(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 113383330 B
(45)授权公告日 2025.05.09

(21)申请号 202080012580.X
(22)申请日 2020.01.31
(65)同一申请的已公布的文献号
申请公布号 CN 113383330 A
(43)申请公布日 2021.09.10
(30)优先权数据
19155755.2 2019.02.06 EP
(85)PCT国际申请进入国家阶段日
2021.08.04
(86)PCT国际申请的申请数据
PCT/IB2020/050789 2020.01.31
(87)PCT国际申请的公布数据
WO2020/161577 EN 2020.08.13

(73)专利权人 国际商业机器公司
地址 美国纽约阿芒克
(72)发明人 U·巴赫尔 R·宾德根 P·默简 J·弗兰克
(74)专利代理机构 北京市金杜律师事务所 11256
专利代理师 姚杰

(51)Int.Cl.
*G06F 21/12*(2006.01)
(56)对比文件
CN 109190386 A,2019.01.11
US 2017364704 A1,2017.12.21
审查员 汤婧

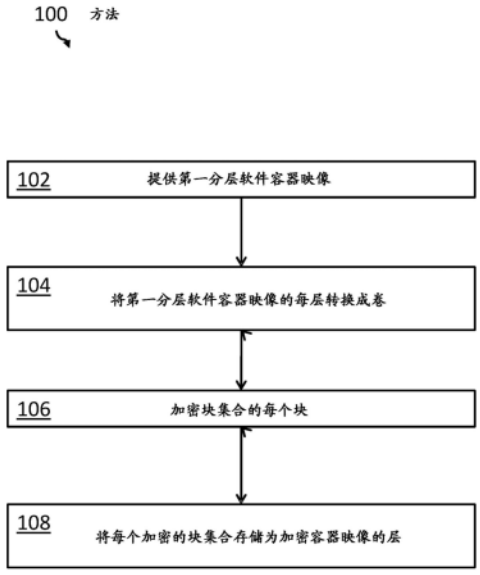
权利要求书3页 说明书14页 附图10页

(54)发明名称

# 安全容器的创建和执行

(57)摘要

可以提供一种用于创建安全软件容器的计算机实现的方法。该方法包括提供第一分层软件容器映像,将第一分层软件容器映像的每层的除了相应元数据之外的所有文件转换为卷,卷包括块的集合,其中每层包括到下一个较低层的增量差,对部分的层的块集合中的每个块进行加密,以及将每个加密的块集合连同未加密的元数据一起存储为加密的容器映像的层,未加密的元数据用于重建等于第一分层的软件容器映像的顺序的块集合的顺序,从而创建安全的加密软件容器。




CN 113383330 B

1.一种用于创建安全软件容器的计算机实现的方法,所述方法包括:

提供第一分层软件容器映像,

将所述第一分层软件容器映像的每层的除了相应元数据之外的所有文件转换成卷,所述卷包括块集合,每层包括内容文件、亲代文件和元数据文件,并且每层包括到下一个较低层的增量差,

对部分的所述层的所述块集合中的每个块进行加密,

将每个加密的所述块的集合连同未加密的元数据一起存储为加密的容器映像的层,所述未加密的元数据用于重建等于所述第一分层软件容器映像的顺序的所述块集合的顺序,

从而创建安全加密软件容器。

2.根据权利要求1所述的方法,其特征在于,所述存储每个加密的所述块的集合还包括:

存储所述第一分层软件容器映像的元数据。

3.根据权利要求1-2中任一项所述的方法,其中存储在所述加密容器映像中的所述层中的每一个使用瘦供给还包括瘦供给元数据。

4.根据权利要求1-2中任一项所述的方法,其中所述加密的容器映像的所述层的每个文件的名称是所述文件的内容的散列值。

5.根据权利要求1-2中任一项所述的方法,还包括

提供虚拟机操作系统、开始程序和解密密钥,所述解密密钥与用于对所述块集合中的每个块进行所述加密的加密密钥相对应。

6.根据权利要求5所述的方法,还包括:

提供用于具有所述虚拟机操作系统的所述虚拟机的启动的安全容器执行环境。

7.根据权利要求6所述的方法,其中所述虚拟机的所述启动包括:

使用所述解密密钥对所述加密的容器映像的块集合进行解密。

8.根据权利要求7所述的方法,还包括:

以所述第一分层软件容器映像的层的所述顺序重建所述第一分层软件容器映像。

9.根据权利要求8所述的方法,其中所述解密的安全容器映像的顶层允许读/写访问,其中所述顶层之下的所述层允许只读访问。

10.根据权利要求5所述的方法,其中所述安全容器执行环境由安全固件保护,所述安全固件防止特权的用户和/或其他进程访问所述虚拟机,并且其中所述虚拟机操作系统、所述开始程序和所述解密密钥各自被加密。

11.根据权利要求10所述的方法,其中所述安全固件与硬件安全模块协作。

12.一种用于创建安全软件容器的安全容器系统,所述系统包括:

接收单元,适于接收第一分层软件容器映像,

转换单元,适于将所述第一分层软件容器映像的每层的除了相应元数据之外的所有文件转换成卷,所述卷包括块集合,每层包括内容文件、亲代文件和元数据文件,并且每层包括到下一个较低层的增量差,

加密模块,适于对部分的所述层的所述块集合中的每个块进行加密,

存储单元,适于将每个加密的所述块的集合连同未加密的元数据一起存储为加密的容器映像的层,所述未加密的元数据用于重建等于所述第一分层软件容器映像的顺序的所述

块集合的顺序，

从而创建安全加密软件容器。

13.根据权利要求12所述的系统，其中，所述存储单元还适于：

存储所述第一分层软件容器映像的元数据。

14.根据权利要求12至13中任一项所述的系统，其中，存储在所述加密的容器映像中的所述层中的每一个使用瘦供给还包括瘦供给元数据。

15.根据权利要求12至13中任一项所述的系统，其中所述加密的容器映像的所述层的每个文件的名称是所述文件的内容的散列值。

16.根据以上权利要求12至13中任一项所述的系统，进一步包括提供模块，该提供模块适于

提供虚拟机操作系统、开始程序和解密密钥，所述解密密钥与用于对所述块集合中的每个块进行所述加密的加密密钥相对应。

17.根据权利要求16所述的系统，其特征在于，所述提供模块还适于：

提供用于具有所述虚拟机操作系统的所述虚拟机的启动的安全容器执行环境。

18.根据权利要求17所述的系统，其中所述虚拟机的所述启动包括：

使用所述解密密钥对所述加密的容器映像的块集合进行解密。

19.根据权利要求18所述的系统，还包括：

重建单元，适于以所述第一分层软件容器映像的所述层的顺序重建所述第一分层软件容器映像。

20.根据权利要求19所述的系统，其中所述解密的安全容器映像的顶层允许读/写访问，其中所述顶层之下的所述层允许只读访问。

21.根据权利要求16所述的系统，其中所述安全容器执行环境由安全固件保护，所述安全固件防止特权的用户和/或其他进程访问所述虚拟机，并且其中所述虚拟机操作系统、所述开始程序和所述解密密钥各自被加密。

22.根据权利要求21所述的系统，还包括

硬件安全模块，适于与所述安全固件协作。

23.一种用于创建安全软件容器的计算机程序产品，所述计算机程序产品包括具有随其体现的程序指令的计算机可读存储介质，所述程序指令可由一个或多个计算系统或控制器执行以使所述一个或多个计算系统：

提供第一分层软件容器映像，

将所述第一分层软件容器映像的每层的除了相应元数据之外的所有文件转换成卷，所述卷包括块的集合，每层包括内容文件、亲代文件和元数据文件，并且每层包括到下一较低层的增量差，

对部分的所述层的所述块集合中的每个块进行加密，

将每个加密的所述块的集合连同未加密的元数据一起存储为加密的容器映像的层，所述未加密的元数据用于重建等于所述第一分层软件容器映像的顺序的所述块集合的顺序，

从而创建安全加密软件容器。

24.根据权利要求23所述的计算机程序产品，其中存储在所述加密容器映像中的所述层中的每一个使用瘦供给并且还包括瘦供给元数据。

25.根据权利要求23至24中任一项所述的计算机程序产品,其中所述加密的容器映像的所述层的每个文件的名称是所述文件的内容的散列值。

# 安全容器的创建和执行

## 技术领域

[0001] 本发明总体上涉及一种安全计算,并且更具体地涉及一种用于创建安全软件容器的计算机实现的方法。本发明进一步涉及一种用于创建安全软件容器的相关安全容器系统以及一种计算机程序产品。

## 背景技术

[0002] 目前,云计算一直是IT(信息技术)行业中最热门的话题之一。个人以及小型和中型公司以及大型企业继续将计算任务外包给操作大型云计算中心的云计算提供商。另一方面,IT行业中最大的关注点之一是数据安全。所以,在私有计算环境(即,企业计算中心)中,安全性是重要的主题,使得云计算环境中的数据和计算机安全性的相关性被推进到C级的大型企业。

[0003] 可以通过在公共网络(例如,互联网)上传输数据时和/或当将数据存储在存储设备上时对相关数据进行加密来解决一些数据安全问题(尤其是鉴于如GDPR(欧盟的通用数据保护条例)的新的政府数据安全法规)。然而,如果数据必须被处理,则数据以及程序可或多或少可由云计算中心的管理员访问。这可以被视为数据安全漏洞的开门(open door)。

[0004] 解决一方面的云计算资源的更高使用与另一方面的更高的数据安全性和数据隐私性要求的潜在冲突的一种尝试是在安全的计算平台上使用安全的虚拟机,例如,采用使用硬件安全模块(hardware security module,HSM)并允许在紧密边界中对数据以及应用和虚拟机(VM)进行加密的可信计算环境的形式。

[0005] 另一方面,虚拟机技术进步和经常使用的当前技术之一是“容器计算”,其基本上可以被视为在基于操作系统的确定范围的沙箱中的多个应用,而没有每个容器的每个虚拟机内的操作系统的完整开销。由此,通常由操作系统和相关中间件提供的许多通用服务可以在一个操作系统中的不同容器之间共享。然而,这可能具有以下结果:从安全角度来看,容器中的数据和程序可能不像其他IT资源(像虚拟机)那样被管理。

[0006] 用于软件容器的最常使用的平台之一是来自Docker公司的Docker引擎。在Linux环境中,它迅速成为封装微服务应用的标准,尽管该技术不仅限于微服务。基本上,其可用于封装货架(of-the-shelf)商业和单片应用,以提供隔离和便携性。

[0007] 在此背景下,已经出现了一系列出版物:文献US 2018/0309747 A1披露了计算机系统和方法,其中代理执行程序与安全模块同时运行,该安全模块在最初执行时从用户获得代理API密钥。这个密钥被传送至网格计算系统。当API密钥有效时通常由密码令牌生成协议从网格接收代理身份令牌,并将其存储在与代理执行程序相关联的安全数据存储中。使用代理自验证因子收集评估代理执行程序的完整性的信息。

[0008] 另一方面,从文献WO 2018/007213A1中已知一种用于安全地管理Docker映像的方法,其包括存储用密钥集加密的数据的安全存储区域。Docker映像包括安全驱动程序,并且该方法包括一系列步骤:(i)仅当提供给安全驱动程序的传入凭证与当前凭证匹配时,安全驱动程序才从受信任的存储检索该密钥集,(ii)安全驱动程序访问安全存储区域并使用密

钥集解密数据,以及(iii)安全驱动程序在Docker映像之外发送数据。

[0009] 然而,在一方面的软件容器的精致管理与另一方面的安全问题之间仍然存在间隙。由此,所提出的概念的目标是为软件容器中的数据和应用提供增加的安全性。

## 发明内容

[0010] 根据本发明的一个方面,可以提供一种用于创建安全软件容器的计算机实现的方法。该方法可以包括提供第一分层软件容器映像,以及将第一分层软件容器映像的每层的除了相应元数据之外的所有文件转换成卷。卷可包括块集合,其中每一层可包括与下一较低层的增量差。该方法可以进一步包括:对部分层的块集合中的每个块进行加密;将每个加密的块集合连同未加密的元数据一起存储为加密的容器映像的层,该未加密的元数据用于重建与第一分层软件容器映像的顺序相等的块集合的顺序。由此,可以创建安全加密的软件容器。

