前交易进行检查。注意,可以执行此过程以强制注入多个不同的串行化先前交易。作为示例,表18提供了一个脚本,该脚本引起与交易X的输入相对应的先前交易的注入。表18中所示的脚本是OP_PREVTXINJECTION脚本的一个示例,该脚本引起与解锁脚本的输入X相对应的先前交易的注入。

[0348] 表18

[0349]

类型	脚本
解锁脚本	<Serialized Previous Tx (for Tx ID X)> <SIGHASH Type> <Serialized Set of Unlocking Tx Fields>
锁定脚本	<X> *// OP_PREVTXINJECTION Below* *// Push <X> to Alt Stack* OP_TOALTSTACK *// Causes Injection of Unlocking Tx (consumes SIGHASH Type)* OP_SWAP OP_*OVER* OP_HASH256 <a><k> OP_GENSIG <PubK A> OP_CHECKSIGVERIFY *// Extract Tx ID <X> from Unlocking Tx* OP_FROMALTSTACK <Script to Extract Tx ID X> *// Create Tx ID from Serialize Previous Tx* OP_SWAP OP_HASH256 *// Check the Tx IDs are the same* OP_EQUAL

[0350] 但是,注意,任何提供SIGHASH类型、根据SIGHASH类型确定的解锁交易字段集合、以及串行化的先前交易的实体都可以获取如本文所述的交易输出。

[0351] 作为另一个示例,表19提供了一个脚本,该脚本引起与正被签名的输入相对应的先前交易的注入。表19中显示的脚本是OP_SELFTXINJECTION脚本的示例,该脚本引起与正被签名的输入相对应的先前交易的注入。

[0352] 表19

[0353]

类型	脚本
解锁脚本	<Serialized Previous Tx (for input being signed)> <SIGHASH Type> <Serialized Set of Unlocking Tx Fields>
锁定脚本	*// Make sure SIGHASH Type is SIGHASH_X | ANYONECANPAY (i.e., it does not matter what SIGHASH type as long as it is combined with SIGHASH_ANYONECANPAY)* OP_OVER <SIGHASH_ANYONECANPAY> OP_AND OP_VERIFY *// Unlocking Tx only has 1 Tx ID, which corresponds to Prev Tx of this Locking Script* <0> OP_PREVTXINJECTION

[0354] 注意,提供SIGHASH类型、根据SIGHASH类型确定的解锁交易字段集合、以及串行化的先前交易的任何实体都可以接收如本文所述的交易输出。

[0355] 图26以流程图形式示出了根据实施例的示例过程2700,在基于脚本的区块链交互中,所述示例过程用于引起先前串行化交易的注入。示例过程2700中的部分或全部可以在配置有可执行指令和/或其他数据的一个或多个计算机系统的控制下执行,并且可以被实现为在一个或多个处理器上共同执行的可执行指令。可执行指令和/或其他数据可以存储在非暂时性计算机可读存储介质上(例如,永久存储在磁、光或闪存介质上的计算机程序)。示例过程2700可以由节点执行。也就是说,节点可执行示例过程2700,用于使得在结合图26描述的基于脚本的区块链交互中注入串行化的先前交易。这样的节点可以由任何合适的计算设备组成(例如,由数据中心中的服务器组成,由客户端计算设备组成,由计算资源服务提供商的分布式系统中的多个计算设备组成,或由任何合适的电子客户端设备例如结合图1描述的计算设备200组成)。

[0356] 示例过程2700包括串行化的操作,其中,执行示例过程2700的系统对未验证交易的解锁脚本和锁定脚本结合执行,从而获得与从未验证交易中提取的交易ID相对应的串行化的先前交易。

[0357] 在示例过程2700的步骤2702中,系统从寻求转移对数字资产的控制的实体获得未验证的交易。未验证的交易包括锁定脚本和解锁脚本。锁定脚本和解锁脚本可与表18和19中所示的锁定脚本和解锁脚本类似。也就是说,锁定脚本包括一组指令,这些指令(例如通过解锁脚本)以输入值的形式存储在主堆栈和备用堆栈中。如果指令集的执行评估为真,则指令集的执行将成功验证未验证的交易。因此,解锁脚本在锁定脚本之前执行,从而在主堆栈和备用堆栈中设置要由锁定脚本使用的值。图26的实施例中的未验证交易的解锁脚本示出串行化的先前交易、SIGHASH类型、和串行化的解锁交易字段集合。

[0358] 由于执行锁定脚本后再执行解锁脚本,在示例过程2700的步骤2704中,系统获得串行化的先前交易、SIGHASH类型、和在解锁脚本中指定的串行化的解锁交易字段集合。在示例过程2700的步骤2706中,系统基于基于SIGHASH的未验证交易的字段值的串行化集合来签名。注意,SIGHASH类型影响在签名生成中使用哪些字段,并且所需的SIGHASH可能取决于所寻求的特定先前交易。例如,表18说明了用于提取先前交易的锁定脚本和解锁脚本,但

不一定提取包含相同锁定脚本的先前交易,表18利用了SIGHASH类型的SIGHASH_ALL。如图24所示,SIGHASH_ALL类型使得能够检索解锁交易的其他先前交易的输入。相反,表19说明了用于提取具有相同锁定脚本的先前交易(即,与正被签名的输入相对应的先前交易)的锁定脚本和解锁脚本,表19利用了SIGHASH类型的SIGHASH_ALL|ANYONECANPAY。如图24所示,SIGHASH类型的SIGHASH_ALL|ANYONECANPAY除去了具有正被签名的输入的先前交易之外的其他先前交易的输入。

[0359] 在示例过程2700的步骤2708中,系统验证先前生成的签名。在示例过程2700的步骤2710中,系统通过执行所获得的串行化的先前交易的双SHA256来产生应当与解锁交易的交易ID匹配的值(即,字段值的串行化集合)。使用子串操作码,在示例过程2700的步骤2712中,系统提取字段值的串行化集合的交易ID,并且在示例过程2700的步骤2714中,系统确定由串行化的先前交易的双SHA256产生的交易ID与串行化的字段值集合的交易ID是否匹配。注意,在各种实施例中,匹配不一定需要相等。例如,如果两个值不相等,但在数学上等同,则可以匹配。又例如,如果两个值对应于公共对象(例如,值)或以某种预定方式互补和/或两个值满足一个或多个匹配标准,则两个值可以匹配。通常,可以使用确定是否存在匹配的任何方式。

[0360] 注意,在图26中示出的示例过程2700中执行的一个或多个操作可以被以各种顺序(包括并行地)和组合来执行。

[0361] 图27以示意图形式示出了根据实施例的示例环境2800,在所述环境中,区块链的状态用于基于脚本的区块链交互。如本文所述,区块头是区块链的区块的一部分,所述区块链包含有关该区块的数据。

[0362] 在步骤2804中,提供区块头2802作为用于本文所述的解锁脚本的数据(表示为<Data 1>)。在实施例中,例如上文结合图5至图6描述的OP_CHECKBLOCKVERIFY脚本用于验证<Data 1>是区块头。在步骤2808中,提供默克尔分支2806作为用于如上文结合图12至图13描述的解锁脚本的数据(表示为<Data 2>)。在步骤2812中,提供交易2810作为用于解锁脚本的数据(表示为<Data 3>)。在实施例中,使用如上文结合图12和13所述的OP_CALCMERKLEROOT脚本从默克尔根<Data 2>和交易<Data 3>计算出默克尔根。

[0363] 在步骤2814中,执行OP_CHECKBLOCKTXVERIFY脚本以验证从<Data 1>提取的HashMerkleRoot与从OP_CALCMERKLEROOT脚本计算的默克尔根相同,所述OP_CALCMERKLEROOT脚本是对上文表9所示的和结合图12、13和16描述的<Data 2>和<Data 3>执行的。

[0364] 在步骤2816中,执行OP_SELFTXINJECTION脚本以验证<Data 3>的交易的解锁脚本包括串行化交易,该串行化交易包含上表18和19中所示的和结合26、27和28所描述的锁定脚本。

[0365] 在步骤2818中,如上文所述,如果2814的OP_CHECKBLOCKTXVERIFY脚本和步骤2816的OP_SELFTXINJECTION成功,则已知具有区块头<Data 1>的区块包括交易<Data 3>,<Data 3>的解锁脚本包括锁定脚本的串行化交易,且锁定脚本可以访问区块链状态。这样的锁定脚本有权访问的信息包括但不限于时间(即,来自一个或多个区块头的nTime字段),伪随机数(即,来自nNonce或HashMerkleRoot,两者都是伪随机数的良好来源)或其他信息,这些信息中的一些或全部可以在锁定脚本中使用。这样的附加信息的示例在图29所示的示例中示

出。

[0366] 在实施例中,脚本用于验证区块头<Data 1>是作为公共区块链的一部分的区块的区块头,而不是如上所述通过使用验证脚本(例如OP_CHECKBLOCKCHAINVERIFY)专门为模拟该公共区块链的区块而建立的区块。在实施例中,脚本用于验证区块头<Data 1>是作为公共区块链的一部分的区块的区块头,可以将区块链1702或如上文所述完整或部分区块头链提供给所述公共区块链。

[0367] 在实施例中,脚本也可以用来验证如上文结合图16所描述的交易和/或区块链状态的其他方面。如上文所述,用来验证交易的其他方面的脚本可以包括交易的多个方面的验证,包括但不限于受让人,转让人,交易数量或交易的其他方面。此外,应注意,用于验证交易其他方面的脚本可包括区块链的各个方面,区块链状态的各个方面,区块头链的各个方面,区块的各个方面或区块链环境的其他各个方面。例如,用于验证交易其他方面的脚本可以验证区块的nTime,区块中的交易数量,两个区块之间的经过的时间量,最小区块高度或区块链环境的其他方面。还应注意,通过在脚本(用于验证交易的其他方面)的末尾不包括"验证"操作码,可以引入分支条件(例如,如果在区块中有超过1000个交易,则爱丽丝可以解锁;如果区块中的交易不超过1000个,则鲍勃可以解锁)。下文结合图29对分支条件的示例进行更详细地描述。

[0368] 尽管在示例环境2800中未示出,但是在实施例中,用于验证交易的其他方面的附加脚本,例如表11中示出的并且结合图16所描述的脚本,可以作为图27中描述的示例环境2800的一部分来执行。

[0369] 图28以流程图形式示出了根据实施例的示例过程2900,所述示例过程为在基于脚本的区块链交互中使用区块链的状态的示例过程。过程2900的部分或全部可以在配置有可执行指令和/或其他数据的一个或多个计算机系统的控制下执行,并且可以被实现为在一个或多个处理器上共同执行的可执行指令。可执行指令和/或其他数据可以存储在非暂时性计算机可读存储介质上(例如,永久存储在磁、光、或闪存介质上的计算机程序)。示例过程2900可以由节点执行。也就是说,节点可执行示例过程2900,所述示例过程用于在结合图28描述的基于脚本的区块链交互中使用区块链的状态。这样的节点可以由任何合适的计算设备组成(例如,由数据中心中的服务器组成,由客户端计算设备组成,由计算资源服务提供商的分布式系统中的多个计算设备组成,或由任何合适的电子客户端设备例如结合图1描述的计算设备200组成)。

[0370] 在示例过程2900的步骤2902中,系统接收如上文结合图16所描述的<Data 1>,<Data 2>和<Data 3>。

[0371] 在示例过程2900的步骤2904中,如以上文结合图16所述,使用OP_CHECKBLOCKCHAINVERIFY来验证<Data 1>是公共区块链的区块。

[0372] 在示例过程2900的步骤2906中,确定<Data 1>是否是公共区块链的区块。如果在示例过程2900的步骤2906中确定<Data 1>不是公共区块链的区块,则在示例过程2900的步骤2908中,<Data 3>的交易被验证不在具有区块头<Data 1>的区块中,并且在示例过程2900的步骤2910中,锁定脚本不可以访问区块链状态。

[0373] 如果在示例过程2900的步骤2906中确定<Data 1>是公共区块链的区块,则在示例过程2900的步骤2912中,如表9中所示的示例,系统使用OP_CHECKBLOCKTXVERIFY脚本验证<

Data 3>的交易在具有区块头<Data 1>的区块中。

[0374] 在示例过程2900的步骤2914中,确定OP_CHECKBLOCKTXVERIFY是否通过。如果在示例过程2900的步骤2914中确定OP_CHECKBLOCKTXVERIFY没有通过,则在示例过程2900的步骤2916中,<Data 3>的交易不在具有区块头<Data1>的区块中,并且在示例过程2900的步骤2910中,锁定脚本不可以访问区块链状态。

[0375] 如果在示例过程2900的步骤2914中确定OP_CHECKBLOCKTXVERIFY通过,则在示例过程2900的步骤2918中,系统使用OP_SELFTXINJECTION脚本(例如表19中所示的脚本)来验证<Data 3>中的交易的解锁脚本包括锁定脚本的串行化交易。

[0376] 在示例过程2900的步骤2920中,确定<Data 3>中的交易的解锁脚本是否包括锁定脚本的串行化交易。如果在示例过程2900的步骤2920中确定<Data3>中的交易的解锁脚本不包括锁定脚本的串行化交易,则在示例过程2900的步骤2910中,锁定脚本不可以访问区块链状态。

[0377] 如果在示例过程2900的步骤2920中确定<Data 3>中的交易的解锁脚本确实包括锁定脚本的串行化交易,则在示例过程2900的步骤2922中,锁定脚本有权访问区块链状态。

[0378] 注意,在图28中示出的示例过程2900中执行的一个或多个操作可以以各种顺序(包括并行地)和组合来执行。

[0379] 图29以示意图形式示出了根据实施例的在基于脚本的区块链交互中使用区块链的状态的示例实施方式3000。示例实施方式3000示出了本文描述的方法的几个方面。在示例实施方式3000中,基于包含风险投资的交易的区块和紧邻的先前区块之间的时间差,决定对包含风险投资的交易的区块添加到公共区块链的时间做风险投资。

[0380] 使用本文描述的方法,可以确定风险投资交易3002是否在具有区块头3014的区块B_2中。类似地,使用本文描述的方法,可以确定具有区块头3012的区块B_1是否是紧邻的先前区块。最终,可以确定区块头3014的nTime是否在区块头3012的nTime之后十分钟内。

[0381] 表20示出了该风险投资交易3002的示例锁定脚本。本文未示出表20所示的脚本中使用的OP_EXTRACTTIME脚本,但是,可以想到,OP_EXTRACTTIME脚本是用于从区块头中提取时间条目的脚本,该区块头与表3中所示的用来提取nBit的脚本类似。

[0382] 表20

[0383]

类型	脚本
解锁脚本	<Data 0> <Data 1> <Data 2> <Data 3> <Data 4> <Data 5>
锁定脚本	*// Duplicate <Data 3>* <2> OP_PICK OP_TOALTSTACK *// <Data 5> is a Unlocking Tx SIGHASH Byte* *// <Data 4> is SIGHASH Type* *// <Data 3> is Previous Tx Byte* OP_SELFTXINJECTION OP_VERIFY *// Duplicate <Data 1> and <Data 2> and check they are chained block headers* OP_OVER OP_OVER <1> OP_CHECKBLOCKCHAINVERIFY *// Extract T1 from <Data 1>* OP_OVER <OP_EXTRACTTIME> *// Extract T2 from <Data 2>* OP_OVER <OP_EXTRACTTIME> *// Subtract* OP_SUB *// Pull <Data 3> from alt stack, and push the time diff to alt stack* OP_FROMALTSTACK OP_SWAP OP_TOALTSTACK *// Check that <Data 3> (the transaction)* *// combined with <Data 2> (the Merkle* *// branch) is the Merkle root in <Data 1>* OP_CHECKBLOCKTXVERIFY *// Check that T2 - T1 is < 10 minutes* OP_FROMALTSTACK <10> OP_LESSTHAN *// If true, check <Data 0> is Alice's* *// Sig, else check if <Data 0> is Bob's* *// Sig* OP_IF <PubK A> OP_CHECKSIG OP_ELSE <PubK B> OP_CHECKSIG OP_ENDIF

[0384] 因此,说明书和附图应被认为是说明性的而不是限制性的。然而,显而易见的是,在不脱离权利要求中所阐述的本发明的更广泛精神和范围的情况下,可以对其进行各种修改和改变。

