(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 108564692 A
(43)申请公布日 2018.09.21

(21)申请号 201810316226.0

(22)申请日 2018.04.10

(71)申请人 周伟
地址 100027 北京市东城区东直门外大街42号宇飞大厦5层无界空间

(72)发明人 周伟

(74)专利代理机构 北京聿华联合知识产权代理有限公司 11611
代理人 朱绘 张文娟

(51)Int.Cl.
G07C 9/00(2006.01)

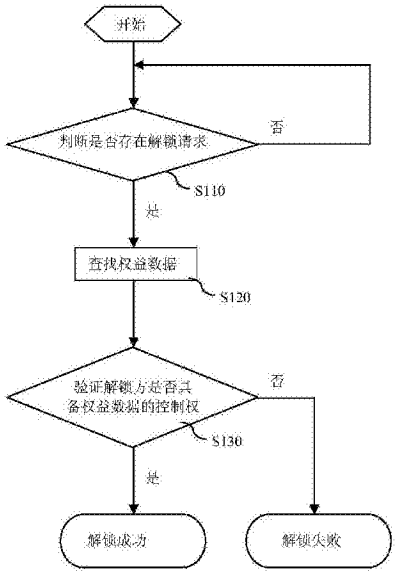
权利要求书2页 说明书11页 附图4页

(54)发明名称

一种基于区块链的解锁验证方法以及解锁系统

(57)摘要

本发明公开了一种基于区块链的解锁验证方法以及解锁系统。所述方法包括：判断当前是否接收到来自解锁方的解锁请求，当存在解锁请求时在区块链上查找包含锁的身份标识的权益数据，其中，所述权益数据是一种寄生在区块链内的数据结构，具备将数据和所有权绑定并进行所有权变更的能力；验证所述解锁方是否具有所述权益数据的控制权以确认所述解锁方是否有所述锁的控制权。根据本发明的方法，可以通过数据形式的钥匙进行解锁操作；相较于现有技术，本发明的解锁验证方法操作简单，具备更高的安全性。



CN 108564692 A

1.一种基于区块链的解锁验证方法，其特征在于，所述方法包括：

判断当前是否接收到来自解锁方的解锁请求，当存在解锁请求时在区块链上查找包含锁的身份标识的权益数据，其中，所述权益数据是一种寄生在区块链内的数据结构，具备将数据和所有权绑定并进行所有权变更的能力；

验证所述解锁方是否具有所述权益数据的控制权以确认所述解锁方是否有所述锁的控制权。

2.根据权利要求1所述的方法，其特征在于，验证所述解锁方是否具有所述权益数据的控制权以确认所述解锁方是否有所述锁的控制权，包括：

从所述权益数据中取出控制者的身份标识；

接收来自所述解锁方的私钥签名，根据所述身份标识验证所述私钥签名的正确性；

当所述私钥签名正确时根据所述私钥签名以及所述身份标识判断所述解锁方是否具有所述权益数据的控制权。

3.根据权利要求2所述的方法，其特征在于，接收来自所述解锁方的私钥签名，包括：

向所述解锁方发送钥匙要求；

接收来自所述解锁方的钥匙数据，其中，所述钥匙数据是所述解锁方根据掌握的私钥对所述钥匙请求签名而生成的；

从所述钥匙数据中提取所述私钥签名。

4.根据权利要求2或3所述的方法，其特征在于，当所述私钥签名正确时根据所述私钥签名以及所述身份标识判断所述解锁方是否具有所述权益数据的控制权，包括：

根据当前接收到的一个或多个正确的私钥签名判断当前的一个或多个解锁方是否具有所述权益数据的控制权。

5.根据权利要求2或3所述的方法，其特征在于，其特征在于：

根据当前接收到的一个或多个正确的私钥签名以及之前接收到的、仍保持有效状态的一个或多个私钥签名判断当前的一个或多个解锁方是否具有所述权益数据的控制权。

6.根据权利要求5所述的方法，其特征在于，针对所述权益数据具有第一私钥签名、第二私钥签名以及第三私钥签名三把正确的私钥签名，其中：

所述第一私钥签名、所述第二私钥签名以及所述第三私钥签名被接收并验证正确后即进入钥匙有效状态，任意两把私钥签名同时有效时即可判定当前的一个或多个解锁方具有所述权益数据的控制权；

所述第一私钥签名被接收后，在所述第一私钥签名对应的解锁方解除私钥签名状态之前，维持所述第一私钥签名的有效状态；

所述第二私钥签名以及所述第三私钥签名被接收后，在进行一次控制权判定后解除所述第二私钥签名以及所述第三私钥签名的有效状态。

7.一种基于区块链的解锁系统，其特征在于，所述系统包括：

钥匙接口，其配置为判断当前是否接收到来自解锁方的解锁请求；

验证数据提取模块，其配置为当存在解锁请求时在区块链上查找包含锁的身份标识的权益数据，其中，所述权益数据是一种寄生在区块链内的数据结构，具备将数据和所有权绑定并进行所有权变更的能力；

验证模块，其配置为验证所述解锁方是否具有所述权益数据的控制权以确认所述解锁

方是否有所述锁的控制权；

控制输出模块，其配置为当所述解锁方具有所述锁的控制权时向所述锁输出解锁指令。

8.根据权利要求7所述的系统，其特征在于：

所述验证数据提取模块还配置为从所述权益数据中取出控制者的身份标识；

所述钥匙接口还配置为接收来自所述解锁方的私钥签名；

所述验证模块还配置为根据所述身份标识验证所述私钥签名的正确性，当所述私钥签名正确时根据所述私钥签名以及所述身份标识判断所述解锁方是否具有所述权益数据的控制权。

9.根据权利要求8所述的系统，其特征在于，所述钥匙接口还配置为：

向所述解锁方发送钥匙要求；

接收来自所述解锁方的钥匙数据，其中，所述钥匙数据是所述解锁方根据掌握的私钥对所述钥匙请求签名而生成的；

从所述钥匙数据中提取所述私钥签名。

10.根据权利要求8或9所述的系统，其特征在于：

所述系统还包括钥匙寄存器，所述钥匙寄存器配置为存储之前接收到的、可以保持有效状态的一个或多个私钥签名；

所述验证模块配置为根据当前接收到的一个或多个正确的私钥签名以及之前接收到的、仍保持有效状态的一个或多个私钥签名判断当前的一个或多个解锁方是否具有所述权益数据的控制权。

