

一种基于智能合约的数据处理方法、设备及存储介质

本申请要求于 2019 年 8 月 29 日提交国家知识产权局、申请号为 201910808351.8, 申请名称为“一种基于智能合约的数据处理方法、设备及存储介质”的中国专利申请的优先权,其全部内容通过引用结合在本申请中。

技术领域

本申请涉及互联网技术领域,尤其涉及一种基于智能合约的数据处理方法、设备及存储介质。

背景技术

服务器固件(Firmware)是指写入服务器的 EPROM(可擦写可编程只读存储器)或 EEPROM(电可擦可编程只读存储器)中的程序。应当理解,服务器固件中可以存储有相应操作系统中的硬件设备的基本参数和程序,能够为该操作系统提供最底层、最直接的硬件控制。

目前,服务器运维人员可以通过离线分析、带内网络或者系统管理中断等固件更新方法对服务器的固件(例如, BIOS(Basic Input Output System, 基本输入输出系统)的固件)进行更新操作,以增加该操作系统的新功能或修复该操作系统的异常。

发明内容

本申请实施例提供一种基于智能合约的数据处理方法、设备及存储介质,可以提高固件更新的安全性和可靠性。

本申请实施例提供了一种基于智能合约的数据处理方法,所述方法由作为合约节点的电子设备执行,包括:

接收执行节点发送的针对第一服务器节点的固件更新请求;所述固件更新请求中至少包含:所述第一服务器节点的更新版本参数;

根据所述固件更新请求调用智能合约,基于所述智能合约从区块链

上获取与所述第一服务器节点相关联的固件版本更新记录和固件版本发布记录；所述固件版本发布记录为所述区块链上的发布节点基于共识机制所确定的；

根据所述固件版本更新记录、所述固件版本发布记录、所述更新版本参数，确定所述固件更新请求的合法性。

本申请实施例提供了一种基于智能合约的数据处理装置，所述装置应用于合约节点，包括：

请求接收模块，用于接收执行节点发送的针对第一服务器节点的固件更新请求；所述固件更新请求中至少包含：所述第一服务器节点的更新版本参数；

合约调用模块，用于根据所述固件更新请求调用智能合约，基于所述智能合约从区块链上获取与所述第一服务器节点相关联的固件版本更新记录和固件版本发布记录；所述固件版本发布记录为所述区块链上的发布节点基于共识机制所确定的；

合法确定模块，用于根据所述固件版本更新记录、所述固件版本发布记录、所述更新版本参数，确定所述固件更新请求的合法性。

本申请实施例提供了一种节点设备，包括：处理器以及存储器；

所述处理器与存储器相连，其中，所述存储器用于存储程序代码，所述处理器用于调用所述程序代码，以执行以下操作：

接收执行节点发送的针对第一服务器节点的固件更新请求；所述固件更新请求中至少包含：所述第一服务器节点的更新版本参数；

根据所述固件更新请求调用智能合约，基于所述智能合约从区块链上获取与所述第一服务器节点相关联的固件版本更新记录和固件版本发布记录；所述固件版本发布记录为所述区块链上的发布节点基于共识机制所确定的；

根据所述固件版本更新记录、所述固件版本发布记录、所述更新版本参数，确定所述固件更新请求的合法性。

本申请实施例提供了一种计算机存储介质，所述计算机存储介质存

储有计算机程序，所述计算机程序包括程序指令，当处理器执行所述程序指令时执行如本申请任一实施例所述的方法。

附图说明

为了更清楚地说明本申请实施例的技术方案，下面将对实施例描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

图1是本申请实施例提供的一种区块链网络拓扑结构的结构示意图；

图2是本申请实施例提供的一种基于智能合约的数据处理方法的流程图示意图；

图3是本申请实施例提供的一种嵌套式哈希链的区块结构的示意图；

图4a和图4b是本申请实施例提供的一种进行版本发布控制的示意图；

图5是本申请实施例提供的另一种基于智能合约的数据处理方法的流程图示意图；

图6是本申请实施例提供的一种获取版本发布行为信息的示意图；

图7是本申请实施例提供的一种构建固件版本更新记录的示意图；

图8是本申请实施例提供的一种通过发布节点进行安全审计的示意图；

图9是本申请实施例提供的一种基于智能合约的数据处理装置的结构示意图；以及

图10是本申请实施例提供的节点设备的结构示意图。

具体实施方式

下面将结合本申请实施例中的附图，对本申请实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例仅仅是本申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本

申请保护的范围。

目前，由于服务器厂商官网中所提供的固件均处于开放下载状态，运维人员无论采用哪种固件更新方法都可以从服务器厂商官网下载到相应机型的固件，并按照所使用的固件更新方法的步骤执行固件升级操作。所以，一旦用于进行固件更新的固件中的数据被非法渗透和篡改，将严重威胁到该操作系统的底层的安全防护，换言之，现有的固件更新方法无法确保固件更新的安全性和可靠性。

请参见图 1，是本申请实施例提供的一种区块链网络拓扑结构的结构示意图。如图 1 所示的区块链网络拓扑结构可以应用于在企业内网中具有大规模服务器的应用场景。如图 1 所示，该区块链网络拓扑结构可以包含图 1 所示的发布层 100、合约层 200 和实施层 300。

如图 1 所示，位于所述发布层 100 中的节点可以称之为发布节点，具体可以包含图 1 所示的发布节点 10a、发布节点 10b 和发布节点 10c。应当理解，该发布层 100 中还可以包含其它发布节点（暂未在图 1 中标示出）。如图 1 所示，位于合约层 200 中的节点可以统称为合约节点，合约节点可以理解为在区块链中能够调用并执行存储在区块链中的智能合约的节点，具体地，该合约层 200 中可以包含图 1 所示的合约节点 20a 和合约节点 20b。如图 1 所示，位于实施层 300 中的节点主要由两类节点所构成，这两类节点中的一类节点可以称之为执行节点，另一类节点可以称之为服务器节点。其中，执行节点具体可以包含图 1 所示的执行节点 30a 和执行节点 30b；同理，服务器节点具体可以包含图 1 所示的服务器节点 40a、服务器节点 40b 和服务器节点 40c。

应当理解，如图 1 所示，位于该区块链网络拓扑结构中的发布节点、合约节点、执行节点以及服务器节点可以统称为区块链网络中的节点。在区块链网络中，这些节点之间可以进行全连接，比如，图 1 所示的发布节点 10a 与合约节点 20a 之间的第一网络连接方式则可以称之为全连接，即本申请实施例可以将能够在一跳之内连接到的网络连接方式称之为全连接。在本申请实施例中，区块链中的节点之间还可以通过物理网

络进行连接，比如，图 1 所示的合约节点 20a 和合约节点 20b 之间的第二网络连接方式则可以需要通过物理网络（例如，虚拟网卡）进行连接，即无法在一跳之内连接到的网络连接方式可以称之为通过物理网络进行物理网络连接。

其中，如图 1 所示的发布节点 10a 可以接收厂商（例如：厂商 A）发布的最新版本固件文件（例如，固件文件 x）。进一步地，发布节点 10a 可以对获取到的固件文件 x 进行文件结构解析，以根据固件文件 x 中的文件内容进一步确定该固件文本 x 的内部版本号（例如，最新固件版本号 AAAA），并计算该版本号对应的哈希值（例如，哈希值 1）。进一步，如图 1 所示，在区块链网络中的发布节点 10a 可以通过与合约节点 20a 之间的全连接向相邻的合约节点 20a 发布该厂商 A 的固件版本发布参数（即最新固件版本号 AAAA、哈希值 1），与此同时合约节点可以将该厂商 A 的版本发布参数传播至该合约层 200 中的其他合约节点（比如，合约节点 20b）。换言之，在本申请实施例中的发布节点可以向合约层 200 中的所有合约节点广播该厂商 A 所发布的固件版本发布参数（即固件版本号 AAAA 和哈希值 1）。

其中，应当理解，发布节点 10a 在向合约节点 20a 发布该固件版本发布参数时，可以通过自身的私钥对该固件版本发布参数进行签名，并可以将签名后的信息和该发布节点 10a 的公钥作为版本发布确认信息，从而可以将该版本发布确认信息一并给到合约节点 20a，以使合约节点 20a 可以通过时间门限机制（即预设的时间窗口 T 内）完成区块链上的共识。其中，在本申请实施例中，在企业内部的区块链网络（例如，私有区块链网络）上，可以将区块链的共识限定在合约层 200 中的合约节点上进行。

比如，合约节点 20a 在接收到发布节点 10a 发送的版本发布确认信息之后，可以在预设的时间窗口 T 内进行等待，若在该时间窗口 T 内有其他发布节点（例如，图 1 所示的发布节点 10b）也广播同样的固件版本发布参数，则间接表明发布节点之间达成了共识，此时，该合约节点

可以将包含该版本发布参数的固件版本发布记录写入区块链，以确保达成合约共识。反之，若在时间窗口 T 之外的其他时间接收到其他发布节点所发布的该厂商 A 发布的固件版本发布参数，则间接表明该厂商 A 发布的版本发布参数是不被认可的，此时，合约节点并不会将在时间窗口 T 内所接收到的由发布节点 10a 所广播的包含版本发布参数的区块写入区块链。由此可见，本申请实施例通过节点之间的共识可以有效地在根源上避免某一个发布节点被非法攻击者入侵而引发的系统性风险。