[0011] 根据本发明的另一方面,可提供一种用于创建安全软件容器的安全容器系统。安全容器系统可以包括接收单元,用于接收第一分层软件容器映像,转换单元,用于将第一分层软件容器映像的每层的除相应元数据外的所有文件转换为卷,卷包括块集合,其中每个层包括与下一个较低层的增量差,加密模块,用于对部分层的块集合中的每个块进行加密,存储单元用于将每个加密的块集合连同未加密的元数据一起存储为加密的容器映像的层,该未加密的元数据用于重建与第一分层软件容器映像的顺序相等的块集合的顺序。因此,可以使安全容器系统能够创建安全加密软件容器。

[0012] 所提出的用于创建安全软件容器的计算机实现的方法可以提供多个优点和技术效果:

[0013] 所提出的概念至少针对三个关键数据和应用安全要求来传递。首先,通过使用从有特权的管理员(甚至是管理程序管理员)隐藏虚拟机(包括其存储器)的内容的技术,实现了从容器主机的角度和其有特权的系统管理员,谁都不能查看和理解容器的映像的内容,即,软件容器文件。而且,当容器在存储器中执行时,可以防止对容器的访问,导致容器的机密和安全的执行。

[0014] 其次,用于存储容器映像的注册表及其任何管理员被排除在理解在容器映像内正在进行什么之外。同样,通常具有(在许多方面不可控制的)对容器映像的访问的该组人可以从理解所管理的软件容器的真实内容排除。例如,它们不能理解和看到文件或可执行代码或数据。作为第三核心优点,可以提及的是,可以保留容器映像属性(如分层、所使用的去重复方法(例如,CoW[写时复制(copy on write)]、瘦供给(thin-provision))等)和典型地处理(如软件容器的管理的典型方面)。除了对所使用的容器映像进行加密之外,容器的未改变的管理和处理可允许在不改变总体解决方案的情况下提升现有解决方案的机密性级别。

[0015] 由此,所提出的概念可以允许在正常有特权的管理员能够访问的环境中运行软件容器工作负荷,并且保持软件容器映像的内容的机密性安全。因此,还可以使用非可信计算环境(例如,其中对系统管理员不存在信任)并且仍然保证程序代码、文件和数据(包括在执行期间的其主存储器/RAM)不能被软件容器的创建者之外的任何其他人看到。

[0016] 这个概念超出了虚拟机的安全执行,可能仅出于较高间接程度的价格而增加安全

密钥,并且失去软件容器的操作优势。在虚拟机的上下文中,直到今天才必须确认在没有安全执行的情况下,不能保证软件容器的内容的机密性。所提出的概念正关闭该缺口,并且明显超出仅名称空间的保护甚至未受保护的虚拟机,以为软件容器提供隔离的工作空间。

[0017] 在下文中,将描述适用于该方法以及相关系统的本发明构思的附加实施例。

[0018] 根据该方法的一个优选实施例,存储每个加密的块集合还可以包括存储第一分层软件容器映像的元数据。元数据可以包括环境变量、命令、要使用的端口等。元数据可以是加密的或未加密的。然而,即使元数据未被加密,未授权人员也不能说关于安全软件容器的真实内容的任何事物。

[0019] 根据该方法的一个有利实施例,存储在加密的容器映像中的层中的每个层(即,作为文件)可以应用瘦供给并且还可以包括瘦供给元数据,例如,关于哪个块具有有效数据。这可导致瘦供给的优点,其中仅实际占据实际保持有效、活动数据的那些存储区域。

[0020] 根据该方法的一个有用实施例,加密的容器映像的层的每个文件的名称是文件的内容的哈希值。由此,可能不需要额外的元数据。可假定通过文件上的内容的熵(entropy),可使用此提议的方法步骤产生唯一文件名。

[0021] 根据有利实施例,该方法还可包括提供虚拟机操作系统、特别是用于安全软件容器的启动程序、和解密密钥。解密密钥可对应于用于对块集合中的每个块进行加密的加密密钥。由此,安全软件容器可被拆封装,各层可被带到原始顺序,并且可启动包括在安全软件容器中的应用。

[0022] 根据另一个有利实施例,该方法还可包括提供安全容器执行环境,安全容器执行环境能够用虚拟机操作系统启动虚拟机。以此方式,可以完成组件的堆栈以实际开始执行封装到安全软件容器中的应用。此外,安全容器执行环境也可以以几乎不可能折衷(compromise)的方式被保护。

[0023] 出于完整性的原因并且根据方法的一个优选实施例,启动虚拟机可以包括使用解密密钥对加密的容器映像的块集合进行解密。只有安全软件容器内的应用的所有者可访问密钥。因此,提供安全软件容器的解密版本和启动所需环境可以一次性自动执行。

[0024] 根据一个另外的优选实施例,该方法也可以包括以第一分层软件容器映像的层的顺序重建第一分层软件容器映像。还包括的指示未加密软件容器的层的原始顺序的元数据可以有助于该方法步骤。

[0025] 根据该方法的一个允许的实施例,解密的安全容器映像之上的层(即,重建的第一分层软件容器映像之上的层)可以允许读/写访问,其中,顶层之下的层允许只读访问。因此,根据瘦供给模型,写入操作(根据写时复制范例)可以仅在其中存储了与文件的先前版本的增量或差异的顶层中执行。从顶层通过所有其他层朝向最低层垂直地(在虚拟意义上)的视图将递送所有信息以重建对应文件的最后版本。

[0026] 根据权利要求5所述的方法的另一个先进且有利的实施例,其中,该安全容器执行环境由安全固件保护(典型地,使用硬件安全模块(HSM),即,密码卡),防止特权用户和/或其他进程访问虚拟机,并且其中虚拟机操作系统、启动程序和解密密钥也各自被加密。

[0027] 由此,根据该方法的另一有利实施例,安全固件与硬件安全模块协作。该技术可为虚拟机以及相关软件容器中的应用和数据递送最安全的执行环境。这种安全模块是执行安全固件的必要要求。在没有这样的基于硬件的设备的情况下,安全固件、管理程序、虚拟机、

或软件容器的(一个或多个)应用都不是可执行的。

[0028] 此外,实施例可以采取相关计算机程序产品的形式,该相关计算机程序产品可从计算机可用或计算机可读介质访问,该计算机可用或计算机可读介质提供用于由计算机或任何指令执行系统使用或与其结合使用的程序代码。出于本说明的目的,计算机可用或计算机可读介质可为可包含用于存储、传达、传播或传输供指令执行系统、设备或装置使用或结合指令执行系统、设备或装置使用的程序的装置的任何设备。

## 附图说明

[0029] 应注意的是,参考不同的主题描述了本发明的实施例。特别地,一些实施例参考方法类型权利要求来描述,而其他实施例参考装置类型权利要求来描述。然而,本领域的技术人员将从以上和以下说明中得出,除非另有说明,除了属于一种类型的主题的特征的任何组合之外,涉及不同主题的特征之间的任何组合,特别是方法类型权利要求的特征和装置类型权利要求的特征之间的任何组合,被视为在本文件内公开。

[0030] 以上定义的方面以及本发明的其他方面从下文将要描述的实施例的示例中是清楚的并且参考实施例的示例进行解释,但是本发明不限于此。

[0031] 将仅以举例方式并且参考以下附图来描述本发明的优选实施例:

[0032] 图1示出了用于创建安全软件容器的本发明的计算机实施的方法的实施例的框图。

[0033] 图2示出了从未加密软件容器到可存储在容器库中的加密软件容器的转换处理的框图。

[0034] 图3示出了用于执行安全软件容器的所涉及的元件的完整堆栈的实施例的框图。

[0035] 图4示出了安全软件容器中的块的瘦供给和链接。

[0036] 图5示出了用于形成安全软件容器的流程图的第一部分的实施例。

[0037] 图6示出了用于形成安全软件容器的流程图的第二部分的实施例。

[0038] 图7示出了用于执行安全软件容器的流程图的第一部分的实施例。

[0039] 图8示出了用于执行安全软件容器的流程图的第二部分的实施例。

[0040] 图9示出安全容器系统的实施例的框图。

[0041] 图10示出了包括安全容器系统的计算系统的框图。

## 具体实施方式

[0042] 在本说明书的上下文中,可以使用以下惯例、术语和/或表达式:

[0043] 术语“安全软件容器”可以表示被保护免受未授权人员的未允许访问的软件容器。由此,软件容器的内容可以被加密,使得未经授权的人员不能确定什么被存储在软件容器内。基本上,软件容器可以被视为由操作系统确定范围机制提供的沙箱,其被用于应用。这些沙箱中的应用对操作系统的其余部分或其他容器沙箱不具有可见性。由于所有应用沙箱都在相同的操作系统实例上运行,因此与直接作为操作系统下的应用来运行这些应用相比,当在容器沙箱中运行应用时创建最小开销。具体地,不应用虚拟化开销。因此,软件容器(最终与相关元数据一起)可以被视为仅在容器沙箱中执行的应用,即,将一个容器沙箱与另一个容器沙箱区分开的核心功能。在容器技术中,容器映像描述包括被呈现给容器实例

的所有文件的应用的不可变封装。应用的“蓝图”可以在容器实例之间共享,并且也可以与其他容器主机交换。容器实例是使用容器映像作为其(根)文件系统的起点的运行容器。目前可以将Docker视为引领的软件容器技术。

[0044] 还可以注意到,并非安全软件容器的所有层都必须被加密。包括公共软件组件(例如,操作系统的组件(例如,Ubuntu))的层可以保持未加密。这些组件对于多个容器可以是共同的,使得可能不要求加密,因为它们的功能性无论如何都是已知的。然而,核心应用块以及相关数据可以被视为软件容器的特征化组件。因此,这些特征化组件可以作为安全软件容器的一部分被加密。然而,出于完整性的原因,可以注意到,当然可以对软件容器的所有层进行加密。如果与容器的核心部分(即,应用)相比,还有可能用另一种加密方法(例如,仅另一密钥或者还有完全不同的加密方法)来对多个软件容器可能共用的那些部分进行加密。

[0045] 术语“第一分层软件容器映像”可以表示通过其可以定义软件容器的层的堆栈。用于不同层的存储技术可以基于瘦供给/稀疏(spares)存储。应用可以仅具有对顶层的写入访问(以及读取访问)。顶层下方的层可以是只读的。

[0046] 术语“元数据”可以表示“关于数据的数据”。这里,元数据可以包括关于软件容器的不同层的信息,特别是关于不同层的正确顺序的信息,以便在解密基于块的安全软件容器之后重建初始软件容器。

[0047] 术语“卷”在此可以表示虚拟存储设备形式的存储系统。其由数据块集合组成。如果卷是虚拟的,其全部数据由用作该虚拟卷的文件备份。当使用瘦供给机制时,可以在卷中相应地标记未使用的数据块,这可以导致服务于虚拟卷的文件的空间消耗减少。

[0048] 术语“块集合”可以表示一组存储块,例如,存储设备的级联块的连续序列。

[0049] 术语“增量差”可以表示安全软件容器的两个不同层之间的增量。该概念通常可以用于瘦供给概念。如果可以为新的和/或更新的数据的序列反映不同层的序列,则可以重建完整的信息(应用以及相关数据)。使用增量差概念使得有可能仅使用真正需要的存储容量(即,瘦供给),尽管每层的块可以虚拟地提供更多空间(高达特定层或卷的存储限制)。

[0050] 术语“部分的层”可以表示可以小于软件容器的层的总数的一定数量的层。如上所述,可以不必加密软件容器的所有层。一些较低层(包括标准软件组件(例如,操作系统的一部分))可以保持未加密。然而,在其他实施例中,这些层也可以被加密。

