



(12) 发明专利申请

(10) 申请公布号 CN 113924748 A

(43) 申请公布日 2022.01.11

(21) 申请号 202080038699.4

(22) 申请日 2020.04.21

(30) 优先权数据

1907397.2 2019.05.24 GB

(85) PCT国际申请进入国家阶段日

2021.11.24

(86) PCT国际申请的申请数据

PCT/IB2020/053762 2020.04.21

(87) PCT国际申请的公布数据

W02020/240289 EN 2020.12.03

(71) 申请人 区块链控股有限公司

地址 安提瓜和巴布达圣约翰教堂街44号菲
茨杰拉德故居

(72) 发明人 杰德·瓦哈伯 张伟

布洛克·多伊龙 克雷格·赖特

(74) 专利代理机构 北京中原华和知识产权代理

有限责任公司 11019

代理人 孙磊 徐民

(51) Int.Cl.

H04L 9/32 (2006.01)

G06Q 40/04 (2012.01)

(54) 发明名称

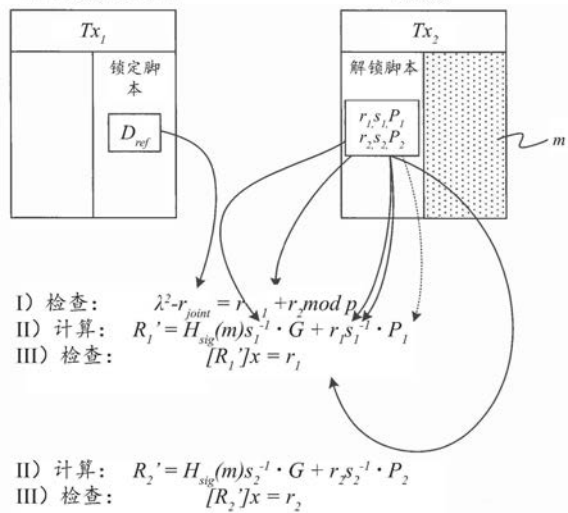
知识证明

(57) 摘要

在区块链网络的节点处:获取第一交易,所述第一交易包括可运行代码,所述可运行代码包括用于评估基于联合 r 值 r_{joint} 定义的挑战的参考数据;接收一笔或多笔第二交易,所述一笔或多笔第二交易包含包括一对ECDSA签名($i=1,2$)中的每个签名的 r 部分 r_i 和 s 部分 s_i 的信息,其中所述一对ECDSA签名中的每个签名基于对应于相应第一公钥 P_i 的相应第一私钥 V_i 对所述一笔或多笔第二交易中的一笔第二交易的一部分进行签名;运行所述代码。所述代码基于所述参考数据和所述 r 部分 r_i 来验证所述挑战是否得以应对。所述挑战包括以下条件: $r_1+r_2=(\lambda^2-r_{\text{joint}})\bmod p$,其中 $r_{\text{joint}}=[R_{\text{joint}}]_x$, $R_{\text{joint}}=R_1+R_2$, p 是素数模, $\lambda=\frac{y_2-y_1}{x_2-x_1}\bmod p$, $R_i=k_i\cdot G$, $x_i=[R_i]_x$, $y_i=[R_i]_y$, k_i 是临时密钥, G 是椭圆曲线生成点。

由爱丽丝设置的难题,
在节点104处评估

包括鲍勃和查理的信息
的交易



1. 一种执行知识证明的计算机实现方法,其基于椭圆曲线数字签名算法ECDSA;其中所述方法包括在区块链网络的验证节点处:

获取第一交易,所述第一交易包括可运行代码,所述代码包括用于评估基于联合 r 值 r_{joint} 定义的挑战的参考数据;

接收一笔或多笔第二交易,所述一笔或多笔第二交易包含至少包括一对第一ECDSA签名 $i=1,2$ 中的每个相应签名的相应 r 部分 r_i 和 s 部分 s_i 的信息,其中所述一对第一ECDSA签名中的每个签名基于对应于相应第一公钥 P_i 的相应第一私钥 V_i 对所述一笔或多笔第二交易中的一笔第二交易的一部分进行签名;

运行所述第一交易的所述代码,所述代码被配置为基于所述第一交易中的所述参考数据和在所述一笔或多笔第二交易中接收的所述 r 部分 r_i 来验证所述挑战是否得以应对,并且在所述挑战得以应对的条件下返回TRUE结果,所述挑战包括以下条件:

$$r_1+r_2=\lambda^2-r_{joint} \bmod p,$$

其中 r_1+r_2 表示标量加法, $r_{joint}=[R_{joint}]_x$,通过椭圆点加法得到的 $R_{joint}=R_1+R_2$, p 是素数模, $\lambda=\frac{y_2-y_1}{x_2-x_1} \bmod p$, $R_i=k_i \cdot G$, $x_i=[R_i]_x$, $y_i=[R_i]_y$, k_i 是临时密钥, G 是椭圆曲线生成点, $[]_x$ 表示 $[...]$ 的 x 坐标, $[]_y$ 表示 $[...]$ 的 y 坐标,“ $\cdot$ ”表示椭圆曲线标量乘法。

2. 根据权利要求1所述的方法,其中接收指示 λ^2 的数据作为在所述一笔或多笔第二交易中的至少一笔第二交易中接收的所述信息的一部分,并且基于所述接收的指示 λ^2 的数据来执行所述确定。

3. 根据权利要求1所述的方法,其中所述第一交易中的所述参考数据包括 p 和 λ^2-r 的值。

4. 根据权利要求1、2或3所述的方法,其中所述方法包括:获取每个所述第一ECDSA签名的所述相应第一公钥;应用所述ECDSA的验证函数以基于所述相应第一公钥和所述签名的部分来验证每个第一ECDSA签名,其中所述代码被配置为在每个所述第一ECDSA签名的所述验证的另一条件下返回所述TRUE结果。

5. 根据权利要求4所述的方法,其中所述获取所述第一公钥包括接收所述第一公钥作为所述一笔或多笔第二交易中的所述信息的一部分。

6. 根据前述任一项权利要求所述的方法,其中所述代码被配置为输出所述TRUE结果,而不管使用谁的公钥作为所述第一ECDSA签名中的一个、一些或全部签名的所述相应第一公钥。

7. 根据前述任一项权利要求所述的方法,其中所述ECDSA基于以下形式的椭圆曲线:

$$y^2=x^3+7.$$

8. 根据前述任一项权利要求所述的方法,其中所述ECDSA是secp256k1算法。

9. 根据前述任一项权利要求所述的方法,其中每个所述第一ECDSA签名是不同的相应第二方的签名,所述挑战至少部分由第一方定义。

10. 根据权利要求9所述的方法,其中每个所述第一ECDSA签名的所述 r 部分和 s 部分由所述相应第二方使用相应的临时密钥 k_i 生成。

11. 根据权利要求10所述的方法,其中每个相应的临时密钥由所述第一方提供给所述相应第二方,反之亦然。

12. 根据权利要求11所述的方法,其中:

$$\begin{aligned}
 P_i &= V_i \cdot G, \\
 k_i &\in [1, n-1], \\
 R_i &= k_i \cdot G, \\
 r_i &= [R_i]_x, \\
 s_i &= k_i^{-1} (H_{\text{sig}}(m) + r_i V_i) \bmod n,
 \end{aligned}$$

其中 k_i 是所述相应的临时密钥, m_i 是所述签名的第二交易的所述相应的签名部分, H_{sig} 是用于在生成所述相应的ECDSA签名时对 m 进行哈希处理的哈希函数, n 是所述生成点的素数阶。

13. 根据权利要求7至12中任一项所述的方法,所述方法包括在由所述代码返回的所述结果为TRUE的条件下为所述第一方触发服务。

14. 根据权利要求7至13中任一项所述的方法,其中所述接收所述一笔或多笔第二交易包括从所述第二方中的一个接收所述第二交易中的每笔交易。

15. 根据权利要求7至14中任一项所述的方法,其中在所述第二交易中接收的所述信息包括至少一个所述第二方的另一加密签名,所述至少一个所述第二方使用该方的另一私钥,所述另一私钥对应于另一公钥。

16. 根据权利要求15所述的方法,其中映射可用于使所述第一方和/或第三方能够基于所述另一公钥来查找所述至少一个第二方的身份。

17. 根据权利要求15或16所述的方法,其中所述代码被配置为使用所述另一公钥验证所述另一加密签名,并在所述另一加密签名被验证的另一条件下返回所述TRUE结果。

18. 根据权利要求7至17中任一项所述的方法,其中在所述第二交易中接收的所述信息更包括使用所述第一方的私钥的所述第一方的加密签名。

19. 根据权利要求7至18任一项所述的方法,其中:

在所述第二交易中接收的所述信息包括所述第一方中的一个、一些或全部中的每一个的附加的ECDSA签名,所述附加的ECDSA签名具有不同值的所述 r 部分(与所述第一ECDSA签名不同),但是使用与所述第一ECDSA签名相同的相应私钥;

所述代码被配置为使用所述相应第一公钥验证每个所述附加的ECDSA签名,并在所述附加的ECDSA签名被验证的另一条件下返回所述TRUE结果。

20. 根据前述任一项权利要求所述的方法,其中所述代码被配置为使得能够将较大的 r 部分值集合中的任何两个 r 部分值用作 r_1 和 r_2 。

21. 根据从属于权利要求7至19任一项的权利要求20所述的方法,其中:

所述集合包括至少三对 r 部分值;

将不同的相应 k 值对分配给至少三个第二方中的每一个,从而使他们能够生成集合中所述对中的相应一个;

r_1 是所述第二方中的一个的所述 r 部分值中的一个, r_2 是所述第二方中的另一个的所述 r 部分值中的一个;

因此,所述代码使得所述至少三个第二方中的任何两个能够应对所述挑战。

22. 根据前述任一项权利要求所述的方法,其中在所述同一第二方中接收所述信息。

23. 根据权利要求22所述的方法,其中所述交易中的每笔交易包括数据结构,所述数据结构包括一个或多个输入和一个或多个输出,其中每个输出包括锁定脚本,每个输入包括

解锁脚本和指向另一交易的输出的指针；

其中所述代码包含所述第一交易的所述锁定脚本，其中所述信息包含所述第二交易的输入中的所述解锁脚本，并且其中所述第二交易的所述输入中的所述指针指向所述第一交易的所述输出；

所述方法包括至少在所述代码返回所述TRUE结果的条件下核实所述交易有效，并且响应于所述核实，包括以下各项中的至少一项：

-将所述第二交易包括在交易池中，以便由所述验证节点挖掘到一个或多个区块中；
和/或

-将所述第二交易转发到所述区块链网络的至少一个其他节点。

24. 根据权利要求1至22中任一项所述的方法，其中所述交易根据基于账户的模型进行配置，所述代码包含包括在所述第一交易中的智能合约。

25. 根据权利要求24所述的方法，其中所述信息是通过多个第二交易接收的。

26. 一种计算机程序，所述计算机程序包含在计算机可读存储器上，并且被配置为当在网络的节点上运行时，执行根据前述任一项权利要求所述的方法。

27. 一种网络的节点，所述节点包括：

存储器，所述存储器包括一个或多个存储单元；

处置装置，所述处置装置包括一个或多个处理单元；

其中，所述存储器存储被设置为在所述处理装置上运行的代码，所述代码被配置为当在所述处理装置上运行时执行根据权利要求1至25中任一项所述的方法。

28. 一种用于记录在区块链中的第一交易，所述第一交易包含在一个或多个计算机可读介质上，其中所述第一交易包括可运行代码；

其中所述代码包括用于基于一笔或多笔第二交易来评估基于联合 r 值 r_{joint} 定义的挑战的参考数据，所述一笔或多笔第二交易包含至少包括一对第一ECDSA签名 $i=1,2$ 中的每个相应签名的相应 r 部分 r_i 和 s 部分 s_i 的信息，其中所述一对第一EDCSA签名中的每个签名基于对应于相应第一公钥 P_i 的相应第一私钥 V_i 对所述一笔或多笔第二交易中的一笔第二交易的一部分进行签名；

其中所述代码被配置为基于所述第一交易中的所述参考数据和在所述一笔或多笔第二交易中接收的所述 r 部分 r_i 来验证所述挑战是否得以应对，并且在所述挑战得以应对的条件下返回TRUE结果，所述挑战包括以下条件：

$$r_1 + r_2 = \lambda^2 \cdot r_{joint} \bmod p,$$

其中 $r_1 + r_2$ 表示标量加法， $r_{joint} = [R_{joint}]_x$ ， $R_{joint} = R_1 + R_2$ (通过椭圆点加法)， p 是素数， $\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod p$ ， $R_i = k_i \cdot G$ ， $x_i = [R_i]_x$ ， $y_i = [R_i]_y$ ， k_i 是临时密钥， G 是椭圆曲线生成点， $[]_x$ 表示 $[...]$ 的 x 坐标， $[]_y$ 表示 $[...]$ 的 y 坐标，“ $\cdot$ ”表示椭圆曲线标量乘法。

知识证明

技术领域

[0001] 本公开涉及一种知识证明,该知识证明经由一种用于记录在区块链中的交易集实现。

背景技术

[0002] 区块链是指一种分布式数据结构形式,其中在点对点(P2P)网络中的多个节点中的每个节点处维护区块链副本。区块链包括一系列数据区块,其中每个区块包括一笔或多笔交易。每笔交易都可以回指序列中的先前交易。交易可以通过提交到网络包括在新区块中。新区块的创建过程称为“挖掘”,该过程涉及多个挖掘节点中的每个挖掘节点争相执行“工作量证明”,即基于等待包括在区块中的未决交易池解决加密难题。

[0003] 区块链中的交易通常用于传递数字资产,即用作价值储存手段的数据。但是也可利用区块链实现区块链上的分层附加功能。例如,区块链协议可允许在交易输出中存储附加用户数据。现代区块链在单一交易中可储存的最大数据容量在不断增加,从而能够并入更复杂的数据。例如,这可用于在区块链中存储电子文档,甚至音频或视频数据。

[0004] 网络中的每个节点可以具有以下三个角色中的任何一个、两个或全部:转发、挖掘和存储。转发节点在整个网络节点中传播交易。挖掘节点将交易挖掘到区块中。存储节点各自对区块链中的已挖掘区块存储自己的副本。为了将交易记录在区块链中,一方将该交易发送到网络中的节点中的一个节点进行传播。接收该交易的挖掘节点可以争相将该交易挖掘到新区块中。每个节点被配置为遵守相同的节点协议,该协议将包括用于确认交易有效的一个或更多个条件。无效交易将不会传播或挖掘到区块中。假定交易已经核实有效,从而在区块链上被接受,则该交易(包括任何用户数据)将因此作为不可改变的公共记录,继续存储在P2P网络中的各个节点处。

[0005] 成功解决工作量证明难题以创建最新区块的矿工通常被奖励一笔称为“区块创始交易”的新交易,该交易会生成新的数字资产金额。工作量证明激励矿工不要欺骗系统,在他们的区块中包括双重花费交易,因为挖掘区块需要大量计算资源,而包括试图双重花费的区块很可能不会被其他节点接受。

[0006] 在“基于输出的”模型(有时称为基于UTXO的模型)中,给定交易的数据结构包括一个或更多个输入和一个或更多个输出。任何可花费输出包括指定数字资产金额的元素,有时称为UTXO(“未花费的交易输出”)。该输出还可以包括指定用于赎回该输出的条件的锁定脚本。每个输入包括指向先前交易中的此类输出的指针,并且还可以包括解锁脚本以用于解锁指向输出的锁定脚本。因此,考虑一对交易,将其称为第一交易和第二交易。第一交易包括指定数字资产金额的至少一个输出,并且包括定义解锁该输出的一个或更多个条件的锁定脚本。第二交易包括至少一个输入和解锁脚本,该至少一个输入包括指向第一交易的输出的指针;该解锁脚本用于解锁第一交易的输出。

[0007] 在此类模型中,当第二交易被发送到P2P网络以在区块链中传播和记录时,在每个节点处应用的有效性条件之一将是:解锁脚本满足在第一交易的锁定脚本中定义的一个或

更多个条件中的所有条件。另一条件将是：第一交易的输出尚未被另一早期有效交易赎回。根据这些条件中的任何一个条件发现第二交易无效的任何节点都不会传播该交易，也不会包括该交易以便挖掘到要记录在区块链中的区块中。

[0008] 另一种交易模型是基于账户的模型。在这种情况下，每笔交易均不通过参考过去交易序列中先前交易的UTXO来定义转移的金额，而是通过参考绝对账户余额进行定义。所有账户的当前状态由矿工单独存储到区块链中，并不断更新。基于账户的模型的交易也能够包括智能合约，其运行在每个节点并同时核实交易有效。

[0009] 在任一模型中，交易可以包括知识证明。“知识证明”是一个技术术语，指一方知道某些数据片段的任何测试，例如将其称为 d 。以基于输出的交易模型为例，一个交易 Tx_1 的输出中的锁定脚本可以包括哈希难题。如果第二交易 Tx_2 的输入指向 Tx_1 的该输出，则 Tx_2 的该输入中的解锁脚本将必须破解该哈希难题，以便成功地赎回 Tx_1 的输出。该哈希难题包括哈希值 h ，其是 d 的哈希值，即 $h = H_{\text{puz}}(d)$ 。该难题还包括一段脚本，当在节点处与 Tx_2 的解锁脚本一起运行时，该一段脚本将从 Tx_2 的解锁脚本中获取声称 d 的数据值 d' ，使用哈希函数 H_{puz} 对该数据值进行哈希处理，并将其与 Tx_1 的锁定脚本中包括的哈希值 h 进行比较。即，如果比较结果为“YES”（或者在本领域术语中为“TRUE”），则检查是否 $h = H_{\text{puz}}(d')$ ，并将仅解锁 Tx_1 的输出。因此，如果 d 包括在 Tx_2 的解锁脚本中以证明 d 的知识，则 Tx_2 的受益人只能解锁 Tx_1 的输出。

[0010] 单独使用传统哈希难题存在的问题在于，不择手段的矿工或其他节点可以在 Tx_2 的解锁脚本中观察到 d ，然后对 Tx_2 创建并挖掘（或发布）自己的 Tx_2^* 版本，在 Tx_2^* 的输出中向自己付款，而不是支付给如 Tx_2 中的预期接收者（例如，鲍勃）。避免这种情况的现有方法是在 Tx_1 的锁定脚本中附加地包括“支付到公钥哈希”（P2PKH）要求。除 d 的知识证明之外，这还需要将预期收款人的加密签名包括在 Tx_2 的解锁脚本中。

[0011] 哈希难题和P2PKH也可以在基于账户的模型中使用智能合约来实现，而不是使用基于输出的模型的锁定脚本和解锁脚本。

[0012] 如本领域技术人员所熟知的，加密签名可以基于私钥 V 来生成，并且可以基于私钥-公钥对中的对应的公钥 P 来验证。在给定通过将私钥 V 应用于消息 m 而生成的签名的情况下，另一方可以使用 P 来验证该签名是使用 V 生成的，而无需知道 V （因此，验证签名本身是另一种知识证明）。

[0013] 用于这种情况的一种算法是基于椭圆曲线加密算法（ECC）执行操作的椭圆曲线数字签名算法（ECDSA）。在这种情况下， P 和 V 之间的关系如下：

$$[0014] \quad P = V \cdot G$$

[0015] 其中 P 是二元向量 (P_x, P_y) ， V 是标量， G 是二元向量 (G_x, G_y) ，该二元向量表示二维椭圆曲线上的预定点（“生成点”）。运算“ $\cdot$ ”是标量椭圆曲线乘法—一种已知运算形式，从椭圆曲线上的点转换到另一点。

[0016] ECDSA签名是一种元组 (r, s) ，由本领域公知的两个元素，即分别为 r 部分（ r ）和 s 部分（ s ）组成。签名 (r, s) 是通过将私钥 V 应用于消息 m 而生成的。在将交易记录在区块链中的情况下， m 将是交易的一部分，除明文中的这部分之外，签名也将被标记到交易上，以使交易核实有效。例如，在基于输出的模型中，签名对 Tx_2 的一部分进行签名，并包括在 Tx_2 的锁定脚本中，以便解锁 Tx_1 的输出。签名部分通常包括交易的输出，所以在不核实签名无效并因此

不核实交易无效的情况下无法更改这些输出。

[0017] 无论使用何种交易模型,签名 (r,s) 的计算方式如下:

[0018] $r = [R]_x$, 其中 $R = k \cdot G$

[0019] $s = k^{-1} (H_{sig}(m) + rV) \bmod n$

[0020] 其中 $[R]_x$ 表示二元向量 $R = (R_x, R_y)$ 的 x 坐标。 k 称为临时密钥,通常从集合 $[1, n-1]$ 中随机选择,其中 n 是素数模, $[1, n-1]$ 是范围介于 1 到 $n-1$ (含) 的实数集合。 H_{sig} 是哈希函数,与哈希难题中使用的哈希函数 H_{puz} 相比,该哈希函数可以是相同或不同形式的哈希函数。

[0021] 在给定签名 (r,s) 、消息 m 和公钥 P 已知的情况下,任何不知道私钥 V 的一方都可以验证签名是使用消息 m 的私钥 V 生成的。这一点可以通过以下方式实现,即计算:

[0022] $R' = H_{sig}(m) s^{-1} \cdot G + r s^{-1} \cdot P$

[0023] 并且验证 $[R']_x = r$ 。签名只有在结果为 TRUE 时才有效,否则无效。这可以视为以下验证,即与公钥 P 相关联的一方确实是消息的签名者。

发明内容

[0024] 本发明涉及在本文中所公开的一种新知识证明,称为“ r 难题”。该知识证明基于参考值,该参考值对应于 ECDSA 签名的 r 部分作为挑战 (即难题) 的基础。该参考值包括在第一交易中 (例如,在 T_{x_1} 的锁定脚本中), 作为要求第二交易包括签名的挑战,该签名包括指定的 r 部分 (例如,在 T_{x_2} 的解锁脚本中), 以便赎回第一交易。通过提供对第二交易中的 r 难题的解,这可证明证明者必须已经知道相应的临时密钥 k ,但无需在第二交易中透露 k 。因此, k 可以用作临时私钥,并且 r 充当相应的临时公钥。

[0025] 特别地,本公开要求一种组合多个 r 部分的 r 难题。例如,每个 r 部分可以对应于不同当事方的临时密钥 k ,然后,该方法可以用于创建只能由共同行动的多个不同当事方赎回的第一交易。

[0026] 根据在本文中所公开的一个方面,提供了一种基于椭圆曲线数字签名算法 (ECDSA) 执行知识证明的计算机实现方法。所述方法包括在区块链网络的验证节点处: 获取第一交易,所述第一交易包括可运行代码,所述代码包括用于评估基于联合 r 值 r_{joint} 定义的挑战的参考数据; 接收一笔或多笔第二交易,所述一笔或多笔第二交易包含至少包括一对第一 ECDSA 签名 $i=1,2$ 中的每个相应签名的相应 r 部分 r_i 和 s 部分 s_i 的信息,其中每个所述第一 ECDSA 签名基于对应于相应第一公钥 P_i 的相应第一私钥 V_i 对所述一笔或多笔第二交易中的一笔第二交易的一部分进行签名。所述方法还包括运行所述第一交易的所述代码。所述代码被配置为基于所述第一交易中的所述参考数据和在所述一笔或多笔第二交易中接收的所述 r 部分 r_i 来验证所述挑战是否得以应对,并且在所述挑战得以应对的条件下返回 TRUE 结果。所述挑战包括以下条件: $r_1 + r_2 = \lambda^2 \cdot r_{joint} \bmod p$, 其中 $r_1 + r_2$ 表示标量加法, $r_{joint} = [R_{joint}]_x$, $R_{joint} = R_1 + R_2$ (通过椭圆点加法), p 是素数模, $\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod p$, $R_i = k_i \cdot G$, $x_i = [R_i]_x$, $y_i = [R_i]_y$, k_i 是临时密钥, G 是椭圆曲线生成点, $[]_x$ 表示 $[...]$ 的 x 坐标, $[]_y$ 表示 $[...]$ 的 y 坐标, “ $\cdot$ ” 表示椭圆曲线标量乘法。

[0027] 所公开的基于 r 难题的知识证明可以简单地用于替代传统的基于哈希难题的知识证明或 P2PKH。然而,可以在实施例利用的基于 r 难题的方法还有另一优点。换句话说,与

P2PKH中的挑战不同,构成第一交易(例如, T_{x_1})中挑战的基础的 r 部分与任何特定身份无关。同时,破解 r 难题证明 k 的知识而无需向验证者透露 k 的事实意味着 r 难题不易受到漏洞的影响,由此矿工或节点操作员可以观察该解并将其插入到他/她自己的第二交易版本中。

[0028] 因此,在实施例中,对于所述多个ECDSA签名中的一个、一些或全部签名,所述代码被配置为输出所述TRUE结果,而不管使用谁的公钥作为所述相应公钥。

[0029] 使用哈希难题作为知识证明并结合P2PKH以避免上述漏洞存在的问题在于,虽然P2PKH确保仅向首先知道 d 的一方付款,但这也意味着 T_{x_1} 的输出与特定的预定接收者或接收者集合相关(可以指定可包括替代接收者的替代条件,但它们仍然必须预先标识)。如本文所认识到的,可能期望允许可由任何未指定方赎回的交易,所述未指定方能够证明特定秘密值的知识,但仍以避免透露该值的方式。

[0030] 例如,假设爱丽丝想要建立第一交易,该交易可由她向其提供密钥的任何一组人员解锁,但她不想预先指定这些人员的身份。第一交易可由证明知道秘密的第二交易赎回。例如,该对交易可用于表示同意协议条款,和/或在该组同意后付款。爱丽丝可能想要立即订立协议,但仅在事实发生之后,才能决定一个或多个受信任人员的子集,以授予代表她签名的签字人授权或授权委托书。

附图说明

[0031] 为了帮助理解本公开的实施例并示出如何实施此类实施例,现将仅通过举例的方式参考附图进行说明,其中:

[0032] 图1是一种用于实现区块链的系统的示意性框图;

[0033] 图2示意性地示出了可记录在区块链中的交易的一些示例;

[0034] 图3是一种用于实现区块链的系统的另一个示意性框图;

[0035] 图4示出了一种用于根据基于输出的模型的节点协议来处理交易的节点软件的示意性框图;

[0036] 图5示意性地示出了示例性交易集;

[0037] 图6A至图6D示意性地示出了椭圆曲线数字签名算法(ECDSA)背后的一些原理;

[0038] 图7是本文中称为 r 难题(或同义地称为 r 挑战)的一种知识证明的一种可能的实现方式的示意图;

[0039] 图8是 r 难题的另一种可能的实现方式的示意图;

[0040] 图9是 r 难题的另一种可能的实现方式的示意图;

[0041] 图10是联合加法 r 难题的示意图;

[0042] 图11是一种用于根据基于账户的模型的节点协议来处理交易的节点软件的示意性框图;

[0043] 图12示意性地示出了ECDSA签名的示例性格式;