应当理解，由于合约层 200 中的合约节点之间可以通过物理网络进行通讯，所以，在合约节点 20a 在接收到该包含该版本发布参数的区块之后，可以将该包含版本发布参数的区块在各合约节点之间进行同步，以进一步达成合约共识，即本申请实施例还可以通过共识机制有效地在区块链上判断该厂商 A 所发布的版本发布参数（即固件版本号 AAAA 和哈希值 1）为最新版本的厂商固件信息。

其中，本申请实施例所描述的区块链是一种特殊的分布式数据库，也可以称之为分布式账本系统。其中，该分布式数据库（即区块链）可以具备以下两个功能：一是可以用来存储信息，即区块链上的节点可以将任何需要保存的信息，通过前述时间门限机制达成共识后写入区块链，并可以在区块链上通过共识机制达成合约共识。应当理解，这里的共识机制还可以包含：工作量证明（PoW）、权益证明（PoS）、实用拜占庭容错（PBFT）等方式，这里将不做限定；二是任何人都可以架设服务器，加入区块链网络，成为一个节点。所以，该分布式账本系统可以理解为一个可以在多个站点、不同地理位置或者多个机构组成的区块链网络里进行分享的资产数据库。在一个区块链网络里的参与者可以获得一个唯一的、真实账本的副本。账本里的任何改动都会所有的副本中被反映出来，反应时间可以在几分钟甚至是几秒内。

应当理解，在同一区块链网络中，可以将区块链的共识达成（即发布节点之间的共识和合约节点之间的共识）与智能合约的执行限定在特定的合约节点上执行。其中，本申请所描述的智能合约是指由更新操作

的执行者（例如，图 1 所示的执行节点 30a）和被更新服务器（即图 1 所示的服务器节点 40a）共同参与的固件更新合约，该固件更新合约可以用于指示合约节点 20a 在接收到执行节点 30a 对服务器节点 40a（即第一服务器节点）进行固件更新的请求（即固件更新请求）时，可以通过从区块链上获取到的与第一服务器节点相关联的固件版本更新记录、固件版本发布记录以及固件更新请求中的更新版本参数，以判断由执行节点 30a 发送的固件更新请求的合法性。鉴于此，本申请实施例表明在合约节点 20a 确定该固件更新请求为合法更新请求时，可以准许执行节点 30a 基于更新版本参数对该第一服务器节点当前所运行的运行版本参数进行更新；反之，则执行节点无法对第一服务器节点进行固件更新。由此可见，通过调用该智能合约可以有效地区分出合法还是非法的固件更新操作，从而可以避免非法情况下的固件更新，进而可以有效地提高固件更新的安全性和可靠性。

其中，在本申请实施例中，区块链在区块结构上可以由一个个区块（block）组成，这些由一个区块所组成的区块链可以称之为嵌套哈希链，即在该嵌套哈希链中的每个区块均会包含前一个区块的哈希以及针对第一服务器节点（例如，图 1 所示的服务器节点 40a）进行固件更新的多个历史版本更新行为信息，包含这些历史版本更新行为信息的版本更新记录可以统称为固件版本更新记录。应当理解，本申请实施例中的区块中可以包含相应的固件版本更新记录；在本申请实施例中，区块中还可以包含相应的固件版本发布记录。为便于理解，本申请实施例可以将包含固件版本更新记录的区块统称为第一区块，并可以将包含固件版本发布记录的区块称之为第二区块。其中，第一区块中可以包含固件版本更新记录，且固件版本更新记录中的版本更新行为信息可以携带第一版本参数和第二版本参数，所述第二版本参数为对所述第一版本参数进行固件更新后的版本参数。由此可见，所述固件版本更新记录可以用于记录对第一服务器节点的固件版本信息进行更新操作这一行为的版本更新行为信息。该第一服务器节点中若有多个固件已完成固件更新

操作，则可以产生多个历史版本更新行为信息。

本申请实施例中的所有合法的固件更新操作所对应的固件版本更新记录和版本发布操作对应的固件版本发布记录都可以记录在区块链中，以便于后续发布节点可以在链上查询每个服务器节点的安全状态，比如，可以对每一服务器节点的链下固件信息和链上固件信息进行安全审计，从而可以从大规模的服务器节点中找出疑似异常的服务器节点进行告警，从而可以采用相应的应急响应流程，比如，可以将疑似异常的服务器进行网络隔离，从而可以有效地提升对非法入侵手段的感知能力，进而可以显著的增加企业内网的安全性。

其中，在本申请实施例中，如图 1 所示，执行节点 30a 在针对服务器节点 40a（即第一服务器节点）进行固件更新之前，可以向图 1 所示的合约节点 20a 发送固件更新请求，从而可以在合约节点 20a 确定该固件更新请求为合法更新请求时，准许执行节点 30a 进行固件更新。本申请实施例在对第一服务器节点（例如，图 1 所示的服务器节点 40b）进行固件更新之后，可以生成新的固件版本更新记录，并可以将包含该新的固件版本更新记录的区块称之为新的第一区块，从而可以将该新的第一区块有序地写入所述区块链中，以便于后续发布节点可以在区块链上对该第一服务器节点的安全状态进行审计。即发布节点可以基于区块链中的第一区块中所存储的该第一服务器节点的链上固件信息和在本地所采集到的链下固件信息，对该第一服务器节点的安全状态进行评估。如果两者是一致的，则可以返回正常值，反之，则说明该第一服务器节点的安全状态出现异常，此时，该发布节点可以通过移动端或者 WEB 端告警信息的方式向该第一服务器节点所对应的用户返回告警信息（即工单信息）。

其中，所述合约节点 20a 获取所述固件更新请求，并基于固件版本发布记录和固件版本更新记录以及更新版本参数对固件更新请求的合法性进行鉴权的具体过程，可以参见如下图 2 至图 8 所对应的实施例。

进一步地，请一并参见图 2，是本申请实施例提供的一种基于智能

合约的数据处理方法的流程示意图。如图 2 所示，所述方法可以应用于作为合约节点的电子设备，具体可以包含以下步骤 S101-步骤 S103。

步骤 S101，接收执行节点发送的针对第一服务器节点的固件更新请求。

具体地，合约节点可以接收执行节点发送的针对第一服务器节点的固件更新请求。应当理解，为了确保固件更新的安全性和可靠性，本申请实施例中的执行节点在对第一服务器节点进行固件更新之前，需要向合约节点发送固件更新请求，所述固件更新请求中至少可以包含：所述第一服务器节点的更新版本参数；在本申请实施例中，所述固件更新请求中还可以包含所述第一服务器节点的运行版本参数。

其中，所述运行版本参数可以包含该第一服务器节点当前运行的旧固件版本信息和该旧固件版本信息的哈希值。在本申请实施例中，所述更新版本参数可以包含当前计划用于对该第一服务器节点进行固件更新的新固件版本信息和该新固件版本信息的哈希值。为便于理解，本申请实施例可以将该第一服务器节点当前运行的旧固件版本信息称之为运行版本信息(例如，固件版本 V1)，并将该旧固件版本的哈希值称之为运行版本哈希值(例如，哈希值 H1)。此外，本申请实施例可以将当前计划用于对该第一服务器节点进行固件更新的新固件版本信息称之为更新版本信息(例如，固件版本 V2)，并将该新固件版本信息的哈希值称之为更新版本哈希值(例如，哈希值 H2)。其中，可以理解的是，这里的运行版本哈希值还可以称之为旧固件哈希值，同理，所述更新版本哈希值还可以称之为新固件哈希值。

其中，在本申请实施例中，执行节点在发送固件更新请求之前，可以将获取到的运行版本参数和更新版本参数称之为后续用于生成固件版本更新记录的输入信息，即该执行节点可以将该第一服务器节点当前运行的运行版本参数(即固件版本 V1、哈希值 H1)和计划用于对该第一服务器节点的固件进行更新的更新版本参数(即固件版本 V2、哈希值 H2)称之为输入信息，并通过该执行节点的私钥对该输入信息进行

签名，并将签名后的第一签名信息和该执行节点的公钥添加至更新请求中，以得到前述固件更新请求。所以，在本申请实施例中，该合约节点在接收到固件更新请求之后，可以通过接收到的该执行节点的公钥对该执行节点的第一签名信息进行签名验证，并在签名验证成功时得到该执行节点所提供的更新版本参数和运行版本参数。

其中，所述执行节点可以为上述图1所对应实施例中位于实施层300中的执行节点30a，所述第一服务器节点可以为上述图1所对应实施例中位于所述实施层300中的服务器节点40a，所述合约节点可以为上述图1所对应实施例中位于合约层200中的合约节点20a。在本申请实施例中，在区块链中的执行节点30a、服务器节点40a和合约节点20a可以统称为区块链网络中的节点。应当理解的是，对于加入区块链中的每个节点而言，区块链均可以为其分配相应的公钥、私钥对。其中，每个节点的公钥可以被该区块链中的其他节点所获悉，但各自的私钥则可以由相应节点单独进行妥善存储。比如，对于服务器节点而言，由于实体为物理设备，则可以基于可信执行环境（TEE）的可信计算环境来保存该服务器节点的私钥，这样，可以确保区块链中的其他节点无法获悉该服务器节点的私钥。又比如，可以采用特定的存储介质来保存执行节点的私钥。由此可见，通过对区块链中的各节点的私钥进行妥善保存，可以确保后续合约执行过程的安全性和可靠性。