[0051] 术语“加密容器映像”可以表示安全软件容器的层的序列,其可以包括加密形式的第一分层软件容器映像的信息。

[0052] 术语“块集合的顺序”可以表示为了重建安全软件容器的不同层的块的序列。

[0053] 术语“安全加密软件容器”在此可以表示安全软件容器。安全软件容器包括加密层的事实可以被认为是在本文档内给定的。

[0054] 术语“瘦供给”可以表示使用虚拟化技术来给出具有比实际可用更多的物理资源的外观。如果系统总是完全备份虚拟化资源,则必须初始地提供所有容量。术语瘦供给可以应用于盘(disk)层,但是可以指用于任何资源的分配方案。例如,计算机中的真实存储器通常可以用进行虚拟化的某种形式的地址转换技术瘦供给运行任务。每个任务可以动作好像它具有分配的真实存储器。所分配的虚拟存储器的总和可被指派给可通常超过真实存储器的总和的任务。这同样适用于层和软件容器。在一些上下文中,瘦供给也可被称为“稀疏

卷”。

[0055] 术语“散列值”可以表示可以用于将任意大小的数据映射到固定大小的数据的散列函数的结果。由散列函数返回的值被称为散列值、散列码、摘要、或简单地散列。散列函数通常与散列表结合使用,散列表是计算机软件中用于快速数据查找的常见数据结构。散列函数通过检测大文件中的重复记录来加速表或数据库查找。一个这样的应用是在DNA序列中发现相似的延伸。因为散列值的计算是单向方法,所以可能无法从散列值唯一地确定原始值。因此,该技术广泛用于密码学中。

[0056] 术语“虚拟机操作系统”可以表示作为在管理程序上执行的虚拟服务器的虚拟机的组件的操作系统。不同的虚拟机操作系统可以不影响其他功能。该技术可以广泛用于类似Linux的操作系统中。然而,大型机和中型操作系统也可以使用该技术。

[0057] 术语“开始程序”可以表示能够发起安全软件容器的执行开始的程序。开始程序还可以接管解密过程的启动并对解密的块重新排序,以便重建第一分层软件容器的原始层。

[0058] 术语“解密密钥”可以表示软件组件工具,用于将加密的上下文重置为其原始上下文,从而使得其可以在没有任何解密功能的情况下被读取。所以,它可以是明文。

[0059] 术语“安全容器执行环境”可以表示可以接受安全加密软件容器(或还有安全加密软件容器映像)并且基于其运行容器实例的软件组件。它可以选择在刚被创建来运行该容器的虚拟机中运行该容器。虚拟机可以是安全的,使得安全容器执行环境的有特权的管理员(在启动虚拟机时其也变成管理程序)可能不能访问运行容器的虚拟机内部的数据。为了实现安全加密软件容器(映像)的解密,它可使用可被提供给(虚拟机)操作系统的解密密钥并在容器实例的启动期间启动程序。

[0060] 术语“硬件安全模块”(HSM)可以表示物理计算设备,物理计算设备可以保护和管理用于强认证的数字密钥并且提供密码处理。这些模块可传统地以插件卡或附接到计算机或网络服务器或直接附接到CPU的外部装置的形式出现。

[0061] 术语“Docker容器”可以表示操作系统级虚拟化(也被称为容器化)的一个实例。它可以表示操作系统特征,其中内核可以允许多个隔离的用户空间实例的存在。此类实例(称为(软件)容器、分区、虚拟环境(VE)、或监狱(jail)(FreeBSD jail或chroot jail))从在它们中运行的程序的观点看上去可以像真实计算机。在普通操作系统上运行的计算机程序可以看到该计算机的所有资源(连接的设备、文件和文件夹、网络共享、CPU功率、可量化的硬件能力)。然而,在容器内运行的程序可能仅看到容器的内容和分配给容器的装置。因此,容器可彼此隔离,类似于虚拟机彼此隔离。这里提出的概念可以有利地利用Docker容器来实现。

[0062] 在下文中,将给出附图的详细描述。附图中的所有说明都是示意性的。首先,给出了用于创建安全软件容器的本发明的计算机实现的方法的实施例的框图。之后,将描述另外的实施例以及用于创建安全软件容器的安全容器系统的实施例。

[0063] 图1示出了用于创建安全软件容器(例如,安全Docker容器)的计算机实现的方法100的实施例的框图。该方法可以包括提供(102)第一分层软件容器映像,具体地,包括“基于增量”(即,基于瘦供给)的层的映像。

[0064] 方法100包括:在104,将第一分层软件容器映像的每一层的除了相应元数据之外的所有文件进一步转换成卷(具体地,包括文件系统的卷)。卷包括块集合(具体地由存储盘

管理的块),其中,每个层包括到下一个较低层的增量差。还可以提及的是,元数据尤其是描述如何处理容器(即,不同层如何彼此相关)所需的所有那些元数据。

[0065] 另外,方法100包括加密对部分的层的块集合中的每个块(106)。这可以通过使用仅安全容器的创建者已知的加密密钥来执行。还可以注意到,并非所有层都必须被加密。具体地,对于所有软件容器而言完全相同的那些层可以不需要加密。示例可以是基本操作系统层,例如,“Ubuntu层”。

[0066] 此外,方法100包括:将每个加密的块集合以及(具体地,被表示为文件)连同用于重建与第一分层软件容器映像的顺序相等的块集合的顺序的未加密的元数据(例如,具有指向亲代(parent)层的指针)一起存储(108)为加密的容器映像的层。这然后可以导致与原始软件容器(即,第一分层软件容器映像)的层数相同数量的块设备。由此,创建安全加密的分层软件容器。

[0067] 图2示出了从未加密软件容器202到可存储在容器库中的加密软件容器的转换处理的框图200。第一分层软件容器映像202(也示出为各个层204)被转换成还包括不同层210的安全软件容器208。使用转换工具206执行转换并且由弓形箭头206a指示。软件容器202是未加密的,而安全软件容器208具有加密的层210。这些可以存储在容器储存库212中。同样在此,示出了具有层210的安全软件容器208,具有符号形式的层210的一个示例。

[0068] 图3示出了用于执行安全软件容器208的所涉及的元件的完整堆栈300的实施例的框图。基于虚拟机的安全执行平台302(或可替代地,运行虚拟机中的容器的安全容器执行环境),容器引擎304管理操作软件容器的基本需求。出于完整性的原因,示出了安全软件容器208(未明确示出),安全软件容器具有其的层210,具体地具有顶部读/写层并且采用加密形式(e)。容器引擎304调用包括管理程序308的容器运行时环境306。管理程序308然后在软件容器208的层中启动内核310(特别是操作系统内核)。箭头312可以指示具有层210的安全软件容器208从容器引擎到虚拟机314中的移动,其中安全软件容器208被解密成可执行的、解密的(d)容器层堆栈204(d)(未明确示出)的层204的解密形式。

[0069] 可以注意到,如果可以应用用于虚拟机的安全执行的技术,像使用硬件安全模块、或运行虚拟机的任何其他技术从而使得有特权的管理程序管理员(例如,根用户)可能无法访问虚拟机的数据,则虚拟机314是可信环境。

[0070] 图4示出了安全软件容器中的块的瘦供给和链接。左上侧(图4(a))上的层402、404、406、408可以是软件容器映像的层的序列。最低层402包括文件410A,而下一上层404包括文件412B。如果文件410A被更新(改变),则其作为文件414A’被写入新的层406。在这个时间点,层406可以是允许对层406进行读取并且尤其是写入访问的顶层。在408中可以识别软件容器的完整的原始内容,其基本上表示通过不同层的垂直视图,从顶层406开始,经由层414到达层402。

[0071] 在右上侧(图4(b)),示出了到稀疏块设备416、418、420、422中的映射。不同层中示出的黑匣子可表示在稀疏块设备中承载层402、404、406的数据的块,并且可用客户端密钥进行加密。因此,图4(b)的堆栈可以表示安全软件容器的数据。将所有加密的块设备文件作为文件写入相应的新容器层。这些层可以包含三个文件。它们自动生成的名称基于文件的相应内容的散列:(i)内容文件(逐块加密的稀疏块回送设备文件),(ii)亲代文件,仅包括前一层的散列(即,指向前一层的文件名的ape),(iii)元数据文件(这可以是可选的),包括

添加到相关层的容器映像元数据,像环境变量、端口设置等。该最后的文件也可以可选地被加密。

[0072] 如果所有层都被安装在彼此的顶部,则所有文件将是可见的。总和亲代防火墙描述了允许以正确的顺序将块设备文件安装在彼此的顶部的层的依赖关系图。这在图4(c)中示出。弓形箭头示出相应亲代层的散列地址的指向函数。

[0073] 图5示出了形成安全软件容器的流程图500的第一部分的实施例。首先,创建稀疏回送设备文件,502。然后,在回送设备文件中创建文件系统,504。从软件容器的层的底部开始,在创建稀疏回送设备文件"D"(508)之前,识别第一(原始)分层软件组件的映像的下一层"L"。设备映射器用于在回送设备文件的先前堆栈的顶部添加回送设备文件"D",510。因此,写入操作将进行到"D",并且所有读取操作通过层堆栈请求它,直到在所请求的偏移处找到块(比较512)。

[0074] 然后,执行回送设备文件的堆栈在挂载点处的挂载操作,514。接下来,将"L"的所有内容添加到挂载点(其将把这些改变写到"D"),516。最后但并非最不重要,卸载挂载点,518。针对所有映像层执行步骤506至518。

[0075] 图6示出了用于形成安全软件容器的图5的流程图500的第二部分600的实施例。该流程图继续,识别602稀疏回送设备用于"D",并且从最底层开始。然后,加密"D"并且创建"D"的加密版本"E",604。这针对所有映像层进行,如流程图左侧的回送箭头所示。

[0076] 然后,识别稀疏回送设备文件"D2",此处过程也从最底层开始(606)。接下来,创建"E"的散列值"H",608。在下一步骤中,再次从最底层开始,执行根据原始映像分层"L"的识别,610。在612处,在任何现有的新容器映像层之上创建新容器映像层"M"。然后,通过将"E"的内容放入被称为"H"内容的新文件中,来创建"M"中的文件(614)。并且,通过将"L"的任何元数据信息放入新文件"H"元数据中,创建"M"中的文件,616。

[0077] 接下来,在618处,确定"L"是否是最底层。如果不是这种情况("N"),则通过将之前的散列值(即,之前的"E"的之前的"H")放入被称为"H"亲代的新文件中来创建"M"中的文件(620)。如果确定618为真(情况"Y"),过程循环回到识别稀疏回送设备文件"D"的步骤606并且针对所有映像层重复此循环。

[0078] 图7示出了用于执行安全软件容器的流程图700的第一部分的实施例。首先,在702处,主机的容器引擎将所有层"E"挂载在彼此的顶部上以组装容器文件系统。在下一步骤中,主机的容器引擎向容器文件系统挂载(704)空读/写。然后,在将文件系统提供(708)给虚拟机之前,容器文件系统创建(706)"安全执行"虚拟机。容器引擎读取(710)用户提供的内核和命令(如initrd)、init和客户端密钥,以对加密的软件容器(即,其内容)进行解密。

[0079] VM启动(712)用户提供的内核/initrd("initrd"代表初始RAM磁盘,其是由Linux内核在引导过程期间使用的临时文件系统),并且启动(714)用户提供的Init进程,即,发起进程/命令。然后,该流程图继续至图8。

[0080] 图8示出了用于执行安全软件容器的流程图700(比较图7)的第二部分800的实施例。init进程基于"H"亲代文件重新创建(802)层的顺序,并且init进程利用用户提供的客户端密钥在运行中(例如,使用"dm-crypt",Linux内核的设备映射器的密码模块)解密(804)所有"H"内容文件。此外,init进程按照正确的顺序组装(806)解密的稀疏回送设备,并且在只读的所有其他层之上组装(808)读写稀疏回送设备文件。