[0044] 图13是针对一种形式的 r 难题的锁定脚本和解锁脚本的示例性实现方式的分步脚本分析。

具体实施方式

[0045] 在一些加密方案中,验证者可能需要确信某人(称为证明者或被挑战者)在所谓的

知识证明中拥有一些信息。这可以简单地通过将该信息直接提供给验证者来实现。或者,可能需要证明者执行依赖于该信息的计算。优选地,所涉及的计算使得验证者他/她自己无需知道该信息即可设置挑战,也无需向验证者透露该信息即可验证证明者知道该信息。对于计算方法,必须对输入数据执行验证计算。由于加密哈希函数具有抗原像和抗冲突特性,因此证明秘密值知识的直接方法是使用加密哈希函数。这种哈希方法可以很容易地集成到许多区块链应用中,因为哈希函数可构成其私钥-公钥密码系统的基本部分。这种类型的知识证明广泛应用于区块链应用,通常称为哈希难题。

[0046] 在基于UTX0的区块链中,哈希难题(经过哈希处理的值的原像)解可以设置为花费条件,因此由矿工执行验证作为交易验证的一部分。然而,在该方法中,交易还必须要求使用特定私钥的签名,否则矿工在将交易包括在区块内之前接收哈希难题解。这将使恶意矿工有机会创建花费交易,其输出指向属于该矿工的地址。

[0047] 在本公开中,提供了一种知识证明,该知识证明可规避该问题,同时仍然允许由矿工(或转发节点)执行核实。为此,知识证明与对应于椭圆曲线数字签名算法(ECDSA)签名的临时密钥相关联。由于该算法中使用的密码原语是许多区块链原生的,因此可以很容易地集成到当前的基础设施中。

[0048] 示例性系统概述

[0049] 图1示出了一种用于实现区块链150的示例性系统100。系统100包括分组交换网络101,通常是诸如互联网的广域网。分组交换网络101包括多个节点104,该多个节点被设置成在分组交换网络101内形成点对点(P2P)覆盖网络106。每个节点104包括对等体的计算机设备,不同的节点104属于不同的对等体。每个节点104包括含一个或多个处理器的处理装置,例如一个或多个中央处理单元(CPU)、加速器处理器、特定应用程序处理器和/或现场可编程门阵列(FPGA)。每个节点还包括存储器,即采用非暂时性计算机可读介质形式的计算机可读存储器。存储器可包括一个或多个存储器单元,其采用一个或多个存储器介质,例如诸如硬盘等的磁介质、诸如固态硬盘(SSD)、闪存或电可擦可编程只读存储器(EEPROM)等的电子媒介和/或诸如光盘驱动器等的光学介质。

[0050] 区块链150包括一系列数据区块151,其中在P2P网络160中的多个节点中的每个节点处维护相应的区块链150副本。区块链中的每个区块151均包括一笔或多笔交易152,其中该上下文中的交易是指一种数据结构。数据结构的性质将取决于用作交易模型或计划的一部分的交易协议类型。给定的区块链通常全程使用一个特定的交易协议。在一种常见的交易协议中,每笔交易152的数据结构至少包括一个输入和至少一个输出。每个输出指定一个金额,该金额表示属于输出被加密锁定的用户103的数字资产值(需要该用户的签名进行解锁,从而进行赎回或花费)。每个输入指向先前交易152的输出,从而链接这些交易。

[0051] 节点104中的至少一些节点扮演转发节点104F的角色,这些节点转发并因此传播交易152。节点104中的至少一些节点扮演挖掘区块151的矿工104M的角色。节点104中的至少一些节点扮演存储节点104S(有时也称为“完整副本”节点)的角色,每个存储节点均在相应的存储器中存储相同区块链150的相应副本。每个矿工节点104M还维护等待挖掘到区块151中的交易152的池154。给定节点104可以是转发节点104F、矿工104M、存储节点104S或其中两个节点或所有节点的任意组合。

[0052] 在给定的当前交易152_j中,输入(或每个输入)包括指针,该指针引用交易序列中

先前交易152i的输出,指定该输出将在当前交易152j中被赎回或“花费”。通常,当前交易可以是池154或任何区块151中的任何交易。尽管为了确保当前交易有效,将需要存在先前交易152i并核实其有效,但是在创建当前交易152j甚至向网络106发送当前交易152j时,不必存在先前交易152i。因此,在本文中,“先前”是指由指针链接的逻辑序列中的前任,而不一定是时间序列中的创建时间或发送时间,因此,不一定排除无序创建或发送交易152i、152j的情况(参见下面关于孤立交易的讨论)。先前交易152i同样可以称为先行交易或前任交易。

[0053] 当前交易152j的输入还包括先前交易152i的输出被锁定到的用户103a的签名。反过来,当前交易152j的输出可以加密锁定到新用户103b。因此,当前交易152j可将先前交易152i的输入中定义的金额转移到当前交易152j的输出中定义的新用户103b。在某些情况下,交易152可具有多个输出,以在多个用户间分割输入金额(其中一个可以是原始用户103a,以便进行变更)。在某些情况下,交易还可以具有多个输入,以将一个或更多个先前交易的多个输出中的金额汇总在一起,并重新分配到当前交易的一个或更多个输出。

[0054] 上述可称为“基于输出的”交易协议,有时也称为未花费的交易输出(UTXO)的协议(其中输出称为UTXO)。用户的总余额不是用区块链中存储的任何一个数字定义的;相反,用户需要特殊“钱包”应用程序105,以整理该用户的所有UTXO值,这些UTXO值分散在区块链151的许多不同交易152中。

[0055] 作为基于账户的交易模型的一部分,另一种类型的交易协议可称为“基于账户的”协议。在基于账户的情况下,每笔交易均不通过参考过去交易序列中先前交易的UTXO来定义转移的金额,而是通过参考绝对账户余额进行定义。所有账户的当前状态由矿工单独存储到区块链中,并不断更新。在此类系统中,交易使用账户的运行交易记录(也称为“头寸”)进行排序。该值由发送者签名作为其加密签名的一部分,并作为交易引用计算的一部分进行哈希处理。此外,可选的数据字段也可以在交易中签名。例如,如果数据字段中包含先前交易的ID,则该数据字段可指向先前交易。

[0056] 无论采用何种类型的交易协议,当用户103希望执行新交易152j时,其希望将新交易从其计算机终端102发送至P2P网络106的节点104中的一个(现在通常是服务器或数据中心,但原则上可以是其他用户终端)。此节点104根据在节点104中的每个节点处应用的节点协议检查交易是否有效。节点协议的详细信息将与相关区块链150中使用的交易协议类型相对应,一起形成整个交易模型。节点协议通常要求节点104检查新交易152j中的加密签名是否与预期签名相匹配,这取决于交易152的有序序列中的先前交易152i。在基于输出的情况下,这可包括检查新交易152j的输入中包含的用户加密签名是否与新交易花费的先前交易152i的输出中定义的条件相匹配,其中该条件通常包括至少检查新交易152j的输入中的加密签名是否解锁新交易的输入所指向的先前交易152i的输出。在一些交易协议中,条件可至少部分地由输入和/或输出中包含的自定义脚本定义。或者,这可仅由节点协议单独确定,或可通过其组合确定。无论采用哪种方式,如果新交易152j有效,当前节点会将其转发到P2P网络106中的一个或多个其他节点104。这些节点104中的至少一些节点还作为转发节点104F,根据相同的节点协议应用相同的测试,从而将新交易152j转发到一个或更多个进一步的节点104,依此类推。通过这种方式,新交易在节点104的整个网络中进行传播。

[0057] 在基于输出的模型中,给定输出(例如,UTXO)是否花费的定义是,根据节点协议,

其是否通过另一个随后交易152j的输入有效赎回。交易有效的另一个条件是其试图花费或赎回的先前交易152i的输出尚未被另一笔有效交易花费/赎回。同样,如果无效,交易152j将不会在区块链中传播或记录。这可防止重复花费,即花费者对同一笔交易的输出花费超过一次。另一方面,基于账户的模型通过保持账户余额防止重复花费。因为同样存在定义的交易顺序,账户余额在任何时候均具有单一定义的状态。

[0058] 除核实之外,节点104M中的至少一些节点在称为挖矿的过程中争先创建交易区块,该过程以“工作量证明”为基础。在挖矿节点104M处,将新交易添加到区块中尚未出现的有效交易的池中。然后,矿工争相通过尝试解决加密难题来组装交易池154中交易152的新的有效区块151。通常情况下,这包括搜索“随机数”值,从而当随机数与交易池154并置且进行哈希处理时,哈希值的输出满足预定条件。例如,预定条件可以是哈希值的输出具有某个预定义的前导零数。哈希函数的特性是,相对于其输入,其具有不可预测的输出。因此,该搜索只能通过强力执行,从而在试图解决难题的每个节点104M处消耗大量的处理资源。

[0059] 解决难题的第一矿工节点104M在网络106上宣布难题解决,提供解决方案作为证明,然后网络中的其他节点104则可以轻松检查该解决方案(一旦给出哈希值的解决方案,就可以直接检查该解决方案是否使哈希值的输出满足条件)。基于已在每个此类节点处检查获胜者的已宣布解决方案,获胜者已为其解决该难题的交易池154之后由充当存储节点104S的节点104中的至少一些节点记录在区块链150中作为新区块151。区块指针155还分配给指向区块链中先前创建的区块151n-1的新区块151n。工作量证明有助于降低重复花费的风险,因为创建新区块151需要大量工作,并且由于包含重复花费的任何区块都可能被其他节点104拒绝,因此挖矿节点104M受到激励,不允许在其区块中包含双重花费。一旦创建,则不可修改区块151,因为其根据相同的协议在P2P网络106中的存储节点104S中的每个存储节点进行识别和维护。区块指针155还向区块151施加顺序。由于交易152记录在P2P网络106中每个存储节点104S处的有序区块中,因此提供了交易的不可变公共分类账。

[0060] 应当注意的是,在任何给定时间争相解决难题的不同矿工104M可能会根据任何给定时间的未挖掘交易池154的不同快照执行该操作,具体取决于他们何时开始搜索解决方案。解决相应难题的人员首先定义新区块151n中包含的交易152,并更新当前未挖掘交易池154。然后,矿工104M继续争相从新定义的未完成池154中创建区块,依此类推。此外,还存在解决可能出现的任何“分叉”的协议,其中两名矿工104M彼此在很短的时间内解决难题,从而传播区块链的冲突视图。简言之,分叉方向最长的成为最终区块链150。

[0061] 在大部分区块链中,获胜矿工104M会自动获得特殊类型的新交易作为奖励,该新交易创建新的数字资产值(与将数字资产金额从一个用户转移至另一个用户的正常交易截然相反)。因此,获胜节点被视为已“挖掘”一定数量的数字资产。这种特殊类型的交易有时称为“生成”交易,其自动形成新区块151n的一部分。该奖励可激励矿工104M争相参与工作量证明。通常情况下,常规(非生成)交易152还将在其输出中的一个输出中指定附加交易费用,以进一步奖励创建其中包含交易的区块151n的获胜矿工104M。

[0062] 由于挖掘中涉及的计算资源,通常至少矿工节点104M中的每个矿工节点采用服务器的形式,该服务器包括一个或多个物理服务器单元,甚至整个数据中心。每个转发节点104M和/或存储节点104S还可采取服务器或数据中心的形式。但是,原则上来说,任何给定节点104均可采用一个用户终端或联网在一起的一组用户终端的形式。

[0063] 每个节点104的存储器均存储被配置为在节点104的处理装置上运行的软件400, 以根据节点协议执行其相应的角色并处理交易152。应当理解的是, 在本文中归因于节点104的任何动作均可通过在相应计算机设备的处理装置上运行的软件400执行。节点软件400可以在应用层的一个或多个应用中实现, 或者在诸如操作系统层或协议层的较低层中实现, 或者在这些层的任何组合中实现。此外, 在本文中使用的“区块链”一词是指一般技术类型的通用术语, 不限于任何特定专有区块链、协议或服务。

[0064] 扮演消费用户角色的多方103中的每一方的计算机设备102也连接到网络101。他们充当交易中的付款人和收款人, 但不一定代表其他方参与挖掘或传播交易。他们不一定运行挖矿协议。出于说明目的, 示出了双方103及其相应的设备102: 第一方103a及其相应的计算机设备102a, 以及第二方103b及其相应的计算机设备102b。应当理解的是, 更多此类当事方103及其相应的计算机设备102可能存在并参与系统, 但为了方便起见, 未进行说明。每一方103均可以是个人或组织。仅出于说明目的, 在本文中, 第一方103a称为爱丽丝(Alice), 第二方103b称为鲍勃(Bob), 但应当理解的是, 这并不仅限于爱丽丝或鲍勃, 且本文对爱丽丝或鲍勃的任何引用均可分别用“第一方”和“第二方”替换。

[0065] 每一方103的计算机设备102包括相应的处理装置, 其包括一个或多个处理器, 例如一个或多个CPU、图形处理单元(GPU)、其他加速器处理器、特定应用程序处理器和/或FPGA。每一方103的计算机设备102还包括存储器, 即采用非暂时性计算机可读介质形式的计算机可读存储器。该存储器可包括一个或更多个存储器单元, 其采用一个或更多个存储器介质, 例如诸如硬盘等磁介质、诸如SSD、闪存或EEPROM等电子媒介和/或诸如光盘驱动器等的光学介质。每一方103的计算机设备102上的存储器存储软件, 其包括被设置为在处理装置上运行的至少一个客户端应用程序105的相应实例。应当理解的是, 在本文中归因于给定方103的任何行动均可通过在相应计算机设备102的处理装置上运行的软件执行。每一方103的计算机设备102包括至少一个用户终端, 例如台式或笔记本电脑、平板电脑、智能手机或诸如智能手表等的可穿戴设备。给定方103的计算机设备102还可包括一个或更多个其他网络资源, 诸如通过用户终端访问的云计算资源。

[0066] 客户端应用程序105最初可通过例如从服务器下载的适当计算机可读存储介质, 或通过诸如可移动SSD、闪存密钥、可移动EEPROM、可移动磁盘驱动器、软盘或磁带等的可移动存储设备、诸如CD或DVD ROM等的光盘或可移动光驱等提供至任何给定方103的计算机设备102。

[0067] 客户端应用程序105至少包括“钱包”功能。这有两个主要功能。其中一个功能是使相应的用户方103创建、签名和发送拟在节点104的整个网络中传播的交易152, 并因此包含在区块链150中。另一个功能是向相应方汇报其目前拥有的数字资产金额。在基于输出的系统中, 该第二功能包括整理分散在区块链150中属于相关方的各种交易152的输出中定义的金額。

[0068] 注意: 虽然各种客户端功能可以描述为集成到给定客户端应用105中, 但这不一定是限制性的, 相反, 在本文中所描述的任何客户端功能可以在由两个或多个不同应用组成的套件中实现, 例如经由API进行接口连接或一个应用作为另一个应用的插件。更通俗地说, 客户端功能可以在应用层或诸如操作系统的较低层或这些层的任意组合实现。下面将根据客户端应用105进行描述, 但应当理解的是, 这不是限制性的。

[0069] 每个计算机设备102上的客户端应用程序或软件105的实例可操作地耦合到P2P网络106的转发节点104F中的至少一个转发节点。这可以启用客户端105的钱包功能,以将交易152发送至网络106。客户端105还可联系一个、一些或所有存储节点104,以在区块链150中查询相应方103作为接收者的任何交易(或实际上在区块链150中检查其他方的交易,因为在实施例中,区块链150是在某种程度上通过其公开可见性提供交易信任的公共设施)。每个计算机设备102上的钱包功能被配置为根据交易协议制定和发送交易152。每个节点104运行软件400,其被配置为根据节点协议核实交易152有效的软件,并且在转发节点104F的情况下转发交易152,以在整个网络106中传播此类交易。交易协议和节点协议相互对应,给定交易协议和给定节点协议一起实现给定的交易模型。区块链150中的所有交易152均采用相同的交易协议(尽管交易协议可允许其内存在不同的交易子类型)。网络106中的所有节点104采用相同的节点协议(尽管其可根据针对该子类型定义的规则区分处理不同的交易子类型,并且不同的节点还可扮演不同的角色,从而实现协议的不同对应方面)。

[0070] 如上所述,区块链150包括一系列区块151,其中每个区块151包括通过如前所述的工作量证明过程创建的一个或更多个交易152的集合。每个区块151还包括区块指针155,其指向区块链中先前创建的区块151,以定义区块151的顺序。区块链150还包括有效交易池154,其等待通过工作量证明过程包含在新的区块中。每笔交易152包括指向先前交易的指针,以定义交易序列的顺序(注:交易152的序列可进行分支)。区块151的区块链一直追溯到创始区块(Gb) 153,该创始区块是区块链中的第一区块。区块链150中早期的一笔或多笔原始交易152指向创始区块153,而非先前交易。

[0071] 当给定方103(比方说爱丽丝)希望发送拟包含在区块链150中的新交易152j时,她将根据相关交易协议(使用其客户端应用程序105中的钱包功能)制定新交易。然后,她将交易152从客户端应用程序105发送至其连接的一个或更多个转发节点104F中的一个。例如,这可以是与爱丽丝的计算机102最近或最佳连接的转发节点104F。当任何给定节点104接收新交易152j时,其将根据节点协议及其相应的角色进行处理。这包括首先检查新接收的交易152j是否满足变为“有效”的特定条件,具体示例稍后将详细讨论。在一些交易协议中,有效条件可通过交易152中包含的脚本在每个交易的基础上进行配置。或者,条件可仅仅是节点协议的内置功能,或通过组合脚本和节点协议进行定义。

[0072] 如果新接收的交易152j通过有效性测试(即:“有效”的条件下),接收交易152j的任何存储节点104S将向在该节点104S处维护的区块链150的副本中的池154中添加新有效交易152。进一步地,接收交易152j的任何转发节点104F随后将有效交易152传播至P2P网络106中的一个或更多个其他节点104。由于每个转发节点104F应用相同的协议,因此假定交易152j有效,这意味着交易很快将在整个P2P网络106中传播。

[0073] 一旦进入在一个或更多个存储节点104处维护的区块链150的副本中的池154中,矿工节点104M将开始竞相解决包括新交易152的池154的最新版本方面的工作量证明难题(其他矿工104M可继续尝试基于池154的旧视角解决难题,但首先解决难题的矿工将定义下一个新区块151的结束位置和新池154的开始位置,最终将有人解决包括爱丽丝的交易152j的池154的一部分的难题)。一旦包括新交易152j的池154完成工作量证明,其将不可变地成为区块链150中区块151中的一个区块的一部分。每笔交易152包括指向早前交易的指针,因此交易的顺序也被不可变地记录下来。

[0074] 不同的节点104可以首先接收给定交易的不同实例,并且因此在一个实例被挖掘到区块150中之前具有关于哪个实例“有效”的冲突视图,此时所有节点104同意所挖掘的实例是唯一的有效实例。如果节点104接受一个实例为有效实例,然后发现第二实例已记录在区块链150中,则该节点104必须接受这一点,并将丢弃(即视为无效)其最初接受的未挖掘实例。

[0075] 基于UTX0的模型

[0076] 图2示出了示例性交易协议。这是基于UTX0的协议的示例。交易152(简称“Tx”)是区块链150的基本数据结构(每个区块151包括一笔或更多笔交易152)。下面将通过参考基于输出或基于“UTX0”的协议进行描述。但这并不限于所有可能的实施例。

[0077] 在基于UTX0的模型中,每笔交易(“Tx”)152包括数据结构,其包括一个或多个输入202和一个或多个输出203。每个输出203可包括未花费的交易输出(UTX0),其可用作另一新交易的输入202的来源(如果UTX0尚未赎回)。UTX0指定数字资产金额(价值储存手段)。它还可包含其来源交易的交易ID以及其他信息。交易数据结构还可包括标头201,其可包括输入字段202和输出字段203的大小指示符。标头201还可包括交易的ID。在实施例中,交易ID是交易数据(不含交易ID本身)的哈希值,且存储在提交至矿工104M的原始交易152的标头201中。

[0078] 比方说爱丽丝103a希望创建转移相关数字资产金额至鲍勃103b的交易152j。在图2中,爱丽丝的新交易152j标记为“Tx₁”。该新交易获取在序列中先前交易152i的输出203中锁定至爱丽丝的数字资产金额,并至少将此类金额中的一部分转移至鲍勃。在图2中,先前交易152i标记为“Tx₀”。Tx₀和Tx₁只是任意的标记,其不一定意味着Tx₀指区块链151中的第一交易且Tx₁指池154中的下一笔交易。Tx₁可指向仍具有锁定至爱丽丝的未花费输出203的任何先前(即先行)交易。

[0079] 当爱丽丝创建其新交易Tx₁时,或至少在她将该新交易发送至网络106时,先前交易Tx₀可能已经有效并包括在区块链150中。该交易此时可能已包括在区块151中的一个区块中,或者可能仍在池154中等待,在这种情况下,该交易将很快包括在新区块151中。或者,Tx₀和Tx₁可以创建并一起发送至网络102;或者,如果节点协议允许缓冲“孤立”交易,Tx₀甚至可以在Tx₁之后发送。本文交易序列上下文中使用的“先前”和“后续”一词是指由交易中指定的交易指针定义的序列中的交易顺序(哪个交易指向哪个其他交易等等)。它们同样可以替换为“前任”和“继任”、“先行”和“后代”或“父项”和“子项”等。这不一定指其创建、发送至网络106或到达任何给定节点104的顺序。然而,指向先前交易(先行交易或“父交易”)的后续交易(后代交易或“子交易”)不会有效除非父交易有效。在父交易之前到达节点104的子交易被视为孤立交易。根据节点协议和/或矿工行为,其可被丢弃或缓冲一段时间,以等待父交易。

[0080] 先前交易Tx₀的一个或多个输出203中的一个包括特定的UTX0,标记为UTX0₀。每个UTX0包括指定UTX0表示的数字资产金额的值以及锁定脚本,该锁定脚本定义后续交易的输入202中的解锁脚本必须满足的条件,以使后续交易有效,从而成功赎回UTX0。通常情况下,锁定脚本将金额锁定至特定方(该金额的交易受益人)。即,锁定脚本定义解锁条件,该解锁条件通常包括以下条件:后续交易的输入中的解锁脚本包括先前交易被锁定到的一方的加密签名。

[0081] 锁定脚本 (亦称scriptPubKey) 是节点协议识别的域特定语言中写入的一段代码。此类语言的特定示例称为“脚本 (Script)” (S大写)。锁定脚本指定花费交易输出203所需的信息,例如爱丽丝签名的要求。解锁脚本出现在交易的输出中。解锁脚本 (亦称scriptSig) 是提供满足锁定脚本标准所需信息的域特定语言中写入的一段代码。例如,其可包含鲍勃的签名。解锁脚本出现在交易的输入202中。

[0082] 因此在示出的示例中, Tx_0 的输出203中的 $UTXO_0$ 包括锁定脚本 $[Checksig P_A]$,该锁定脚本需要爱丽丝的签名 $Sig P_A$,以赎回 $UTXO_0$ (严格来说,是为了使试图赎回 $UTXO_0$ 的后续交易有效)。 $[Checksig P_A]$ 包含爱丽丝的公私密钥对中的公钥 P_A 。 Tx_1 的输入202包括指向 Tx_1 的指针 (例如,通过其交易ID ($TxID_0$),其在实施例中是整个交易 Tx_0 的哈希值)。 Tx_1 的输入202包括在 Tx_0 中标识 $UTXO_0$ 的索引,以在 Tx_0 的任何其他可能输出中对其进行标识。 Tx_1 的输入202进一步包括解锁脚本 $\langle Sig P_A \rangle$,该解锁脚本包括爱丽丝的加密签名,该签名由爱丽丝通过将其密钥对中的私钥应用于预定的部分数据 (有时在密码学中称为“消息”) 创建。爱丽丝需要签名以提供有效签名的数据 (或“消息”) 可通过锁定脚本、节点协议或其组合进行定义。

[0083] 当新交易 Tx_1 到达节点104时,该节点应用节点协议。这包括一起运行锁定脚本和解锁脚本,以检查解锁脚本是否满足锁定脚本中定义的条件 (其中该条件可包括一个或多个标准)。在实施例中,这涉及并置两个脚本:

[0084] $\langle Sig P_A \rangle \langle P_A \rangle || [Checksig P_A]$

[0085] 其中“||”表示并置,“ $\langle \dots \rangle$ ”表示将数据放在堆栈上,“ $[\dots]$ ”表示由解锁脚本组成的函数 (在该示例中指基于堆栈的语言)。同样,脚本可以使用公共堆栈一个接一个地运行,而不是并置脚本。无论采用哪种方式,当一起运行时,脚本使用爱丽丝的公钥 P_A (包括在 Tx_0 的输出的锁定脚本中),以认证 Tx_1 的输入中的锁定脚本是否包含爱丽丝签名预期部分的数据时的签名。预期的部分数据本身 (“消息”) 也需要包括在 Tx_0 中,以便执行此认证。在实施例中,签名的数据包括整个 Tx_0 (因此不需要包括一个单独的元素来明文指定签名的部分数据,因为其本身便已存在)。

[0086] 本领域技术人员将熟悉通过公私密码进行认证的细节。基本上而言,如果爱丽丝已通过使用其私钥加密签署消息,则给定爱丽丝的公钥和明文中的消息 (未加密消息),诸如节点104等其他实体可认证加密版本的消息必须已经由爱丽丝签名。签署通常包括对消息进行散列,签署哈希值和将此标记到消息的明文版本作为签名,从而使公钥的任何持有者能够认证签名。因此,应当注意的是,在实施例中,在本文中对签名特定数据片段或交易部分等的任何引用可以意味着对该数据片段或交易部分的哈希值进行签名。

[0087] 如果 Tx_1 中的解锁脚本满足 Tx_0 的锁定脚本中指定的一个或多个条件 (因此,在所示示例中,如果在 Tx_1 中提供了爱丽丝的签名并进行认证),则节点104认为 Tx_1 有效。如果是存储节点104S,这意味着其将添加至等待工作量证明的交易154池。如果是转发节点104F,则其将交易 Tx_1 转发到网络106中的一个或多个其他节点104,从而将在整个网络中传播。一旦 Tx_1 有效并包括在区块链150中,这将把 Tx_0 中的 $UTXO_0$ 定义为已花费。请注意, Tx_1 仅在花费未花费的交易输出203时才有效。如果试图花费另一交易152已经花费的输出,则即使满足所有其他条件, Tx_1 也将无效。因此,节点104还需要检查先前交易 Tx_0 中引用的 $UTXO$ 是否已经花费 (已经形成另一有效交易的有效输入)。这是为何区块链150对交易152施加定义的顺

序很重要的原因之一。在实践中,给定节点104可维护单独的数据库,标记已花费交易152的UTXO 203,但最终定义UTXO是否已花费取决于是否在区块链150中形成了另一有效交易的有效输入。

[0088] 如果给定交易152的所有输出203中指定的总金额大于其所有输入202所指向的总金额,则这是大多数交易模型中的另一失效依据。因此,此类交易将不会传播或挖掘到区块151中。

[0089] 请注意,在基于UTXO的交易模型中,给定UTXO需要作为一个整体使用。不能“留下”UTXO中定义为已花费的一部分金额,而同时又花费另一部分。但UTXO的金额可以在下一个交易的多个输出之间分割。例如, Tx_0 的UTXO₀中定义的金额可以在 Tx_1 中的多个UTXO之间分割。因此,如果爱丽丝不想将UTXO₀中定义的所有金额都给鲍勃,她可以使用剩余部分在 Tx_1 的第二输出中自己找零钱,或者支付给另一方。