其中，应当理解，区块链通过为上述实施层300中的执行节点分配相应的公钥、私钥对，等效于赋予了这些执行节点特殊身份，从而可以有效地提高非法攻击者伪造执行节点的身份来非法操作服务器固件的难度。

步骤S102，根据所述固件更新请求调用智能合约，基于所述智能合约从区块链上获取与所述第一服务器节点相关联的固件版本更新记录和固件版本发布记录；所述固件版本发布记录为所述区块链上的发布节点基于共识机制所确定的；

具体地，合约节点可以调用所述智能合约以获取所述第一服务器节

点在所述区块链中的区块链地址；所述区块链地址为所述区块链根据所述第一服务器节点的公钥信息进行散列计算后所唯一确定的；进一步地，合约节点可以基于所述区块链地址从所述区块链上获取与所述第一服务器节点相关联的第一区块和第二区块；进一步地，合约节点可以在所述第一区块中将携带第一版本参数和第二版本参数的历史版本更新行为信息，确定为与所述第一服务器节点相关联的固件版本更新记录；所述第二版本参数为对所述第一版本参数进行固件更新后的版本参数；进一步地，合约节点可以从所述第二区块中将与所述第一服务器节点相关联的历史版本发布行为信息，确定为固件版本发布记录。

其中，在本申请实施例中，如上述图 1 所述的区块链网络拓扑结构可知，区块链网络中的每一个节点（即每一个分布式节点）在区块链系统中均可以作为单独的个体进行信息的接收、处理、反馈。换言之，位于该区块链网络中的这些节点既接收信息也产生信息，且节点之间可以通过维护共同的区块链来保持通信。

其中，在本申请实施例中，在区块链网络中的节点（比如，合约节点）可以创造新的区块，在新的区块被创造后会以广播的形式通知该区块链网络中的其他节点（例如，上述图 1 所对应实施例中的其他合约节点），其他节点会对这个区块进行验证，当该区块链网络中有超过 51% 的节点验证通过后，确认达成节点之间的共识（例如，合约共识），此时，这个新的区块就可以被添加到区块链上。

其中，在本申请实施例中，合约节点在通过智能合约对服务器节点的固件进行更新管控的过程中，可以记录执行节点每一次通过第二版本参数对服务器节点中的固件（例如，上述第一服务器节点，为上述图 1 所对应实施例中的服务器节点 40a）的第一版本参数进行固件更新的版本更新行为信息。本申请实施例可以对服务器节点中的一个固件或者多个固件进行更新操作，这里将不进行限定。

在本申请实施例中，执行节点每完成一次对服务器节点中的固件进行更新的操作，则可以生成一个固件更新日志信息，即生成一个子固件

版本更新记录；其中，一个固件更新日志信息（即子固件版本更新记录）中可以包含对该第一服务器节点中的某个固件进行固件更新的版本更新行为信息。在本申请实施例中，该合约节点可以进一步将包含每个固件的版本更新行为信息的子固件版本更新记录统称为第一服务器节点中的固件的固件版本更新记录。

本申请实施例中可以将对第一服务器节点中的固件进行更新前的旧固件版本信息和旧固件哈希值称之为第一版本参数，将对第一服务器节点中的固件进行更新后的新固件版本信息和新固件哈希值称之为第二版本参数。

其中，在对服务器节点 40a 进行固件更新的应用场景中，可以将所有的已确定合法的更新结果（即固件版本更新记录）有序地写入区块链中，这样，服务器节点在当前接收到针对上述步骤 S101 中的固件更新请求时，可以基于该服务器节点的区块链地址从区块链中获取与该第一服务器节点的固件相关联的第一区块和第二区块。本申请实施例中的第一区块可以包含：至少一个第一区块，每个第一区块中均可以携带相应更新时间戳对应的固件版本更新记录，该固件版本更新记录中可以包含一个或者多个子固件版本更新记录。为便于进行理解，本申请实施例可以将每个子固件版本更新记录统称为第一服务器节点中的相应固件的固件版本更新记录。应当理解，每完成一次对该第一服务器节点的相应固件的版本信息的更新操作，即可以生成一个子固件版本更新记录，从而可以得到一个固件版本更新记录。

其中，在本申请实施例中，在上一区块生成到新区块生成的这个区块生成时间段内，若执行节点仅对第一服务器节点中的一个固件完成了更新操作，则可以得到一个子固件版本更新记录，此时，可以将该子固件版本更新记录作为新区块中的固件版本更新记录。在本申请实施例中，在这个区块生成时间段内，若执行节点已对第一服务器节点中的多个固件完成了更新操作，则可以得到多个子固件版本更新记录，此时，可以将这些子固件版本更新记录统称为该新区块中的固件版本更新记录。

同理，本申请实施例中的第二区块可以包含：至少一个第二区块，每个第二区块中均可以携带相应发布时间戳对应的固件版本发布记录，应当理解，每完成一次对该第一服务器节点的相应固件的版本信息的发布操作，即可以生成一个固件版本发布记录。

其中，应当理解，在本次智能合约的调用过程中，可以将先前调用智能合约所得到的固件版本更新记录中的版本更新行为信息称之为历史版本更新行为信息，并将先前得到的固件版本发布记录中的版本发布行为信息称之为历史版本发布行为信息。

为便于理解，本申请实施例以对第一服务器节点中的目标固件进行固件更新后所得到的固件版本更新记录为例，以阐述该目标固件的固件版本更新记录与第一区块之间的关联关系。进一步地，请参见图 3，是本申请实施例提供的一种嵌套式哈希链的区块结构的示意图。如图 3 所示的嵌套式哈希链中至少可以包含图 3 所示的区块 N、区块 N+1 和区块 N+2。其中，N 可以为正整数。如图 3 所示的每个区块中均记录有对第一服务器节点进行固件更新后的固件更新日志信息，本申请实施例可以将每个固件更新日志信息（即子固件版本更新记录）统称为固件版本更新记录。

其中，如图 3 所示的每个区块（即第一区块）均可以包含前一个区块（即前一个第一区块）的哈希（称之为 Header）以及相应的固件版本更新记录，即该嵌套式哈希链中的区块中的数据可以具备唯一性和可追溯性。其中，如图 3 所示，区块 N 中可以包含该区块 N 的前一个区块的哈希值（即图 3 所示的区块 N-1 的哈希）和对第一服务器进行固件更新操作后所得到的固件版本更新记录 1。同理，区块 N+1 中可以包含该区块 N+1 的前一个区块的哈希值（即图 3 所示的区块 N 的哈希）和对第一服务器节点进行另一固件更新操作后所得到的固件版本更新记录 2；以此类推，区块 N+2 中可以包含该区块 N+2 的前一个区块的哈希值（即图 3 所示的区块 N+1 的哈希）和对第一服务器节点进行又一固件更新操作后所得到的固件版本更新记录 3。应当理解，图 3 所示的区块 N、区

块 N+1、区块 N+2 可以被称为第一区块。这些第一区块中可以包含对第一服务器节点中的目标固件（例如，固件 K）的固件版本信息进行更新操作后的所得到的子固件版本更新记录。比如，区块 N 中可以包含该固件 K 的子固件版本更新记录，区块 N+1 中可以包含该固件 K 的子固件版本更新记录，区块 N+2 中可以包含该固件 K 的子固件版本更新记录。

其中，如图 3 所示，为便于理解，本申请实施例以第一区块中的一个区块（例如，区块 N+1）为例，阐述该区块 N+1 中的固件版本更新记录 2 中可以包含多项子固件版本更新记录，具体可以为图 3 所示的更新记录 1、更新记录 2、更新记录 3 和更新记录 4。其中，更新记录 1 可以为在区块 N+1 中所查询到的该固件 K 的子固件版本更新记录。

其中，如图 3 所示的区块 N+1 中可以包含图 3 所示的区块头 10 和区块体 20。如图 3 所示，区块头 10 中可以包含前一区块（即图 3 所示的区块 N）的哈希值、时间戳、计算难度值、为生成区块 N+1 所设置的随机数，以及默克尔树根（应当理解，该默克尔树根可以为该区块 N+1 的哈希值，即该区块 N+1 的哈希值可以作为图 3 所示的区块 N+2 中的区块头中的父区块哈希值）。应当理解，本申请实施例可以将前一区块的哈希值称之为父区块哈希值。如图 3 所示的区块头 10 中的时间戳可以唯一标识一个区块在区块链中的位置。此外，如图 3 所示的区块体 20 中可以包含区块 N+1 生成之前、区块 N 生成之后的这段时间内针对该第一服务器节点的固件进行更新的所有历史版本更新行为信息。在本申请实施例中，每完成对第一服务器节点中的一个固件进行更新操作，则可以生成一个子固件版本更新记录（即可以生成一个更新记录，或者一个固件更新日志信息），本申请实施例可以将这段时间内的所有已完成的固件更新操作所对应的更新记录称之为图 3 所示的固件版本更新记录 2，即该固件版本更新记录 2 中可以包含多个子固件版本更新记录。具体地，如图 3 所示的更新记录 1、更新记录 2、更新记录 3 和更新记录 4 可以通过默克尔树的形式组织在一起。

