(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111316595 B
(45)授权公告日 2024.09.03

(21)申请号 201880072801.5
(22)申请日 2018.10.29
(65)同一申请的已公布的文献号
申请公布号 CN 111316595 A
(43)申请公布日 2020.06.19
(30)优先权数据
1718505.9 2017.11.09 GB
1719998.5 2017.11.30 GB
1720768.9 2017.12.13 GB
1801753.3 2018.02.02 GB
1805948.5 2018.04.10 GB
1806444.4 2018.04.20 GB
(85)PCT国际申请进入国家阶段日
2020.05.09
(86)PCT国际申请的申请数据
PCT/IB2018/058433 2018.10.29
(87)PCT国际申请的公布数据
WO2019/092543 EN 2019.05.16

(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 亚历山卓·科瓦奇 西蒙·马蒂奥
帕特里克·蒙特利斯基
史蒂芬·文森特
(74)专利代理机构 北京市竞天公诚律师事务所 11770
专利代理师 陈果

(51)Int.Cl.
H04L 9/32(2006.01)
H04L 9/08(2006.01)
H04L 9/40(2022.01)

(56)对比文件
Eberhardt.Zokrates -A Toolbox for zkSNARKs on Ethereum.https://www.youtube.com/watch?V=sSlryw_b5j_0&t=4s.2017,第0-16分钟. (续)

审查员 胡平平

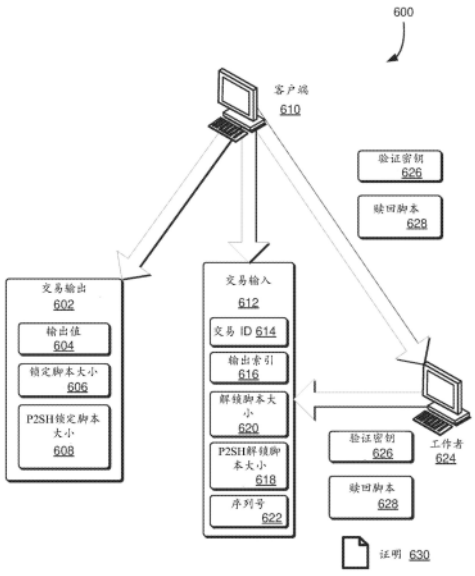
权利要求书2页 说明书21页 附图8页

(54)发明名称

用于保护验证密钥不被更改并验证正确性证明的有效性的系统

(57)摘要

本发明涉及分布式账本技术,例如,基于共识的区块链。区块链交易可能包括被锁定脚本锁定(encumber)的数字资源,该锁定脚本对使用被锁定资源(例如,转移对被锁定资源的所有权/控制)之前必须满足的一组条件进行编码。工作者(例如,计算机系统)执行一个或多个计算来生成证明,该证明被编码为解锁脚本的一部分。验证算法可以利用证明、验证密钥和附加数据(诸如与工作者相关联的加密材料(例如,数字签名)),来验证交易的数字资产应该被转移。作为该交易的验证结果,任何第三方都能够检验合约是否被执行校正,而不是重新执行该合约,从而节省计算能力。



CN 111316595 B

[转续页]

[接上页]

(56)对比文件

antonopoulos.Mastering Bitcon-Unlocking Digital Cryptocurrencies.o'reilly.2014,第124-138页.

Matteo campanelli.Zero-Knowledge Contingent Payments Revisited:Attacks and Payments for Services.http://eprint.iacr.org/2017/566.pd.2017,第1-26页.

1.一种计算机实现的方法,包括:

生成交易的交易输出,所述交易输出包括数字资产的指示和锁定脚本,所述锁定脚本对用于转移对数字资产的控制的一组条件进行编码,至少部分地基于验证密钥和证明来确定这组条件的满足;

生成所述交易的交易输入,所述交易输入包括:

与所述交易输出相关联的标识符;以及

包含所述证明的解锁脚本;以及

至少部分地基于所述锁定脚本和所述解锁脚本来验证这组条件的满足;以及

响应于验证这组条件的满足,转移对数字资产的控制;

其中,客户端利用所述验证密钥和与所述客户端相关联的第一数字证书对所述交易输入进行编码,并且工作者利用所述证明和与所述工作者相关联的第二数字证书对所述交易输入进行编码;

其中所述数字资产不属于虚拟货币。

2.根据权利要求1所述的计算机实现的方法,其中,所述验证密钥包括第一多个有限域元素,并且所述证明包括第二多个有限域元素。

3.根据权利要求2所述的计算机实现的方法,其中,所述有限域元素是椭圆曲线上的点。

4.根据权利要求2所述的计算机实现的方法,其中,所述有限域元素以压缩格式编码。

5.根据权利要求1所述的计算机实现的方法,其中,所述锁定脚本包括指令,所述指令在所述解锁脚本未满足这组条件的情况下,为所述数字资产的提供者收回所述数字资产。

6.根据权利要求1所述的计算机实现的方法,其中:

所述解锁脚本还包括赎回脚本,其中,所述验证密钥和所述赎回脚本需满足这组条件;并且

所述锁定脚本对这组条件的一个条件进行编码,所述条件为所述赎回脚本的散列与预定值匹配。

7.根据权利要求6所述的计算机实现的方法,其中,所述赎回脚本的大小为小于或等于520字节。

8.根据权利要求6所述的计算机实现的方法,其中,所述解锁脚本包括所述验证密钥的一个或多个元素。

9.根据权利要求6所述的计算机实现的方法,其中,所述交易是基于脚本散列的交易。

10.根据权利要求1所述的计算机实现的方法,其中,所述交易是根据基于区块链的协议的标准交易。

11.根据权利要求1所述的计算机实现的方法,其中,所述锁定脚本和所述解锁脚本以基于堆栈的脚本语言编码。

12.根据权利要求1所述的计算机实现的方法,其中,这组条件包括双线性约束。

13.一种系统,包括:

处理器;以及

包括可执行指令的存储器,所述可执行指令由处理器执行使得所述系统执行根据权利要求1至12中任一项所述的计算机实现的方法。

14.一种非暂时性计算机可读存储介质,其上存储有可执行指令,所述可执行指令由计算机系统的处理器执行使得所述计算机系统至少执行根据权利要求1至12中任一项所述的计算机实现的方法。

用于保护验证密钥不被更改并验证正确性证明的有效性的系统

技术领域

[0001] 本发明通常涉及区块链技术,更具体地,涉及通过利用锁定脚本在区块链上执行零知识(zero-knowledge)协议以保护验证密钥不被更改并验证正确性证明的有效性。本发明进一步利用加密和数学技术来加强与通过区块链网络进行的电子传输相关的安全性。本发明特别适合但不限于在智能合约生成和执行中使用。

背景技术

[0002] 在本文中,术语“区块链”可指若干种类型的电子的、基于计算机的分布式账本中的任何一种。这些包括基于共识的区块链和交易链技术、许可及未许可的账本、共享账本及其变体。应当注意的是,包括非商用应用在内的替代的区块链实现方式和协议也落在本发明的范围内。

[0003] 区块链是一种点对点(peer-to-peer)电子账本,该电子账本被实现为基于计算机的、去中心化的分布式系统,该系统由区块组成,而区块又可以由交易(transaction)和其他信息组成。在一些示例中,“区块链交易”是指对包括数据和一组条件的字段值的结构化集合进行编码的输入消息,其中,满足这组条件是将字段集写入区块链数据结构的先决条件。例如,每个交易都是对区块链系统中参与者之间数字资产的控制的转移进行编码的数据结构,并且每个交易包括至少一个输入和至少一个输出。在一些实施例中,“数字资产”是指与使用权相关联的二进制数据。在一些实现方式中,可以通过将数字资产的至少一部分从第一实体重新关联到第二实体来执行数字资产的控制转移。区块链的每个区块可能都包含前一区块的散列,以便将区块链接在一起,以创建一个永久的、不可更改的所有交易的记录,这些交易自区块链创建以来写入区块链。

[0004] 在一些示例中,“基于堆栈的脚本语言”是指支持各种基于堆栈或面向堆栈的执行模型和操作的编程语言。即,基于堆栈的脚本语言可以利用堆栈。通过使用堆栈,可以将值推送至堆栈顶部,或者从堆栈顶部弹出。对堆栈执行的各种操作会导致将一个或多个数值推送至堆栈顶部或从堆栈顶部弹出。例如,OP_EQUAL操作从堆栈中弹出前两项,将其进行比较,并将结果(例如,在相等的情况下为1,在不相等的情况下为0)推送至堆栈顶部。对堆栈执行的其他操作,例如,OP_PICK,可允许从堆栈顶部以外的位置选择项。在一些实施例使用的某些脚本语言中,可以有至少两个堆栈:主堆栈(main stack)和备用堆栈(alternate stack)。脚本语言的一些操作可以将项从一个堆栈的顶部移动到另一堆栈的顶部。例如,OP_TOALTSTACK将值从主堆栈的顶部移动到备用堆栈的顶部。应当注意的是,在某些情况下,基于堆栈的脚本语言可能不仅限于以严格的后进先出(LIFO)方式进行的操作。例如,基于堆栈的脚本语言可以支持将堆栈中的第n个项复制或移动到顶部的操作(例如,分别为OP_PICK和OP_ROLL)。可将以基于堆栈的脚本语言编写的脚本推送至逻辑堆栈,其中可以使用任何合适的数据结构,例如,向量、列表或堆栈,来实现该逻辑堆栈。

[0005] 为了将交易写入区块链,必须对交易进行“验证”。网络节点执行工作,以确保每个

交易有效,无效交易则被网络拒绝。一个节点可以具有不同于其他节点的有效性标准。因为区块链的有效性是基于共识的,所以如果大多数节点同意交易是有效的,则该交易被认为是有效的。安装在节点上的软件客户端部分地通过执行未花费的交易输出(UTXO)锁定和解锁脚本对引用UTXO的交易执行该验证工作。如果锁定和解锁脚本的执行评估为真(TRUE),并且满足其他验证条件(如果适用的话),则由节点验证该交易。已验证的交易被传播到其他网络节点,随后网络节点可以选择将该交易包括在区块链中。因此,为了将交易写入区块链,该交易必须i)由接收该交易的第一节点进行验证(如果交易得到验证,则该节点将该交易中继到网络中的其他节点);ii)添加到新区块中;并且iii)将交易添加到过去交易的公共账本中。当区块链添加了足够数量的区块以使交易实际上不可逆转时,该交易被视为已确认。本文提到的UTXO不涉及比特币的交易。

[0006] 数字企业家已经开始探索使用加密安全系统以及可存储在区块链上的数据来实现新系统。如果区块链能够用于自动化任务和流程,则将是非常有利的。此类解决方案能够利用区块链的优势(例如,事件的永久防篡改记录、分布式处理等),同时在其应用中更加通用。

[0007] 本公开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可以是记录在区块链交易中的机器可读且可执行程序。基于区块链的计算机程序可以包括可处理输入以产生结果的规则,然后可以根据这些结果执行动作。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”。与用自然语言编写的传统合约不同,智能合约可以是旨在用于自动执行机器可读合约或协议的条款的计算机程序。

[0008] 另一与区块链相关的感兴趣的领域是使用‘令牌’来表示现实世界实体并且经由区块链对其进行转移。潜在的敏感或秘密项可以由没有可辨别意义或价值的令牌来表示。因此,令牌充当允许从区块链引用现实世界项的标识符。

[0009] 在实施例中,尽管与特定实体的交互可以在智能合约中的特定步骤进行编码,但智能合约也可以自动执行和自我实施。智能合约是机器可读和可执行的。在一些示例中,自动执行是指成功执行智能合约,以实现UTXO的转移。注意,在这些示例中,能够导致UTXO转移的“实体”是指能够创建解锁脚本而无需证明知道某个秘密的实体。换言之,可以验证解锁交易而无需验证数据源(例如,创建解锁交易的实体)有权访问加密秘密(例如,私有非对称密钥、对称密钥等)。此外,在这样的示例中,自我实施是指使区块链网络的验证节点根据约束来实施解锁交易。在一些示例中,在技术意义上使用的“解锁”UTXO(也称为“花费UTXO”),是指创建引用UTXO并作为有效执行的解锁交易。

[0010] 区块链交易输出包括锁定脚本和关于数字资产的所有权的信息。锁定脚本也可以称为负载(encumbrance),其通过指定转移UTXO所需满足的条件来“锁定”数字资产。例如,锁定脚本可要求在解锁脚本中提供某些数据来解锁相关联的数字资产。锁定脚本也称为“脚本公钥”。一种要求一方提供数据以解锁数字资产的技术包括在锁定脚本内部嵌入数据的散列。

发明内容

[0011] 因此,期望提供在这些方面中的一个或多个方面改进区块链技术的方法和系统。

[0012] 现在已经设计出这样一种改进的解决方案。

[0013] 根据本发明,可以提供一种用于区块链网络的节点的计算机实现的方法,该计算机实现的方法包括:生成交易的交易输出,该交易输出包括数字资产指示和锁定脚本,该锁定脚本对用于转移对数字资产的控制的一组条件进行编码,至少部分地基于验证密钥和证明来确定这组条件的满足;生成该交易的交易输入,该交易输入包括:与交易输出相关联的标识符以及包含该证明的解锁脚本;并且至少部分地基于锁定脚本和解锁脚本来验证这组条件的满足;以及响应于验证这组条件的满足,转移对数字资产的控制。

