(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 109684849 B
(45)授权公告日 2023.06.27

(21)申请号 201811209405.0

(22)申请日 2018.10.17

(65)同一申请的已公布的文献号
申请公布号 CN 109684849 A

(43)申请公布日 2019.04.26

(30)优先权数据
2017-201956 2017.10.18 JP

(73)专利权人 佳能株式会社
地址 日本国东京都大田区下丸子3丁目30-2

(72)发明人 清水将太

(74)专利代理机构 北京怡丰知识产权代理有限公司 11293
专利代理师 迟军 李艳丽

(51)Int.Cl.
G06F 21/57(2013.01)

(56)对比文件
CN 101042720 A,2007.09.26
CN 106775716 A,2017.05.31
US 2005005101 A1,2005.01.06
WO 2012057632 A2,2012.05.03
CN 104794393 A,2015.07.22
Alessio Gaspar 等.Root-kits & loadable kernel modules: exploiting the Linux kernel for fun and (educational) profit.Journal of Computing Sciences in Colleges.2006,第 244-250页.
杨霞;雷林;吴新勇;吴开均;桑楠;.采用数字签名技术的可信启动方法研究.电子科技大学学报.2016,(第03期),第448-452页.

审查员 刘莹

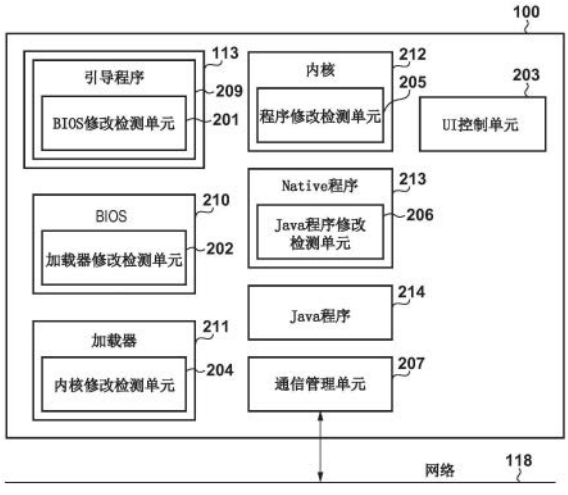
权利要求书2页 说明书8页 附图11页

(54)发明名称

信息处理装置、其控制方法和存储介质

(57)摘要

本发明涉及信息处理装置、其控制方法和存储介质。该信息处理装置在启动引导程序之后依次启动多个模块。各个模块使用用于验证接下来要启动的模块的签名的验证信息,来检测对接下来要启动的模块的修改,并且签名验证成功的情况下,启动接下来要启动的模块。此外,各个模块预先保持所述验证信息以及其自身的签名。



CN 109684849 B

1.一种信息处理装置,其能够被操作以在通过嵌入式控制器启动引导程序之后依次启动多个模块,各个模块包括:

检测单元,其使用用于验证接下来要启动的模块的签名的验证信息,来检测对接下来要启动的模块的修改,以及

启动单元,其在检测单元进行的签名验证成功的情况下,启动接下来要启动的模块,

其中,所述多个模块至少包括基本输入/输出系统、加载器、第一内核和第一程序,并且所述多个模块按基本输入/输出系统、加载器、第一内核和第一程序的顺序被启动,

其中,各个模块预先保持验证信息以及其自身的签名,并且

其中,所述信息处理装置至少保持除了第一内核之外的第二内核作为备用,并且在第二内核作为引导目标被启动的情况下,以与第一内核相同的方式执行检测单元的验证处理和启动单元的启动处理。

2.根据权利要求1所述的信息处理装置,其中,

用于执行所述引导程序的所述嵌入式控制器,包括检测单元和启动单元,并且预先保持验证信息。

3.根据权利要求1所述的信息处理装置,其中,在检测单元进行的签名验证失败的情况下,启动单元停止启动所述信息处理装置。

4.根据权利要求1所述的信息处理装置,其中,

所述引导程序和基本输入/输出系统存储在ROM中,加载器、第一内核以及所述第一程序存储在闪存存储器中,在所述第一程序之后要被启动的第二程序存储在HDD中。

5.根据权利要求4所述的信息处理装置,其中,

所述闪存存储器还存储所述第二程序,并且

在检测单元针对存储在HDD中的第二程序进行的签名验证失败的情况下,启动所述第二程序的模块的启动单元将存储在闪存存储器中的第二程序重新展开到HDD中,并启动所述第二程序。

6.根据权利要求1所述的信息处理装置,其中,

启动单元根据用户输入替换接下来要启动的模块并进行启动,

各个模块预先保持接下来能够启动的多个模块中的各个模块的验证信息以及其自身的签名。

7.根据权利要求1所述的信息处理装置,

所述信息处理装置还包括控制各个模块的主控制器,

其中,所述嵌入式控制器具有与所述主控制器分离的存储器和处理器。

8.一种信息处理装置的控制方法,所述信息处理装置能够被操作以在通过嵌入式控制器启动引导程序之后依次启动多个模块,其中,

各个模块包括:

使用用于验证接下来要启动的模块的签名的验证信息,来检测对接下来要启动的模块的修改,以及

在签名验证成功的情况下,启动接下来要启动的模块,

其中,所述多个模块至少包括基本输入/输出系统、加载器、第一内核和第一程序,并且所述多个模块按基本输入/输出系统、加载器、第一内核和第一程序的顺序被启动,

其中，各个模块预先保持验证信息以及其自身的签名，并且

其中，所述信息处理装置至少保持除了第一内核之外的第二内核作为备用，并且在第二内核作为引导目标被启动的情况下，以与第一内核相同的方式执行验证处理和启动处理。

9.一种计算机可读存储介质，其存储用于使计算机执行信息处理装置的控制方法的各个步骤的计算机程序，所述信息处理装置能够被操作以在通过嵌入式控制器启动引导程序之后依次启动多个模块，其中，

各个模块包括：

使用用于验证接下来要启动的模块的签名的验证信息，来检测对接下来要启动的模块的修改，以及

在签名验证成功的情况下，启动接下来要启动的模块，

其中，所述多个模块至少包括基本输入/输出系统、加载器、第一内核和第一程序，并且所述多个模块按基本输入/输出系统、加载器、第一内核和第一程序的顺序被启动，

其中，各个模块预先保持验证信息以及其自身的签名，并且

其中，所述信息处理装置至少保持除了第一内核之外的第二内核作为备用，并且在第二内核作为引导目标被启动的情况下，以与第一内核相同的方式执行验证处理和启动处理。

信息处理装置、其控制方法和存储介质

技术领域

[0001] 本发明涉及信息处理装置、其控制方法和存储介质。

背景技术

[0002] 对于计算机系统中的软件漏洞来说，修改(alter)软件并利用计算机的攻击是一个问题。作为此类攻击的对策，非专利文件("About TPM",[online],[search on October 12,2017],Insight International Inc.,Internet:<http://www.insight-intl.com/jigyou/securitysystem/about_TPM/about_TPM20.html>)提出了TPM(Trusted Platform Module,可信平台模块)，TPM旨在加强TCG(Trusted Computing Group,可信计算组)在计算机系统(平台)的计算环境中的安全性。在TPM中，在启动BIOS/UEFI和引导加载器(boot loader)时，从CRTM(Core Root of Trust for Measurement,可信测量核心根)连续获得BIOS/UEFI和引导加载器的哈希值，并且在一个接一个地扩展TPM的PCR的同时，启动BIOS/UEFI和引导加载器。此外，在TPM中，通过在启动完成之后确定TPM中的PCR值是否与正确答案值匹配来进行修改检测。这种安全启动被称为可信引导(安全引导)。注意，因为CRTM或TPM是不能被改写的构造，所以在理论上，如下的攻击是不可能的：该攻击是对用于执行修改检测处理的部分进行修改来避免修改检测处理。

[0003] 但是，在前述传统技术中存在下述的问题。例如，对于没有TPM的构造(例如低成本设备)，无法实现可信引导。在这种构造的情况下，在用于执行修改验证处理的应用程序的启动完成之后，执行修改验证处理。因此，即使已经进行了修改，也不可能立即检测到修改，并且因为在BIOS或OS操作到一定程度之后检测到修改，因此为被恶意修改的应用程序提供了例如运行空间。

发明内容

[0004] 本发明使得能够实现如下机制：即使在不具有TPM的构造的情况下，也能够在启动时检测对系统的修改。

[0005] 本发明的一个方面，提供一种信息处理装置，其能够被操作以在通过嵌入式控制器启动引导程序之后依次启动多个模块，各个模块包括：检测单元，其使用用于验证接下来要启动的模块的签名的验证信息，来检测对接下来要启动的模块的修改，以及启动单元，其在检测单元进行的签名验证成功的情况下，启动接下来要启动的模块，并且其中，各个模块预先保持验证信息以及其自身的签名。