应当理解，图 3 所示的默克尔树的构建过程是一个递归计算散列值（即

递归计算哈希值)的过程。以图 3 中的更新记录 1 和更新记录 2 为例,通过 SHA256 哈希计算方法可以计算得到更新记录 1 对应的哈希值 1;同样,通过 SHA256 哈希计算方法可以计算得到更新记录 2 对应的哈希值 2。进一步地,将哈希值 1 和哈希值 2 串联起来,继续做散列变换,可以得到图 3 所示的哈希值 12。以此类推,对于更新记录 3 和更新记录 4 而言,通过这样一层一层地递归计算,可以得到图 3 所示的哈希值 34,从而可以进一步将哈希值 12 和哈希值 34 串联起来进行散列变换,直到最后剩下一个根(即图 3 所示的哈希值 1234)。此时,本申请实施例可以将该最后得到的所有交易信息的哈希值作为区块 N+1 的默克尔树根。可以看到,默克尔树的可扩展性很好,不管更新记录有多少,最后都可以产生一个默克尔树以及固定长度的默克尔树根。应当理解,本申请实施例中的默克尔树的结构可以用于在后续进行固件版本追溯过程中确保版本查找的高效性。

应当理解,在本申请实施例中,可以使用混合固件来源的同一区块链网络对第一服务器节点中的固件进行版本管理控制(比如,版本发布控制和版本更新控制),即需要在版本管控过程中对服务器厂商加以区分。比如,对不同厂商发布的固件版本信息进行固件更新的过程中,可以使用不同的合约节点进行版本更新控制。例如,可以通过上述图 1 所示的合约节点 20a 对厂商 A 发布的固件版本信息进行固件更新,并可以通过上述图 1 所示的合约节点 20b 对厂商 B 发布的固件版本信息进行固件更新。此外,在本申请实施例中,还可以建立不同的区块链网络以管理不同厂商发布的固件版本信息的固件更新。这里将不对区块链网络的具体构建形式进行限制。

进一步地,为便于理解,本申请实施例以对第一服务器节点中的一个固件(例如,前述的固件 K)作为目标固件进行更新操作为例,合约节点在调用智能合约时,可以获取该第一服务器节点所在的区块链中的区块链地址,并在该区块链地址对应的区块链记录中查阅该固件 K 的历史更新情况和历史发布情况,即本申请实施例可以从所述区块链上获取

与所述第一服务器节点相关联的第一区块和第二区块，从而可以在第一区块中将携带目标固件的第一版本参数和第二版本参数的历史版本更新行为信息，作为与第一服务器节点相关联的固件版本更新记录。与此同时，合约节点还可以从所述第二区块中将该与所述第一服务器节点中的固件 K 相关联的历史版本发布行为信息，作为固件版本发布记录，以进一步执行步骤 S103，从而可以对当前由执行节点所发送的固件更新请求的合法性进行区分，以便于提高固件更新的安全性和可靠性。

其中，应当理解，所述固件版本发布记录为所述区块链上的发布节点基于共识机制所确定的，即本申请实施例可以在发布节点之间达成共识时，确定某个发布节点所发布的新的固件版本发布参数是可以被合约节点所认可的，从而可以进一步在合约节点之间达成合约共识时，将该包含新的固件版本发布参数的区块写入区块链进行存储。

为便于理解，进一步地，请参见图 4a 和图 4b，是本申请实施例提供的一种进行版本发布控制的示意图。如图 4a 所示，合约节点 1 在接收到发布节点 1（例如，可以为上述图 1 所对应实施例中的发布节点 10a）所广播的该固件 K 的新的固件版本发布参数（即固件版本参数 1）时，可以在图 4a 所示的时间窗口 T 内进行等待，即合约节点可以在上述图 1 所对应实施例中的预设时间窗口 T 内等待接收到其他发布节点所发布的相同固件版本发布参数。其中，该合约节点得到固件版本发布参数的具体过程可以参见图 4b 所示的步骤 S1 至步骤 S3。如图 4b 所示，发布节点 1 可以接收厂商（例如，厂商 A）所发布的最新版本固件文件，并可以对接收到的最新版本固件文件进行文件结构解析，从而可以根据文件内容进一步确定文件的内部版本号，并计算该内部版本号的哈希值。应当理解，这里的内部版本号指的是该厂商 A 针对第一服务器节点中的固件 K 发布的最新固件版本信息，该固件 K 的最新固件版本信息和该最新固件版本信息的哈希值可以称之为该固件 K 的固件版本发布参数，并将携带该固件版本发布参数的固件发布请求发送到图 4b 所示的合约节点 1，以使得合约节点 1 可以执行图 4b 所示的步骤 S5。

其中,在本申请实施例中,在区块链中,为确保数据传输的完整性,发布节点 1 在将固件 K 的固件版本发布参数发送到图 4b 所示的合约节点 1 之前,还会通过发布节点 1 的私钥对该固件版本发布参数(即图 4a 所示的固件版本发布参数 1)进行签名,从而可以将签名后的签名结果以及该发布节点 1 的公钥一并给到合约节点 1,以使合约节点 1 在接收到发布节点 1 发送的固件发布请求时,可以通过该发布节点 1 的公钥对签名结果进行签名验证,以得到该固件发布请求中所携带的固件版本发布参数 1。

如图 4a 所示,合约节点 1 在得到固件版本参数 1 之后,会在时间窗口 T 内进行等待,若在该时间窗口 T 内存在其他发布节点(比如,图 4a 所示的发布节点 2)所发布的相同固件版本参数(即固件版本参数 1),则该合约节点 1 可以间接确定发布节点 1 和发布节点 2 之间达成了共识,此时,该合约节点 1 可以得到包含固件版本发布参数 1 的区块,并可以请求将该包含固件版本发布参数 1 的区块写入区块链。

其中,在本申请实施例中,合约节点 1 在将包含该固件版本发布参数 1 的区块作为新的区块(例如,区块 A)写入区块链之前,需要将该区块 A 通过物理网络同步至上述图 1 所对应实施例中的合约层 200 中的其他合约节点,如图 4a 所示,可以通过物理网络将区块 A 同步至合约节点 2、合约节点 3、...、合约节点 n,从而可以在其他合约节点中存在超过 51%的合约节点确认达成合约共识时,将包含该固件版本发布参数 1 的区块(即区块 A)写入该区块链。由此可见,通过将节点之间的共识限定在指定功能的合约节点(例如,合约节点 1)中,可以通过合约节点进行有效地版本发布控制,即可以在判定出厂商 A 发布的固件版本信息为最新版本的厂商固件信息时,将区块 A 写入区块链。此时,本申请实施例可以将该区块 A 称之为具有最大发布时间戳的第二区块。即该具有最大发布时间戳的第二区块中所包含固件版本发布记录中的版本发布行为信息可以称之为第一历史版本发布行为信息。此外,在区块链中的与该第一服务器节点的固件 K 相关联的其它第二区块所包含固件

版本发布记录中的版本发布行为信息可以称之为第二历史版本发布行为信息。

其中，本申请实施例可以将与该第一服务器节点相关联的包含固件版本发布记录的区块称之为第二区块，合约节点可以从第二区块中获取与第一服务器节点中的目标固件（即上述固件 K）相关联的历史版本发布行为信息，从而可以将获取到的历史版本发布行为信息作为相应固件的固件版本发布记录。通过将上述更新版本参数与固件版本发布记录进行比较，可以判断是否将目标固件（即固件 K）的固件版本信息更新到最新的固件版本发布参数 1 中的固件版本信息。

步骤 S103，根据所述固件版本更新记录、所述固件版本发布记录、所述更新版本参数，确定所述固件更新请求的合法性。

具体地，合约节点可以基于所述固件版本更新记录中的所述第一版本参数，对所述更新版本参数进行回滚检测；若检测到所述第一版本参数与所述更新版本参数相同，则确定存在版本回滚，确定所述固件更新请求为非法更新请求；所述非法更新请求用于指示所述执行节点不具备对所述第一服务器节点的版本信息进行固件更新的权限；若检测到所述第一版本参数与所述更新版本参数不相同，则确定不存在版本回滚，基于所述更新版本参数以及所述固件版本发布记录，确定所述固件更新请求的合法性。

其中，所述固件更新请求中还包含：所述第一服务器节点的运行版本参数；此时，合约节点基于所述更新版本参数以及所述固件版本发布记录，确定所述固件更新请求的合法性的具体步骤可以为：在与所述固件版本发布记录相关联的历史版本发布行为信息中获取第一版本发布行为信息；所述第一版本发布行为信息为所述历史版本发布行为信息中具有最大版本发布时间戳的版本发布行为信息；进一步地，若检测到所述第一版本发布行为信息中包含所述更新版本参数，则确定所述固件更新请求为合法更新请求。

其中，在本申请实施例中，为防止版本回滚，可以根据查阅到的该

固件 K 的历史版本更新情况,对上述用于对固件 K 的固件版本信息进行固件更新的更新版本参数进行回滚检测,若检测到该固件 K 的这些固件版本更新记录(例如,上述图 3 所示的更新记录 1 所对应的子固件版本更新记录)中的第一参数包含该更新版本参数中的更新版本信息(例如,固件版本号 BBBB)和更新版本哈希值(例如,哈希值),则确定存在版本回滚,从而可以确定该固件更新请求为非法更新请求。在本申请实施例中,可以获取与该目标固件(即固件 K)的所有历史版本更新行为信息,每个历史版本更新行为信息均可以被称为一个固件版本更新记录。