[0385] 其他变型在本发明的精神内。因此,尽管所公开的技术易于进行各种修改和替代,

但是其示出的某些实施例在附图中示出并且已经在上文进行了详细描述。然而,应当理解,无意将本发明限制为所公开的一种或多种特定形式,相反,其意图是涵盖所附权利要求书中所定义的落入本发明的精神和范围内的所有修改,替代构造和等同物。

[0386] 在描述所公开的实施例的上下文中(特别是在所附权利要求的上下文中)术语“一个”和“一种”(“a”,“an”,“the”)以及类似指代的使用应解释为涵盖单数和复数,除非本文另有说明或与上下文明显矛盾。除非另有说明,否则术语“包括(comprising”),“具有(having)”,“包括(including)”和“包含(“containing)”应解释为开放式术语(即,意思是“包括但不限于”)。术语“连接”在未修改的情况下指的是物理连接时,应理解为部分或全部包含在内,连接在一起,即使有某些中介物。本文中数值范围的引用仅旨在用作分别指代落入该范围内的每个单独值的简写方法,除非在本文另外指出并且每个单独值都被并入说明书中使得其类似于在本文中被单独叙述。术语“集合”(例如,“一组项目”)或“子集”的使用应被解释为包括一个或多个成员的非空集合,除非上下文另外指出或有矛盾。此外,对应集合的术语“子集”不一定表示对应集合的适当子集,而是该子集和对应集合可以相等,除非上下文另外指出或有矛盾。

[0387] 除非另有明确说明或与上下文明显矛盾,否则应将“A、B、和C中的至少一个”或“A、B和C中的至少一个”形式的连接语言结合上下文一起理解,通常用于表示项目、术语等可以是A或B或C,也可以是A和B和C的集合的任何非空子集。例如,在具有三个项目的集合的说明性示例中,连接语“A、B、和C中的至少一个”和“A、B和C的至少一个”指的是以下集合中的至少一个:{A},{B},{C},{A,B},{A,C},{B,C},{A,B,C}。因此,这种连接语通常不意在表示某些实施例要求存在至少一个A,至少一个B和至少一个C。

[0388] 本文描述的过程的操作可以以任何合适的顺序执行,除非本文另外指出或与上下文明显矛盾。本文描述的过程(或其变型和/或组合),可以通过硬件或其组合,在配置有可执行指令的一个或多个计算机系统的控制下执行,并且可以实现为在一个或多个处理器上共同执行的代码(例如,可执行指令,一个或多个计算机程序或一个或多个应用程序)。该代码可以例如以计算机程序的形式存储在计算机可读存储介质上,该计算机程序包括可由一个或多个处理器执行的多个指令。所述计算机可读存储介质可以是非暂时性的。

[0389] 除非另外要求,否则本文提供的任何和所有示例或示例性语言(例如,“例如”)的使用仅旨在更好地阐明本发明的实施例,并且不对本发明的范围构成限制。说明书中的任何语言都不应解释为表示任何未要求保护的元素对于实施本发明必不可少。

[0390] 本文描述了本发明的实施例,包括发明人已知的用于实施本发明的最佳方式。通过阅读前述说明,那些实施例的变型对于本领域普通技术人员而言将变得显而易见。发明人期望有经验的技术人员适当地采用这样的变型,并且发明人希望以不同于本文具体描述的方式来实践本发明的实施例。因此,本发明的范围包括适用法律所允许的所附权利要求中记载的主题的所有修改和等同物。而且,除非本文另外指出或与上下文明显矛盾,否则上述元件在其所有可能的变型中的任何组合均被本发明的范围涵盖。

[0391] 应注意,在描述本发明的实施例的上下文中,除非另有说明,否则与可执行指令(也称为代码,应用程序,代理等)相关的表达的使用表示指令正在由机器执行,从而使机器执行指定的操作。所述可执行指令执行的操作为:“指令”通常不在没有辅助的情况下执行(例如,数据的传输、计算等)。

[0392] 应当注意,上述实施例说明而不是限制本发明,并且本领域技术人员将能够在不脱离由所附权利要求限定的本发明的范围的情况下设计许多替代实施例。在权利要求中,括号中的任何附图标记不应被解释为对权利要求的限制。词语“包括(comprising)”和“包括(comprises)”等不排除任何权利要求或说明书整体中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(comprises)”是指“包括(includes)或者由……组成(consists of)”,而“包括(comprising)”是指“包括(including)或者由……组成(consisting of)”。元件的单数引用不排除这些元件的复数引用,反之亦然。本发明可以通过包括几个不同元件的硬件以及通过适当编程的计算机来实现。在列举了几个装置的设备权利要求中,这些装置中的几个可以由同一个硬件来实现。在相互不同的从属权利要求中引用某些措施这一事实并不表示这些措施的组合不能有利地使用。

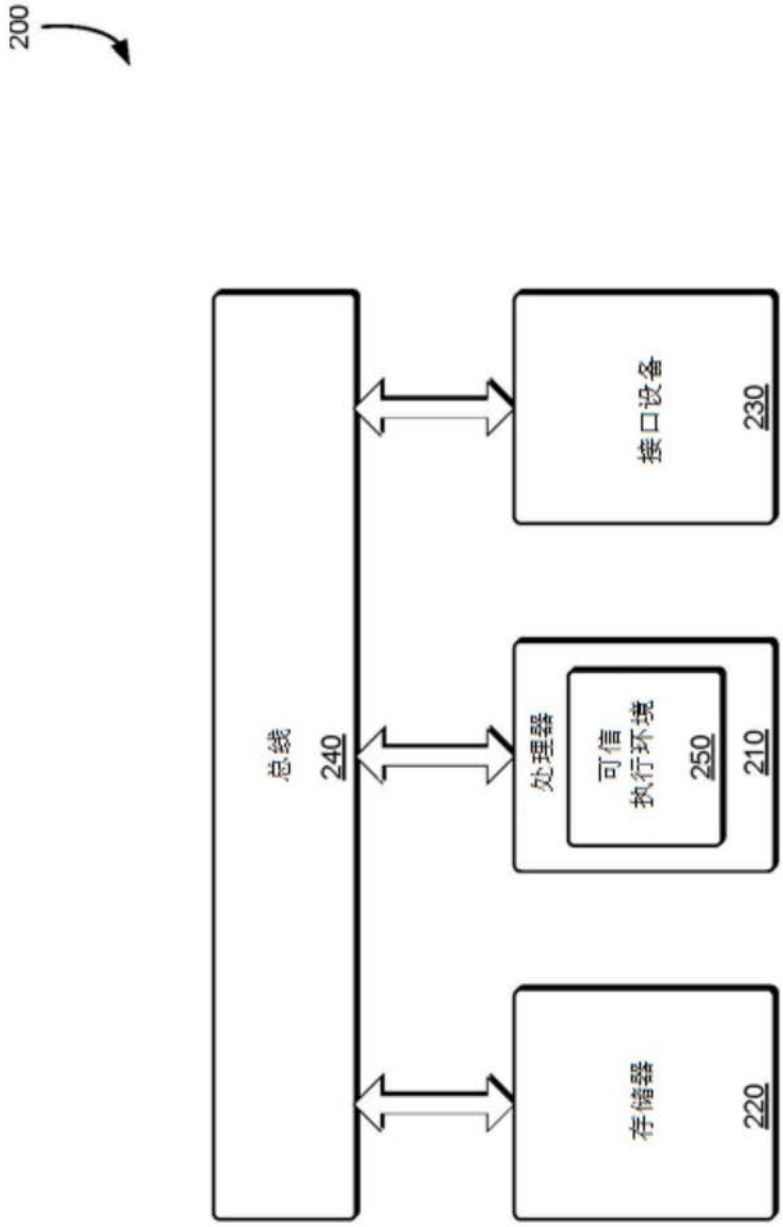


图1

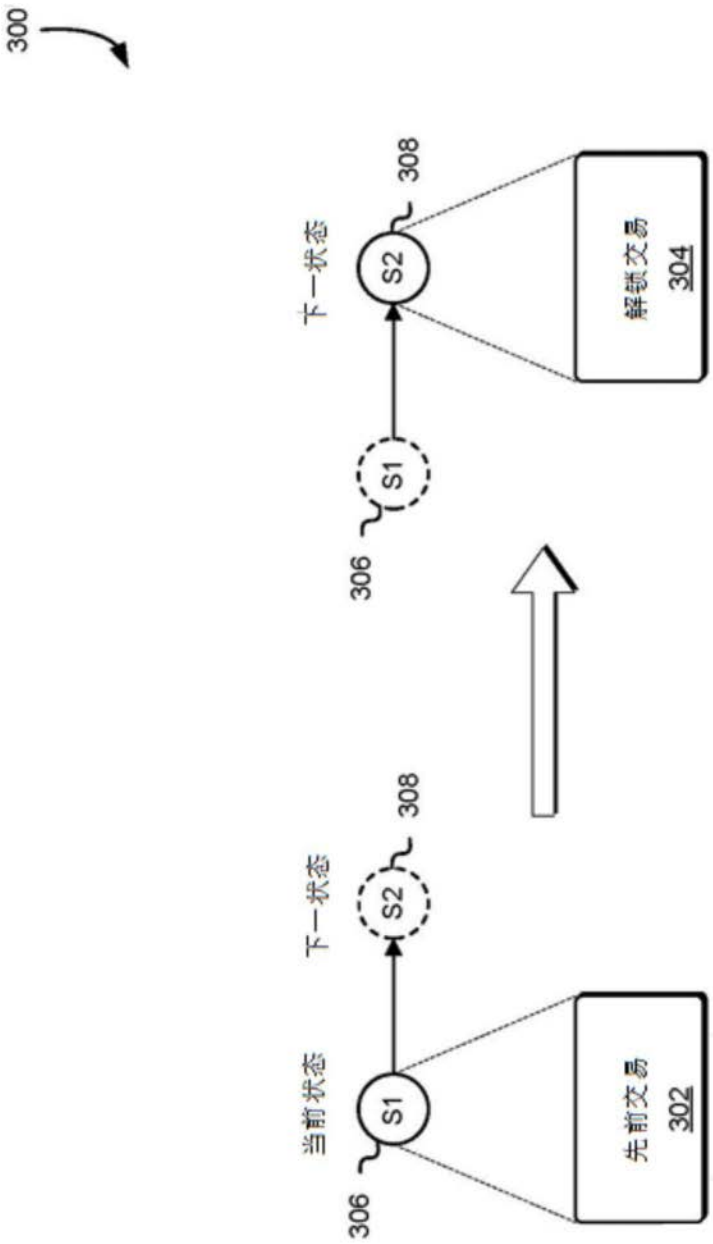


图2

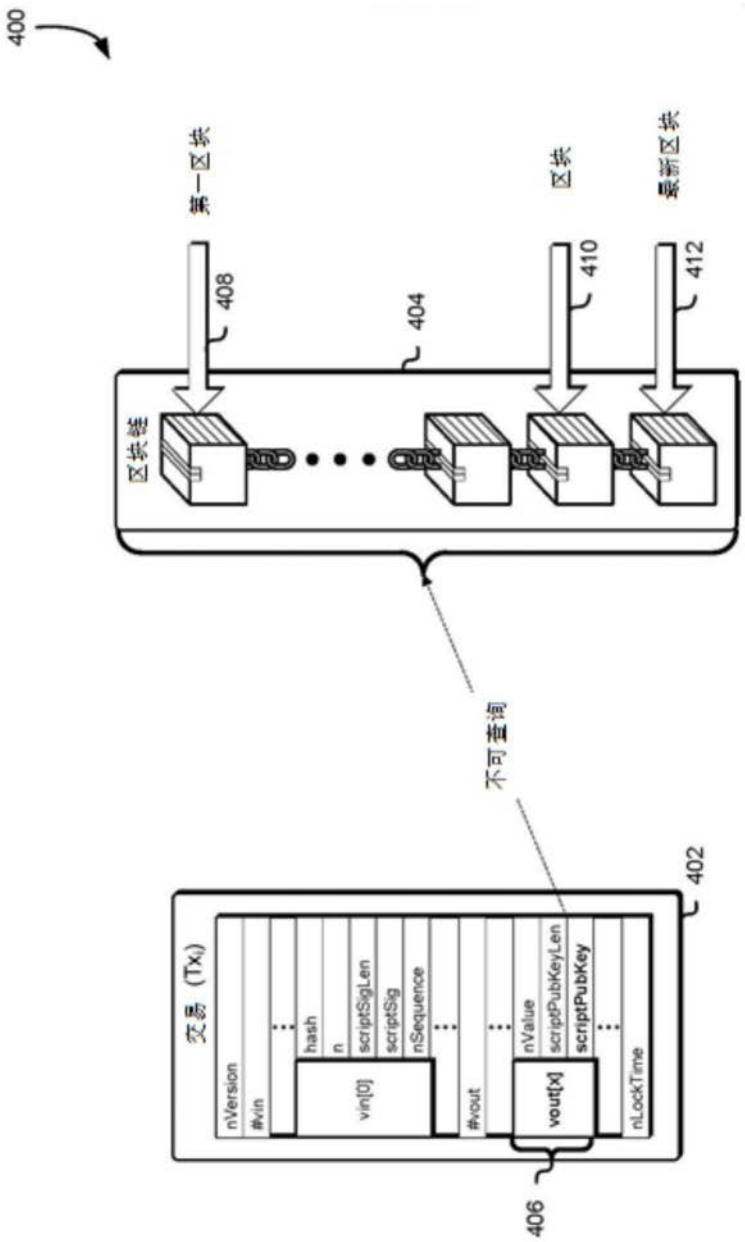

400
第一区块
区块
最新区块
408
404
410
412
区块链
不可查询
402
交易 (Tx)
nVersion
#vin
hash
n
scriptSigLen
scriptSig
nSequence
vin[0]
#vout
nValue
scriptPubKeyLen
scriptPubKey
vout[x]
nLockTime
406

图3

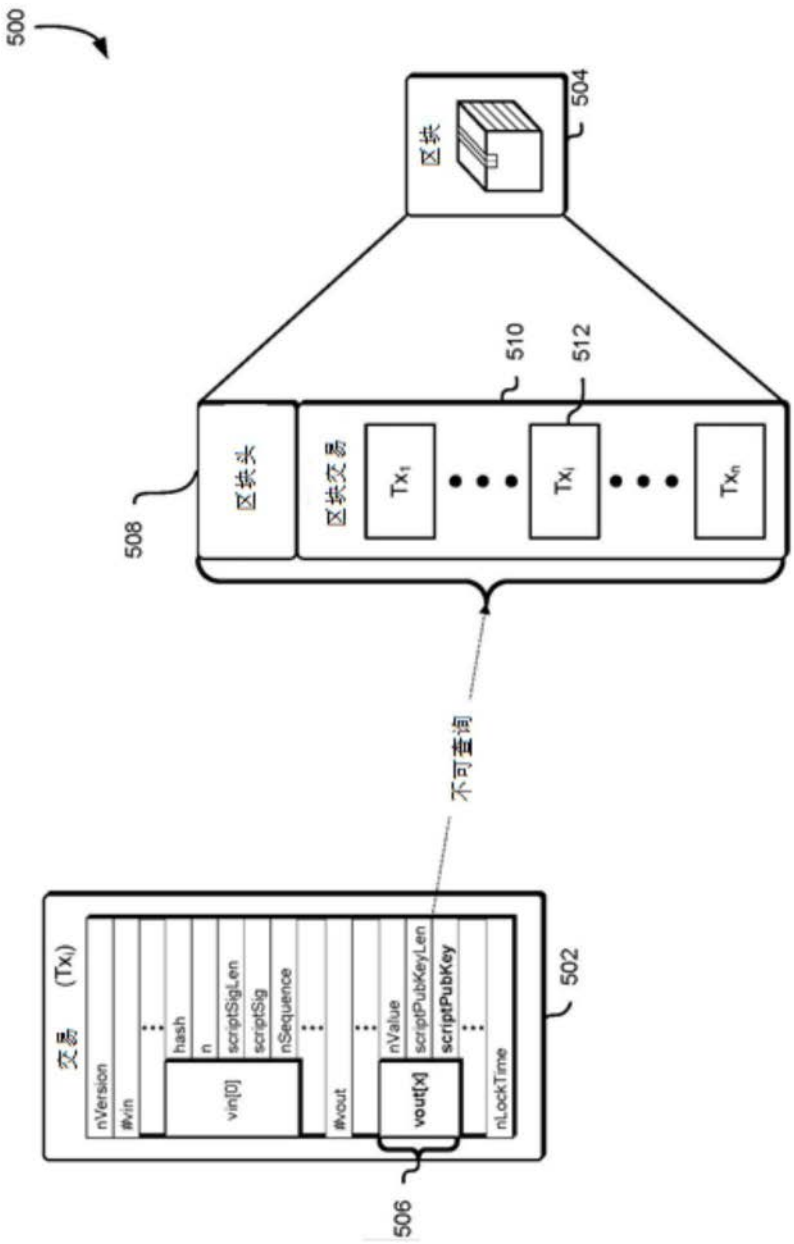

500
区块
504
510
512
区块头
区块交易
508
Tx1
Txi
Txn
不可查询
交易 (Txi)
nVersion
#vin
hash
n
scriptSigLen
scriptSig
nSequence
vin[0]
#vout
nValue
scriptPubKeyLen
scriptPubKey
vout[x]
nLockTime
502
506