一种基于区块链的解锁验证方法以及解锁系统

技术领域

[0001] 本发明涉及计算机领域，具体涉及一种基于区块链的解锁验证方法以及解锁系统。

背景技术

[0002] 锁是人类日常生产生活中一种常见的装置。通常，每把锁拥有与其物理结构对应的钥匙，没有钥匙就无法开锁。但是，由于钥匙需要随身携带，钥匙丢失的情况时有发生，造成了很大不便。并且，为了避免锁被非法打开，锁与钥匙间的匹配方式愈加复杂，钥匙的形状也变得越来越复杂。这不仅造成了锁和钥匙的制造成本的不断攀升，而且由于钥匙的形状复杂度过高，导致某些类型的钥匙在丢失后很难重新制造补充，只能连同锁一起整体更换，这更加提高了锁的使用成本。

[0003] 为了在提高锁与钥匙间的匹配复杂程度的基础上控制锁匙成本，在现有技术中提出了非物理匹配的锁匙结构，例如采用密码输入或是磁卡方式的电子锁。但是，由于磁卡仍然存在被破解、复制的可能，而密码也存在被遗忘、破解的可能性，因此，现有技术中的电子锁在使用上仍有很多不便。

发明内容

[0004] 本发明提供了一种基于区块链的解锁验证方法，所述方法包括：

[0005] 判断当前是否接收到来自解锁方的解锁请求，当存在解锁请求时在区块链上查找包含锁的身份标识的权益数据，其中，所述权益数据是一种寄生在区块链内的数据结构，具备将数据和所有权绑定并进行所有权变更的能力；

[0006] 验证所述解锁方是否具有所述权益数据的控制权以确认所述解锁方是否有所述锁的控制权。

[0007] 在一实施例中，验证所述解锁方是否具有所述权益数据的控制权以确认所述解锁方是否有所述锁的控制权，包括：

[0008] 从所述权益数据中取出控制者的身份标识；

[0009] 接收来自所述解锁方的私钥签名，根据所述身份标识验证所述私钥签名的正确性；

[0010] 当所述私钥签名正确时根据所述私钥签名以及所述身份标识判断所述解锁方是否具有所述权益数据的控制权。

[0011] 在一实施例中，接收来自所述解锁方的私钥签名，包括：

[0012] 向所述解锁方发送钥匙要求；

[0013] 接收来自所述解锁方的钥匙数据，其中，所述钥匙数据是所述解锁方根据掌握的私钥对所述钥匙请求签名而生成的；

[0014] 从所述钥匙数据中提取所述私钥签名。

[0015] 在一实施例中，当所述私钥签名正确时根据所述私钥签名以及所述身份标识判断

所述解锁方是否具有所述权益数据的控制权，包括：

[0016] 根据当前接收到的一个或多个正确的私钥签名判断当前的一个或多个解锁方是否具有所述权益数据的控制权。

[0017] 在一实施例中，其特征在于：

[0018] 根据当前接收到的一个或多个正确的私钥签名以及之前接收到的、仍保持有效状态的一个或多个私钥签名判断当前的一个或多个解锁方是否具有所述权益数据的控制权。

[0019] 在一实施例中，针对所述权益数据具有第一私钥签名、第二私钥签名以及第三私钥签名三把正确的私钥签名，其中：

[0020] 所述第一私钥签名、所述第二私钥签名以及所述第三私钥签名被接收并验证正确后即进入钥匙有效状态，任意两把私钥签名同时有效时即可判定当前的一个或多个解锁方具有所述权益数据的控制权；

[0021] 所述第一私钥签名被接收后，在所述第一私钥签名对应的解锁方解除私钥签名状态之前，维持所述第一私钥签名的有效状态；

[0022] 所述第二私钥签名以及所述第三私钥签名被接收后，在进行一次控制权判定后解除所述第二私钥签名以及所述第三私钥签名的有效状态。

[0023] 本发明还提出了一种基于区块链的解锁系统，所述系统包括：

[0024] 钥匙接口，其配置为判断当前是否接收到来自解锁方的解锁请求；

[0025] 验证数据提取模块，其配置为当存在解锁请求时在区块链上查找包含锁的身份标识的权益数据，其中，所述权益数据是一种寄生在区块链内的数据结构，具备将数据和所有权绑定并进行所有权变更的能力；

[0026] 验证模块，其配置为验证所述解锁方是否具有所述权益数据的控制权以确认所述解锁方是否有所述锁的控制权；

[0027] 控制输出模块，其配置为当所述解锁方具有所述锁的控制权时向所述锁输出解锁指令。

[0028] 在一实施例中：

[0029] 所述验证数据提取模块还配置为从所述权益数据中取出控制者的身份标识；

[0030] 所述钥匙接口还配置为接收来自所述解锁方的私钥签名；

[0031] 所述验证模块还配置为根据所述身份标识验证所述私钥签名的正确性，当所述私钥签名正确时根据所述私钥签名以及所述身份标识判断所述解锁方是否具有所述权益数据的控制权。