为便于理解,进一步地,请参见表 1,是本申请实施例提供的一种固件 K 的固件版本更新记录表。其中,应当理解,每对该固件 K 进行一次固件更新,均可以生成一个该固件 K 的固件版本更新记录。其中,固件版本更新记录 A 为执行节点第一次对该第一服务器节点中的固件 K 进行固件更新后所得到的,固件版本更新记录 B 为执行节点第二次对该第一服务器节点中的固件 K 进行固件更新后所得到的。

表 1

固件版本更新记录 A	第一版本参数 A1	固件版本信息 1	V1
		固件哈希值 1	H1
	第二版本参数 A2	固件版本信息 2	V2
		固件哈希值 2	H2
固件版本更新记录 B	第一版本参数 B1	固件版本信息 2	V2
		固件哈希值 2	H2
	第二版本参数 B2	固件版本信息 3	V3
		固件哈希值 3	H3

其中,合约节点在调用智能合约时,可以从与该第一服务器节点相关联的第一区块中查询所有与目标固件(即固件 K)相关联的固件版本更新记录,并可以按照更新的时间前后关系构建得到表 1 所示的固件版本更新记录表。如上述表 1 所示,若从所有的第一区块中查询到的固件 K 的固件版本更新记录为表 1 所示的固件版本更新记录 A 和固件版本更

新记录 B，且从与该第一服务器节点相关联的第一区块中未获取到该固件 K 的其他固件版本更新记录，则本申请实施例中的执行节点可以将固件版本更新记录 B 中的第二版本参数 B2 中的固件版本信息 3 和固件哈希值 3 作为该固件 K 当前正在运行的运行版本参数，即合约节点所接收到的固件更新请求中可以包含上述表 1 所示的第二版本参数 B2，还可以包含计划用于对该固件 K 进行固件更新的更新版本参数（例如，固件版本信息 4 和固件哈希值 4）。此时，合约节点可以在该固件版本更新记录表中拿到每个固件版本更新记录中的第一版本参数（即上述表 1 所示的第一版本参数 A1 和第一版本参数 B1），进而可以根据这两个第一版本参数对更新版本参数进行回滚检测。若这两个第一版本参数均与更新版本参数不相同，则可以确定不存在版本回滚。反之，若这两个第一版本参数中存在任何一个第一版本参数与更新版本参数相同，则确定存在版本回滚。

其中，应当理解，在判断是否相同的时候，是用更新版本参数中的更新版本信息和更新版本哈希值分别与第一版本参数的固件版本参数中的固件版本信息和固件哈希值进行比较。比如，可以将更新版本参数中的更新版本信息与第一版本参数 B1 中的固件版本信息 2 进行比较，并将更新版本参数中的更新版本哈希值与第一版本参数 B1 中的固件哈希值 2 进行比较，从而可以在所述更新版本信息（即固件版本信息 4）与表 1 中的固件版本信息 2 相同，且更新版本哈希值（即固件哈希值 4）与表 1 中的固件哈希值 2 相同时，确定存在版本回滚。

应当理解，在通过固件版本发布记录中的第一版本参数对更新版本参数进行回滚检测的时候，还可以通过遍历的方式先依次找到包含所述固件更新请求中的运行版本参数的固件版本更新记录（即上述固件版本更新记录 B），以根据固件版本更新记录 B 中第一版本参数（即上述表 1 中的第一版本参数 B1）对更新版本参数进行回滚检测，如果第一版本参数 B1 中的固件版本信息 2 和固件哈希值 2 均与更新版本参数中的固件版本参数 4 和固件版本哈希值 4 不相同，则可以继续向上一区块中的固

件版本更新记录进行搜索，直到确定与该第一服务器节点中的固件 K 相关联的所有第一区块的固件版本更新记录中均不存在与该更新版本参数相同的第一版本参数，则可以确定不存在版本回滚。

其中，所述非法更新请求用于指示所述执行节点不具备对所述第一服务器节点的版本信息进行固件更新的权限。由此可见，若合约节点在上述固件 K 的固件版本更新记录中找到相应的更新版本参数，则确定存在版本回滚，即执行节点曾使用该更新版本参数对固件 K 进行过固件更新，此时，合约节点会将这种情况下的针对固件 K 进行的固件更新操作视为非法操作，换言之，合约节点可以确定上述固件更新请求为非法更新请求。在本申请实施例中，若检测到上述表 1 中的每个第一版本参数均与所述更新版本参数不相同，则确定不存在版本回滚，即合约节点在确定当前用于对固件 K 的版本信息进行固件更新的更新版本参数不曾用于对该固件 K 进行过固件更新时，可以进一步基于所述更新版本参数以及所述固件版本发布记录，判断所述固件更新请求的合法性。

同理，若固件版本发布记录中包含所述更新版本参数，则需要进一步判断该更新版本参数中的固件版本信息是否为最新发布的固件版本发布参数中的最新发布的固件版本信息，若判断为是，则可以在两者的哈希值一致的情况下，确定通过合约检测，即确定所述固件更新请求为合法更新请求，反之，则确定为非法更新请求，此时，合约节点会将该非法的固件更新请求进行驳回处理。在本申请实施例中；若该更新版本信息并不是最新的固件版本信息，则需要从所述区块链中查找已采用所述更新版本参数进行固件更新的 M 个第二服务器节点，并可以进一步获取 M 个第二服务器节点的固件版本更新信息，若检测到所述 M 个第二服务器节点的版本更新行为信息中的第二参数均包含所述运行版本参数时，确定所述固件更新请求为合法更新请求，反之，则确定所述固件更新请求为非法更新请求。

在本申请实施例中，若需要对第一服务器节点中的目标固件进行更新的更新版本信息的版本更新参数已经存在于区块链上，但不是最新的

固件版本信息,则需要在获取到的M个第二服务器节点中,检测每个第二服务器节点是否均采用与该第一服务器节点相同的更新方式(即需要相同运行版本参数和相同更新版本参数)。若判断为是,则可以确定前述固件更新请求为合法更新请求。

在本申请实施例中,第一终端可以在接收到执行节点发送的针对第一服务器节点的固件更新请求时,调用智能合约对执行节点所发起的固件更新请求进行智能更新管控,以确保固件更新的安全性。其中,该固件更新请求中可以携带计划对该第一服务器节点的固件进行更新的更新版本参数,进一步地,合约节点可以从区块链上获取与第一服务器节点相关联的固件版本更新记录和固件版本发布记录,从而可以基于获取到的固件版本更新记录和固件版本发布记录对该固件更新请求中的更新版本参数进行合约检测,从而可以将满足合约检测结果的固件更新请求确定为合法更新请求,反之,可以将不满足合约检测结果的固件更新请求确定为非法更新请求。由此可见,合约节点对于针对该第一服务器节点的底层固件的所有更新操作,均可以通过智能合约有效地区分出合法和非法的固件更新操作,以提高固件更新的可靠性。

进一步地,请参见图5,是本申请实施例提供的另一种基于智能合约的数据处理方法的流程示意图,所述方法可以应用于充当合约节点的设备,所述方法可以包含以下步骤:

步骤S201,接收执行节点发送的针对第一服务器节点的固件更新请求。

其中,所述固件更新请求中至少包含:所述第一服务器节点的更新版本参数。

步骤S202,根据所述固件更新请求调用智能合约,基于所述智能合约从区块链上获取与所述第一服务器节点相关联的固件版本更新记录和固件版本发布记录;所述固件版本发布记录为所述区块链上的发布节点基于共识机制所确定的。

具体地,调用所述智能合约获取所述第一服务器节点在所述区块链

中的区块链地址；所述区块链地址为所述区块链根据所述第一服务器节点的公钥信息进行散列计算后所唯一确定的；基于所述区块链地址从所述区块链上获取与所述第一服务器节点相关联的第一区块和第二区块；在所述第一区块中将携带第一版本参数和第二版本参数的历史版本更新行为信息，确定为与所述第一服务器节点相关联的固件版本更新记录；所述第二版本参数为对所述第一版本参数进行固件更新后的版本参数；从所述第二区块中将与所述第一服务器节点相关联的历史版本发布行为信息，确定为固件版本发布记录。

步骤 S203，基于所述固件版本更新记录中的所述第一版本参数，对所述更新版本参数进行回滚检测。

具体地，合约节点在执行完步骤 S203 之后，可以根据回滚检测结果确定是执行步骤 S204，还是执行步骤 S205。

步骤 S204，若检测到所述第一版本参数与所述更新版本参数相同，则确定存在版本回滚，确定所述固件更新请求为非法更新请求。

其中，所述非法更新请求用于指示所述执行节点不具备对所述第一服务器节点的版本信息进行固件更新的权限；应当理解，所述合约节点在确定固件更新请求为非法更新请求之后，可以进一步跳转执行步骤 S207 至步骤 S208。

步骤 S205，若检测到所述第一版本参数与所述更新版本参数不相同，则确定不存在版本回滚，基于所述更新版本参数以及所述固件版本发布记录，确定所述固件更新请求的合法性。