[0006] 本发明的另一方面，提供一种信息处理装置的控制方法，所述信息处理装置能够被操作以在通过嵌入式控制器启动引导程序之后依次启动多个模块，其中，各个模块包括：使用用于验证接下来要启动的模块的签名的验证信息，来检测对接下来要启动的模块的修改，以及在签名验证成功的情况下，启动接下来要启动的模块。

[0007] 本发明的又一方面，提供一种计算机可读存储介质，其存储用于使计算机执行信息处理装置的控制方法的各个步骤的计算机程序，所述信息处理装置能够被操作以在通过

嵌入式控制器启动引导程序之后依次启动多个模块，其中，各个模块包括：使用用于验证接下来要启动的模块的签名的验证信息，来检测对接下来要启动的模块的修改，以及在签名验证成功的情况下，启动接下来要启动的模块。

[0008] 通过下面参照附图对示例性实施例的描述，本发明的其他特征将变得清楚。

附图说明

[0009] 图1是根据实施例的多功能外围设备的硬件构造图。

[0010] 图2A和图2B是根据实施例的多功能外围设备的软件构造图。

[0011] 图3A至图3E是例示根据实施例的启动时的操作的模式图。

[0012] 图4是用于描述根据实施例的处理过程的流程图。

[0013] 图5是用于描述根据实施例的处理过程的流程图。

[0014] 图6是用于描述根据实施例的处理过程的流程图。

具体实施方式

[0015] 现在将参照附图详细描述本发明的优选实施例。应当注意，除非另有具体说明，否则这些实施例中阐述的部件的相对布置、数值表达式和数值不限制本发明的范围。

[0016] <第一实施例>

[0017] 下面参照附图给出关于本发明的第一实施例的描述。注意，以下实施例不旨在限制本发明的权利要求的范围，并且并非根据以下实施例描述的特征的所有组合都是解决根据本发明的问题的手段所必需的。另外，通过将多功能外围设备(数字多功能外围设备/MFP)作为根据实施例的信息处理装置的示例给出描述。然而，应用范围不限于多功能外围设备，并且应用于任何信息处理装置。

[0018] <信息处理装置的硬件构造>

[0019] 首先，参照图1，描述嵌入式控制器113和作为根据本实施例的信息处理装置的多功能外围设备100的硬件构造。多功能外围设备100配设有CPU 101、ROM(只读存储器(Read-Only Memory))102、RAM(随机存取存储器(Random Access Memory))103、HDD(硬盘驱动器(Hard Disk Drive))104、网络I/F控制单元105、扫描器I/F控制单元106、打印机I/F控制单元107、面板控制单元108、扫描器111、打印机112、嵌入式控制器113、闪存存储器(flash memory)114和LED 117。此外，嵌入式控制器113配设有CPU 115和RAM 116。

[0020] CPU 101执行多功能外围设备100的软件程序以综合地控制设备整体。ROM 102是只读存储器，并且存储多功能外围设备100的BIOS(基本输入/输出系统(Basic Input/Output System))、固定参数等。RAM 103是随机存取存储器，并且用于例如在CPU 101控制多功能外围设备100时存储程序或临时数据。HDD 104是硬盘驱动器，并且存储一些应用程序和各种数据。闪存存储器114存储诸如加载器、内核和应用程序等的各种模块。

[0021] 嵌入式控制器113的CPU 115执行嵌入式控制器113的软件程序以在多功能外围设备100中进行一些控制。RAM 116是随机存取存储器，并且用于例如当CPU 115控制多功能外围设备100时存储程序或临时数据。对于嵌入式控制器113，多功能外围设备100配设有用于综合控制系统的主控制器。主控制器被构造为至少包括CPU 101、ROM 102和RAM 103。

[0022] 网络I/F控制单元105控制利用网络118的数据发送和接收。扫描器I/F控制单元

106进行控制以通过扫描器111读取原稿。打印机I/F控制单元107控制打印机112进行的打印处理等。面板控制单元108控制作为触摸面板的操作面板110，显示各种信息，并控制从用户输入的指令。总线109将CPU 101、ROM 102、RAM 103、HDD 104、网络I/F控制单元105、扫描器I/F控制单元106和打印机I/F控制单元107彼此连接。此外，总线109还将面板控制单元108、嵌入式控制器113和闪存存储器114彼此连接。来自CPU 101的控制信号或装置之间的数据信号经由总线109发送和接收。LED 117根据需要点亮，并用于向外部传送软件或硬件的异常。

[0023] <信息处理装置的软件构造>

[0024] 接下来，参照图2A，描述由根据本实施例的多功能外围设备100保持的软件模块。多功能外围设备100包括嵌入式控制器113内的引导程序209，作为软件模块。此外，多功能外围设备100包括BIOS 210、加载器211、内核212、Native程序213、Java(注册商标)程序214、UI控制单元203和通信管理单元207。

[0025] 通信管理单元207控制连接到网络118的网络I/F控制单元105，网络I/F控制单元105经由网络118与外部单元进行数据的发送和接收。UI控制单元203接收经由面板控制单元108向操作面板110的输入，并进行根据该输入的处理或向操作面板110的画面输出。

[0026] 引导程序209是在接通多功能外围设备100的电源时由嵌入式控制器113的CPU 115执行的程序，并且除了执行与启动相关的处理之外，还包括用于检测对BIOS的修改的BIOS修改检测单元201。BIOS 210是在执行引导程序209之后由CPU 101执行的程序，并且除了执行与启动有关的处理之外，还包括用于检测对加载器211的修改的加载器修改检测单元202。

[0027] 加载器211是在BIOS 210的处理完成之后由CPU 101执行的程序，并且除了执行与启动有关的处理之外，还具有用于检测对内核的修改的内核修改检测单元204。内核212是在加载器211的处理完成之后由CPU 101执行的程序，并且除了执行与启动有关的处理之外，还具有用于检测对Native程序213的修改的Native修改检测单元205。

[0028] Native程序213是由CPU 101执行的程序，并且包括用于与多功能外围设备100的Java(注册商标)程序214协作地提供各种功能的多个程序。例如，Native程序213包括用于控制扫描器I/F控制单元106或打印机I/F控制单元107的引导程序或程序。内核212从Native程序调用引导程序，并执行启动处理。另外，作为Native程序213的程序之一，Native程序213具有用于检测对Java程序的修改的Java程序修改检测单元。

[0029] Java程序214是由CPU 101执行的程序，并且是与多功能外围设备100的Native程序213协作地提供各种功能的程序。例如，Java程序214是用于在操作面板110上显示画面的程序。

[0030] <启动过程>

[0031] 下面参照图3A和图3B描述用于启动多功能外围设备100的过程。图3A例示了在不进行修改检测的情况下启动多功能外围设备100的顺序。引导程序209启动BIOS 210，BIOS 210启动加载器211，加载器211启动内核212，并且内核212启动来自从Native程序213的引导程序。引导程序启动Java程序214，随后，通过Native程序213和Java程序214协作来提供多功能外围设备100的功能。以这种方式以预定顺序进行各个模块的启动控制，并且当前一模块的启动完成时执行后续模块的启动处理。

[0032] 图3B例示了在进行修改检测的同时启动多功能外围设备100的顺序。如图所示，在进行修改检测的同时按照如下顺序进行启动：引导程序209，然后是BIOS 210、加载器211、内核212、Native程序213和Java程序214。由紧接在前面启动的模块进行对要启动的模块的修改检测。例如，由引导程序209进行对BIOS 210的修改检测。另外，图3B表示各个程序、数字签名(以下称为签名)的存储位置和用于验证签名的公钥(验证信息)的存储位置。

[0033] 下面假设引导程序209和BIOS 210存储在ROM 102中，并且加载器211、内核212和Native程序(第一程序)213存储在闪存存储器114中。此外，假设Java程序214(第二程序)存储在HDD 104中。

[0034] BIOS签名验证公钥(public key)300存储在引导程序209中，并且BIOS签名302和加载器验证公钥303存储在BIOS 210中。加载器签名304和内核验证公钥305存储在加载器211中。另外，内核签名306和Native程序验证公钥307存储在内核212中，并且Native程序签名309和Java程序验证公钥308存储在Native程序213中。此外，Java程序签名310存储在Java程序214中。期望在多功能外围设备100的出厂之前预先将这些公钥和签名分配给程序。在根据本实施例的多功能外围设备100中，附图标记为201、202和204至206的各检测单元通过验证接下来要启动的各程序(各模块)来进行修改检测。并且在没有问题的情况下，启动接下来的程序。

[0035] 〈处理过程〉

[0036] 接下来，参照图4，描述在启动根据本实施例的多功能外围设备100时的处理过程。当接入多功能外围设备100的电源时，引导程序209从ROM 102被读入RAM 116，并由CPU 115执行。