[0032] 在一实施例中，所述钥匙接口还配置为：

[0033] 向所述解锁方发送钥匙要求；

[0034] 接收来自所述解锁方的钥匙数据，其中，所述钥匙数据是所述解锁方根据掌握的私钥对所述钥匙请求签名而生成的；

[0035] 从所述钥匙数据中提取所述私钥签名。

[0036] 在一实施例中：

[0037] 所述系统还包括钥匙寄存器，所述钥匙寄存器配置为存储之前接收到的、可以保持有效状态的一个或多个私钥签名；

[0038] 所述验证模块配置为根据当前接收到的一个或多个正确的私钥签名以及之前接

收到的、仍保持有效状态的一个或多个私钥签名判断当前的一个或多个解锁方是否具有所述权益数据的控制权。

[0039] 根据本发明的方法，可以通过数据形式的钥匙进行解锁操作；相较于现有技术，本发明的解锁验证方法操作简单，具备更高的安全性。

[0040] 本发明的其它特征或优点将在随后的说明书中阐述。并且，本发明的部分特征或优点将通过说明书而变得显而易见，或者通过实施本发明而被了解。本发明的目的和部分优点可通过在说明书、权利要求书以及附图中所特别指出的步骤来实现或获得。

附图说明

[0041] 附图用来提供对本发明的进一步理解，并且构成说明书的一部分，与本发明的实施例共同用于解释本发明，并不构成对本发明的限制。在附图中：

[0042] 图1是根据本发明一实施例的方法流程图；

[0043] 图2以及图3是根据本发明实施例的方法的部分流程图；

[0044] 图4以及图5是根据本发明不同实施例的系统结构简图。

具体实施方式

[0045] 以下将结合附图及实施例来详细说明本发明的实施方式，借此本发明的实施人员可以充分理解本发明如何应用技术手段来解决技术问题，并达成技术效果的实现过程并依据上述实现过程具体实施本发明。需要说明的是，只要不构成冲突，本发明中的各个实施例以及各实施例中的各个特征可以相互结合，所形成的技术方案均在本发明的保护范围之内。

[0046] 锁是人类日常生产生活中一种常见的装置。通常，每把锁拥有与其物理结构对应的钥匙，没有钥匙就无法开锁。但是，由于钥匙需要随身携带，钥匙丢失的情况时有发生，造成了很大不便。并且，为了避免锁被非法打开，锁与钥匙间的匹配方式愈加复杂，钥匙的形状也变得越来越复杂。这不仅造成了锁和钥匙的制造成本的不断攀升，而且由于钥匙的形状复杂度过高，导致某些类型的钥匙在丢失后很难重新制造补充，只能连同锁一起整体更换，这更加提高了锁的使用成本。

[0047] 为了在提高锁与钥匙间的匹配复杂程度的基础上控制锁匙成本，在现有技术中提出了非物理匹配的锁匙结构，例如采用密码输入或是磁卡方式的电子锁。但是，由于磁卡仍然存在被破解、复制的可能，而密码也存在被遗忘、破解的可能性，因此，现有技术中的电子锁在使用上仍有很多不便。

[0048] 针对上述问题，本发明提出了一种用于电子锁的解锁验证方法。区块链是分布式数据存储、点对点传输、共识机制、加密算法等计算机技术的新型应用模式。所谓共识机制是区块链系统中实现不同节点之间建立信任、获取权益的数学算法。狭义来讲，区块链是一种按照时间顺序将数据区块以顺序相连的方式组合成的一种链式数据结构，并以密码学方式保证的不可篡改和不可伪造的分布式账本。广义来讲，区块链技术是利用块链式数据结构来验证与存储数据、利用分布式节点共识算法来生成和更新数据、利用密码学的方式保证数据传输和访问的安全、利用由自动化脚本代码组成的智能合约来编程和操作数据的一种全新的分布式基础架构与计算范式。

[0049] 具体的，在本发明的方法中，将电子锁的钥匙保存在区块链上。由于区块链是基于网络保存的，因此不存在丢失的可能。并且，基于区块链的难破解程度，锁的安全性也被大大提高。根据本发明的方法，可以通过数据形式的钥匙进行解锁操作；相较于现有技术，本发明的解锁验证方法操作简单，具备更高的安全性。

[0050] 接下来基于附图详细描述根据本发明实施例的方法的详细流程，附图的流程图中示出的步骤可以在包含诸如一组计算机可执行指令的计算机系统中执行。虽然在流程图中示出了各步骤的逻辑顺序，但是在某些情况下，可以以不同于此处的顺序执行所示出或描述的步骤。

[0051] 如图1所示，在一实施例中，本发明的方法包括以下流程：

[0052] 判断当前是否接收到来自解锁方的解锁请求(S110)；

[0053] 当存在解锁请求时在区块链上查找包含锁的身份标识的权益数据(S120)；

[0054] 验证解锁方是否具有权益数据的控制权以确认所述解锁方是否有所述锁的控制权(S130)。

[0055] 权益数据是一种寄生在区块链内的数据结构，具备将数据和所有权绑定并进行所有权变更的能力。具体的，在一实施例中，权益数据指的是“未花费的交易输出”(Unspent Transaction Outputs，UTXO)。通过使用具有UTXO特性的公链作为载体，以承载锁的身份标识以及其他解锁相关信息。这里需要指出的是，优选的，作为载体的权益数据可以是当前流通中的得到普遍认可的具有UTXO特性的公链。