图4

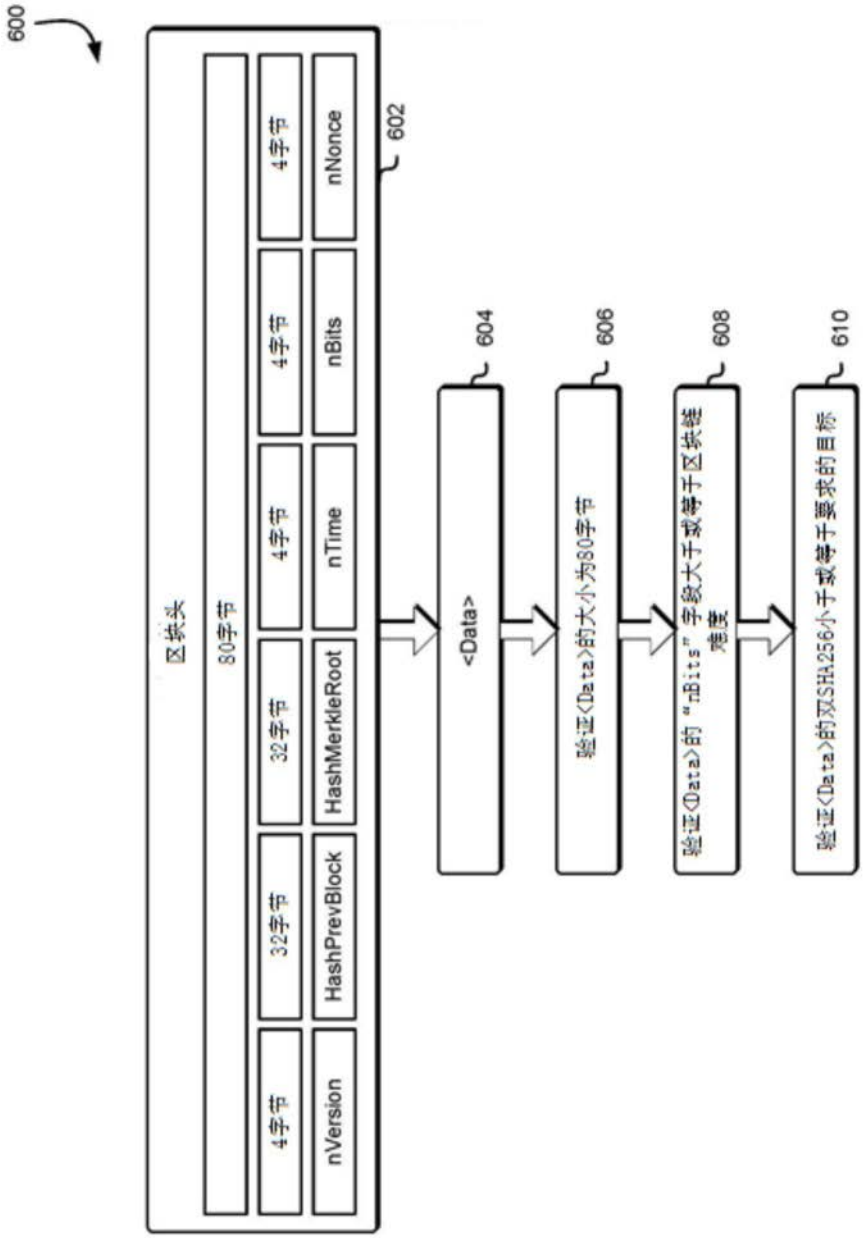

600
602
区块头
80字节
4字节
nVersion
32字节
HashPrevBlock
32字节
HashMerkleRoot
4字节
nTime
4字节
nBits
4字节
nNonce
604
<Data>
606
验证<Data>的大小为80字节
608
验证<Data>的“nBits”字段大于或等于区块链难度
610
验证<Data>的双SHA256小于或等于要求的目标

图5

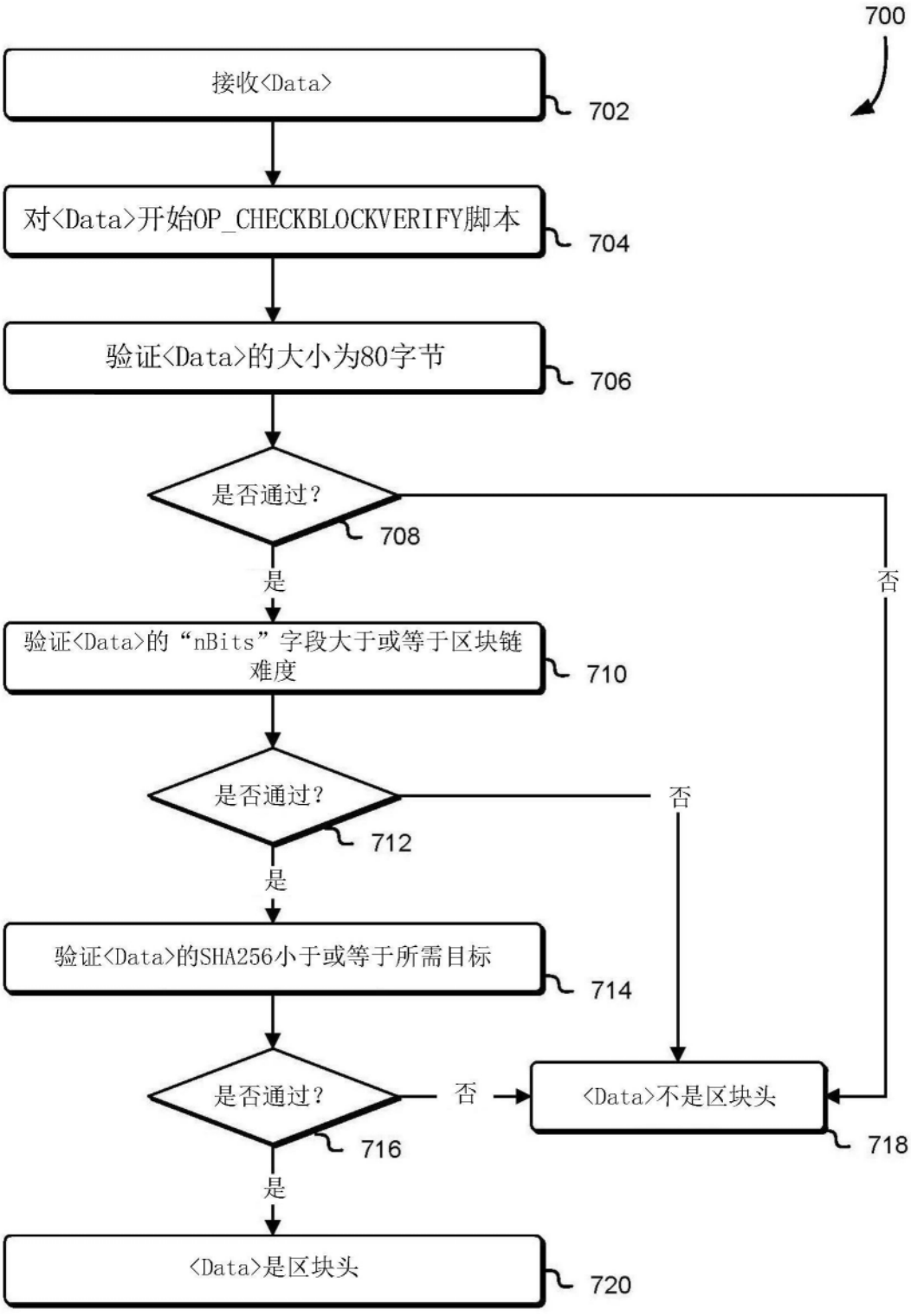

700
接收<Data>
702
对<Data>开始OP_CHECKBLOCKVERIFY脚本
704
验证<Data>的大小为80字节
706
是否通过?
708
是
否
验证<Data>的“nBits”字段大于或等于区块链难度
710
是否通过?
712
否
是
验证<Data>的SHA256小于或等于所需目标
714
是否通过?
716
否
<Data>不是区块头
718
是
<Data>是区块头
720

图6

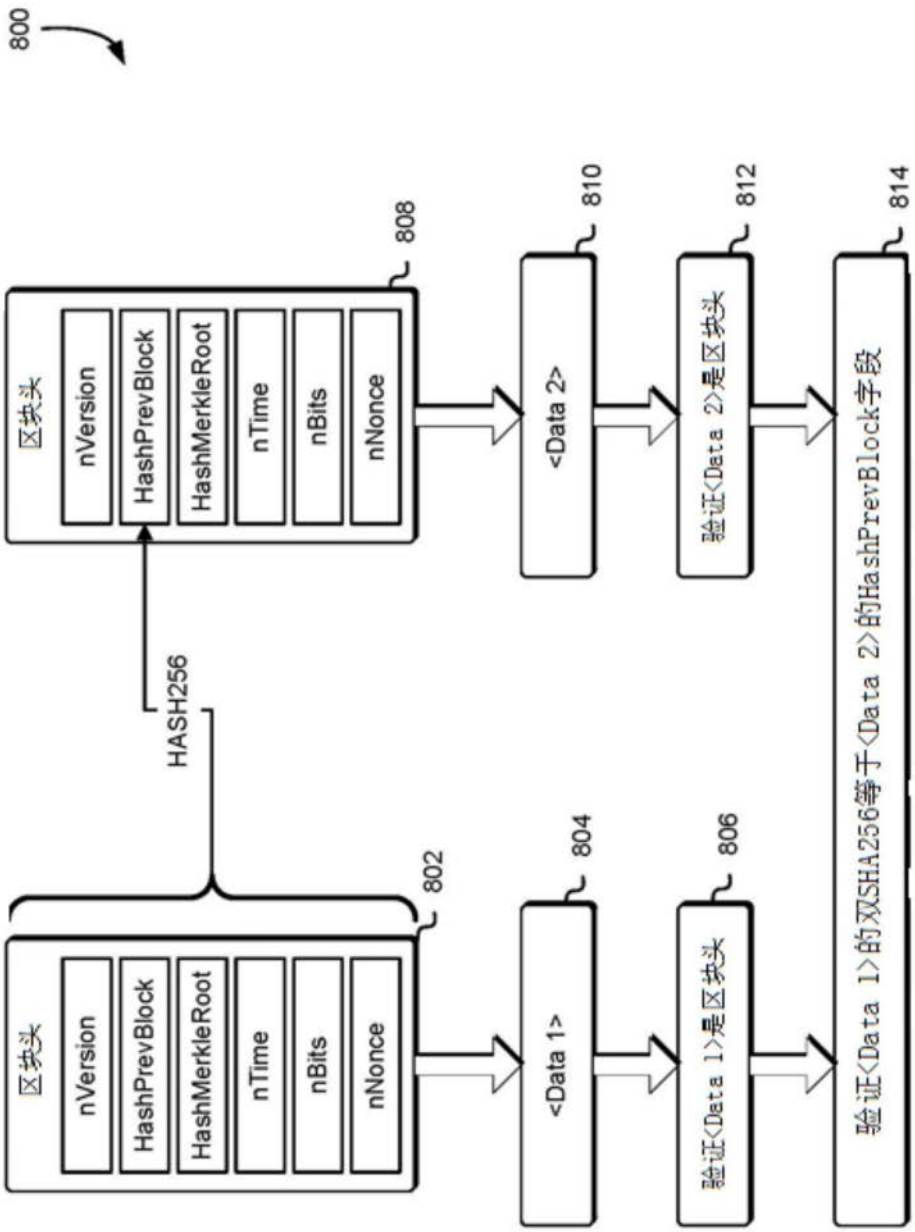

800
区块头
nVersion
HashPrevBlock
HashMerkleRoot
nTime
nBits
nNonce
808
<Data 2>
810
验证<Data 2>是区块头
812
验证<Data 1>的双SHA256等于<Data 2>的HashPrevBlock字段
814
HASH256
802
区块头
nVersion
HashPrevBlock
HashMerkleRoot
nTime
nBits
nNonce
<Data 1>
804
验证<Data 1>是区块头
806

图7

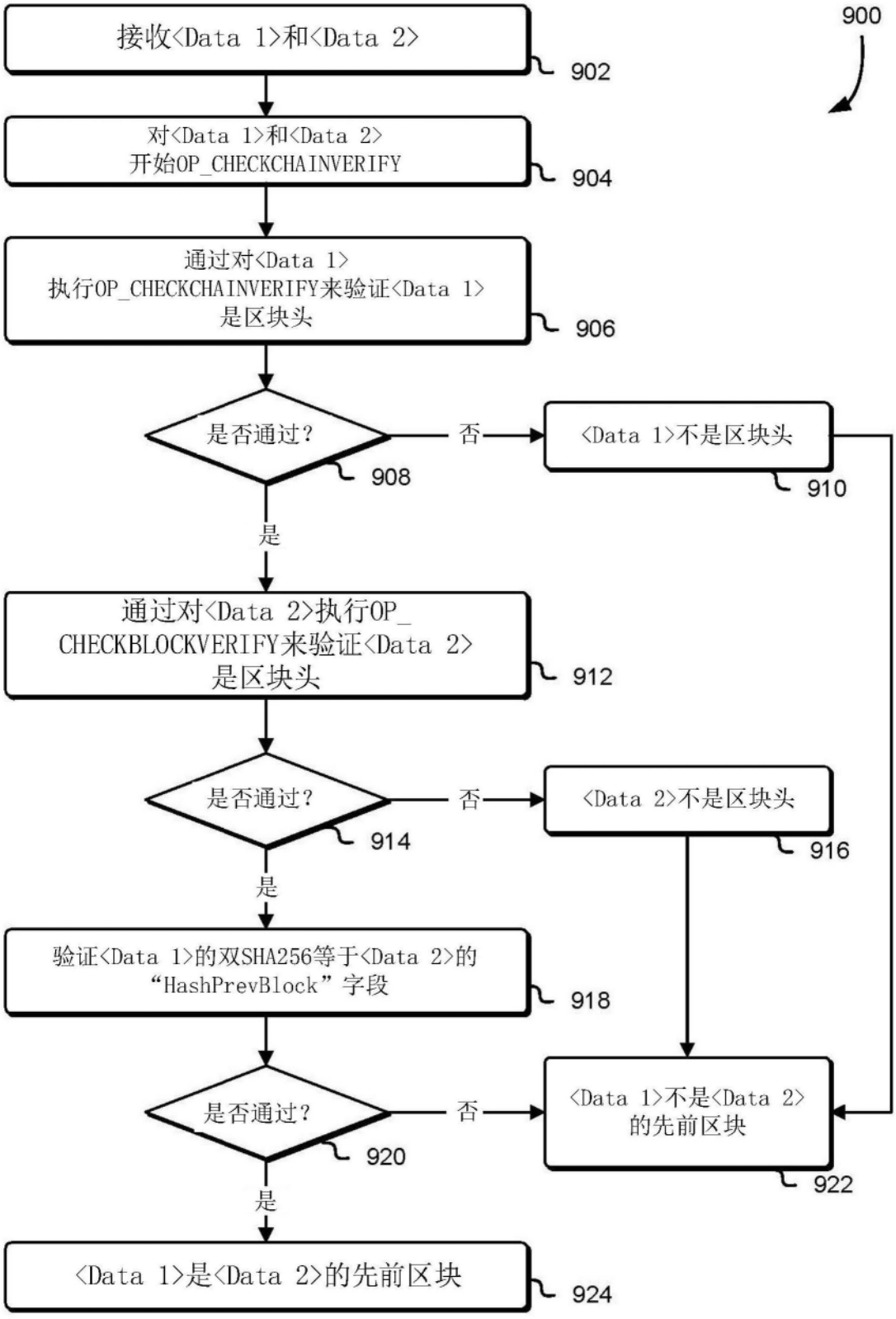

900
接收<Data 1>和<Data 2>
902
对<Data 1>和<Data 2>
开始OP_CHECKCHAINVERIFY
904
通过对<Data 1>
执行OP_CHECKCHAINVERIFY来验证<Data 1>
是区块头
906
是否通过?
908
否
<Data 1>不是区块头
910
是
通过对<Data 2>执行OP_
CHECKBLOCKVERIFY来验证<Data 2>
是区块头
912
是否通过?
914
否
<Data 2>不是区块头
916
是
验证<Data 1>的双SHA256等于<Data 2>的
“HashPrevBlock”字段
918
是否通过?
920
否
<Data 1>不是<Data 2>
的先前区块
922
是
<Data 1>是<Data 2>的先前区块
924

