(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102682244 B
(45)授权公告日 2016.12.07

(21)申请号 201210020096.9

(22)申请日 2012.01.21

(65)同一申请的已公布的文献号
申请公布号 CN 102682244 A

(43)申请公布日 2012.09.19

(30)优先权数据
2011-027652 2011.02.10 JP

(73)专利权人 索尼公司
地址 日本东京都

(72)发明人 森田直

(74)专利代理机构 北京市柳沈律师事务所 11105
代理人 郭定辉

(51)Int.Cl.
G06F 21/77(2013.01)

(56)对比文件
KR 10-1012872 B1,2011.02.08,
KR 10-1012872 B1,2011.02.08,
US 2004/0227652 A1,2004.11.18,
US 2008/0127113 A1,2008.05.29,
US 2010/0246829 A1,2010.09.30,
US 2008/0301814 A1,2008.12.04,

审查员 李莎

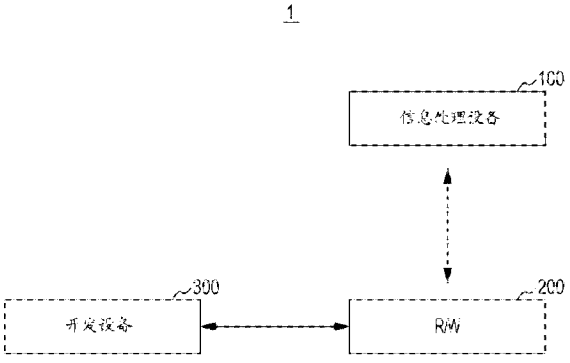
权利要求书2页 说明书15页 附图14页

(54)发明名称

信息处理设备和程序执行方法

(57)摘要

提供了信息处理设备、程序执行方法和计算机。该信息处理设备包括:程序执行单元,在具有防篡改性能的环境下,解释并且执行以过程语言创建的计算机程序的代码,其中,以在由该程序执行单元执行的该计算机程序内的函数为单位设置安全属性和验证密钥,以及其中该程序执行单元利用验证密钥执行验证处理以执行该函数,这样使得可以根据该安全属性执行该函数。



CN 102682244 B

1.一种信息处理设备，包括：

程序执行单元，在具有防篡改性能的环境下，解释并且执行以过程语言创建的计算机程序的代码，

其中，以在由所述程序执行单元执行的所述计算机程序内的函数为单位提供安全属性和验证密钥，以及

其中，所述程序执行单元利用执行所述函数的所述验证密钥执行验证处理，这使得能够根据所述安全属性执行所述函数，

其中，进一步以在由所述程序执行单元执行的所述计算机程序中的变量为单位提供安全属性和验证密钥，以及

其中，所述程序执行单元利用参考所述变量并且执行所述函数的所述验证密钥执行验证处理，这使得能够根据所述安全属性参考所述变量并且执行所述函数，

其中，当安全属性已经添加到由所述计算机程序定义的函数时，通过利用对所述函数指定的验证密钥进行验证，所述程序执行单元能够在没有根据对所述函数内使用的其他函数和变量的安全属性设置的验证的情况下执行所述函数。