[0090] 在实践中,爱丽丝通常还将需要包括获胜矿工的费用,因为现在仅靠区块创始交易的奖励币通常不足以激励挖掘。如果爱丽丝未包括矿工的费用, Tx_0 可能会被矿工节点104M拒绝,因此,尽管技术上有效,但仍然不会传播并包括在区块链150中(如果矿工104M不愿意,矿工协议不会强制他们接受交易152)。在一些协议中,挖掘费不需要其自身的单独输出203(即不需要单独的UTXO)。相反,给定交易152中输入202所指向的总金额与输出203所指定的总金额之间的任何差额都将自动提供给获胜矿工104。例如,假设指向UTXO₀的指针是 Tx_1 的唯一输入,而 Tx_1 只有一个输出UTXO₁。如果UTXO₀中指定的数字资产的金额大于UTXO₁中指定的金额,则该差额将自动提供给获胜矿工104M。替代地或附加地,这不一定排除可以在其自身交易152的其中一个UTXO 203中明确指定矿工费用。

[0091] 爱丽丝和鲍勃的数字资产由区块链150中任何位置的任何交易152中的锁定至他们的未花费UTXO组成。因此,通常情况下,给定方103的资产分散在整个区块链150的各种交易152的UTXO中。区块链150中的任何位置均未存储定义给定方103的总余额的一个数字。客户端应用程序105的钱包功能的作用是将锁定至相应方且在其他随后交易中尚未花费的各种UTXO值整理在一起。通过查询在任何存储节点104S(例如,与相应方的计算机设备102最近或最佳连接的存储节点104S)处存储的区块链150副本,可以实现这一点。

[0092] 请注意,脚本代码通常用示意图表示(即非精确语言)。例如,可写入[Checksig P_A]表示[Checksig P_A]=OP_DUP OP_HASH160<H(P_A)>OP_EQUALVERIFY OP_CHECKSIG。“OP...”是指脚本语言的特定操作码。OP_CHECKSIG(又称“Checksig”)是脚本操作码,其取两个输入(签名和公钥),并使用椭圆曲线数字签名算法(ECDSA)验证签名的有效性。在运行时,移除脚本中任何出现的签名(‘sig’),但在由‘sig’输入验证的交易中仍保留附加要求,诸如哈希难题。再如,OP_RETURN是脚本语言操作码,用于创建交易的不可花费输出,其可以将元数据储存在交易中,从而将元数据不可变地记录在区块链150中。例如,元数据可包括需存储在区块链中的文件。

[0093] 签名 P_A 是数字签名。在实施例,这基于使用椭圆曲线secp256k1的ECDSA。数字签名对特定的数据段进行签名。在实施例,对于给定交易,签名将对部分交易输入以及全部或部分交易输出进行签名。对输出的特定部分进行签名取决于SIGHASH标志。SIGHASH标志是包含在签名末尾的4字节代码,用于选择签名的输出(并因此在签名时固定)。

[0094] 锁定脚本有时称为“scriptPubKey”,指其包括相应交易被锁定到的当事方的公

钥。解锁脚本有时称为“scriptSig”，指其提供相应的签名。但是更通俗地说，在区块链150的所有应用中，UTXO赎回的条件并不一定包括对签名进行认证。更通俗地说，脚本语言可用于定义任何一个或更多个条件。因此，可以优选更为通用的术语“锁定脚本”和“解锁脚本”。

[0095] 可选的侧信道

[0096] 图3示出了用于实现区块链150的另一系统100。除了附加的通信功能外，该系统100与图1所示的内容基本相同。爱丽丝和鲍勃的每台计算机设备102a, 120b上的客户端应用程序分别包括附加通信功能。也就是说，这可使爱丽丝103a建立与鲍勃103b分离的侧信道301（在任何一方或第三方的鼓动下）。侧信道301能够独立于P2P网络实现数据交换。此等通信有时候被称为“链下”通信。比如，当交换爱丽丝与鲍勃之间的交易152时不想将该交易（仍未）发布到P2P网络106或挖掘到区块150，可以采用此等通信，直到其中一方选择将该交易广播到网络106。替代地或附加地，该侧信道301可以用于交换任何其他交易相关数据，例如密钥、协商的金额或条款、数据内容、等等。

[0097] 通过与P2P覆盖网络106相同的分组交换网络101可建立侧信道301。此外/或者，通过诸如移动蜂窝网络等不同网络、或者诸如本地无线网络等局域网、或者甚至爱丽丝和鲍勃的设备1021, 102b之间的直接有线或无线连接可以建立侧信道301。一般而言，本文所指的侧信道301可包括经由一种或多种联网技术或者通信介质的任何一个或多个链路，用于“链下”（即独立于P2P覆盖网络106）交换数据。在多个链路被使用的情况下，整个链下链路的捆绑或集合才可以被称为侧信道301。因此，需要注意的是，虽然爱丽丝和鲍勃通过侧信道301对特定的信息或数据片段或者诸如此类进行交换，但这并不一定意味着所有这些数据片段必须通过相同的链路或甚至同一类型网络进行发送。

[0098] 节点软件

[0099] 图4示出了在基于UTXO或基于输出的模型的示例中，在P2P网络106的每个节点104上运行的节点软件400的示例。节点软件400包括协议引擎401、脚本引擎402、堆栈403、应用级决策引擎404以及一个或更多个区块链相关功能模块的集合405。在任何给定节点104处，这些模块可以包括以下三个模块中的任何一个、两个或全部：挖掘模块405M、转发模块405F和存储模块405S（取决于该节点的一个或多个角色）。协议引擎401被配置为识别交易152的不同字段，并根据节点协议处理此类字段。当接收到具有指向另一先前交易 152_{m-1} (Tx_{m-1}) 的输出（例如，UTXO）的输入的交易 152_m (Tx_m) 时，协议引擎401标识 Tx_m 中的解锁脚本并将其传递给脚本引擎402。协议引擎401还基于 Tx_m 的输入中的指针来标识和检索 Tx_{m-1} 。如果 Tx_{m-1} 尚未在区块链150上，则可以从相应节点自身的未决交易池154中检索 Tx_{m-1} ；或者，如果 Tx_{m-1} 已在区块链150上，则可以从存储在相应节点或另一节点104处的区块链150中的区块151的副本中检索。无论采用哪种方式，脚本引擎401都会标识 Tx_{m-1} 的指向输出中的锁定脚本，并将其传递给脚本引擎402。

[0100] 因此，脚本引擎402具有 Tx_{m-1} 的锁定脚本和来自 Tx_m 的相应输入的解锁脚本。例如，图4中示出的 Tx_1 和 Tx_2 ，但同样可以应用于任何交易对，诸如 Tx_0 和 Tx_1 等。如前所述，脚本引擎402同时运行两个脚本，这将包括根据正在使用的基于堆栈的脚本语言（例如，脚本）将数据放置到堆栈403上并从该堆栈中检索数据。

[0101] 通过同时运行脚本，脚本引擎402确定解锁脚本是否满足锁定脚本中定义的一个或更多个标准，即解锁脚本是否对包括锁定脚本的输出进行解锁？脚本引擎402将该确定的

结果返回给协议引擎401。如果脚本引擎402确定解锁脚本确实满足在相应的锁定脚本中指定的一个或更多个标准,则返回结果“TRUE”。否则,返回结果“FALSE”。

[0102] 在基于输出的模型中,来自脚本引擎402的结果“TRUE”是交易有效性的条件之一。通常,还必须满足由协议引擎401评估的一个或更多个进一步协议级条件;例如, Tx_m 的输出中所指向的数字资产的总金额不超过输入指定的总金额,并且 Tx_{m-1} 的指向输出尚未被另一有效交易花费。协议引擎401评估来自脚本引擎402的结果以及一个或更多个协议级条件,并且只有当它们都为TRUE时,协议引擎401才核实交易 Tx_m 有效。协议引擎401将交易是否有效的指示输出到应用级决策引擎404。只有在 Tx_m 确实核实有效的条件下,决策引擎404才可以选择控制挖掘模块405M和转发模块405F中的一个或两个来执行它们涉及 Tx_m 的相应区块链相关函数。这可以包括:挖掘模块405M,该挖掘模块将 Tx_m 添加到节点的相应池154以挖掘到区块151中;和/或转发模块405F,该转发模块将 Tx_m 转发到P2P网络106中的另一节点104。然而,应当注意的是,在实施例中,虽然决策引擎404不会选择转发或挖掘无效交易,相反,这并不一定意味着,仅因为交易有效,该决策引擎就必须触发该交易的挖掘或转发。可选地,在实施例中,决策引擎404可以在触发这些函数中的一个或两个函数之前应用一个或更多个附加条件。例如,如果节点是挖掘节点104M,则决策引擎可以仅在交易有效且预留足够挖掘费用的条件下选择挖掘该交易。

[0103] 此外,还应当注意的是,在本文中,术语“TRUE”和“FALSE”不一定限于返回仅以单个二进制数(位)形式表示的结果,尽管这确实是一种可能的实现方式。更通俗地说,“TRUE”可以指指示成功或肯定结果的任何状态,而“FALSE”可以指指示不成功或不肯定结果的任何状态。例如,在基于账户的模型(图4中未示出)中,可以通过节点104对签名的隐式协议级核实和智能合约的附加肯定输出的组合来指示结果为“TRUE”(如果两个单独的结果均为TRUE,则认为总体结果为TRUE)。

[0104] 示例性交易集

[0105] 图5示出了根据在本文中所公开的实施例使用的交易集152。该交易集包括:第零交易 Tx_0 、第一交易 Tx_1 和第二交易 Tx_2 。应当注意的是,“第零”、“第一”和“第二”只是便利标签。它们并不一定意味着这些交易将立即相继放置在区块151或区块链150中,也不意味着第零交易是区块151或区块链150中的初始交易。这些标签不一定意味着任何关于他们的交易被发送到网络106的顺序的信息。他们仅指逻辑序列,其中下一交易的输入指向一个交易的输出。需记住的是,在一些系统中,可以在其子交易之后将父交易发送至网络106(在这种情况下,“孤立”子交易将在一个或多个节点104处缓冲一段时间,同时等待父交易到达)。

[0106] 第零交易 Tx_0 还可称为就本发明而言的源交易,因为其充当锁定至爱丽丝103a的数字资产金额的来源。第一交易 Tx_1 还可称为就本发明而言的挑战交易或难题交易,充当根据第二交易 Tx_2 有条件地从源交易 Tx_0 转移数字资产金额的中介,从而提供 r 难题的解。第二交易 Tx_2 还可称为证明交易或花费交易,因为该交易将提供第一交易中 Tx_1 设置的 r 难题的解,并将所得到的付款锁定至证明者(或者证明者代表的潜在受益人)。实施例可以通过示例的方式进行描述,其中证明者(第二方)恰好是鲍勃,但根据稍后的讨论应当理解的是, r 难题实际上允许任何第二方成为证明者,而不管其身份如何,只要他们提供破解 r 难题的有效签名。

[0107] 如图5所示,源交易 Tx_0 包括至少一个输出 203_0 (例如, Tx_0 的输出0),其指定数字资

产的金额,并且进一步包含将该输出锁定至爱丽丝103a的锁定脚本。这意味着源交易 Tx_0 的锁定脚本要求至少满足一种条件,即试图解锁输出(并且因此赎回数字资产的金额)的任何交易的输入必须在其解锁脚本中包含爱丽丝的加密签名(即使用爱丽丝的公钥)。在这种意义上, Tx_0 的输出中定义的金额可以说是由爱丽丝拥有。该输出可以称为UTX0。就本发明而言, Tx_0 的输入指向的先前交易的输出并不特别重要(只要足以涵盖 Tx_0 的总输出)。

[0108] 在这种情况下,解锁源交易 Tx_0 的输出的交易是第一交易 Tx_1 (挑战交易)。因此, Tx_1 具有至少一个输入 202_1 (例如, Tx_1 的输入0),该输入包括指向 Tx_0 的相关输出的指针(所示示例中 Tx_0 的输出0),并且进一步包括被配置为根据该输出的锁定脚本中定义的条件解锁 Tx_0 所指向输出的解锁脚本,其至少要求爱丽丝的签名。 Tx_0 的锁定脚本所需的爱丽丝的签名需要对 Tx_1 的一部分进行签名。在一些协议中, Tx_1 需要签名的部分可以是 Tx_1 的解锁脚本中定义的设置。例如,这可以通过SIGHASH标志设置,该标志附加至签名,为一个字节,因此就数据而言,解锁脚本显示为: $\langle \text{Sig } P_A \rangle \langle \text{sighashflag} \rangle \langle P_A \rangle$ 。或者,需要签名的部分可以仅仅是 Tx_1 的固定部分或默认部分。无论采用哪种方式,拟签名的部分都通常不包括解锁脚本本身,并且可能不包括 Tx_1 的部分或全部输入。然而, Tx_1 的签名部分将至少包括输出 203_1 ,该输出包含 r 难题(见下文,在该示例中, Tx_1 的输出0)。

[0109] 第一交易 Tx_1 具有至少一个输出 203_1 (例如, Tx_1 的输出0,输出也可称为UTX0)。第一交易 Tx_1 的输出未锁定至任何一方。就像 Tx_0 ,其具有至少一个输出(例如, Tx_1 的输出0),指定随后拟转移的数字资产金额,并且进一步包括锁定脚本,该锁定脚本定义解锁输出并且因此赎回该金额的所需内容。然而,该锁定脚本允许由提供 r 难题解的任何一方解锁其输出。

[0110] 第二交易(花费交易) Tx_2 具有至少一个输入 202_2 (例如, Tx_2 的输入0),该输入包括指向 Tx_1 的前述输出(Tx_1 的输出0,如示例所示)的指针,该输入还包括解锁脚本,该解锁脚本被配置为基于满足 Tx_1 的锁定脚本中定义的解锁条件的一个或多个要求解锁 Tx_1 的所述输出。根据在本文中所公开的实施例,解锁条件至少包括以下要求,即对应的解锁脚本包括 r 难题的解。 r 难题包括基于椭圆曲线加密算法(ECC)签名的 r 部分的 Tx_1 的锁定脚本中定义的挑战,该挑战可以由任何一方(在这种情况下恰巧是鲍勃)应对,包括他们在 Tx_2 的解锁脚本中的签名(或至少其 s 部分)。应当注意的是,与 Tx_0 的锁定脚本不同,任何一方的签名都可以用于解锁 Tx_1 中的锁定条件,只要它是应对 r 挑战(即 r 难题)的有效签名即可。稍后将更详细地讨论这方面的示例。在这里,仅选择鲍勃作为证明者或第二方的示例,但 r 难题实际上允许任何第二方作为证明者,例如查理(Charlie)、多拉(Dora)、以西结(Ezekiel)等。在一些实施例中, Tx_1 中的解锁条件还可以取决于一个或多个其他条件,例如还要求将爱丽丝的签名包括在 Tx_2 的解锁脚本中。

[0111] 第二交易 Tx_2 具有至少一个输出 202_2 (例如, Tx_2 的输出0),该输出指定转移给鲍勃的数字资产金额和将此锁定至鲍勃的锁定脚本(即,要求进一步后续交易在要花费的解锁脚本中包括鲍勃的签名)。在这种意义上,目标交易 Tx_2 的输出可以说是由鲍勃拥有。该输出同样可以称为UTX0。

[0112] 由证明者签名(例如,如果是鲍勃,则为 $\text{Sig } P_B$)签名的 Tx_2 的部分将至少包括该输出 203_2 ,即将付款锁定至证明者的输出(在该示例中, Tx_2 的输出0)。

[0113] 在实施例中, Tx_1 的输出 203_1 中的锁定脚本可能定义用于解锁输出的多个替代条件,例如多个替代 r 难题。在这种情况下,如果满足替代解锁条件中的任何一个替代解锁条

件,则 T_{x_2} 的输入202₂中的解锁脚本解锁 T_{x_1} 的输出。

[0114] 第零交易(即源交易) T_{x_0} 可以由爱丽丝、证明者(例如,鲍勃)或第三方生成,其通常需要先方的签名,爱丽丝从该先方获得 T_{x_0} 的输入中定义的金额。其可由爱丽丝、鲍勃、先方或其他第三方发送至网络106。

[0115] 第一交易(即挑战交易) T_{x_1} 还可以由爱丽丝、证明者(例如,鲍勃)或第三方生成。因为在实施例中需要爱丽丝的签名,所以其可以由爱丽丝生成。或者,其可以由鲍勃或第三方生成作为模板,然后发送至爱丽丝进行签名,例如通过侧信道301发送。然后,爱丽丝可自己将已签名交易发送至网络106,或者将其发送至鲍勃或第三方以供他们转发至网络106,或者仅发送她的签名供鲍勃或第三方组装到已签名 T_{x_1} 并转发至网络106。在发送 T_{x_1} 至网络106之前的任何链下交换均可通过侧信道301执行。

[0116] 第二交易(即证明交易或花费交易) T_{x_2} 可以由爱丽丝、证明者(例如,鲍勃)或第三方生成。由于第一版本需要证明者的签名和/或数据,其可由鲍勃生成。或者,其可以由爱丽丝或第三方生成作为模板,然后发送至鲍勃进行签名,例如通过侧信道301发送至鲍勃。然后,鲍勃可自己将已签名交易发送至网络106,或者将其发送至爱丽丝或第三方以供他们转发至网络106,或者仅发送他的签名和供爱丽丝或第三方组装到已签名 T_{x_2} 并转发至网络。

[0117] 应当理解的是,存在可以生成和组装交易的不同元素的多个位置,以及用于随后将其直接或间接地发送至P2P网络106的最终目的地的各种方法。所公开技术的实施方式的范围不限于这些方面中的任何一个。

[0118] 还应当理解的是,本文中诸如“由爱丽丝”、“由鲍勃”和“由第三方”等词组可分别用作“由爱丽丝103a的计算机设备102a”、“由鲍勃103b的计算机设备102b”和“由第三方的计算机设备”的简略语。此外,需再次注意,给定方的设备可包括由该方使用的一个或多个用户设备、或者诸如该方使用的云资源等服务器资源,或这些的任何组合。这不一定限制在单个用户设备上执行的行为。

[0119] 椭圆曲线数字签名算法(ECDSA)

[0120] 在许多不同区块链架构中,公钥密码学用作保护交易的基础。公钥密码学的用途包括公钥加密和数字签名方案。公钥密码学建立在以下原理之上,即某些函数易于计算,但在不具备特殊知识的情况下难以求逆。此类函数称为陷门函数,而求逆此类函数所需的特殊知识称为该函数的陷门。易于计算意味着在合理的时间范围内计算给定输入(或一组输入)的陷门函数在计算上是可行的,而难以求逆意味着在不具备陷门知识的情况下从结果推断该输入(或那些输入)在计算上是不可行的。

[0121] 在公钥密码学的上下文中,密钥对是指公钥(任何人都可以自由获得)和相应的私钥(假定为秘密,因为这仅对特定实体或组是已知的)。公钥可定义陷门函数,而对应的私钥是求逆该函数所需的陷门。

[0122] 在公钥加密上下文中,加密基于陷门函数(即,沿“正向方向”执行加密),而解密基于陷门函数求逆(即,沿“反向方向”执行解密),这仅在陷门已知时可行。

[0123] 在数字签名上下文中,使用公钥沿正向方向执行签名验证,而沿反向方向执行签名生成,并且只能使用私钥可行地执行签名生成。

[0124] 在区块链上下文中,基于公钥密码学的数字签名用作对交易进行加密签名和验证交易签名的基础。

[0125] ECC是一种公钥密码学,利用椭圆曲线的数学特性,与DSA(数字安全算法)等其他加密方案相比具有各种优点。

[0126] 椭圆曲线数字签名算法(ECDSA)是指一类数字签名方案,其使用ECC作为数字签名生成和验证基础。ECDSA的某些原理概述如下。

[0127] 在数学术语中,ECC利用素数阶有限域上椭圆曲线的代数结构。有限域是指一个有限的元素集合和一组相关联的乘法、加法、减法和除法运算,当应用于该集合中的元素时,这些运算满足一般算术规则(关联性、交换性等)。也就是说,在“一般”意义上不需要加法、乘法等运算,但它们的行为本质上是相同的。

[0128] 椭圆曲线运算:

[0129] 在ECC的上下文中,加法、减法和乘法运算分别是椭圆曲线点加法(在本文中表示为“+”)、椭圆曲线点减法(在本文中表示为“-”)和椭圆曲线标量乘法(在本文中表示为“•”)。加法和减法运算分别应用于椭圆曲线上的两个点,并返回椭圆曲线上的第三个点;然而,乘法运算应用于椭圆曲线上的标量和单个点,并返回椭圆曲线上的第二个点。相反,除法是按标量定义的。

[0130] 为了便于说明,图6A示出了 $\mathbb{R}^2$ 中的椭圆曲线 ϵ , $\mathbb{R}^2$ 是所有实值二维坐标的集合, $(x,y) \in \mathbb{R}^2$ 表示 $\mathbb{R}^2$ 的元素。椭圆曲线 ϵ 是满足以下等式的点的集合:

$$[0131] \quad \epsilon: y^2 = x^3 + ax + b$$

[0132] 加法: ϵ 的数学特性在于,在给定椭圆曲线 ϵ 上的任意两个点A,B的情况下,A和B相交的线将仅与 ϵ 和一个附加点重新相交,表示为C;A和B的椭圆曲线加法,即A+B,定义为C的“反射”:取与C相交的水平线,C的反射是与该线相交的椭圆曲线上的另一点。该定义适用于A=B的情况,修改后的C现在是 ϵ 的切线在A处与 ϵ 重新相交的点。通过将表示为 ∞ 的无穷远处的点定义为椭圆曲线上的点,并且任何垂直线在该点处与椭圆曲线相交(例如,标记为D和E的点垂直水平对齐,因此D+E= ∞),使得该定义适用于与两点相交的线垂直的情况。

[0133] 减法/加法逆运算:上述反射定义适用于任何点,并提供椭圆曲线点减法的定义:A-B是A与B的反射之和。B的反射更正式地称为B的“加法逆运算”,反过来表示为-B。使用该表示法,椭圆曲线减法可以用数学表示法定义为:

$$[0134] \quad A - B = A + (-B)。$$

[0135] 因此,在图6B中,C=- (A+B) 和 (A+B) = -C。另请注意,在这个定义下,D=-E,反映了代数结构的一般规则,即椭圆曲线上的任意点与其加法逆运算的椭圆点加法是无穷远处的点,即

$$[0136] \quad A + (-A) = \infty \quad \forall A \in \epsilon$$

[0137] 无穷远处的点 ∞ 更正式地称为“单位元素”(请注意与一般算术的并行性和偏离:在一般算术中,任何数a与其加法逆运算-a的和为0,其中0是一般算术的单位元素)。单位元素 ∞ 的另一特性反映了一般算术,即对于包含 ∞ 本身的 ϵ 上的任意点A的A+ ∞ =A(类似于任何实数a的语句a+0=0)

[0138] 乘法:根据椭圆曲线点加法的定义,椭圆曲线标量乘法的定义如下:椭圆曲线点A与整数v的乘法定义为:

$$[0139] \quad v \cdot A = \underbrace{A + \dots + A}_{v \text{ 次}}$$

[0140] 换句话说,作为 v ,椭圆曲线点 A 与它本身相加。

[0141] 注意:椭圆曲线标量乘法在本领域中也称为椭圆曲线点乘法。这两个术语在本公开中具有相同的含义。

[0142] 除法/乘法逆运算:关于标量定义除法运算:给定标量 v ,在标量 v^{-1} 处定义其“乘法逆运算”,使得:

$$[0143] \quad vv^{-1} = 1。$$

[0144] 图6A提供了上述运算的直观可视化,其中在包括所有实数 $\mathbb{R}$ 的无限域上定义 ε 。

[0145] 图6B更精确地示出了上述运算在ECC上下文中的实际应用方式,因为它示出了由等式定义的椭圆曲线 ε_n :

$$[0146] \quad \varepsilon_n: y^2 = x^3 + ax + b \pmod{p}$$

[0147] 其中 p 是素数(素数模), mod 表示模运算。满足上述等式的点的集合是有限的,在图6B中,除一个点之外的所有点表示为白色圆圈;其余点是单位元素 ∞ 。

[0148] 素数 p 构成椭圆曲线定义的一部分,可以自由选择。为了使椭圆曲线具有良好的加密特性, p 应该足够大。例如,在某些区块链模型中指定256位 p 。

[0149] 相比之下,下标“ n ”在本文中是指在上面定义的点加法下由椭圆曲线点形成的组的阶数(这可以简称为椭圆曲线 ε_n 的顺序),参见下文。

[0150] 换句话说, n 是组的阶数, p 是域的阶数。总共将有 n 个椭圆曲线点。椭圆曲线上的每个点由两个数字/坐标 (x, y) 表示,其中 x 和 y 都在范围 $-(p-1), \dots, 0, \dots, (p-1)$ 内。

[0151] 在形式上, ε_n 被视为在有限域 $\mathbb{F}_p$ 上定义的椭圆曲线,包括介于0到 $p-1$ (含)(表示为 $[0, p-1]$)的整数集 $[0, p-1] \cup \{\infty\}$ 加上单位元素 ∞ ,以及本质上上述算术运算集(经过一些修改-请参见下文)。

[0152] 可以看出,图6B中的 ε_n 表现出与图6A中的 ε 类似的水平对称,这是素数文件上椭圆曲线的一般特性,因此 ε_n 上点的加法逆运算的定义仍然适用。有些点没有水平对齐的对应点(例如, $(0, 0)$),此类点是它们自己的加法逆运算。

[0153] 与 ε_n 上两点 A 和 B 相交的“线” $l_{A,B}$ 成为有限的点集合,由较小的黑色圆圈表示,满足类似的几何要求,椭圆曲线标量乘法的定义仍然适用。与图6A类似,图6B示出了点 $A+B=-C$,该点是点 $C=-(A+B)$ 的加法逆运算,线 $l_{A,B}$ 在该点处与 ε_n 重新相交。

[0154] ε_n 上任意两个点的椭圆曲线加法 $A+B=-C$ 可以由以下等式进行代数定义:

$$[0155] \quad A = (x_A, y_A),$$

$$[0156] \quad B = (x_B, y_B),$$

$$[0157] \quad C = (x_C, y_C) = -(A+B),$$

$$[0158] \quad x_C = (\lambda^2 - x_A - x_B) \pmod{p},$$

$$[0159] \quad y_C = (\lambda(x_C - x_A) + y_A) \pmod{p},$$

$$[0160] \quad = (\lambda(x_C - x_B) + y_B) \pmod{p},$$

[0161] 其中

$$[0162] \quad \lambda = (y_A - y_B)(x_A - x_B)^{-1} \pmod{p} \text{ 如果 } A \neq B,$$

[0163] 和

[0164] $\lambda = (2y_A)^{-1}(3x_A^2 + a) \bmod p$ 如果 $A = B$ 。

[0165] 出于上述目的,将整数 v 的乘法逆运算 v^{-1} 的定义修改为:

[0166] $v^{-1}v \equiv 1 \pmod{p}$ 。

[0167] 换句话说,整数 v 的乘法逆运算是 $v \bmod p$ 的模逆。

[0168] $B = -A$ 的情况是特殊的,通过引入单位元素 ∞ 来解决,正如前面提到的,在这种情况下, $A+B=A+(-A)=\infty$ 。 $B=\infty$ 的情况也是特殊的,如上文所述破解,即 $A+\infty=A$ 。

[0169] 椭圆曲线标量乘法的定义采用该椭圆曲线加法的定义,否则保持不变。

[0170] 在其他上下文中,标量 v 的乘法逆运算 v^{-1} 的定义为:

[0171] $v^{-1}v \equiv 1 \pmod{n}$

[0172] 在上下文中,可以清楚地看到是否定义了关于 $\bmod n$ 或 $\bmod p$ 的乘法逆运算。

[0173] 在实践中,要确定一个数字应视为 $\bmod n$ 还是 $\bmod p$,可以应用以下检查:

[0174] 1. 该数字是否表示EC点的坐标?