图8

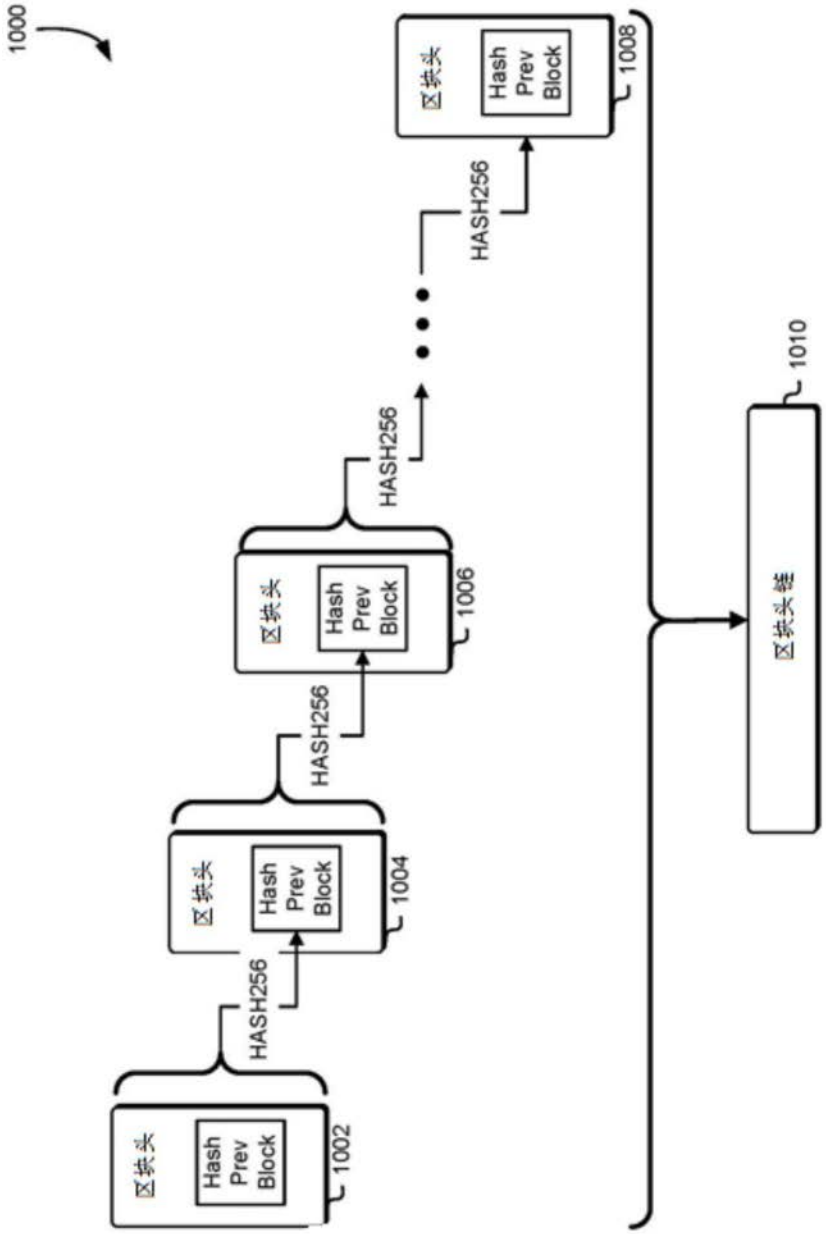

1000
区块头
Hash Prev Block
1002
HASH256
区块头
Hash Prev Block
1004
HASH256
区块头
Hash Prev Block
1006
HASH256
区块头
Hash Prev Block
1008
1010
区块头链

图9

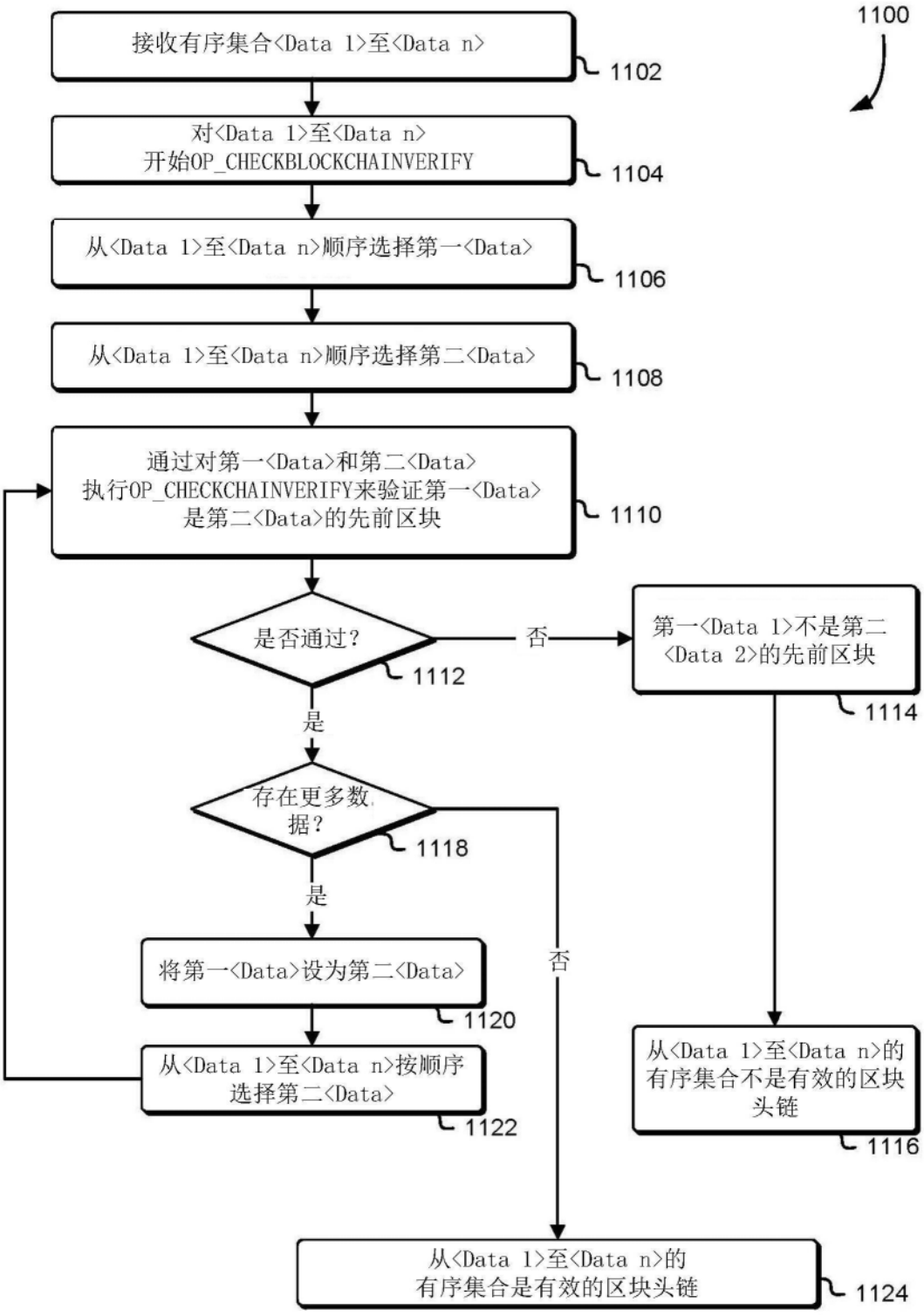

1100
接收有序集合<Data 1>至<Data n>
1102
对<Data 1>至<Data n>
开始OP_CHECKBLOCKCHAINVERIFY
1104
从<Data 1>至<Data n>顺序选择第一<Data>
1106
从<Data 1>至<Data n>顺序选择第二<Data>
1108
通过对第一<Data>和第二<Data>
执行OP_CHECKCHAINVERIFY来验证第一<Data>
是第二<Data>的先前区块
1110
是否通过?
1112
否
第一<Data 1>不是第二
<Data 2>的先前区块
1114
是
存在更多数
据?
1118
是
否
将第一<Data>设为第二<Data>
1120
从<Data 1>至<Data n>按顺序
选择第二<Data>
1122
从<Data 1>至<Data n>的
有序集合不是有效的区块
头链
1116
从<Data 1>至<Data n>的
有序集合是有效的区块头链
1124

图10

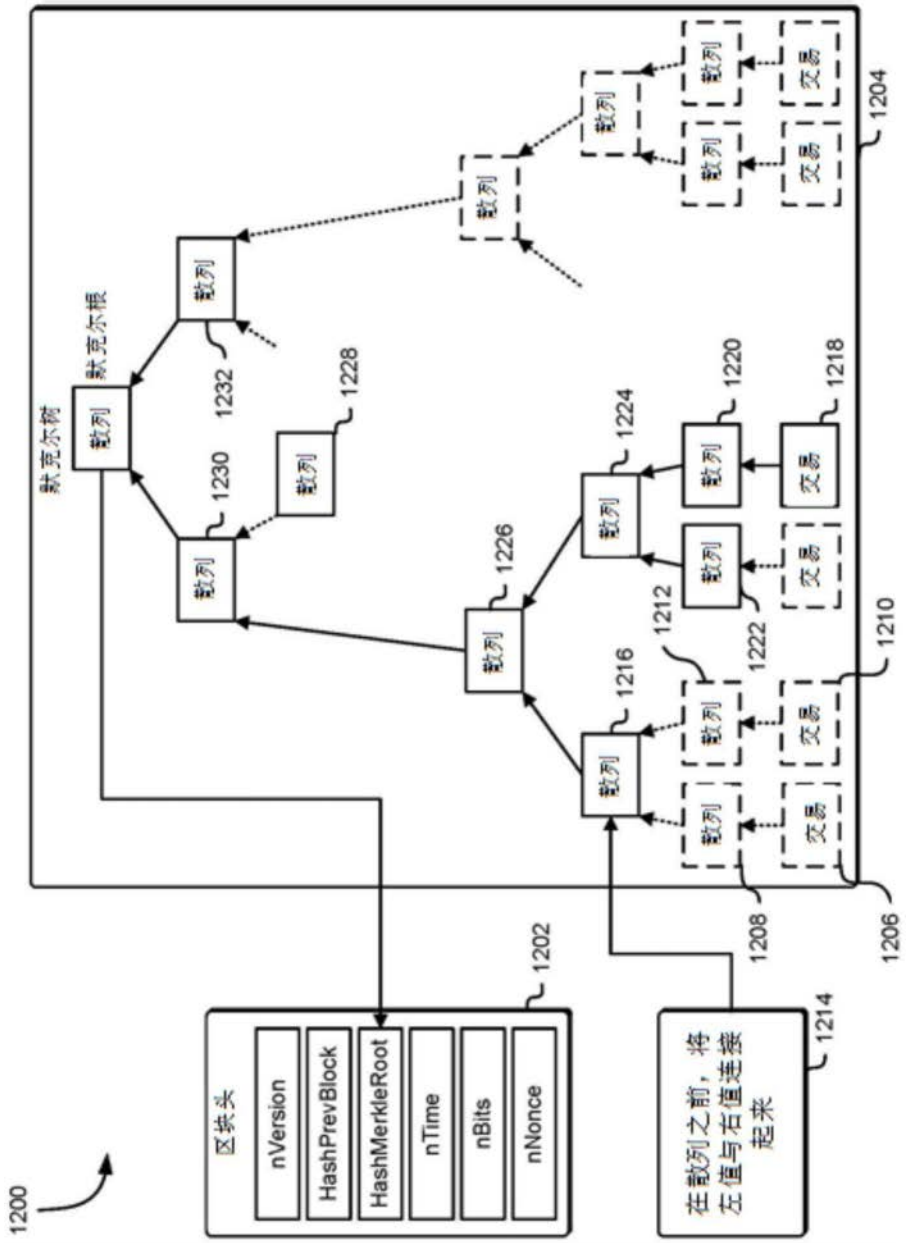

1200
区块头
nVersion
HashPrevBlock
HashMerkleRoot
nTime
nBits
nNonce
1202
在散列之前，将
左值与右值连接
起来
1214
默克尔树
默克尔根
散列
交易
1204
1206
1208
1210
1212
1216
1218
1220
1222
1224
1226
1228
1230
1232

图11

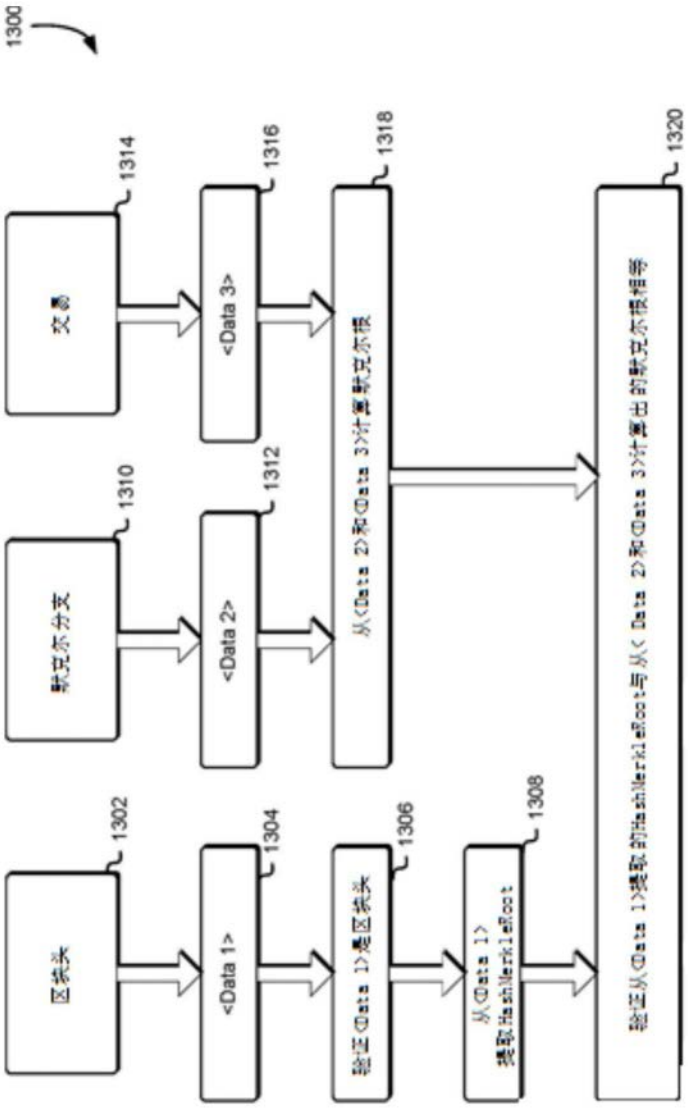

1300
1314
交易
1316
<Data 3>
1318
从<Data 2>和<Data 3>计算默克尔根
1320
验证从<Data 1>提取的blockMerkleRoot与从< Data 2>和<Data 3>计算出的默克尔根相等
1310
默克尔分支
1312
<Data 2>
1302
区块头
1304
<Data 1>
1306
验证<Data 1>是区块头
1308
从<Data 1>
提取blockMerkleRoot