[0014] 该验证密钥可以包括第一多个有限域元素,并且该证明包括第二多个有限域元素。

[0015] 该有限域元素可以是椭圆曲线上的点。

[0016] 该有限域元素可以以压缩格式编码。

[0017] 优选地,该方法可以包括客户端利用验证密钥和与该客户端相关联的第一数字证书对该交易输入进行编码,并且工作者利用该证明和与该工作者相关联的第二数字证书对该交易输入进行编码。

[0018] 该锁定脚本可以包括指令,该指令在解锁脚本未满足这组条件的情况下,为数字资产的提供者收回数字资产。

[0019] 该解锁脚本还可以包括赎回脚本,其中,该验证密钥和该赎回脚本对足够的信息进行编码,以确定满足这组条件的至少一个子集;并且该锁定脚本对这组条件的一个条件进行编码,该条件为赎回脚本的散列与预定值匹配。

[0020] 该赎回脚本可以小于或等于预定的最大值,例如,520字节。

[0021] 该解锁脚本可以包括该验证密钥的一个或多个元素,该赎回脚本可以包括该验证密钥的至少一些剩余元素。解锁脚本和赎回脚本可以共同包括该验证密钥。

[0022] 该交易可以根据基于区块链的协议的标准交易。

[0023] 诸如锁定脚本和解锁脚本等脚本可以利用基于堆栈的脚本语言中的命令和数据来编码,其中,该命令和数据以后进先出的顺序放置在堆栈上。

[0024] 这组条件可以包括一个或多个双线性约束。

[0025] 还期望提供一种系统,包括:处理器;以及包括可执行指令的存储器,该可执行指令由处理器执行使得该系统执行如上所述的任一方法。

[0026] 还期望提供一种非暂时性计算机可读存储介质,其上存储有可执行指令,该可执行指令由计算机系统的一个或多个处理器执行使得该计算机系统至少执行如上所述的任一方法。

[0027] 本发明可以被描述为一种验证方法/系统和/或一种用于控制经由区块链交换或转移数字资产的控制方法/系统。在一些实施例中,数字资产是令牌。如下所述,本发明也可以被描述为用于通过区块链网络或平台执行操作的新型、改进的和有利的方式的安全方法/系统。

[0028] 本文提到的数字资产不属于虚拟货币。本文提到的令牌不属于虚拟货币。

附图说明

[0029] 参考本文描述的实施例,本发明的这些和其他方面将变得清楚并得以阐明。现在将参考附图,仅以示例的方式描述本发明的实施例,其中:

[0030] 图1示出了可以实现各种实施例的区块链环境；

[0031] 图2示出了根据各种实施例的可用于实现协议的计算环境；

[0032] 图3示出了适于执行可验证计算的环境的示图；

[0033] 图4示出了根据实施例的示例图，其中，使用压缩与未压缩编码来表示椭圆曲线上的点，该编码适用于与例如锁定或解锁脚本等区块链脚本一起使用；

[0034] 图5示出了一个示图，其中，客户端提供验证密钥，以用于对证明进行验证；

[0035] 图6示出了一个示图，其中，工作者提供验证密钥，以用于对证明进行验证；

[0036] 图7示出了根据实施例的用于生成赎回脚本的流程700的示图；以及

[0037] 图8示出了可以用于实践本发明的至少一个实施例的计算设备。

具体实施方式

[0038] 首先参考图1，其示出了根据本申请的实施例的与区块链相关联的示例区块链网络100。在该实施例中，示例区块链网络100包括被实现为点对点分布式电子设备的区块链节点，每个节点运行软件和/或硬件的实例，该实例执行遵循区块链协议的操作，该区块链协议至少部分地得到节点102的运营商的同意。在一些示例中，“节点”是指分布在区块链网络中的点对点电子设备。

[0039] 在一些实施例中，节点102可以由任何合适的计算设备(例如，由数据中心中的服务器、客户端计算设备(例如，台式电脑、膝上型计算机、平板电脑、智能手机等)、计算资源服务提供商的分布式系统中的多个计算设备或者如图8的计算设备800等任何合适的电子客户端设备)组成。在一些实施例中，节点102具有接收数据消息或对象的输入，该数据消息或对象表示所提议的交易(例如，交易104)。在一些实施例中，可以查询节点以获取其维护的信息，例如，获取交易104的状态信息。

[0040] 如图1所示，一些节点102通信耦合到一个或多个其他节点102。这种通信耦合可以包括有线或无线通信中的一种或多种。在该实施例中，每个节点102维护区块链中所有交易的“账本”的至少一部分。以这种方式，账本将是分布式账本。由影响账本的节点处理的交易可由一个或多个其他节点验证，从而保持账本的完整性。

[0041] 至于哪些节点102可以与哪些其他节点通信，假设消息是区块链协议指示应该转发的消息，则示例区块链网络100中的每个节点能够与一个或多个其他节点102通信，使得在节点之间传递的消息可以在示例区块链网络100(或其某个重要部分)中传播，这就足够了。一个这样的消息可能是由一个节点102，例如，节点102A，发布所提议的交易，然后该交易将沿着路径，例如，路径106，传播。另一这样的消息可能是发布提议包含到区块链上的新区块。

[0042] 在一个实施例中，至少一些节点102是执行如解决加密问题等复杂计算的网络节点。解决加密问题的网络节点为区块链创建新区块，并将该新区块广播给其他节点102。其他节点102验证网络节点的工作，并且在验证后，将该区块接受到区块链中(例如，通过将其添加到区块链的分布式账本中)。在一些示例中，区块是一组交易，通常标记有时间戳和前一区块的“指纹”(例如，散列)。以这种方式，每个区块可以链接到前一区块，从而创建链接区块链中区块的“链”。在实施例中，通过节点102的共识将有效区块添加到区块链。同样在一些示例中，区块链包括已验证区块的列表。

[0043] 在一个实施例中,至少一些节点102作为验证节点来操作,这些验证节点如本申请所描述的那样验证交易。在一些示例中,交易包括提供数字资产的所有权证明的数据以及用于接受或转移数字资产的所有权/控制的条件。在一些示例中,“解锁交易”是指将由前一个交易的UTXO指示的数字资产的至少一部分和与区块链地址相关联的实体重新关联(例如,转移所有权或控制)的区块链交易。在一些示例中,“前一个交易”是指包含解锁交易所引用的UTXO的区块链交易。在一些实施例中,交易包括“锁定脚本”,该“锁定脚本”用条件来锁定(encumber)交易,该条件必须在所有权/控制可以被转移(“解锁”)之前被满足。

[0044] 在一些实施例中,区块链地址是与实体相关联的字母数字字符串,其中对数字资产的至少一部分的控制被转移/重新关联到该实体。在一些实施例中实现的一些区块链协议中,在与实体相关联的公钥和区块链地址之间存在一对一的对应关系。在一些实施例中,交易的验证包括验证锁定脚本和/或解锁脚本中指定的一个或多个条件。在交易104成功验证后,验证节点将交易104添加到区块链,并将其分发给节点102。

[0045] 本文所述的系统和方法涉及使锁定脚本能够保护验证密钥V_K不被更改并检验证明π的有效性,从而允许在交易验证期间在区块链上执行零知识协议。

[0046] 可验证计算是一种允许生成计算证明的技术。在一个实施例中,客户端利用这种技术将对输入x上的函数f的评估外包给在本文称为工作者的另一计算实体。在一些情况下,客户端在计算上是有限制的,使得客户端执行函数的评估是不可行的(例如,使用客户端可用计算资源的计算的预期运行时间超过最大可接受阈值),尽管情况未必如此,但一般而言,客户端可以基于任何合适的标准,例如,计算运行时间、计算成本(例如,分配计算资源以执行函数评估的财务成本)等,来委托对输入x上的函数f的评估。

[0047] 在一个实施例中,工作者是任何合适的计算实体,例如,在本申请的其他地方更详细描述的区块链节点。在一个实施例中,工作者(例如,区块链节点)评估输入x上的函数f,并生成输出y和输出y的正确性的证明π,该证明可以由其他计算实体(例如,如上所述的客户端)和/或区块链网络的其他节点来验证。证明也可以称为论证,其可以比实际计算更快地验证,因此,通过验证证明的正确性,而不是重新计算在输入x上的函数f来确定由上述工作者生成的输出的正确性,可以减少计算开销(例如,减少供电及运行计算资源相关的电力开销和成本)。在零知识可验证计算中,工作者向客户端提供证据,证明工作者知晓具有特定属性的输入。

[0048] 零知识证明的一个有效变体是zk_SNARK(简洁非交互式的知识论证)。在一个实施例中,所有基于配对的zk_SNARK包括一个过程,在该过程中,工作者使用通用组操作来计算多个组元素,并且验证者使用多个配对乘积方程来检验证明。在一个实施例中,线性交互式证明在有限域上工作,并且工作者和验证者的消息包括编码、引用或以其他方式包括可用于确定域元素的向量的信息。

[0049] 在一个实施例中,本文描述的系统和方法允许区块链的网络节点执行一次计算(例如,输入x上的函数f的评估)并生成可用于验证输出的正确性的证明,其中,评估该证明的正确性在计算上比评估该函数要便宜。在这种情况下,操作和任务的成本(即,多贵)可以是指执行操作或任务的计算复杂度。在一个实施例中,计算复杂度是指执行排序算法的平均计算成本或最坏情况计算成本,例如,堆排序算法和快速排序算法都具有平均计算成本O(n logn),但是快速排序具有最坏情况计算成本$O(n^2)$,而堆排序具有最坏情况计算成本O

(n logn)。在一个实施例中,评估输入x上的函数f的平均计算成本和/或最坏情况计算成本比评估证明的正确性的平均计算成本和/或最坏情况计算成本要贵。因此,本文描述的系统和方法的使用是非常有利的,并且例如可以允许运行计算上更昂贵的合约,因为这样的合约不会成比例增加验证区块链所需的时间。进一步的优点可以包括减少验证者系统的功耗,从而提高验证者计算机系统的效率,并降低与运行这种验证者计算机系统以评估证明的正确性相关的能源成本。目前,必须在所有节点上执行和验证智能合约,这一约束限制了智能合约的复杂性。本文描述的方法和系统可以用来实现一种通过执行一次合约来生成正确性证明从而提高区块链的效率的系统,并且基于由工作者提供的正确性证明和由客户端提供的验证密钥,区块链的所有节点可以验证合约的有效性。这样,区块链的效率通过增加智能合约的吞吐量得以提高,这些智能合约可以由区块链的节点一起执行和/或允许计算更昂贵的智能合约。

[0050] 在一个实施例中,可以从输入/输出数据以及在零知识协议的设置阶段生成的并与证明π一起使用的公共参数中提取验证密钥V_K或其部分,以验证由工作者提供的所谓的正确性计算证明。例如,如上文和下文更详细描述的,允许锁定脚本的系统和方法保护验证密钥V_K不被更改并检验证明π的有效性,由此允许零知识协议在交易验证期间在区块链上执行。因此,本申请提供了使用区块链脚本来执行验证阶段的系统和方法,用于存储在计算的验证中所使用的元素。

[0051] 图2示出了根据各种实施例的可用于实现协议的计算环境200。可以使用区块链技术来实现该协议,以存储正确性证明,并将“按构造正确(correct-by-construction)”的加密方法与智能合约相结合。在一个实施例中,公共可验证计算方案包括三个阶段:设置阶段、计算阶段和验证阶段。

[0052] 设置阶段可以作为流程的一部分来执行,以外包计算任务的执行。如下所述,客户端可以指将计算任务的执行委托给工作者的实体,例如,客户或客户端计算机系统,该实体可以是不同的计算机系统。一般来说,客户端可以出于各种原因委托计算任务的执行,包括但不限于有限的计算资源、计算资源的缺乏、与利用客户端计算机系统来执行任务相关联的财务成本、与利用客户端计算机系统来执行任务相关联的能源成本(例如,依赖电池供电的移动设备或膝上型电脑可以利用工作者来执行计算密集型任务,从而节省电力并延长电池动力设备的使用)等。

[0053] 在一个实施例中,设置阶段包含客户端、客户、组织的雇员或任何其他使用具有精确语义的正式语言编写合约的合适的实体。可以用高级编程语言,例如,C或Java等来编写合约。一般来说,合约可以用可转换为由计算机系统操纵的格式的任何语言或语法来表示。在一个实施例中,具有有限目的的域特定语言可以提供类型安全(type-safety),并且可以利用受限的表达能力。生成的源代码可能是合约的精确描述。

[0054] 编译器202可以是包括可执行代码的任何硬件、软件或其组合,如果可执行代码由计算机系统的一个或多个处理器执行,则使系统将源代码206视为输入并产生电路。编译器202可以指基于已经编译成机器可读格式(例如,二进制代码)的指令来执行或完成指令的计算机程序。应当注意的是,虽然示出了编译器202,但是可以利用解释器、汇编器和其他合适的软件和/或硬件组件来将源代码转换成电路。在一个实施例中,该电路是运算电路,该运算电路包括携带来自域F的值并连接到逻辑和/或运算门的导线。在一个实施例中,系统