[0037] 在步骤S401中，引导程序209中包括的BIOS修改检测单元201对BIOS进行签名验证，并确定签名验证是否成功。具体地，BIOS修改检测单元201将BIOS 210、加载器验证公钥303和BIOS签名302从闪存存储器114读取到RAM 116中。此外，BIOS修改检测单元201使用BIOS验证公钥300进行BIOS签名302的验证并确定验证是否成功。当签名验证失败时，处理进行到步骤S403，并且BIOS修改检测单元201使LED 117点亮并结束处理。同时，如果签名验证成功，则BIOS修改检测单元201将电流施加到CPU 101，并且引导程序的处理结束。随后，流程图转到由CPU 101执行的步骤S402及其后的处理。

[0038] 当施加电流时，在步骤S402中，CPU 101将BIOS 210和加载器验证公钥303从闪存存储器114读入RAM 103，并启动BIOS 210。后续处理全部描述为由CPU 101进行的处理。

[0039] 当启动BIOS 210时，处理转到步骤S404。在步骤S404中，BIOS 210执行各种初始化处理，并且BIOS 210中包括的加载器修改检测单元202将加载器211、内核验证公钥305和加载器签名304从闪存存储器114读入RAM 103中。此外，加载器修改检测单元202使用加载器验证公钥305来进行对加载器签名304的验证，并确定验证是否成功。当签名验证失败时，处理前进到步骤S412，并且加载器修改检测单元202在操作面板110上显示错误消息，并且处理结束。同时，当签名验证成功时，加载器修改检测单元204结束处理，前进到步骤S405，并且BIOS210启动读入RAM 103中的加载器211。

[0040] 当启动加载器211时，处理转到步骤S406。在步骤S406中，加载器211执行各种初始化处理，并且加载器211中包括的内核修改检测单元204将内核212、Native程序验证公钥307和内核签名306从闪存存储器114读入RAM 103。此外，在步骤S406中，内核修改检测单元

204使用内核验证公钥305进行对内核签名306的验证，并确定验证是否成功。当签名验证失败时，处理前进到步骤S412，并且内核修改检测单元204在操作面板110上显示错误消息，并且处理结束。同时，当签名验证成功时，内核修改检测单元204结束处理，前进到步骤S407，并且加载器211启动读入RAM 103的内核212。

[0041] 当启动内核212时，处理转到步骤S408。在步骤S408中，内核212执行各种初始化处理。此外，内核212中包括的程序修改检测单元205将Native程序213、Java程序验证公钥308和Native程序签名309从闪存存储器114读入RAM 103中。在步骤S408中，程序修改检测单元205使用Native程序验证公钥307来进行对Native程序签名309的验证，并确定验证是否成功。当签名验证失败时，在步骤S412中，程序修改检测单元205在操作面板110上显示错误消息，并且处理结束。同时，当签名验证成功时，程序修改检测单元205结束处理并在步骤S409中启动Native程序213。

[0042] 当启动Native程序213中的、用于执行修改检测处理的Java程序修改检测单元206时，处理转到步骤S410。在步骤S410中，Java程序修改检测单元206将Java程序214和Java程序签名310从HDD 104读入RAM 103。在步骤S410中，Java程序修改检测单元206使用Java程序验证公钥308进行对Java程序签名310的验证，并确定验证是否成功。当签名验证失败时，处理进行到步骤S412，并且Java程序修改检测单元206在操作面板110上显示错误消息，并且处理结束。同时，当签名验证成功时，Java程序修改检测单元206结束处理，并且Native程序213在步骤S411中启动Java程序214。

[0043] 如上所述，根据本实施例的信息处理装置在启动引导程序之后依次启动多个模块。各个模块使用用于验证接下来要启动的模块的签名的验证信息，来检测对接下来要启动的模块的修改，并且当验证签名成功时启动接下来要启动的模块。此外，各个模块预先保持验证信息及其自身签名。由此，即使没有TPM构造，本信息处理装置也可以检测修改，而不管是BIOS 210、加载器211、内核212、Native程序213还是Java程序214中的哪一个被修改。另外，因为紧接在启动各个软件程序之前进行修改检测并且如果发现修改就立即停止启动，因此能够保证多功能外围设备100始终仅在安全状态下操作，并且被修改的软件程序连一瞬间都不会操作。另外，根据另一方面，本信息处理装置可以提供用于检测系统中的修改的更具鲁棒性的机制。

[0044] <第二实施例>

[0045] 下面，将描述本发明的第二实施例。在上述第一实施例中，存在HDD 104存储Java程序的构造。因为HDD 104是用于数据存储的区域，所以它可以被读取或写入，并且被修改的可能性最高。在上述第一实施例中，因为当存在修改时停止多功能外围设备100的启动，因此存在可用性降低的问题。因此，在本实施例中，描述即使存储在HDD 104中的Java程序被修改也不会降低可用性的方法。

[0046] <启动过程>

[0047] 接下来，参照图3C，描述本实施例中的各个程序、签名、公钥的存储位置和启动顺序。如图3C所示，用于启动各个模块的过程类似于在上述第一实施例中描述的图3B。然而，在本实施例中，与上述第一实施例不同，除了存储在HDD 104中之外，Java程序还存储在闪存存储器114中。此外，与Java程序214的签名310相同的签名330与Java程序331一起存储在闪存存储器114中。只要签名330不被修改，它就与存储在HDD 104中的签名310相同。

[0048] 〈处理过程〉

[0049] 接下来，参照图5，描述在启动根据本实施例的多功能外围设备100时的处理过程。因为步骤S401至步骤S407、步骤S409、步骤S410和步骤S412具有与上述第一实施例类似的处理，因此省略其描述。

[0050] 在步骤S407中，当启动内核212时，执行各种初始化处理，并且处理进行到步骤S501。此外，内核212中包括的程序修改检测单元205将Native程序213和Java程序214从闪存存储器114读入RAM 103中。此外，程序修改检测单元205将Java程序验证公钥308、Native程序签名309、以及Java程序331的签名330从闪存存储器114读入RAM 103中。

[0051] 随后，在步骤S501中，程序修改检测单元205使用Native程序验证公钥307来进行对Native程序213的签名309的验证。此外，程序修改检测单元205使用Java程序验证公钥308进行对Java程序331的签名330的验证并确定两个签名验证是否成功。当两个签名验证都成功时，处理转到步骤S409。同时，当任一签名验证失败时，处理转到步骤S412。

[0052] 当步骤S409的处理结束并且在步骤S410中确定Java程序214的签名验证失败时，处理转到步骤S502。在步骤S502中，Java程序修改检测单元206从HDD 104中删除Java程序214，并执行重新展开处理以将存储在闪存存储器114中的Java程序331及其签名330写入HDD 104。当重新展开处理结束时，Java程序修改检测单元206结束处理，处理进行到步骤S411，并且启动Java程序214。

[0053] 如上所述，借助于本实施例，Java程序除了被保持在HDD 104中之外，还保持在闪存存储器114中。因此，即使修改了被修改的可能性高的HDD 104的Java程序，由于通过重新展开闪存存储器114中的Java程序331来恢复修改，因此也能够避免可用性的降低。

[0054] 〈第三实施例〉

[0055] 下面，将说明本发明的第三实施例。如图3D所示，存在这样的情况：所采用的构造是多功能外围设备100保持多个内核和程序，并且要启动的内核和程序被加载器替换。利用这样的构造，尝试在上述第一或第二实施例的构造下启动内核B 220而不是内核212的情况下，存在如下问题：因为没有内核B 220的签名，尽管内核B 220没有被修改，也会将其检测为被修改，并且不会启动。因此，在本实施例中，描述即使利用保持不同内核和不同程序的构造也进行修改检测和启动的方法。

[0056] 〈软件构造〉

[0057] 首先，参照图2B，描述由根据本实施例的多功能外围设备100保持的软件模块的构造的示例。因为附图标记201至214与图2A中的相同，所以省略其描述。

[0058] 加载器223是在BIOS 210的处理完成之后由CPU 101执行的程序，并且除了执行与启动有关的处理之外，还具有用于检测对内核的修改的内核修改检测单元224。加载器223根据经由操作面板110的用户输入替换要启动的内核。

[0059] 内核B 220是由CPU 101执行的、与内核212不同的程序，其执行与启动有关的处理，并且还具有用于对Native程序B 222进行修改检测的程序修改检测单元221。Native程序B 222是由CPU 101执行的程序，并且为多功能外围设备100提供更新功能。Native程序B 222由内核B 220调用，并提供用于更新内核212、Native程序213或Java程序214的功能。注意，Native程序B 222不限于更新功能，并且可以是提供其他功能的程序。

[0060] 〈启动过程〉

[0061] 接下来，参照图3E，描述如下处理流程：可以根据加载器223要启动的内核是内核212还是内核B 220来替换和启动修改检测的目标。

[0062] 假设加载器223包括加载器223的签名304、用于验证内核212的公钥305以及用于验证内核B 220的公钥340。此外，内核B的签名341和用于验证Native程序B的公钥342包括在内核B 220中，并且用于Native程序B的签名343包括在Native程序B 222中。期望在多功能外围设备100出厂之前将这些公钥和签名预先分配给程序。以这种方式，加载器223包括用于接下来可以启动的多个内核的各公钥(验证信息)。

