(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 105308610 A
(43)申请公布日 2016.02.03

(21)申请号 201380051413.6

(22)申请日 2013.03.26

(85)PCT国际申请进入国家阶段日
2015.03.31

(86)PCT国际申请的申请数据
PCT/CA2013/000288 2013.03.26

(87)PCT国际申请的公布数据
WO2014/153635 EN 2014.10.02

(71)申请人 爱迪德技术有限公司
地址 荷兰霍夫多普

(72)发明人 G·古德斯 M·利奇

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 王洪斌 蒋骏

(51)Int.Cl.
G06F 21/57(2006.01)

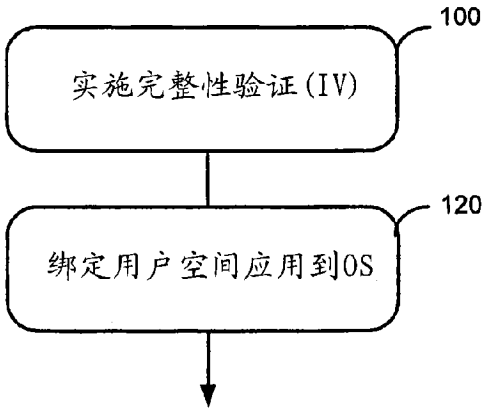
权利要求书3页 说明书9页 附图10页

(54)发明名称

用于设备上的平台和用户应用安全性的方法和系统

(57)摘要

提供了一种用于计算设备上的平台和用户应用安全性的方法和系统。该方法包括:验证计算设备上操作系统代码的完整性,以便在计算设备的操作系统中建立可信任的执行环境;并且响应于操作系统代码的完整性验证成功,将计算设备上的用户空间应用绑定到计算设备上的操作系统。



CN 105308610 A

1. 一种增强计算设备安全性的方法，包括：

验证计算设备的操作系统代码的完整性，以便在计算设备的操作系统中建立可信任的执行环境；并且

响应于操作系统代码的完整性验证成功，将计算设备的用户空间的应用绑定到计算设备的操作系统。

2. 根据权利要求 1 所述的方法，其中验证操作系统代码的完整性包括：

验证磁盘上和 / 或存储器内的操作系统映像的完整性，

所述方法包括：响应于所述磁盘上和 / 或存储器内的操作系统映像的完整性验证的成功，准予用户空间应用和操作系统的绑定。

3. 根据权利要求 2 所述的方法，其中验证操作系统代码的完整性包括：

响应于所述磁盘上和 / 或存储器内的操作系统映像的完整性验证的成功，验证存储器内操作系统映像的连续和增量的完整性，

响应于所述磁盘上和 / 或存储器内的操作系统映像的连续和增量的完整性验证的成功，准予用户空间的应用和操作系统的绑定。

4. 根据权利要求 1 所述的方法，包括：

响应于所述操作系统代码的完整性验证的成功，验证用户空间应用的完整性，以便在用户空间应用中建立可信任执行环境，并且

响应于用户空间应用的完整性验证的成功，准予用户空间应用和操作系统的绑定。

5. 根据权利要求 4 所述的方法，其中验证用户空间应用的完整性包括：

验证磁盘上的用户空间应用映像的完整性，

响应于磁盘上的用户空间应用映像的完整性验证的成功，准予用户空间应用和操作系统的绑定。

6. 根据权利要求 4 所述的方法，其中验证用户空间应用的完整性包括：

验证磁盘上的用户空间应用映像的完整性，并且

响应于磁盘上用户空间应用映像的完整性验证的成功，验证存储器内用户空间应用映像的完整性，

响应于存储器内用户空间应用映像的完整性验证的成功，准予用户空间应用和操作系统的绑定。

7. 根据权利要求 6 所述的方法，包括：

在用户空间应用绑定到操作系统之后，执行存储器内用户空间应用映像的持续和增量的完整性验证。

8. 根据权利要求 7 所述的方法，包括：

在存储器内用户空间应用映像的连续和增量的完整性验证失败时，消除用户空间应用与操作系统之间的绑定。

9. 根据权利要求 6 所述的方法，包括：

在用户空间应用绑定到操作系统之后，重新签署存储器内用户应用的映像。

10. 根据权利要求 9 所述的方法，包括：

重新签署存储器内用户应用的映像之后，验证存储器内用户应用映像的完整性。

11. 根据权利要求 4 所述的方法，其中验证用户空间应用的完整性包括：

验证磁盘上用户空间应用映像的完整性，并且

响应于磁盘上用户空间应用映像的完整性验证的成功，验证存储器内用户空间应用映像的完整性，所述方法包括：

在绑定用户空间应用和操作系统之前，签署存储器内用户空间应用的映像。

12. 根据权利要求 11 所述的方法，包括执行存储器内用户空间应用映像的连续和增量完整性的验证，其中响应于存储器内用户空间应用映像的连续和增量完整性验证的成功，准予用户空间应用和操作系统的绑定。

13. 根据权利要求 11 或 12 所述的方法，包括通过加载器将磁盘上用户应用的内容定位到存储器中，其中验证存储器中用户空间应用映像的完整性包括验证存储器中除加载器的重定位以外的用户空间应用映像的完整性，以及其中签署存储器中用户空间应用的映像包括签署存储器内包含重定位的用户空间应用的映像。

14. 根据权利要求 4 所述的方法，其中验证用户空间应用的完整性包括：

验证磁盘上用户空间应用映像的完整性，并且

响应于磁盘上用户空间应用映像的完整性验证的成功，验证存储器中用户空间应用映像的完整性，所述方法包括：

在绑定用户空间应用和操作系统之前，签署存储器内用户空间应用的映像，

如果完整性验证是成功的，则实施用户空间应用到操作系统的绑定。

15. 根据权利要求 14 所述的方法，其中：

响应于用户空间应用和操作系统的绑定，重新签署存储器中用户空间应用的映像。

16. 根据权利要求 15 所述的方法，包括：

响应于重新签署存储器中用户空间应用映像，执行存储器内用户空间应用映像的连续和增量完整性验证。

17. 根据权利要求 14-16 中任意一个所述的方法，包括通过加载器将磁盘上用户应用的内容定位到存储器中，其中验证存储器中用户空间应用映像的完整性包括验证存储器中除加载器的重定位以外的用户空间应用映像的完整性，以及其中签署存储器中用户空间应用的映像包括签署存储器内包含重定位的用户空间应用的映像。

18. 根据权利要求 1-17 中任一个所述的方法，包括：

响应于所有完整性验证的成功，向与计算设备的资源有关的用户应用授予一个或多个特权。

19. 根据权利要求 18 所述的方法，包括：

当任何完整性验证失败时，撤回分配给用户应用的任何特权。

20. 根据权利要求 1 所述的方法，其中将用户空间应用绑定到操作系统包括：

通过使用操作系统在分配给用户空间应用的存储器中恢复对应用户应用文件的原始版本的代码段和 / 或数据段，以用来执行用户应用文件原始版本的功能。

21. 根据权利要求 20 所述的方法，其中恢复代码段和 / 或数据段包括：

通过使用操作系统将所述用户应用文件的原始版本的代码段和 / 或数据段注入到存储器中。

22. 根据权利要求 20 所述的方法，其中恢复代码部分和 / 或数据部分包括：

恢复从用户应用文件的原始版本中提取的代码段和 / 或数据段；和 / 或

修复和 / 或解密代码和 / 或数据，以便恢复用户应用的原始版本的代码段和 / 或数据段。

23. 一种增强计算设备安全性的系统，包括 ：

处理器，配置为 ：

验证计算设备上的操作系统代码的完整性，以便在计算设备的操作系统中建立可信任执行环境 ；并且

响应于操作系统代码的完整性验证成功，将计算设备的用户空间应用绑定到计算设备的操作系统上。

24. 一种存储一个或多个程序的计算机可读存储介质，所述一个或多个程序包括指令，当上述指令由移动设备中的计算机处理器执行时，使得处理器执行如权利要求 1 所述的方法。

用于设备上的平台和用户应用安全性的方法和系统

技术领域

[0001] 本发明涉及一种计算机系统和方法，更特别地，涉及一种用于计算设备上的平台和用户应用安全性的系统和方法。

背景技术

[0002] 保护计算设备（诸如移动电话）上的用户空间的代码已经变得越来越重要。目前存在有多种用于这类设备上的应用代码的安全性的方法：这些方法中的一种是自主访问控制（discretionary access control，DAC）。按照惯例，操作系统（OS）使用DAC来基于用户ID限制对对象的访问，由此保护对象不受未经授权的访问。在DAC之下的对象的所有者有做出策略决策和/或指派安全性属性的能力。使用DAC，许可或拒绝例如用于执行某个用户应用代码的其他用户权限就成为可能。然而，DAC方法的根本局限性在于存在“根”用户，“根”用户是免除了由DAC强加在所有其他用户ID上的限制，因而恶意用户可以通过仅仅获取根特权（也被称为“使设备生根”）来完全绕过DAC。

[0003] SE Linux是另一种方法，其通过强加强制访问控制(Mandatory Access Control，MAC)来保护Linux平台，顾名思义，MAC不是基于用户ID的自主性的。SE Linux方法提供了一种基于策略的安全性机制，其中对文件、驱动程序和其他系统资源的每个可能取得特权的访问，均基于不仅仅用户ID而且基于发起请求的执行进程来被准予或拒绝。使用SE Linux的根本难点（和其没有更广泛使用的原因）在于：所述安全策略数据是非常复杂的（几乎都是这样），因为其必须描述系统上的每个资源以及对那些资源可能或者可能不被允许访问的每个进程的完整矩阵。此外，每次在设备中安装新软件时都需要对策略进行更新。事实上，这在实际操作中是难管理的，因此很大程度上阻碍了SE Linux在真实世界的采用。