使用电路$\mathcal{C}$来生成二次程序Q 208,该二次程序包括提供原始电路$\mathcal{C}$的完整描述的一组多项式。

[0055] 在一个实施例中,编译器202能够识别编程语言(例如,C或Java)的实质子集,包括但不限于:预处理器指示、静态初始化器、全局和局部函数、区块范围(block-scoped)变量、数组、数据结构、指针、函数调用、函数运算符(例如,函子)、条件和循环语句以及算术的和以比特为单位的布尔运算符(bitwise Boolean operator)。在一个实施例中,编译器202按照编程语言的标准不支持整组命令(在某些情况下,这可能旨在防止某些类型的算法在智能合约中执行,例如,禁止递归算法)。在一个实施例中,编译器将源代码的表达扩展成运算门语言,以产生运算电路。Campanelli,M.等人(2017年)在"重新审视零知识或有支付:服务的攻击和支付(Zero-Knowledge Contingent Payments Revisited:Attacks and Payments for Services)"中,以及Tillich,S.和Smart,B在"适用于MPC和FHE的基本功能电路(Circuits of Basic Functions Suitable For MPC and FHE)"中,都曾设想过电路实现方式。运算电路可以被编译器202或任何其他合适的硬件、软件或其组合(例如,图2中未示出的软件模块)用来构建二次算术问题(QAP)。根据实施例,二次程序被编译成用于客户端(例如,密钥生成和验证)和工作者(例如,计算和证明生成)的一组加密程序。在一些实施例中,如在英国专利申请号1718505.9中描述的那些运算电路优化技术可以用于减少工作者确定智能合约的结果所需的资源。

[0056] 在一个实施例中,密钥生成器204是包括可执行代码的硬件、软件或其组合,该可执行代码如果由计算机系统的一个或多个处理器执行,则使得系统生成评估密钥和验证密钥,以形成二次程序。在Gennaro,R.等人(2013年)的"二次跨程序和无PCPs的简洁NIZKs(Quadratic Span Programs and Succinct NIZKs without PCPs)"中设想了将计算编码为二次程序的技术。在一个实施例中,二次算术问题(QAP)Q对域F上的电路$\mathcal{C}$进行编码,并且包含一组m+1多项式:

[0057] $V=\{v_k(x)\}$,$W=\{w_k(x)\}$,$Y=\{y_k(x)\}$

[0058] 其中,$0\leq k\leq m$。此外,也定义了目标多项式t(x)。给定函数f,其将F的n个元素作为输入并输出n'个元素,其中,$N=n+n'$。那么,如果$\{c_1,\cdots,c_N\}\in F^N$是f的输入和输出组的有效赋值并且存在系数列表$\{c_{N+1},\cdots,c_m\}$,则Q计算f,使得t(x)除以p(x):

[0059]
$$p(x) = \left(v_0(x)+\sum_{k=1}^{m} c_k \cdot v_k(x)\right)\cdot\left(w_0(x)+\sum_{k=1}^{m} c_k \cdot w_k(x)\right) - \left(y_0(x)+\sum_{k=1}^{m} c_k \cdot y_k(x)\right)$$

[0060] 因此,在一个实施例中,必须存在某个多项式h(x),使得$h(x)\cdot t(x)=p(x)$。Q的大小是m,其次数是t(x)的次数。

[0061] 在一个实施例中,为运算电路构建QAP包括为电路中的每个乘法门g选取任意根$r_g\in F$,并将目标多项式定义为$t(x)=\prod_g(x-r_g)$。在一个实施例中,索引$k\in\{1...m\}$与电路的每个输入和乘法门的每个输出相关联。V中的多项式对每个门的左输入进行编码,W对每个门的右输入进行编码,Y对输出进行编码。例如,如果第k条导线是门g的左输入,则$v_k(r_g)=$

1,否则$v_k(r_g)=0$。因此,对于特定的门g及其根r_g,前面的方程式可以如下简化:

[0062] $$\left(\sum_{k=1}^{m} c_k \cdot v_k(r_g)\right)\cdot\left(\sum_{k=1}^{m} c_k \cdot w_k(r_g)\right)$$
$$=\left(\sum_{k\in I_{left}} c_k\right)\cdot\left(\sum_{k\in I_{right}} c_k\right) = c_g y_k(r_g) = c_g$$

[0063] 门的输出值等于其输入的乘积。整除性检验分解成deg(t(x))个单独的检验,t(x)的每个门g和根r_g具有一个检验,使得$p(r_g)=0$。加法门和乘以常数门对QAP的大小和次数没有影响。

[0064] 在一个实施例中,在域Fp上定义QAP,其中,p是大质数。在一个实施例中,期望Fp上的QAP有效地计算任何可用模p的加法和乘法来表示的函数。运算分离门可以被设计成将运算导线$a\in Fp$(已知在$[0,2^{k-1}]$中)转换成k个二进制输出导线。因此,由此可见,可以使用运算门来表示布尔函数。例如,NAND(a,b)=1-ab。每个嵌入式布尔门只需要一次乘法运算。此外,诸如分离(split)之类的新门可以被定义为独立的,并与其他门组合在一起。给定输入$a\in F_p$(已知在$[0,2^{k-1}]$中),分离门输出k个导线,该k个导线保持a的二进制数字$a_1,\ldots,a_k$,使得$\sum^k 2^{i-1}a_i=a$,每个a_i为0或1。

[0065] 最后,系统生成所有证明者和验证者使用的公共参数作为设置阶段的一部分。应当注意的是,评估密钥E_K和验证密钥V_K是使用客户端选择的秘密值导出的。密钥生成器204可以结合密钥生成算法利用二次算术程序(QAP)来生成评估密钥E_K 210和验证密钥V_K 212。

[0066] 在一个实施例中,执行计算任务包括由工作者计算输入216上的函数(即,用于评估f(x)的过程)。在一个实施例中,工作者是客户端可以向其委托计算任务的任何合适的计算机系统。在一个实施例中,输入216包括证明工作者身份的信息,例如,使用与工作者相关联的私钥生成的数字签名。在一个实施例中,工作者是客户端为成功的计算支付费用(例如,通过数字资产的转移)的计算机系统。在一个实施例中,客户端向工作者提供输入x和评估密钥E_K,工作者将评估模块214用于计算程序来计算输出y(即,y=f(x),其中,输入是x,函数是f),并且使用评估密钥E_K 210来产生正确性证明218。在一个实施例中,评估模块是包括指令的硬件和/或软件,该指令如果由计算机系统的一个或多个处理器执行,则使得计算机系统评估QAP 208的内部电路导线的值并产生QAP的输出y。

[0067] 在一个实施例中,二次程序的每个多项式$v_k(x)\in F$被映射到双线性组中的元素$g^{v_k(s)}$,其中,s是由客户端选择的秘密值,g是该组的生成器,F是g的离散对数的域。在一个实施例中,对于给定的输入,工作者评估电路,以获得对应于二次程序的系数c_i的输出和内部电路导线的值。因此,工作者可以评估$v(s)=\sum_{k\in\{m\}} c_k \cdot v_k(s)$,以获得$g^{v(s)}$;计算w(s)和y(s);计算$h(x)=p(x)/t(x)=\sum^d h_i \cdot x^i$;并使用评估密钥中的$h_i$和$g^{s(i)}$项计算$g^{h(s)}$。在一个实施例中,正确性证明218包括$(g^{v(s)},g^{w(s)},g^{y(s)},g^{h(s)})$,并且验证者使用双线性映射来检验$p(s)=h(s)\cdot t(s)$..在一个实施例中,证明π存储在区块链222上,以供以后使用,或者可以由多方验证而无需证明者单独与这些方中的每一个交互。在一个实施例中,可以执行对正确性证明的电路存储的评估,以解锁由交易的锁定脚本锁定(encumber)的数字资产。

[0068] 在一个实施例中,证明π被广播到区块链网络,并且验证者220被用于验证证明。在一个实施例中,验证者220是任何合适的计算实体,例如,区块链上的节点。还应注意的是,

在某些情况下,生成评估密钥E_K和验证密钥V_K的同一计算实体也验证该证明。在一个实施例中,区块链的节点可以使用验证密钥V_K和证明π来验证支付交易,从而在验证成功的情况下验证合约。该协议的一个要求是,即使工作者知道验证密钥V_K,也不能提供不正确的证明。因此,在该协议中,公共参考串(CRS)由客户端或至少公布了评估密钥E_K和验证密钥V_K的可信第三方产生。在一个实施例中,任何计算实体都可以使用公布的验证密钥V_K来验证计算。

[0069] 通过使用本文描述的技术,客户端能够部分混淆交易数据,例如,区块链交易的接收者的身份。在一个实施例中,解锁脚本不公开接收者的地址和接收者的公钥。然而,在一些情况下,交易的价值(例如,转移的数字资产的数量)对于区块链网络的节点可能是可见的。在一个实施例中,客户端利用上文和下文描述的加密技术将锁定脚本转换成二次算术程序,工作者利用这些加密技术解决算术程序,以生成证明。

[0070] 一般来说,客户端能够使用标准交易,例如,P2PK和P2PKH,向交易对方或工作者付款。例如,在一个实施例中,客户端将P2PK锁定脚本转换成运算电路并广播支付交易,该支付交易包括从该电路导出的难题。交易对方或工作者接收该电路,提供恰当的输入(例如,证明工作者身份的信息(例如,在客户端和工作者之间共享的秘密或使用工作者的私钥生成的数字签名)),并运行电路,以生成正确性证明π。在一个实施例中,证明被用于解锁数字资产,此外,可能的情况是,识别交易对方或工作者的信息(例如,与交易对方或工作者相关联的公钥和/或数字签名)没有以清晰的格式记录到区块链。

[0071] 在一个实施例中,根据上文和/或下文描述的技术生成验证密钥和相应的证明。因此,给验证者提供验证密钥V_K和证明π:

[0072] $$V_K = \left\{ \begin{array}{c} \mathcal{P} \\ \mathcal{Q} \\ \alpha_v \mathcal{Q} \\ \alpha_w \mathcal{Q} \\ \alpha_w \mathcal{P} \\ \alpha_y \mathcal{Q} \\ \beta \mathcal{P} \\ \beta \mathcal{Q} \\ r_y t(s) \mathcal{P} \\ r_v v_i(s) \mathcal{P} \\ r_w w_i(s) \mathcal{Q} \\ r_y y_i(s) \mathcal{P} \end{array} \right\}_{i=0..N}$$

[0073] $$Proof\ \pi = \left\{ \begin{array}{c} \sum_{i=N+1}^{m} a_i r_v v_i(s)\mathcal{P} \\ \sum_{i=N+1}^{m} a_i \alpha_v r_v v_i(s)\mathcal{P} \\ \sum_{i=N+1}^{m} a_i r_w w_i(s)\mathcal{Q} \\ \sum_{i=N+1}^{m} a_i \alpha_w r_w w_i(s)\mathcal{P} \\ \sum_{i=N+1}^{m} a_i r_y y_i(s)\mathcal{P} \\ \sum_{i=N+1}^{m} a_i \alpha_y r_y y_i(s)\mathcal{P} \\ \sum_{i=N+1}^{m} a_i (r_v \beta v_i(s) + r_w \beta w_i(s) + r_y \beta y_i(s))\mathcal{P} \\ \sum_{i=0}^{d} h_i s^i \mathcal{Q} \end{array} \right\}$$

[0074] 使得验证者计算多个椭圆曲线乘法(例如,对每个公共输入变量使用一个椭圆曲线乘法)和五对检验,其中一对检验包括附加的配对乘法。

[0075] 给定验证密钥V_K、证明π和$(a_1, a_2, \cdots, a_N)$,以验证t(x)除以p(x),由此$(x_{N+1}, \cdots, x_m) = f(x_0, \cdots, x_N)$,验证者进行如下操作。首先,检验全部的三个α项目:

[0076] $$e(\alpha_v r_v V_{\text{mid}}(s)\mathcal{P}, \mathcal{Q}) = e(r_v V_{\text{mid}}(s)\mathcal{P}, \alpha_v \mathcal{Q})$$

[0077] $$e(\alpha_w r_w W_{\text{mid}}(s)\mathcal{P}, \mathcal{Q}) = e(\alpha_w \mathcal{P}, r_w W_{\text{mid}}(s)\mathcal{Q})$$

[0078] $$e\left(\alpha_y r_y Y_{\text{mid}}(s)\mathcal{P}, \mathcal{Q}\right) = e\left(r_y Y_{\text{mid}}(s)\mathcal{P}, \alpha_y \mathcal{Q}\right)$$

[0079] 其中,$V_{\text{mid}}(s) = \sum_{i=N+1}^{m} a_i v_i(s)$,$W_{\text{mid}}(s) = \sum_{i=N+1}^{m} a_i w_i(s)$并且$Y_{\text{mid}}(s) = \sum_{i=N+1}^{m} a_i y_i(s)$。然后,验证者检验项目β:

[0080] $$e\left(r_v V_{\text{mid}}(s)\mathcal{P} + r_y Y_{\text{mid}}(s)\mathcal{P}, \beta\mathcal{Q}\right) \cdot e(\beta\mathcal{P}, r_w W_{\text{mid}}(s)\mathcal{Q}) = e(Z_{\text{mid}}(s)\mathcal{P}, \mathcal{Q})$$

[0081] 并且$Z_{\text{mid}}(s) = \sum_{i=N+1}^{m} a_i \left(r_v \beta v_i(s) + r_w \beta w_i(s) + r_y \beta y_i(s)\right)$。最后,验证者检验整除性要求:

[0082] $$e(r_v V(s)\mathcal{P}, r_w W(s)\mathcal{Q}) = e\left(r_y Y(s)\mathcal{P}, \mathcal{Q}\right) \cdot e\left(r_y t(s)\mathcal{P}, h(s)\mathcal{Q}\right)$$