[0063] 〈处理过程〉

[0064] 接下来，参照图6，描述在启动根据本实施例的多功能外围设备100时的处理过程。因为步骤S401至步骤S404和步骤S406至步骤S412的处理类似于图4中的处理，所以省略其描述。

[0065] 在步骤S600中，当启动加载器223时，进行各种初始化处理。接下来，在步骤S601中，加载器223确定内核B 220是否经由操作面板110被选为启动目标。如果尚未选择内核B 220作为启动目标，则转到步骤S406的处理。同时，当选择内核B 220时，加载器223中包括的内核修改检测单元224将内核B 220、用于验证Native程序B的公钥342和内核B的签名341从闪存存储器114读入RAM 103。在步骤S602中，内核修改检测单元224使用用于验证内核B的公钥342来对内核B的签名341进行验证，并确定验证是否成功。如果签名验证失败，则处理转到步骤S412。同时，当签名验证成功时，内核修改检测单元221结束处理，并且在步骤S603中，加载器223启动被读入到RAM 103的内核B 220。

[0066] 当启动内核B 220时，进行各种初始化处理，并且内核B 220中包括的程序修改检测单元221将Native程序B 222和用于Native程序B的签名343从闪存存储器114读入RAM 103。在步骤S604中，程序修改检测单元221使用用于验证Native程序B的公钥342来对Native程序B的签名343进行验证，并确定验证是否成功。如果签名验证失败，则处理转到步骤S412。同时，当签名验证成功时，程序修改检测单元221结束处理并在步骤S605中启动Native程序B 222。在启动Native程序B 222时，向用户提供更新功能。

[0067] 如上所述，借助于本实施例，即使具有保持多个内核和多个程序的构造，也能够在对多个内核和多个程序进行修改检测之后进行启动，并且能够实现类似于上述第一和第二实施例的效果。

[0068] 〈变型例〉

[0069] 本发明不限于上述实施例，并且可以有各种变化。在上述第一至第三实施例中，公钥被描述为彼此不同，但它们可以是相同的。另外，描述了ROM 102、闪存存储器114和HDD 104作为各程序的存储位置，但这不是对存储位置的限制，并且存储位置可以是不同的存储介质。另外，程序的存储位置不需要是描述的位置，例如，可以采用将加载器223存储在ROM 102中的构造。

[0070] 其他实施例

[0071] 还可以通过读出并执行记录在存储介质(也可更完整地称为“非暂时性计算机可读存储介质”)上的计算机可执行指令(例如，一个或更多个程序)以执行上述实施例中的一个或更多个的功能，和/或包括用于执行上述实施例中的一个或更多个的功能的一个或更多个电路(例如，专用集成电路(ASIC))的系统或装置的计算机，来实现本发明的实施例，并

且,可以利用通过由系统或装置的计算机例如读出并执行来自存储介质的计算机可执行指令以执行上述实施例中的一个或更多个的功能,并且/或者控制一个或更多个电路以执行上述实施例中的一个或更多个的功能的方法,来实现本发明的实施例。计算机可以包括一个或更多个处理器(例如,中央处理单元(CPU)、微处理单元(MPU)),并且可以包括分开的计算机或分开的处理器的网络,以读出并执行计算机可执行指令。计算机可执行指令可以例如从网络或存储介质被提供给计算机。存储介质可以包括例如硬盘、随机存取存储器(RAM)、只读存储器(ROM)、分布式计算系统的存储器、光盘(诸如压缩光盘(CD)、数字通用光盘(DVD)或蓝光光盘(BD)™)、闪存存储器装置以及存储卡等中的一个或更多个。

[0072] 本发明的实施例还可以通过如下的方法来实现,即,通过网络或者各种存储介质将执行上述实施例的功能的软件(程序)提供给系统或装置,该系统或装置的计算机或是中央处理单元(CPU)、微处理单元(MPU)读出并执行程序的方法。

[0073] 虽然已经参照示例性实施例对本发明进行了描述,但是应该理解,本发明不限于所公开的示例性实施例。应当对权利要求的范围给予最宽的解释,以使其涵盖所有这些变型例以及等同的结构及功能。

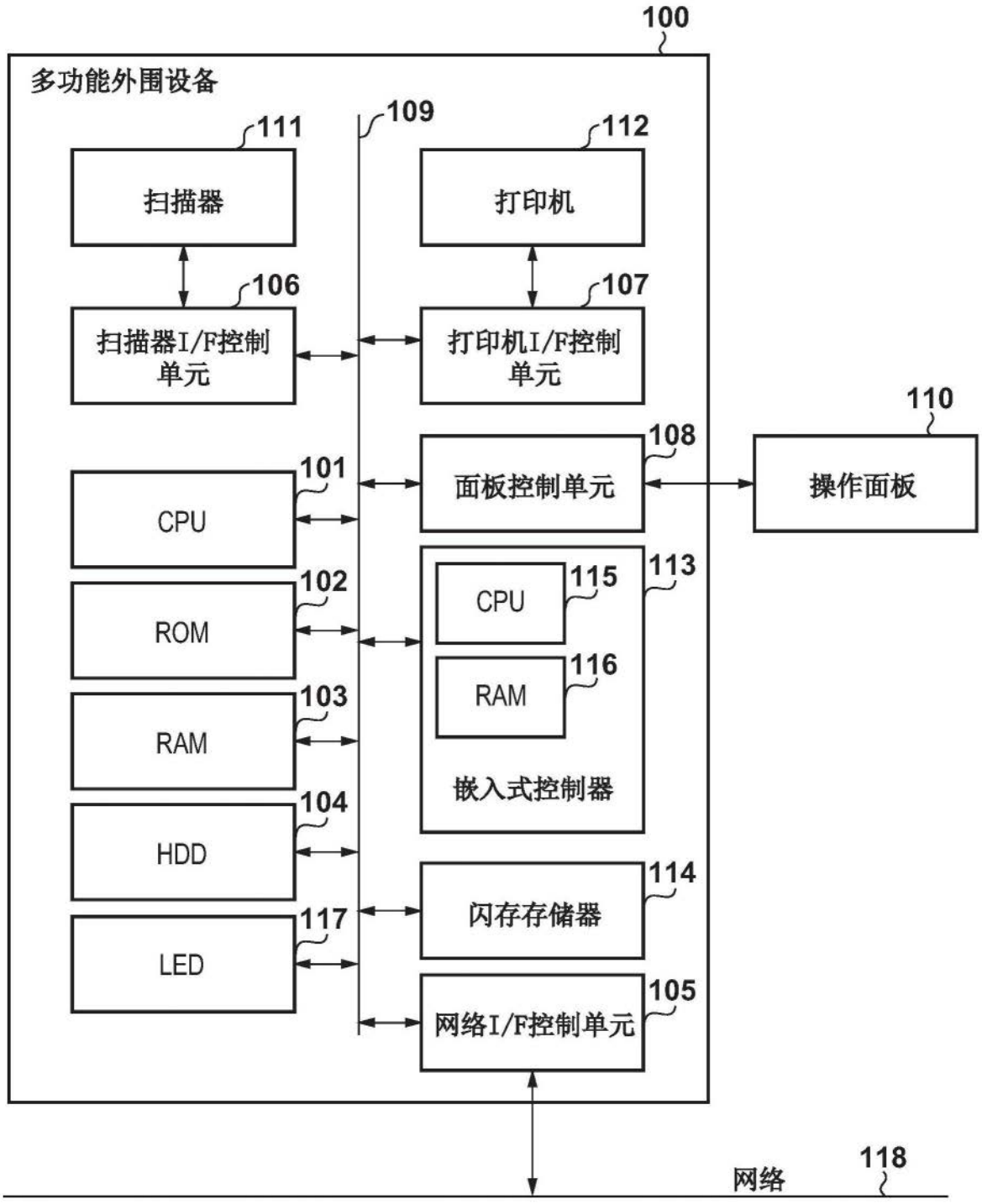

100
多功能外围设备
111
扫描器
109
112
打印机
106
扫描器I/F控制单元
107
打印机I/F控制单元
110
108
面板控制单元
操作面板
101
CPU
102
ROM
103
RAM
104
HDD
117
LED
113
CPU
115
RAM
116
嵌入式控制器
114
闪存存储器
105
网络I/F控制单元
网络
118