[0081] 另外,init进程再次使用客户端提供的解密密钥将读写回送文件加密(810)为新的“H”内容文件。最后但并非最不重要,init进程加密指向最顶部的只读层的新的“H”亲代文件(812)。

[0082] 出于完整性的原因,图9示出了安全容器系统900的实施例的框图。系统900包括适于接收第一分层软件容器映像的接收单元902、转换单元904,其用于将第一分层软件容器映像中每一层中除了相应元数据之外的所有文件转换为包括块集合的卷,其中,每个层包括与下一个较低层的增量差。

[0083] 此外,系统900包括加密模块906,其用于对部分的层的块集合中的每个块进行加密,以及存储单元908,适于将每个加密的块集合连同未加密的元数据一起存储为加密的容器映像的层,未加密的元数据用于重建与第一分层软件容器映像的顺序相等的块集合的顺序。由此,创建安全加密软件容器。

[0084] 本发明的实施例可以与几乎任何类型的计算机一起实现,而不管平台适于存储和/或执行程序代码。图10作为实例示出适于执行与所提出的方法相关的程序代码的计算系统1000。

[0085] 计算系统1000仅是合适的计算机系统的一个示例,并且不旨在对在此描述的本发明的实施例的使用范围或功能提出任何限制,而不管计算机系统1000是否能够被实现和/或执行上文阐述的任何功能。在计算机系统1000中,存在可与许多其他通用或专用计算系统环境或配置一起操作的组件。可以适合于与计算机系统/服务器1000一起使用的众所周知的计算系统、环境和/或配置的示例包括,但不限于,个人计算机系统、服务器计算机系统、瘦客户机、厚客户机,手持式或膝上型设备、多处理器系统、基于微处理器的系统、机顶盒、可编程消费电子产品、网络PC、小型计算机系统、大型计算机系统和包括任何上述系统或设备的分布式云计算环境,等等。计算机系统/服务器1000可在由计算机系统1000执行的计算机系统可执行指令(例如程序模块)的一般上下文中描述。一般而言,程序模块可包括执行特定任务或实现特定抽象数据类型的例程、程序、对象、组件、逻辑、数据结构等。计算机系统/服务器1000可在其中由通过通信网络链接的远程处理设备执行任务的分布式云计算环境中实践。在分布式云计算环境中,程序模块可位于本地和远程计算机系统存储介质(包括存储器存储设备)两者中。

[0086] 如图所示,计算机系统/服务器1000以通用计算设备的形式示出。计算机系统/服务器1000的组件可以包括但不限于一个或多个处理器或处理单元1002、系统存储器1004、和将包括系统存储器1004的不同系统组件耦合到处理器1002的总线1006。总线1006表示若干类型的总线结构中的任一种总线结构中的一种或多种,包括存储器总线或存储器控制器、外围总线、加速图形端口、以及使用各种总线架构中的任一种的处理器或局部总线。作为示例而非限制,此类架构包括工业标准架构(ISA)总线、微通道架构(MCA)总线、增强型ISA(EISA)总线、视频电子标准协会(VESA)局部总线、和外围组件互连(PCI)总线。计算机系统/服务器1000通常包括各种计算机系统可读介质。这样的介质可以是可由计算机系统/服务器1000访问的任何可用介质,并且它包括易失性和非易失性介质、可移动和不可移动介质两者。

[0087] 系统存储器1004可包含易失性存储器形式的计算机系统可读介质,例如随机存取存储器(RAM)1008和/或高速缓冲存储器1010。计算机系统/服务器1000还可以包括其他可

移动/不可移动、易失性/非易失性计算机系统存储介质。仅作为示例,存储系统1012可被提供用于从不可移动、非易失性磁介质(未示出,并且通常被称为“硬盘驱动器”)读取和向其写入。虽然未示出,但是可以提供用于从可移除非易失性磁盘(例如,“软盘”)读取和向其写入的磁盘驱动器,以及用于从可移除非易失性光盘(诸如CD-ROM、DVD-ROM或其他光学介质)读取或向其写入的光盘驱动器。在这样的实例中,每一个都可以通过一个或多个数据介质接口连接到总线1006。如下面将进一步描绘和描述的,存储器1004可以包括具有被配置为执行本发明的实施例的功能的一组(例如,至少一个)程序模块的至少一个程序产品。

[0088] 具有一组(至少一个)程序模块1016的程序/实用工具,以及操作系统、一个或多个应用程序、其他程序模块、和程序数据可以,通过示例而非限制的方式,存储在存储器1004中。操作系统、一个或多个应用程序、其他程序模块、和程序数据中的每一者或其某一组合可包含联网环境的实施。如本文所述,程序模块1016通常执行本发明的实施例的功能和/或方法。

[0089] 计算机系统/服务器1000还可以与一个或多个外部设备1018通信,外部设备诸如键盘、定点设备、显示器1020等;使得用户能够与计算机系统/服务器1000交互的一个或多个设备;和/或使计算机系统/服务器1000能够与一个或多个其他计算设备通信的任何设备(例如,网卡、调制解调器等)。这样的通信可以经由输入/输出(I/O)接口1014发生。此外,计算机系统/服务器1000可以经由网络适配器1022与诸如局域网(LAN)、通用广域网(WAN)和/或公共网络(例如,互联网)之类的一个或多个网络通信。如所描绘的,网络适配器1022可以经由总线1006与计算机系统/服务器1000的其他部件通信。应当理解,尽管未示出,其他硬件和/或软件组件可以与计算机系统/服务器1000结合使用。示例包括但不限于:微代码、设备驱动器、冗余处理单元、外部磁盘驱动器阵列、RAID系统、磁带驱动器和数据归档存储系统等。

[0090] 另外,用于创建安全软件容器的安全容器系统900可以附接到总线系统1006。

[0091] 已经出于说明的目的呈现了本发明的不同实施例的描述,但并不旨在是穷尽性的或局限于所披露的实施例。在不背离所描述的实施例的范围和精神的情况下,许多修改和变化对本领域的普通技术人员而言将是显而易见的。这里使用的术语被选择以最佳地解释实施例的原理、实际应用或对市场上存在的技术的技术改进,或者使得本领域普通技术人员能够理解这里公开的实施例。

[0092] 本发明可以体现为系统、方法和/或计算机程序产品。计算机程序产品可包含上面具有计算机可读程序指令的计算机可读存储介质,计算机可读程序指令用于致使处理器执行本发明的方面。

[0093] 该介质可以是用于传播介质的电子、磁性、光学、电磁、红外或半导体系统。计算机可读介质的示例可包括半导体或固态存储器、磁带、可移动计算机磁盘、随机存取存储器(RAM)、只读存储器(ROM)、刚性磁盘和光盘。光盘的当前示例包括致密盘只读存储器(CD-ROM)、致密盘读/写(CD-R/W)、DVD和蓝光盘。

[0094] 计算机可读存储介质可以是可以保留和存储指令以供指令执行设备使用的有形设备。计算机可读存储介质可以是,例如但不限于,电子存储设备、磁存储设备、光存储设备、电磁存储设备、半导体存储设备、或前述各项的任何合适的组合。计算机可读存储介质的更具体例子的非穷举列表包括以下:便携式计算机盘、硬盘、随机存取存储器(RAM)、只读

存储器(ROM)、可擦除可编程只读存储器(EPROM或闪存),静态随机存取存储器(SRAM)、便携式致密盘只读存储器(CD-ROM)、数字通用盘(DVD)、记忆棒、软盘、具有记录在其上的指令的机械编码设备(诸如穿孔卡片)或凹槽中的凸起结构、以及上述的任意合适的组合。如本文中所使用的计算机可读存储介质不应被解释为瞬态信号本身,诸如无线电波或其他自由传播的电磁波、通过波导或其他传输介质传播的电磁波(例如,通过光纤电缆的光脉冲)、或通过导线传输的电信号。

[0095] 本文所述的计算机可读程序指令可从计算机可读存储介质下载到相应的计算/处理设备,或经由网络(例如,互联网、局域网、广域网和/或无线网络)下载到外部计算机或外部存储设备。网络可以包括铜传输电缆、光传输光纤、无线传输、路由器、防火墙、交换机、网关计算机和/或边缘服务器。每个计算/处理设备中的网络适配器卡或网络接口从网络接收计算机可读程序指令,并转发计算机可读程序指令以存储在相应计算/处理设备内的计算机可读存储介质中。

[0096] 用于执行本发明的操作的计算机可读程序指令可以是汇编指令、指令集架构(ISA)指令、机器指令、机器相关指令、微代码、固件指令、状态设置数据、或者以一种或多种编程语言的任意组合编写的源代码或目标代码,编程语言包括面向对象的编程语言,例如Smalltalk、C++等,以及常规的过程式编程语言,例如“C”编程语言或类似的编程语言。计算机可读程序指令可以完全地在用户的计算机上执行、部分地作为独立软件包在用户的计算机上执行、部分地在用户的计算机上部分在远程计算机上执行、或者完全地在远程计算机或服务器上执行。在后一种情形中,远程计算机可以通过任何类型的网络(包括局域网(LAN)或广域网(WAN))连接到用户的计算机,或者可以连接到外部计算机(例如,通过使用互联网服务提供商的互联网)。在一些实施例中,电子电路(包括例如可编程逻辑电路、现场可编程门阵列(FPGA)或可编程逻辑阵列(PLA))可以通过利用计算机可读程序指令的状态信息使电子电路个性化,来执行计算机可读程序指令,以便执行本发明的方面。

[0097] 本文中参考根据本发明的实施例的方法、设备(系统)和计算机程序产品的流程图说明和/或框图描述本发明的方面。应当理解,流程图和/或框图的每个方框以及流程图和/或框图中各方框的组合,都可以由计算机可读程序指令来实现。

[0098] 这些计算机可读程序指令可以被提供给通用计算机、专用计算机、或其他可编程数据处理装置的处理器,以产生机器,这样指令通过计算机或其他可编程数据处理装置的处理器执行,创建用于实现在流程图和/或方框图的一个或多个方框中指定的功能/动作的装置。这些计算机可读程序指令还可存储在可指导计算机、可编程数据处理装置、和/或其他设备以特定方式起作用的计算机可读存储介质中,使得具有存储在其中的指令的计算机可读存储介质包括制品,该制品包括实现流程图和/或框图中的一个或多个方框中规定的功能/动作的各方面的指令。

[0099] 计算机可读程序指令还可以加载到计算机、其他可编程数据处理装置、或另一设备上,使得在计算机、其他可编程装置、或其他设备上执行一系列操作步骤,以产生计算机实现的过程,使得在计算机、其他可编程装置、或另一设备上执行的指令,实现流程图和/或框图中的一个或多个方框中规定的功能/动作。

[0100] 附图中的流程图和/或框图图示了根据本发明的不同实施例的系统、方法和计算机程序产品的可能实现的架构、功能和操作。对此,流程图或框图中的每个方框可以代表模

块、段、或指令的一部分,其包括用于实现规定的逻辑功能的一个或多个可执行指令。在一些替代实现方式中,框中所标注的功能可以不以图中所标注的顺序发生。例如,取决于所涉及的功能,连续示出的两个框实际上可以基本上同时执行,或者这些框有时可以以相反的顺序执行。还将注意的是,框图和/或流程图中的每个框、以及框图和/或流程图中的框的组合可以由基于专用硬件的系统来实现,基于专用硬件的系统执行指定的功能或动作或执行专用硬件与计算机指令的组合。

[0101] 在此使用的术语仅用于描述具体实施例的目的并且不旨在限制本发明。如在此使用的,除非上下文另外清楚地指示,单数形式“一(a)”、“一个(an)”和“该(the)”旨在也包括复数形式。将进一步理解的是,当在本说明书中使用术语“包括(comprises)”和/或“包括(comprising)”时,其指定所陈述的特征、整体、步骤、操作、元件、和/或组件的存在,但是不排除一个或多个其他特征、整体、步骤、操作、元件、组件、和/或其组合的存在或添加。