[0004] 因此，需要提供一种新的系统和方法，用于提高计算设备的安全性。

发明内容

[0005] 本发明的目的是提供一种能够消除或减轻现有系统的至少一个缺点 的方法和系统。

[0006] 根据本公开的一个方面，提供了一种增强计算设备的安全性的方法，其包括：验证计算设备的操作系统代码的完整性以便在计算设备的操作系统中建立可信任执行环境；以及响应于操作系统代码的完整性验证成功，将计算设备的用户空间的应用绑定到计算设备的操作系统。

[0007] 根据本公开的另一个方面，提供了一种存储一个或多个程序的计算机可读存储介质，所述一个或多个程序包括指令，当上述指令由计算设备中的计算机处理器执行时，使得所述处理器执行增强计算设备的安全性的方法。

[0008] 根据本公开的另一个方面，提供了一种增强计算设备的安全性的系统，包括：处理器，配置为：验证计算设备的操作系统代码的完整性以便在计算设备的操作系统中建立可

信任的执行环境；以及响应于所述操作系统代码的完整性验证成功，将计算设备的用户空间的应用绑定到计算设备的操作系统。

附图说明

[0009] 从其中参照附图的下列描述，本发明的这些及其他特征将变得更加显然，其中：

图 1A-1H 是图示增强计算设备的安全性的示例的流程图；

图 2A-2D 是图示用于安全平台的示例性构建和供应过程的示意图；

图 3A 是图示在安全平台中的系统启动过程的一个示例的示意图，系统启动过程包括创建内核和内核代理；

图 3B 是图示系统启动过程的另一个示意图，系统启动过程包括在内核空间建立可信任执行环境 (TEE)；

图 4A 是图示在安全平台中的用户应用启动过程的一个示例的示意图，用户应用启动过程包括创建用户应用过程；

图 4B 是图示所述用户应用启动过程的另一个示意图，用户应用启动过程包括用户应用进程的完整性验证；

图 4C 是图示所述用户应用启动过程的又一个示意图，用户应用启动过程包括将用户应用进程绑定到内核空间；以及

图 5 是图示具有开源 OS 的移动设备的一个示例的示意图，针对该开源 OS，附图 3A 和 3B 中所示的用户启动过程以及附图 4A ～ 4C 中所示的用户应用启动过程可以被实施。

[0010] 为图示说明的简明和清楚起见，附图中的元件未必是按比例的，只是示意性的和非限制性的，公知的组件可以被省略，而且在不同的附图中相同的附图标记指示相同元件，除非另有说明。

[0011] 在下面的描述中，术语“设备（多个）”、“平台（多个）”和“系统（多个）”可以互换使用，且术语“可执行”、“进程”、“模块”和“组件”可以互换使用。在下面的描述中，术语“内核（可执行）进程”、“内核（可执行）模块”和“内核”可以互换使用。在下面的描述中，术语“内核代理（可执行）进程”，“内核代理模块”和“内核代理”可以互换使用。

详细说明

[0012] 参照附图 1-5 仅以举例的方式在下文中描述各实施例。

[0013] 在本公开中，提供了一种用于增强计算设备的安全性的安全平台和方法。用于增强计算设备的安全性的安全平台和方法适用于例如但不限于任何计算设备，计算设备包括移动电话、游戏机、平板、机顶盒、电视机或其他消费电子设备。本公开的实施例仅以举例的方式就如下的任何平台进行了描述，所述平台使用开源操作系统 (OS)（例如但不限于，Linux 或 Android™OS），并且这些示例不应该被解释为限制本发明的预定范围。

[0014] 参考图 1A-1H，图示了在运行时增强计算设备的安全性的过程的示例。所述过程包括验证 (100) 计算设备的代码的完整性以便在计算设备中建立软件 (SW) TEE，以及响应于完整性验证（“IV”）的结果，绑定 (120) 计算设备的用户空间的应用到计算设备的 OS。代码的 IV 包括 OS 的 IV(110) 和用户空间的应用的 IV(130) 中的任一个或者多个。

[0015] 在非限制性的示例中，IV(100) 步骤包括 OS 代码的 IV(110) 以验证该计算设备具有用于在 OS 空间中建立 SW TEE 的可信任 OS。如果 OS 代码的 IV(110) 是成功的，则确定所

述计算设备的OS是被充分信任的。在非限制性的示例中，当OS的IV(110)成功时，实施绑定(120)步骤。在非限制性的示例中，OS的IV(110)包括以下项中的任一个或者多个：磁盘上OS映像的IV，存储器内OS映像的IV，以及存储器内OS映像的连续/增量IV。在非限制性的示例中，在磁盘上和存储器内OS映像的IV成功(112)时，绑定(120)步骤被实施。在非限制性的示例中，在磁盘上和存储器内OS映像的IV(112)成功时，实施存储器内OS映像的连续/增量IV(114)(参见附图1B)。如果存 储器内OS映像的连续/增量IV(114)成功，则实施绑定(120)步骤。

[0016] 在非限制性的示例中，IV(100)步骤包括用户空间应用的IV(130)以验证该计算设备在用户空间中具有SW TEE(参见例如附图1C-1H)。如果用户空间应用的IV(130)成功，则确定所述计算设备的用户空间被充分信任。在非限制性的示例中，如果OS映像的完全IV(110)是成功的，则实施用户空间应用的IV(130)，这在OS空间和用户空间中建立SW TEE。在非限制性的示例中，用户空间应用的IV(130)包括如下项中的任一个或者多个：磁盘上用户空间应用映像的IV(132)，存储器内用户空间应用映像的IV(134)，以及存储器内用户空间应用映像的连续/增量IV(150、170)。这些IV中的任一个或多个可以随后实施。在非限制性的示例中，当用户空间应用映像的磁盘上映像的完全IV(132)成功时，绑定(120)步骤被实施。在非限制性的示例中，当磁盘上用户空间应用映像的完全IV(132)成功时，存储器内用户空间应用映像的完全IV(134)被实施。在非限制性的示例中，存储器内用户空间应用映像的完全IV(134)是通过跳过重定位(例如：用于加载磁盘上用户应用的内容的加载器的重定位)来实施的。在非限制性的示例中，当磁盘上和存储器内用户空间应用映像的完全IV成功时，绑定(120)步骤被实施。

[0017] 在非限制性的示例中，在用户空间应用映像的IV(130)之后，存储器内用户空间应用(包括重定位)的映像被签署(140)(参见例如附图1E-1H)。在非限制性的示例中，在签署存储器内用户空间应用映像(140)后，存储器内用户空间应用映像的连续/增量IV(150)基于(与签署(140)步骤关联的)完全签名得以实施。在非限制性的示例中，当存储器内用户空间应用映像的连续/增量IV(150)成功时，绑定(120)步骤得以实施(参见例如附图1E)。在非限制性的示例中，在签署存储器内用户空间应用映像(140)后，绑定(120)步骤被实施，然后存储器内用户空间应用映像被重签署(160)，因为它可能已经由于绑定步骤的结果而发生变化。在非限制性的示例中，在重新签署存储器内用户空间应用映像(160)后，存储器内应用映像的连续/增量IV(170)基于(与重签署(160)步骤关联的)完全签名得以实施(参见附图1F)。在非限制性的示例中，在绑定(120)步骤后，存储器内应用映像的连续/增量IV(170)被实施，因为绑定(120)步骤可能改变用户应用映像(参见附图1G)。

[0018] 在非限制性的示例中，绑定(120)步骤包括向用户空间应用恢复 丢失的代码段和/或数据段(或“MP”)。在非限制性的示例中，如果任何IV失败，则MP被撤销。在非限制性的示例中，在绑定(120)步骤之后，存储器内用户应用映像的连续/增量IV(170)被实施(参见例如附图1G)，然后当有任何IV失败，则绑定(或MP)被撤销。

[0019] 在非限制性的示例中，如果所有被执行IV均成功，则系统把特权授予给用户应用(190)(例如：对计算设备的资源的特权访问)(参见附图1H)。系统可以根据用户应用和OS绑定(120)的结果把特权授予给用户应用(190)。如果任意IV失败，则特权将被撤回。

[0020] 在非限制性的示例中，计算设备的OS是基于内核的OA，并且OS的IV(110)包括在系统启动期间内核空间的IV以便创建可信任的OS内核空间。内核映像的IV可以由基于内核的OS的内核代理来执行。内核映像的IV包括以下项中任一个或多个：磁盘上内核映像的IV，存储器内内核映像的IV，磁盘上内核代理映像的IV，存储器内内核代理映像的IV。

[0021] 在非限制性的示例中，附图1A-1G的IV步骤、绑定步骤120、签署步骤(140)、重新签署步骤(160)、和/或授予步骤(190)由基于内核的OS的内核代理来实施。在非限制性的示例中，内容在分配给用户应用进程的虚拟存储区域(VMA)中被重定位。在非限制性的示例中，在绑定步骤(120)之后，在VMA中重定位的内容可以被重新签署(160)以用于将来的IV验证，因为一些形式的绑定实际上改变VMA它们自己的内容。