图12

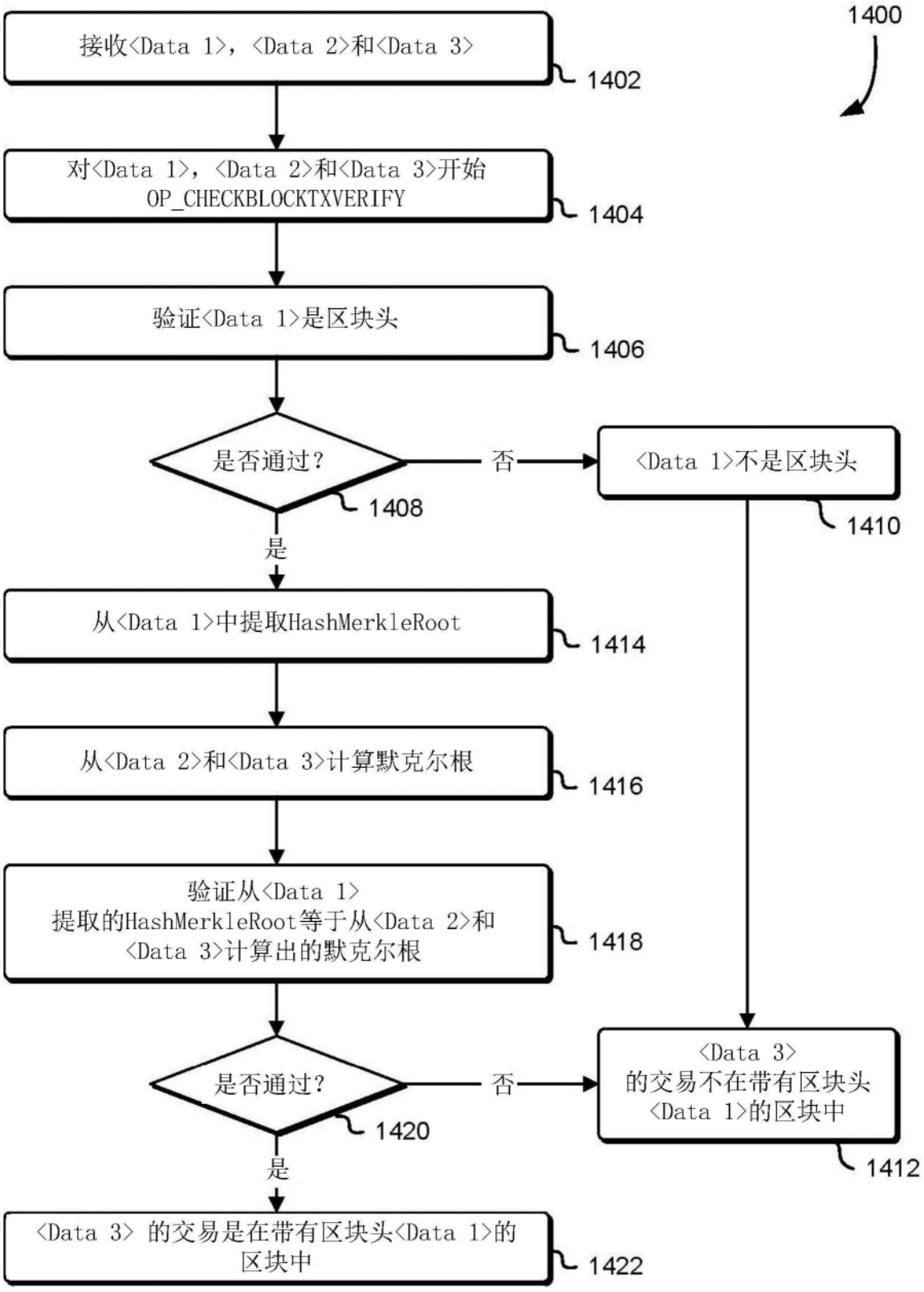

1400
接收<Data 1>，<Data 2>和<Data 3>
1402
对<Data 1>，<Data 2>和<Data 3>开始OP_CHECKBLOCKTXVERIFY
1404
验证<Data 1>是区块头
1406
是否通过？
1408
否
<Data 1>不是区块头
1410
是
从<Data 1>中提取HashMerkleRoot
1414
从<Data 2>和<Data 3>计算默克尔根
1416
验证从<Data 1>提取的HashMerkleRoot等于从<Data 2>和<Data 3>计算出的默克尔根
1418
是否通过？
1420
否
<Data 3>的交易不在带有区块头<Data 1>的区块中
1412
是
<Data 3> 的交易是在带有区块头<Data 1>的区块中
1422

图13

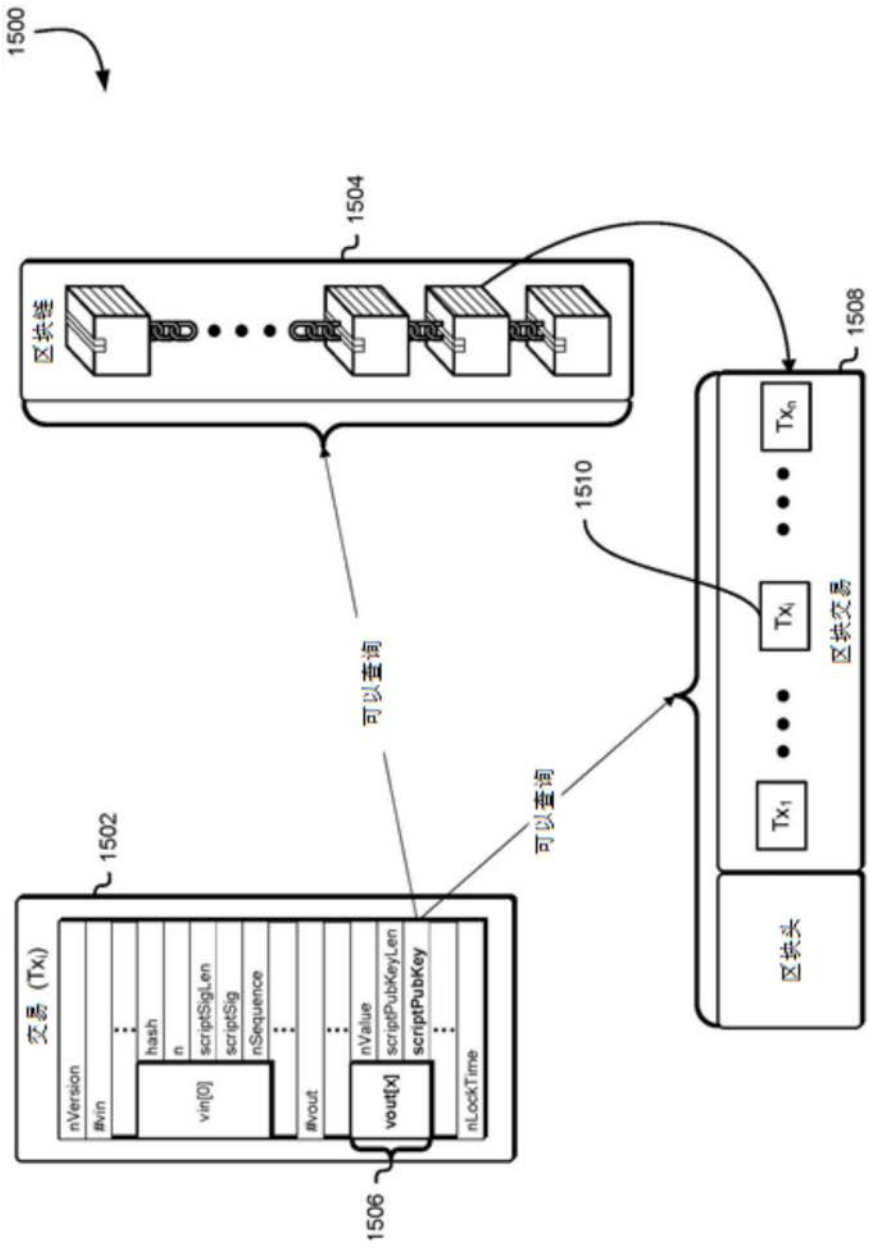

1500
1504
区块链
1508
1510
Tx_n
Tx_i
Tx_1
区块交易
区块头
可以查询
可以查询
1502
交易 (Tx_i)
nVersion
#vin
hash
n
scriptSigLen
scriptSig
nSequence
vin[0]
#vout
nValue
scriptPubKeyLen
scriptPubKey
vout[x]
nLockTime
1506

图14

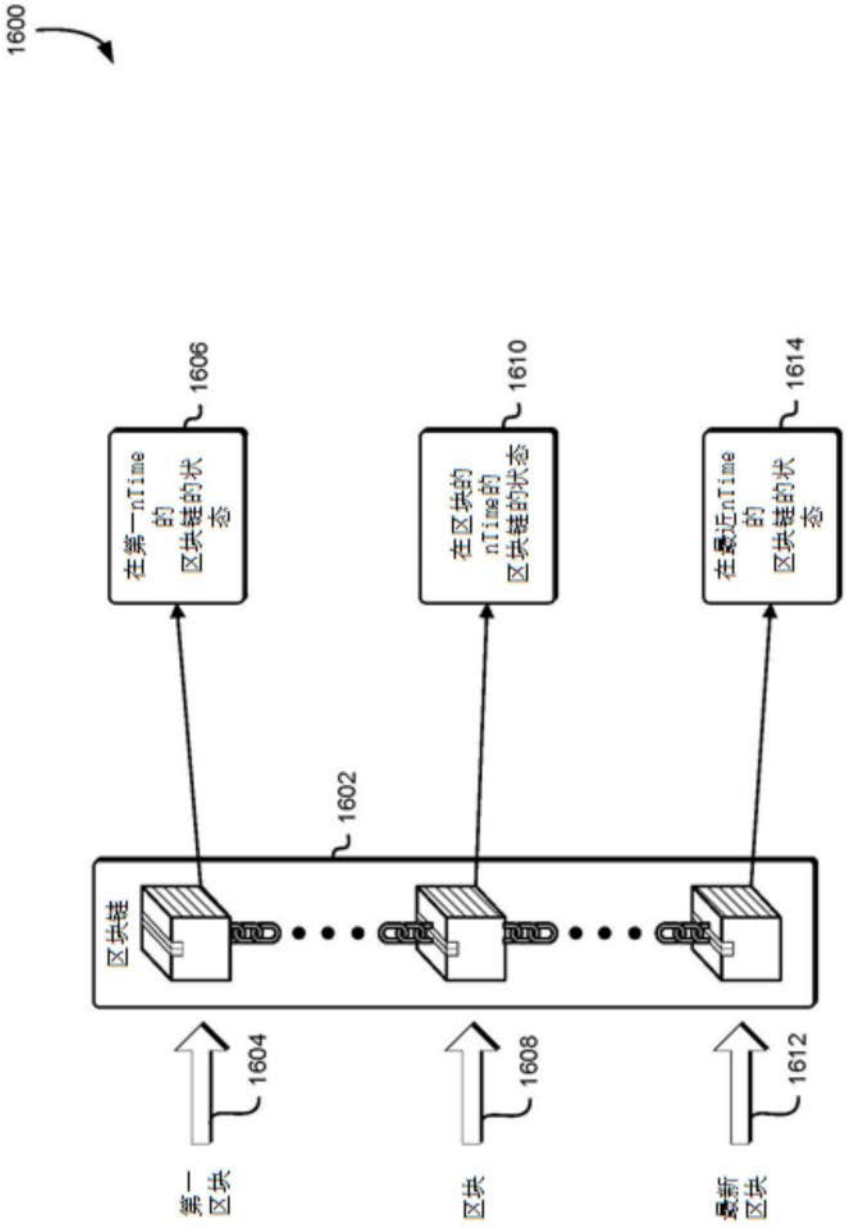

1600
1606
在第一nTime的区块链的状态
1610
在区块的nTime的区块链的状态
1614
在最近nTime的区块链的状态
1602
区块链
1604
1608
1612

图15

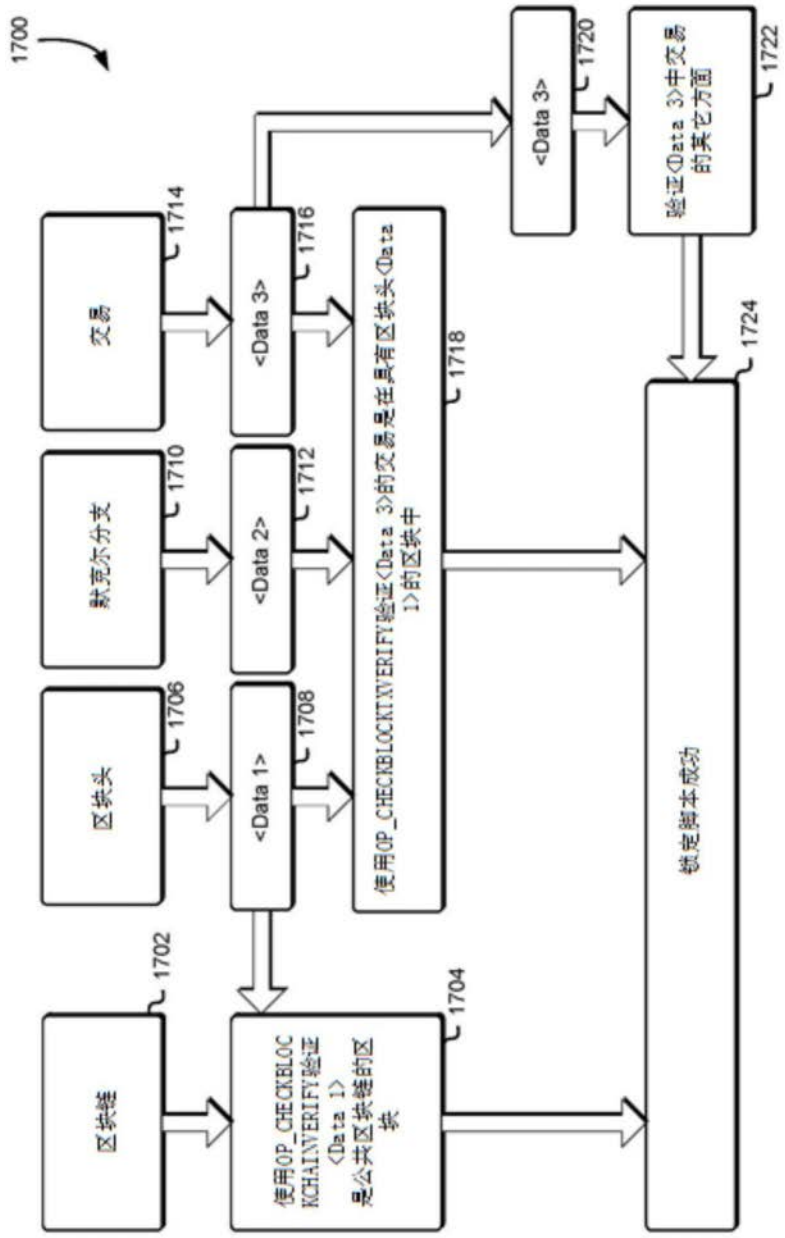

1700
交易
1714
默克尔分支
1710
区块头
1706
区块链
1702
<Data 3>
1716
<Data 2>
1712
<Data 1>
1708
使用OP_CHECKBLOCKTXVERIFY验证<Data 3>的交易是在具有区块头<Data 1>的区块中
1718
使用OP_CHECKBLOCKCHAINVERIFY验证<Data 1>是公共区块链的区块
1704
<Data 3>
1720
验证<Data 3>中交易的其它方面
1722
锁定脚本成功
1724