[0175] a. 如果是,则应视为 $\bmod p$

[0176] 2. 该数字是否用于乘以EC点?

[0177] a. 如果是,则应视为 $\bmod n$

[0178] 应当注意的是,在某些情况下,两项检查都会给出肯定答案,在这种情况下,该数字必须是 $\bmod p$ 和 $\bmod n$ 。

[0179] 椭圆曲线加密算法(ECC)

[0180] 椭圆曲线算术提供了隐藏秘密值的独特功能,并构成了许多现代加密系统的基础。特别地,有限域上椭圆曲线点的标量乘法求逆是一个棘手的问题(在计算上是不可行的)。

[0181] 私钥 V 采用整数形式,对应的公钥 P 是椭圆曲线 ϵ_n 上从“生成点” G 推导出的点 P ,该点也是椭圆曲线 ϵ_n 上的一个点,如下所示:

$$[0182] \quad P = V \cdot G = \underbrace{G + \dots + G}_{V \text{ 次}}$$

[0183] 其中“ $\cdot$ ”表示由 a 、 b 和 n (椭圆曲线参数)定义的椭圆曲线 ϵ_n 上的椭圆曲线标量乘法。

[0184] 椭圆曲线 ϵ_n 的阶数 n 相对于生成点 G 定义为满足以下等式的标量:

[0185] $n \cdot G = \infty$,

[0186] 因此, n 既取决于椭圆曲线的选择,也取决于生成点 G 的选择。虽然 n 可以指椭圆曲线 ϵ_n 的阶数,但更准确地说 n 是生成点 G 相对于椭圆曲线的(由该生成点生成的子组的)阶数。在实际的ECC上下文中,应选择 G ,使得 n 较大且为素数。因此,在实践中, p 和 n 都是素数,但通常彼此不相等。

[0187] 对于足够大的 V ,实际执行 V 椭圆曲线加法推导 P 是困难的,即在计算上是不可行的。然而,如果 V 已知,则 P 可以通过利用椭圆曲线运算的代数特性来更有效地计算。可用于计算 P 的高效算法的一个示例是“双加”算法,重要的是,这只有在 V 已知的情况下才能实现。

[0188] 相反,如果 V 未知,则即使 G 和 P 已知,也没有在计算上可行的方法来推导出 V (即求逆标量乘法)(这就是所谓的“离散对数问题”)。攻击者可以尝试通过从 G 开始并重复执行椭

圆曲线点加法对P进行“蛮力”计算直到达到P;此时,他将知道V是他必须执行的椭圆曲线点加法的次数;但事实证明,这在计算上是不可行的。因此,V在上述意义上满足了陷门的要求。

[0189] 在ECC中,公钥P、生成器密钥G和椭圆曲线 ϵ_n 是公开的,并且假定是已知的,而私钥V是秘密的。

[0190] 椭圆曲线数字签名验证算法 (ECDSA)

[0191] 在区块链系统中,用户或其他实体通常持有用于证明其身份的私钥V,相应的公钥P将通过以下等式计算:

[0192] $P = V \cdot G$

[0193] 私钥V可用于使用ECDSA对数据m(“消息”)进行签名。

[0194] 例如,可以在以下内容中找到ECDSA的更多信息,其全部内容以引入方式并入本文:“RFC 6979-数字签名算法 (DSA) 和椭圆曲线数字签名算法 (ECDSA) 的确定性使用”, Tools.ietf.org, 2019年。

[0195] 图6C示出了签名生成函数(签名生成器600,其生成公钥-私钥对(V,P)的ECDSA签名(r,s))的示意性功能框图。EDSA签名是一对值,在本文中分别称为r部分(r)和s部分(s)。

[0196] 签名生成基于用于推导出公钥P的相同椭圆曲线 ϵ_n 和生成点G,因此椭圆曲线参数a,b和n以及生成点G被示为签名生成器600的输入。

[0197] 签名生成器600的临时密钥生成器602生成“临时”密钥 $k \in [1, n-1]$,即范围介于1到n-1(含)。

[0198] r部分生成器604根据k计算对应的临时公钥,如下所示:

[0199] $R = k \cdot G$

[0200] 然后取计算点的x坐标($[]_x$ 表示取椭圆曲线点的x坐标的过程):

[0201] $r = [R]_x$

[0202] 这是签名的r部分。

[0203] s部分生成器606使用 $k \bmod n$ 的模逆 k^{-1} (即, $k^{-1}k \equiv 1 \pmod{n}$)如上所述)和消息m的哈希值(表示为 $H(m)$,必要时截断)计算签名的s部分(s),如下所示:

[0204] $s = k^{-1}(H(m) + rV) \bmod n$

[0205] 在本示例中,消息m包括要包括在交易608中的数据(本示例中的一个或多个交易输出)。这可以称为对消息m进行签名的过程,并且该消息m可以称为交易的已签名部分。

[0206] 消息m和签名(r,s)反过来构成交易608的一部分。在本示例中,签名(r,s)作为解锁脚本的一部分包括在交易608的输入中。

[0207] 图6D示出了用于验证交易608的签名验证函数(签名验证器)620的示意性功能框图。由签名验证器620执行的计算基于相同的椭圆曲线 ϵ_n 和生成点G,如上所述,它们是公开的。

[0208] 虽然签名需要私钥V作为输入,即需要知道私钥才能生成有效的签名,但核实签名(r,s)只需要签名对(r,s)、消息m和公钥P。为了验证签名,签名验证器620对交易m的已签名部分进行哈希处理(应用与用于生成签名(r,s)的哈希函数相同的哈希函数H)。然后使用以下计算执行验证过程:

[0209] $R' = H(m) s^{-1} \cdot G + r s^{-1} \cdot P$

[0210] 当且仅当 $[R']_x = r$ 时签名才有效 (即签名验证成功), 否则签名无效 (即签名验证失败)。在本示例中, r 表示包括在交易 608 中的签名的 r 部分。

[0211] 例如, 在签名验证过程中使用的公钥 P 可以在先前交易的锁定脚本中指定。在这种情况下, 使用在先前交易的锁定脚本中指定的公钥以及 (后续) 交易 608 的已签名部分 m 和签名 (r, s) 来执行签名验证, 并且除非已经基于对应于在先前交易中指定的公钥 P 和后续交易 608 的已签名部分 m 的私钥 V 生成签名 (r, s) , 否则该签名验证将失败。因此, 只有持有私钥 V 的人才能声明先前交易的输出 (通常通过将他们自己的公钥包括在后续交易 608 的输出中), 并且在不核实签名 (r, s) 无效的情况下无法更改后续交易 608 的已签名部分 m 。

[0212] r 难题

[0213] 下面介绍一种基于 ECDSA 的新知识证明。作为说明, 挑战者是第一方爱丽丝, 她通过自己创建并向 P2P 区块链网络 106 发布 Tx_1 , 或者通过向第三方提供必要的详细信息以供他们组装到 Tx_1 中并发布, 在第一交易 Tx_1 中设置 r 难题。验证者 (实际运行证明的一方) 是网络的节点 104 的操作者, 例如矿工。通过向网络 106 发布 Tx_2 来提供 r 难题的解。证明者可以是任何第二方, 因为 r 难题本身与身份无关, 但是作为示例, 下面可以按照证明者恰好是鲍勃的情况来描述。证明者可以自己创建和发布 Tx_2 , 也可以向第三方提供将必要的详细信息以供他们组装到 Tx_2 中并发布。

[0214] 加密哈希函数提供了一种确定性地隐藏输入的方法, 其中输入中的微小变化会导致输出中的不可预测的变化。传统哈希函数包括 MD5、RIPEMD-160、SHA-1 和 SHA-256 [5], 其中每种哈希函数都具有抗冲突特性 (找到产生相同输出的两个输入的可能性极小) 和抗原像特性 (在给定哈希值 $h = H(d)$ 的情况下, 极难找到输入 d)。

[0215] 传统哈希难题可以如下设置。这种想法是建立第一交易 Tx_1 , 该交易允许第二交易 Tx_2 在其输入中包括某些特定数据的条件下由第二交易 Tx_2 赎回其输出。

[0216] 在区块链交易中, 第一方 (爱丽丝) 可以简单地使用锁定脚本中的哈希值 h 创建非标准交易 Tx_1 , 如下所示:

[0217] OP_HASH160<h>OP_EQUALVERIFY

[0218] 其中 $h = H_{puz}(d)$ 和 H_{puz} 是难题中使用的哈希函数 (在上面的示例中, 根据锁定脚本, 该哈希函数必须是 HASH160, 但在其他实现方式中可以使用另一形式的哈希函数)。要赎回包括该锁定脚本的 UTXO, 将需要后续交易的解锁脚本中的哈希难题解。因此, 具有地址 $Addr_Bob$ 的第二方的花费交易 Tx_2 将使用仅需包含 d 的解锁脚本来构建。

$TxID_2$	
输入	输出
[0219] 0. $TxID_1$ 解锁脚本: $\langle d \rangle$	0. 地址: $Addr_Bob$ 金额: $\{VALUE\}$

[0220] 其中 $TxID_i$ 是 Tx_i 的交易 ID。锁定脚本表示: 从 Tx_2 的输入中的解锁脚本中获取数据值 d , 对其进行哈希处理, 并检查是否等于 Tx_1 的输出中的锁定脚本中包括的哈希值 h 。因此, 通过在 Tx_2 的解锁脚本中提供 d 来解锁输出。

[0221] 在本示例中,在看到具有 Tx_2 的哈希难题解的用户交易之后,第一个接收到该交易的矿工可能会恶意拒绝该交易,并创建一个具有相同哈希难题解的新的延展版本 Tx_2^* ,但会将输出更改为他们自己的地址 $Addr_Miner$ 。然后,恶意矿工可以尝试将 Tx_2^* 挖掘到他/她自己的区块151,并且他们在 Tx_2 被挖掘之前成功,则恶意矿工而不是鲍勃将收到付款。

$TxID_2^*$	
输入	输出
[0222] 1. $TxID_1$ 解锁脚本: $\langle d \rangle$	1. 地址: $Addr_Miner$ 金额: $\{VALUE\}$

[0223] 数字签名通常用于区块链交易,以证明所有权并赎回未花费的交易输出(UTXO)。这使 Tx_1 等交易的输出能够锁定至特定方。最常见的示例是支付到公钥哈希(P2PKH)交易,其中交易的输出锁定至公钥的特定哈希值(也充当该方的地址)。公钥P的锁定脚本为:

[0224] $OP_DUP\ OP_HASH\ 160\langle h_p \rangle OP_EQUALVERIFY\ OP_CHECKSIG$

[0225] 其中 $h_p = H_{sig}(P)$ 和 H_{sig} 是签名中使用的哈希函数(在上面的示例中,根据锁定脚本,该哈希函数必须是HASH160,但在其他实现方式中可以使用另一形式的哈希函数)。为了能够将此UTXO用作另一交易的输入,必须使用P提供具有有效ECDSA签名的解锁脚本:

[0226] $\langle sig \rangle \langle P \rangle$

[0227] 整个字符串(解锁脚本+锁定脚本)由矿工评估,检查是否提供了正确的公钥,并且签名是否有效且对应于P。锁定脚本基本上表示:从 Tx_2 的输入中的解锁脚本中获取公钥P,对其进行哈希处理,并检查是否等于 Tx_1 的输出中的锁定脚本中包括的哈希值 h_p ;并且还使用基于的ECDSA验证函数的 Tx_2 的解锁脚本中的公钥P来验证签名sig,前提是具备 Tx_2 的已签名部分的知识。ECDSA验证函数由 $OP_CHECKSIG$ 操作码调用。

[0228] 因此,只能通过在 Tx_2 的解锁脚本中提供基于对应于P的私钥V签名的有效签名sig来解锁输出。

[0229] 将这一点与哈希难题结合在一起,可以通过要求预期接收者的数字签名以及哈希难题解来纠正上述漏洞。锁定脚本将构建为:

[0230] $OP_HASH160\langle h \rangle OP_EQUALVERIFY\ OP_DUP\ OP_HASH160\langle h_p \rangle OP_EQUALVERIFY\ OP_CHECKSIG$

[0231] 对应的解锁脚本必须为:

[0232] $\langle sig \rangle \langle P \rangle \langle d \rangle$ 。

[0233] 然而,这会限制谁能够将其赎回给公钥P的所有者。本文认识到,这在某些应用中可能不是所希望的,例如在爱丽丝希望仅在设置难题之后才保留指定签字人授权的能力的情况下。

[0234] 本文认识到,可以通过利用ECDSA签名中的r部分来模拟哈希难题功能,该r部分可以是临时随机值。ECDSA签名由r和s两个主要部分组成。如上所述, $r = [k \cdot G]_x$ 。代替传统哈希难题 $h = H(d)$,逆椭圆曲线加法的难解性可形成类似难题,在本文中称为r难题。要破解该难题,需要获取解值k,其中k是对应于r的临时密钥。

[0235] 对于传统哈希难题,在破解难题时,存在在区块链透露d的风险。然而,对于r-难题,从未透露k。相反,会透露r,可根据r和签名证明具备k的知识。

[0236] 为了模拟哈希难题功能,r难题的创建者可以首先对一些其他原像数据进行哈希以获取值k,因为k必须是固定大小,而哈希难题的原像数据可以具有任何长度(并且哈希函数的一个特性是它会输出固定长度的值,而不管输入数据的长度如何)。例如,如果使用256位长的私钥/临时密钥,则应该对r难题的原像数据进行哈希处理以获取k。然而,也可以直接选择k的某个合适长度的值并将其直接用作秘密值(即,不需要从某个其他先前原像中推导出该秘密值)。

[0237] 该方法可用于任何使用ECDSA签名进行花费的区块链系统。作为说明,下面将介绍基于UTXO的模型中的示例性实现方式。在脚本语言中,OP_CHECKSIG操作码需要堆栈上的签名和公钥(公钥位于堆栈的顶部,签名位于堆栈的正下方)。对于r难题,脚本被配置为检查所提供的签名中的r值与用于r难题挑战的值是否相同。换句话说,该脚本不仅将检查签名在公钥上是否有效(通过OP_CHECKSIG),还将确保签名是使用r难题的r值创建的,该值将事先发布在区块链上。

[0238] 现在参考图7至图10讨论r难题的一些示例性实现方式。在每种情况下,证明者(例如,鲍勃)都通过对Tx₂的一部分进行签名来创建签名(r,s)。这种形式的签名有时也可称为“sig”。在加密签名的上下文中,已签名部分也称为“消息”(m)。已签名部分(消息)m至少包括Tx₂的输出203₂,其将最终付款锁定至鲍勃。如果有多个输出,则m可以包括部分或全部输出。如果使用锁定时间,m也可以包括其他部分。然而,通常不包括解锁脚本本身(当然必须至少不包括签名本身)。要签名为消息m的Tx₂的部分可以由Sighash设置,也可以是协议的默认功能或固定功能。

[0239] 图7示出了最简单的实现方式。Tx₁中的锁定脚本包括这里标记为r'的引用实例或r部分。在该方法中,Tx₂中的解锁脚本仅需至少包含鲍勃签名的s部分(s)。它还可以包括相应于鲍勃用于对m进行签名的私钥V的公钥P。Tx₁的锁定脚本被配置为当由节点104处的脚本引擎402运行时,从Tx₂的解锁脚本中获取s和P并执行以下操作:

[0240] I) $R' = H_{sig}(m) s^{-1} \cdot G + r' s^{-1} \cdot P$, 和

[0241] II) 检查 $[R']_x = r'$,

[0242] 其中r'是从Tx₁的锁定脚本中获取的,s和m是从Tx₂的解锁脚本中获取的。鲍勃的公钥P也可以从解锁脚本Tx₂中获取,或者可以通过其他方式获知。 H_{sig} 是用于在生成第一ECDSA签名时对m进行哈希处理的哈希函数。它可以是任何形式的哈希函数。无论其采取何种形式,都可以假定该哈希函数的形式(类型)是预定的并且在两端都是已知的。G是一个固定的公知向量值。

[0243] 锁定脚本被配置为在检查为TRUE的条件下返回“TRUE”结果,否则返回“FALSE”结果。在UTXO的情况下,运行锁定脚本和解锁脚本的TRUE(即成功)结果是交易有效的必要条件。因此,Tx₂的有效性可用作r难题结果的代理。或者换句话说,Tx₂的有效性取决于提供r难题的解。即,如果鲍勃没有通过r难题,则他的交易Tx₂将不会在网络106上传播,也不会记录在区块链150中(并且不会赎回在Tx₁的输出中定义的任何付款)。

[0244] 虽然图7的示例在数学意义上可能是最简单的,但这并不一定意味着与任何给定的节点协议或脚本语言集成都是最简单的。如果花费者在解锁脚本中仅提供<s>和<P>,而

不是 $\langle r, s \rangle$ 和 $\langle P \rangle$, 则脚本必须考虑到这一点。操作I) - II) 不是标准Checksig类型操作码的操作。OP_CHECKSIG操作码期望签名采用DER格式, 因此如果解锁脚本中只提供了 $\langle s \rangle$ 值, 那么锁定脚本中将需要一些额外的操作码(OP_CAT用于级联等), 以便生成DER格式的有效签名。图8简单地示出了替代示例, 虽然在数学上涉及额外的步骤, 但实际上更简单地集成了Script等脚本语言, 这些脚本语言已经具有专用操作码, 用于调用基于从Tx₂的输入中获取的r和s的ECDSA签名验证。

[0245] 还应注意: 在所有可能的实施例中不必将P包括在Tx₂中。事实上, 根据对消息m和(r, s) (或在这种情况下, (r', s)) 的了解, 可以计算出公钥的两个可能的值P和-P (但不知道具体情况)。然后可以使用两个验证来标识哪一个是正确的, 或者替代地, 可以在Tx₂中包括一个位标志, 以指示要使用两个可能的解中的哪一个。后一种方法目前在一些基于账户的协议中使用。然而, 它倾向于不在当前基于UTXO的协议中使用, 在这些协议中, 脚本语言 (例如, Script) 没有用于根据(r, s) 和m计算P和-P的运算操作码。然而, 不应该排除可以引入一个操作码或者可以简单地将操作显式编码到锁定脚本中的可能性。另一种可能是, 爱丽丝已经知道或已经访问P或通过侧信道301接收操作码。然而, 这将需要单独查找才能将P映射到Tx₂。

[0246] 图8中示出了另一示例性实现方式。在这里, r难题要求Tx₂的解锁脚本显式包括r部分的提交实例r。Tx₁的锁定脚本包括对r部分的测试, 该测试包括要与提交实例r进行比较的r部分的引用实例r'。在此方法中, Tx₂中的解锁脚本必须至少包含鲍勃签名的r部分(r) 和s部分(s)。它还可以包括相应于鲍勃用于对m进行签名的私钥V的公钥P。Tx₁的锁定脚本被配置为当由节点104处的脚本引擎402运行时, 从Tx₂的解锁脚本中获取r、s和P并执行以下操作:

[0247] I) 检查 $r' = r$,

[0248] II) 计算 $R' = H_{\text{sig}}(m) s^{-1} \cdot G + r s^{-1} \cdot P$,

[0249] III) 检查 $[R']_x = r$,

[0250] 其中r' 是从Tx₁的锁定脚本中获取的, s、r和m是从Tx₂的解锁脚本中获取的。鲍勃的公钥P也可以从解锁脚本Tx₂获取, 或者可以通过其他方式获知, 例如如前所述从(r, s) 和m或(r, s) 和m推导出。

[0251] 锁定脚本被配置为在步骤I) 和III) 中的检查均为TRUE的条件下返回结果“TRUE”, 否则返回结果“FALSE”。同样, 在基于UTXO的情况下, 这使得能够根据r难题知识证明的结果来确定交易的有效性。应当注意的是, 数字I-III不一定表示顺序。检查I) 可以在II) - III) 之前或之后执行, III) 确实必须在II) 之后执行。

[0252] 在图8的方法中, 步骤II) 和III) 单独是由ECDSA验证函数执行的常规操作。因此, 在大多数协议中, 它们可以由专用操作码调用, 例如脚本中现有的Checksig操作码(OP_CHECKSIG)。步骤I) 可以使用通用操作码单独编码到锁定脚本中 (稍后给出示例)。也不排除步骤II) 和III) 原则上可以使用通用操作码而不是使用Checksig等专用操作码来显式编码。

[0253] 在一个示例性交易协议中, 交易ECDSA签名使用ASN.1 (抽象语法表示法一) DER (可区别编码规则) 编码格式, 如图12所示。第一字节字段包含标志0x30, 表示ASN.1序列号。第二字节字段包含以十六进制表示的序列长度。第三字节字段包含标志0x02, 表示ASN.1整

数。之后, ECDSA签名的r值包含在接下来的32或33字节中。字段应为32字节, 但是, 如果r的第一字节大于0x7f (第一位为1), 则在r值前面添加0的加法字节, 使其长度为33字节。这是DER格式编码的结果, 该编码将整数的第一位解释为符号。额外的零字节被添加到值的开头, 这样它就不会被解释为负值。ECDSA签名的s值也是如此。最后, 向DER编码添加一个字节字段, 即哈希类型(ht), 该字段对应于交易中的签名类型(SIGHASH_ALL、SIGHASH_NONE等)。

[0254] 考虑爱丽丝(A) 想要创建一个r难题交易的情况, 在该交易中, 任何获得该难题解的人都可以花费。为此, 她将创建如下所示的新交易 Tx_1 。输入部分包括正在花费的先前交易 Tx_0 的解锁脚本。为简单起见, 假设它是使用爱丽丝签名和公钥的标准P2PKH。输出部分包括锁定脚本(脚本公钥), 换句话说就是r难题挑战。如图12所示, 签名可能在某些协议中使用DER编码格式, 因此脚本必须从已编码签名中提取r的值, 然后检查它是否等于 $\langle r \rangle$ 。之后, 脚本必须检查公钥上的签名是否有效。图5示出了该脚本工作原理的更详细描述。粗体的操作码本质上只是从签名中提取r的一种方式。

$TxID1$	
输入	输出
[0255] 任何花费 输入	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle r \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

[0256] 下面示出了对应的解锁脚本, 其中签名 sig_r 使用r, 花费者鲍勃(B) 可以使用任何私钥/公钥对计算签名。应当注意的是, sig_r 是(r, s)。

[0257] $\langle P_B \rangle \langle sig_r \rangle$

[0258] 图13示出了分步脚本分析。

[0259] 临时密钥k可以由爱丽丝生成并提供给鲍勃(以及可选地一个或多个其他潜在证明者)。可选地, k可以由鲍勃生成并提供给爱丽丝, 以设置只有鲍勃(或鲍勃选择与之共享k的任何人) 才能破解的r难题。在任何一种情况下, 证明者鲍勃都必须相信发送者爱丽丝不会自己花掉交易, 因为她知道r难题的解(k)。为了防止出现这种情况, 证明者鲍勃可以创建难题, 然后将r值发送给爱丽丝, 以便她在创建r难题交易时使用。之后, 鲍勃可以在以后使用任何私钥/公钥对赎回输出, 只要他保留值k, 这就是r-难题的解, 并且可以被视为密钥的一种形式。另一方面, 在某些情况下, 爱丽丝知道k这一事实可能是一个有利的特征。例如, 这可以用来创建私钥难题, 并通过该难题进行通用的原子交换。

[0260] 图9通过与支付到公钥哈希(P2PKH) 相类比, 示出了r难题的另一示例, 其在本文中称为“支付到r难题哈希”(P2RPH)。为了提高安全性和私密性, r值可以在放置到 Tx_1 之前进行哈希处理(其将通过网络106的节点104传播并被放置在区块链150上)。与P2PKH类似, 区块链上只有公钥的哈希值, 而不是公钥本身, r难题也是如此。

[0261] 在这里, r难题再次要求 Tx_2 的解锁脚本包括r部分的提交实例r。 Tx_1 的锁定脚本再次包括对r部分的测试, 但这一次是以 r' 哈希值形式的r部分的压缩实例的形式, 即 $h = H_{puz}(r')$ 。这将与提交实例r进行比较。在此方法中, Tx_2 中的解锁脚本同样必须至少包含鲍勃签名的r部分(r) 和s部分(s)。它还可以包括相应于鲍勃用于对m进行签名的私钥V的公钥P。 Tx_1 的锁定脚本被配置为当由节点104处的脚本引擎402运行时, 从 Tx_2 的解锁脚本中获取r、s

和P并执行以下操作：

[0262] I) 检查 $h = H_{\text{puz}}(r)$ ，

[0263] II) 计算 $R' = H_{\text{sig}}(m) s^{-1} \cdot G + r s^{-1} \cdot P$ ，

[0264] III) 检查 $[R']_x = r$ ，

[0265] 其中 h 是从 Tx_1 的锁定脚本中获取的， s 、 r 和 m 是从 Tx_2 的解锁脚本中获取的。哈希值 $h = H_{\text{puz}}(r)$ ，其中 H_{puz} 是在 r 的哈希难题中使用的哈希函数。它可以是任何形式的哈希函数。它可以是 H_{sig} 的相同或不同形式的哈希函数。无论采取何种形式，都可以假定 H_{puz} 的形式是预定的并且在两端都是已知的。鲍勃的公钥 P 也可以从解锁脚本 Tx_2 获取，或者可以通过其他方式获知，例如如前所述从 (r, s) 和 m 或 (r, s) 和 m 推导出。

[0266] 锁定脚本被配置为在步骤I) 和III) 中的检查均为TRUE的条件下返回结果“TRUE”，否则返回结果“FALSE”。检查I) 可以在II) -III) 之前或之后执行，III) 确实必须在II) 之后执行。

[0267] 同样，与图8的情况一样，步骤II) 和III) 单独是由ECDSA验证函数执行的常规操作。因此，在大多数协议中，它们可以由专用操作码调用，例如脚本中现有的Checksigs操作码(OP_CHECKSIG)。步骤I) 可以使用通用操作码单独编码到锁定脚本中。

[0268] 交易挑战 Tx_1 中的锁定脚本示例如下所示：

$TxID1$	
输入	输出
[0269] 任何花费 输入	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_HASH160 $\langle h \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

[0270] 可以使用在发送者和接收者之间一致的任何类型的哈希函数。然而，为了与P2PKH标准保持一致，使用OP_HASH160、双重哈希SHA-256和RIPEMD-160。