[0022] 在非限制性的示例中，具有OS的平台的安全性是通过强制访问控制(MAC)机制来增强的。MAC约束用户执行应用代码或者访问系统资源的能力，而无论用户是“根”还是伪装成“根”。可信任OS内核空间的内核代理监控和拦截从用户空间到内核的特定系统调用。因此内核代理能够在持续进行的基础上监控用户空间的进程的创建（又名“调用应用”）并且施用进程状态验证来验证调用方。术语“进程状态验证”指的是这样的进程，它确保调用进程的整个可执行存储器是被完全签名/确认的代码，从而防止常见的代码注入攻击（诸如过度写入代码空间或共享库插填(shim)）。内核代理确认可执行映像（磁盘上和存储器内中的任一个或者多个），生成整个进程空间可执行存储器的可核实签名，并绑定用户空间应用到可信任内核空间。以这种方式，内核代理KA被用于在内核空间建立软件(SW)TEE，并且（至少为感兴趣的指定用户空间应用）把SW TEE扩展到用户空间。

[0023] 在非限制性的示例中，进程状态验证基于签名和验证（例如，附图1A-1H中的IV）来实施，确保没有任何可执行代码能够从原始签名的可执行体的内容中被修改。进一步地，由于内核代理调配所有代码（包括共享库）的加载，所以阻止了（例如经由共享库插填(shimming)的）无签名代码的注入。

[0024] 在非限制性的示例中，基于内核的OS的内核代理通过使用缺失段（“MP”）技术绑定用户空间代码到可信任的内核空间代码。绑定用户空间应用到基于内核的OS能够防止恶意攻击者从磁盘上用户应用文件完全地得到可执行的代码并在另一个设备上运行它。这个基于MP的绑定技术还防止攻击者关闭一些用户应用的保护。

[0025] 参见附图2A-2D，图示了在具有基于内核的OS的计算设备上构建安全平台并供应利用该安全平台的准备的过程的例子。构建和供应准备过程包括供应和签署系统可执行体。在非限制性的示例中，所述供应过程包括用户应用可执行体供应200A和内核代理可执行体供应200D。在非限制性的示例中，签署系统可执行体200B过程包括用户应用可执行体的代码签署200C。

[0026] 用户应用可执行体供应200A（附图2A）：原始用户应用可执行体202（附图2A中的“用户应用可执行体”）里的代码段和/或数据段210（附图2A中的“MP”）被用于供应用户应用文件202。所述代码段和/或数据段210可以包括但不限于例如加密密钥、其它凭证值、秘密和/或例如公开号为2010/0296649、2011/0150213和2011/0235803的美国专利申请（其通过引用方式并入本文）中所公开的白盒表数据。

[0027] 在非限制性的示例中，代码段和/或数据段210通过利用供应工具204从原始用户应用文件202中提取并且被哑元值所替换。所得到的用户应用的可执行映像（文件）206

包含了除所述代码段和/或数据段 210 以外的用户应用的代码和/或数据的原始版本。在运行时，内核代理（例如附图 2B 的 212 和附图 2D 的 236 中所示的“KA.ko”）将这缺失的代码段和/或数据段 210 注入到用户空间进程的可执行存储器中，以恢复原始用户应用文件 202 的该代码段和/或数据段 210。

[0028] 在另外的非限制性的示例中，用户应用文件 202 中所述代码段和/或数据段 210 由供应工具 204 来损坏或加密。在运行时，内核代理（例如附图 2B 的 212 和附图 2D 的 236 中所示的“KA.ko”）将修补或解密所述受损或加密的代码段和/或数据段，并将所得的代码段和/或数据段注入到用户空间进程的可执行存储器用来恢复丢失的原始代码段和/或数据段 210。

[0029] 签署可执行体 200B（附图 2B）：为了鉴权和保护系统可执行体 212，签名工具 214 对可执行体 212 进行数字签名并且生成签名的可执行体 216。在非限制性的示例中，系统可执行体 212 包括含有内核映像（例如，Vmlinux）的可执行体文件、内核代理可执行体文件（“KA.ko”）、多个指定的共享库（也被称为动态链接库）、“/lib/*.so”、以及用户应用（“用户应用”）的可执行体文件。签名的可执行体 216 防止攻击者用无签名可执行体的各版本替换已签名的可执行体以及用于明确标识可执行体代码，该可执行体代码被视为是由内核代理保护和/或授予特权的。

[0030] 在非限制性的示例中，用户应用可执行体在用户应用可执行体供应 200A 之后在 200B 被签署。在另一非限制的示例中，在所述代码段和/或数据段 210 的提取不改变要被签署的用户应用可执行体的情况下，用户应用可执行体可以在用户应用可执行体供应 200A 之前在 200B 被签署。

[0031] 代码签名 200C（附图 2C）：在非限制性的示例中，用户应用可执行体文件 222（附图 2C 中的“用户应用 . 可执行体”）由签名工具 224 来签署。表示代码内容的权利的凭证 228 由签名工具 224 创建。凭证 228 使用凭证加密密钥（“VEK”）进行加密保护，以防止检测/修改。凭证 228 被附加到已签名的用户应用可执行体文件 226。这个凭证 228 被用于在运行时确认用户代码。第三方（例如，文件 202 的可信任发布者）可进一步签署用户应用。

[0032] 用户应用可执行体的供应 200A 和代码签名 200C 可以独立地实施，这取决于 MP 技术。例如，如果数据段 210 被从用户应用文件 202 提取，代码签名 200C 则独立于用户应用供应 200A 而被实施。如果用户应用文件 202 的代码段 210 被供应工具 204 改变，则其改变了可执行体文件的内容（和因此计算出的签名）并且代码签名 200C 因此在用户应用供应 200A 后实施。

[0033] 内核代理可执行体供应 200D（附图 2D）：在非限制性的示例中，内核代理可执行体文件 232 通过使用供应工具 234 而被配置，该供应工具 234 取凭证加密密钥 VEK、Vmlinux. 凭证以及与用于供应 (200A) 用户应用可执行体的代码和/或数据 210 相关联的缺失代码段和/或数据段 230 作为输入，以使所 得的内核代理 236 通过提供缺失的代码和/或数据 210 而能够恢复原始的用户应用功能。所述代码和/或数据 230 可以是原始用户应用文件 202 中的代码段和/或数据段 210，或者是用于恢复代码段和/或数据段 210 的代码段和/或数据段。在非限制性的示例中，内核代理可执行体文件 236 在 200B 被签署。

[0034] 在运行时，响应于在存储器内创建用户应用可执行体映像的调用，内核代理实施

使用凭证对（磁盘上的）用户应用文件的 IV（例如，附图 2C 的 228），实施存储器内相关联用户应用进程的 IV，并且在用户应用进程的可执行存储器内恢复原始用户应用文件的代码段和 / 或数据段 210。

[0035] 内核代理被配置成除了其他方面特别在系统中实施代码或数据映像的 IV，包括例如（例如，在 HDD 或闪存上的）静态的或磁盘上的完整性验证和（例如，在随机存取存储器 (RAM) 中的）动态的或存储器中的完整性验证。内核代理被配置成除了其他方面特别作为观察器来动态监控内核组件、安全的启动组件、所有受保护的应用和与之相关联的共享库的完整性。

[0036] 所述 IV 是这样被实施的，例如但不限于通过计算对象（例如：系统组件、应用）的签名（例如，哈希值），然后比较那个签名与该签名的已知的有效值。如果所计算的值与存储的已知有效值相同，则内核代理就假定对象尚未被攻击者修改。然而，如果所计算的值与存储的已知有效值不同，则内核代理就假定对象已被修改，并且不能再被信任用于执行它曾被预定要执行的功能，或者其不拥有最初曾分配给所述对象的相同特权。

[0037] 参见附图 3A 和 3B，图示了安全平台中系统启动过程的一个示例。在系统启动过程中，平台中内核空间 300 被初始化，以使任何未签名的内核代码或未通过 IV 验证的已签名内核代码不被许可加载，这在内核空间 300 中建立软件 TEE。

[0038] 系统启动过程除其他外特别包括：执行 OS 引导加载过程 302（附图 3A 中“引导加载器”），其将加载和启动操作系统。通过使用引导加载器 302，包括可执行内核 304 的系统可执行体文件被加载（附图 3A 的步骤 1）。内核可执行体文件 304 见于例如但不限于 Vmlinux 映像文件中。当内核文件 304 被加载时，在存储器中创建内核可执行进程 306（附图 3A 的步骤 2）。内核进程 306 被执行以加载内核代理可执行映像（文件）308（附图 3A 的步骤 3）。通过加载内核代理可执行体文件 308 在存储器中创建内核代理可执行进程 310（附图 3A 的步骤 4）。

[0039] 例如，内核代理文件 308 对应于附图 2D 中的内核代理文件 236，图 2D 中缺失的代码段和 / 或数据段 230 已经被附连到所述文件。

[0040] 初始化期间内核代理 310 获得执行控制权，以在内核空间 300 中执行 IV 操作。在非限制性的示例中，内核代理 310 实施 IV 操作，包括：(1) 存储器内的 IV 以验证内核映像本身的完整性 306（附图 3B 的步骤 3）；(2) 磁盘上的 IV 以验证内核代理可执行体文件 308 的完整性（附图 3B 的步骤 4）；以及 (3) 存储器内的 IV 以验证内核代理 310 可执行体映像本身的完整性（附图 3B 的步骤 5）中。所有这些组件的完整性由内核代理模块 310 例如通过使用与分配给内核代理 310 的内核代理安全存储（例如附图 5 的 532）中的数据的比较加以确认。