图16

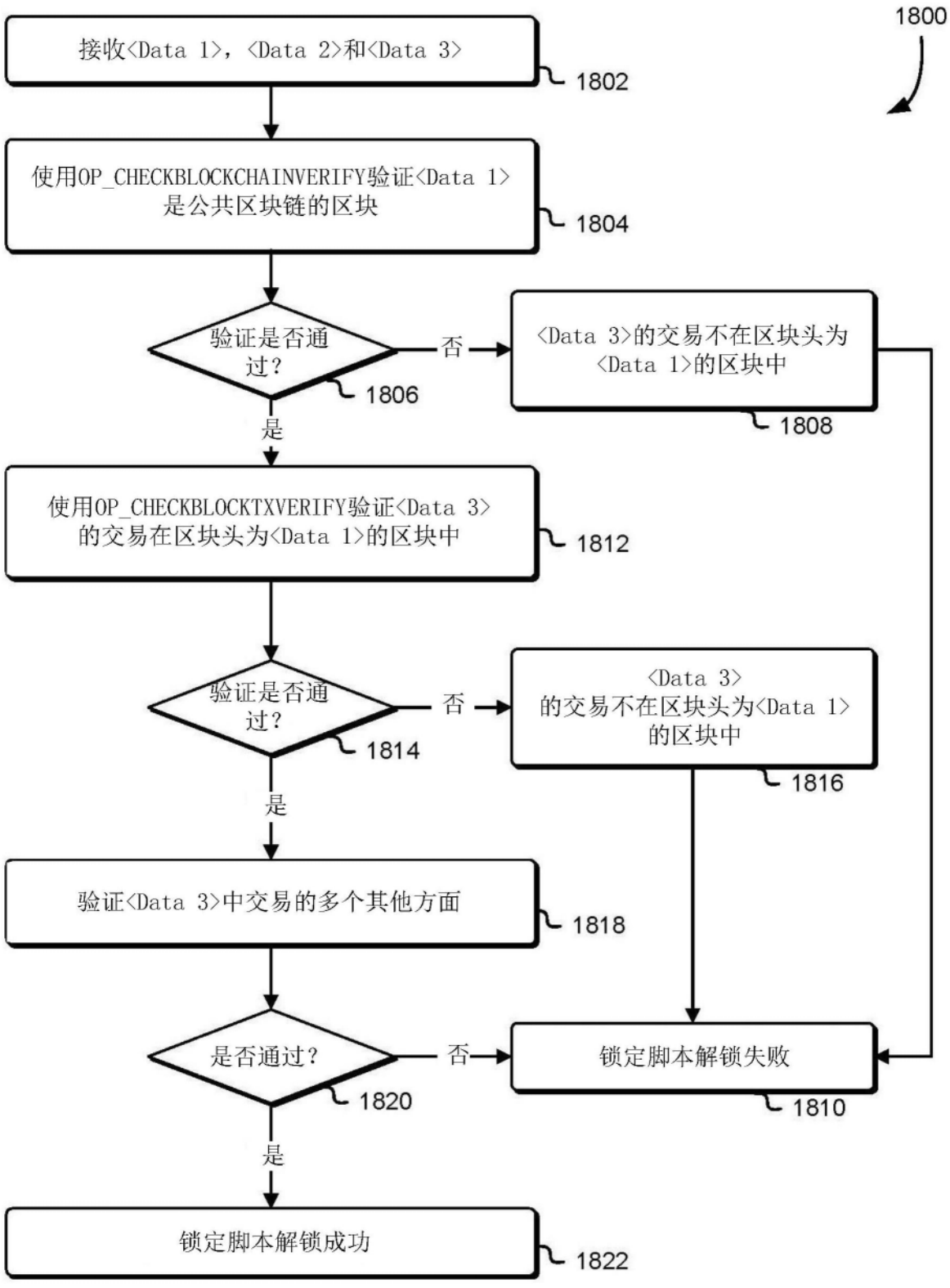

1800
接收<Data 1>，<Data 2>和<Data 3>
1802
使用OP_CHECKBLOCKCHAINVERIFY验证<Data 1>是公共区块链的区块
1804
验证是否通过？
1806
否
<Data 3>的交易不在区块头为<Data 1>的区块中
1808
是
使用OP_CHECKBLOCKTXVERIFY验证<Data 3>的交易在区块头为<Data 1>的区块中
1812
验证是否通过？
1814
否
<Data 3>的交易不在区块头为<Data 1>的区块中
1816
是
验证<Data 3>中交易的多个其他方面
1818
是否通过？
1820
否
锁定脚本解锁失败
1810
是
锁定脚本解锁成功
1822

图17

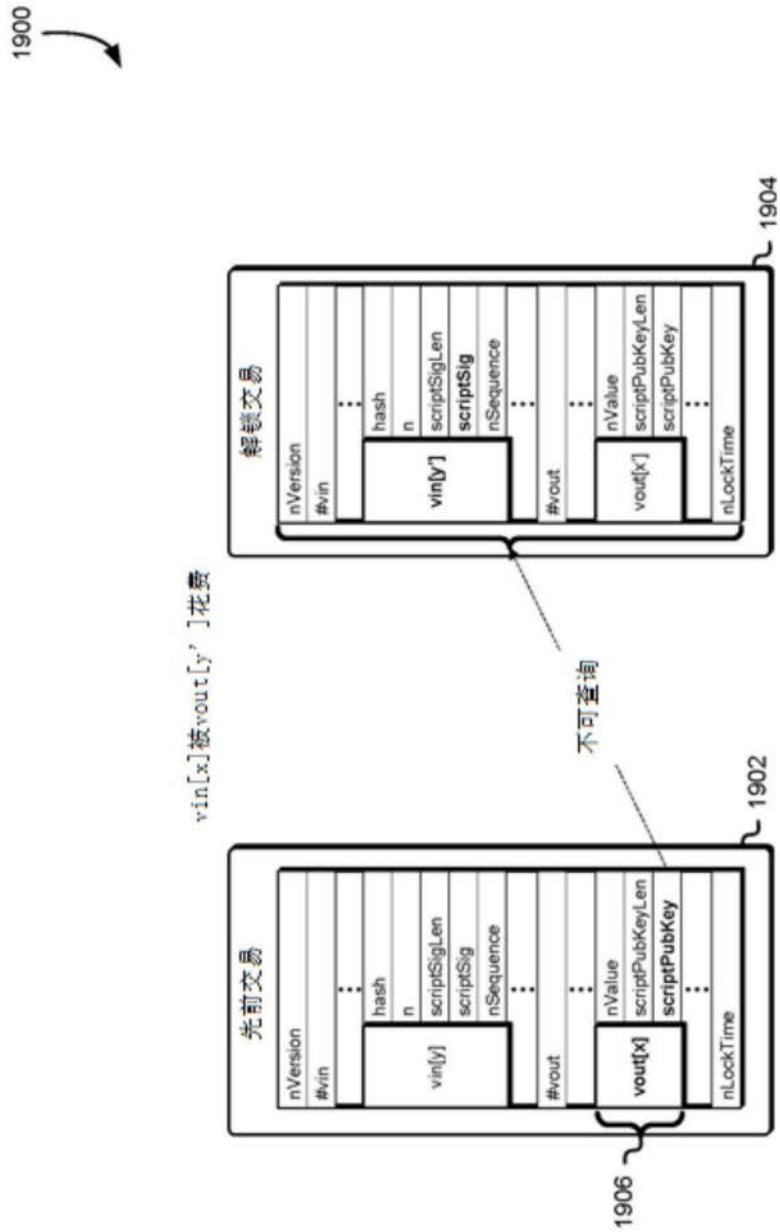

1900
1904
解锁交易
nVersion
#vin
hash
n
scriptSigLen
scriptSig
nSequence
vin[y']
#vout
nValue
scriptPubKeyLen
scriptPubKey
vout[x']
nLockTime
vin[x]被vout[y']花费
不可查询
1902
先前交易
nVersion
#vin
hash
n
scriptSigLen
scriptSig
nSequence
vin[y]
#vout
nValue
scriptPubKeyLen
scriptPubKey
vout[x]
nLockTime
1906

图18

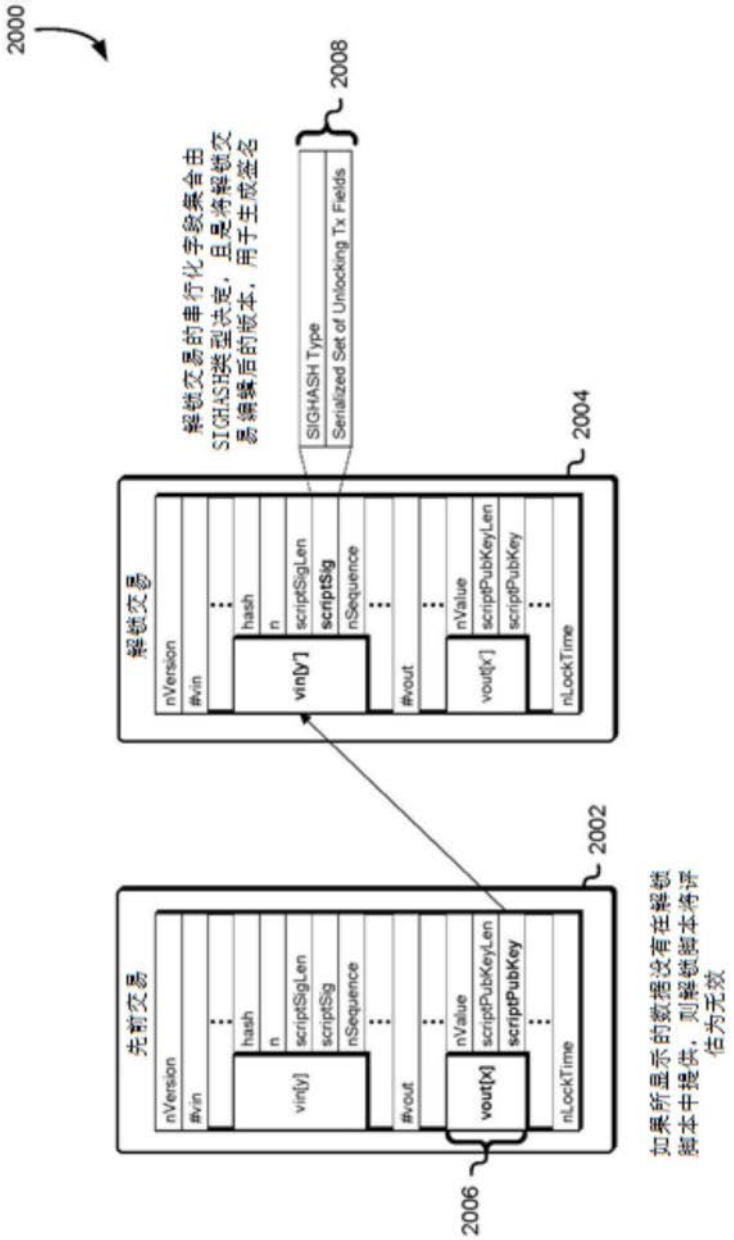

2000
2008
解锁交易的串行化字段集合由SIGHASH类型决定，且是将解锁交易编辑后的版本，用于生成签名
SIGHASH Type
Serialized Set of Unlocking Tx Fields
2004
解锁交易
nVersion
#vin
hash
n
scriptSigLen
scriptSig
nSequence
#vout
nValue
scriptPubKeyLen
scriptPubKey
nLockTime
vin[y]
vout[x]
2002
先前交易
scriptPubKey
2006
如果所显示的数据没有在解锁脚本中提供，则解锁脚本将评估为无效

图19

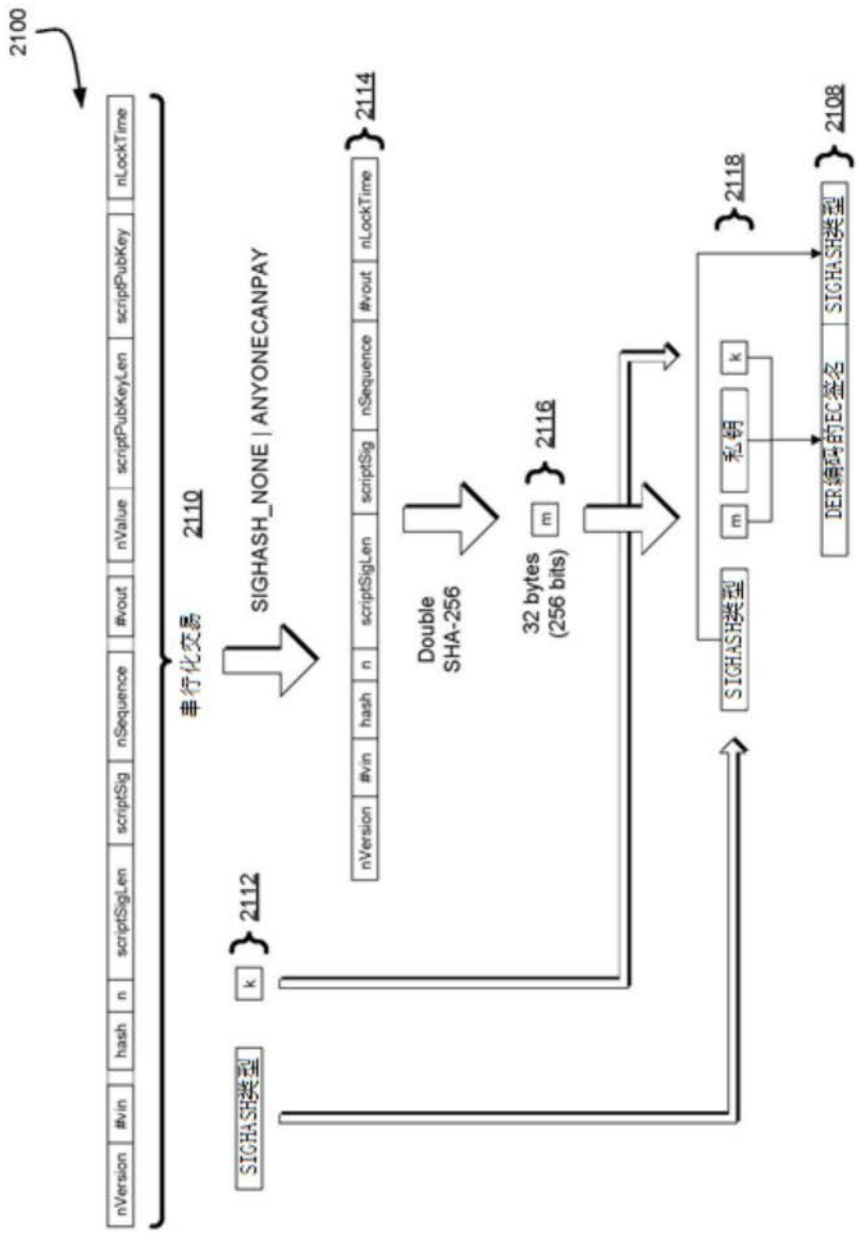

2100
nVersion
#vin
hash
n
scriptSigLen
scriptSig
nSequence
#vout
nValue
scriptPubKeyLen
scriptPubKey
nLockTime
串行化交易
2110
SIGHASH_NONE | ANYONECANPAY
2112
k
SIGHASH类型
nVersion
#vin
hash
n
scriptSigLen
scriptSig
nSequence
#vout
nLockTime
2114
Double SHA-256
32 bytes (256 bits)
m
2116
SIGHASH类型
m
私钥
k
2118
DER编码的EC签名
SIGHASH类型
2108

图20

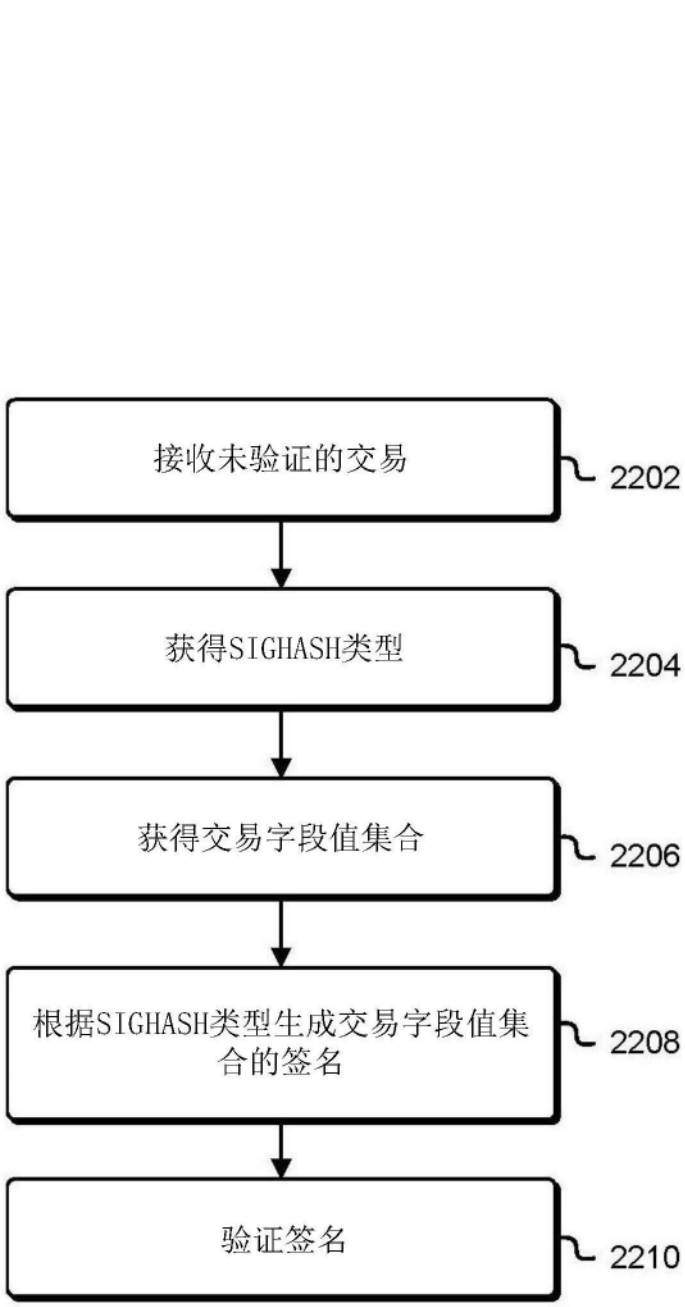

2200
接收未验证的交易
2202
获得SIGHASH类型
2204
获得交易字段值集合
2206
根据SIGHASH类型生成交易字段值集合的签名
2208
验证签名
2210