[0102] 所附权利要求中的所有装置或步骤加上功能元件的对应结构、材料、动作和等效物旨在包括用于结合其他要求保护的元件(如具体要求保护的)来执行功能的任何结构、材料、或动作。本发明的描述是出于说明和描述的目的而呈现的,但不旨在是穷尽性的或局限于所披露的形式的本发明。在不脱离本发明的范围和精神的情况下,许多修改和变化对本领域的普通技术人员将是显而易见的。选择和描述这些实施例是为了最好地解释本发明的原理和实际应用,并且使得本领域的普通技术人员能够针对适合于所考虑的具体用途的具有不同修改的不同实施例理解本发明。

[0103] 作为以上内容的概述,可以在一系列条款中描述本发明的总体概念:

[0104] 1.根据第一条,提供了一种用于创建安全软件容器的计算机实现的方法,所述方法包括:提供第一分层软件容器映像,将所述第一分层软件容器映像的每层的除了相应元数据之外的所有文件转换成卷,所述卷包括块集合,其中,每层包括到下一个较低层的增量差,

[0105] 对部分的所述层的所述块集合中的每个块进行加密,

[0106] 将每个加密的所述块的集合连同未加密的元数据一起存储为加密的容器映像的层,所述未加密的元数据用于重建等于所述第一分层软件容器映像的顺序的所述块集合的顺序,从而创建安全加密软件容器。

[0107] 2.根据第1条所述的方法,其中所述存储每个加密的所述块的集合还包括:

[0108] 存储所述第一分层软件容器映像的元数据。

[0109] 3.根据第1或2条所述的方法,其中存储在所述加密容器映像中的所述层中的每一个使用瘦供给还包括瘦供给元数据。

[0110] 4.根据以上任意一条所述的方法,其中所述加密的容器映像的所述层的每个文件的名称是所述文件的内容的散列值。

[0111] 5.根据以上任意一条所述的方法,还包括

[0112] 提供虚拟机操作系统、开始程序和解密密钥,所述解密密钥与用于对所述块集合中的每个块进行所述加密的加密密钥相对应。

[0113] 6.根据第5条所述的方法,还包括:

[0114] 提供用于具有所述虚拟机操作系统的所述虚拟机的启动的安全容器执行环境。

[0115] 7.根据第6条所述的方法,其中所述虚拟机的所述启动包括:

[0116] 使用所述解密密钥对所述加密的容器映像的块集合进行解密。

[0117] 8.根据第7条所述的方法,还包括:

[0118] 以所述第一分层软件容器映像的层的所述顺序重建所述第一分层软件容器映像。

[0119] 9.根据第8条所述的方法,其中所述解密的安全容器映像上的层允许读/写访问,其中所述顶层之下的所述层允许只读访问。

[0120] 10.根据第5到9条所述的方法,其中所述安全容器执行环境由安全固件保护,所述安全固件防止特权的用户和/或其他进程访问所述虚拟机,并且其中所述虚拟机操作系统、所述开始程序和所述解密密钥各自被加密。

[0121] 11.根据第10条所述的方法,其中所述安全固件与硬件安全模块协作。

[0122] 12.还提供了一种用于创建安全软件容器的安全容器系统,所述系统包括:

[0123] 接收单元,适于接收第一分层软件容器映像,转换单元,适于将所述第一分层软件容器映像的每层的除了相应元数据之外的所有文件转换成卷,所述卷包括块集合,其中,每层包括到下一个较低层的增量差,

[0124] 加密模块,适于对部分的所述层的所述块集合中的每个块进行加密,

[0125] 存储单元,适于将每个加密的所述块的集合连同未加密的元数据一起存储为加密的容器映像的层,所述未加密的元数据用于重建等于所述第一分层软件容器映像的顺序的所述块集合的顺序,

[0126] 从而创建安全加密软件容器。

[0127] 13.根据第12条所述的系统,其中,所述存储单元还适于:

[0128] 存储所述第一分层软件容器映像的元数据。

[0129] 14.根据第12或13条所述的系统,其中,存储在所述加密的容器映像中的所述层中的每一个使用瘦供给还包括瘦供给元数据。

[0130] 15.根据第12至14条中任一项所述的系统,其中所述加密的容器映像的所述层的每个文件的名称是所述文件的内容的散列值。

[0131] 16.根据第12至15条中任一项所述的系统,进一步包括提供模块,该提供模块适于提供虚拟机操作系统、开始程序和解密密钥,所述解密密钥与用于对所述块集合中的每个块进行所述加密的加密密钥相对应。

[0132] 17.根据第16条所述的系统,其特征在于,所述提供模块还适于:

[0133] 提供用于具有所述虚拟机操作系统的所述虚拟机的启动的安全容器执行环境。

[0134] 18.根据第17条所述的系统,其中所述虚拟机的所述启动包括:

[0135] 使用所述解密密钥对所述加密的容器映像的块集合进行解密。

[0136] 19.根据第18条所述的系统,还包括:

[0137] 重建单元,适于以所述第一分层软件容器映像的所述层的顺序重建所述第一分层软件容器映像。

[0138] 20.根据第19条所述的系统,其中所述解密的安全容器映像之上的层允许读/写访问,其中所述顶层之下的所述层允许只读访问。

[0139] 21.根据第16条所述的系统,其中所述安全容器执行环境由安全固件保护,所述安全固件防止特权的用户和/或其他进程访问所述虚拟机,并且其中所述虚拟机操作系统、所述启动程序和所述解密密钥各自被加密。

[0140] 22.根据第21条所述的系统,还包括

[0141] 硬件安全模块,适于与所述安全固件协作。

[0142] 23.一种用于创建安全软件容器的计算机程序产品,所述计算机程序产品包括具有随其体现的程序指令的计算机可读存储介质,所述程序指令可由一个或多个计算系统或控制器执行以使所述一个或多个计算系统:

[0143] 提供第一分层软件容器映像,将所述第一分层软件容器映像的每层的除了相应元数据之外的所有文件转换成卷,所述卷包括块的集合,其中每层包括到下一较低层的增量差,

[0144] 对部分的所述层的所述块集合中的每个块进行加密,

[0145] 将每个加密的所述块的集合连同未加密的元数据一起存储为加密的容器映像的层,所述未加密的元数据用于重建等于所述第一分层软件容器映像的顺序的所述块集合的顺序,从而创建安全加密软件容器。

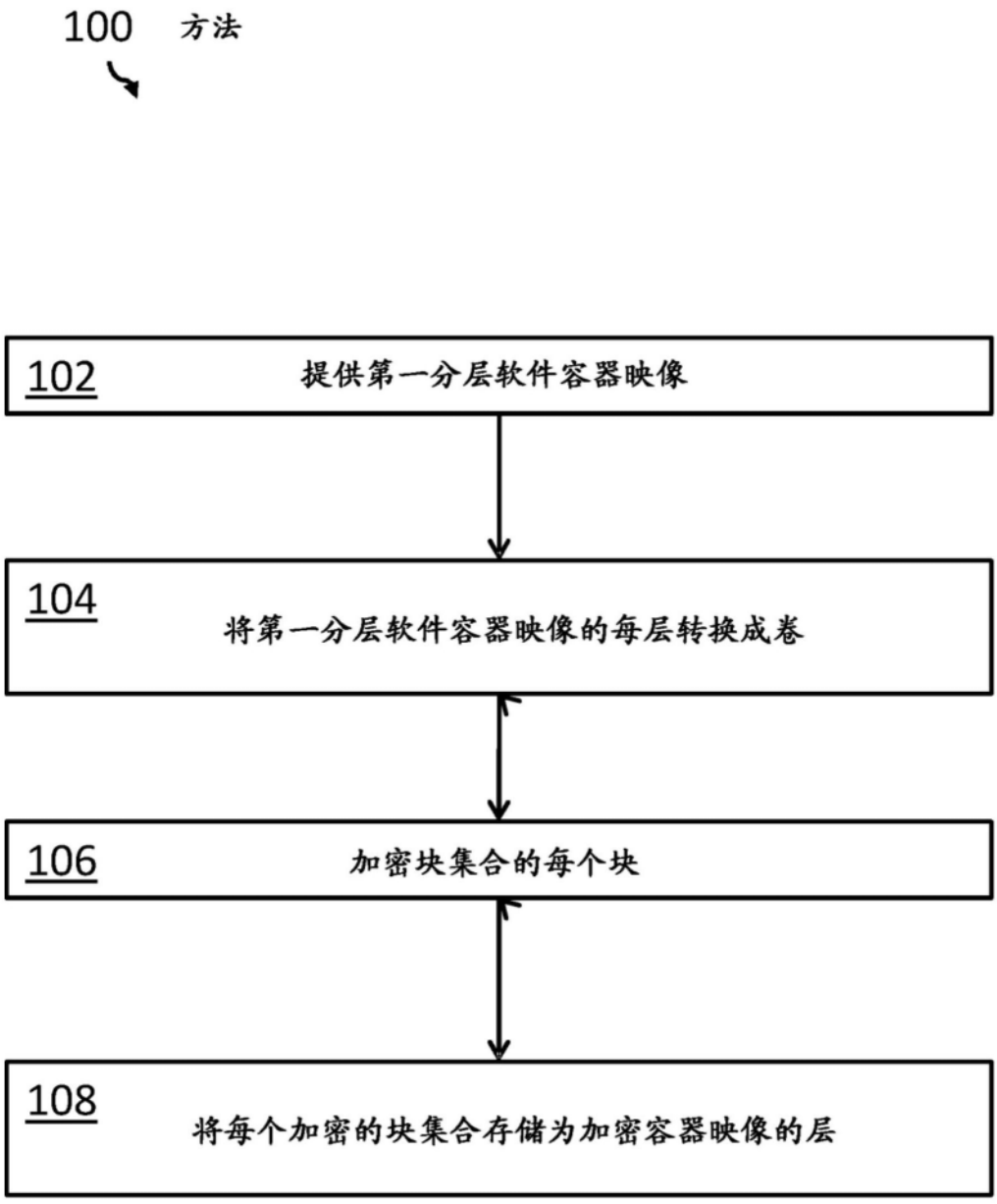

100
方法
102
提供第一分层软件容器映像
104
将第一分层软件容器映像的每层转换成卷
106
加密块集合的每个块
108
将每个加密的块集合存储为加密容器映像的层