[0083] 其中,$r_v V(s)\mathcal{P} = \sum_{i=0}^{m} r_v a_i v_i(s)\mathcal{P}$,$r_w W(s)\mathcal{Q} = \sum_{i=0}^{m} r_w a_i w(s)\mathcal{Q}$,$r_y Y(s)\mathcal{P} = \sum_{i=0}^{m} r_y a_i y(s)\mathcal{P}$,并且$h(s)\mathcal{Q} = \sum_{i=0}^{d} h_i \mathcal{Q}$。

[0084] 因此,根据实施例,在考虑来自上述部分的符号和本申请描述的示例后,验证包括以下元素的一组对检验:

[0085] $$e(\pi_2, V_K^2) = e(\pi_1, V_K^3)$$

[0086] $$e(\pi_4, V_K^2) = e(\ V_K^5, \pi_3,)$$

[0087] $$e(\pi_6, V_K^2) = e(\pi_5, V_K^6)$$

[0088] $$e\big((\pi_1 + \ \pi_6), V_K^2\big) = e(\pi_7, V_K^2)$$

[0089] $$e\Big(\big(a_0 V_K^{10} + \ a_1 V_K^{11} + \ a_2 V_K^{12} + \ a_3 V_K^{13} + \ a_4 V_K^{14} + \ \pi_2 + \ a_7 V_K^{15} \ \big), \big(a_0 V_K^{16} + \ a_1 V_K^{17} + \ a_2 V_K^{18} + \ a_3 V_K^{19} + \ a_4 V_K^{20} + \ \pi_4 + \ a_7 V_K^{21} \ \big)\Big)$$
$$= e\Big(\big(a_0 V_K^{22} + \ a_1 V_K^{23} + \ a_2 V_K^{24} + \ a_3 V_K^{25} + \ a_4 V_K^{26} + \ \pi_6 + \ a_7 V_K^{15} \ \big), V_K^2\Big) * e(V_K^9, \pi_8)$$

[0090] 图3示出了用于协调可验证计算的执行的示图300。客户端302、工作者304和验证者306可以是区块链网络的节点。客户端302可以是可包括可执行代码的任何合适的计算机系统，该可执行代码如果由计算机系统的一个或多个处理器执行，则使得计算机系统接收智能合约308。在一个实施例中，智能合约308是用高级编程语言编码的源代码，例如，C、C++或Java。在一个实施例中，诸如编译器、解释器和/或汇编器之类的软件可以被用来将智能合约308转换成运算电路310，该运算电路310由携带来自域$\mathbb{F}$的值并连接到加法和乘法门的“导线”组成。应当注意的是，运算电路是指可以由包括一系列物理门的物理电路实现的逻辑电路(例如，使用晶体管-晶体管逻辑(TTL)集成电路(例如，7400系列门、触发器、缓冲器、解码器、多路复用器等))，这一系列物理门通过物理导线连接。

[0091] 在一个实施例中，客户端302向工作者304提供运算电路310和电路的输入312。电路310可用于生成二次程序Q，该二次程序Q包括一组提供原始电路完整描述的多项式。在任一情况下，工作者304可以在输入312上执行电路$\mathcal{C}$或二次程序Q，以生成一个或多个输出314。在一些实施例中，期望工作者(即，证明者)获得有效抄本$\{\mathcal{C}, x, y\}$作为输出，该抄本是对电路导线的赋值，使得赋值给输入导线的值是x，中间值对应于$\mathcal{C}$中的每个门的正确操作，赋值给输出导线的值是y；如果所要求的输出是不正确的(即，$y \neq \mathcal{P}(x)$)，则不存在$\{\mathcal{C}, x, y\}$的有效抄本。在一个实施例中，期望工作者提供电路导线的值的子集，其中，所选择的电路导线的值的子集对于工作者来说不是提前已知的。

[0092] 在实施例中，输出y、内部电路导线(或其子集)的值和评估密钥E_K用于产生正确性证明316。该证明π可以存储在区块链上并由多方验证，而无需工作者304单独与多方交互。以这种方式，验证者306可以使用公共验证密钥V_K和证明π来验证支付交易，从而验证合约。在一些情况下，如果验证失败，则客户端302可以收回支付交易锁定(encumber)的数字资产。在一些情况下，验证者306和客户端302是相同的计算机系统。

[0093] 图4示出了本申请的实施例的示图400。具体地，图4描绘了以各种格式编码的椭圆曲线402上的点，其中这些格式适用于基于区块链的脚本。

[0094] 在各种实施例中，椭圆曲线点可以在与交易相关联地执行的锁定和解锁脚本中编

码。可以用基于堆栈的脚本语言编写这些脚本。例如,验证密钥V_K可以包括$\{V_K^1, V_K^2, V_K^3, .., V_K^n\}$,证明π可以包括一组元素$\{\pi_1, ..., \pi_8\}$,其中,$V_K^i$和$\pi_j$是有限域$\mathbb{F}_p$,$E(\mathbb{F}_p)$上的椭圆曲线上的点。图4示出了示例图400,其中,椭圆曲线402上的点使用压缩(P_C)或未压缩(P_U)编码来表示,该编码适用于与例如锁定或解锁脚本等区块链脚本一起使用。

[0095] 在一个实施例中,假设$P \in E(\mathbb{F}_p)$是椭圆曲线上的点。如果$P \neq \mathcal{O}$,则该点由其仿射坐标表示。在未压缩编码404中,点P由指示未压缩编码的信息(例如,前缀406(例如,下文描述的参数'C'))和两个域元素(x和y坐标408和410)来表示,而在压缩编码中,该点仅由其x坐标和唯一标识y坐标的附加位来表示。因此,在一个实施例中,P_U=C||X||Y,其中:

[0096] C=0x04

[0097] X=FieldElements2OctetString(x)

[0098] Y=FieldElements2OctetString(y)

[0099] 其中,双管运算符"||"是指串联操作,FieldElements2OctetString()函数可用于将域(例如,有限域)的元素转换为正好包含八位的八位位组串。关于压缩编码,P_C=C||X,其中:

[0100] $$C = \begin{cases} 0x02 & \text{if y even} \\ 0x03 & \text{if y odd} \end{cases}$$

[0101] X=FieldElements2OctetString(x)

[0102] 在一个实施例中,未压缩编码404包括有限域(即$P \in E(\mathbb{F}_p)$)的点P,例如,椭圆曲线上的点的x坐标408和y坐标410。未压缩编码404还包括可用于确定编码是未压缩编码的信息(例如,在x坐标408和y坐标410前加/后加的前缀406值)。相反,压缩编码412包括有限域的点P的编码(例如,如上所述),以及可用于确定编码是压缩编码的信息。例如,压缩编码412A编码前缀414和点P的x坐标408,其中,至少部分地基于前缀414来确定相应的y坐标。例如,前缀414表示y坐标是偶数。相反,第二压缩编码412B编码不同的前缀416和点P的x坐标408,并用于确定奇数的y坐标。

[0103] 应当注意的是,一般来说,未压缩点P_U可以以任何合适的格式表示,该格式可以编码用于确定P_U是未压缩椭圆曲线点的信息、该点的x坐标和该点的y坐标。类似地,压缩点P_C可以以任何合适的格式表示,该格式可以编码用于确定P_C是压缩椭圆曲线点的信息和用于确定该点的x坐标和y坐标的压缩信息。

[0104] 作为一个示例,考虑secp256k1椭圆曲线的点的表示。在表示未压缩点的实施例中,未压缩椭圆曲线P的编码位串中的第一个八位字节是0x04,其后是对应于点的X坐标和点的Y坐标的两个256位数字(即P=C||X||Y)。作为一个示例,尽管在本申请的范围内还考虑了其他合适的编码格式,例如,基本编码规则(BER)和规范编码规则(CER),但是使用了可辨别编码规则(DER)编码格式。

[0105] 在一个实施例中,包括椭圆曲线点的长度的单字节脚本操作码连接到实际点(例如,将该长度连接到实际点的前面)。例如,如果:

[0106] x=0xe3b01684a8a8b66f8e44203db5869b4dcb74a0afc905ae9197ed74a8d6cecdcc

[0107] y＝0x6424d186a23687532c8b20911defc2f42c93749b3736857912c6abe2dc3f01d1

[0108] 则在一个实施例中,压缩和未压缩脚本P_C和P_U分别是:

[0109] 脚本:P_C:0x21||

[0110] 0x03e3b01684a8a8b66f8e44203db5869b4dcb74a0afc905ae9197ed74a8d6cecdcc

[0111] 脚本:P_U:0x41||

[0112] 0x04e3b01684a8a8b66f8e44203db5869b4dcb74a0afc905ae9197ed74a8d6cecdcc6424d186a23687532c8b20911defc2f42c93749b3736857912c6abe2dc3f01d1

[0113] 在一个实施例中,客户端是智能合约的一方,并且确定利用区块链来执行合约。作为确定的一部分,客户端向区块链发布支付转移,以将客户端控制的数字资产转移到公布证明π的一方(例如,工作者)。如上所述的验证者计算机系统可以执行验证程序,来确定证明π是正确的。

[0114] 根据协议,客户端可以为产生和生成证明π而生成公共参考串。在一些实施例中,公共参考串由可信方(例如,第三方)生成。应当注意的是,虽然可信第三方可以生成公共参考串,但是将可信第三方引入协议是可选的-该协议需要客户端和工作者,并且可以在没有额外的第三方的情况下执行。

[0115] 继续该协议,计算的验证作为交易验证阶段的一部分进行。在一个实施例中,客户端或工作者被限制为提供$\{V_K^1, V_K^2, V_K^3, .., V_K^n\}$作为锁定脚本和/或解锁脚本的一部分。可以通过多种方式实现这种约束。例如,在一个实施例中,工作者通过注入串行化解锁交易来提供验证密钥V_K,该串行化解锁交易在其一个解锁脚本中包含有$\{V_K^1, V_K^2, V_K^3, .., V_K^n\}$。作为第二示例,客户端广播交易,其中,锁定脚本包括$\{V_K^1, V_K^2, V_K^3, .., V_K^n\}$和在验证阶段使用的方程式,并创建由包含验证密钥$V_K$的SIGHASH_NONE|SIGHASH_ANYONECANPAY签名的交易输入。客户端将该输入传递给需要添加$\{\pi_1, .., \pi_8\}$、签名和广播的工作者。作为又一示例,工作者通过使用固定长度的脚本散列(例如,20字节的脚本散列)来提供验证密钥V_K,该固定长度的脚本散列在为执行付费的交易的锁定脚本中。当然,这些仅仅是如何提供$\{V_K^1, V_K^2, V_K^3, .., V_K^n\}$作为锁定脚本、解锁脚本或任何其他合适的脚本或操作的一部分的说明性示例,根据任何合适的区块链协议来执行这些脚本或操作作为交易验证的一部分。在一个实施例中,工作者提供$\{\pi_1, .., \pi_8\}$作为解锁脚本的一部分,并且如果证明π满足输出脚本中设置的条件,则授权证明π。

[0116] 在一个实施例中,基于区块链的系统支持根据Script等脚本语言执行操作(也称为操作码和命令)。在一个实施例中,交易包括锁定(输出)和解锁(输入)脚本,并且可以被识别为标准类型列表中的一个。例如,有五种标准类型:支付给公钥(P2PK)、支付给公钥散列(P2PKH)、多重签名、支付给脚本散列(P2SH)和OP_RETURN。

[0117] 在运行本文描述的提议zk协议时,诸如公共参考串、证明π和/或其部分等信息存储在区块链上。此外,作为验证过程的一部分,也可以是系统在交易验证过程中提取在堆栈上推送的特定元素并如上所述从验证阶段检查配对的情况。

[0118] 交易可包括嵌入到其输入和输出中的称为脚本的小程序,这些小程序指定了如何以及由谁来访问交易的输出。使用基于堆栈的脚本语言编写这些脚本。在一个实施例中,协

议包括对交易的各种技术规则和基于语法的限制,这些规则和限制确定交易是标准交易还是非标准交易。在一个实施例中,对在执行堆栈上推送的元素的大小和输入脚本的总大小有限制。例如,在执行堆栈上推送的每个元素被限制为520字节,每个输入脚本被限制为1650字节。在一个实施例中,在脚本执行之后,堆栈恰好包括一个非假元素。输入脚本不能包含除OP_PUSHDATA之外的任何OP代码(赎回脚本部分除外)。在一个实施例中,应当注意,不同的区块链系统可能将待包含的不同数量和/或单位的数字资产定义为输出值。在一个实施例中,最小输出值为零(即,不需要转移数字资产)。偏离这些规则的交易被认为是非标准的。

[0119] 作为验证阶段的一部分,验证者(例如,客户端)提取验证密钥V_K和证明π的元素,并在配对检验中使用。在一个实施例中,验证检验包括满足一组约束的双线性映射。作为一个示例,考虑满足形式$e(V_K^1, \pi_1 + \pi_2) = e(V_K^1, \pi_1) * e\ (V_K^1, \pi_2)$的约束的双线性映射e(x,y)=$2^{xy}$,如下所示:

[0120] $e(3,4+5) = 2^{3*9} = 2^{27}$

[0121] $e(3,4)*e(3,5) = 2^{3*4}2^{3*5} = 2^{12}2^{15} = 2^{27}$

[0122] 应当注意的是,上述这种配对是出于说明的目的,尽管对于在加密系统中的应用来说,这种逐字映射可能不是很有用,但是应当注意的是,验证双线性约束的过程是相同的,其中,双线性映射在椭圆曲线上扩展。同样,出于说明的目的,下面示出了一个示例,说明如何使用脚本在验证阶段从公共参考串(例如,验证密钥V_K)和证明π中提取和使用信息:

[0123]

验证密钥	$V_K = \{V_K^1, V_K^2, V_K^3, V_K^4, V_K^5, V_K^6\}$
证明	$\pi = \{\pi_1, \pi_2, \pi_{3,,}.., \pi_8\}$
验证检验	$e(\pi_2, V_K^2) = e(\pi_1, V_K^3)$ $e(\pi_4, V_K^2) = e(\ V_K^5, \pi_3,)$ $e(\pi_6, V_K^2) = e(\pi_5, V_K^6)$ ...