[0056] 进一步的，如图2所示，在一实施例中，验证解锁方是否具有权益数据的控制权包括以下步骤：

[0057] 从权益数据中取出控制者的身份标识(S210)；

[0058] 接收解锁方的私钥签名(S220)；

[0059] 根据控制者的身份标识验证解锁方的私钥签名的正确性(S230)；

[0060] 当私钥签名正确时根据私钥签名以及控制者的身份标识判断解锁方是否具有权益数据的控制权(S240)。

[0061] 具体的，在一实施例中，控制者的身份标识为控制者的公钥。在另一实施例中，控制者的身份标识为控制者的地址。

[0062] 进一步的，为了防止解锁方用于解锁的私钥签名被盗取，如图3所示，在一实施例中，接收来自解锁方的私钥签名包括以下步骤：

[0063] 向解锁方发送钥匙要求(S310)；

[0064] 接收来自解锁方的钥匙数据(S320)，其中，该钥匙数据是解锁方根据掌握的私钥对钥匙请求签名而生成的；

[0065] 从接收到的钥匙数据中提取私钥签名(S330)。

[0066] 进一步的，在一实施例中，钥匙要求包含一段随机数，解锁方对接收到的随机数签名生成钥匙数据。由于每次开锁向解锁方输出的随机数都不同，因此就避免了钥匙数据被盗用的情况。

[0067] 具体的，在一实施例中，锁在区块链上查找包含有自身ID标识的那个UTXO，通过验证UTXO的控制权以确认是否有锁的控制权(该UTXO的控制权即锁的控制权)。具体过程是：取出控制者的公钥(或地址)，然后向试图开锁者发送一个要求，要求其对自己生成的一个

随机数用私钥签名，并发回给自己，如果签名正确，那么说明对方持有正确的钥匙，即验证通过，可以开锁。

[0068] 进一步的，考虑到同一锁具有多个可开锁者的情况，在一实施例中，根据私钥签名以及控制方的身份标识判断解锁方是否具有权益数据的控制权时采用多重签名机制。对应一把锁，具有多个正确的私钥签名，其中任意一个正确的私钥签名通过验证后即可具有权益数据的控制权。

[0069] 进一步的，考虑到多个解锁者合作开锁的情况，在一实施例中，根据私钥签名以及控制方的身份标识判断解锁方是否具有权益数据的控制权时采用多重签名机制，具体的：

[0070] 根据当前接收到的一个或多个正确的私钥签名判断当前的一个或多个解锁方是否具有权益数据的控制权。

[0071] 例如，2/3签名，表示一共存在3个私钥签名正确的解锁方，其中，有任意2个同时通过验证，即可断定其具有权益数据的控制权，可以开锁。

[0072] 进一步的，考虑到不同的解锁者具备不同的开锁权限，例如，在某些应用场景中，某一开锁者可以独自开锁，而某些开锁者必须和其他人配合才能开锁。在一实施例中，在多重签名机制的基础上为不同的私钥签名分配不同的权重。在验证时，只有当前通过验证的所有私钥签名的总权重大于预设值时才判定具有权益数据的控制权，可以开锁。

[0073] 进一步的，在一实施例中，在多重签名机制的基础上还引入了有效状态设定。即，当一个特定的私钥签名被确认正确后，其可以保持一段时间的有效状态，有效状态的存在时间根据私钥拥有者的具体权限而设定。在该私钥签名保持有效状态的期间，其他的私钥签名可以被验证。

[0074] 在判断当前是否可以开锁时，并不是对当前通过验证的所有私钥签名进行综合的权益数据的控制权判定，而是对当前处于有效状态的所有私钥签名进行综合的权益数据的控制权判定。即根据当前接收到的一个或多个正确的私钥签名以及之前接收到的、仍保持有效状态的一个或多个私钥签名判断当前的一个或多个解锁方是否具有权益数据的控制权。

[0075] 例如，2/3签名，一共存在3个私钥签名正确的解锁方，其中，3个私钥签名均被设置为通过验证后立即进入有效状态，并且，解锁方A的私钥签名被设置为有效状态保持一天。

[0076] 解锁方A在某一时刻进行解锁操作，其私钥签名通过验证，但是由于当时只有一个私钥签名通过验证，因此不具有权益数据的控制权，不可以开锁。由于解锁方A的私钥签名被设置为有效状态保持一天，这样，在一天内，其他两个解锁方中的任意一个执行了开锁操作，其私钥签名通过验证后，处于有效状态的私钥签名就是2个，此时就具有权益数据的控制权，可以开锁。

[0077] 这样，虽然是需要两个签名配合的2/3签名验证机制，但是不需要2个解锁方同时执行解锁操作。

[0078] 进一步的，在一实施例中，针对权益数据具有第一私钥签名、第二私钥签名以及第三私钥签名三把正确的私钥签名，其中：

[0079] 第一私钥签名、第二私钥签名以及第三私钥签名被接收并验证正确后即进入钥匙有效状态，任意两把私钥签名同时有效时即可判定当前的一个或多个解锁方具有所述权益数据的控制权；

[0080] 第一私钥签名被接收后，在第一私钥签名对应的解锁方解除私钥签名状态之前，维持第一私钥签名的有效状态；

[0081] 第二私钥签名以及第三私钥签名被接收后，在进行一次控制权判定后解除第二私钥签名以及第三私钥签名的有效状态。

[0082] 在一个具体应用场景中使用上述方案。在租房应用场景中：

[0083] • 房东是个人行为，通常不受约束，如果其可以随意开锁进门，这对租房者不利；

[0084] • 中介公司持有的钥匙在使用时通常比较谨慎，因为其需要维护良好的信用和口碑。因此中介公司不会随意开锁去危害租房者的利益，即使有个别极端情况，概率上是比房东更受控的；

[0085] • 而租房者是使用者，需要随时可以开锁，不应该受到约束。

[0086] 因此为锁配备3把不同的钥匙，应用2/3签名，并分别分配给3方各自持有。具体如下：

[0087] • 中介公司持有1把钥匙，开锁操作单次有效；

[0088] • 租户持有1把钥匙，开锁操作单次有效；

[0089] • 房东持有1把钥匙，开锁操作永久有效(在全部3方持有者没有任何变更的情况下)，即：相当于这把钥匙一直插在钥匙孔里。由于是2/3签名，再有任一另一把钥匙插进来，即可开锁。