[0041] 另外，内核代理 310 被执行以对内核可执行体映像和内核代理可执行体映像增量地执行所有存储器内的 IV，这将检测动态攻击。

[0042] 通过完成 IV 操作，内核代理 310 确认：安装在平台中的 OS 是预定的 OS，使得内核 306 在内核代理 310 的引导加载和启动 (bring-up) 之间没有被修改，并且内核代理 310 执行它曾被预定要执行的功能。内核空间 300 被完全验证和信任，因此内核代码和内核代理代码以完全安全的方式运行。如果内核代理 310 检测到自身在不可信任的环境中运行，其将采取适当的动作，如关闭自己，以及消除已经分发给用户应用的任何有特权的资产（或

资源，例如，OS、硬件组件、网络组件）。

[0043] 参见附图 4A-4C，图示了安全平台中的用户应用启动过程的一个示例。一旦内核代理 310 完成附图 3A 和 3B 的系统启动过程，平台对用户应用启动过程已经就绪，并且内核代理 310 监控用于创建用户空间进程的系统调用。在这一点上，内核空间 300 提供完全可信任的环境。

[0044] 来自用户应用可执行体 402 用于创建用户应用进程的系统调用（附图 4A 的“进程创建”）在内核 306 中被截取（附图 4A 的步骤 1）。响应于进程创建系统调用，Linux 安全性模块 (LSM)312 在内核代理 310 中提供钩子（附图 4A 的步骤 2）。

[0045] 用户应用文件 402 包含例如在附图 2C 中所示的代码签名过程 200C 中签署的用户应用可执行体文件和附图 2A 中缺失的代码段和 / 或数据段 210。

[0046] 内核代理 310 然后通过使用嵌入式凭证 404（例如，附图 2C 的 228）实施用户应用文件 402 的磁盘上 IV（附图 4A 的步骤 3）。如果用户应用的 IV 验证失败，或者如果凭证未使用正确凭证加密密钥（例如，附图 2C 和 2D 的 VEK）进行加密，或者如果根本不存在凭证，则用户应用文件仍然可以被许可去加载，但它不会授予任何特权。以这种方式，篡改的应用代码被与完全无签名的代码进行同样的处理，并且（其中用户自由地安装任意软件的）设备的“正常”操作将被保护。可替代地，内核代理 310 可以针对设备安全性采取更受限制的方式和拒绝加载已签名但被篡改的代码，或者严格的多得多的限制，拒绝加载未签名代码。最后的这种方法可以被称为白名单方法，其中只有（“在白名单上”的）已签名的代码会被允许运行。

[0047] IV 可通过例如但不限于将凭证 404 中的签名与对应于用户应用资源的已知值进行比较来进行。用户应用的资源可以包括存储在安全存储中的应用签名证书和 IV 信息。如果签名值与所存储的应用签名证书（即，已知的有效值）相同，则内核代理 310 就假定磁盘上的用户应用文件 402 尚未被攻击者修改，并且其权限或特权没有被修改。如果签名值不同于已知的有效值，则内核代理 310 就假定用户应用文件 402 已经被修改，并且不能再被信任来执行它曾被预定执行的功能。

[0048] IV 结果经由 LSM 312 发送回内核 306（例如，如果 IV 验证成功，则“通过”）（附图 4A 的步骤 4）。响应于接收到 IV 的确认（“通过”），内核 306 将执行与“进程创建”系统调用相关联的动作（附图 4A 的步骤 5），以使得在用户空间 400 中创建新的用户应用进程 406，它是空的（附图 4A 的步骤 6）。

[0049] 用户应用内容被重新定位到由内核分配的虚拟存储区域 VMA 中，以供由用户应用进程 406 使用。为了重新定位所述内容，可执行体加载器 410 按照附图 4B 中所示的那样处理可执行体文件。例如，当加载可执行和链接格式 (ELF) 可执行体时，ELF 加载器 410 是可执行代码，其被配置成处理用户应用 402 的 ELF 文件磁盘映像，在用户应用进程 406 中创建一个或多个 VMA（例如，附图 4B 所示的 408-1、408-2、...），以及将包含在 ELF 文本部分中的可执行代码重新定位到 VMA 中。这里，由 ELF 加载器 410 加载的所有可执行代码采用 VMA 的形式。

[0050] 重新定位的内容通常是分支目标地址，并且因此影响应用进程的控 制流，导致有可能成为攻击目标。内核代理 310 按照附图 4B 中所示的那样实施 VMA 的存储器内的 IV 验证，这将跳过重定位。内核代理 310 可以签署包括重定位的 VMA。

[0051] 在 IV 验证成功的情况下，准许恢复丢失的代码和 / 或数据（或“MP”）。内核代理 310 通过在用户应用进程 406 存储器中恢复原始用户应用可执行体的丢失的代码段和 / 或数据段（或“MP”）420 来绑定用户应用进程 406 到内核代理（和 SWTEE）310，如附图 4C 所示。代码段和 / 或数据段 420 例如是附图 2A 中的代码和 / 或数据 210，代码和 / 或数据 210 已经在用户应用供应步骤中被提取，被损坏和 / 或加密。

[0052] 内核代理 310 在 VMA（多个）中的适当位置注入 MP420。在非限制性的示例中，用户应用供应过程（附图 2A 的 200A）中使用的哑元值（多个）被替换为相应的原始代码和 / 或数据（附图 2A 的 210）。在非限制性的示例中，在用户应用供应过程（附图 2A 的 200A）中被损坏和 / 或加密的代码段和 / 或数据段被恢复。在 MP 420 是代码段的情况下，该代码段的恢复改变了 VM 内容的签名。因此，内核代理 310 重新签署在 VMA 中重新定位的内容，以得到整个 VMA 内容的全新签名，如附图 4B 所示。当该进程被调用时，内核代理 310 随后执行增量 / 正在进行中的存储器内整个 VMA 内容的 IV，以用于进一步的验证。

[0053] 通过完成 VMA 的 IV 和恢复丢失的代码段和 / 或数据段 420 到用户应用进程 406 的存储器中，用户应用进程 406 被充分验证且被信任，并且能够按照初始预定的那样进行操作（作为对比，如果在任何阶段 IV 验证失败，缺失的代码段和 / 或数据段将不被 KA 提供，并且应用进程将不按照初始预定的那样进行操作）。作为结果的用户应用模块 406 执行用户应用的原始版本（例如，附图 2A 的 202）曾被预定要执行的功能，但是作为结果的用户应用模块 406 现在被绑定到内核代理 310，以及它的用户应用的可执行体映像的成功确认。通过重新签署和验证 VMA 的内容，检测对重定位的内容作出的任何改变都是可能的。

[0054] 要接受或拒绝用于创建用户空间进程的调用，内核代理 310 可以添加进一步的标准，包括但不限于系统 / 应用完整性、应用权限、应用行为、用于其它可能正在运行的应用的安全性上下文和远程命令。

[0055] 参见附图 5，图示了其中实施进程状态验证的安全平台 500 的一个示例。平台 500 的底层包括例如含有中央处理单元 (CPU) 504 和存储器（只读存储 器 (ROM)）506 的片上系统 (SOC) 502 组件，所述存储器中驻留有基本输入 / 输出系统 (BIOS) 408 和数字证书 510。平台 500 的最高层是设备应用层，其包括一个或多个用户应用 520a、520b，如果适宜，每个用户应用均具有签名 522。介于中间的层包括开放源 OS 525、OS 内核 530、用于内核代理 534 的安全存储 532、系统调用接口 536、硬盘驱动 (HDD) 存储设备或闪速存储器 540、以及引导加载器 550。

[0056] 内核代理 534 维持并具有对所保护的数据存储 532 的单独访问权，在所保护的数据存储 532 中，内核代理 534 保持与内核资源访问控制的内核代理性能、完整性验证、应用许可和应用资源访问控制有关的信息。尽管安全存储 532 在图 5 中示出为单独组件，但是安全存储 532 存在于硬盘或闪存 540 中。安全存储 532 可以作为片上系统底层 502 中的安全存储区域存在。

[0057] 内核代理 534 是 Linux 安全模块接口 (LSM I/F) 顺应的。内核代理 534 形成 OS 内核 530 的构成整体所需要的和不可拆卸的部分，在没有 OS 内核 530 的情况下，基于内核的 OS 和 / 或应用 520a、520b 将停止正常工作。为了使内核代理 534 抗篡改、修改和逆向工程的攻击，内核代理 534 可以使用软件保护技术进行保护，软件保护技术诸如但不限于在每篇均授予 Chow 等人的美国专利 6,594,761、6,779,114、6,842,862 和 7,506,177 中更加

详细描述的那些技术。

[0058] 内核代理 534 的功能的一个示例是监视基于内核的 OS 和加载到平台 500 上的用户应用 520a、520b 这两者的完整性，并检测 OS 530、安全引导 550 和用户应用 520a、520b 的任何违反行为。IV 可以按图 3B 和 4B 所示的那样进行。内核代理 534 的功能的另一个示例是将（磁盘上的）用户应用可执行体中缺失的代码段和 / 或数据段（图 4C 的“MP”420）注入到用户应用进程的可执行存储器中，如果该用户应用进程的可执行存储器已被签署 / 验证的话。

[0059] 内核代理 534 的功能的另一个例子可以包括控制对 OS 内核资源和数据的应用访问，其中访问控制决策可以由内核代理基于但不限于以下因素做出，诸如 :OS 内核完整性、应用完整性、应用上下文，以及由任何给定的受信任根权威机构授予的特权。基于 OS 内核完整性的访问控制决策确定内核是否已被修改、被替换、被添加、或以未经授权的方式被部分消除。所述访问控制决策也可以确定安全启动进程是否成功完成（即没有篡改）。如果 OS 内核已经被修改、替换、添加或部分消除或安全引导过程不能得到肯定的验证，则所述确定会用于无效内核代理或应用或诸如媒体播放器的安全应用会通常在其下运行的许多假定条件。基于应用完整性的访问控制决策确定正试图访问 OS 内核资源的应用是否已被以任何方式修改（例如，在应用中插入恶意软件，或者由其他恶意软件修改），或与该应用相关联的特权是否被修改（例如来给予用于访问系统资源的且未曾由认证权威机构授权的特权）。