[0124] 应当注意的是,上面提供的示例仅仅用于说明配对检验的示例,并且可以包括附加约束,例如,上面结合图2讨论的那些约束。

[0125] 有多种技术可以提供验证密钥V_K用于验证证明π。图5是说明性示图500,其中,客户端510提供验证密钥V_K,用于验证证明π。在一个实施例中,客户端创建交易输出502,其中,该交易包括输出值504(例如,向执行合约的工作者支付的数字资产)和锁定脚本508,或与输出值504和锁定脚本508相关联,该锁定脚本508执行包括满足一组约束的双线性映射的验证检验。在一个实施例中,锁定脚本至少部分基于客户端的数字签名来验证验证密钥V_K的完整性。在某些情况下,锁定脚本可以规定,如果验证失败,则客户端能够收回该交易相关的上述数字资产。在一些区块链系统中,交易在与锁定脚本相关联的情况下编码指示锁定脚本的大小(例如,以字节为单位)的参数,该参数可以称为锁定脚本大小506,如结合图5所示。

[0126] 作为一个示例,可以基于以下内容来描述锁定脚本:

[0127] OP_IF

[0128] //验证V_K的完整性

[0129] <PubKey Alice>OP_CHECKSIGVERIFY

[0130] OP_0OP_PICK OP_DUP<V_K^6>OP_EQUALVERIFY

[0131] OP_1OP_SUB OP_PICK OP_DUP<V_K^5>OP_EQUALVERIFY

[0132] OP_(|V_K|-i)OP_SUB OP_PICK OP_DUP<V_K^i>OP_EQUALVERIFY

[0133] //验证配对

[0134] //$e(\pi_2, V_K^2) = e(\pi_1, V_K^3)$

[0135] //左部

[0136] OP_DEPTH OP_3OP_SUB OP_PICK

[0137] OP_5OP_PICK OP_PAIRING OP_TOALTSTACK

[0138] //右部

[0139] OP_DEPTH OP_2OP_SUB OP_PICK

[0140] OP_4OP_PICK OP_PAIRING OP_TOALTSTACK

[0141] //空栈

[0142] 14*times{OP_DROP}

[0143] OP_FROMALTSTACK OP_FROMALTSTACK OP_EQUALVERIFY[1]

[0144] //针对$e(\pi_i, V_K^j)$

[0145] OP_DEPTH OP_(i+1)OP_SUB OP_PICK

[0146] OP_(|V_K|-j+1)OP_PICK

[0147] OP_PAIRING OP_TOALTSTACK

[0148] //针对$e(V_K^j, \pi_i)$

[0149] OP_(|V_K|-j)OP_PICK

[0150] OP_DEPTH OP_(i+1)OP_SUB OP_PICK

[0151] OP_PAIRING OP_TOALTSTACK

[0152] <PubKey Bob>OP_CHECKSIG

[0153] OP_ELSE

[0154] <n days>OP_CHECKSEQUENCEVERIFY

[0155] <PubKey Alice>OP_CHECKSIG

[0156] OP_ENDIF

[0157] 应当注意的是,上面提供的示例是对锁定脚本的描述,而不一定是对锁定脚本508本身的逐字描述。例如,上面在括号中描述的一些域(例如,“<PubKey Client>”,其可以指客户端的公钥)没有逐字地包含在锁定脚本508中。类似地,括号中的一些域是以数学计算为基础,例如,如上所述,“OP_(i+1)”可能不逐字地包含在锁定脚本中,而是指至少部分地基于数学计算而确定的操作码或命令,在这种情况下,“i”是指证明π的元素的数量,因此,在$\pi = \{\pi_1, .., \pi_8\}$的示例中,“OP_(i+1)”可以在锁定脚本中以基于堆栈的脚本语言表示为“OP_7”。还应当注意的是,在上面的双斜线“//”后面的文本是指与可执行代码不对应的注释。例如,“//验证配对”并不映射到可执行代码,而只是向人们表明,在注释之后的文本是

用来执行配对的验证。

[0158] 应当再次注意的是,如上所述的交易输出502仅仅是说明性的,并且可以存在这样的变化即交易可以包括采用任何合适的衡量单位的任何合适的支付量,并且可以存在各种锁定脚本,例如,如上所述利用锁定脚本的一些或全部功能的锁定脚本。一般来说,锁定脚本508可以是执行一组配对检验的任何合适的一组命令。在一个实施例中,锁定脚本508包括使用与客户端相关联的至少一个非对称公钥对验证密钥V_K进行完整性检验。在一个实施例中,如果验证失败,则锁定脚本允许客户端收回数字资产。在一个实施例中,锁定脚本508包括在阈值持续时间之后将交易标记为无效的指令(例如,对应于“<n days>”的值,如上所述)。

[0159] 客户端510可以是本申请中其他地方描述的那些客户端,其可以创建引用上述交易502的交易输入512,并将其传输给交易对方(例如,意图计算有效证明π的工作者)。在一个实施例中,交易输入512可以编码与交易输出502相关联的标识符514、与交易输出502相关联的交易输出索引516(例如,在一些情况下,该索引是基于零的)、解锁脚本520、指示解锁脚本大小(例如,以字节为单位)的参数(其可以称为解锁脚本大小518)、序列号522及其任何合适的组合。例如,在一些实施例中,解锁脚本大小没有在交易输入512中被明确编码,而是可以导出的(例如,通过检测指示解锁脚本结束的特定终止序列)。在一个实施例中,解锁脚本518包括验证密钥V_K 524和与客户端相关联的数字签名526。在一些情况下,解锁脚本518包括附加信息,例如,控制验证过程执行的分支信息的指示。

[0160] 作为一个示例,可以基于以下内容来描述解锁脚本:

[0161] $<V_K^1> <V_K^2> <V_K^3> <V_K^4> <V_K^5> <V_K^6> <\text{Sig A}> 1$

[0162] 应当注意的是,上面提供的示例是对解锁脚本的描述,而不一定是对解锁脚本本身的逐字描述。例如,上面括号中描述的一些域没有逐字包含在解锁脚本518中。回到上面描述解锁脚本的示例,解锁脚本518可以包括验证密钥V_K 524的有序或无序元素序列、与客户端相关联的数字签名526和分支信息。在一个实施例中,数字签名526包括散列类型标志。在一个实施例中,散列标志是SIGHASH_NONE|SIGHASH_ANYONECANPAY。解锁脚本518可以包括执行控制信息,该执行控制信息可与锁定脚本结合使用来验证交易。例如,在上述示例中,“1”表示脚本应该进入解锁脚本中的控制语句的第一分支(例如,OP_IF分支,而不是OP_ELSE分支)。

[0163] 在一个实施例中,工作者528计算证明π530,并且将该证明与和工作者相关联的数字签名532附加到解锁脚本518,并且广播该交易(例如,向客户端和/或区块链网络的一个或多个节点广播)。

[0164] 作为一个示例,可以基于以下内容来描述具有证明和工作者数字签名的解锁脚本:

[0165] $<\text{Sig B}> <\pi_1> <\pi_2> ..<\pi_8> <V_K^1> <V_K^2> <V_K^3> <V_K^4> <V_K^5> <V_K^6> <\text{Sig A}> 1$

[0166] 因此,在一个实施例中,根据以上描述的解锁脚本能够满足锁定脚本设置在输出上的条件,并且可以允许花费数字资产(例如,如输出值所示)。

[0167] 有多种技术可以提供验证密钥V_K,用于验证证明π。图6是说明性示图600,其中,工作者提供验证密钥V_K,用于验证证明π。交易输出602可包括输出值604、锁定脚本606(可选

地,在一些系统中)和锁定脚本608。在一个实施例中,交易输出612包括交易ID、输出索引616、解锁脚本大小618、解锁脚本618和序列号622,如上面结合图5所述。应当注意的是,虽然图6中描述的交易的结构可以依照图5中描述的结构,但是图6可特别包括根据P2SH交易的锁定脚本608和解锁脚本618。客户端610可以生成验证密钥V_K和赎回脚本628,并将其提供给工作者624。

[0168] 在一个实施例中,区块链系统支持各种类型的交易。在一个实施例中,所支持的交易(例如,标准交易)是基于脚本散列的交易。一般来说,基于脚本散列的交易是指其中验证解锁脚本的有效性包括提供与指定的散列值匹配的脚本的任何交易。例如,解锁脚本620包括赎回脚本628,锁定脚本608包括至少一个条件,即由解锁脚本提供的赎回脚本的散列与指定值匹配。例如,可以基于以下内容来描述锁定脚本:

[0169] OP_HASH160<20-byte hash of redeem script>OP_EQUAL

[0170] 在一些情况下,工作者624提供验证密钥V_K 626,用于验证阶段。在一个实施例中,检验验证阶段的解锁条件可以存储在赎回脚本中。在一个实施例中,P2SH交易的锁定脚本包括赎回脚本的散列,并且在一些情况下,赎回脚本是保密的(例如,由工作者加密),并且仅在指示进行转移输出值时才被披露。

[0171] 在一个实施例中,客户端610通过识别赎回脚本来创建P2SH未花费交易输出,并对其应用散列(例如,HASH160)。为了转移UTXO,客户端创建包括和/或引用赎回脚本的输入脚本(其引用UTXO)。应当注意的是,在一个实施例中,尽管基于区块链的系统可能对可包括在赎回脚本中的数据的类型或数量有限制,但赎回脚本是适于存储任意数据的。例如,对于可以发布到堆栈上的数据的大小有限制(例如,PUSHDATA操作被限制为520字节的数据)。在一个实施例中,赎回脚本分别包括压缩和未压缩的点P_C和P_U,如结合图4所述。因此,在一个实施例中,可存储在赎回脚本中的椭圆曲线点的最大数量是15个压缩点(例如,15个点*34字节/点=510字节)或7个未压缩点(例如,7个点*66字节/点=462字节)。在一个实施例中,解锁脚本618包括可以一起用于满足锁定脚本608中被编码的一组条件的证明630、赎回脚本628和命令(例如,操作码)。

[0172] 图7是根据实施例的用于生成赎回脚本的流程700的示意图。在一个实施例中,使用硬件、软件或其组合来实现流程700。执行该流程的合适系统包括工作者,该工作者提供与图6的讨论相关的P2SH解锁脚本。

[0173] 在一个实施例中,系统确定验证密钥V_K具有足够的基数。在一个实施例中,系统确定赎回脚本的大小(702),该赎回脚本包括验证密钥的每个元素和检验一组约束的脚本,其中,赎回脚本和证明π足以解锁相应的锁定脚本。系统可以确定赎回脚本的大小是否超过预定阈值(704),该阈值可以基于区块链协议施加的大小限值,例如,区块链协议可以要求赎回脚本的大小小于或等于520字节。如果赎回脚本具有足够的大小,则系统生成赎回脚本和解锁脚本(706),其中,赎回脚本包括验证密钥和检验一组约束的脚本,解锁脚本包括证明π,其中,赎回脚本和解锁脚本一起包括足以验证支付交易的一组命令。

[0174] 例如,在验证密钥V_K的基数小于16的情况下,可能是:

[0175]

赎回脚本	$<V_K^1>$ $<V_K^2>$ $<V_K^3>$ $<V_K^4>$ $<V_K^5>$ $<V_K^6>$ OP_DEPTH OP_3 OP_SUB OP_PICK OP_5 OP_PICK OP_PAIRING OP_TOALTSTACK OP_DEPTH OP_2 OP_SUB OP_PICK OP_4 OP_PICK OP_PAIRING OP_TOALTSTACK 14*{OP_DROP} OP_FROMALTSTACK OP_FROMALTSTACK OP_EQUALVERIFY **[1]** OP_DEPTH OP_5 OP_SUB OP_PICK OP_5 OP_PICK OP_PAIRING OP_TOALTSTACK OP_1 OP_PICK OP_DEPTH OP_3 OP_SUB OP_PICK OP_PAIRING OP_TOALTSTACK 14*{OP_DROP} OP_FROMALTSTACK OP_FROMALTSTACK OP_EQUALVERIFY **[2]**
解锁脚本	OP_1 $<\pi_1>$ $<\pi_2>$ ..$<\pi_8>$ <Redeem Script>
锁定脚本	OP_HASH160 <20-byte-hash of [Redeem Script]> OP_EQUAL

[0176] 其中,赎回脚本和解锁脚本足以解锁锁定脚本。

[0177] 如果赎回脚本超过预定阈值,则系统生成解锁脚本,该解锁脚本将验证密钥V_K的一个或多个元素存储在赎回脚本之前的解锁脚本的部分中(708),使得赎回脚本的总大小在阈值内,并且利用上述验证密钥的元素和约束生成赎回脚本(710)。在一个实施例中,可以在解锁脚本中的其他地方对验证密钥V_K的一个或多个元素进行编码。例如,在V_K的基数大于15的情况下,可能是:

[0178]

赎回脚本	OP_DUP OP_TOALTSTACK OP_HASH160 <20-byte-hash of V_K^6> OP_EQUALVERIFY OP_DUP OP_TOALTSTACK OP_HASH160 <20-byte-hash of V_K^5> OP_EQUALVERIFY OP_DUP OP_TOALTSTACK OP_HASH160 <20-byte-hash of V_K^4> OP_DUP OP_TOALTSTACK OP_HASH160 <20-byte-hash of V_K^3> OP_EQUALVERIFY OP_DUP OP_TOALTSTACK OP_HASH160 <20-byte-hash of V_K^2> OP_EQUALVERIFY OP_DUP OP_TOALTSTACK OP_HASH160 <20-byte-hash of V_K^1> OP_EQUALVERIFY OP_FROMALTSTACK OP_FROMALTSTACK OP_FROMALTSTACK OP_FROMALTSTACK FROMALTSTACK OP_FROMALTSTACK OP_DEPTH OP_3 OP_SUB OP_PICK OP_5 OP_PICK OP_PAIRING OP_TOALTSTACK OP_DEPTH OP_2 OP_SUB OP_PICK OP_4 OP_PICK OP_PAIRING OP_TOALTSTACK 14*{OP_DROP} OP_FROMALTSTACK OP_FROMALTSTACK OP_EQUALVERIFY **[1]**
解锁脚本	OP_1 $<\pi_1>$ $<\pi_2>$ ..$<\pi_8>$ $<V_K^1>$ $<V_K^2>$ $<V_K^3>$ $<V_K^4>$ $<V_K^5>$ $<V_K^6>$ $<Redeem\ Script>$
锁定脚本	OP_HASH160 <20-byte-hash of [Redeem Script]> OP_EQUAL

[0179] 其中,赎回脚本和解锁脚本足以解锁锁定脚本。

[0180] 因此,通过在锁定脚本中编码验证阶段所需的阶段,交易的验证应该成为具有zk协议的验证阶段的等效过程。在一个实施例中,可以如下使用OP_VERIFYPROOF操作码:

[0181]

类型	脚本
解锁脚本	OP_1 < π_1 > < π_2 > ..< π_8 > < V_K^1 > < V_K^2 > < V_K^3 > < V_K^4 > < V_K^5 > < V_K^6 >

锁定脚本	// 提取 π_i
	OP_DEPTH OP_(i+1) OP_SUB OP_PICK
	//提取V_K^j
	OP_(|V_K|-j + 1) OP_PICK
	//配对调用
	OP_PAIRING OP_TOALTSTACK
	//提取π_k
	OP_DEPTH OP_(k+1) OP_SUB OP_PICK
	//提取V_K^m
	OP_(|V_K|-m + 1) OP_PICK
	//配对调用
	OP_PAIRING OP_TOALTSTACK
	//空栈
	(|V_K| +| π |) * OP_DROP
	//验证操作的结果
	OP_FROMALTSTACK OP_FROMALTSTACK OP_EQUALVERIFY

[0182] 解锁和锁定脚本可以以任何合适的方式来实现。解锁脚本以任何合适的方式来实现，其中，解锁脚本编码<π_1>..<π_8>和验证密钥V_K。类似地，锁定脚本可以以任何合适的方式来实现，其中π_i和V_K^j被提取并配对调用被执行。在一个实施例中，如上所述的OP_PAIRING是支持具有高效双线性映射的椭圆曲线的操作码，例如，该曲线可以是由等式$y^2=x^3+3$定义的bn128曲线。

[0183] 图8是可用于实践本申请至少一个实施例的计算设备800的说明性简化框图。在各种实施例中，计算设备800可以用于实现上面示出和描述的任何系统。例如，计算设备800可以用于用作数据服务器、网页服务器、便携式计算设备、个人计算机或任何电子计算设备。如图8所示，计算设备800可以包括一个或多个处理器802，在一些实施例中，处理器802经由总线子系统804与多个外围子系统通信并可操作地耦合。在一些实施例中，这些外围子系统包括存储子系统806、一个或多个用户接口输入设备812、一个或多个用户接口输出设备814和网络接口子系统816，其中存储子系统806包括存储器子系统808和文件/磁盘存储子系统810。这种存储子系统806可以用于临时或长期存储信息。

[0184] 在一些实施例中，总线子系统804提供用于使计算设备800的各种组件和子系统能够如预期的那样彼此通信的机制。尽管总线子系统804被示意性地示为单个总线，但是总线子系统的替代实施例可以利用多个总线。在一些实施例中，网络接口子系统816给其他计算设备和网络提供接口。在一些实施例中，网络接口子系统816用作从其他系统接收数据且从计算设备800向其他系统发送数据的接口。在一些实施例中，总线子系统804用于传送诸如细节、搜索词等数据。

[0185] 在一些实施例中，用户接口输入设备812包括一个或多个用户输入设备，例如，键盘；指向设备，例如，集成鼠标、跟踪球、触摸面板或图形输入板；扫描仪；条形码扫描仪；包含到显示器中的触摸屏；音频输入设备，例如，语音识别系统、麦克风；以及其他类型的输入设备。通常，术语“输入设备”的使用旨在包括用于向计算设备800输入信息的所有可能类型

的设备和机构。在一些实施例中,一个或多个用户接口输出设备814包括显示子系统、打印机或非视觉显示器,例如,音频输出设备等。在一些实施例中,显示子系统包括阴极射线管(CRT)、平板设备(例如,液晶显示器(LCD)、发光二极管(LED)显示器或投影显示器或其他显示设备。通常,术语“输出设备”的使用旨在包括用于从计算设备800输出信息的所有可能类型的设备和机构。一个或多个用户接口输出设备814可以用于例如呈现用户界面,以便于当交互适当时用户与执行所描述的流程和其变形的应用程序进行交互。

[0186] 在一些实施例中,存储子系统806提供计算机可读存储介质,用于存储提供本申请至少一个实施例的功能的基本编程和数据结构。在一些实施例中,当由一个或多个处理器执行应用程序时,该应用程序(程序、代码模块、指令)提供本申请一个或多个实施例的功能,并且在实施例中,应用程序存储在存储子系统806中。这些应用模块或指令可以由一个或多个处理器802执行。在各种实施例中,存储子系统806另外提供用于存储根据本申请使用的数据的储存库。在一些实施例中,存储子系统806包括存储器子系统808和文件/磁盘存储子系统810。

[0187] 在一些实施例中,存储器子系统808包括多个存储器,例如,用于在程序执行期间存储指令和数据的主随机存取存储器(RAM)818和/或其中可以存储固定指令的只读存储器(ROM)820。在一些实施例中,文件/磁盘存储子系统810为程序和数据文件提供非暂时性持久(非易失性)存储,并且可以包括硬盘驱动器、软盘驱动器以及相关联的可移动介质、光盘只读存储器(CD-ROM)驱动器、光学驱动器、可移动介质盒或其他类似的存储介质。

[0188] 在一些实施例中,计算设备800包括至少一个本地时钟824。在一些实施例中,本地时钟824是表示从特定开始日期起经过的滴答数量的计数器,并且在一些实施例中,本地时钟824整体位于计算设备800内。在各种实施例中,本地时钟824用于以特定的时钟脉冲同步计算设备800及其包括的子系统的处理器中的数据传输,并且可以用于协调计算设备800和数据中心中的其他系统之间的同步操作。在另一实施例中,本地时钟是可编程间隔定时器。

[0189] 计算设备800可以是多种类型中的任何一种,包括便携式计算机设备、平板电脑、工作站或下面描述的任何其他设备。另外,计算设备800可以包括另一设备,在一些实施例中,该另一设备可以通过一个或多个端口(例如,USB、耳机插孔、闪电连接器等)连接到计算设备800。在一些实施例中,这种设备包括接受光纤连接器的端口。因此,在一些实施例中,该设备将光信号转换成电信号,并通过将该设备连接到计算设备800的端口传输这些电信号,以进行处理。由于计算机和网络的不断变化的性质,图8中描绘的计算设备800的描述仅旨在作为说明该设备的优选实施例的特定示例。具有比图8中描绘的系统更多或更少组件的许多其他配置是可能的。

[0190] 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列出的元件或步骤之外的其他元件或步骤的存在。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。在列举了若干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。

[0191] 本文引用的所有参考文献(包括出版物、专利申请和专利)通过引用并入本文,如同单独地和具体地指示每个参考文献通过引用并入本文,并且在本文中完整地阐述每个参考文献。