图21

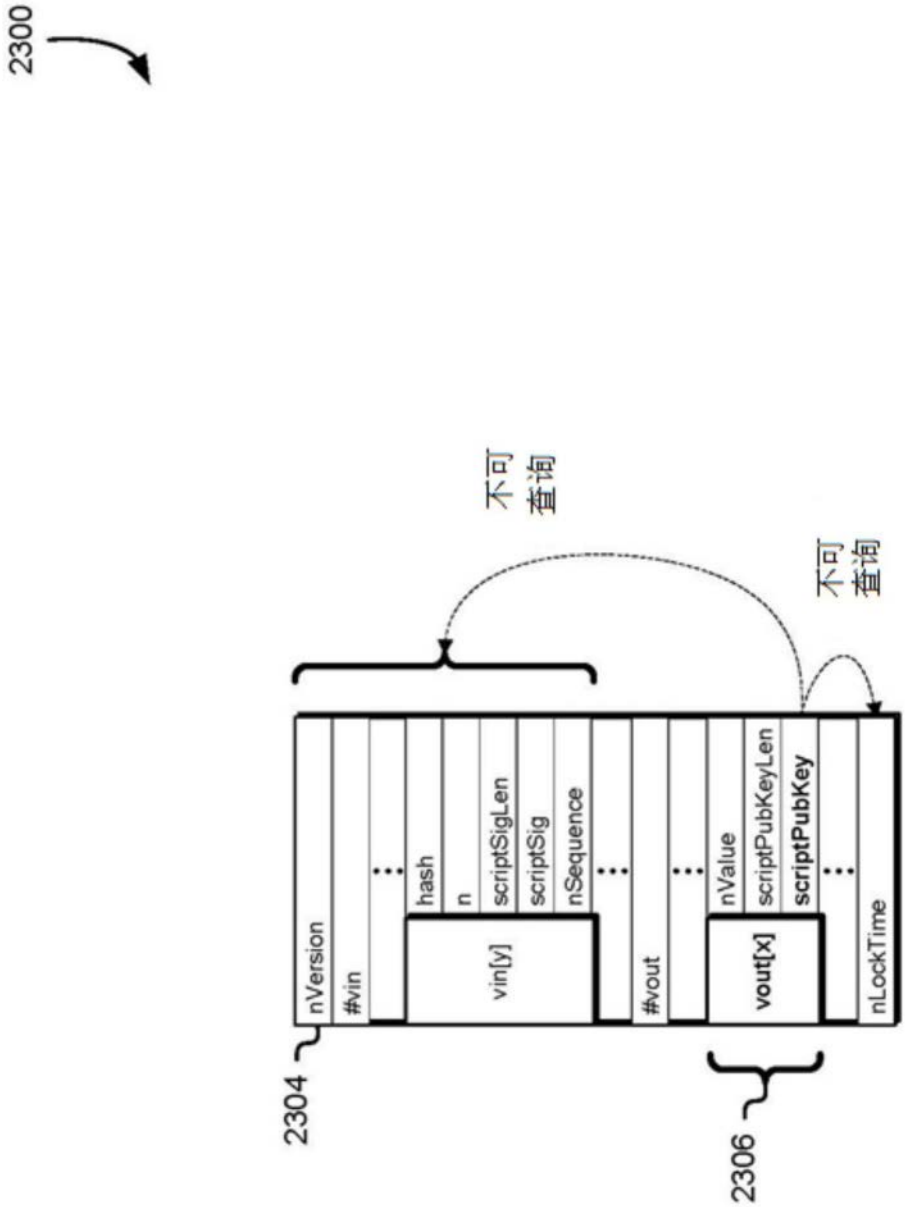

2300
不可查询
不可查询
nVersion
#vin
hash
n
scriptSigLen
scriptSig
nSequence
vin[y]
#vout
nValue
scriptPubKeyLen
scriptPubKey
vout[x]
nLockTime
2304
2306

图22

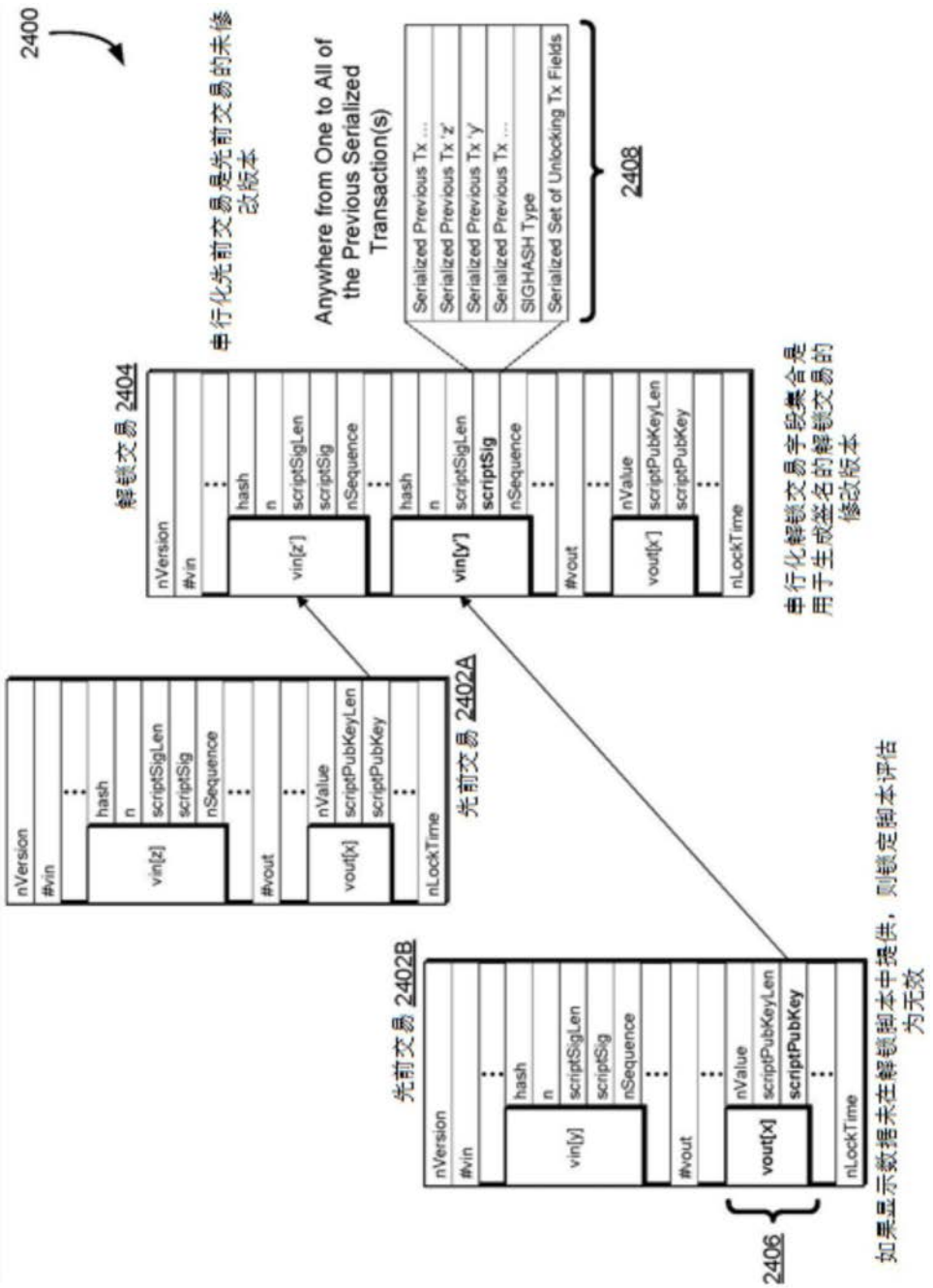

2400
串行化先前交易是先前交易的未修改版本
Anywhere from One to All of the Previous Serialized Transaction(s)
Serialized Previous Tx ...
Serialized Previous Tx 'z'
Serialized Previous Tx 'y'
Serialized Previous Tx ...
SIGHASH Type
Serialized Set of Unlocking Tx Fields
2408
解锁交易 2404
nVersion
#vin
hash
n
scriptSigLen
scriptSig
nSequence
vin[z]
vin[y]
#vout
nValue
scriptPubKeyLen
scriptPubKey
vout[x]
nLockTime
串行化解锁交易字段集合是用于生成签名的解锁交易的修改版本
先前交易 2402A
先前交易 2402B
2406
如果显示数据未在解锁脚本中提供，则锁定脚本评估为无效

图23

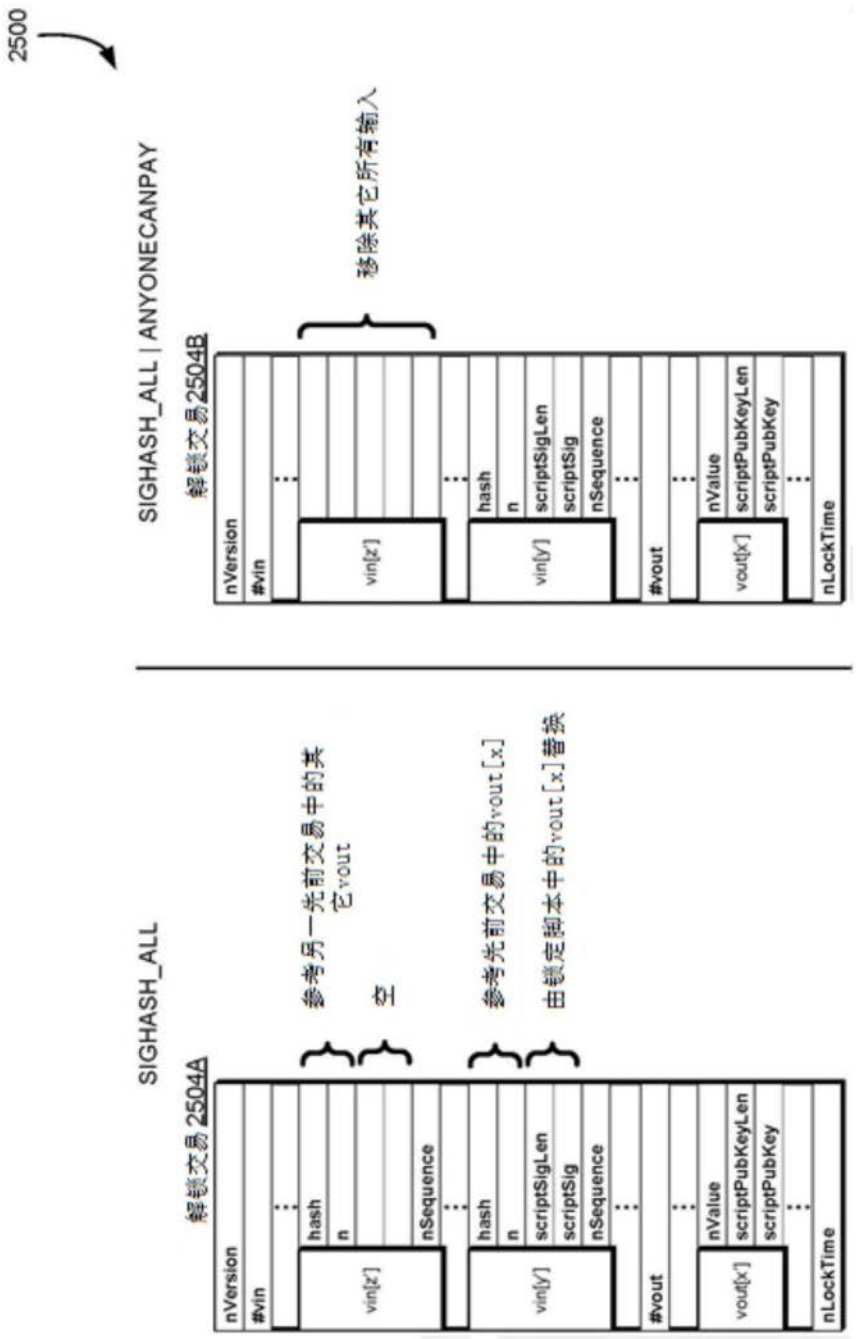

2500
SIGHASH_ALL | ANYONECANPAY
解锁交易2504B
移除其它所有输入
nVersion
#vin
vin[z]
hash
n
nSequence
vin[y]
scriptSigLen
scriptSig
nSequence
#vout
vout[x]
nValue
scriptPubKeyLen
scriptPubKey
nLockTime
SIGHASH_ALL
解锁交易2504A
参考另一先前交易中的其它vout
空
参考先前交易中的vout[x]
由锁定脚本中的vout[x]替换

图24

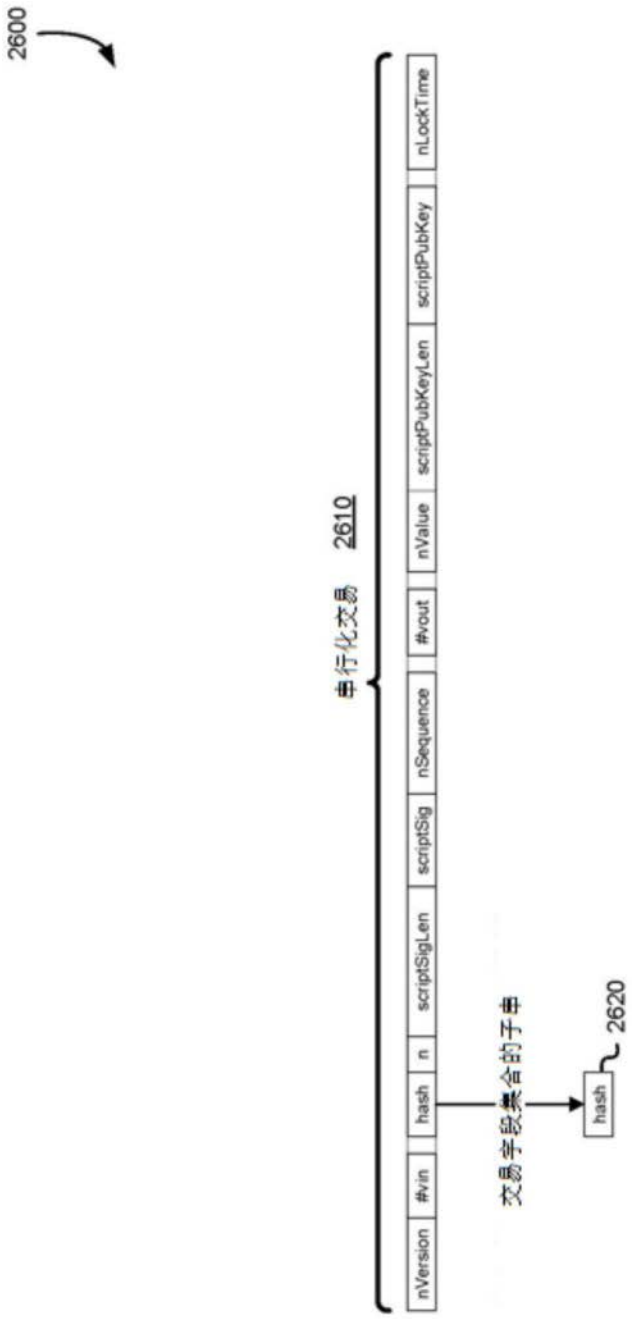

2600
2610
串行化交易
nVersion
#vin
hash
n
scriptSigLen
scriptSig
nSequence
#vout
nValue
scriptPubKeyLen
scriptPubKey
nLockTime
交易字段集合的子串
hash
2620