具体地，所述固件更新请求中还可以包含：所述第一服务器节点的运行版本参数；在与所述固件版本发布记录相关联的历史版本发布行为信息中获取第一版本发布行为信息；所述第一版本发布行为信息为所述历史版本发布行为信息中具有最大版本发布时间戳的版本发布行为信息；若检测到所述第一版本发布行为信息中包含所述更新版本参数，则确定所述固件更新请求为合法更新请求。在本申请实施例中，若检测到所述第一版本发布行为信息不包含所述更新版本参数，且所述历史版本

发布行为信息中的第二版本发布行为信息包含所述更新版本参数，则从所述区块链中查找已采用所述更新版本参数进行固件更新的 M 个第二服务器节点；M 为大于 2 的正整数；进一步地，获取所述 M 个第二服务器节点的版本更新行为信息；在检测到所述 M 个第二服务器节点的版本更新行为信息中均包含所述运行版本参数时，确定所述固件更新请求为合法更新请求；在本申请实施例中，在检测到所述 M 个第二服务器节点的版本更新行为信息中存在不包含所述运行版本参数的版本更新行为信息时，确定所述固件更新请求为非法更新请求。

为便于理解，进一步，请参见图 6，是本申请实施例提供的一种获取版本发布行为信息的示意图。如图 6 所示的合约节点 1 可以从图 6 所示的区块链上获取前述固件 K 的版本发布行为信息；所述版本发布行为信息可以包含第一版本发布行为信息和第二版本发布行为信息。应当理解，通过将版本更新参数中的更新版本信息 1 与所有版本发布行为信息中的固件版本发布信息进行比较，可以快速判断出链上是否存在图 6 所示的更新版本信息 1。若判断存在，则可以进一步判断该更新版本信息 1 是否为最新发布固件版本发布信息。应当理解，若请求更新的版本为最新版本，则可以直接将该更新版本信息的哈希值与最新发布的固件版本发布信息的哈希值进行比较。如果两者的哈希值相同，即确定固件版本的哈希值一致，则可以通过合约检查，即可以确定上述步骤 S201 中由执行节点所发送的固件更新请求为合法更新请求。在本申请实施例中，如果两者的哈希值不同，则确定未能通过合约检测，从而可以进一步跳转执行下述步骤 S207-步骤 S208。

其中，所述第一版本发布行为信息为该合约节点 1 基于第一服务器节点的区块链地址从区块链上所查询到的固件 K 的最近一次的发布状态，此时，该第一版本发布行为信息可以为所述固件 K 的历史版本发布行为信息中具有最大版本发布时间戳的版本发布行为信息；所以，本申请实施例可以通过该第一版本发布行为信息快速确定执行节点所请求更新的版本（即更新版本信息 1）是否为最新版本。

应当理解,如图 6 所示,在确定请求更新的版本不是最新版本时,则可以获取已采用该更新版本信息 1 进行固件更新的 M 个其他服务器节点, M 个其他服务器节点中的每个服务器节点可以称之为第二服务器节点。此时,合约节点可以从上述图 6 所示的区块链网络中基于每个第二服务器节点的区块链地址分别查询到与相应第二服务器节点相关联的固件版本更新记录,即可以获取 M 个第二服务器节点的版本更新行为信息。

应当理解,若检测到 M 个第二服务器节点的版本更新行为信息中均包含所述运行版本参数时,确定所述固件更新请求为合法更新请求。即合约节点可以在确定已经有 M 个第二服务器节点采用此固件更新方式更新过固件 K 时,也可以批准执行节点对第一服务器节点中的固件 K 采用相同的固件更新方式进行固件更新。比如, M 个第二服务器节点都曾由当前运行的第一固件版本信息更新为请求更新的第二固件版本信息时,可以使执行节点将第一服务器节点的固件 K 的固件版本信息由所述第一固件版本信息更为为第二固件版本信息。

鉴于此,在本申请实施例中,在同一企业的内网环境下,非法攻击者可能会攻陷少量机器(即第一服务器节点),并进一步对第一服务器节点中的固件进行更新操作,但通过将针对该第一服务器节点的更新操作与大多数(即 M 个)采用该固件更新方式的其他机器(即第二服务器节点)的更新行为相比,可以在确定不一致的情况下,有效地将针对该第一服务器节点的更新操作视为非法操作,从而可以快速地在大规模服务器集群中将该非法操作对应的第一服务器节点确定为异常节点。

由此可见,如图 5 所示,合约节点在执行完上述步骤 S205 之后,可以在确定固件更新请求为合法更新请求的情况下进一步执行步骤 S206;在本申请实施例中,该合约节点还可以在确定固件更新请求为非法更新请求时,进一步执行以下步骤 S207-步骤 S208。

步骤 S206,在确定所述固件更新请求为合法更新请求时,向所述执行节点返回用于对所述第一服务器节点进行固件更新的反馈信息;所述

反馈信息用于指示所述执行节点将所述第一服务器节点的固件版本信息由所述运行版本参数中的运行版本信息更新为所述更新版本参数中的更新版本信息。

步骤 S207, 在检测到所述固件更新请求为非法更新请求时, 则确认合约执行失败, 向所述执行节点退回所述非法更新请求;

步骤 S208, 在接收到所述执行节点发送的针对所述第一服务器节点的下一固件更新请求时, 将所述下一固件更新请求标识为所述非法更新请求, 所述非法更新请求用于指示所述发布节点在进行安全审计时生成针对所述执行节点的告警信息。

在本申请实施例中, 合约节点在执行完上述步骤 S206 之后, 还可以执行以下步骤: 接收所述第一服务器节点针对所述更新版本参数进行签名后的确认信息; 所述确认信息用于指示所述执行节点已完成对所述第一服务器节点的固件版本信息的更新; 通过所述第一服务器节点的公钥信息对所述确认信息进行签名验证, 并在签名验证成功时确定完成对所述智能合约的调用; 基于所述确认信息和所述固件更新请求, 对所述第一服务器节点的固件版本更新记录进行更新, 基于所述第一服务器节点的区块链地址将更新后的固件版本更新记录写入所述区块链。

为便于理解, 进一步地, 请参见图 7, 是本申请实施例提供的一种构建固件版本更新记录的示意图。如图 7 所示的固件版本更新记录 4 是由输入信息和输出信息构成。如图 7 所示, 输入信息可以为合约节点在执行智能合约的过程中所涉及到的节点 A 所提供的运行版本参数和更新版本参数。所以, 在执行节点 A 向合约节点发送固件更新请求之前, 该执行节点 A 需要提供当前计划用于对第一服务器节点的固件进行更新的更新版本参数和该第一服务器节点当前正在运行的运行版本参数, 此时, 该第一服务器节点可以将固件 K 的运行版本参数 (即图 7 所示的旧固件版本 V1 和旧固件哈希值 H1) 和更新版本参数 (即图 7 所示的新固件版本 V2 和新固件哈希值 H2) 称之为输入信息。在本申请实施例中, 执行节点 A 在得到输入信息时, 可以通过自己的私钥对该输入信息进行

签名得到第一签名信息,并将该执行节点 A 的公钥连同由该执行节点 A 签名后所得到的第一签名信息一并添加到固件更新请求中发送给合约节点。所以,合约节点在调用智能合约的过程中,可以通过该执行节点 A 的公钥对该第一签名信息进行签名验证,以从固件更新请求中得到前述输入参数(即输入信息),即可以得到固件 K 的更新版本参数和运行版本参数。

如图 7 所示,输出信息可以包含服务器节点 B(即第一服务器节点)所采集并上报的新的运行版本参数,该新的运行版本参数可以为采用更新版本参数对运行版本参数进行固件更新后所得到的,此时,运行在第一服务器节点中的新的运行版本信息可以为图 7 所示的新固件版本为 V2、新固件哈希值为 H2 的更新版本参数。如图 7 所示,第一服务器节点在将确认信息发送给合约节点之前,可以将当前采集到的新的运行版本参数作为输出信息,并通过服务器节点 B 的私钥对该输出信息进行签名,得到第二签名信息,并将第二签名信息和该服务器节点 B 的公钥一并添加到输出信息中发送到合约节点。所以,合约节点在接收到该服务器节点 B 发送的确认信息之后,可以通过该服务器节点 B 的公钥对该确认信息进行签名验证,以得到该服务器节点 B 提供的输出信息,即可以得到该服务器节点 B 所提供的新的运行版本参数。

进一步地,合约节点可以基于图 7 所示的输入信息和输出信息,确定与所述第一服务器节点相关联的目标版本更新行为信息,并可以基于所述目标版本更新行为信息对所述第一服务器节点的固件版本更新记录进行更新。即如图 7 所示,该合约节点可以根据输入信息、执行节点 A 对输入信息的签名以及执行节点 A 的公钥、输出信息、服务器节点 B 对输出信息的签名以及服务器节点 B 的公钥,生成图 7 所示的固件版本更新记录 4,并可以将固件版本更新记录 4 写入图 7 所示的区块链 20;在本申请实施例中,图 7 所示的区块链 20 可以与前述图 6 所对应实施例中的区块链 10 属于同一区块链网络,这里将不对其进行限制。