[0060] 在本公开的实施例中的每个元件可被实现为硬件（例如，通用和 / 或专用计算机处理器）、载波中的软件 / 程序、或它们的任意组合。软件代码，或者其全体或者其部分，可以被存储在计算机可读介质中（例如，像 ROM（诸如 CD ROM 或半导体 ROM），或者磁记录介质（例如软盘或硬盘），并且可以由计算机处理器来执行。该程序可以采用源代码、目标代码、代码中间源和目标代码（诸如部分编译的形式）的形式、或采用任何其他的形式。可被嵌入在载波中的表示软件代码的计算机数据信号可以通过通信网络进行传输。所述载体可以是任何实体或能够承载程序的设备。此外，载体可以是可传输载体（例如电信号或光信号），其可以经由电缆或光缆或者通过无线电或其他手段来传送。当程序被包含在这样的信号中时，载体可以由这种电缆或其他设备或装置构成。可替代地，载体可以是其中嵌入程序的集成电路，所述集成电路适于执行相关方法或在相关方法的执行中使用。

[0006]]通过举例的方式已经描述了一个或多个当前优选实施例。对于本领域技术人员显而易见的是 :在不脱离在权利要求中限定的本发明的范围的情况下，可以做出许多变化和修改。

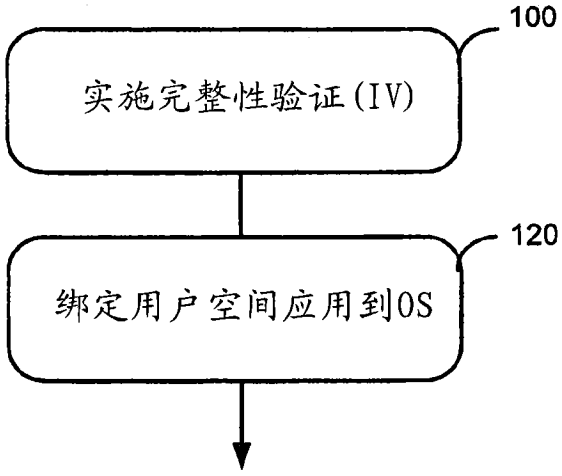

100
实施完整性验证(IV)
120
绑定用户空间应用到OS

图 1A

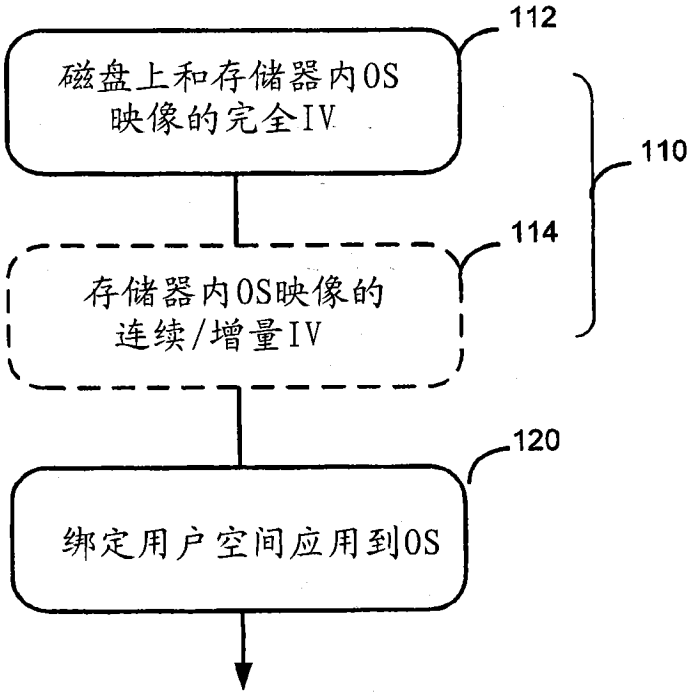

112
磁盘上和存储器内OS
映像的完全IV