图1

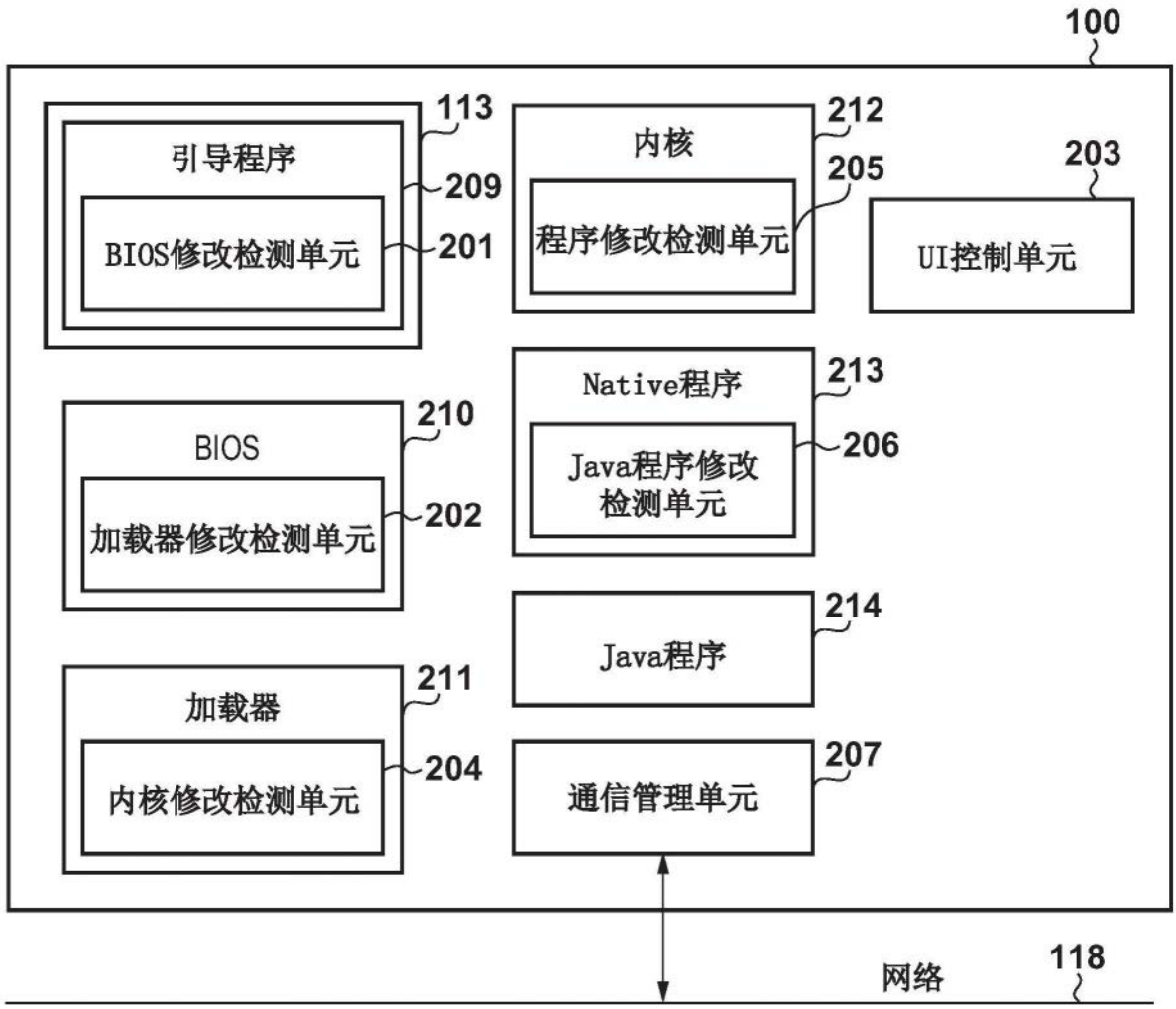

100
113
引导程序
209
BIOS修改检测单元
201
212
内核
205
程序修改检测单元
203
UI控制单元
Native程序
213
Java程序修改
检测单元
206
210
BIOS
加载器修改检测单元
202
214
Java程序
211
加载器
内核修改检测单元
204
207
通信管理单元
网络
118

图2A

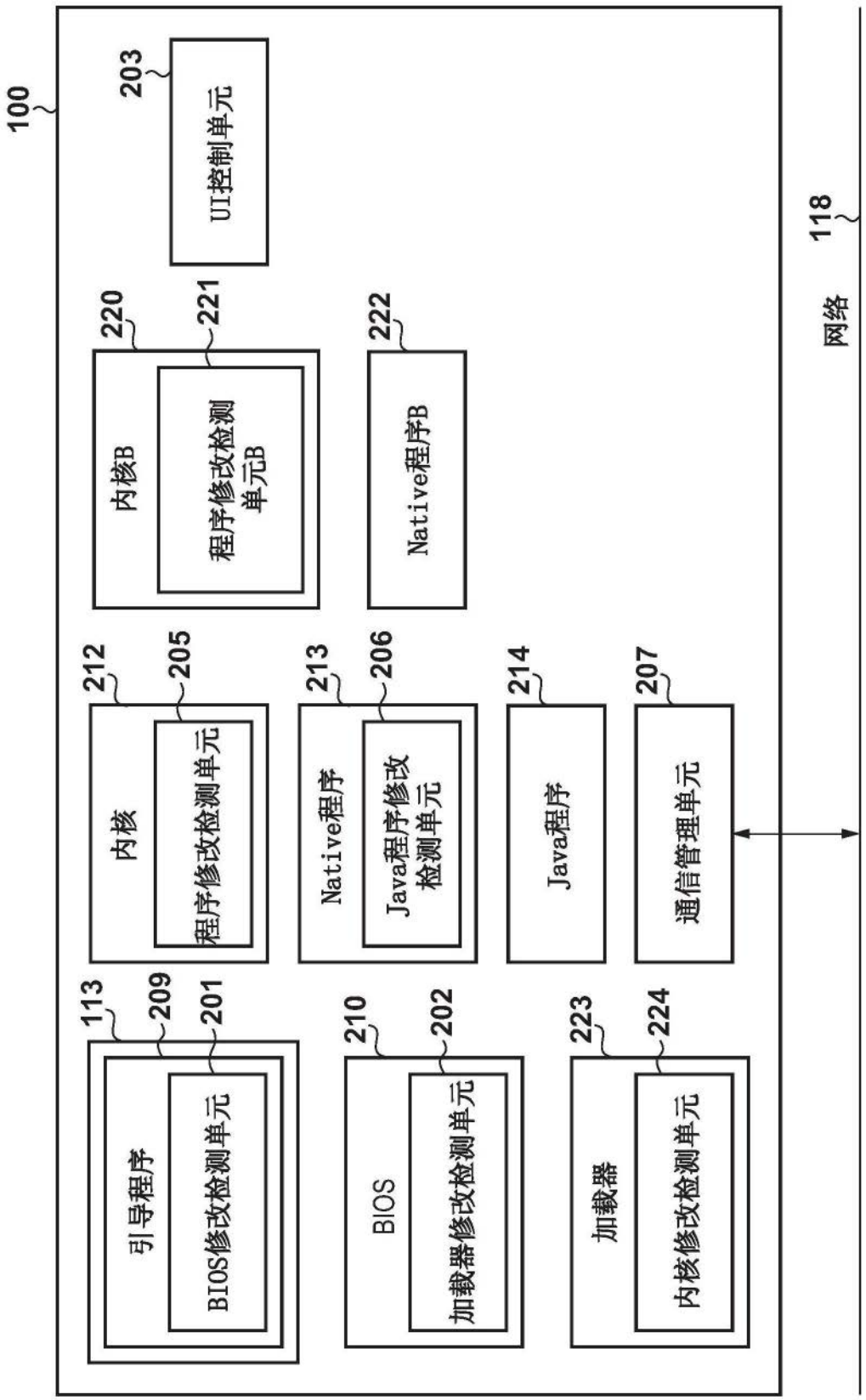

100
203
UI控制单元
118
网络
220
221
内核B
程序修改检测单元B
222
Native程序B
212
205
内核
程序修改检测单元
213
206
Native程序
Java程序修改检测单元
214
Java程序
207
通信管理单元
113
209
201
引导程序
BIOS修改检测单元
210
202
BIOS
加载器修改检测单元
223
224
加载器
内核修改检测单元