其中,在本申请实施例中,为便于理解,可以基于前述图 3 所示的

嵌套哈希链的结构,在区块 N+2 后增加一个包含固件版本更新记录 4 的区块 (即区块 N+3)。其中,合约节点可以计算该固件版本更新记录 4 的哈希值,若在区块 N+2 生成之后且区块 N+3 生成之前的这个时间段内,不存在对该第一服务器节点中的其他固件的更新操作,则可以将该固件版本更新记录 3 的哈希值作为区块 N+3 的默克尔树根,并将前一区块的哈希值作为区块 N+3 的区块头中的父区块哈希值,进而可以在区块 N+2 后将生成的区块 N+3 写入同一区块链。其中,在本申请实施例中,当有新的区块生成的时候,仍需要对该新生成的区块在全网进行广播,以便于其他节点可以发现该新的区块,进而可以在达成共识之后确定将该区块 N+3 写入区块 N+2 所在的区块链以进行有序存储。

为便于理解,进一步地,请参见图 8,是本申请实施例提供的一种通过发布节点进行安全审计的示意图。如图 8 所示的区块链 30 可以为将包含前述图 7 所示的固件版本更新记录 4 的区块 N+3 写入至上述图 3 所对应实施例中的嵌套哈希链之后所形成的区块链。

在本申请实施例中,通过将所有的更新结果 (即前述固件版本更新记录 4) 记录于区块链上,可以为后续安全状态审计提供了不可篡改、不易丢失的分布式数据基准,从而可以实现对于任一机器的安全状态审计。在进行安全状态审计的审计过程中,发布节点可以自动采集并分析从被审计的机器上读取到的固件数据,并在确认链上固件信息与链下固件信息不一致时,迅速对其异常进行告警,进而可以有效提升对于高级威胁入侵手段的感知能力,并可以显著增强了企业内网的安全性。

在本申请实施例中,执行节点在完成对第一服务器节点中的固件进行更新操作之后,区块链上的其他节点 (例如,采集节点) 也可以用于在链下不断地收集该第一服务器节点中的固件信息,收集频率可以为每次重新启动该第一服务器节点时,启动对该第一服务器节点的固件信息的收集;在本申请实施例中,也可以定时或者周期性地启动对该第一服务器节点的固件信息的收集。其中,所述收集方式不限于使用自动采集和分析工具 (例如,统一的可扩展固件接口工具等)。发布节点可以对

位于区块链上的任一服务器节点的安全状态进行安全审计，以通过比较这些服务器节点的链下固件信息和链上固件信息，来确定这些服务器节点的安全状态。

为便于理解，本申请实施例可以以这些服务器节点中的一个服务器节点（即上述图 1 所对应实施例中的服务器节点 40a 作为第一服务器节点）为例，以阐述发布节点对该服务器节点 40a（即第一服务器节点）进行安全审计的具体实现过程。其中，在本申请实施例中，发布节点也可以定时或周期性地从服务器节点 40a 中读取当前运行在该服务器节点 40a 中的固件信息，并将当前读取到的固件信息称之为链下固件信息。

应当理解，对于采集节点而言，采集节点也可以定时或周期性的将从服务器节点 40a（即第一服务器节点）上所读取到的固件信息，回传给数据中台，然后数据中台可以将读取到的固件信息给到后台，从而可以在后台对读取到的固件信息进行筛选和过滤，以得到该第一服务器节点的链下固件信息。这样，后续发布节点需要对该第一服务器节点的安全状态进行安全审计时，可以从后台得到该第一服务器节点的链下固件信息。

此外，发布节点还可以从区块链上基于该第一服务器节点的区块链地址获取最近一次的更新状态，即可以找到对该第一服务器节点中的固件进行固件更新后的最新的固件版本更新记录（即图 8 所示的位于具有最大时间戳的区块 N+3 中的固件版本更新记录 4），从而可以将该固件版本更新记录 4 中所记载的第一服务器节点的固件信息（即链上固件信息）作为前述链下固件信息的比较基准。从而可以在确定链下固件信息和链上固件信息两者一致时，确定第一服务器节点通过本次安全审计，从而可以向第一服务器节点所属的用户或者终端返回正常值。在本申请实施例中，若确定链下固件信息和链上固件信息两者进行比较后的结果不一致，则说明该第一服务器节点的固件的完整性状态出现异常，此时将会通过移动端、WEB 端告警信息（也称之为工单告警）的方式，将告警信息返回给管理员或该第一服务器节点所归属的用户。

其中，工单信息（即告警信息）可以包括但不限于：（1）事件时间（即对第一服务器节点进行安全审计的时间）；（2）业务信息，如机器负责人、所属区域、所属业务、对应IP、对应公钥；（3）上报的固件版本与固件哈希；（4）链上记录的机器信息，如机器公钥、固件版本与固件哈希；（5）上一次更新事件，包括更新时间、更新执行者信息等。

应当理解，如果工单信息确认被入侵的危险状态，则可以启动应急响应流程，比如，可以将存在异常的机器进行网络隔离，并可以进一步进行详细安全审查。

在本申请实施例中，第一终端可以在接收到执行节点发送的针对第一服务器节点的固件更新请求时，调用智能合约对执行节点所发起的固件更新请求进行智能更新管控，以确保固件更新的安全性。其中，该固件更新请求中可以携带计划对该第一服务器节点的固件进行更新的更新版本参数，进一步地，合约节点可以从区块链上获取与第一服务器节点相关联的固件版本更新记录和固件版本发布记录，从而可以基于获取到的固件版本更新记录和固件版本发布记录对该固件更新请求中的更新版本参数进行合约检测，从而可以将满足合约检测结果的固件更新请求确定为合法更新请求，反之，可以将不满足合约检测结果的固件更新请求确定为非法更新请求。由此可见，合约节点对于针对该第一服务器节点的底层固件的所有更新操作，均可以通过智能合约有效地区分出合法和非法的固件更新操作，以提高固件更新的可靠性。

进一步地，请参见图9，是本申请实施例提供的一种基于智能合约的数据处理装置的结构示意图，该数据处理装置1可以应用在上述图1所对应实施例中的合约节点20a。如图9所示，该数据处理装置1可以包括：请求接收模块10、合约调用模块20以及合法确定模块30，进一步地，该数据处理装置1还可以包括：确认接收模块40、签名验证模块50和记录更新模块60。

请求接收模块10，用于接收执行节点发送的针对第一服务器节点的固件更新请求；所述固件更新请求中至少包含：所述第一服务器节点的

更新版本参数；

合约调用模块 20，用于根据所述固件更新请求调用智能合约，基于所述智能合约从区块链上获取与所述第一服务器节点相关联的固件版本更新记录和固件版本发布记录；所述固件版本发布记录为所述区块链上的发布节点基于共识机制所确定的；

其中，所述合约调用模块 20 包括：地址获取单元 201、区块获取单元 202、更新记录确定单元 203 以及发布记录确定单元 204；

地址获取单元 201，用于调用所述智能合约以获取所述第一服务器节点在所述区块链中的区块链地址；所述区块链地址为所述区块链根据所述第一服务器节点的公钥信息进行散列计算后所唯一确定的；

区块获取单元 202，用于基于所述区块链地址从所述区块链上获取与所述第一服务器节点相关联的第一区块和第二区块；

更新记录确定单元 203，用于在所述第一区块中将携带第一版本参数和第二版本参数的历史版本更新行为信息，确定为与所述第一服务器节点相关联的固件版本更新记录；所述第二版本参数为对所述第一版本参数进行固件更新后的版本参数；

发布记录确定单元 204，用于从所述第二区块中将与所述第一服务器节点相关联的历史版本发布行为信息，确定为固件版本发布记录。

其中，地址获取单元 201、区块获取单元 202、更新记录确定单元 203 以及发布记录确定单元 204 的具体实现方式可以参见上述图 2 所对应实施例中对步骤 S102 的描述，这里将不再继续进行赘述。

合法确定模块 30，用于根据所述固件版本更新记录、所述固件版本发布记录、所述更新版本参数，确定所述固件更新请求的合法性。

其中，所述合法确定模块 30 包括：回滚检测单元 301、非法确定单元 302 以及合法确定单元 303；

回滚检测单元 301，用于基于所述固件版本更新记录中的所述第一版本参数，对所述更新版本参数进行回滚检测；

非法确定单元 302，用于若检测到所述第一版本参数与所述更新版

本参数相同，则确定存在版本回滚，确定所述固件更新请求为非法更新请求；所述非法更新请求用于指示所述执行节点不具备对所述第一服务器节点的版本信息进行固件更新的权限；

合法确定单元 303，用于若检测到所述第一版本参数与所述更新版本参数不相同，则确定不存在版本回滚，基于所述更新版本参数以及所述固件版本发布记录，确定所述固件更新请求的合法性。

其中，所述固件更新请求中还包含：所述第一服务器节点的运行版本参数；

所述合法确定单元 303 包括：第一发布子单元 3031、第一确定子单元 3032。进一步地，所述合法确定单元还包括：节点获取子单元 3033、行为获取子单元 3034、第二确定子单元 3035、第三确定子单元 3036、反馈子单元 3037、请求退回子单元 3038 以及非法标识子单元 3039；

第一发布子单元 3031，用于在与所述固件版本发布记录相关联的历史版本发布行为信息中获取第一版本发布行为信息；所述第一版本发布行为信息为所述历史版本发布行为信息中具有最大版本发布时间戳的版本发布行为信息；