[0090] 这样以来，租户和中介公司任意一方都可以使用自己的钥匙开锁，但房东不行，他必须再找个人帮他开锁。

[0091] 在上述设置中。但房东也应该有开门的需求，那么可以找中介公司和租户任一方协助开门，这同时保证了其它两方的知情权；而房东也需要在必要的时候能够维权，那么我们也正好兼容此需求！在上述设置方案中，房东可以随时拔下钥匙(即：通知锁让本钥匙失效)，此时租户和中介公司单方面都无法开锁，他们必须再找一方协商解决问题。不过房东通常没有理由这样做，除非利益受到损害。仅能拔下钥匙的维权方式设计，也避免了维权过猛而造成的伤害。

[0092] 在房东拔下钥匙后，如果仍希望避免租户和中介公司串通单方面开锁，可以在锁程序中给房东的钥匙设置投票权超过1的权重，这样租户和中介的权重之和低于2，仍无法开锁，必须经过房东的同意并重新插入钥匙。即：房东拔下钥匙可以迫使租户和中介公司积极应对房东的维权要求。

[0093] 进一步的，在一实施例中，钥匙的转让交接与权益数据的转让交接操作无异。进一步的，在一实施例中，在多签名机制下，需要多方签署这笔交易。权益数据转让交接操作完成后，这个携带锁ID的UTXO将归接收地址的控制者所有，控制者的数量也可以变更，各方的投票权重可以由植入UTXO内的协议控制。UTXO控制权的变更意味着锁的控制权变更，即等同于钥匙的变更交接：钥匙进入了新的使用者手中。使用者的变更可以仅仅是租户，也可以是其它任意各方使用者，这取决于(或决定着)UTXO的接收地址如何构造。

[0094] 具体的，在一实施例中，将锁的身份标识植入到UTXO，需要创建协议数据结构并完成一个支付操作，此时该协议化的UTXO便存在于区块链上了。

[0095] 例如，这是一个正常的交易(transaction)数据结构：

[0096]

{
"hash":"3b06b1e9d70217d5e02644703fe79f54355b0ea05cd535787f5a6c627f1c",
"ver":1,
"vin_sz":2,
"vout_sz":1,
"lock_time":0,
"size":404,
"in":[
{
"prev_out":{
"hash":"022e05bdfa2e148bc1882cb7a81506b8316fee6957b11625126d075a8cf8791b",
"n":0
},
"scriptSig":"304402203c1db72394263dd50070b91bb1da9125c591f15772dfc628f41447dabb7798a10220302dc6e7e8c81e24da9a99d5ac7233b90156a410051a50bb29d55aabbf0ff24d0102b8c918bd169a5e669cc149549f822dd5f2c50872eb83172a1c69172277fe378f"
},
{
// 另一个输入，省略…
}
],
"out":[

[0097]
```
{
 "value":12.51603279
 "scriptPubKey":"OP_DUP OP_HASH160
69e02e18b5705a05dd6b283d517716c894b3d42e
   OP_EQUALVERIFY OP_CHECKSIG"
}
]
}
```

[0098] 在scriptPubKey字段位置插入协议信息。协议内容格式定义为如下Uri格式：

[0099] lock://id/e10exxxx.../sig/2*3/pubs/62e*73a*84b/addrs/1ac*2bd*2ce/weights/1.2*0.9*0.9/povs/0*1*1

[0100] 将该Uri+OP_DROP插入到scriptPubKey字段值的OP_DUP前面，即可完成植入。其中：lock是协议的识别标识；id表示锁的ID；sig表示多签名规则；pubs表示多签验证需要使用的公钥信息，这里仅填开头最短的部分，用于辨识与weights权重设置对应的公钥；addrs表示多签验证需要使用的公钥hash地址信息(与pubs二选一，其中的“*”是分隔符，将多个地址或者公钥分开)；weights表示每个公钥的投票权重；povs表示组合有效期，0为永久有效，1为单次有效。

[0101] 基于本发明的方法，本发明还提出了一种基于区块链的解锁系统。如图4所示，在一实施例中，解锁系统400配合锁401以及钥匙402，系统400包括：

[0102] 钥匙接口410，其配置为判断当前是否接收到来自解锁方的解锁请求，解锁请求由钥匙402输出；

[0103] 验证数据提取模块420，其配置为当存在解锁请求时在区块链403上查找包含锁401的身份标识的权益数据；

[0104] 验证模块430，其配置为验证解锁方是否具有权益数据的控制权以确认解锁方是否有锁401的控制权；

[0105] 控制输出模块440，其配置为当解锁方具有锁401的控制权时向锁401输出解锁指令。

[0106] 锁401在接收到解锁指令时即执行开锁操作。

[0107] 进一步的，在一实施例中：

[0108] 验证数据提取模块420还配置为从权益数据中取出控制者的身份标识(公钥或地址)；

[0109] 钥匙接口410还配置为接收来自解锁方的私钥签名，私钥签名由钥匙402输出；

[0110] 验证模块430还配置为根据控制者的身份标识验证解锁方的私钥签名的正确性，当解锁方的私钥签名正确时根据解锁方的私钥签名以及控制者的身份标识判断解锁方是否具有权益数据的控制权。

[0111] 进一步具体的，在一实施例中，钥匙接口410还配置为：

[0112] 向解锁方发送钥匙要求，即向钥匙402输出钥匙请求；

[0113] 接收来自解锁方的钥匙数据，其中，钥匙数据是解锁方根据掌握的私钥对钥匙请求签名而生成的，即，钥匙402根据掌握的私钥对钥匙请求签名而生成钥匙数据；

[0114] 从钥匙数据中提取私钥签名。

[0115] 进一步的，如图5所示，在一实施例中：

[0116] 系统还包括钥匙寄存器550，钥匙寄存器550配置为存储之前接收到的、可以保持有效状态的一个或多个私钥签名；

[0117] 验证模块530配置为根据当前接收到的一个或多个正确的私钥签名以及之前接收到的、仍保持有效状态的一个或多个私钥签名判断当前的一个或多个解锁方是否具有权益数据的控制权。