[0271] 下面示出了对应的解锁脚本（与先前部分相同），其中签名 sig_r 使用 r ，花费者鲍勃(B)可以使用任何私钥/公钥对计算签名：

[0272] $\langle P_B \rangle \langle \text{sig}_r \rangle$

[0273] 因此，图9的示例与图8类似，不同之处在于它使用 r 部分的哈希值作为 r 难题的基础，而不是使用 r 的未变换实例。

[0274] 应当注意的是，在上述任何一种情况下，都不排除 Tx_1 的解锁脚本可能会对“TRUE”结果施加额外标准。例如，锁定时间或附加签名要求。

[0275] 上述任何一种技术的示例用例都作为一般知识挑战。考虑任何具有某个解 k 或可以哈希处理到 k 的解的挑战。然后，爱丽丝可以创建一个 r 难题，该难题与所述难题相耦合。即，她可以定义 $r = [k \cdot G]_x$ 。

[0276] 例如，爱丽丝是一名数学教授。她可以构建一个 r 难题交易 Tx_1 ，其中潜在的 k 值是激励学生破解的数学问题的解。无论谁设计出解决方案，都可以使用该解来创建签名 (r, s) ，其中 r 将匹配锁定脚本中的值，因此，从而申请奖励。签名不仅提供了真实性，还充当了解的知识证明，而不会将解决方案透露给其他任何人。因此， r 难题提供了一种安全机制来证明具备某些解或一般信息知识，而不存在暴露风险。它优雅地重用解锁脚本中所需的签名，并允许找到解的任何人以隐私方式申请奖励，因为任何公钥 P 都可以使用。

[0277] 该方案还可用作令牌或数字票证的形式。例如,活动组织者可以向参与者发放不同值的 k 作为数字票证。当参与者想要参加活动时,他们可以通过使用 r 难题来证明他们具备秘密令牌的知识。

[0278] 作为另一示例用例, r 难题可以用作签字人授权方案,其中一方可以将签名权委托给另一方。考虑 r 难题交易 Tx_1 ,该交易只有在提供具有与锁定脚本匹配的 r 值的签名时才能解锁。这意味着只有知道 k 值(其中, $[k \cdot G]_x = r$)的人才能生成此类签名。然而,如果此人将 k 的知识传递给其他人,那么这实际上是授权其他人代表他或她签名。

[0279] 例如,假设爱丽丝想要接收投递物品,但她担心自己可能无法接收投递物品。她向鲍勃和查理提供一份 k 副本,以便他们可以代表她接收投递物品。如果戴夫(Dave)正在投递包裹,她必须获取带有预期 r 值的签名,才能将包裹提供给鲍勃。

[0280] 在这样的场景中,可以将 k 视为充当临时私钥,将 r 视为充当临时公钥;它们分别类似于 V 和 P ,不同之处在于 k 和 r 与特定身份无关。

[0281] 可选安全功能#1

[0282] 如果发布了基于 k 的签名,则知道 k 的值的任何人都可以推导出用于创建签名的密钥 V 的值。这可以通过在下面的签名等式中破解 V 来实现。

[0283] $s = k^{-1} (H(m) + rV) \bmod n$

[0284] 求解 V ,可以得到:

[0285] $V = r^{-1} (sk - H(m)) \bmod n$

[0286] 这不会带来重大风险,因为在许多情况下,交易的接收者是唯一知道 k 的人。在其他情况下,花费者必须小心,永远不要重复使用私钥 V ,该私钥用于对 r 难题的解进行签名。良好的安全实践表明,用户最好不要重复使用公钥/私钥对 (P, V) ,而是在收到新资金时始终使用新的公钥/私钥对。

[0287] 原则上,公钥-私钥对 (P, V) 是“永久的”。换句话说,它可以使用许多次。随机临时密钥 k 的使用应确保这一点。然而,已经发生了随机数生成器实现不佳的事件。

[0288] 如果使用相同的临时密钥 k 和相同的私钥对两个不同的消息进行签名,则可以从这两个签名推导出私钥 V 。即在给定 (r, s) 和 k 的情况下,可以算出 V ,其中 $r = [k \cdot G]_x$ 和 V 是签名中使用的公钥 P 的私钥。如果随机数生成器在签名过程中失败,它可能会生成与上次相同的随机数,从而将私钥泄露给公众。为了解决这个问题,人们开始避免重复使用公钥,而不是修复随机数生成器。

[0289] 在本示例中,如果爱丽丝知道 k 但她不知道鲍勃的公钥的私钥 V 。当爱丽丝将 k 传递给鲍勃时,鲍勃将能够通过使用其私钥提供 (r, s) 来破解这个难题。当爱丽丝看到签名时,由于她知道 k ,因此她将能够推导出 V 。这可能不是鲍勃所希望的。因此,鲍勃最好避免重复使用 (P, V) 。

[0290] 但是,这样做存在的一个问题在于,鲍勃的公钥 P 不能用作标识鲍勃的持久方式。

[0291] 为了解决这个问题,根据在本文中所公开的实施例,鲍勃可以使用具有对应公钥 P_2 的单独私钥 V_2 时在 Tx_2 中包括鲍勃的附加签名 sig_2 。他还包括 P_2 和额外的签名。因此,存在两种类型的公钥-私钥对。第一种类型是为一次性使用而动态生成的公钥-私钥对。另一种类型是根据一些额外协议(例如,HD钱包)生成的公钥-私钥对。鲍勃可以将第一种类型的密钥对用于 r 难题签名,并将第二种类型的密钥对用于第二签名。

[0292] 然后,基于公钥与身份之间的映射,爱丽丝可以使用另一公钥来查找鲍勃的身份,例如鲍勃的适当名称、用户名或网络地址。例如,该映射可以在将公钥映射到身份的公共数据库中可用,或者该映射可以简单地在爱丽丝与鲍勃之间预先约定(例如,私下存储在爱丽丝的计算机设备102a上)。

[0293] 再次考虑签字人授权用例。例如,爱丽丝想要接收投递物品,但可能无法亲自接收投递物品。她向鲍勃和查理提供一份 k 副本,以便他们可以代表她接收投递物品。戴夫正在投递包裹,他必须获取带有预期 r 值的签名。现在,假设戴夫还需要验证接收者的身份,以满足其记录或法规要求。

[0294] 假设鲍勃有接收投递物品的机会。如果鲍勃基于 k 生成其公钥和签名,则爱丽丝和查理都能够计算出鲍勃的私钥 V 。如果公钥仅供一次性使用,这就不是问题。然而,如果鲍勃将来需要该公钥来证明自己的身份,则不是理想的选择。

[0295] 为了解决该问题,实施例可以在 Tx_2 中包括另一签名,该签名独立于可用于标识鲍勃的 r 难题。例如,可以将额外的签名和对应的公钥 P_2 添加到戴夫接受的同一交易中的OP_RETURN输出(不可花费的输出)。替代方案是在 r 难题交易的锁定脚本中包括额外的OP_CHECKSIG。通过浏览交易和用于额外签名的公钥,爱丽丝可以知道谁代表她签名。

[0296] 在其他一些情况下,可能会担心值 k 在使用之前泄露。为了解决该问题,爱丽丝可以将P2PKH添加到 r 难题交易以提高安全性。假设爱丽丝想将其签名权委托给鲍勃。爱丽丝从鲍勃那里获取一次性公钥 P_2 ,并创建 r 难题交易,该交易不仅指定 r 值,而且还指定额外的公钥 P_2 。

[0297] 为了使爱丽丝自己也能够签名,爱丽丝可以选择创建2取1的MultiSig。锁定脚本的示例如下:

$TxID$	
输入	输出
[0298] 任何花费输入	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\{r\}$ OP_EQUALVERIFY OP_OVER OP_CHECKSIGVERIFY OP_CHECKSIGVERIFY OP_1 $\{Alice's PK\}$ $\{Bob's P_2\}$ OP_2 OP_CHECKMULTISIG

[0299] 应当注意的是, r 难题提高了灵活性,因为爱丽丝可以选择何时将 r 难题的解(即签名权)传递给鲍勃。她可以决定是否继续,即使在交易被挖掘之后也是如此。

[0300] 如果 k 泄露,则人们可以发现用于使用已泄露 k 进行签名的私钥。然而,还有另一私钥 V_2 :该私钥链接到可用于标识鲍勃的公钥的私钥。要使输出受到攻击,攻击者必须获取两个独立的秘密,这比仅泄露其中一个更加困难。

[0301] 应当注意的是,在上述示例中, Tx_2 的锁定脚本通过常规的P2PKH锁定至鲍勃的额外公钥 P_2 (通过额外的签名而不是 r 难题中使用的签名解锁)。 r 难题技术为用户提供了额外的选择。在一些应用中,可能需要使用 r 难题,以便允许证明者应对挑战,而不管身份如何。另一方面,在一些其他应用中,哈希难题和P2PKH的组合可能仍然是合乎需要的,并且 r 难题可以可选地与其结合使用。将在下一节来更详细地讨论这方面。

[0302] 然而,如果对应于 P_2 的额外签名是查找身份和/或确保安全所需的,但是没有像在

P2PKH中那样预先绑定到特定证明者的身份的 Tx_1 的锁定脚本,则可以相应地修改上述锁定脚本。换句话说,它只能在额外的签名上包含Checksigs,而不能在对应的公钥 P_2 上包含OP_EQUALVERIFY。

[0303] P2PKH+P2PRPH

[0304] 与将 r 难题用作知识证明的方式相同,P2PKH输出也是对应于P2PKH输出中的公钥的私钥的知识证明。这实际上是通过将难题 k 替换为映射到P2PKH输出中的公钥 $P_{\text{puzzle}} = V_{\text{puzzle}} \cdot G$ 的私钥 V_{puzzle} 来实现的。在 r 难题的实施例中,知识证明随附公钥,该公钥可由证明者选择并可用于链接到身份。在P2PKH中,通常的花费签名和公钥(证明具备秘密难题的知识)必须随附另一签名和公钥才能链接到特定身份。通常,证明者可以向P2PKH解锁添加另一签名和公钥,该签名和公钥对应于锁定脚本的另一OP_CHECKSIG,如下所示。

$TxID$	
输入	输出
[0305] 任何花费 输入	OP_DUP OP_HASH160 $\langle H(P_{\text{puzzle}}) \rangle$ OP_EQUALVERIFY OP_CHECKSIGVERIFY OP_CHECKSIG

[0306] 对应的解锁脚本如下所示:

[0307] $\langle \text{sig}P_{\text{ID}} \rangle \langle P_{\text{ID}} \rangle \langle \text{sig}P_{\text{puzzle}} \rangle \langle P_{\text{puzzle}} \rangle$

[0308] 应当注意的是, $\langle \text{sig}P_{\text{ID}} \rangle \langle P_{\text{ID}} \rangle$ 与难题甚至交易之间没有加密链接。实际上,矿工和任何人都可以用另一公钥上的另一签名替换它们,这种方式在一定程度上类似于矿工拦截未解决的哈希难题。矿工或拦截者无法更改消息以将资金发送给自己(与未解决的哈希难题一样)。然而,没有什么能够阻止他们将 $\langle \text{sig}P_{\text{ID}} \rangle \langle P_{\text{ID}} \rangle$ 替换为自己的身份,使其看起来与自己的身份相关联。

[0309] 另一方面,基于在本文中所公开的技术,可以在同一脚本中同时使用P2PKH和P2RPH来强制第二签名上的加密链接,以免像上述情况那样被拦截和替换。

$TxID$	
输入	输出
[0310] 任何花费 输入	OP_DUP OP_HASH160 $\langle H(P_{\text{puzzle}}) \rangle$ OP_EQUALVERIFY OP_CHECKSIGVERIFY OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_HASH160 $\langle H(r_{\text{puzzle}}) \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

[0311] 在锁定脚本中, $P_{\text{puzzle}} = V_{\text{puzzle}} \cdot G$,而 $r = [R]_x = [V_{\text{puzzle}} \cdot G]_x$,使得它们在本质上是等效的。它们实际上并不相等,因为公钥既可以压缩,也可以解压缩,而且无论哪种方式,都会在公钥前面加上前缀。还可以在从签名提取 r 值时显式地将前缀添加到脚本中,以获取 $\langle H(P_{\text{puzzle}}) \rangle = \langle H(r_{\text{puzzle}}) \rangle$,如下所示。然而,这并没有真正带来多少好处。

TxID	
输入	输出
[0312] 任何花费 输入	OP_DUP OP_HASH160 $\langle H(P_{\text{puzzle}}) \rangle$ OP_EQUALVERIFY OP_CHECKSIGVERIFY OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_PUSHDAT1 $\langle r - \text{prefix} \rangle$ OP_SWAP OP_CAT OP_HASH160 $\langle H(r_{\text{puzzle}}) \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG

[0313] (应当注意的是,在锁定脚本的示意性表示中, $H(\dots)$ 用三角括号 $\langle \rangle$ 表示,实际上是指哈希值,而不是哈希函数。三角括号 $\langle \rangle$ 表示将该值放置在堆栈上。)

[0314] 这两笔交易的对应解锁脚本如下所示:

[0315] $\langle P_{\text{ID}} \rangle \langle r_{\text{puzzle}}, s \rangle \langle \text{sig}_{P_{\text{puzzle}}} \rangle \langle P_{\text{puzzle}} \rangle$

[0316] 可选安全功能#2

[0317] 上述方法的另一潜在安全漏洞是签名可伪造性。这可能会被试图申领资金的矿工利用(类似于哈希难题)。(从花费者)收到交易的矿工可以更改交易以将资金发送给自己,同时使用与原始交易中使用的签名相同的签名。

[0318] 具体操作如下:

[0319] 将 $P = V \cdot G$ 设为公钥/私钥对,用于对由 m 表示的原始交易进行签名,以获取签名 (r, s) ,使得:

[0320] $r = [k \cdot G]_x$,

[0321] $s = k^{-1} (H(m) + rV) \bmod n$ 。

[0322] 要花费该交易,花费者将使用以下解锁脚本:

[0323] $\langle P \rangle \langle r, s \rangle$

[0324] 收到该交易的矿工可以使用以下新解锁脚本将交易更改为新交易(由 m' 表示),通过该新交易将资金发送给自己:

[0325] $\langle P' \rangle \langle r, s \rangle$

[0326] 其中 $P' = V' \cdot G$ 是公钥/私钥对,使得:

[0327] $V' = V + r^{-1} [H(m) - H(m')]$,

[0328] $P' = P + r^{-1} [H(m) - H(m')] \cdot G$ 。

[0329] 应当注意的是,矿工不需要知道 V' (因为他们不知道 V)。使用以下计算完成验证过程:

[0330] $R' = H(m) s^{-1} \cdot G + r s^{-1} \cdot P$

[0331] 当且仅当 $(R')_x = r$ 时签名才有效,否则签名无效。

[0332] 对于新交易 m' 和新解锁脚本,验证过程如下:

[0333] $R' = H(m') s^{-1} \cdot G + r s^{-1} \cdot P'$

[0334] $= H(m') s^{-1} \cdot G + r s^{-1} \cdot \{P + r^{-1} [H(m) - H(m')] \cdot G\}$

[0335] $= r s^{-1} \cdot P + H(m) s^{-1} \cdot G$

[0336] $= r$

[0337] 应当注意的是,素数表示法在这里的含义与前面不同(在该上下文中,素数不指引用实例)。

[0338] 为了解决该潜在漏洞,各实施例可以在另一消息 m_{sighash} 的解锁脚本中包括另一额外签名 sig' ,除非矿工知道密钥 V ,否则他们将无法提供该签名。在这种情况下,解锁脚本为:

[0339] $\langle \text{sig}' \rangle \langle P \rangle \langle \text{sig}_r \rangle$

[0340] sig' 可能是不同消息 m_{sighash} 上的签名,因此要更改消息,只需使用与原始消息不同的SIGHASH标志(例如,SIGHASH_NONE而不是默认标志SIGHASH_ALL)。此外, sig' 必须使用 r 的不同值,以便它不会泄露私钥(因为私钥可以从使用相同临时密钥的两个签名推导出)。最后,交易需要在末尾包括另一OP_CHECKSIG,如下所示。

<i>TxID</i>	
输入	输出
[0341] 任何花费 输入	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle r \rangle$ OP_EQUALVERIFY OP_OVER OP_CHECKSIGVERIFY OP_CHECKSIG

[0342] 这必须使用与 r 难题相同的公钥 P ,以便只有知道公钥 P 的私钥 V 的人才能创建另一签名,因此上述攻击无法实现。

[0343] 攻击者试图将公钥替换为攻击者不知道其私钥的另一公钥。为了防御该攻击,挑战还要求具备私钥的知识。在这种情况下,一个签名是不够的。因此,需要两个签名。这两个签名均被视为同一私钥的知识证明。这是安全的,因为挑战强调这两个签名将拥有不同的临时密钥。

[0344] 联合加法 r 难题

[0345] r 难题方法也可以扩展到包括联合加法 r 难题,其基本上是两个单一 r 难题的组合($R_{\text{joint}} = R_1 + R_2$,这里的“+”表示椭圆曲线点加法)。因此,证明者需要破解这两个难题,才能花费锁定的资金。然而,与上述方法类似,在这种情况下,创建了两个难题 r_1 和 r_2 ,每个难题分别具有自己的原像或解 k_1 和 k_2 ,其中 $r_i = [k_i \cdot G]_x$ 。该方法的工作原理如下。

[0346] Tx_1 中的解锁脚本需要一个使用 k_1 的签名和另一个使用 k_2 的签名。必须从每个签名中提取 r_i 并验证它们与 r 之间的关系。应当注意的是,不仅应该简单地将 r_1 和 r_2 相加,因为它们是两个椭圆曲线点的 x 坐标。相反,由于两个值的和与两个点的和的 x 坐标不同,因此必须执行点加法才能得到和。

[0347] 回顾有关椭圆曲线点加法的一些事实,考虑由以下等式定义的素数阶 n 椭圆曲线:

[0348] $y^2 = x^3 + ax + b \pmod{p}$,

[0349] 其中 p 是素数。根据曲线的组结构,可将曲线 $P_1 = (x_1, y_1)$ 和 $P_2 = (x_2, y_2)$ 上的两点相加,以得到第三点 $P_{\text{joint}} = (x_3, y_3)$,如下所示。应当注意的是,这里的 P 用作椭圆曲线上一般点的符号(不指公钥 P)。

[0350] $P_{\text{joint}} = P_1 + P_2$ 。

[0351] 点 P_{joint} 定义为椭圆曲线上的点,该点穿过连接 P_1 和 P_2 且随后围绕 x 轴反射的线。用于点相加的公式为:

$$[0352] \quad x_3 = \lambda^2 - x_1 - x_2 \bmod p \quad (1)$$

$$[0353] \quad y_3 = \lambda(x_1 - x_3) - y_1 \bmod p, \quad (2)$$

[0354] 其中对于 $P_1 \neq P_2$ 的情况:

$$[0355] \quad \lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod p \quad (3)$$

[0356] 对于 $P_1 = P_2$ 的情况:

$$[0357] \quad \lambda = \frac{3x_1^2 + a}{2y_1} \bmod p. \quad (4)$$

[0358] 例如,一些常见的区块链使用secp256k1约定,其中椭圆曲线参数由 $a=0$ 和 $b=7$ 给定。

[0359] 可以删除等式(4),因为除非对 r_i 使用相同的值,否则这些点不会相同。此外,还可以删除等式(2),因为主要关注 x 坐标(因为正在处理 r)。关注的主要等式为等式(1)。为了免于执行模逆($\frac{y_2 - y_1}{x_2 - x_1}$)和模方(λ^2)的需要,可以在创建交易并在脚本中使用该值之前链下计算 λ^2 。基本上,在这种情况下,将 x 替换为 r ,等式(1)会变为:

$$[0360] \quad r_{\text{joint}} \equiv \lambda^2 - r_1 - r_2 \bmod p \quad (5)$$

[0361] 重新设置后,最终可得到:

$$[0362] \quad r_1 + r_2 \equiv \lambda^2 - r_{\text{joint}} \bmod p \quad (6)$$

[0363] 通过使第一交易 Tx_1 的锁定脚本挑战第二交易 Tx_2 以在其解锁脚本中包括满足 $r_1 + r_2 = \lambda^2 - r_{\text{joint}} \bmod p$ 的 r_1 和 r_2 的值,可以在 r 难题中运用这一点。优选地, Tx_1 的锁定脚本还将验证相应的签名 (r_1, s_1) 和 (r_2, s_2) ,以便确认联合 r 难题的解的两个部分 r_1 和 r_2 中的每个部分对应于有效签名。

[0364] 图10示出了示例性实现方式。第二交易 Tx_2 的解锁脚本包括至少两个签名: (r_1, s_1) 和 (r_2, s_2) ,由相应私钥 V_1 和 V_2 生成。例如,基于鲍勃和查理的相应私钥,签名可以是两个不同当事方的签名。解锁脚本还可以包括他们的相应公钥 P_1 和 P_2 ,该相应公钥分别对应于 V_1 和 V_2 。第一交易 Tx_1 的锁定脚本包括用作 r_{joint} 的参考的参考数据 D_{ref} 的一部分。这可以包括联合 r 值 r_{joint} 本身的直接值或 $(\lambda^2 - r)$ 的值或 $(\lambda^2 - r) \bmod n$ 的值。

[0365] Tx_1 的锁定脚本被配置为当由节点104处的脚本引擎402运行时,从 Tx_2 的解锁脚本中至少获取两个签名的 r 部分 r_1 和 r_2 以及 s 部分 s_1 和 s_2 并执行以下操作:

$$[0366] \quad \text{I) 检查 } r_1 + r_2 = \lambda^2 - r_{\text{joint}} \bmod p,$$

$$[0367] \quad \text{II) 计算 } R_1' = H_{\text{sig}}(m) s_1^{-1} \cdot G + r_1 s_1^{-1} \cdot P_1,$$

$$[0368] \quad \text{III) 检查 } [R_1']_x = r_1,$$

$$[0369] \quad \text{IV) 计算 } R_2' = H_{\text{sig}}(m) s_2^{-1} \cdot G + r_2 s_2^{-1} \cdot P_2,$$

$$[0370] \quad \text{V) 检查 } [R_2']_x = r_2,$$

[0371] 注意:在等式末尾写上“ $\bmod p$ ”意味着等式中的所有运算(包括等式)都是在 $\bmod p$ 下完成的。

[0372] 此外,还可以从 Tx_2 中的解锁脚本中获取公钥 P_1 和 P_2 。或者,如前所述,可选地借助包括在 Tx_2 中的一位标志,可以从其相应的签名部分 (r, s) 推导出公共签名 P 。作为另一替代方案,不排除验证者已经通过某种其他方式知道并查找了公钥中的一个或两个。

[0373] 当锁定脚本与解锁脚本一起运行时,它们在三项检查I)、III)和V)全部为TRUE的条件下输出“TRUE”结果。这些检查可以以任何顺序执行,并且上面的附图标记不一定意味着强加的顺序(当然,检查III)必须在相应的计算II之后执行,并且检查V)必须在相应的计算IV之后执行)。此外,还应当注意的是,上述符号并不一定意味着在两个不同签名中使用的 H_{sig} 必须是相同形式的哈希函数。

[0374] Tx_1 的锁定脚本中的参考数据 D_{ref} 可以包括 r_{joint} 本身的直接值,其在检查I)中使用。在这种情况下, λ^2 和 p 中的一个或两个可以预先存储在相应节点104的存储器中,并从存储器中取出以用于检查。或者, λ^2 和 p 中的一个或两个可以作为单独的值包括在 Tx_1 的锁定脚本中的参考数据 D_{ref} 中,并从中取出以用于检查。作为另一可能性, λ^2 和 p 中的一个或两个可以包括在证明交易 Tx_2 的解锁脚本中,并且作为要在检查I)中使用的解的一部分呈现。此外,还应当注意的是, p 和 λ^2 不需要来自同一来源,并且可以从这些来源的任何组合中获取。此外,还应当注意的是, λ^2 可以等效地以 λ 的形式存储或呈现,并且在检查I)中特别地计算平方。

[0375] 在另一替代方案中, Tx_1 的锁定脚本中的参考数据 D_{ref} 可以包括 (λ^2-r) 的值或 $(\lambda^2-r) \bmod p$ 的值。在前一种情况下,仅需从其他位置获取 p 。同样,它可以是在节点104处预先存储的默认值,或者可以是包括在 Tx_1 的锁定脚本中的另一元素 D_{ref} ,或者可以包括作为证明交易 Tx_1 的解锁脚本所呈现的解的一部分。

[0376] 安全的 r 难题切换

[0377] 此外,可选扩展使用联合 r 难题来防止私钥泄漏漏洞(请参见前面“可选安全功能1”下的讨论)。如果许多实体具备 k 的知识并且其中一个实体在签名中使用 k ,那么具备 k 的知识的每个人都可以推导出用于签名的私钥。这种想法是使用联合 r 难题将 k 分成两个部分 k_1 和 k_2 ,使得 $k_1+k_2=k$,并且类似地 $R_1+R_2=R$ (使用EC点加法),因为 $R_i=k_i \cdot G$ 。这样做的方法几乎与联合 r 难题(第4节)相同,但 k_1 和 k_2 并不是事先设定的。这样做的方法类似于脚本中的点加法技术。在锁定脚本中,设置 R 的 x 和 y 坐标。要解锁或使用UTXO,花费者必须提供 λ 以及 R_1 和 R_2 的坐标。然后,在锁定脚本以及 R 的 x 和 y 坐标中,锁定脚本必须使用以下等式检查 R_1 和 R_2 坐标是否在椭圆曲线上:

$$[0378] \quad y^2 = x^3 + ax + b.$$

[0379] 之后,脚本必须检查等式1和等式4(如下所示)是否适用。不需要验证等式2,因为 r 值仅包含点 R 的 x 坐标。

$$[0380] \quad x_3 = \lambda^2 - x_1 - x_2 \bmod p \quad (1)$$

$$[0381] \quad \lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod p \quad (3)$$

$$[0382] \quad \lambda(x_2 - x_1) = y_2 - y_1 \bmod p \quad (4)$$

[0383] 应当注意的是,证明者选择 k_1 和 k_2 的任何值,使得 $k_1+k_2=k$,以便知道 k 的任何其他人都不能推导出鲍勃(或查理等)的私钥,因为他们不知道鲍勃选择的值 k_1 和 k_2 。他们只知道这两个值的和等于 k 。解锁脚本必须包括如下所示的元素:

$$[0384] \quad \langle sig'_1 \rangle \langle P_{B1} \rangle \langle sig_{r1} \rangle \langle y_{r1} \rangle \langle sig'_2 \rangle \langle P_{B2} \rangle \langle sig_{r2} \rangle \langle y_{r2} \rangle \langle x_{inv} \rangle$$

[0385] 示例性用例

[0386] 例如,如果取用要调用的 λ^2-r 的值 $\langle simplified_r \rangle$,可以将其直接添加到 Tx_1 的锁

定脚本中,如下所示。

TxID	
输入	输出
[0387] 任何未花费的 输入	OP_DUP OP_4 OP_PICK OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_SWAP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP OP_ADD $\langle n \rangle$ OP_MOD $\langle simplified_r \rangle$ OP_EQUALVERIFY OP_OVER OP_CHECKSIGVERIFY OP_CHECKSIGVERIFY OP_OVER OP_CHECKSIGVERIFY OP_CHECKSIG

[0388] 锁定脚本的工作原理如下。

[0389] - 第一行获取位于堆栈顶部的两个签名 $\langle sig_{r_1} \rangle$ 和 $\langle sig_{r_2} \rangle$ 的副本。