2.根据权利要求1所述的信息处理设备，

其中，在所述安全属性中设置对定义所述变量或者所述函数的码元的访问属性、验证密钥版本以及对验证密钥表的指针。

3.根据权利要求1所述的信息处理设备，

其中，通过利用系统密钥进行验证，所述信息处理设备能够改变由所述程序执行单元执行的所述计算机程序定义的所述函数和所述变量的定义。

4.根据权利要求1所述的信息处理设备，

其中，当定义所述函数的函数公式指定为码元时，需要关于利用所述码元和所述函数使用的函数和变量的所有验证密钥进行验证。

5.根据权利要求1所述的信息处理设备，还包括：

二进制数据转换单元，以预定格式编码从外部设备传送的二进制数据。

6.根据权利要求5所述的信息处理设备，

其中，所述二进制数据转换单元将不以所述预定格式使用的符号添加到在转换为所述验证处理而传送的所述二进制数据之后获得的数据的报头。

7.根据权利要求1所述的信息处理设备，进一步包括：

具有非易失性特性的存储单元，存储由所述程序执行单元执行的所述程序中使用的变量定义和函数定义。

8.根据权利要求1所述的信息处理设备，

其中，由所述程序执行单元执行的所述计算机程序能够在任意定时以变量和函数为单位设置安全属性和验证密钥。

9.根据权利要求8所述的信息处理设备，

其中，响应于定义变量或者函数的指令以码元定义专用指令设置所述安全属性。

10.根据权利要求8所述的信息处理设备，

其中，在利用定义变量或者函数的命令定义变量或者函数后，以码元定义专用指令设置所述安全属性。

11.根据权利要求1所述的信息处理设备，

其中，设置所述安全属性进一步限制定义函数的信息的读取和改变。

12.根据权利要求1所述的信息处理设备，

其中，至少所述程序执行单元包括防篡改性能。

13.根据权利要求1所述的信息处理设备，

其中，所述信息处理设备是IC芯片装入其内的IC卡。

14.一种程序执行方法，包括：

在具有防篡改性能的环境下，解释并且执行以过程语言创建的计算机程序的代码的步骤，

其中，以在解释和执行代码时执行的所述计算机程序内的函数为单位提供安全属性和验证密钥，以及

其中，在解释和执行代码时执行利用执行所述函数的所述验证密钥的验证处理，这使得能够根据所述安全属性执行所述函数，

其中，进一步以在解释和执行代码时执行的所述计算机程序中的变量为单位提供安全属性和验证密钥，以及

其中，在解释和执行代码时执行利用参考所述变量并且执行所述函数的所述验证密钥的验证处理，这使得能够以根据所述安全属性参考所述变量并且执行所述函数，

其中，当安全属性已经添加到由所述计算机程序定义的函数时，通过利用对所述函数指定的验证密钥进行验证，所述程序执行单元能够在没有根据对所述函数内使用的其他函数和变量的安全属性设置的验证的情况下执行所述函数。

信息处理设备和程序执行方法

技术领域

[0001] 本公开涉及信息处理设备、程序执行方法和计算机程序。

背景技术

[0002] 最近几年，提供有IC(集成电路)和能够以非接触方式进行通信的非接触天线模块的卡广泛分布。这种卡例如被称为非接触IC卡，并且以非接触方式与另一设备通信。采用非接触IC卡的非接触通信已经用于交通票、电子货币、ID卡、进入和离开房间的管理等，并且其用途不断扩大。

[0003] 根据用途，能够在IC芯片上执行程序的非接触IC卡包括OS(操作系统)。装入这种非接触IC卡的OS的示例包括例如MULTOS(Mel语言)、Java OS(Java(注册商标)语言)等。可以通过将虚拟机(VM)保存在ROM上并且将执行代码写在非易失性存储器上来实现这种OS。

发明内容

[0004] 然而，根据装入这种IC卡内的OS，下载和定义用于将可操作API调入文件系统中的程序，该OS本身不包括编译和调试(debugging)功能，并且在使用OS期间不再动态定义该功能。

[0005] 根据MULTOS或者Java OS，容器(container)可以被定义为防火墙，每种服务的程序都可以下载到该容器，并且可以通过验证该容器来执行程序。然而，保护程序中的各变量和函数取决于写程序的方式。

[0006] 由于多种开发工具用于开发IC卡的应用程序，所以需要理解该工具，并且设置该工具花费时间。此外，对于IC卡的应用程序的开发过程，需要利用开发工具创建并调试代码、最终将程序下载到IC卡上，以及对IC卡执行最终调试。因为该原因，存在的问题是，开发过程复杂，并且如果在对IC卡的最终调试中需要重做(rework)，则校正该程序花费时间。

[0007] 此外，在将该应用程序下载到IC卡后，需要交换编码的二进制数据以使用安全功能，并且需要使用专用解释工具。

[0008] 作为将应用程序提供到IC卡的方法，第2001-184472号日本未审专利申请公开披露了一种技术，根据该技术，可以增强通用性和切换应用程序(游戏程序)。然而，却没有提到使用安全功能。

[0009] 因此，希望提供一种新颖的和改进的信息处理设备、程序执行方法以及计算机程序，根据其可以容易地以必要的严密安全性来开发并且安装应用程序。

[0010] 根据本公开的实施例，提供了一种信息处理设备，包括：程序执行单元，用于在具有防篡改性能的环境下，解释并且执行以过程语言创建的计算机程序的代码，其中以在由该程序执行单元执行的计算机程序内的函数为单位提供安全属性和验证密钥，以及其中该程序执行单元利用验证密钥执行验证处理以执行该函数，这样使得可以根据安全属性执行该函数。

[0011] 可以进一步以在由该程序执行单元执行的计算机程序内的变量为单位提供安全

属性和验证密钥，以及该程序执行单元可以利用验证密钥执行验证处理，以参考变量并且执行函数，这样使得可以根据安全属性参考变量并且执行函数。

[0012] 当安全属性附加到由计算机程序定义的函数时，程序执行单元可以通过利用对函数指定的验证密钥进行验证，在不根据对其他函数设置的安全属性和所述函数内使用的变量进行验证的情况下执行该函数。