[0118] 进一步的，在一实施例中，锁401被设置为当没有解锁指令或上一解锁指令失效时即执行闭锁操作。

[0119] 进一步的，在一实施例中，锁401被设置为当接收到闭锁指令时执行闭锁操作。钥匙接口410还配置为判断当前是否接收到来自解锁方的闭锁请求。针对闭锁请求的验证流程同解锁请求，针对闭锁请求，当解锁方被验证具有权益数据的控制权时，控制输出模块440向锁输出闭锁指令。

[0120] 进一步的，在一实施例中，系统还包括锁标识模块，锁标识模块能够在重置时生成锁的身份标识(ID)(确切地说，是符合高阶最小熵分布的安全随机数)，并能够在特定操作下(如按住某按键)允许查询该ID，以备创建协议UTXO和钥匙时使用。

[0121] 在一实施例中，系统在最初和需要控制的锁建立控制关联时，即是创建一个协议化的UTXO，同时也是钥匙的创建过程。分为以下步骤：

[0122] 1.系统获得锁的控制权，确认对锁的控制权；

[0123] 2.重置并生成新的锁ID；

[0124] 3.根据即将分配钥匙的各方所提供的公钥，生成UTXO的目标地址；

[0125] 4.创建协议数据，打包并发送交易，至此协议化UTXO已生成。

[0126] 在一具体的应用场景中，系统随时联网查询其控制的锁的ID对应的UTXO，进而验证该UTXO的控制权以确认是否拥有钥匙。但需要注意一点：该UTXO必须与初始化时的UTXO处于同一条支付流转链上，不可以是任意UTXO，以防恶意破坏。在操作上，应遵循如下逻辑：锁程序应当存储初始化时的UTXO的txid(区块链账本上记录的一条交易的id)，若发现该UTXO已经被支付，则应当查找支付目的地的UTXO并更新锁内存储的txid，而忽略其它任何包含本ID的UTXO(它们可能是恶意混淆，当然自己也可以故意混淆以防止追踪)。有txid的跟踪，可以不要求锁ID必须唯一。

[0127] 进一步的，在一实施例中，系统还包括UTXO缓存器，UTXO缓存器用于缓存查询到的UTXO(txid)，在网络连接出现故障的时候(表现为抛出异常)，将直接使用缓存的UTXO(txid)进行验证。

[0128] 具体的，根据本发明的系统一个具体应用场景如下所述。

[0129] (一)单签名钥匙

[0130] 协议化UTXO的收款地址是一个公钥或公钥hash，即，钥匙是单签名钥匙(也可以说这把锁是单钥匙孔的锁)，这需要在初始化时与生成的协议保持一致。开锁步骤如下：

[0131] ①用户发出开锁申请，使用钥匙硬件向解锁系统输出开锁请求。

[0132] ②解锁系统联网连接区块链节点查询指定txid的UTXO,检查其内置的协议是否包含自身ID,并检查该UTXO是否已经被花费。若未被花费,进行下一步;否则一直向后追溯到最新没有被花费的UTXO,并用其txid覆盖锁内之前缓存的txid,然后进行下一步。

[0133] ③解锁系统生成一个随机数并输出一条报文,报文中同时携带锁ID的前几位以示区别。该报文可简单表示如下:

[0134] {"id":"d62a9e","random":"34576578901823912"}

[0135] ④钥匙硬件收到该报文,用控制该UTXO的私钥签名该随机数,并回复报文(其中sig表示对随机数的签名,pub表示公钥):

[0136]

{"id":"d62a9e",

"sig":"304402203c1db72394263dd50070b91bb1da9125c591f15772dfc628f41447dabb7798a10220302dc6e7e8c81e24da9a99d5ac7233b90156a410051a50bb29d55aabbf0ff24d01",

"pub":"02b8c918bd169a5e669cc149549f822dd5f2c50872eb83172a1c69172277fe378f"}

[0137] ⑤解锁系统收到报文的回复,先判断pub是否与UTXO中的公钥或公钥hash匹配,若匹配,则用公钥解密sig数据,看结果是否与自己发出去的随机数random相等,相等表示验证通过(即:插入了正确的钥匙)。

[0138] ⑥若上一步验证通过,则解锁系统向锁发出解锁指令,锁在接收到解锁指令后驱动马达执行物理开锁操作,完成开锁;否则广播回复报文钥匙不匹配。

[0139] (二)多签名钥匙

[0140] 协议化UTXO的收款地址是多个公钥或公钥hash。由于多签名钥匙的开锁组合有效期分为单次有效,和永久有效。这里定义单次有效为从该钥匙请求开锁到实际物理开锁的整个时间段内(开锁即清除状态)。永久有效是指从当前UTXO产生到被花费的整个时间段内(更新txid即清除状态)。开锁步骤如下:

[0141] ①用户发出开锁申请,使用钥匙硬件向解锁系统输出开锁请求。

[0142] ②解锁系统联网连接区块链节点查询指定txid的UTXO,检查其内置的协议是否包含自身ID,并检查该UTXO是否已经被花费。若未被花费,进行下一步;否则一直向后追溯到最新没有被花费的UTXO,并用其txid覆盖锁内之前存储的txid,然后进行下一步;

[0143] ③解锁系统生成一个随机数并广播一条报文,报文中同时携带锁ID的前几位以示区别。该报文可简单表示如下:

[0144] {"id":"d62a9e","random":"34576578901823912"}

[0145] ④钥匙硬件收到该报文,用对该UTXO有一部分控制权的私钥签名该随机数,并回复报文(其中sig表示对随机数的签名,pub表示公钥,script表示多签名收款脚本):