[0390] - 第二行从 $\langle sig_{r_1} \rangle$ 中提取 r_1 。

[0391] - 第三行交换位于堆栈顶部的两个项目,以获取位于堆栈顶部的 $\langle sig_{r_2} \rangle$ 并从中提取 r_2 。

[0392] - 第四行将 r_1 与 r_2 相加,mod p ,并检查该值是否等于 $\langle simplified_r \rangle$,如(6)中所示。

[0393] - 第五行检查第一对签名在其相应公钥上是否有效。

[0394] - 第六行检查第二对签名在其相应公钥上是否有效。

[0395] Tx_2 中的相应解锁脚本。解锁脚本将包含: r_1 和 r_2 (它们是R的x坐标),它们的相应y坐标 y_1 和 y_2 以及 $x_{inv} = (r_1 - r_2)^{-1}$ 的值。在给定示例中,应如下所示,其中签名 $sig_{r_1}P_{B1}$ 使用 r_1 , $sig_{r_2}P_{B2}$ 使用 r_2 ,证明者鲍勃(B)可以使用任何私钥/公钥对计算签名。

[0396] $\langle sig'_1 \rangle \langle P_{B1} \rangle \langle sig_{r_1} \rangle \langle y_{r_1} \rangle \langle sig'_2 \rangle \langle P_{B2} \rangle \langle sig_{r_2} \rangle \langle y_{r_2} \rangle \langle x_{inv} \rangle$ 。

[0397] 额外的签名 sig' 是为提高安全性而附加的可选功能(请参见“可选安全功能#2”的部分)。然而,并非所有可能的实施例都必须如此。应当注意的是, sig'_1 使用 r_1 的不同 r 值, sig'_2 使用 r_2 的不同 r 值。正如前面部分所指出的,这是为了防止其他人修改交易和伪造签名。

[0398] 该方案可以扩展到涵盖I加法,使得 $r = r_1 + \dots + r_i$ 。由于该方案可以用于两点加法,因此通过归纳,可以通过多次重复用于两点加法的技术来将其扩展到涵盖I加法。

[0399] 作为加法 r 难题的示例性用例,这种想法可用于允许一组参与者解锁交易的某些输出。

[0400] 考虑一组椭圆曲线点 $\{R_1, R_2, R_3, R_4, R_5, R_6\}$,使得 $R_1 + R_2 = R_3 + R_4 = R_5 + R_6 = R$,其中 $R_i = k_i \cdot G$ 。

[0401] 假设一组有三个人,鲍勃、查理和多拉。然后,可以通过执行以下操作来构建一个由 r 难题交易仿真的3取2的多重签名UTX0:

[0402] 1. 将 k_1 和 k_3 分配给爱丽丝。

[0403] 2.将 k_2 和 k_5 分配给鲍勃。

[0404] 3.将 k_4 和 k_6 分配给查理。

[0405] 4.构建具有执行以下步骤的锁定脚本的交易：

[0406] (i) 从 sig_{r_1} 和 sig_{r_2} 中提取 r_1 和 r_2 的部分

[0407] (ii) 检查 x_{inv} 实际上是逆运算,即 $(r_1 - r_2)^{-1} \cdot (r_1 - r_2) = 1$

[0408] (iii) 检查点是否在EC中,即 $y_{r_1}^2 = (r_1^3 + ar_1 + b) \bmod p$ 和 $y_{r_2}^2 = (r_2^3 + ar_2 + b) \bmod p$

[0409] (iv) 计算 $\lambda := (y_{r_1} - y_{r_2})(r_1 - r_2)^{-1} \bmod p$ 均部分

[0410] (v) 检查 $r = \lambda^2 - r_1 - r_2 \bmod n$,其中 r 是否事先指定

[0411] 每个当事方使用两个 k 的原因是,任意两个R点的总和涵盖三个人的子集。例如, $R_1 + R_2$ 涵盖爱丽丝和鲍勃的子集(在3取2的多重签名中),而 $R_1 + R_2$ 涵盖爱丽丝和查理的子集,依此类推。

[0412] 应当注意的是,步骤4中描述的锁定脚本需要两个签名 (r', s') 和 (r'', s'') ,其中 $r' + r'' = \text{simplified_r}$ (应当注意的是,此上下文中的素数符号是指不同的签名,而不是前面部分中使用素数与知识证明中提交的实例进行对比的引用实例)。这些签名中的第一签名来自一个当事方。第二签名来自另一当事方。

[0413] 因此,有三种可能的方法来解锁脚本：

[0414] 1. 爱丽丝和鲍勃,分别用 k_1 和 k_2 进行签名。

[0415] $\langle \text{sig}'_1 \rangle \langle P_{B1} \rangle \langle \text{sig}_{r1} \rangle \langle y_1 \rangle \langle \text{sig}'_2 \rangle \langle P_{B2} \rangle \langle \text{sig}_{r2} \rangle \langle y_2 \rangle \langle x_{\text{inv}} \rangle$

[0416] 2. 爱丽丝和查理,分别用 k_3 和 k_4 进行签名。

[0417] $\langle \text{sig}'_3 \rangle \langle P_{B3} \rangle \langle y_3 \rangle \langle \text{sig}_{r3} \rangle \langle \text{sig}'_4 \rangle \langle P_{B4} \rangle \langle \text{sig}_{r4} \rangle \langle y_4 \rangle \langle x_{\text{inv}} \rangle$

[0418] 3. 鲍勃和查理,分别用 k_5 和 k_6 进行签名。

[0419] $\langle \text{sig}'_5 \rangle \langle P_{B5} \rangle \langle y_5 \rangle \langle \text{sig}_{r5} \rangle \langle \text{sig}'_6 \rangle \langle P_{B6} \rangle \langle \text{sig}_{r6} \rangle \langle y_6 \rangle \langle x_{\text{inv}} \rangle$

[0420] 假设存在受信任第三方,该第三方生成所有秘密并将其分配给该组。该假设可在组织环境(例如,商业实体)中得到证明。

[0421] 可以通过选择不同的 R_i 点组合来推广应用。例如,设置 $R = R_1 + R_2 + R_4$ 将强制所有三个成员签名。

[0422] 基于账户的模型中的替代实现方式

[0423] 上文在很大程度上是根据基于输出的模型(例如,基于UTXO的模型)中的实现方式来描述的。然而,应当理解的是,这不是限制性的。图11示出了使用基于账户的模型的一种可能的替代实现方式。

[0424] 简而言之,在基于账户的模型中, r 难题功能可以包括在由用户调用的智能合约函数中。一方可以在智能合约中设置 r 难题值(或经过哈希处理的 r 难题值),然后另一方随后将签名提供给智能合约。

[0425] 在UTXO区块链架构中,第一交易的解锁脚本中包含的要求必须由第二交易的锁定脚本来满足,以便第二交易被视为有效并记录在区块链中。在当前上下文中,这是有益的,因为它利用了矿工在交易核实过程中已经完成的工作。作为当前上下文中的具体示例,交易已添加到区块链这一事实意味着它已由整个区块链网络中的节点验证,而这又意味着其

锁定脚本满足某些特定的有用要求。相关方不需要自己检查是否满足这些要求,他们可以根据交易已在区块链中成功重新编码这一事实简单地假设满足这些要求。这是因为脚本必须在完成时返回“TRUE”结果才能使交易有效(为了使交易有效可能还有其他要求),并且如果脚本返回“FALSE”结果(根据在本文中所使用的术语,这包括脚本失败的情况,例如因为OP_VERIFY操作码终止脚本),则交易无效。

[0426] 然而,在其他区块链模型(例如,某些基于账户的架构)中,不一定会反映交易有效性与交易代码运行结果之间的这种相互依赖关系。例如,在某些智能合约区块链中,交易可能是有效的,因此可以用于记录在区块链上,前提是它们满足区块链协议强加的一组“基本”有效性要求。因此,即使第二交易不满足包含在第一交易的代码中的某些要求,第二交易仍然可能被视为有效并记录在区块链中。例如,第一交易的代码可以是智能合约代码。

[0427] 假设第二交易是针对由第一交易创建的智能合约账户,则取决于智能合约代码来确定如何响应该交易,例如,如果不满足某些要求,则可以忽略(或者返回FALSE结果),而如果满足该要求,则可以使用从智能合约账户的余额中扣除并贷记的数字资产金额来奖励证明者(或者以其他方式返回TRUE结果)。从某种意义上说,这将智能合约(代理)的“代理级”处理(即,在智能合约代码中显式编码)从节点“隐式”执行的“协议级”处理(即对交易执行的处理)中进行抽象化处理,以确定其是否满足区块链网络借以操作的区块链协议强加的有效性要求。因此,在此类区块链架构中,在各个交易中由协议级的节点作出的“有效/无效”决定可以与通过智能合约在代理级针对该交易返回的“TRUE/FALSE”结果分离,因为交易可以被确定为在协议级有效,但仍然在代理级返回FALSE结果。

[0428] 这与UTXO架构相关,在UTXO架构中,需要返回“TRUE”结果的脚本才能使交易有效;如果脚本终止或完成,在堆栈上留下除TRUE以外的任何内容,则交易无效(如在本文中所使用的术语,这些结果中的任何一个结果都会构成“FALSE”的结果)。

[0429] 交易有效的基本要求之一可以是交易包括有效签名。因此,尽管在上述UTXO示例中,签名由挑战交易本身的代码来验证(例如,使用已验证签名并针对签名验证返回TRUE/FALSE的OP_CHECKSIG操作码,或者使用以相同方式检查签名并且附加地验证结果为TRUE的OP_CHECKSIGVERIFY操作码,如果不是,则脚本终止),在替代区块链架构中,签名可以在上述意义上由处理节点隐式验证,这可以避免需要在交易代码本身中对签名检查进行编码。

[0430] 作为当前上下文中的具体示例,交易可以在协议级视为有效,例如因为它包括有效签名,但是仍在应用级返回FALSE结果,例如因为不满足某些其他要求。

[0431] 图11示出了用于根据基于账户的模型来处理交易的节点软件400的替代方案,节点软件在此标记为400acc。该节点软件400acc的实例可以在网络106的基于账户的版本的每个节点104处实现。基于账户的节点软件400acc包括基于账户的协议引擎401acc、合约引擎402acc(在某种程度上类似于脚本引擎402)、应用级决策引擎404以及一个或多个区块链相关功能模块405的集合。在任何给定节点104处,这些模块可以包括以下三个模块中的任何一个、两个或全部:挖掘模块405M、转发模块405F和存储模块405S(取决于该节点的一个或多个角色)。协议引擎401acc被配置为识别交易的不同字段,并根据节点协议处理此类字段。节点软件400acc还在相应节点104的存储器中维护多个账户中的每个账户的账户状态406。这些账户可以例如包括爱丽丝、证明者(例如,鲍勃)和/或根据爱丽丝和证明者之间将要制定的合约而被借记或贷记的另一方的账户。合约引擎402acc被设置为根据在交易中接

收的智能合约的结果来修改账户状态。智能合约也称为“代理”。

[0432] 图11还示出了一对交易 TX_1^{acc} 和 TX_2^{acc} ,它们可以实现如上所述关于图7至图10的相同或相似的r难题功能。每笔交易包括源账户地址1102 (在源地址字段中) 和目标账户地址1103 (在目标地址字段中)。第一交易 TX_1^{acc} 包括源账户地址1102a和目标账户地址1103a。第二交易 TX_2^{acc} 包括源账户地址1102b和目标账户地址1103b。第一交易 TX_1^{acc} 还包括智能合约1101。智能合约1101可以包括爱丽丝的挑战(难题)。它可以由爱丽丝创建,也可以由代表爱丽丝的第三方使用爱丽丝提供的详细新创建。可选地,第二交易 TX_2^{acc} 可以包括用于携带用户指定的净荷数据的一个或多个自由数据字段1104。可以包括由证明者(例如,鲍勃)提供的难题的解的至少一部分。交易 TX_1^{acc} 和 TX_2^{acc} 还分别由爱丽丝和证明者签名。每笔交易还包括相应当事方的签名1105a、1105b。

[0433] 交易在网络106上广播。当协议引擎401acc接收到每笔交易时,它会隐式地验证签名1105是否有效。即,这是协议引擎401acc的固有特征,不需要在智能合约1101中指定。因此,协议引擎401acc至少在相应签名有效的条件下核实用于转发和/或挖掘的每笔交易有效。它还可能需要一个或多个附加条件才能满足有效性要求。如果有效,则应用级决策引擎404可以选择是否控制挖掘模块405M和/或转发模块405F分别挖掘和/或转发交易。

[0434] 在这种基于账户的模型中,爱丽丝、鲍勃和智能合约本身分配有不同的账户,具有不同的账户地址。交易被视为“从”其源地址字段中的地址发送“至”其目标地址字段中的地址。为了创建智能合约账户,将包含智能合约字节码的交易上载到交易中的区块链。对于此类账户创建交易,目标字段中的目标地址1103应为区块链中从未使用过的地址,并且一旦交易被接受,该地址就会成为新创建的智能合约账户的地址。此后,可以将另一交易发送到该地址,以便“调用”智能合约,即使智能合约的字节码能够根据该另一交易运行。“目标”地址1103充当用于制定合约的中介地址—爱丽丝将 TX_1^{acc} 发送到该地址以创建指定一个或多个要求的智能合约;鲍勃将 TX_2^{acc} 发送到同一地址以调用智能合约,转而使智能合约验证 TX_2^{acc} 是否满足那些指定要求。“源”地址1102指定作为合约当事方的用户的账户—在智能合约确定 TX_2^{acc} 确实满足指定要求的情况下,智能合约可以被配置为从其自身账户余额中扣除数字资产金额,并使得在 TX_2^{acc} 中具有源地址1102b的账户(即鲍勃的账户)余额以该金额(直观地,通过发送 TX_2^{acc} ,鲍勃有效地请求智能合约(如在目标地址字段中标识的)贷记其账户(如在源地址字段中标识的)。

[0435] 当协议引擎401acc接收到 X_2^{acc} 时,在其有效的条件下,它将查找与 TX_2^{acc} 中的目标地址1103b匹配的账户。假设 TX_1^{acc} 已处理并有效,该账户将凭借 TX_1^{acc} 而存在,并将与 TX_1 中提供的智能合约代码相关联。作为响应,协议引擎401acc控制合约引擎402acc从 TX_1^{acc} 运行智能合约1101,根据合约中定义的标准,从智能合约的一个或多个字段获取数据作为操作数数据。操作数数据可以例如包括来自一个或多个自由数据字段1104的数据和/或来自签名字段1105b的签名。在 TX_2^{acc} 的操作数数据满足在 TX_1^{acc} 的智能合约1101中

定义的一个或多个标准的条件下,合约引擎402acc根据在智能合约1101中定义的修改来修改一个或多个当事方(爱丽丝、证明者和/或一个或多个第三方)的账户状态406。否则,不对账户状态406进行该修改。然而,应当注意的是,在某些基于账户的系统中,智能合约的结果不是交易有效性的条件。因此,如果 Tx_2^{acc} 未能满足在 Tx_1^{acc} 的智能合约1101中设置的标准,则 Tx_2^{acc} 仍将作为失败交易的记录被传播和挖掘到区块中。它还可以仍然影响挖掘费用(因此协议引擎401仍然可以修改当事方之一和获胜矿工的账户状态406)。

[0436] 为了实现r难题,可以将r难题功能中的至少一些编码到 Tx_1^{acc} 的智能合约1101中,并且可以在 Tx_2^{acc} 的一个或多个数据字段1104中呈现解。例如,这可以用于实现图7的变体。可选地,可以利用协议引擎401acc的一些隐式签名验证功能,例如实现图8至图10的变体之一。在图8至图10的情况下,当协议引擎401acc验证 Tx_2^{acc} 的签名时,步骤II)和步骤III)可以是协议引擎401acc的隐含功能(请记住,签名验证本身是由协议引擎401acc实现的节点协议的固有特征)。因此,只需要在 Tx_1^{acc} 的智能合约1101中将步骤I)置于这一点之上。智能合约检查I)的结果是否为TRUE以及协议引擎401ac是否指示 Tx_2^{acc} 有效。如果两者都是肯定的,则声明验证的总体结果为“TRUE”,即鲍勃已经成功应对r难题设置的挑战。应对注意的是,在图8至图10的实现方式中,只有在图9和图10的情况下的数据值d需要包括在自由数据字段1104中。签名信息包括在签名字段1105b中。

[0437] 智能合约账户还具有索引的“数据寄存器”(未示出),它们是与账户相关联的(逻辑)数据存储元素。在上述UTXO模型中,值嵌入锁定脚本本身中,并且对于特定的智能合约代码1101也可能是如此。然而,智能合约的智能合约字节码可以替代地或附加地在存储在其一个或多个账户寄存器中的数据上运行。此外,通常可以在创建智能合约账户之后将值存储在智能合约账户寄存器中。因此,例如,可以通过包含智能合约字节码的挑战交易 $Tx_{1,\alpha}^{acc}$ 来创建智能合约账户。然后,可以向(现在存在的)智能合约账户发送单独的“中间”交易 $Tx_{1,\beta}^{acc}$,其作用是将特定值v存储在智能合约账户的寄存器\$中。智能合约可以被配置为仅接受来自指定的源账户地址(例如)的此类数据,例如,最初创建智能合约的同一方(爱丽丝)。当接收到 Tx_2^{acc} 时,由合约引擎402acc执行的操作(例如,“访问寄存器\$,并将该值与 Tx_2^{acc} 的数据字段\$中的值进行比较”)由挑战交易 $Tx_{1,\alpha}^{acc}$ 中提供的智能合约字节码定义;但\$中存储的值已由中间交易 $Tx_{1,\beta}^{acc}$ 设置。根据在本文中所使用的术语, $Tx_{1,\alpha}^{acc}$ 仍然被视为设置一个或多个要求的挑战交易,只是现在可以引用在一个或多个中间交易(例如, $Tx_{1,\beta}^{acc}$)中提供的数据来定义这些要求。

[0438] 因此,在一些实现方式中,挑战交易 $Tx_{1,\alpha}^{acc}$ 可以定义r难题的操作(例如,将证明交易 Tx_2^{acc} 的签名的r部分与寄存器\$中的值进行比较,以查看它们是否匹配等),但是\$中与证明交易 Tx_2^{acc} 的r部分进行比较的值可能已经由中间交易 $Tx_{1,\beta}^{acc}$ 设置。

[0439] 还请注意:一些基于账户的模型不需要将公钥P包括在签名1105中。相反,只需包

括1位标志flg即可。如上所述,可以从(r,s)和消息推导出两个可能的密钥P和-P。标志flg用于指示这两种可能的解中的哪一种方案实际上是对应于证明者用于对 Tx_2^{acc} 中的消息进行签名的私钥V的公钥。协议引擎401acc使用(r,s)和flg来推导出证明者的公钥P,而不是在 Tx_2^{acc} 中显式地接收它。该技术在基于输出的模型中也是可能的,并且不特定于基于账户的模型,但是在许多当前基于输出的模型中使用的脚本语言中恰巧没有用于从r和s推导出P的专用操作码,因此使用基于堆栈的语言的现有通用操作码将此功能显式地编码到解锁脚本中将是复杂的。还应当注意的是,某些基于账户的模型从用于对该交易进行签名的公钥推导出该交易的源地址。因此,源地址不必在交易中单独编码,并且在公钥从签名推导出的情况下,这意味着源地址也可以间接从签名推导出。

[0440] 此外,在一些基于账户的模型中,一笔交易 Tx_1^{acc} 的智能合约可能从两笔(或多笔)不同的第二交易 $Tx_{2,1}^{acc}$ 、 $Tx_{2,2}^{acc}$ 接收输入。因此,在第一交易 Tx_1 的智能合约中设置的挑战将由来自两笔(或多笔)不同的第二交易 $Tx_{2,1}^{acc}$ 、 $Tx_{2,2}^{acc}$ 的数据的组合来应对。例如,一笔第二交易 $Tx_{2,1}^{acc}$ 可以包括鲍勃的签名 (r_1, s_1) (如果需要,则为公钥 P_1 或标志 flg_1),而另一笔第二交易 $Tx_{2,2}^{acc}$ 可以包括查理的签名 (r_2, s_2) (如果需要,则为他的公钥 P_2 或标志 flg_2)。它们可以一起用于破解 Tx_1 的智能合约中设置的联合加法难题。

[0441] 结论

[0442] 应当理解的是,上述实施例仅通过示例的方式进行描述。

[0443] 更通俗地说,根据在本文中所公开的教导的第一实施例,提供了一种基于椭圆曲线数字签名算法(ECDSA)执行知识证明的计算机实现方法。所述方法包括在区块链网络的验证节点处:获取第一交易,所述第一交易包括可运行代码,所述代码包括用于评估基于联合r值 r_{joint} 定义的挑战的参考数据;接收一笔或多笔第二交易,所述一笔或多笔第二交易包含至少包括一对第一ECDSA签名 $i=1,2$ 中的每个相应签名的相应r部分 r_i 和s部分 s_i 的信息,其中每个所述第一ECDSA签名基于对应于相应第一公钥 P_i 的相应第一私钥 V_i 对所述一笔或多笔第二交易中的一笔第二交易的一部分进行签名;运行所述第一交易的所述代码。所述代码被配置为基于所述第一交易中的所述参考数据和在所述一笔或多笔第二交易中接收的所述r部分 r_i 来验证所述挑战是否得以应对,并且在所述挑战得以应对的条件下返回TRUE结果。所述挑战包括以下条件:

[0444] $r_1 + r_2 = \lambda^2 - r_{joint} \mod p$,

[0445] 其中 $r_1 + r_2$ 表示标量加法, $r_{joint} = [R_{joint}]_x$, $R_{joint} = R_1 + R_2$ (通过椭圆点加法), p 是素数模, $\lambda = \frac{y_2 - y_1}{x_2 - x_1} \mod p$, $R_i = k_i \cdot G$, $x_i = [R_i]_x$, $y_i = [R_i]_y$, k_i 是临时密钥, G 是椭圆曲线生成点, $[]_x$ 表示 $[...]$ 的x坐标, $[]_y$ 表示 $[...]$ 的y坐标, “ $\cdot$ ”表示椭圆曲线标量乘法。

[0446] 所述参考数据可以包括所述联合r值 r_{joint} 或 $\lambda^2 - r_{joint}$ 的值或 $(\lambda^2 - r_{joint}) \mod p$ 的值。

[0447] 在实施例中,所述第一交易中的所述参考数据可以包括所述联合r值 r_{joint} 本身。在这种情况下, p 可以预先存储在所述验证节点处,也可以包括在所述参考数据中,或者可以接收作为在所述一笔或多笔第二交易中的至少一笔第二交易中接收的所述信息的另一部分。类似地, λ 或 λ^2 可以预先存储在所述验证节点处,也可以包括在所述参考数据中,或者可

以接收作为在所述一笔或多笔第二交易中的至少一笔第二交易中接收的所述信息的另一部分。此外,还应当注意的是,在本文中所述的 r_{joint} 、 p 、 λ 或 λ^2 等某些值包括在所述参考数据中或从其他位置存储或提供等,则可以直接或等效地表示所述值,所述值可以变换或编码的形式表示,使得所述值能够由所述节点通过求逆所述变换来确定。

[0448] 根据在本文中所公开的教导的第二可选实施例,可以提供一种根据所述第一实施例所述的方法,其中:接收指示 λ^2 的数据作为在所述一笔或多笔第二交易中的至少一笔第二交易中接收的所述信息的一部分,并且基于所述接收的指示 λ^2 的数据来执行所述确定。

[0449] 或者,所述第一交易中的所述参考数据可以包括 λ^2-r 。在这种情况下, p 可以预先存储在所述验证节点处,也可以作为所述第一交易中的所述参考数据的一部分,或者可以接收作为在所述一笔或多笔第二交易中接收的所述信息的另一部分。或者,所述第一交易中包括的所述参考数据可以包括 $(\lambda^2-r_{\text{joint}}) \bmod p$,在这种情况下,不需要在任何第二交易中接收附加元素,也不需要在此验证节点处预先存储附加元素。

[0450] 根据本公开的第三可选实施例,可以提供一种根据所述第一实施例所述的方法,其中:所述第一交易中的所述参考数据包括 p 和 λ^2-r 的所述值。

[0451] 根据第四可选实施例,可以提供一种根据所述第一、第二或第三实施例所述的方法,其中:所述方法包括:获取每个所述第一ECDSA签名的所述相应第一公钥;应用所述ECDSA的验证函数以基于所述相应第一公钥和所述签名的部分来验证每个第一ECDSA签名,其中所述代码被配置为在每个所述第一ECDSA签名的所述验证的另一条件下返回所述TRUE结果。

[0452] 在基于输出的模型(例如,基于UTXO的模型)中,所述ECDSA验证函数可以由所述第一交易的输出(例如,UTXO)的锁定脚本中的操作码调用。所述操作码可以调用预先存储在所述验证节点上的所述ECDSA验证函数的实例。或者,例如在基于账户的模型中,ECDSA验证函数可以是所述节点的隐式函数,该函数作为节点协议的一部分自动运行,而不需要由所述第一交易中的所述代码(在基于账户的情况下可以是智能合约)显式调用。作为另一替代方案,不排除可以将所述ECDSA验证显式编码到所述代码中。

[0453] 根据第五可选实施例,可以提供一种根据所述第四实施例所述的方法,其中:所述获取所述第一公钥包括接收所述第一公钥作为所述一笔或多笔第二交易中的所述信息的一部分。

[0454] 或者,所述获取可以包括:从所述相应的ECDSA签名的所述 r 部分和 s 部分的组合推导出每个所述第一公钥。

[0455] 作为另一替代方案,不排除所述获取可以包括例如通过与所述第二交易相关联的侧信道接收所述第一公钥;或者在数据源中查找所述第一公钥。

[0456] 根据第六可选实施例,可以提供一种根据所述第一至第五实施例中任一个所述的方法,其中:所述代码被配置为输出所述TRUE结果,而不管使用谁的公钥作为所述第一ECDSA签名中的一个、一些或全部签名的所述相应第一公钥。

[0457] 根据第七可选实施例,可以提供一种根据所述第一至第六实施例中任一个所述的方法,其中:所述ECDSA基于以下形式的椭圆曲线: $y^2=x^3+7$ 。

[0458] 根据第八可选实施例,可以提供一种根据所述第一至第七实施例中任一个所述的方法,其中:所述ECDSA是secp256k1算法。

[0459] 根据第九可选实施例,可以提供一种根据所述第一至第八实施例中任一所述的方法,其中:每个所述第一ECDSA签名是不同的相应第二方的签名,所述挑战至少部分由第一方定义。

[0460] 例如,每一方可以是相应的个人,也可以是可以接受的组织。

[0461] 根据第十可选实施例,可以提供一种根据所述第九实施例所述的方法,其中:每个所述第一ECDSA签名的所述r部分和s部分由所述相应第二方使用相应的临时密钥 k_i 生成。

[0462] 根据第十一可选实施例,可以提供一种根据所述第十实施例所述的方法,其中:每个相应的临时密钥由所述第一方提供给所述相应第二方,反之亦然。

[0463] 根据第十二可选实施例,可以提供一种根据所述第十一实施例所述的方法,其中: $P_i = V_i \cdot G, k_i \in [1, n-1], R_i = k_i \cdot G, r_i = [R_i]_x, s_i = k_i^{-1} (H_{sig}(m) + r_i V_i) \bmod n$,其中 k_i 是所述相应的临时密钥, m_i 是所述签名的第二交易的所述相应的签名部分, H_{sig} 是用于在生成所述相应的ECDSA签名时对 m 进行哈希处理的哈希函数, n 是所述生成点的素数阶。