图2B

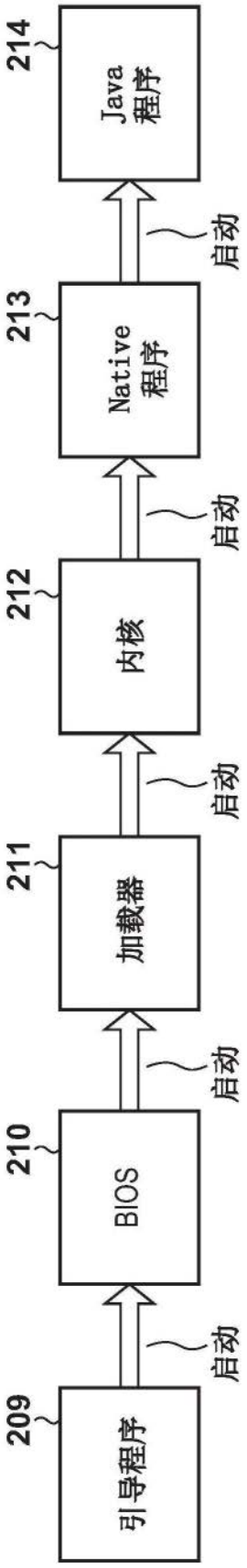

214
Java
程序
启动
213
Native
程序
启动
212
内核
启动
211
加载器
启动
210
BIOS
启动
209
引导程序

图3A

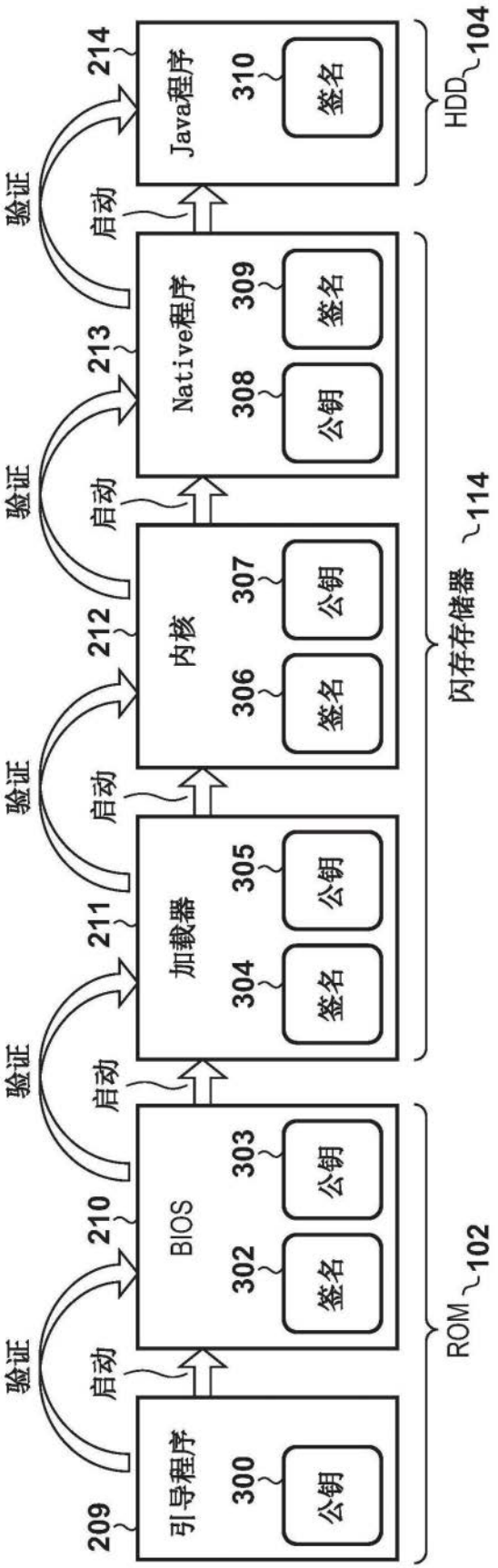

209
引导程序
300
公钥
验证
启动
210
BIOS
302
签名
303
公钥
ROM
102
211
加载器
304
签名
305
公钥
212
内核
306
签名
307
公钥
213
Native程序
308
公钥
309
签名
214
Java程序
310
签名
HDD
104
闪存存储器
114

图3B

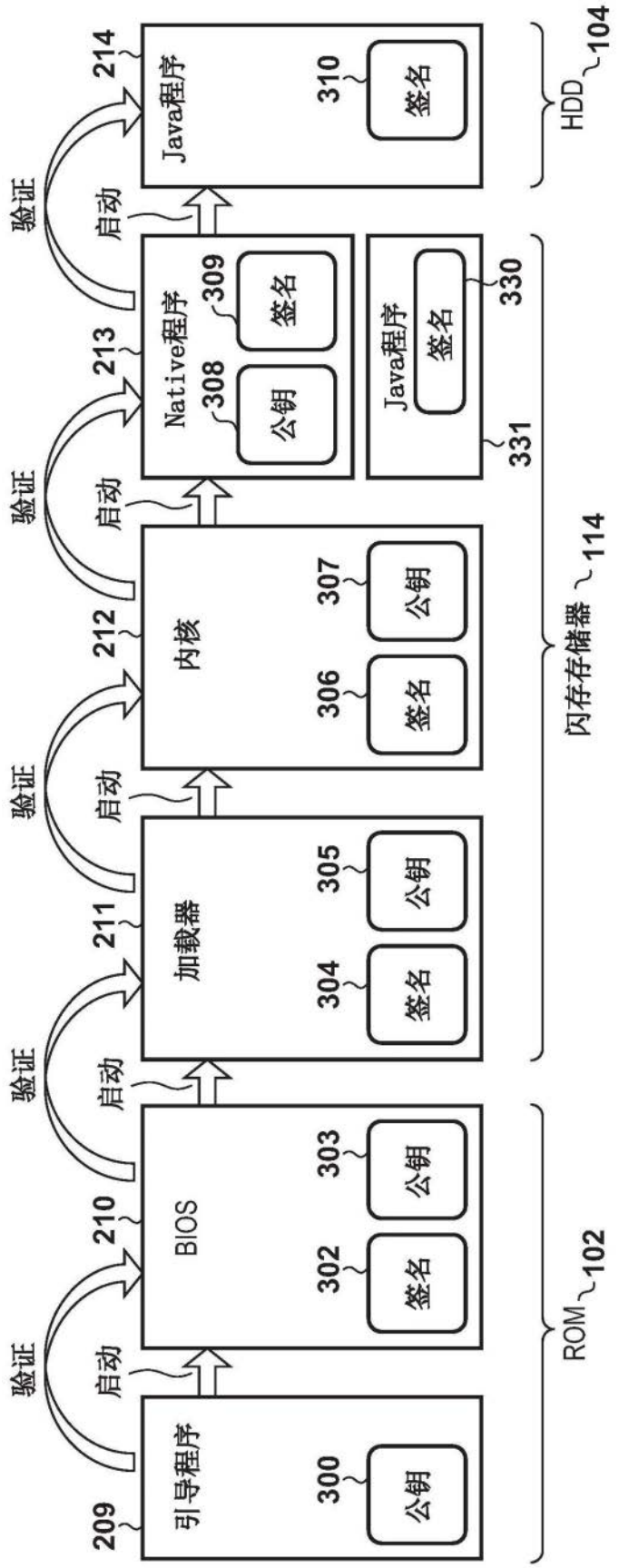

209
引导程序
300
公钥
验证
启动
210
BIOS
302
签名
303
公钥
211
加载器
304
签名
305
公钥
212
内核
306
签名
307
公钥
213
Native程序
308
公钥
309
签名
214
Java程序
310
签名
331
Java程序
签名
330
ROM
102
闪存存储器
114
HDD
104

图3C

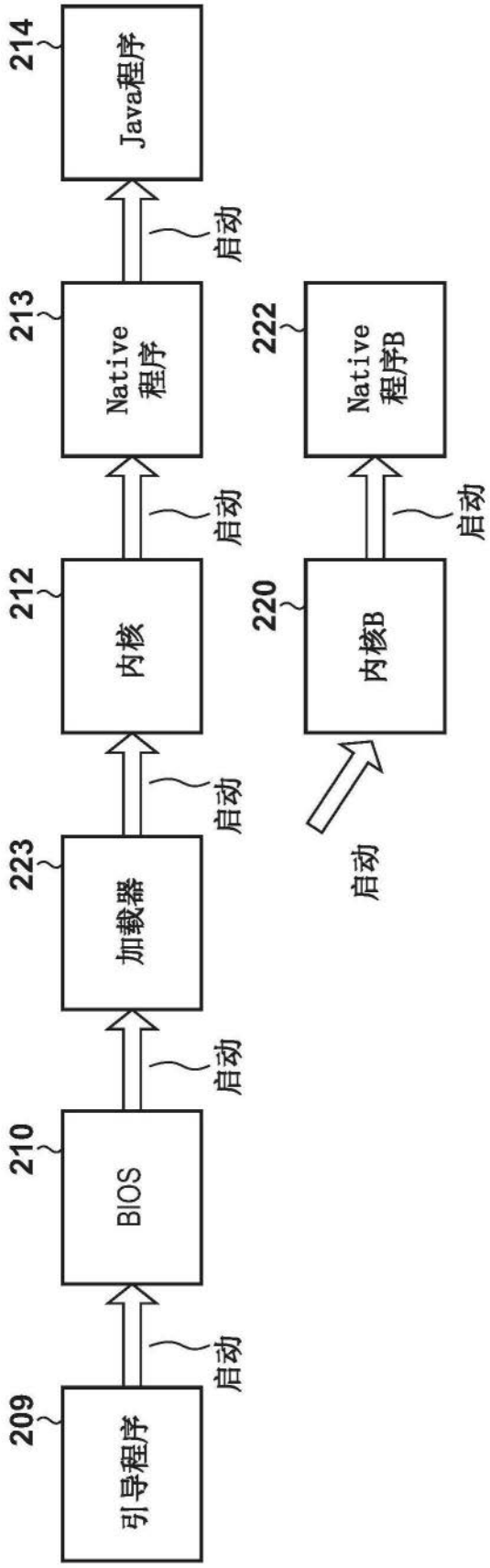

209
引导程序
启动
210
BIOS
启动
223
加载器
启动
212
内核
启动
213
Native
程序
启动
214
Java程序
220
内核B
启动
222
Native
程序B
启动