[0146]

{"id":"d62a9e",

"sig":"304402203c1db72394263dd50070b91bb1da9125c591f15772dfc628f41447dabb

[0147]

7798a10220302dc6e7e8c81e24da9a99d5ac7233b90156a410051a50bb29d55aabbf0ff24d01",

"pub":"02b8c918bd169a5e669cc149549f822dd5f2c50872eb83172a1c69172277fe378f", "script":"xxxxxxxx"}

[0148] ⑤解锁系统收到报文,先判断hash(script)是否与UTXO中的地址匹配,若匹配,则用公钥解密sig数据,看结果是否与自己发出去的随机数random相等,相等表示验证通过(即:插入了正确的钥匙);

[0149] ⑥解锁系统检查是否存在永久有效的钥匙已经保持了状态;

[0150] ⑦解锁系统根据协议内容,判定当前钥匙是否为永久有效,是否需要保持状态;

[0151] ⑧解锁系统累加权重之和,看是否满足投票最低要求。满足则表示通过开锁申请;

[0152] ⑨若上一步验证通过,则解锁系统向锁发出解锁指令,锁在接收到解锁指令后驱动马达执行物理开锁操作,完成开锁;否则广播回复报文钥匙不匹配。

[0153] 虽然本发明所公开的实施方式如上,但所述的内容只是为了便于理解本发明而采用的实施方式,并非用以限定本发明。本发明所述的方法还可有其他多种实施例。在不背离本发明实质的情况下,熟悉本领域的技术人员当可根据本发明做出各种相应的改变或变形,但这些相应的改变或变形都应属于本发明的权利要求的保护范围。

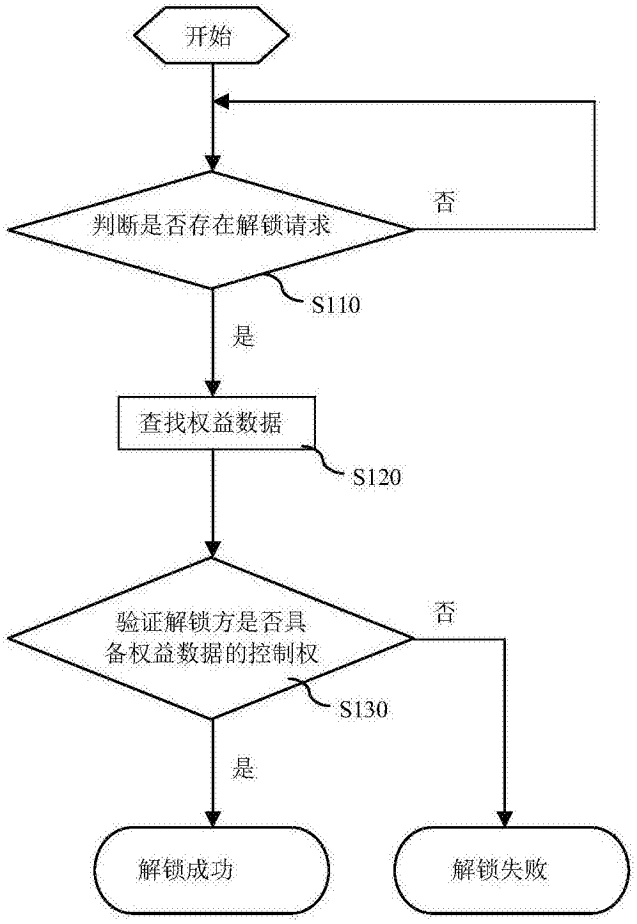

开始
判断是否存在解锁请求
否
S110
是
查找权益数据
S120
验证解锁方是否具
备权益数据的控制权
否
S130
是
解锁成功
解锁失败

图1

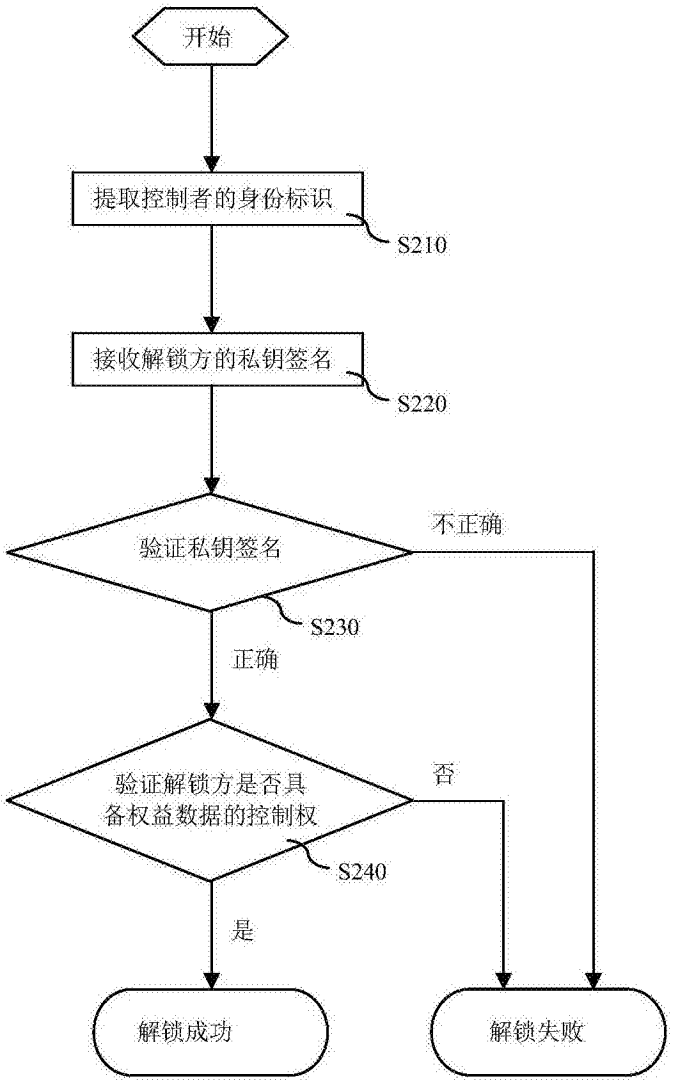

开始
提取控制者的身份标识
S210
接收解锁方的私钥签名
S220
验证私钥签名
不正确
S230
正确
验证解锁方是否具
备权益数据的控制权
否
S240
是
解锁成功
解锁失败