110
114
存储器内OS映像的
连续/增量IV
120
绑定用户空间应用到OS

图 1B

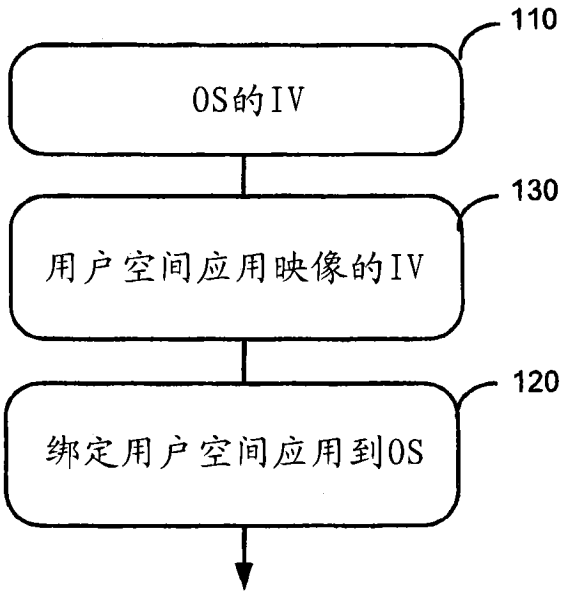

110
OS的IV
130
用户空间应用映像的IV
120
绑定用户空间应用到OS

图 1C

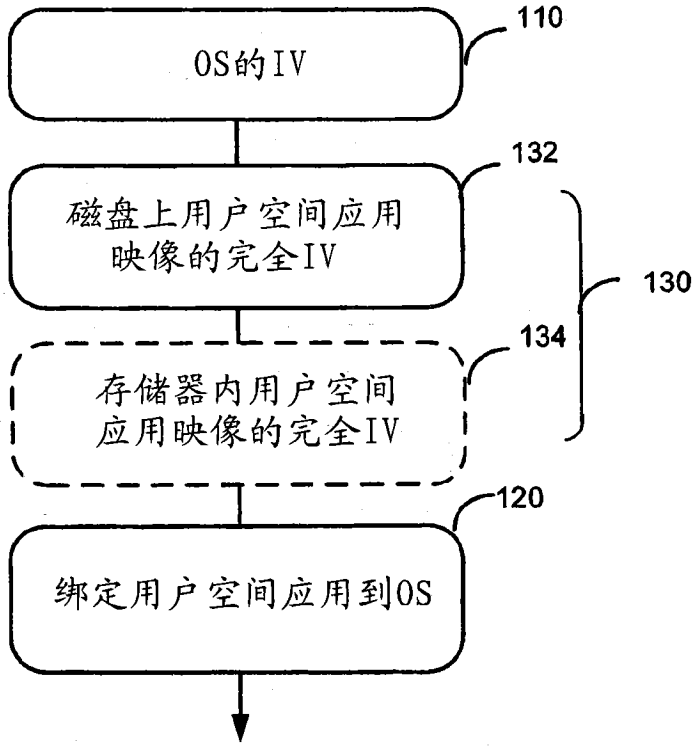

110
OS的IV
132
磁盘上用户空间应用
映像的完全IV
130
134
存储器内用户空间
应用映像的完全IV
120
绑定用户空间应用到OS

图 1D

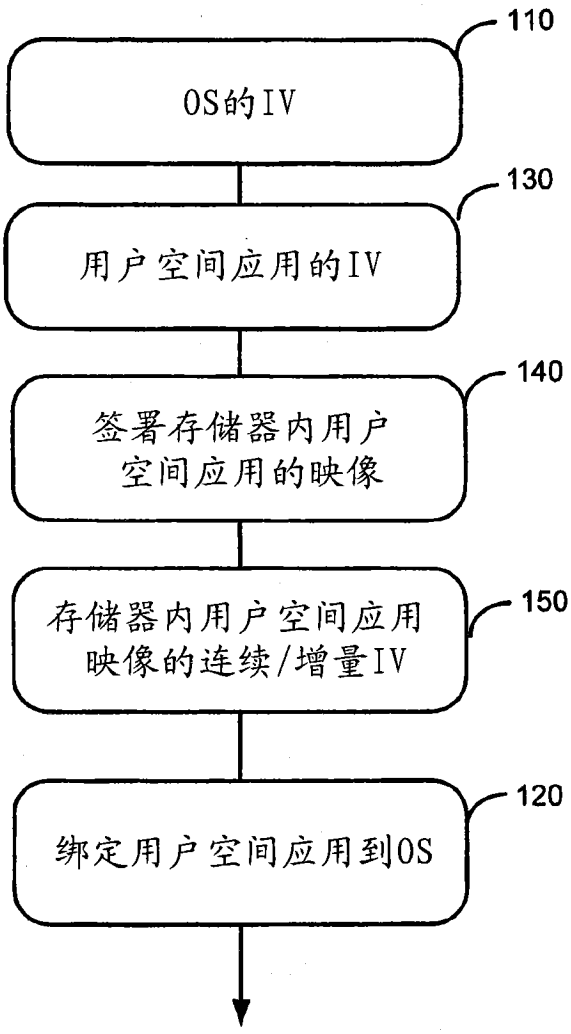

110
OS的IV
130
用户空间应用的IV
140
签署存储器内用户空间应用的映像
150
存储器内用户空间应用映像的连续/增量IV
120
绑定用户空间应用到OS

图 1E

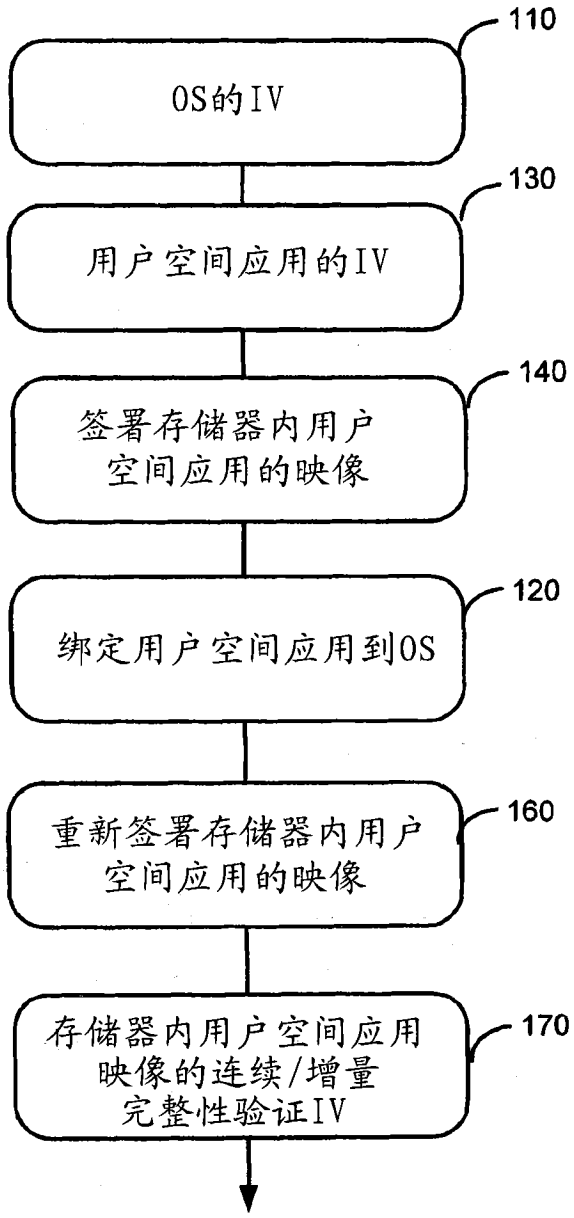

110
OS的IV
130
用户空间应用的IV
140
签署存储器内用户空间应用的映像
120
绑定用户空间应用到OS
160
重新签署存储器内用户空间应用的映像
170
存储器内用户空间应用映像的连续/增量完整性验证IV

图 1F

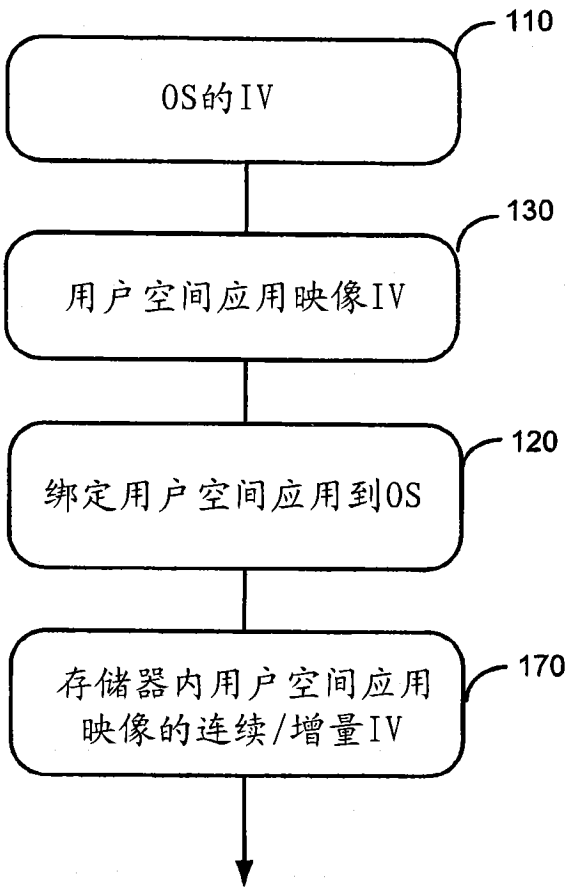


图 1G

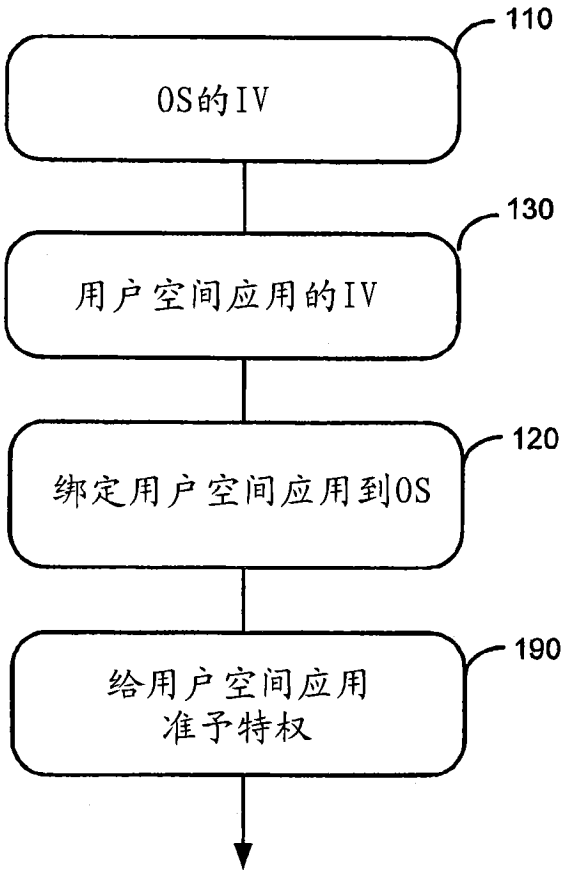