图1

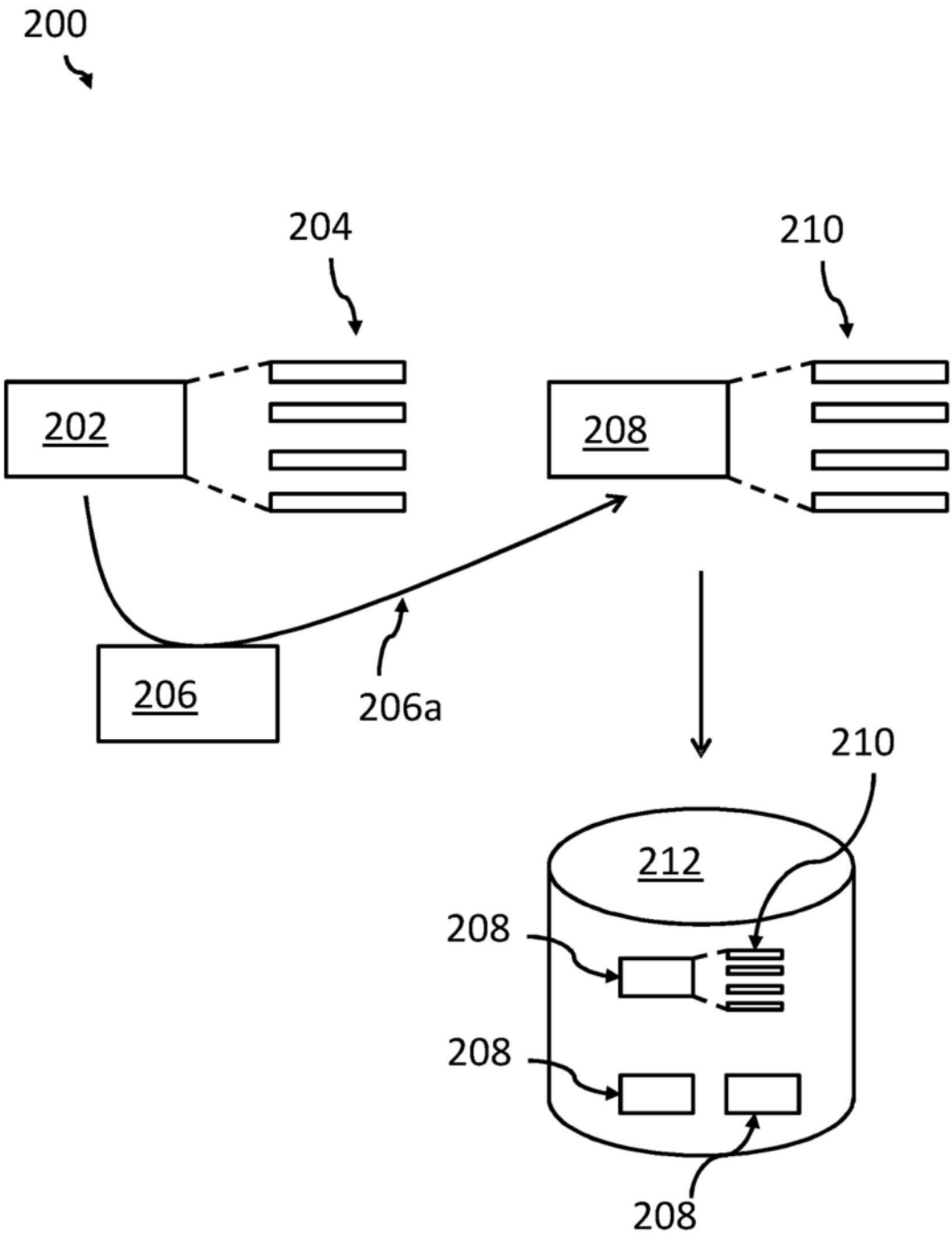



图2

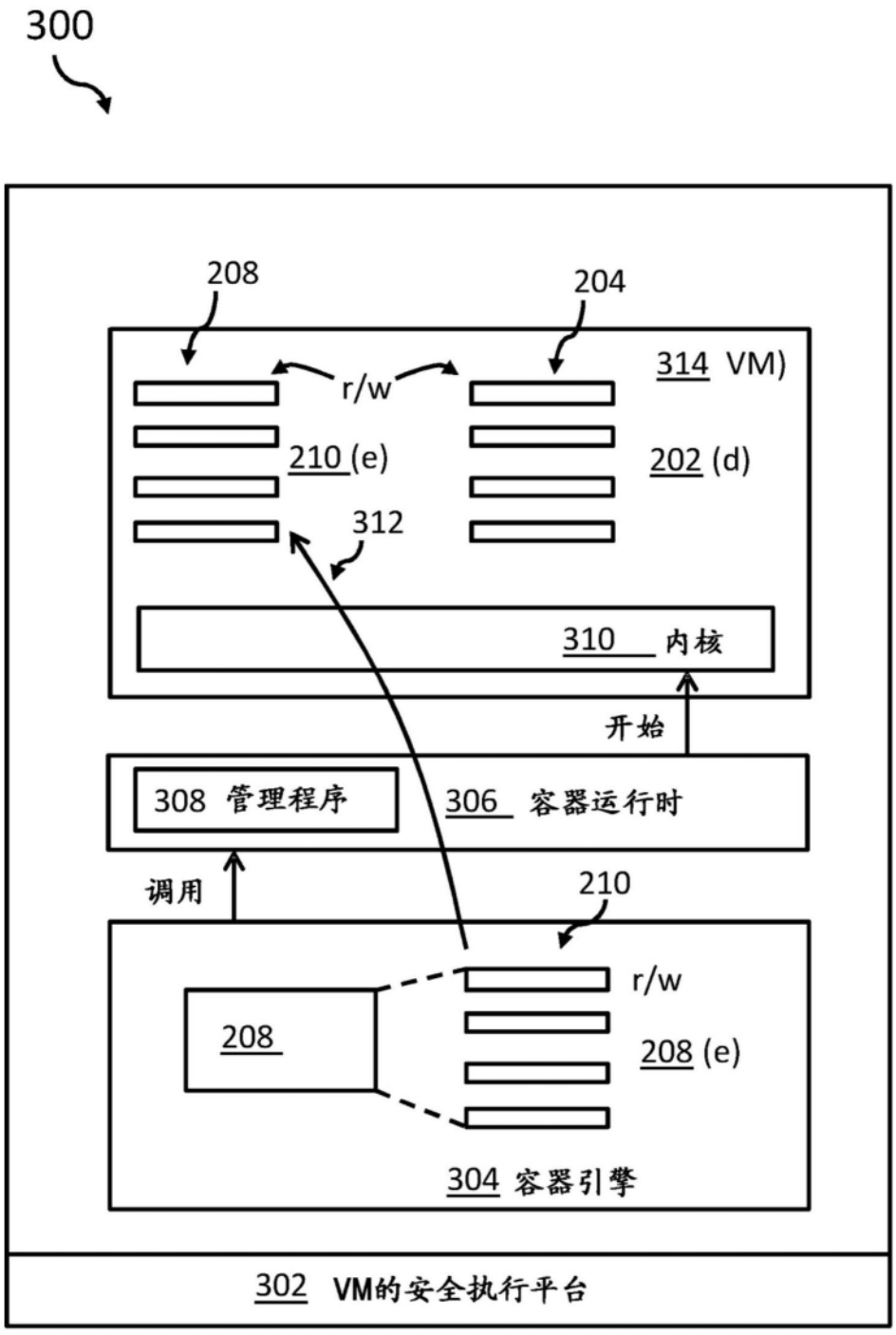
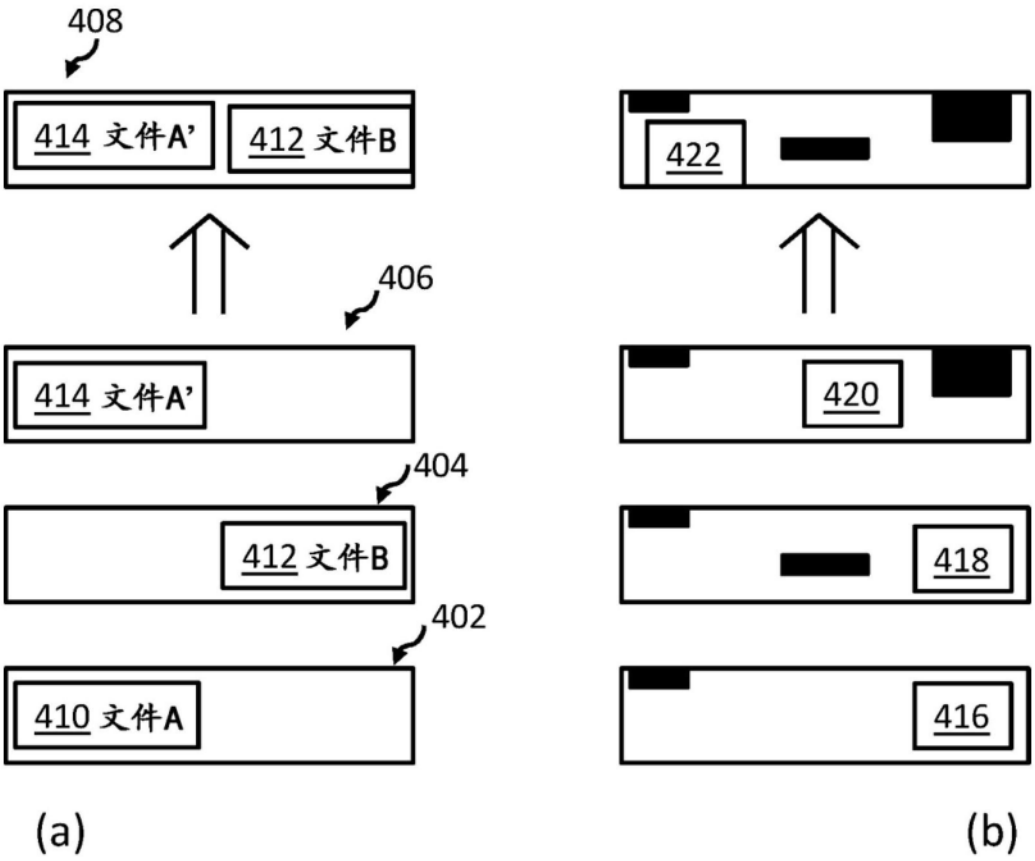

300
208
204
r/w
314 VM)
210 (e)
202 (d)
312
310 内核
开始
308 管理程序
306 容器运行时
调用
210
r/w
208
208 (e)
304 容器引擎
302 VM的安全执行平台

图3


408
414 文件A'
412 文件B
406
414 文件A'
404
412 文件B
402
410 文件A
422
420
418
416
(a)
(b)

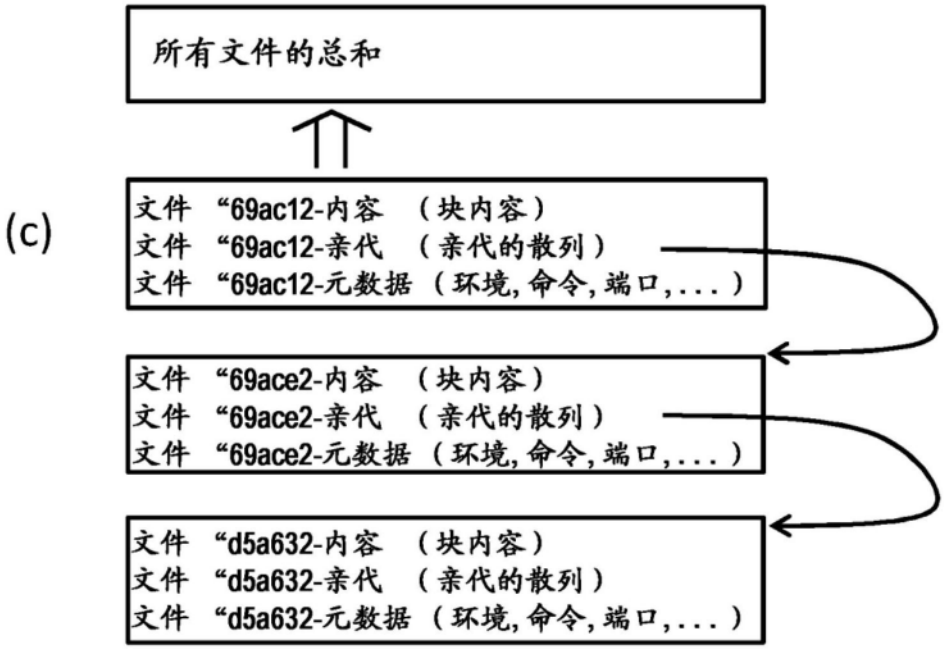

所有文件的总和
(c)
文件 “69ac12-内容 （块内容）
文件 “69ac12-亲代 （亲代的散列）
文件 “69ac12-元数据 （环境,命令,端口,...）
文件 “69ace2-内容 （块内容）
文件 “69ace2-亲代 （亲代的散列）
文件 “69ace2-元数据 （环境,命令,端口,...）
文件 “d5a632-内容 （块内容）
文件 “d5a632-亲代 （亲代的散列）
文件 “d5a632-元数据 （环境,命令,端口,...）