图3D

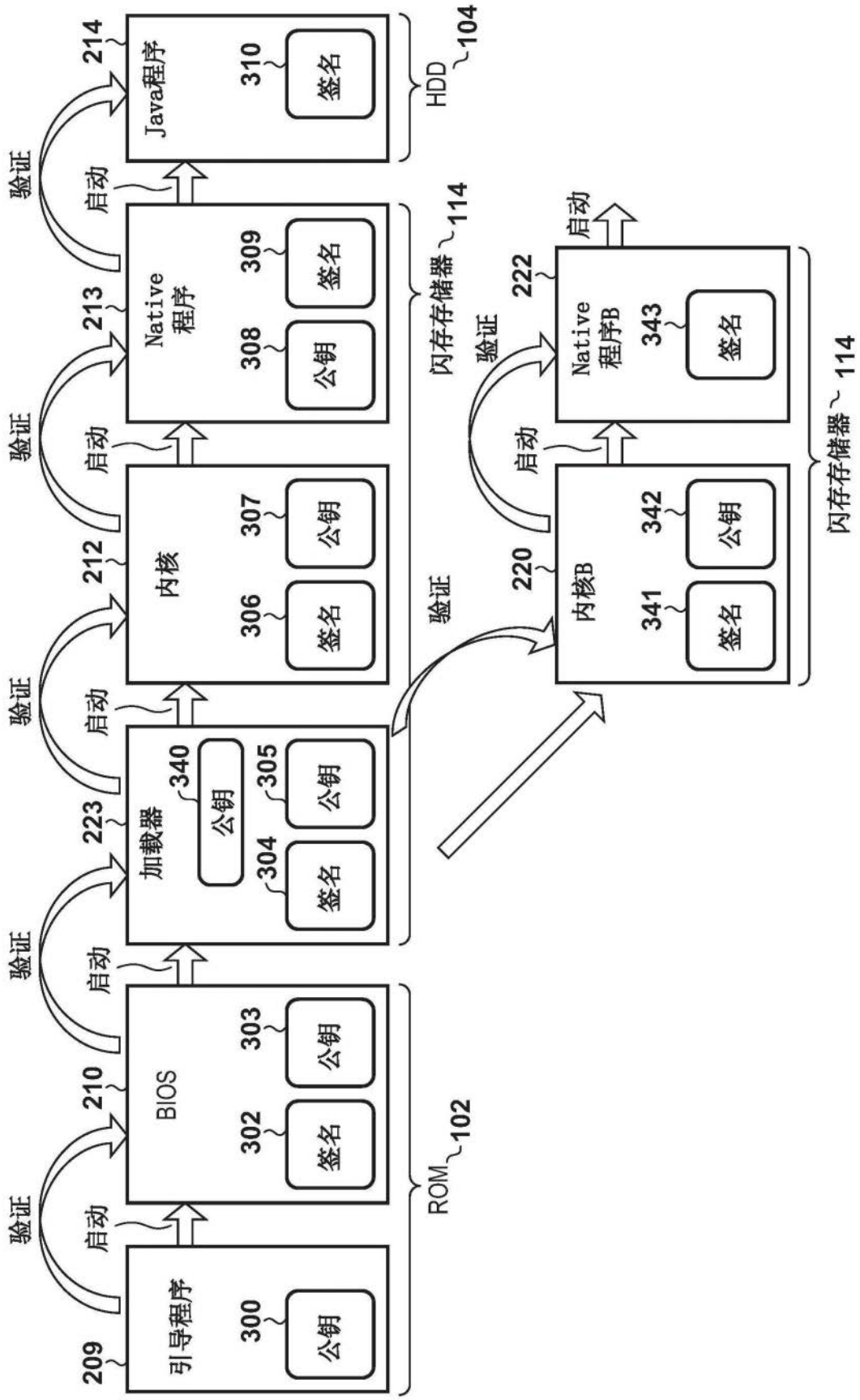

ROM
102
HDD
104
闪存存储器
114
闪存存储器
114
209
引导程序
300
公钥
210
BIOS
302
签名
303
公钥
223
加载器
304
签名
340
公钥
305
公钥
212
内核
306
签名
307
公钥
213
Native
程序
308
公钥
309
签名
214
Java程序
310
签名
220
内核B
341
签名
342
公钥
222
Native
程序B
343
签名
验证
启动

图3E

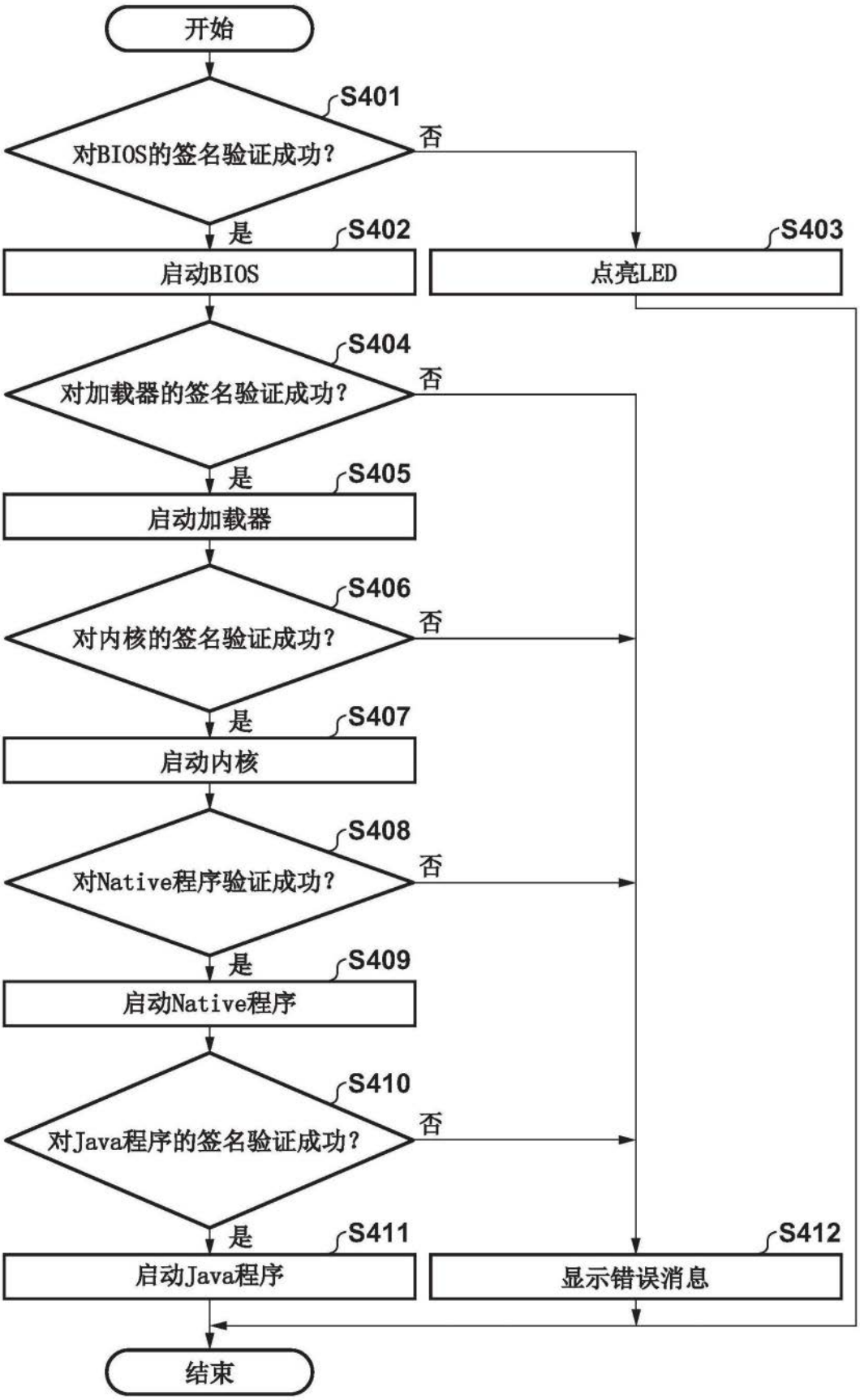

开始
S401
对BIOS的签名验证成功?
否
是
S402
启动BIOS
S403
点亮LED
S404
对加载器的签名验证成功?
否
是
S405
启动加载器
S406
对内核的签名验证成功?
否
是
S407
启动内核
S408
对Native程序验证成功?
否
是
S409
启动Native程序
S410
对Java程序的签名验证成功?
否
是
S411
启动Java程序
S412
显示错误消息
结束