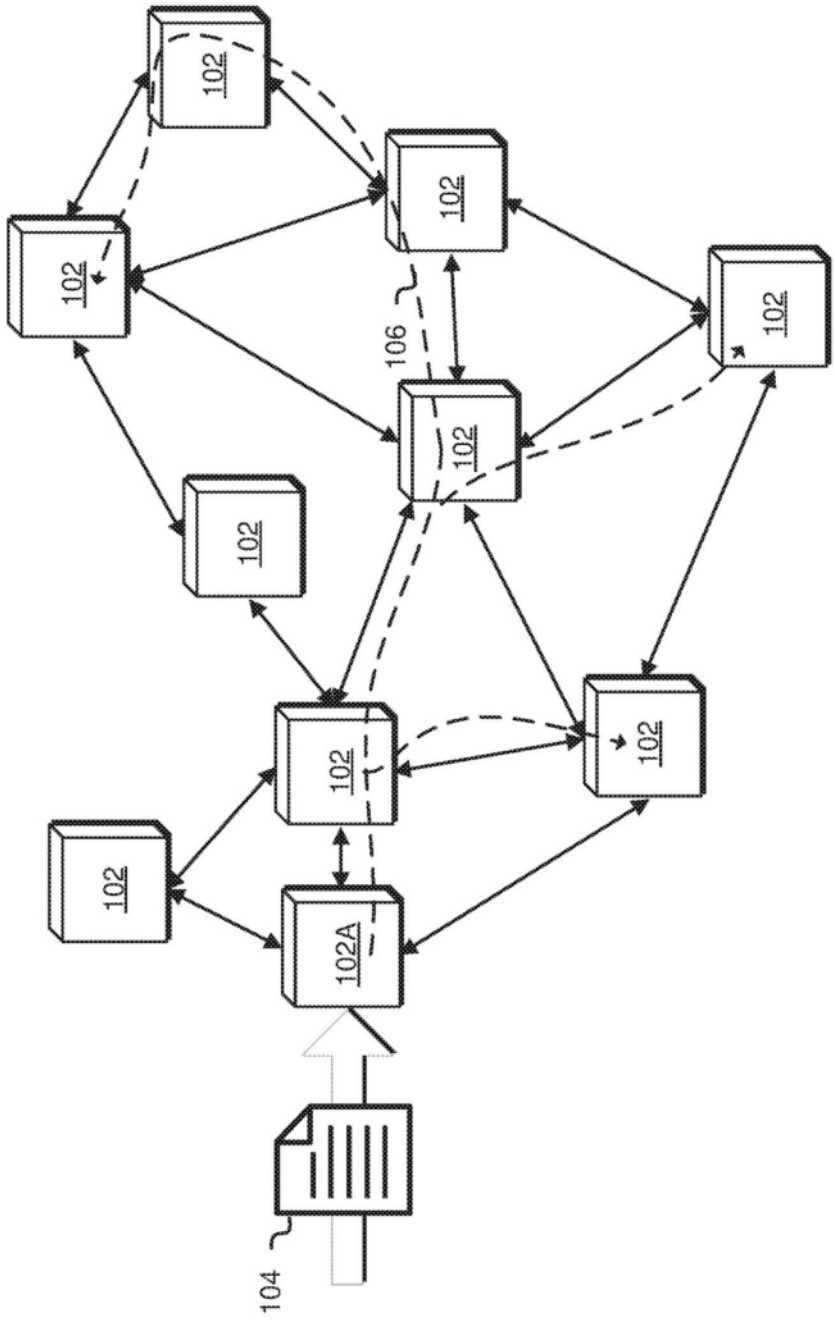


图1

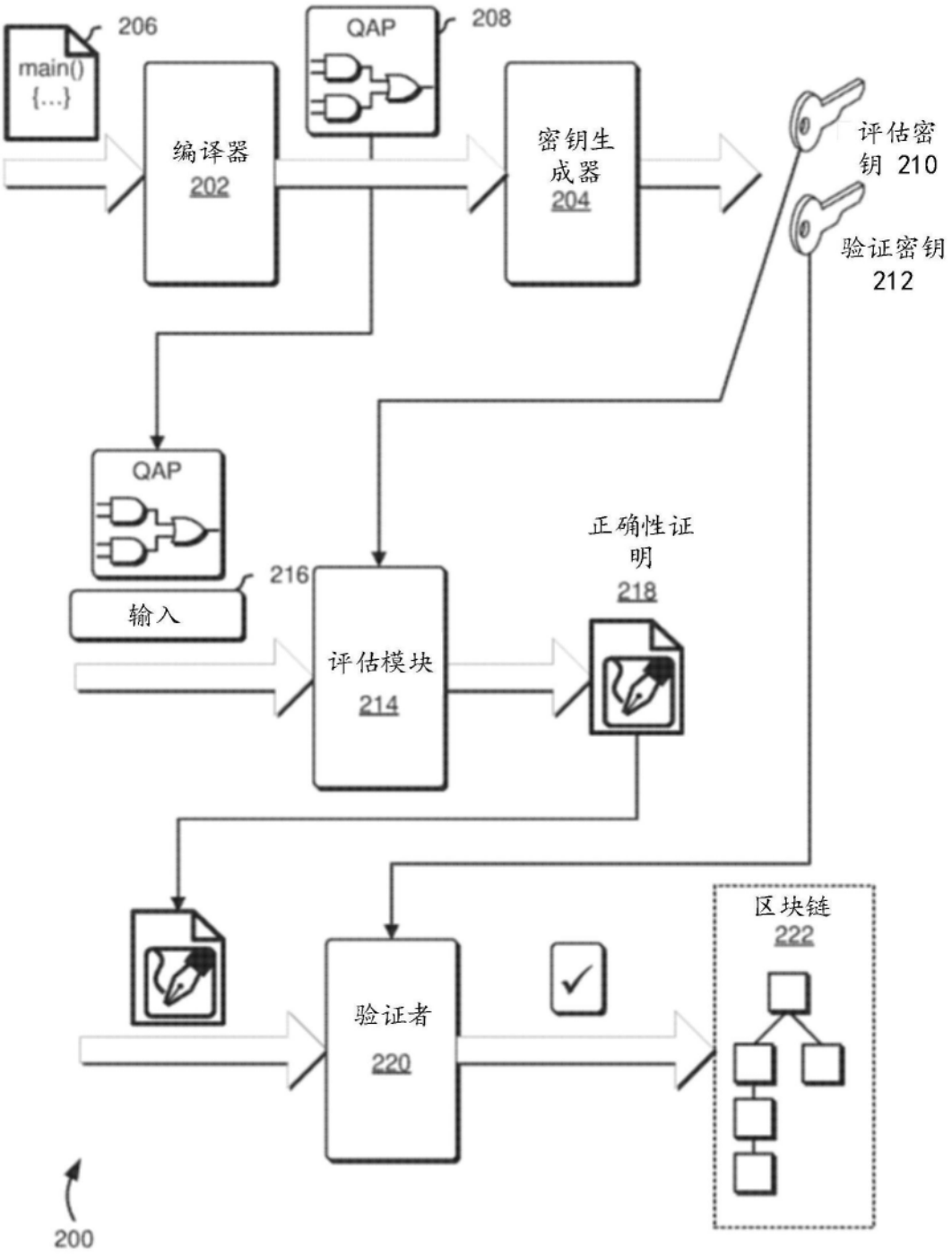

206
main()
{...}
编译器
202
QAP
208
密钥生
成器
204
评估密
钥 210
验证密钥
212
QAP
输入
216
评估模块
214
正确性证
明
218
区块链
222
验证者
220
200

图2

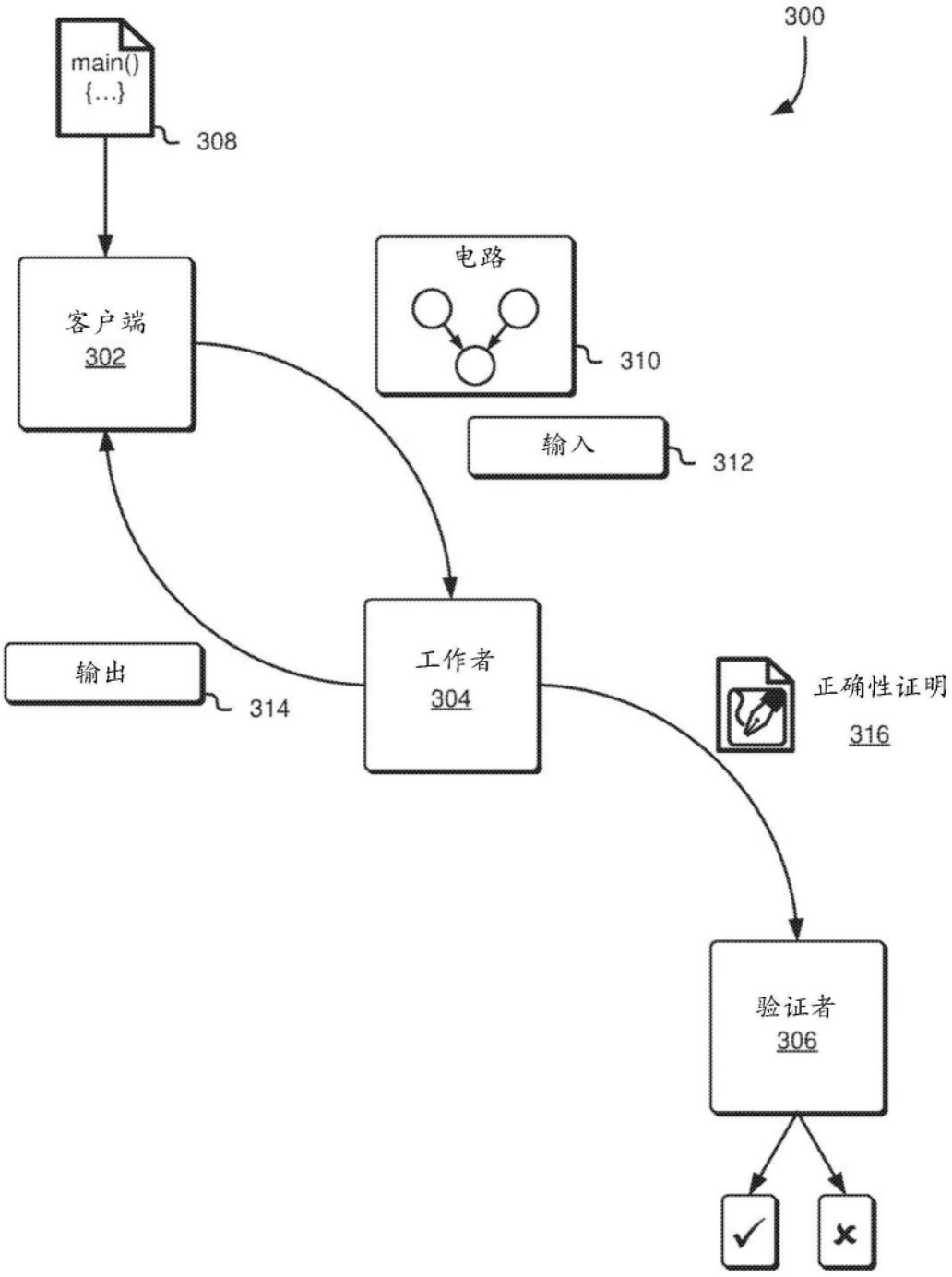

300
main()
{...}
308
客户端
302
电路
310
输入
312
工作者
304
输出
314
正确性证明
316
验证者
306

图3

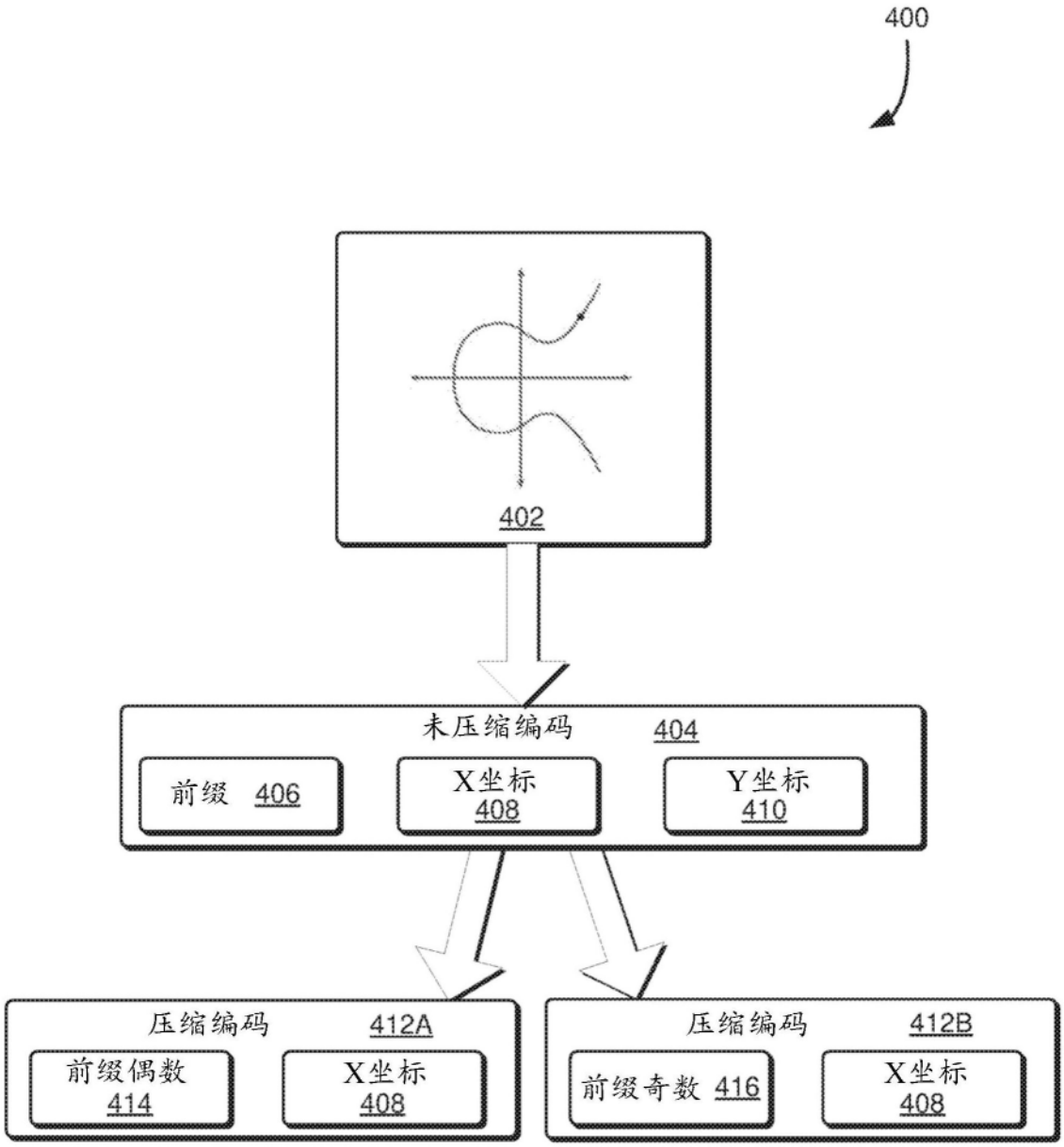

400
402
未压缩编码 404
前缀 406
X坐标 408
Y坐标 410
压缩编码 412A
前缀偶数 414
X坐标 408
压缩编码 412B
前缀奇数 416
X坐标 408

图4

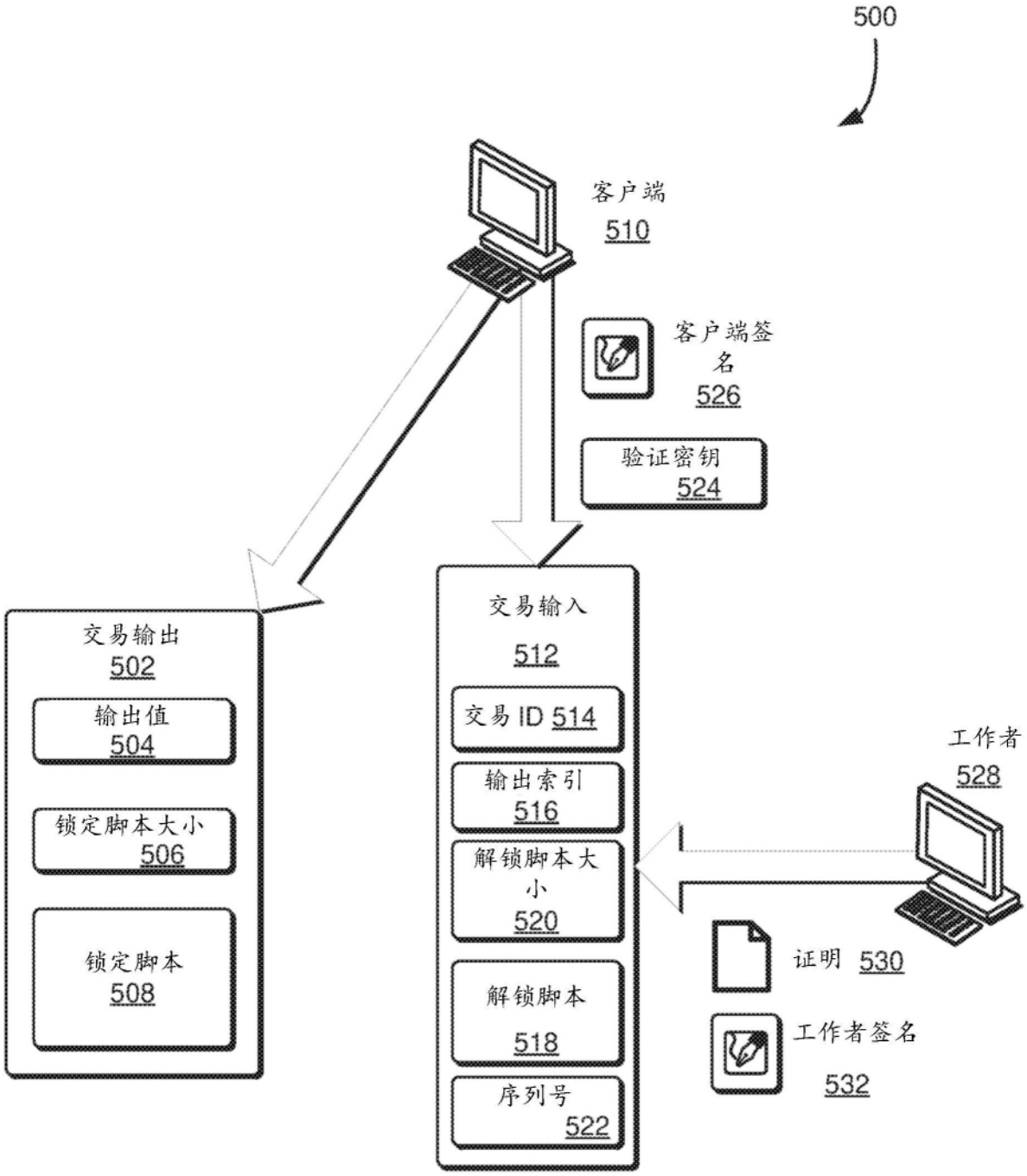

500
客户端
510
客户端签名
526
验证密钥
524
交易输出
502
输出值
504
锁定脚本大小
506
锁定脚本
508
交易输入
512
交易 ID 514
输出索引
516
解锁脚本大小
520
解锁脚本
518
序列号
522
工作者
528
证明 530
工作者签名
532