图4

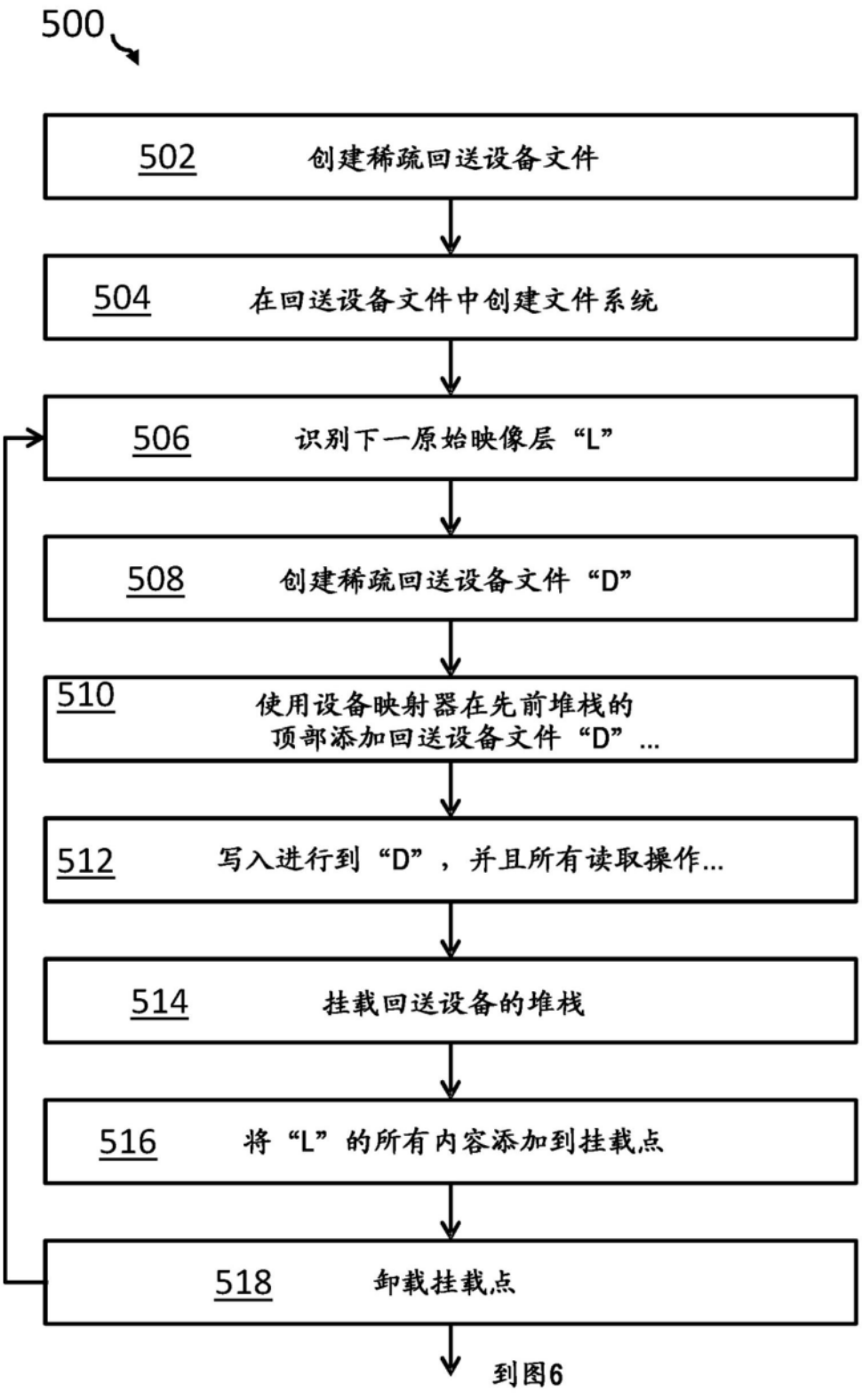

500
502
创建稀疏回送设备文件
504
在回送设备文件中创建文件系统
506
识别下一原始映像层“L”
508
创建稀疏回送设备文件“D”
510
使用设备映射器在先前堆栈的
顶部添加回送设备文件“D”...
512
写入进行到“D”，并且所有读取操作...
514
挂载回送设备的堆栈
516
将“L”的所有内容添加到挂载点
518
卸载挂载点
到图6

图5

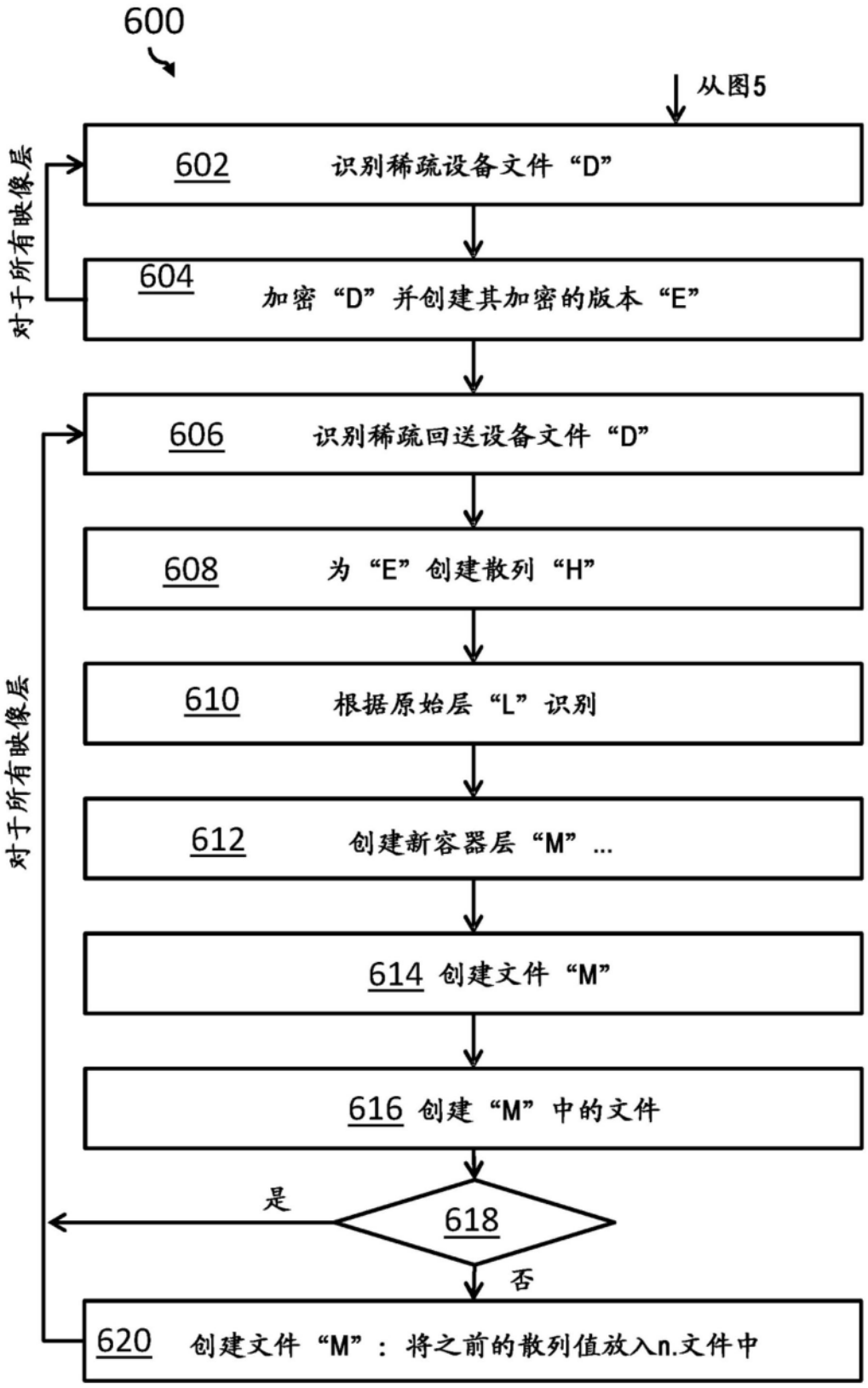

600
从图5
602
识别稀疏设备文件“D”
对于所有映像层
604
加密“D”并创建其加密的版本“E”
606
识别稀疏回送设备文件“D”
608
为“E”创建散列“H”
610
根据原始层“L”识别
对于所有映像层
612
创建新容器层“M”...
614 创建文件“M”
616 创建“M”中的文件
是
618
否
620
创建文件“M”：将之前的散列值放入n.文件中

图6

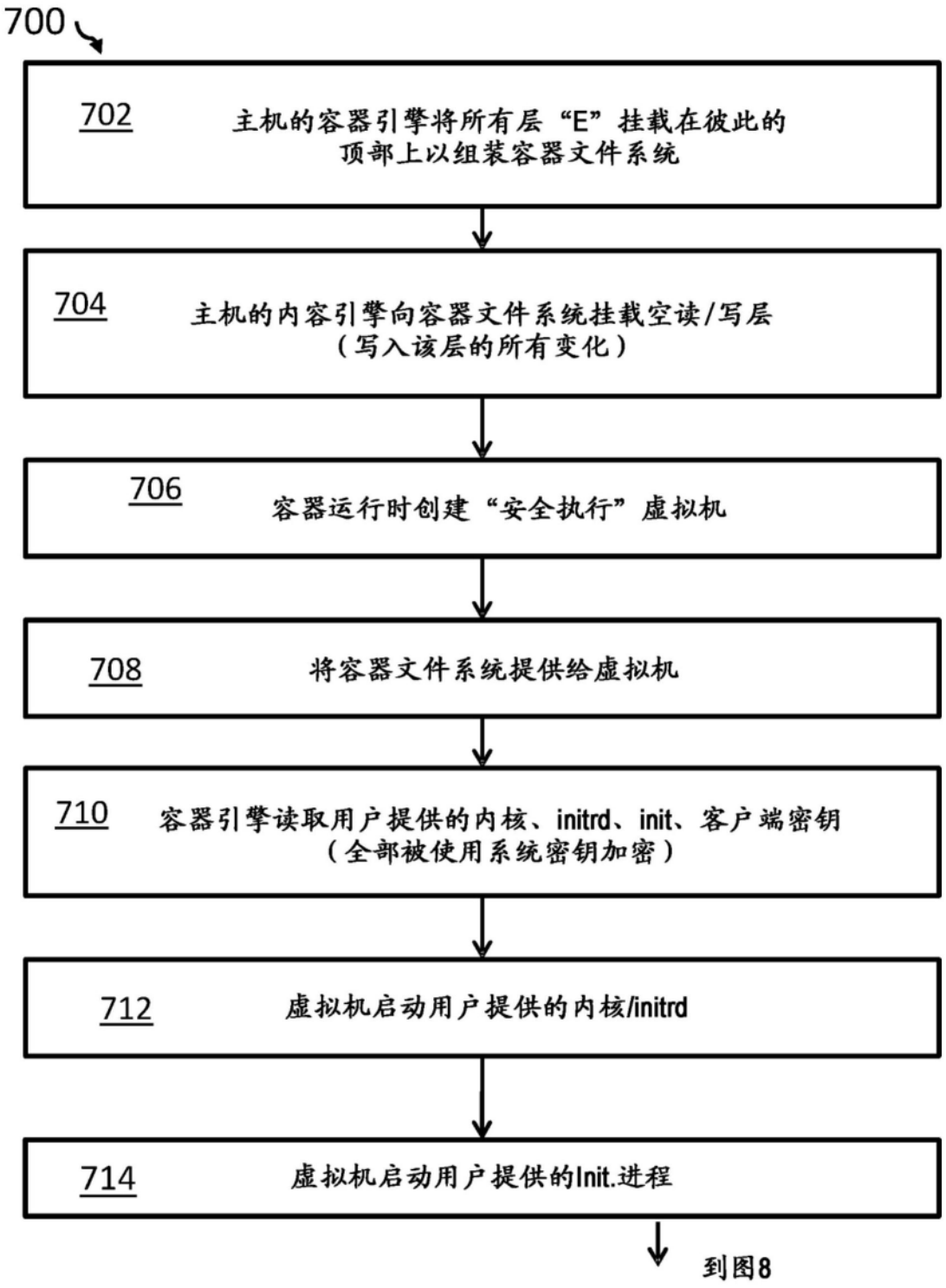

700
702
主机的容器引擎将所有层“E”挂载在彼此的顶部上以组装容器文件系统
704
主机的内容引擎向容器文件系统挂载空读/写层（写入该层的所有变化）
706
容器运行时创建“安全执行”虚拟机
708
将容器文件系统提供给虚拟机
710
容器引擎读取用户提供的内核、initrd、init、客户端密钥（全部被使用系统密钥加密）
712
虚拟机启动用户提供的内核/initrd
714
虚拟机启动用户提供的Init.进程
到图8