图2

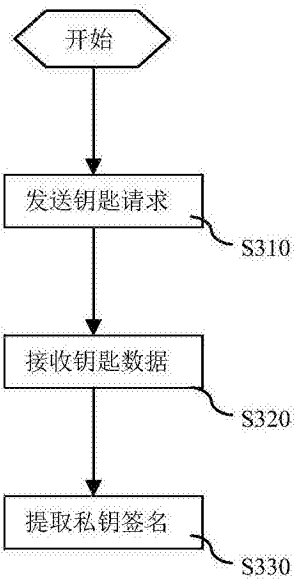

开始
发送钥匙请求
S310
接收钥匙数据
S320
提取私钥签名
S330

图3

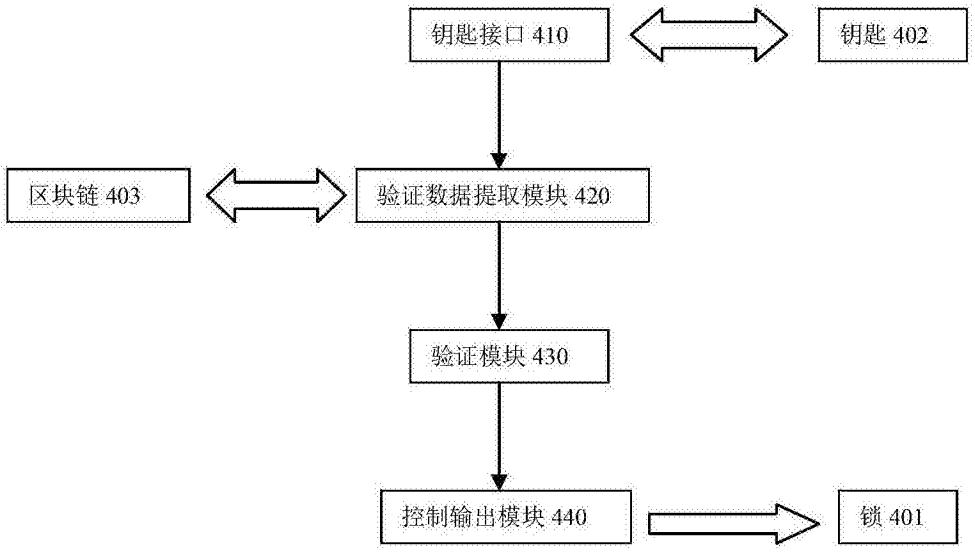

钥匙接口 410
钥匙 402
区块链 403
验证数据提取模块 420
验证模块 430
控制输出模块 440
锁 401

图4

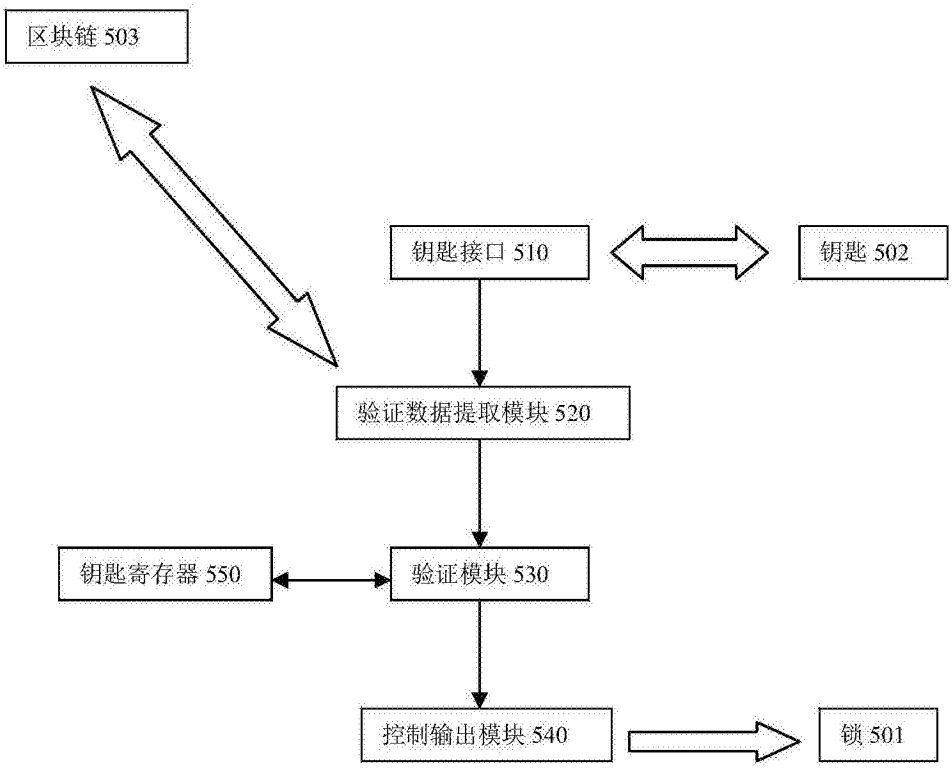

区块链 503
钥匙接口 510
钥匙 502
验证数据提取模块 520
钥匙寄存器 550
验证模块 530
控制输出模块 540
锁 501

图5