图 1H

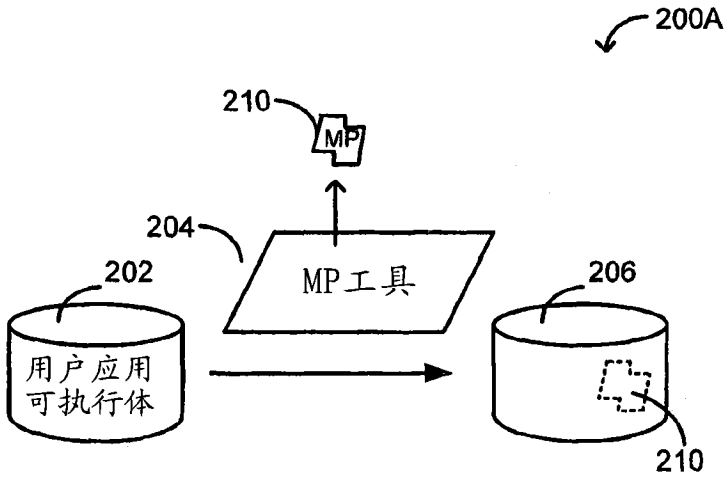


图 2A 用户应用可执行体供应

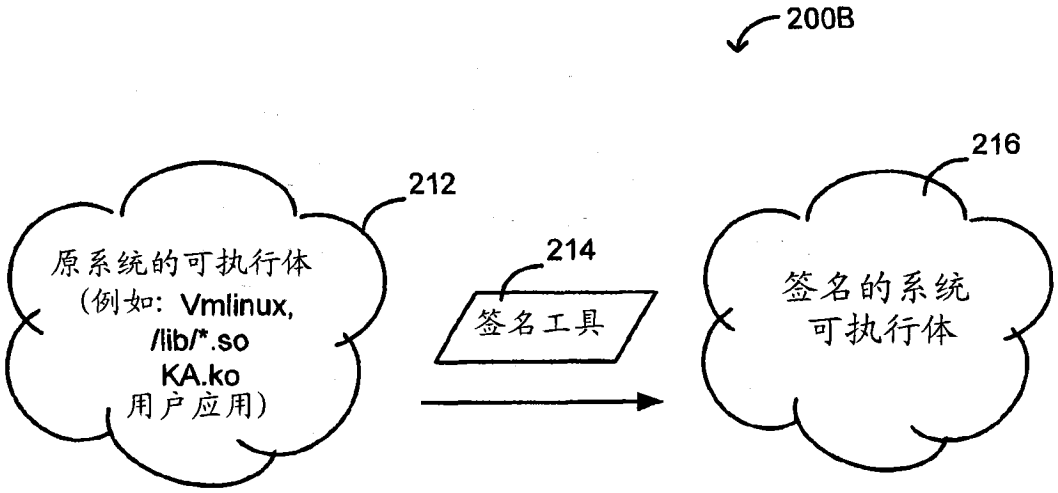


图 2B 签署可执行体

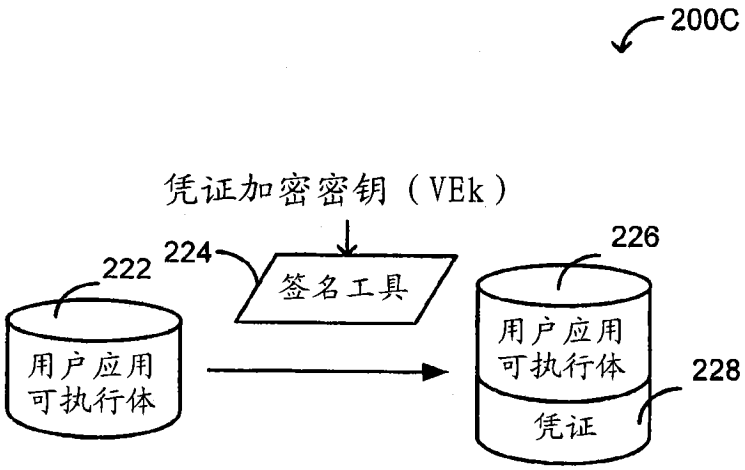


图 2C 代码签名

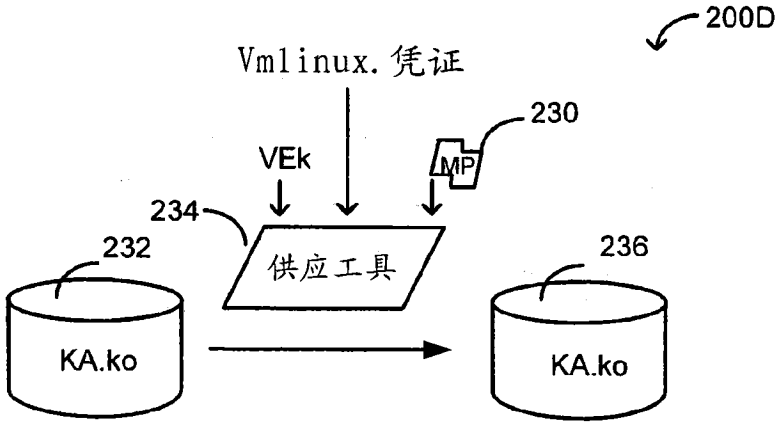


图 2D 内核代理可执行体供应

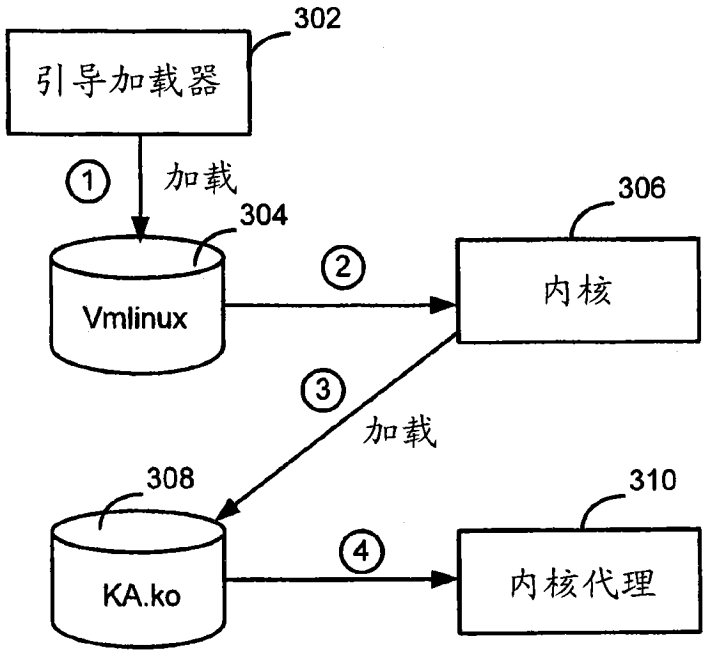

302
引导加载器
1
加载
304
306
Vmlinux
2
内核
3
加载
308
310
KA.ko
4
内核代理

图 3A

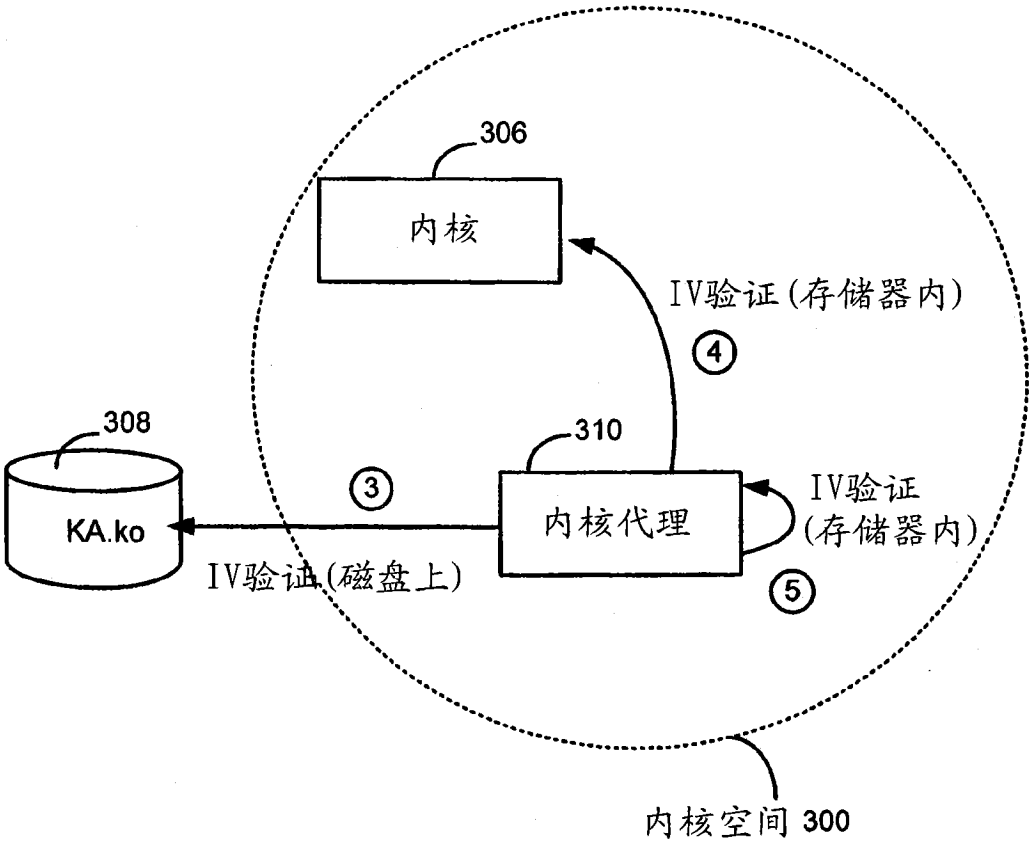

306
内核
IV验证(存储器内)
4
308
310
3
IV验证
(存储器内)
KA.ko
内核代理
IV验证(磁盘上)
5
内核空间 300

图 3B

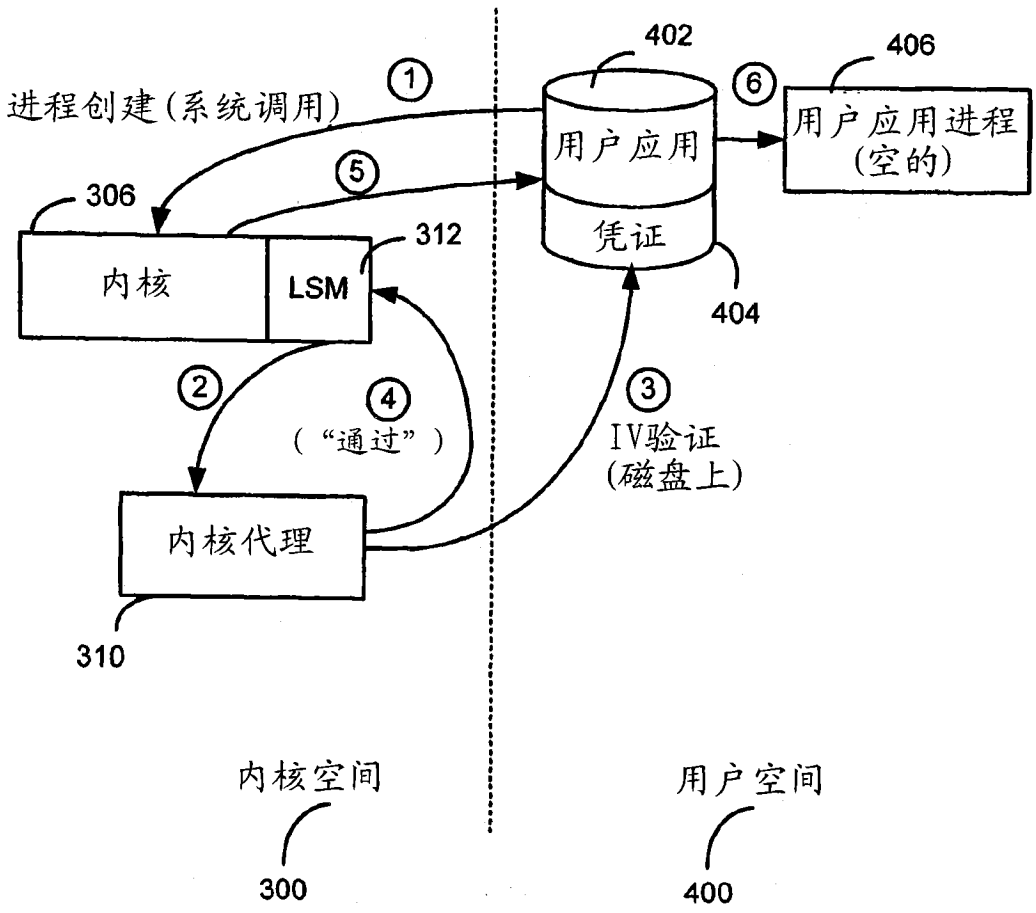

进程创建(系统调用)
1
402
406
6
用户应用
用户应用进程
(空的)
5
306
312
内核
LSM
凭证
404
2
4
("通过")
3
IV验证
(磁盘上)
内核代理
310
内核空间
用户空间
300
400