图7

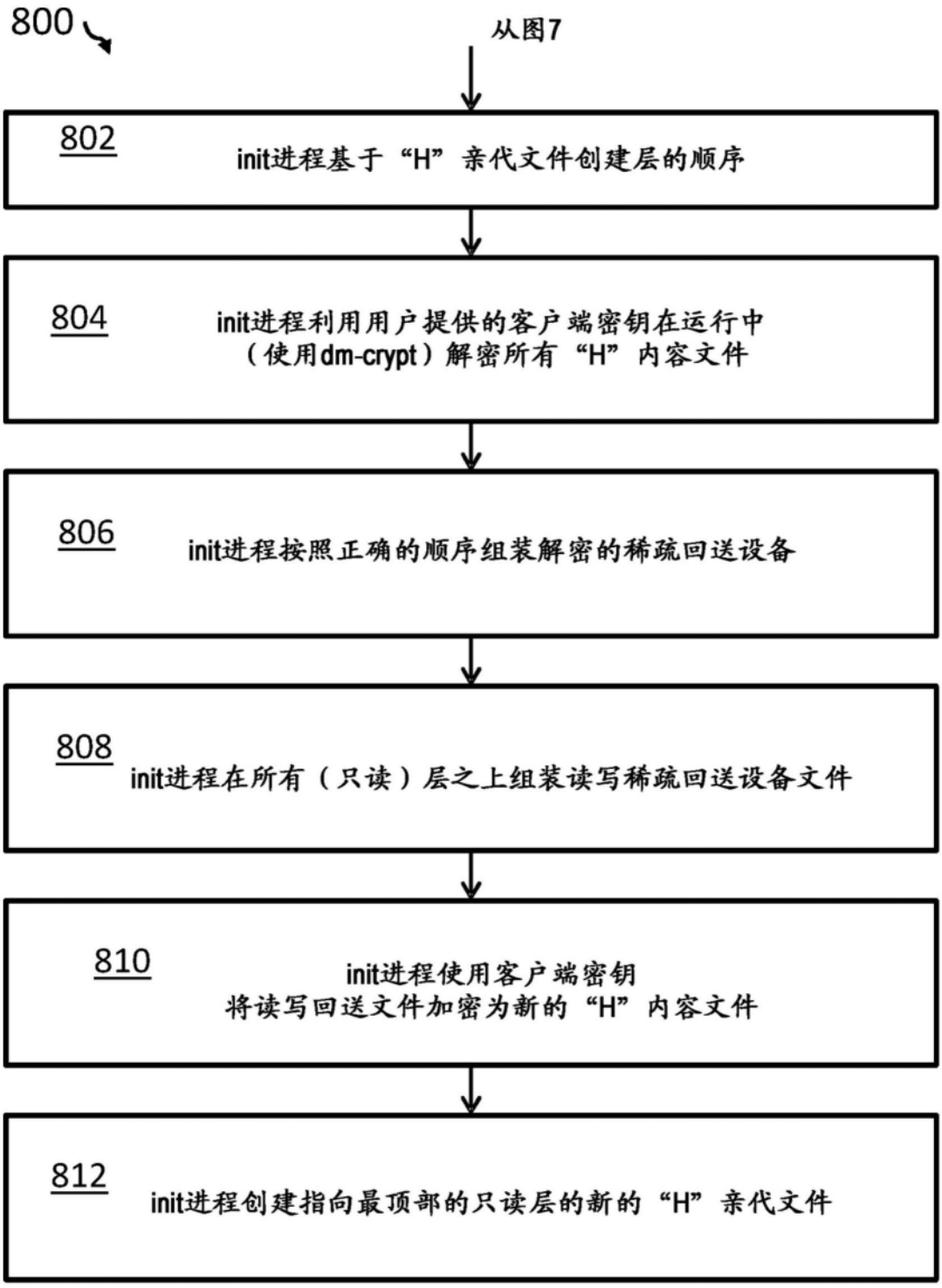

800
从图7
802
init进程基于“H”亲代文件创建层的顺序
804
init进程利用用户提供的客户端密钥在运行中
（使用dm-crypt）解密所有“H”内容文件
806
init进程按照正确的顺序组装解密的稀疏回送设备
808
init进程在所有（只读）层之上组装读写稀疏回送设备文件
810
init进程使用客户端密钥
将读写回送文件加密为新的“H”内容文件
812
init进程创建指向最顶部的只读层的新的“H”亲代文件

图8

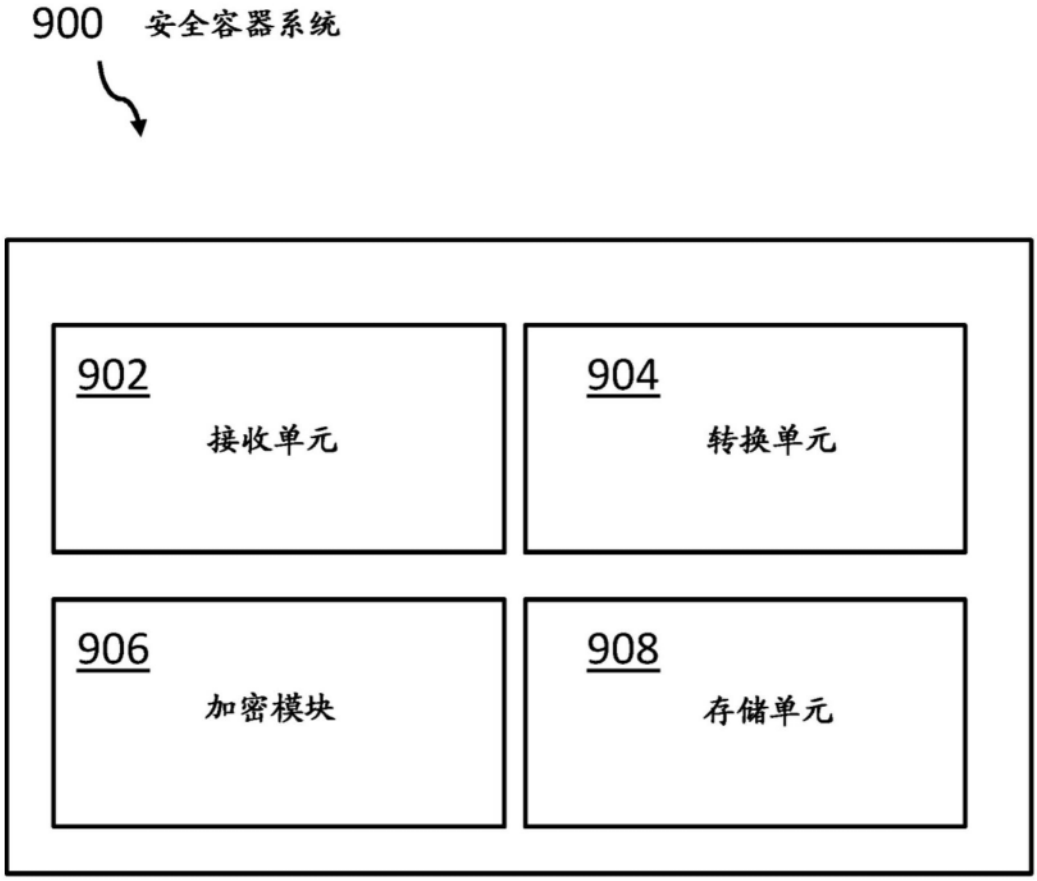



图9

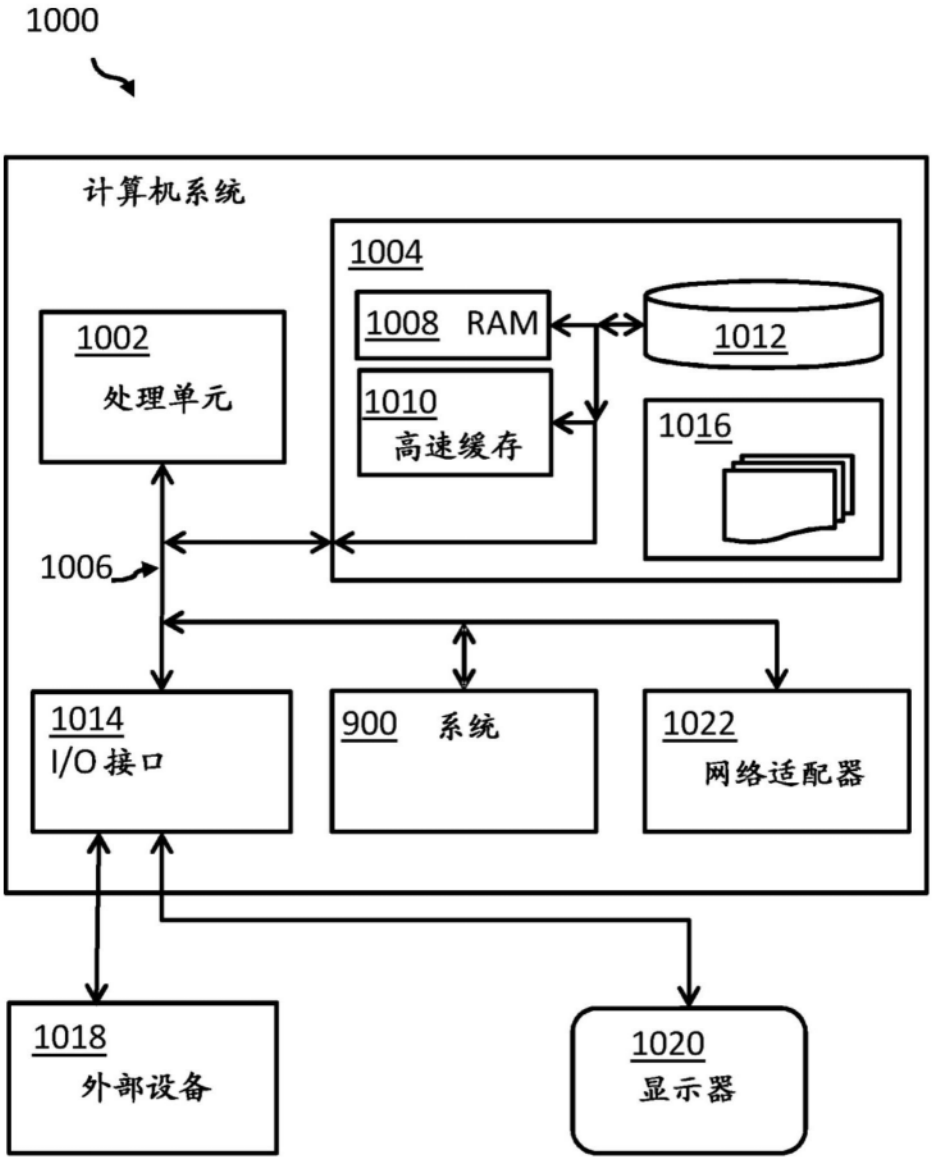

1000
计算机系统
1004
1008 RAM
1012
1002
处理单元
1010
高速缓存
1016
1006
1014
I/O 接口
900 系统
1022
网络适配器
1018
外部设备
1020
显示器

图10