图4

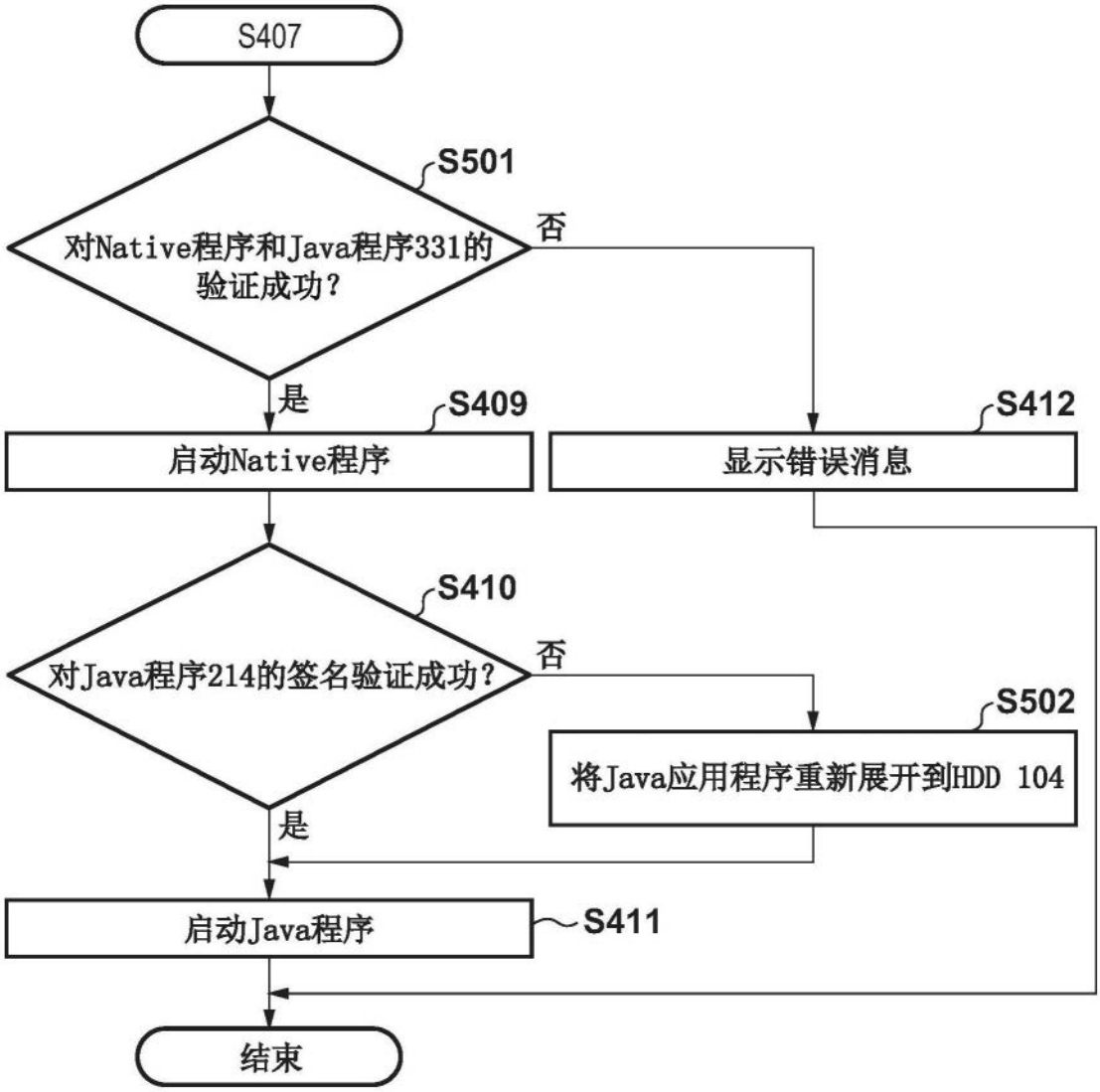

S407
S501
对Native程序和Java程序331的验证成功?
否
是
S409
启动Native程序
S412
显示错误消息
S410
对Java程序214的签名验证成功?
否
S502
将Java应用程序重新展开到HDD 104
是
启动Java程序
S411
结束

图5

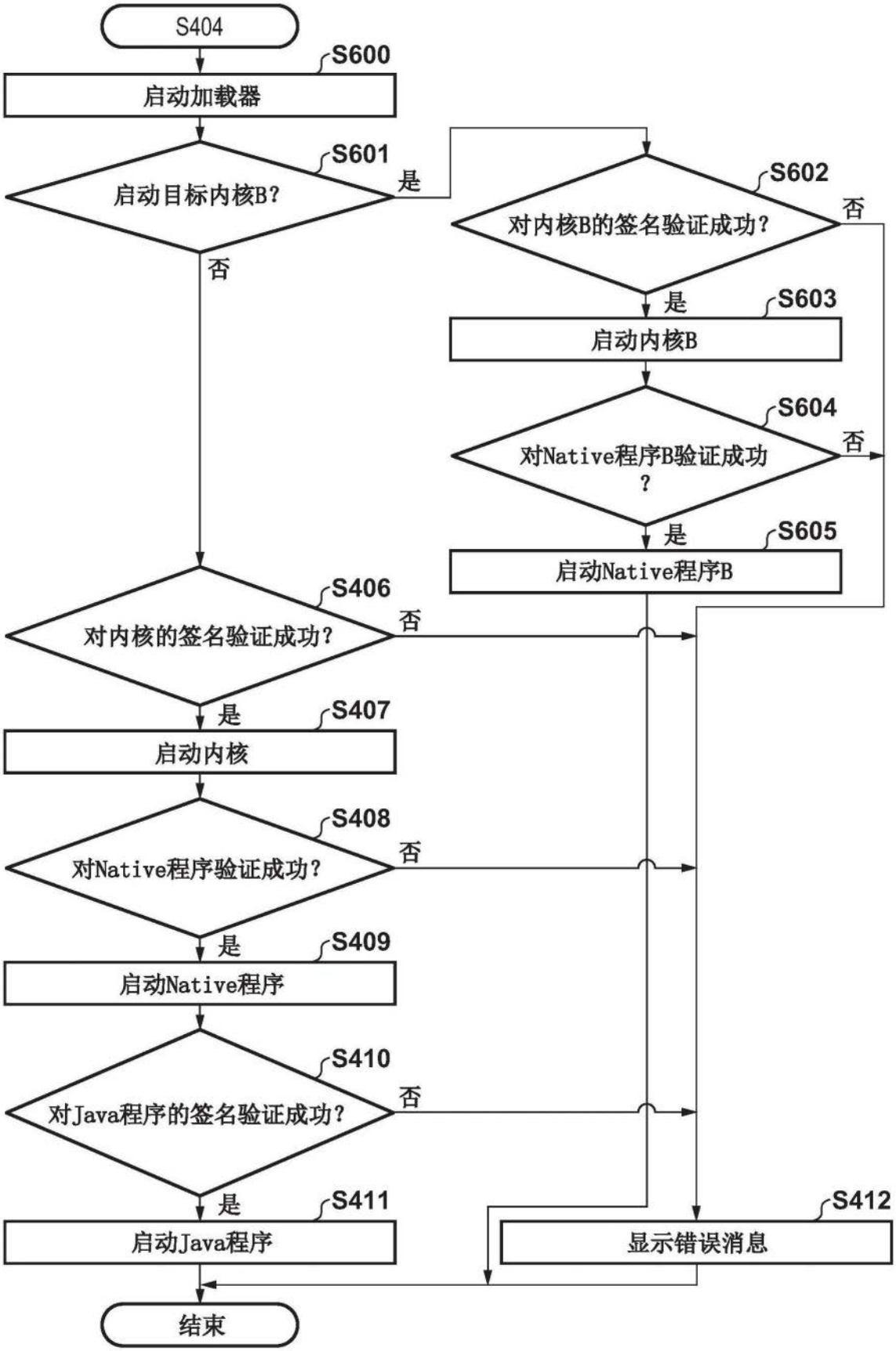

S404
S600
启动加载器
S601
启动目标内核B?
是
否
S602
对内核B的签名验证成功?
否
是
S603
启动内核B
S604
对Native程序B验证成功?
否
是
S605
启动Native程序B
S406
对内核的签名验证成功?
否
是
S407
启动内核
S408
对Native程序验证成功?
否
是
S409
启动Native程序
S410
对Java程序的签名验证成功?
否
是
S411
启动Java程序
S412
显示错误消息
结束

图6