图 4A

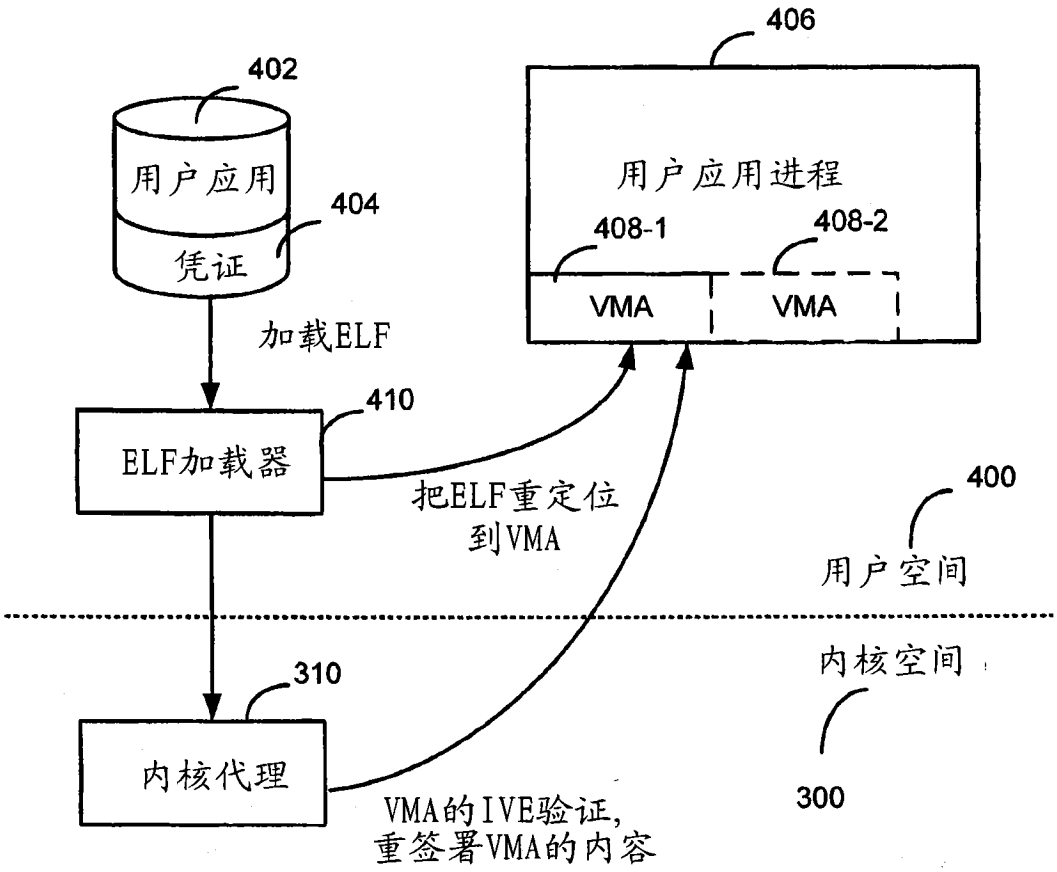

406
402
用户应用
404
凭证
用户应用进程
408-1
408-2
VMA
VMA
加载ELF
410
ELF加载器
把ELF重定位
到VMA
400
用户空间
内核空间
310
内核代理
300
VMA的IVE验证,
重签署VMA的内容

图 4B

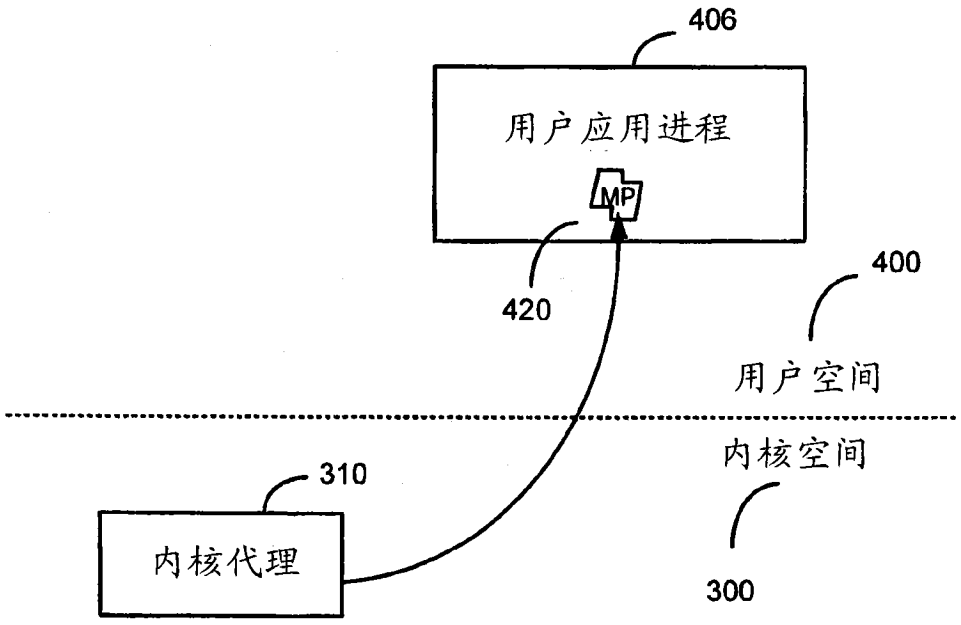

406
用户应用进程
MP
420
400
用户空间
内核空间
310
内核代理
300

图 4C

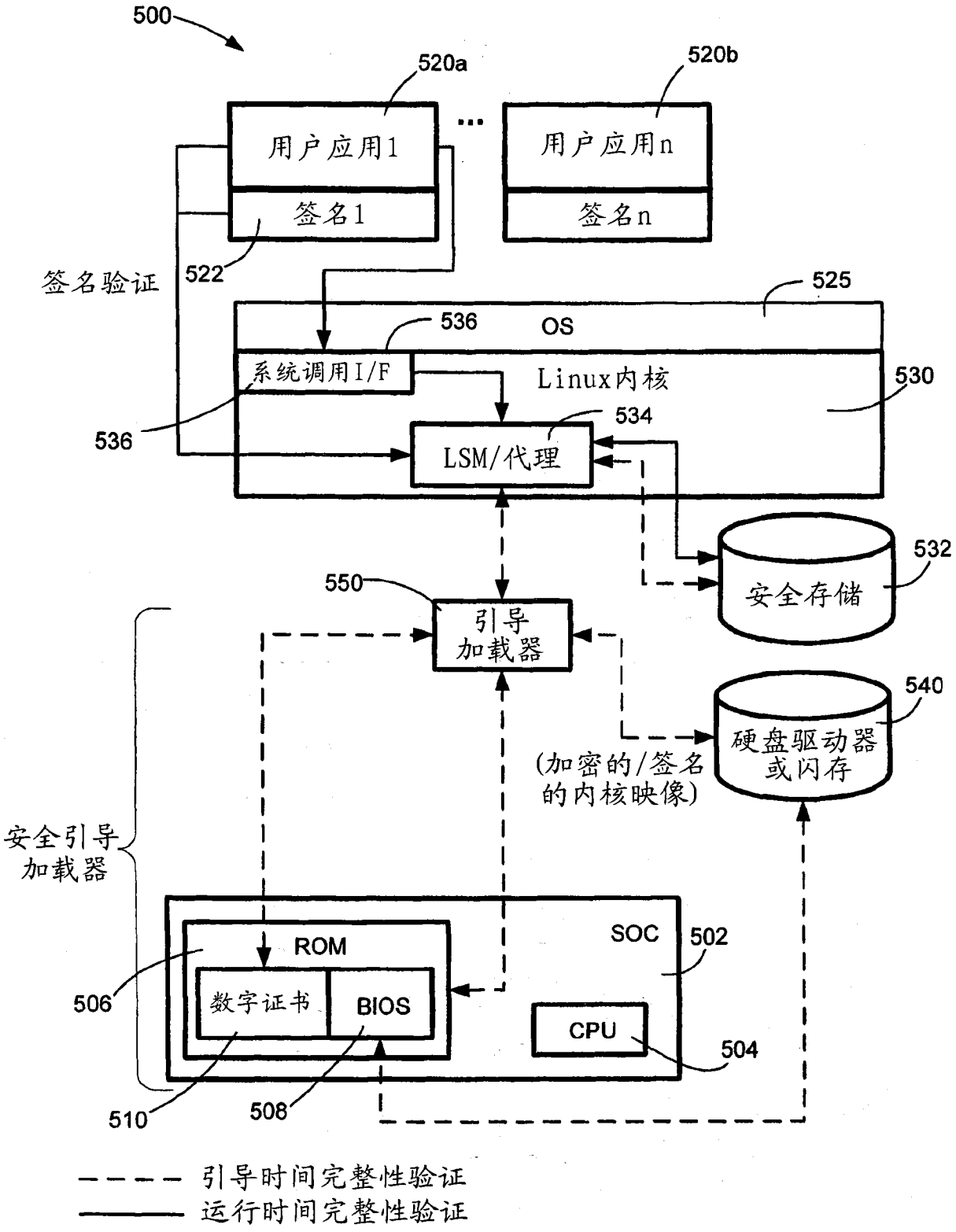

500
520a
520b
用户应用1
...
用户应用n
签名1
签名n
签名验证
522
525
536
OS
系统调用I/F
Linux内核
530
536
534
LSM/代理
532
安全存储
550
引导
加载器
540
硬盘驱动器
或闪存
(加密的/签名
的内核映像)
安全引导
加载器
ROM
SOC
502
506
数字证书
BIOS
CPU
504
510
508
引导时间完整性验证
运行时间完整性验证

图 5