第一确定子单元 3032，用于若检测到所述第一版本发布行为信息中包含所述更新版本参数，则确定所述固件更新请求为合法更新请求。

在本申请实施例中，节点获取子单元 3033，用于若检测到所述第一版本发布行为信息不包含所述更新版本参数，且所述历史版本发布行为信息中的第二版本发布行为信息包含所述更新版本参数，则从所述区块链中查找已采用所述更新版本参数进行固件更新的 M 个第二服务器节点；M 为大于 2 的正整数；

行为获取子单元 3034，用于获取所述 M 个第二服务器节点的版本更新行为信息；

第二确定子单元 3035，用于在检测到所述 M 个第二服务器节点的版本更新行为信息中均包含所述运行版本参数时，确定所述固件更新请求为合法更新请求；

第三确定子单元 3036, 用于在检测到所述 M 个第二服务器节点的版本更新行为信息中存在不包含所述运行版本参数的版本更新行为信息时, 确定所述固件更新请求为非法更新请求。

在本申请实施例中, 反馈子单元 3037, 用于在确定所述固件更新请求为合法更新请求时, 向所述执行节点返回用于对所述第一服务器节点进行固件更新的反馈信息; 所述反馈信息用于指示所述执行节点将所述第一服务器节点的固件版本信息由所述运行版本参数中的运行版本信息更新为所述更新版本参数中的更新版本信息。

在本申请实施例中, 请求退回子单元 3038, 用于在检测到所述固件更新请求为非法更新请求时, 则确认合约执行失败, 向所述执行节点退回所述非法更新请求;

非法标识子单元 3039, 用于在接收到所述执行节点发送的针对所述第一服务器节点的下一固件更新请求时, 将所述下一固件更新请求标识为所述非法更新请求, 所述非法更新请求用于指示所述发布节点在进行安全审计时生成针对所述执行节点的告警信息。

其中, 所述告警信息是由所述发布节点根据从所述区块链上所获取到的所述第一服务器的链上更新版本参数和从所述第一服务器节点的本地所采集到的更新后的运行版本参数进行安全审计后所得到的。

其中, 第一发布子单元 3031、第一确定子单元 3032、节点获取子单元 3033、行为获取子单元 3034、第二确定子单元 3035、第三确定子单元 3036、反馈子单元 3037、请求退回子单元 3038、非法标识子单元 3039 的具体实现方式可以参见上述图 2 所对应实施例中步骤 S103 的描述, 这里将不再继续进行赘述。

其中, 回滚检测单元 301、非法确定单元 302、合法确定单元 303 的具体实现方式可以参见上述图 2 所对应实施例中步骤 S103 的描述, 这里将不再继续进行赘述。

在本申请实施例中, 确认接收模块 40, 用于接收所述第一服务器节点针对所述更新版本参数进行签名后的确认信息; 所述确认信息用于指

示所述执行节点已完成对所述第一服务器节点的固件版本信息的更新；

签名验证模块 50，用于通过所述第一服务器节点的公钥信息对所述确认信息进行签名验证，并在签名验证成功时确定完成对所述智能合约的调用；

记录更新模块 60，用于基于所述确认信息和所述固件更新请求，对所述第一服务器节点的固件版本更新记录进行更新，基于所述第一服务器节点的区块链地址将更新后的固件版本更新记录写入所述区块链。

其中，所述记录更新模块 60 包括：第一信息确定单元 601、第二信息确定单元 602、更新行为确定单元 603 以及记录写入单元 604；

第一信息确定单元 601，用于将所述固件更新请求中的更新版本参数、运行版本参数作为用于构建与所述第一服务器节点相关联的固件版本更新记录的输入信息；

第二信息确定单元 602，用于将所述确认信息中所携带的所述更新版本参数作为用于构建与所述第一服务器节点相关联的固件版本更新记录的输出信息；

更新行为确定单元 603，用于基于所述输入信息和所述输出信息，确定与所述第一服务器节点相关联的目标版本更新行为信息，基于所述目标版本更新行为信息对所述第一服务器节点的固件版本更新记录进行更新；

记录写入单元 604，用于基于所述服务器节点的区块链地址将更新后的固件版本更新记录写入所述区块链。

其中，所述第一信息确定单元 601、第二信息确定单元 602、更新行为确定单元 603、记录写入单元 604 的具体实现方式可以参见上述图 2 所对应实施例中对固件版本更新记录的描述，这里将不再继续进行赘述。

其中，所述请求接收模块 10、合约调用模块 20、合法确定模块 30 的具体实现方式可以参见上述图 2 所对应实施例中对步骤 S101-步骤 S103 的描述，这里将不再继续进行赘述。在本申请实施例中，确认接收

模块 40、签名验证模块 50 和记录更新模块 60 的具体实现方式可以参见上述图 5 所对应实施例中对步骤 S201-步骤 S208 的描述，这里将不再继续进行赘述。

在本申请实施例中，第一终端可以在接收到执行节点发送的针对第一服务器节点的固件更新请求时，调用智能合约对执行节点所发起的固件更新请求进行智能更新管控，以确保固件更新的安全性。其中，该固件更新请求中可以携带计划对该第一服务器节点的固件进行更新的更新版本参数，进一步的，合约节点可以从区块链上获取与第一服务器节点相关联的固件版本更新记录和固件版本发布记录，从而可以基于获取到的固件版本更新记录和固件版本发布记录对该固件更新请求中的更新版本参数进行合约检测，从而可以将满足合约检测结果的固件更新请求确定为合法更新请求，反之，可以将不满足合约检测结果的固件更新请求确定为非法更新请求。由此可见，合约节点对于针对该第一服务器节点的底层固件的所有更新操作，均可以通过智能合约有效地区分出合法和非法的固件更新操作，以提高固件更新的可靠性。

进一步地，请参见图 10，是本申请实施例提供的节点设备的结构示意图。如图 10 所示，所述节点设备 1000 可以应用于上述图 1 对应实施例中的合约节点 20a，所述节点设备 1000 可以包括：处理器 1001、网络接口 1004 和存储器 1005，此外，所述节点设备 1000 还可以包括：用户接口 1003 和至少一个通信总线 1002。其中，通信总线 1002 用于实现这些组件之间的连接通信。其中，用户接口 1003 可以包括显示屏(Display)、键盘 (Keyboard)，用户接口 1003 还可以包括标准的有线接口、无线接口。网络接口 1004 在本申请实施例中可以包括标准的有线接口、无线接口（如 WI-FI 接口）。存储器 1005 可以是高速 RAM 存储器，也可以是非不稳定的存储器(non-volatile memory)，例如至少一个磁盘存储器。存储器 1005 在本申请实施例中还可以是至少一个位于远离前述处理器 1001 的存储装置。如图 10 所示，作为一种计算机存储介质的存储器 1005 中可以包括操作系统、网络通信模块、用户接口模块以及设备控制应用

程序。

该节点设备 1000 中的网络接口 1004 还可以与区块链中的其它节点设备（如应用服务器）进行连接，且用户接口 1003 还可以包括显示屏（Display）、键盘（Keyboard）。在图 10 所示的节点设备 1000 中，网络接口 1004 可提供网络通讯功能；而用户接口 1003 主要用于为用户提供输入的接口；而处理器 1001 可以用于调用存储器 1005 中存储的应用程序，以实现：

接收执行节点发送的针对第一服务器节点的固件更新请求；所述固件更新请求中至少包含：所述第一服务器节点的更新版本参数；

根据所述固件更新请求调用智能合约，基于所述智能合约从区块链上获取与所述第一服务器节点相关联的固件版本更新记录和固件版本发布记录；所述固件版本发布记录为所述区块链上的发布节点基于共识机制所确定的；

根据所述固件版本更新记录、所述固件版本发布记录、所述更新版本参数，确定所述固件更新请求的合法性。

应当理解，本申请实施例中所描述的节点设备 1000 可执行前文所对应实施例中对所述基于智能合约的数据处理方法的描述，也可执行前文所对应实施例中对所述基于智能合约的数据处理装置 1 的描述，在此不再赘述。另外，对采用相同方法的有益效果描述，也不再进行赘述。

此外，这里需要指出的是：本申请实施例还提供了一种计算机存储介质，且所述计算机存储介质中存储有前文提及的基于智能合约的数据处理装置 1 所执行的计算机程序，且所述计算机程序包括程序指令，当所述处理器执行所述程序指令时，能够执行前文所对应实施例中对所述基于智能合约的数据处理方法的描述，因此，这里将不再进行赘述。另外，对采用相同方法的有益效果描述，也不再进行赘述。对于本申请所涉及的计算机存储介质实施例中未披露的技术细节，请参照本申请方法实施例的描述。

本领域普通技术人员可以理解实现上述实施例方法中的全部或部

分流程，是可以通过计算机程序来指令相关的硬件来完成，所述的程序可存储于一计算机可读取存储介质中，该程序在执行时，可包括如上述各方法的实施例的流程。其中，所述的存储介质可为磁碟、光盘、只读存储记忆体（Read-Only Memory, ROM）或随机存储记忆体（Random Access Memory, RAM）等。

以上所揭露的仅为本申请较佳实施例而已，当然不能以此来限定本申请之权利范围，因此依本申请权利要求所作的等同变化，仍属本申请所涵盖的范围。