[0464] 对于两个不同的签名,签名的消息 m 可以不同或相同。对于两个签名, n 的值和所述哈希函数 H_{sig} 的形式可以相同。

[0465] 根据第十三可选实施例,可以提供一种根据所述第七至第十二实施例中任一所述的方法,所述方法包括:在由所述代码返回的所述结果为TRUE的条件下为所述第一方触发服务。

[0466] 例如,所述服务可能已由所述第一方委托,可以代表所述第一方执行,和/或可以为了所述第一方的利益而执行。所述服务可以是计算机化服务,并且所述触发可以包括自动触发所述服务。

[0467] 通过将所述临时密钥提供给所述第二方(“鲍勃”),这使所述第一方(“爱丽丝”)能够让鲍勃代表她对所述服务进行签名,而无需鲍勃向所述节点透露所述临时密钥 k 或将其发布在所述区块链上。此外,由于该过程不限于任何特定私钥(或其对应的公钥),这意味着爱丽丝还可以将所述临时密钥的副本提供给第三方(“查理”),然后鲍勃和查理中的任一方都可以成功地对所述服务进行签名。这是可能的,因为所述r部分用作挑战基础,并且所述r部分未映射到任何特定身份。

[0468] 根据第十四可选实施例,可以提供一种根据所述第七至第十三实施例中任一所述的方法,其中:所述接收所述一笔或多笔第二交易包括从所述第二方中的一个接收所述第二交易中的每笔交易。

[0469] 或者,可以改为从第三方接收所述一笔或多笔第二交易,所述第三方从所有所述第二方接收签名。

[0470] 根据第十五可选实施例,可以提供一种根据所述第七至第十四实施例中任一所述的方法,其中:在所述第二交易中接收的所述信息包括至少一个所述第二方的另一加密签名,所述至少一个所述第二方使用该方的另一私钥,所述另一私钥对应于另一公钥。

[0471] 所述另一签名可以是ECC签名或另一类型,例如RSA签名。

[0472] 根据第十六可选实施例,可以提供一种根据所述第十五实施例所述的方法,其中:映射可用于使所述第一方和/或第三方能够基于所述另一公钥来查找所述至少一个第二方的身份。

[0473] 例如,所述身份可以是所述第二方的个人姓名、公司名称、用户名或网络地址。例

如,所述第三方可以是上述服务的提供商。

[0474] 根据第十七可选实施例,可以提供一种根据所述第十五或第十六实施例所述的方法,其中:所述代码被配置为使用所述另一公钥验证所述另一加密签名,并在所述另一加密签名被验证的另一条件下返回所述TRUE结果。

[0475] 根据第十八可选实施例,可以提供一种根据所述第七至第十七实施例中任一所述的方法,其中:在所述第二交易中接收的所述信息包括所述第一方使用所述第一方的私钥的加密签名。

[0476] 在实施例中,所述方法可以包括获取对应于所述第一方的所述私钥的公钥,其中所述代码被配置为验证所述第二方的所述加密签名,并在所述第一方的所述加密签名被验证的另一条件下返回所述TRUE结果。

[0477] 在实施例中,映射可用于使所述第二方和/或第三方能够基于所述第一方的所述公钥来查找所述第一方的身份。例如,所述第一方的所述身份可以是所述第一方的个人姓名、公司名称、用户名或网络地址。

[0478] 根据第十九可选实施例,可以提供一种根据所述第七至第十八实施例中任一所述的方法,其中:在所述第二交易中接收的所述信息包括所述第一方中的一个、一些或全部中的每一个的附加的ECDSA签名,所述附加的ECDSA签名具有不同值的所述r部分(与所述第一ECDSA签名不同),但是使用与所述第一ECDSA签名相同的相应私钥;所述代码被配置为使用所述相应第一公钥验证每个所述附加的ECDSA签名,并在所述附加的ECDSA签名被验证的另一条件下返回所述TRUE结果。

[0479] 在实施例中,所述附加的ECDSA签名可以对不同于所述相应的第一ECDSA签名的消息进行签名。

[0480] 根据第二十可选实施例,可以提供一种根据所述第一至第十九实施例中任一所述的方法,其中:所述代码被配置为使得能够将较大的r部分值集合中的任何两个r部分值用作 r_1 和 r_2 。

[0481] 根据第二十一可选实施例,可以提供一种根据从属于所述第七至第十九实施例中任一所述的第二十实施例所述的方法,其中:所述集合包括至少三对r部分值;将不同的相应k值对分配给至少三个第二方中的每一个,从而使他们能够生成集合中所述对中的相应一个; r_1 是所述第二方中的一个的所述r部分值中的一个, r_2 是所述第二方中的另一个的所述r部分值中的一个;因此,所述代码使得所述至少三个第二方中的任何两个能够应对所述挑战。

[0482] 根据第二十二可选实施例,可以提供一种根据所述第一至第二十一实施例中任一所述的方法,其中:在所述同一第二方中接收所述信息。即,所述一笔或多笔第二交易仅为一笔第二交易。

[0483] 根据第二十三可选实施例,可以提供一种根据所述第二十二实施例所述的方法,其中:所述交易中的每笔交易包括数据结构,所述数据结构包括一个或多个输入和一个或多个输出,其中每个输出包括锁定脚本,每个输入包括解锁脚本和指向另一交易的输出的指针;所述代码由所述第一交易的所述锁定脚本组成,其中所述信息由所述第二交易的输入中的所述解锁脚本组成,并且其中所述第二交易的所述输入中的所述指针指向所述第一交易的所述输出;所述方法包括至少在所述代码返回所述TRUE结果的条件下核实所述交易

有效,并且响应于所述核实,包括以下各项中的至少一项:将所述第二交易包括在交易池中,以便由所述验证节点挖掘到一个或多个区块中;和/或将所述第二交易转发到所述区块链网络的至少一个其他节点。

[0484] 对于包括所述第一和第二交易的多笔交易中的每笔交易,所述网络的所述节点中的至少一些节点被配置为在所述交易有效的条件下传播每笔交易,并且所述网络的所述节点中的至少一些节点被配置为在所述交易有效的条件下将每笔交易记录在所述区块链的至少一部分的副本中。所述第二交易的有效性至少取决于返回所述TRUE结果的所述代码。

[0485] 根据第二十四可选实施例,可以提供一种根据所述第一至第二十二实施例中任一所述的方法,其中:所述交易根据基于账户的模型进行配置,所述代码由包括在所述第一交易中的智能合约组成。

[0486] 根据第二十五可选实施例,可以提供一种根据所述第二十四实施例所述的方法,其中:在多笔第二交易中接收所述信息。

[0487] 例如,所述多笔第二交易可以包括从每个所述第二方接收的相应交易。在每笔第二交易中接收的所述信息可以包括所述相应第二方的所述r部分 r_i 。在每笔第二交易中接收的所述信息可以包括所述相应第二方的所述公钥 P_i 。在每笔第二交易中接收的所述信息可以包括所述相应第二方的所述另一签名和/或所述附加的签名。

[0488] 在实施例中,在任一模型中,所述验证节点可以是挖掘节点、转发节点和/或存储所述区块链的至少一部分的存储节点(例如,存储所述区块链的完整副本的完整副本存储节点)。在实施例中,所述获取所述第一交易可以包括从第一方(例如,上述第一方)接收所述第一交易的至少一部分。在实施例中,所述获取所述第一交易可以包括从所述第一方接收所述第一交易。或者,在实施例中,所述获取所述第一交易可以包括在所述验证节点处形成所述第一交易。在实施例中,所述获取所述第一交易可以包括至少从所述第一方接收所述r部分的所述引用实例,并在所述节点中的所述一个节点处形成所述第一交易。在实施例中,所述获取所述第一交易可以包括形成所述第一交易,包括在所述验证节点处生成所述r部分。

[0489] 在实施例中,所述接收所述第二交易可以包括从第二方(例如,上述第二方)接收所述第二交易。在实施例中,所述第二交易至少部分由所述第二方生成。在实施例中,所述第二交易由所述第二方生成。在实施例中,所述接收所述第二交易可以包括直接或经由所述第一方或第三方从所述第二方接收所述第二交易。在实施例中,所述第二交易由第三方至少基于由所述第二方提供给所述第三方的所述第一ECDSA签名的所述s部分(以及在实施例中所述第一ECDSA签名的所述r部分的所述提交实例和/或所述数据元素)生成。

[0490] 根据本文中所公开的教导的第二十六可选示例,可以提供一种计算机程序,所述计算机程序包含在计算机可读存储器上,并且被配置为当在网络的节点上运行时,执行根据所述第一至第二十五实施例中任一所述的方法。

[0491] 根据本文中所公开的教导的第二十七实施例,提供了一种网络的节点,所述节点包括:存储器,所述存储器包括一个或多个存储单元;处置装置,所述处置装置包括一个或多个处理单元;其中所述存储器存储被设置在所述处理装置上运行的代码,所述代码被配置为当在所述处理装置上时,执行根据所述第一至第二十五实施例中任一所述的方法。

[0492] 根据在本文中所公开的教导的第二十八实施例,提供了用于记录在区块链中的第

一交易,所述第一交易包含在一个或多个计算机可读介质上,并且包括可运行代码。所述代码包括用于基于一笔或多笔第二交易来评估基于联合r值 r_{joint} 定义的挑战的参考数据,所述一笔或多笔第二交易包含至少包括一对第一ECDSA签名 $i=1,2$ 中的每个相应签名的相应r部分 r_i 和s部分 s_i 的信息,其中所述一对第一ECDSA签名中的每个签名基于对应于相应第一公钥 P_i 的相应第一私钥 V_i 对所述一笔或多笔第二交易中的一笔第二交易的一部分进行签名。所述代码被配置为基于所述第一交易中的所述参考数据和在所述一笔或多笔第二交易中接收的所述r部分 r_i 来验证所述挑战是否得以应对,并且在所述挑战得以应对的条件下返回TRUE结果,所述挑战包括以下条件:

$$[0493] \quad r_1 + r_2 = \lambda^2 \cdot r_{joint} \bmod p,$$

[0494] 其中 $r_1 + r_2$ 表示标量加法, $r_{joint} = [R_{joint}]_x$, $R_{joint} = R_1 + R_2$ (通过椭圆点加法), p 是素数模, $\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod p$, $R_i = k_i \cdot G$, $x_i = [R_i]_x$, $y_i = [R_i]_y$, k_i 是临时密钥, G 是椭圆曲线生成点, $[]_x$ 表示 $[...]$ 的x坐标, $[]_y$ 表示 $[...]$ 的y坐标, “ $\cdot$ ”表示椭圆曲线标量乘法。

[0495] 在实施例中,所述第一交易可以根据在本文中所公开的任何实施例或其他特征进行配置。

[0496] 一旦给出本文的公开内容,所公开技术的其他变体或用例对于本领域技术人员可能变得显而易见。本公开的范围不受所描述的实施例限制,而仅受随附权利要求限制。

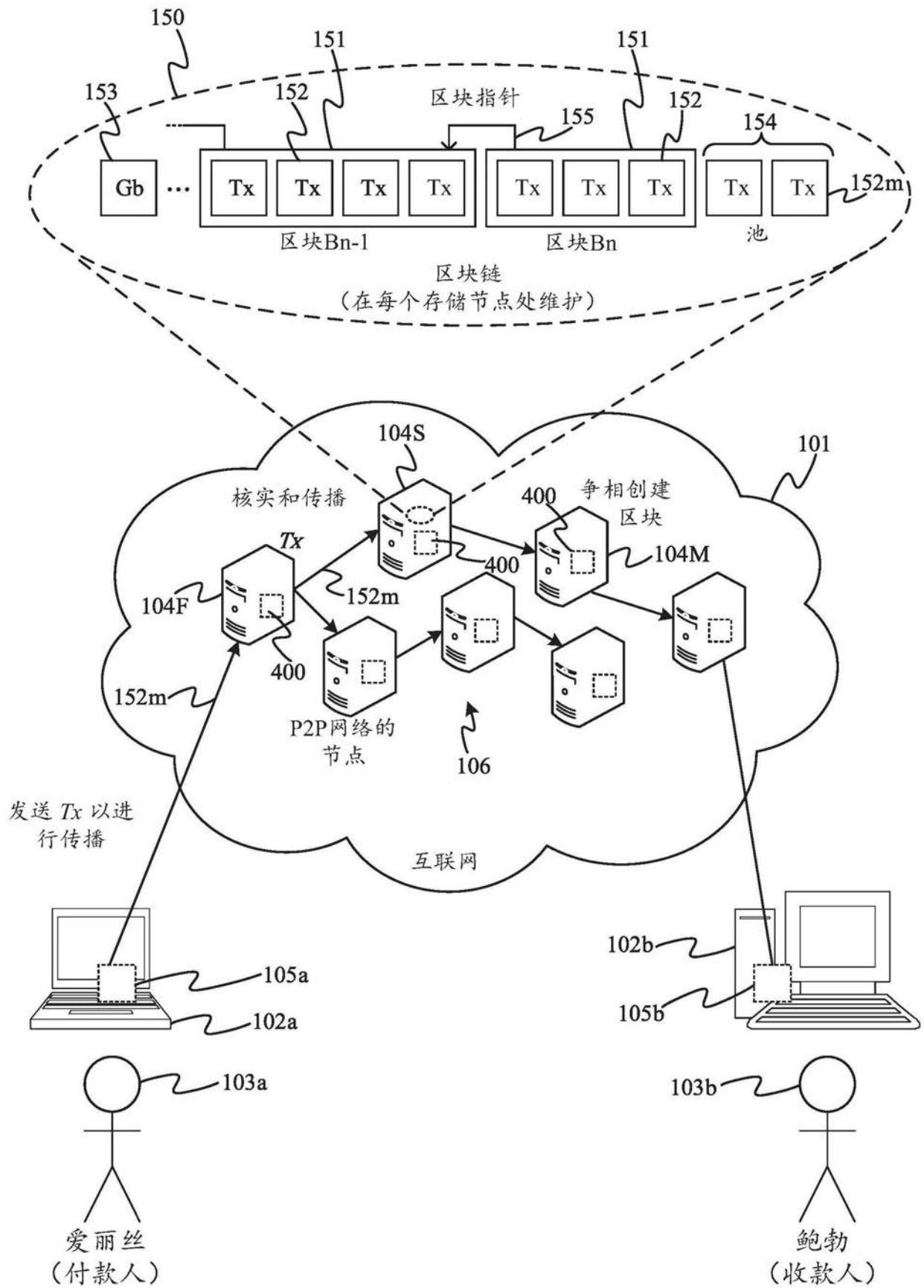
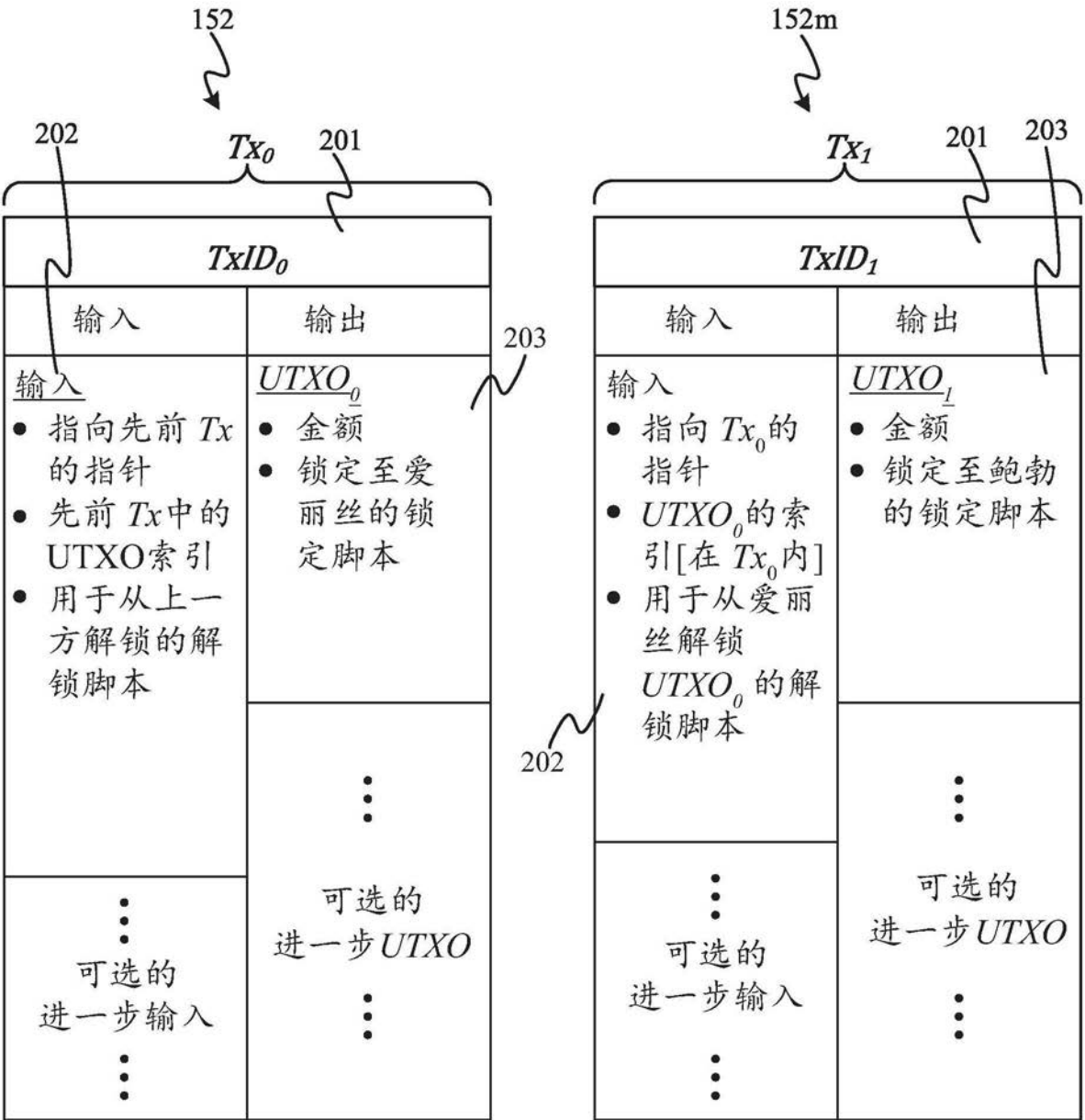


图1



从爱丽丝到鲍勃的交易

通过运行下述内容进行核实：爱丽丝的锁定脚本（来自 Tx_0 的输出）和爱丽丝的解锁脚本（作为 Tx_1 的输入）。这可检查 Tx_1 是否满足爱丽丝的锁定脚本中定义的条件。