图25

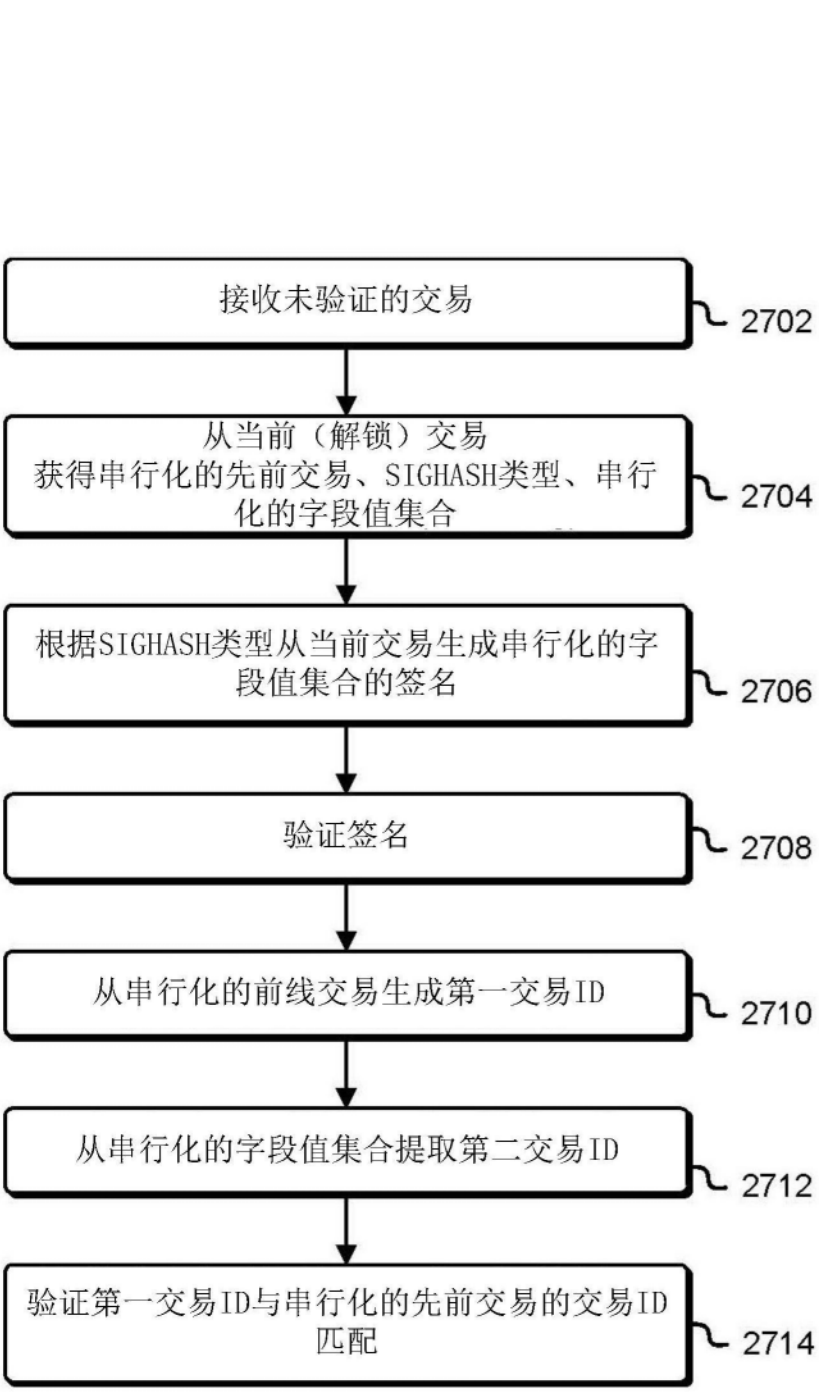

2700
接收未验证的交易
2702
从当前（解锁）交易
获得串行化的先前交易、SIGHASH类型、串行化的字段值集合
2704
根据SIGHASH类型从当前交易生成串行化的字段值集合的签名
2706
验证签名
2708
从串行化的前线交易生成第一交易ID
2710
从串行化的字段值集合提取第二交易ID
2712
验证第一交易ID与串行化的先前交易的交易ID匹配
2714

图26

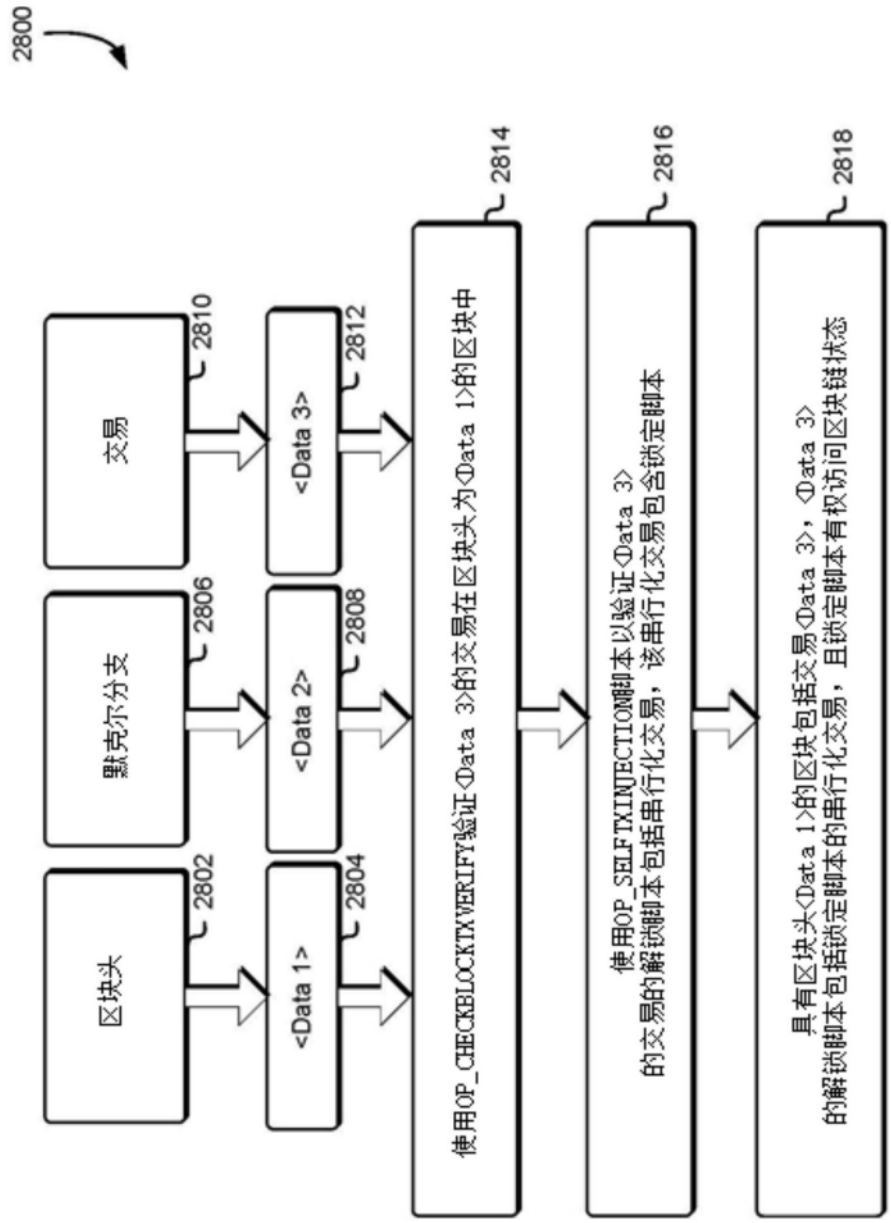

2800
区块头
2802
<Data 1>
2804
默克尔分支
2806
<Data 2>
2808
交易
2810
<Data 3>
2812
使用OP_CHECKBLOCKTXVERIFY验证<Data 3>的交易在区块头为<Data 1>的区块中
2814
使用OP_SELFTXINJECTION脚本以验证<Data 3>
的交易的解锁脚本包括串行化交易，该串行化交易包含锁定脚本
2816
具有区块头<Data 1>的区块包括交易<Data 3>，<Data 3>
的解锁脚本包括锁定脚本的串行化交易，且锁定脚本有权访问区块链状态
2818

图27

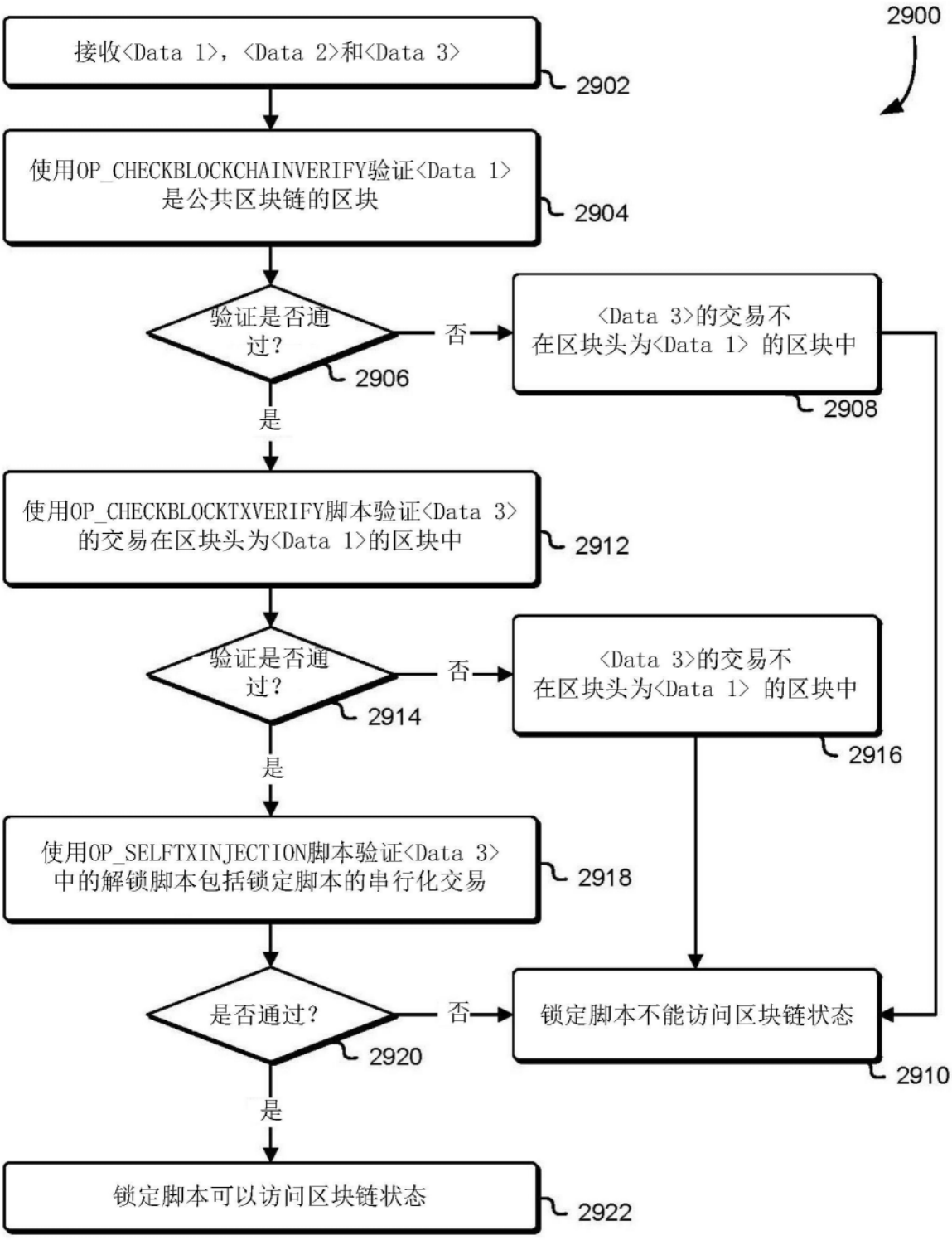

2900
接收<Data 1>，<Data 2>和<Data 3>
2902
使用OP_CHECKBLOCKCHAINVERIFY验证<Data 1>是公共区块链的区块
2904
验证是否通过？
2906
否
<Data 3>的交易不在区块头为<Data 1>的区块中
2908
是
使用OP_CHECKBLOCKTXVERIFY脚本验证<Data 3>的交易在区块头为<Data 1>的区块中
2912
验证是否通过？
2914
否
<Data 3>的交易不在区块头为<Data 1>的区块中
2916
是
使用OP_SELFTXINJECTION脚本验证<Data 3>中的解锁脚本包括锁定脚本的串行化交易
2918
是否通过？
2920
否
锁定脚本不能访问区块链状态
2910
是
锁定脚本可以访问区块链状态
2922

图28

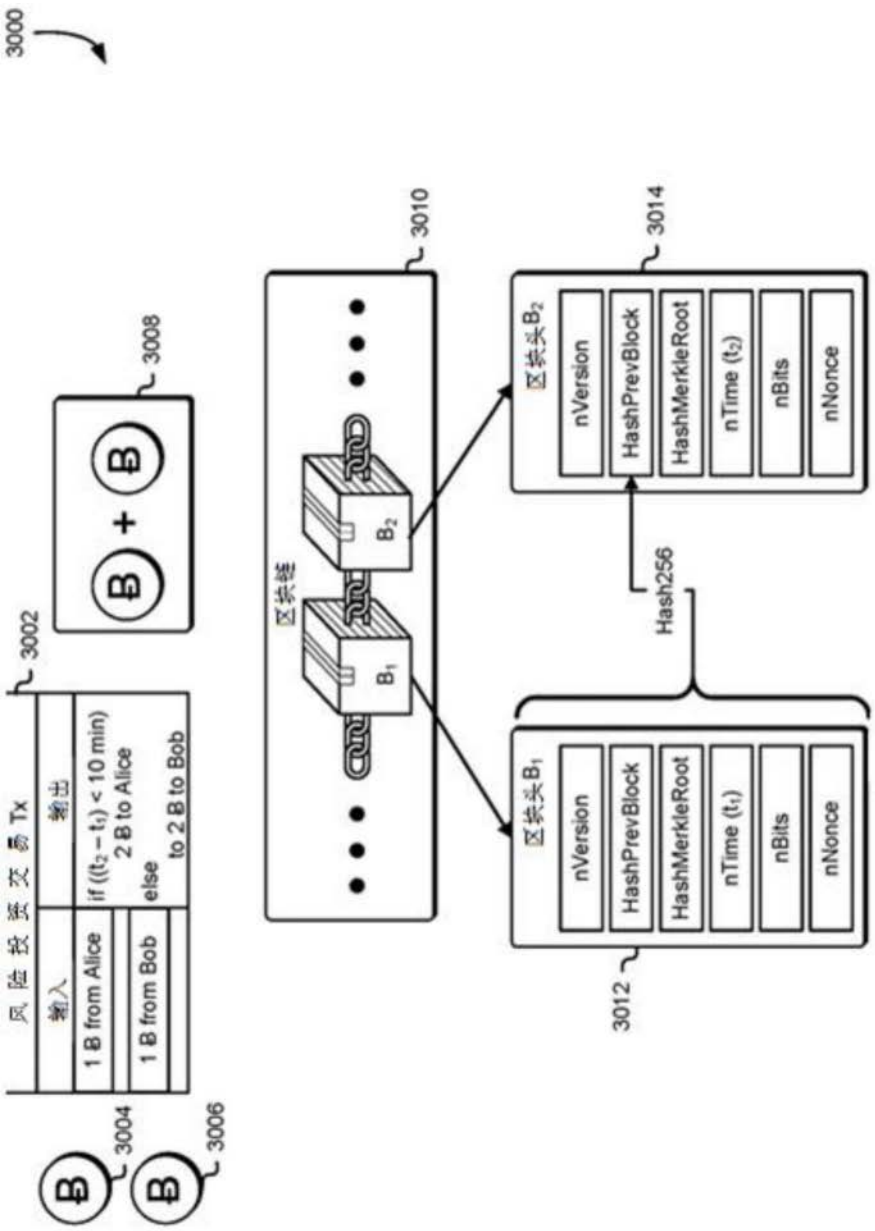

3000
3002
风险投资交易Tx
输入
输出
1 B from Alice
if ((t2 − t1) < 10 min)
2 B to Alice
1 B from Bob
else
to 2 B to Bob
3004
3006
3008
3010
区块链
B1
B2
3012
区块头B1
nVersion
HashPrevBlock
HashMerkleRoot
nTime (t1)
nBits
nNonce
Hash256
3014
区块头B2
nVersion
HashPrevBlock
HashMerkleRoot
nTime (t2)
nBits
nNonce

图29