图5

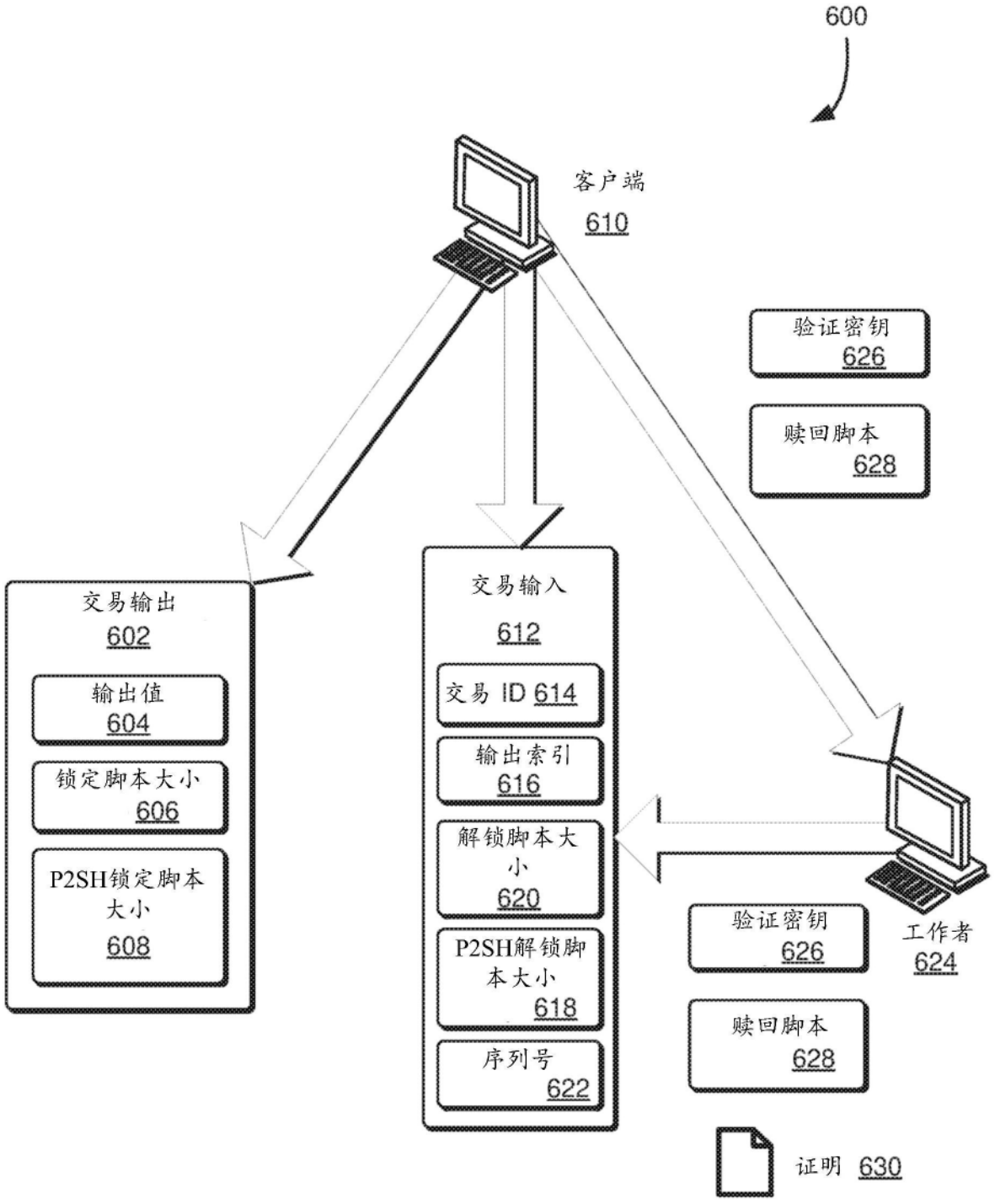

600
客户端
610
验证密钥
626
赎回脚本
628
交易输出
602
输出值
604
锁定脚本大小
606
P2SH锁定脚本大小
608
交易输入
612
交易 ID 614
输出索引
616
解锁脚本大小
620
P2SH解锁脚本大小
618
序列号
622
工作者
624
验证密钥
626
赎回脚本
628
证明 630

图6

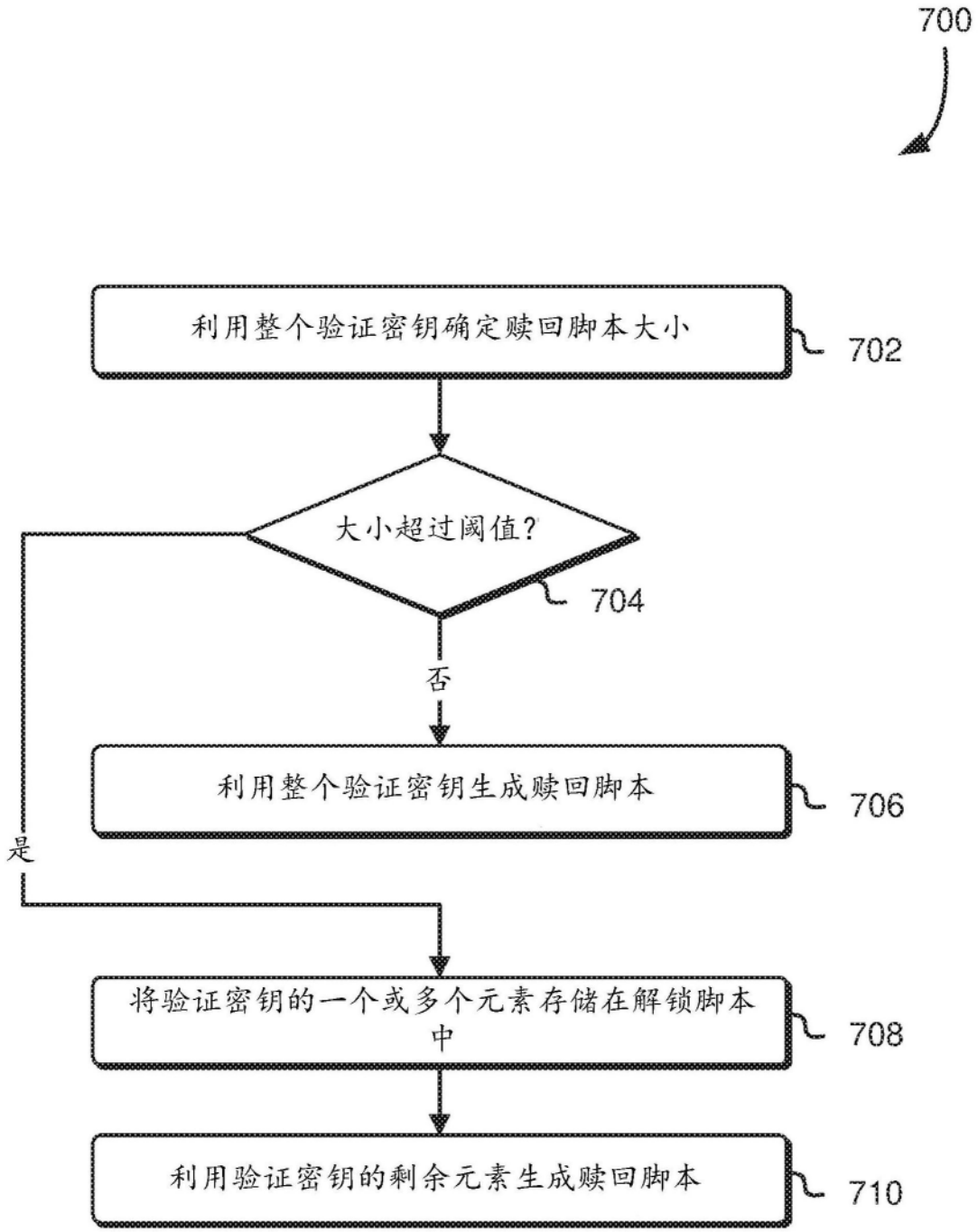

700
利用整个验证密钥确定赎回脚本大小
702
大小超过阈值?
704
否
利用整个验证密钥生成赎回脚本
706
是
将验证密钥的一个或多个元素存储在解锁脚本中
708
利用验证密钥的剩余元素生成赎回脚本
710

图7

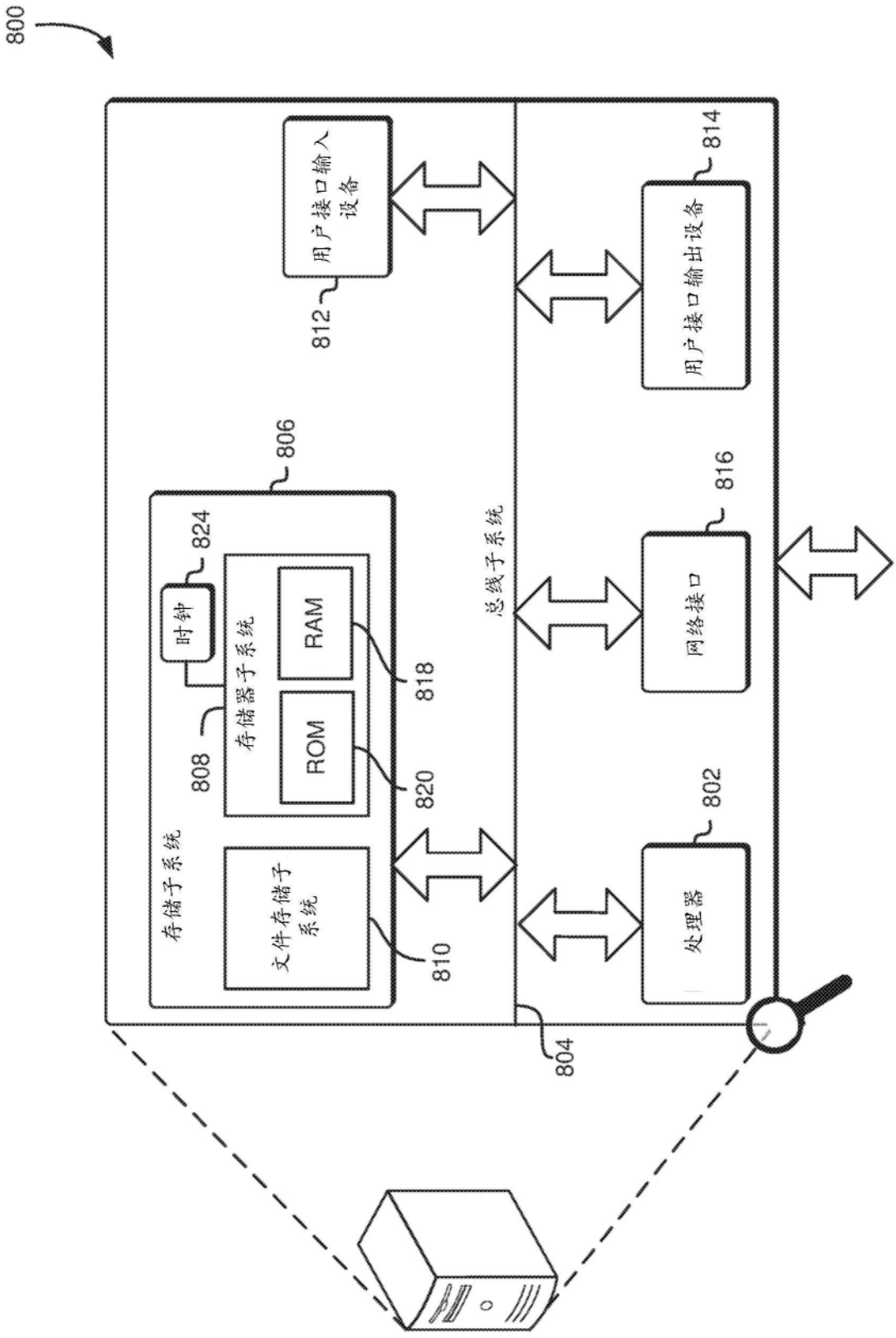

800
用户接口输入设备
812
用户接口输出设备
814
806
824
时钟
存储器子系统
RAM
818
808
ROM
820
总线子系统
网络接口
816
存储子系统
文件存储子系统
810
处理器
802
804

图8