图2

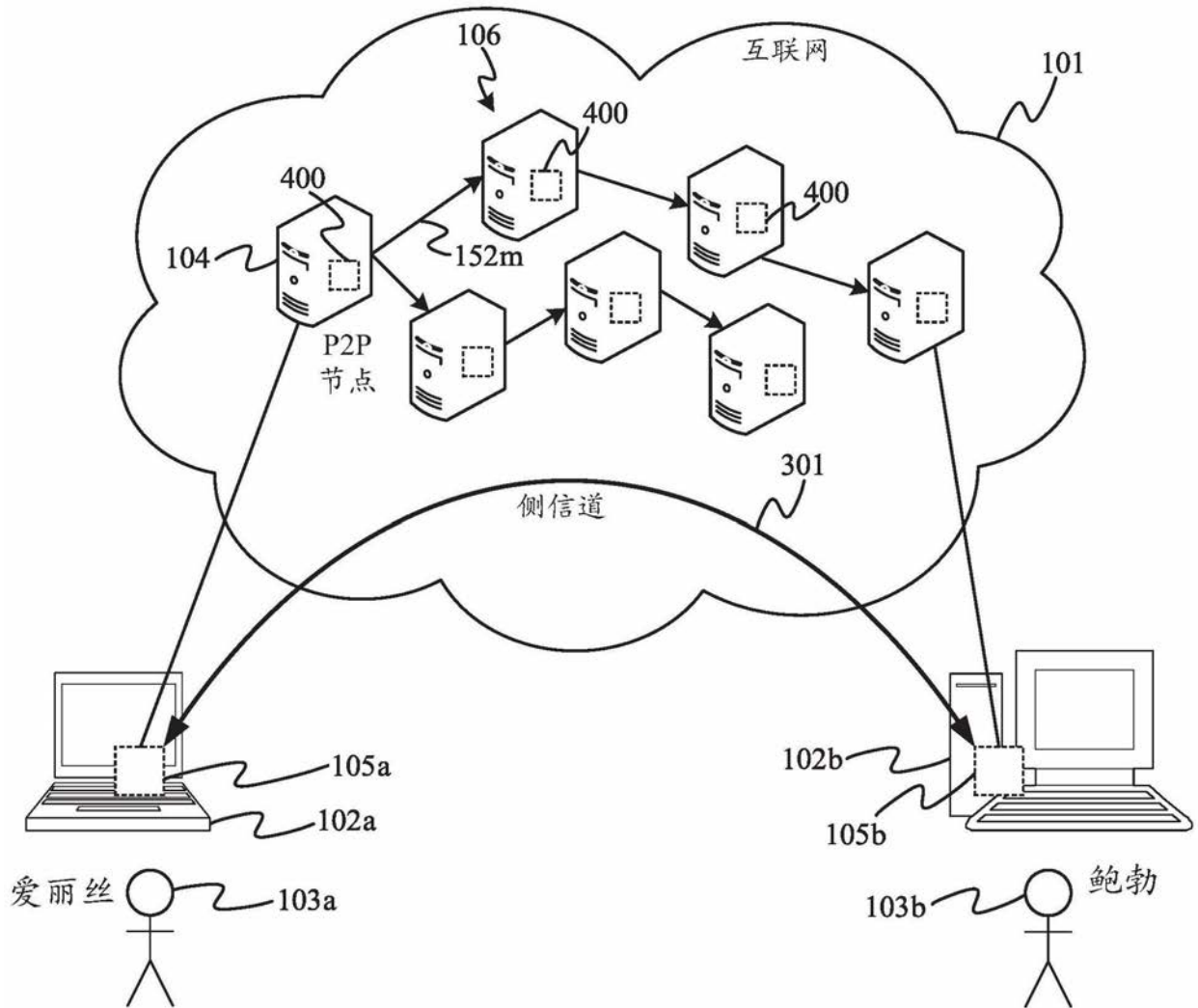


图3

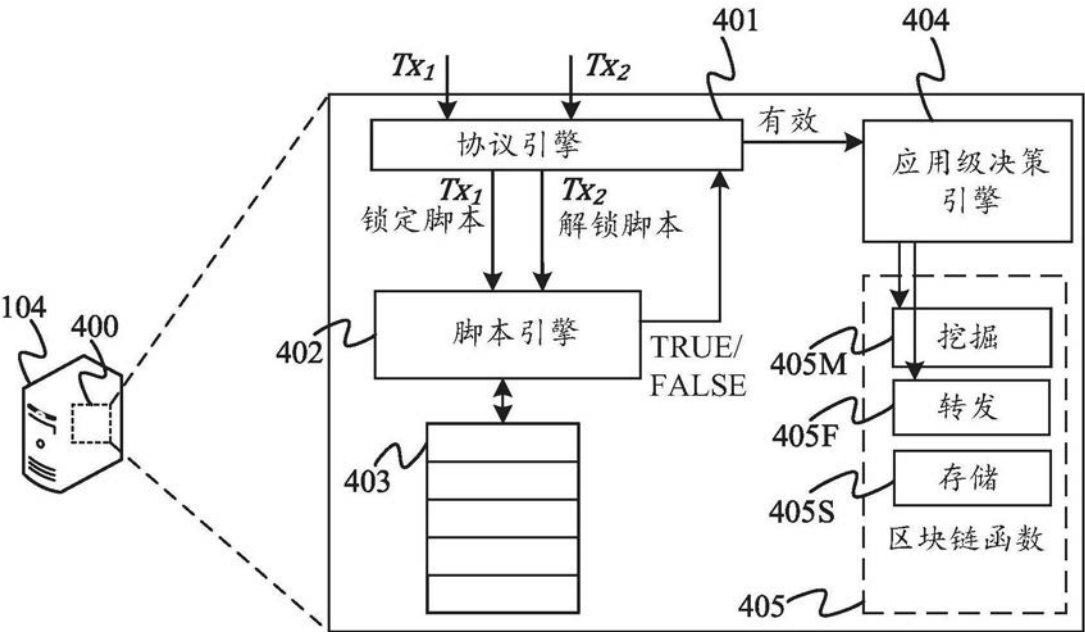


图4

第零交易（源交易）

第一交易（挑战交易）

第二交易（证明交易）

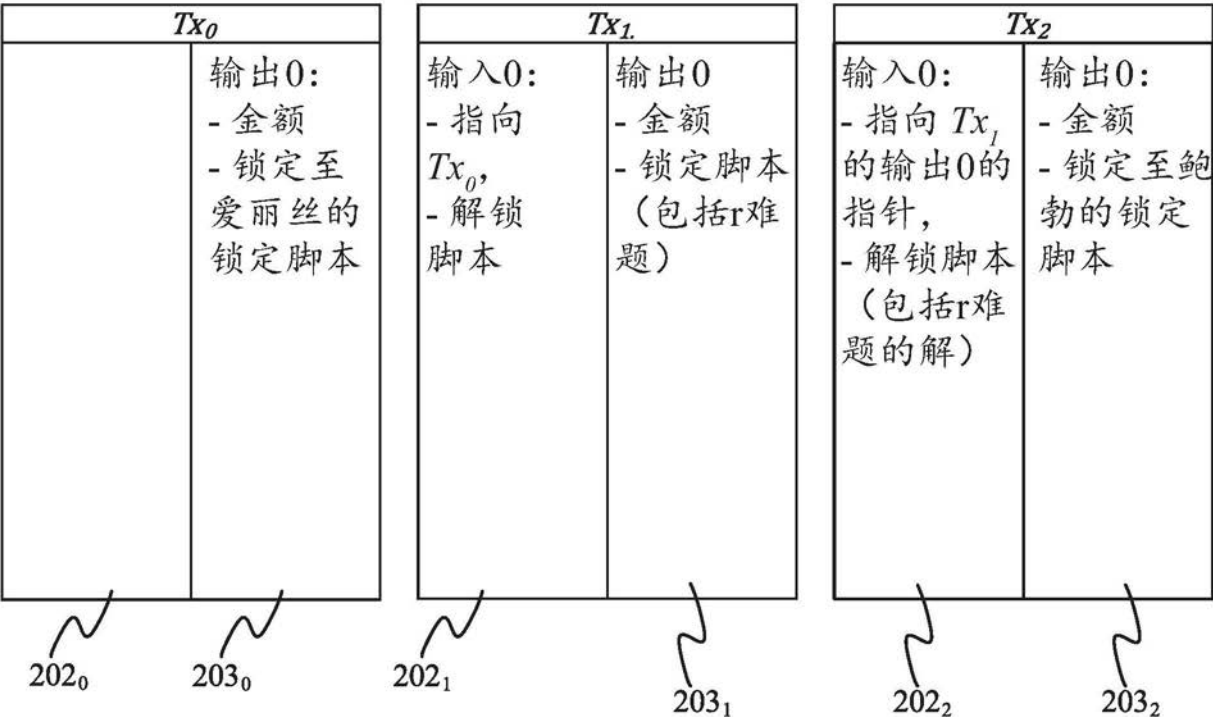


图5

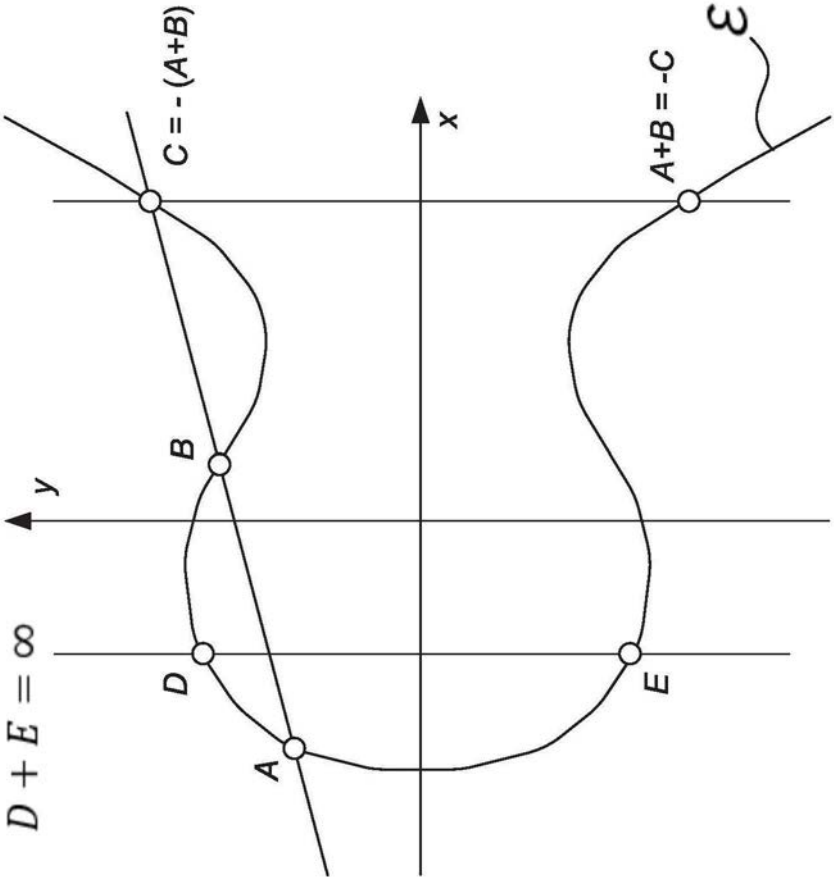


图6A

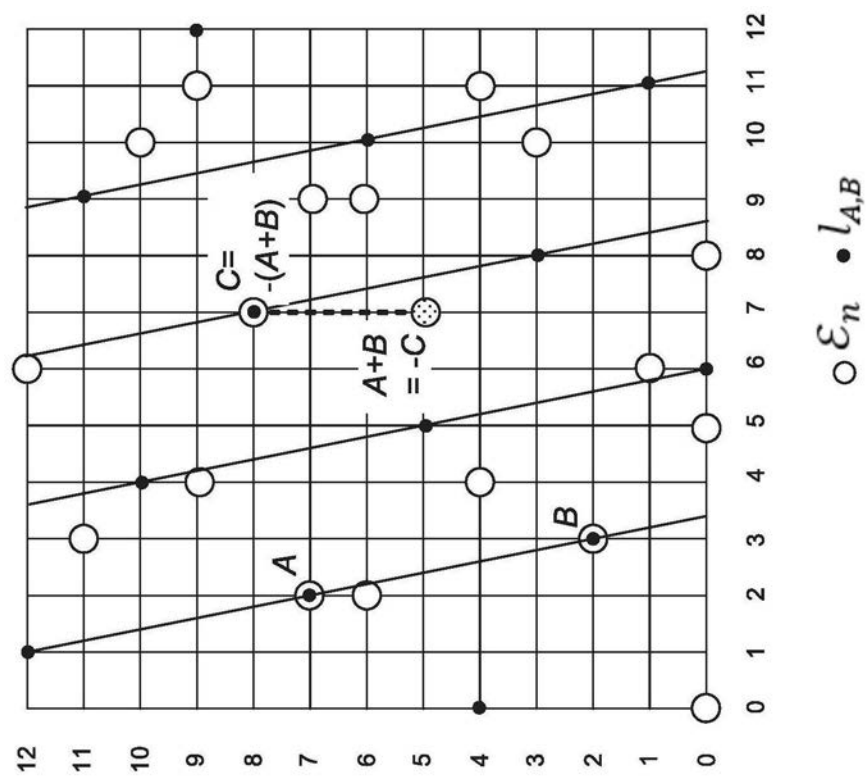


图6B

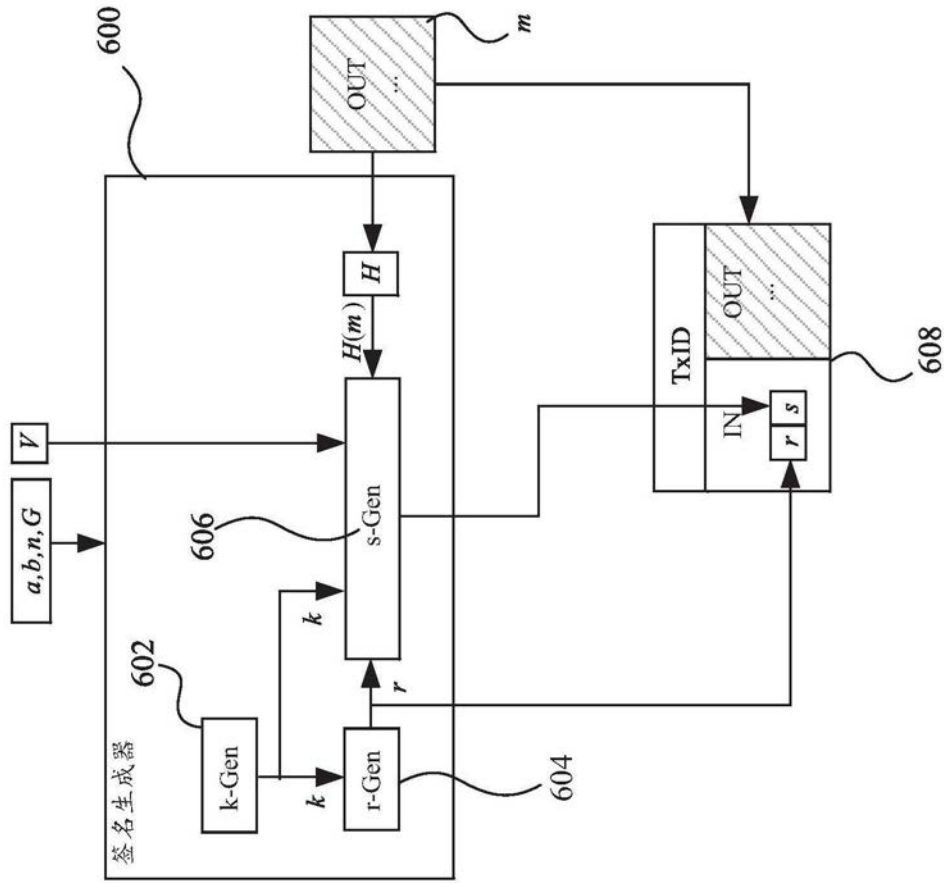


图6C

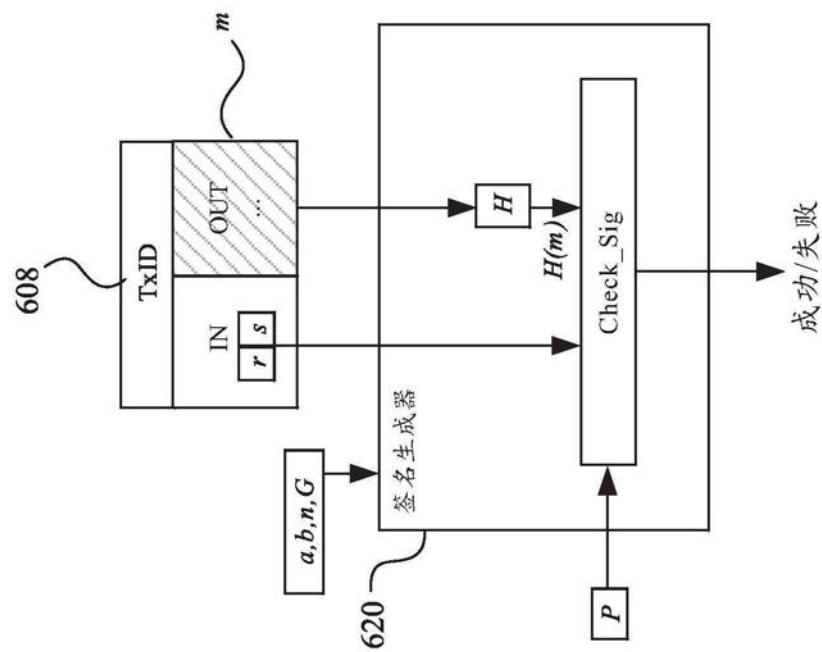


图6D

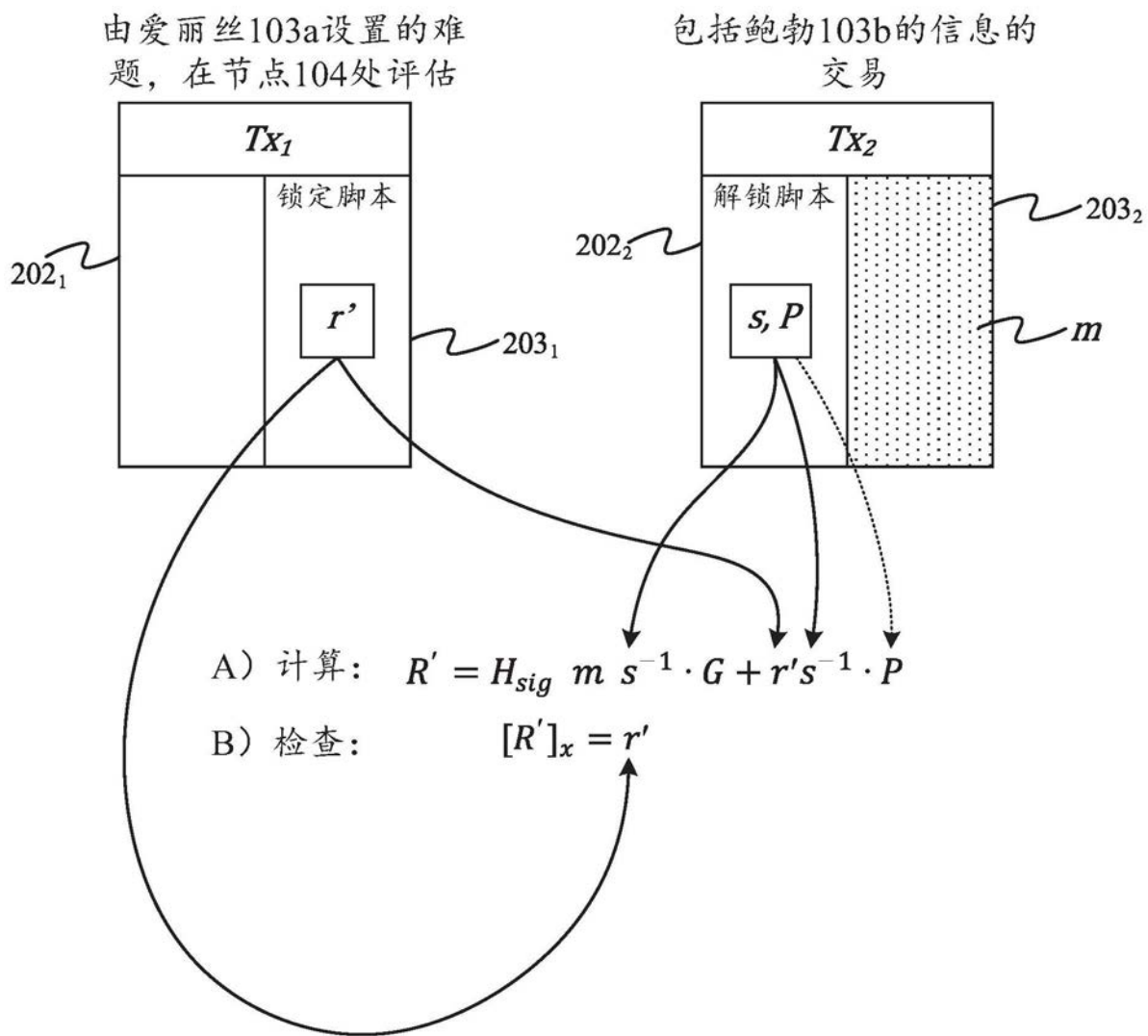


图7

由爱丽丝103a设置的难题，在节点104处评估

包括鲍勃103b的信息的交易

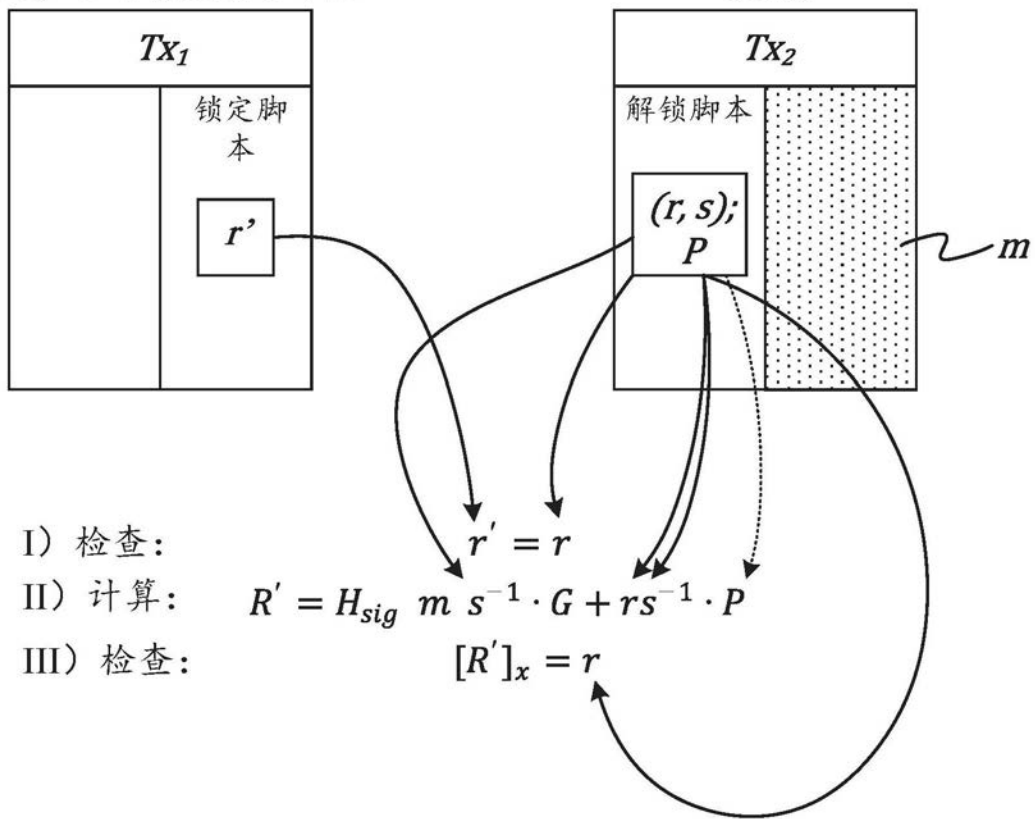


图8

由爱丽丝103a设置的难题，在节点104处评估

包括鲍勃103b的信息的交易

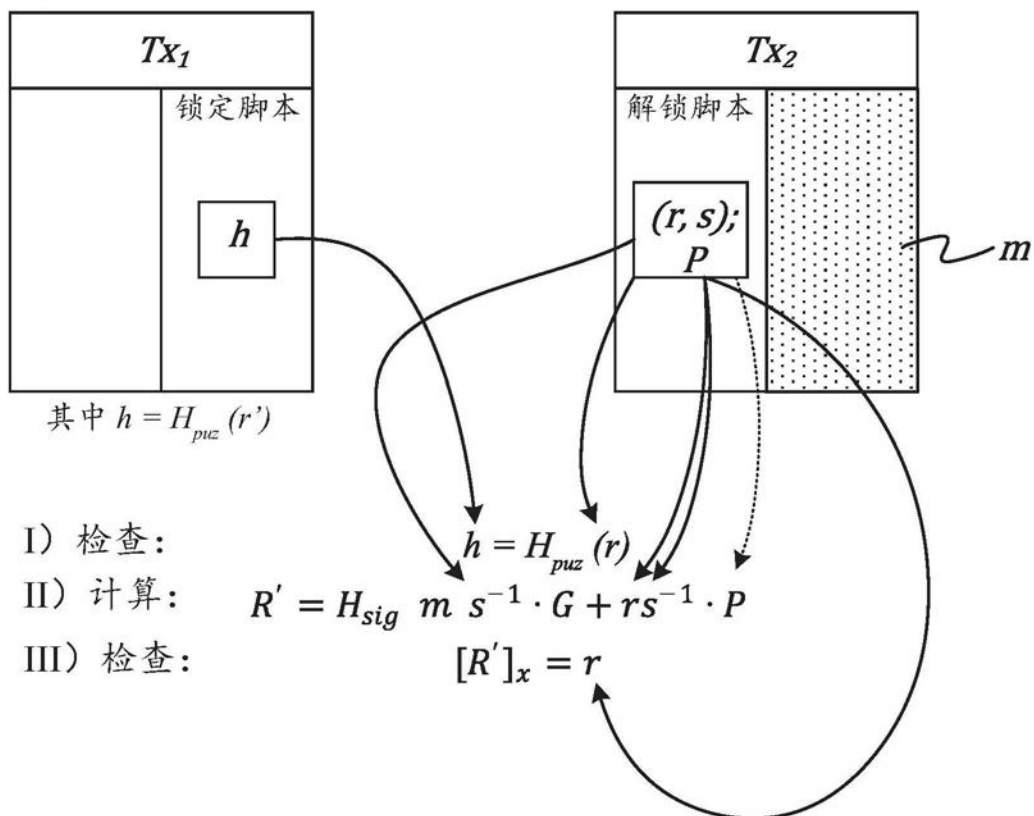


图9

由爱丽丝设置的难题，
在节点104处评估

包括鲍勃和查理的信息
的交易

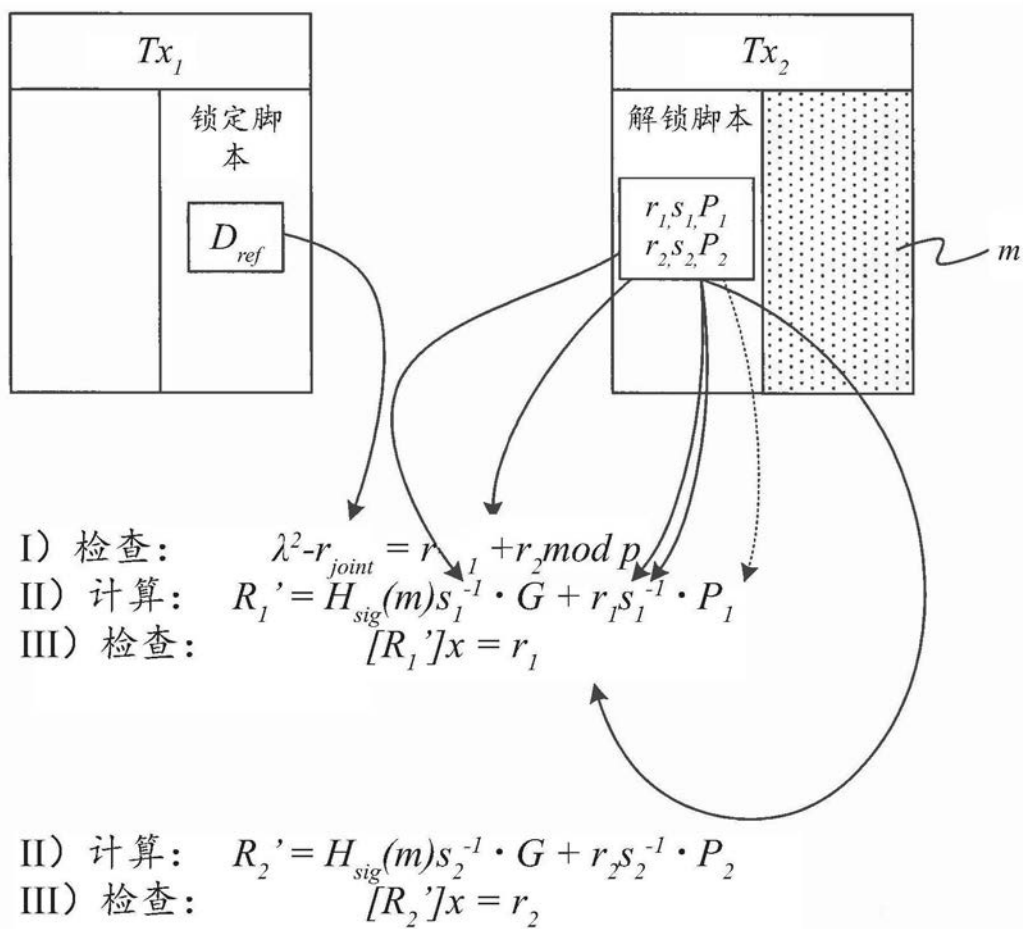


图10

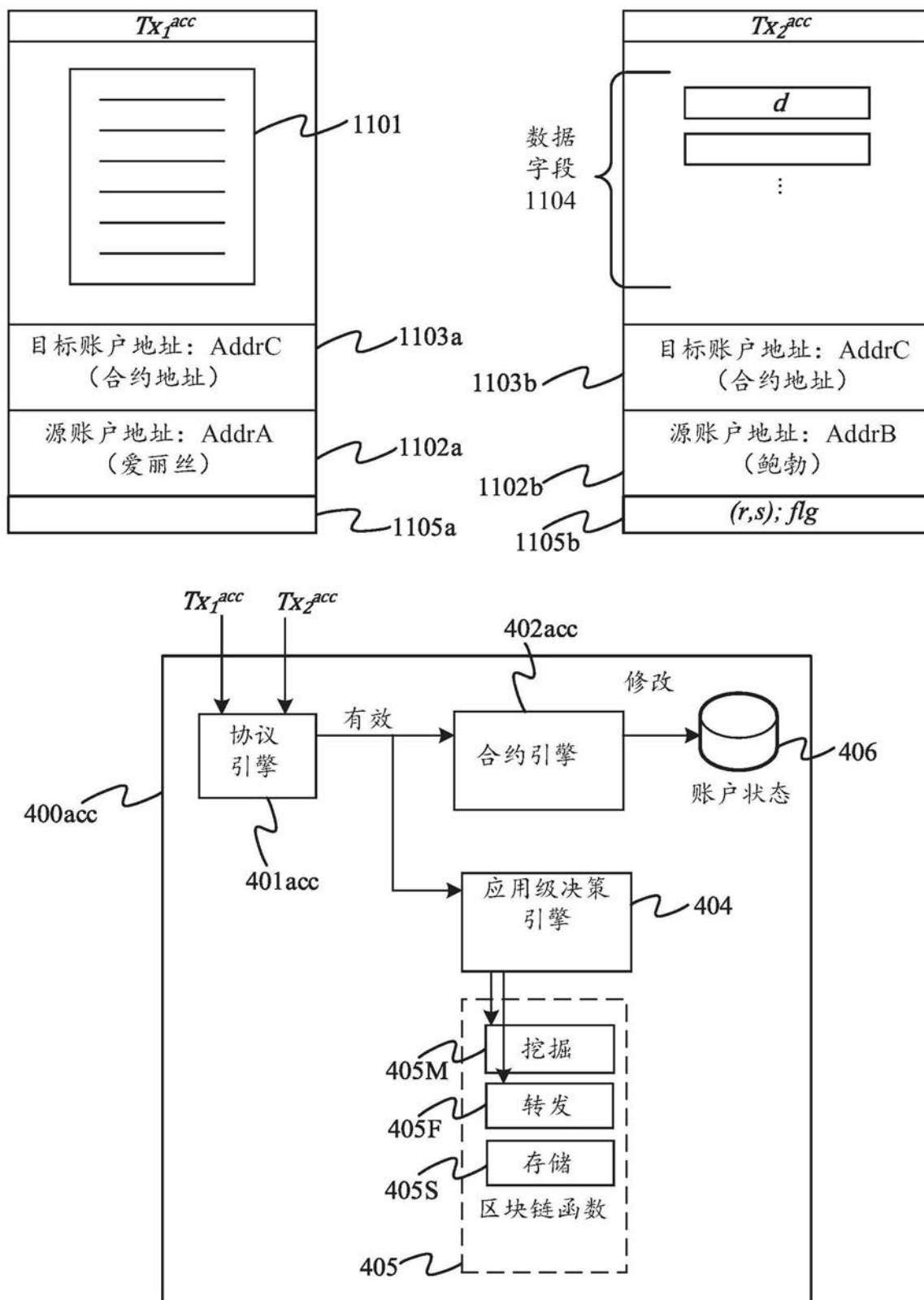


图11



图12

堆栈	脚本	描述
空	$\langle P_g \rangle \langle sig_r \rangle$ OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle t \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	scriptSig和scriptPubKey结合使用
$\langle P_g \rangle \langle sig_r \rangle$	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle t \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	解锁脚本常数推送至堆栈
$\langle P_g \rangle \langle sig_r \rangle \langle sig_r \rangle \langle OP_3 \rangle$	OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle t \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS (sig) 已复制, 且常数 (3) 推送至堆栈
$\langle P_g \rangle \langle sig_r \rangle \langle sig_r \rangle [3:] \langle sig_r [3:] \rangle$	OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle t \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS (sig) 在第三字节处拆分, 恰好在len(t)之前
$\langle P_g \rangle \langle sig_r \rangle \langle sig_r [3:] \rangle \langle OP_1 \rangle$	OP_SPLIT OP_SWAP OP_SPLIT OP_DROP $\langle t \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS项目的第二部分已删除, 且常数 (1) 推送至堆栈
$\langle P_g \rangle \langle sig_r \rangle \langle sig_r [4:] \rangle \langle sig_r [3:4] \rangle$	OP_SPLIT OP_DROP $\langle t \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS拆分1字节 (在sig的字节4处), 并交换堆栈上的前两个项目
$\langle P_g \rangle \langle sig_r \rangle \langle sig_r [4:36/37] \rangle \langle sig_r [36/37:] \rangle$	OP_DROP $\langle t \rangle$ OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS的第二部分 (从索引4开始的sig) 在len(t)= <sig [3: 4]> 字节处拆分, 其中将为36或37, 因为t将为32或33字节
$\langle P_g \rangle \langle sig_r \rangle \langle sig_r [4:36/37] \rangle \langle t \rangle$	OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	拖动ToS (sig的其余部分) 并推送至 $\langle t \rangle$ 堆栈
$\langle sig_r \rangle \langle P_g \rangle$	OP_CHECKSIG	检查 $\langle sig_r [4: 36/37] \rangle = \langle t \rangle$
TRUE	空	检查前两个堆栈项目的签名

图13