[0013] 可以在安全属性中设置定义变量或者函数的码元的访问属性、验证密钥版本以及验证密钥表的指针。

[0014] 该信息处理设备可以通过利用系统密钥进行验证，来改变由程序执行单元执行的计算机程序定义的函数和变量的定义。

[0015] 当定义函数的函数公式指定为码元时，需要利用码元和函数使用的函数和变量的所有验证密钥进行验证。

[0016] 该信息处理设备还可以包括二进制数据转换单元，其以预定格式编码从外部设备传送的二进制数据。

[0017] 该二进制数据转换单元可以将预定格式中未使用的符号附加到在转换所传送的所述二进制数据获得的数据的报头，以用于进行验证处理。

[0018] 该信息处理设备还可以包括：具有非易失性的存储单元，存储由该程序执行单元执行的程序中使用的变量定义和函数定义。

[0019] 由该程序执行单元执行的计算机程序可以在任意定时以变量和函数为单 位设置安全属性和验证密钥。

[0020] 响应于定义变量或者函数的指令，可以以码元定义专用指令设置该安全属性。

[0021] 在以定义变量或者函数的命令定义了变量或者函数之后，可以以码元定义专用指令设置该安全属性。

[0022] 设置该安全属性可以进一步限制读取和改变定义函数的信息。

[0023] 至少该程序执行单元可以包括防篡改性能。

[0024] 该信息处理设备可以是IC芯片装入其内的IC卡。

[0025] 根据本公开的另一实施例，提供了一种程序执行方法，包括：在具有防篡改性能的环境下，解释并且执行以过程语言创建的计算机程序的代码，其中以在解释和执行代码时执行的计算机程序内的函数为单位设置安全属性和验证密钥，以及其中在解释和执行代码时执行利用验证密钥的验证处理以执行函数，这样使得可以根据该安全属性执行函数。

[0026] 也可以以在解释和执行代码时执行的计算机程序的变量为单位提供安全属性和验证密钥，以及可以在解释和执行代码时执行利用验证密钥的验证处理，以参考变量并且执行函数，这样使得可以根据安全属性来参考变量并且执行函数。

[0027] 根据本公开的又一实施例，提供了一种计算机程序，包括：使得计算机在具有防篡改性能的环境下，解释并且执行以过程语言创建的计算机程序的代码，其中以在使得计算机解释和执行代码时执行的计算机程序内的函数为单位提供安全属性和验证密钥，以及其中在使得计算机解释和执行代码时执行利用验证密钥的验证处理以执行函数，这样使得可以根据安全属性执行函数。

[0028] 以在使得计算机解释和执行代码时执行的计算机程序的变量为单位提供安全属性和验证密钥，以及在使得计算机解释和执行代码时执行利用验证密钥的验证处理，以参

考变量并且执行函数，这样使得可以根据安全属性参考变量并且执行函数。

[0029] 根据本公开，如上所述，提供了新颖的和改进的信息处理设备、程序执行方法以及计算机程序，根据其可以容易地以必要的严密安全性开发并且安装应用程序，并且通过网络可以安全地装入来自客户机PC的新应用。

附图说明

[0030] 图1是示出根据本公开实施例的信息处理系统的配置示例的说明图；

[0031] 图2是示出根据本公开实施例的信息处理设备的硬件配置的说明图；

[0032] 图3是示出可以由列表处理模块定义并且被称为码元的数据结构的说明图；

[0033] 图4是示出配置列表结构的构成元素的配置示例的说明图；

[0034] 图5是示出用于存储在码元的名称区内存储的名称的名称存储表的结构示例的说明图；

[0035] 图6是示出存储验证密钥的验证密钥表的结构示例的说明图；

[0036] 图7是示出图3所示码元、图4所示构成元素、图5所示名称存储表以及图6所示验证密钥表的对应关系的说明图；

[0037] 图8是示出在起动CPU执行的列表处理模块时的处理的流程图；

[0038] 图9是示出CPU执行的列表处理模块的模式转移的说明图；

[0039] 图10是示出码元注册序列和安全函数激活序列的流程图；

[0040] 图11是示出码元注册序列和安全函数激活序列的流程图；

[0041] 图12是示出对其进行安全设置的变量的列表结构示例的说明图；

[0042] 图13是示出已经对其进行了安全设置的函数的列表结构示例的说明图；

[0043] 图14是示出根据修改例的信息处理设备的硬件配置的说明图；

[0044] 图15是示出现有技术中的应用开发模型的说明图；

[0045] 图16是示出根据本公开实施例的信息处理系统的应用开发模型的说明图；以及

[0046] 图17是示出根据本公开实施例的开发设备的硬件配置的框图。

具体实施方式

[0047] 下面将参考附图详细描述本公开的优选实施例。此外，对本说明和附图中具有基本相同的功能配置的部件赋予相同的附图标记，并且不重复描述它们。

[0048] 以下面的顺序进行描述。

[0049] <1.本公开的实施例>

[0050] [1-1.系统配置示例]

[0051] [1-2.硬件配置示例]

[0052] [1-3.程序结构示例]

[0053] [1-4.信息处理设备的修改例]

[0054] [1-5.应用开发模型的比较]

[0055] [1-6.开发设备的硬件配置]

[0056] <2.结论>

[0057] <1.本公开的实施例>

[0058] [1-1.系统配置示例]

[0059] 首先，描述根据本公开实施例的系统配置示例。图1是示出根据本公开实施例的信息处理系统1的配置示例的说明图。下面将参考图1描述根据本公开实施例的信息处理系统1的配置示例。

[0060] 如图1所示，根据本公开实施例的信息处理系统1包括信息处理设备100、读写器(R/W)200和开发设备300。

[0061] 信息处理设备100在其中包括抗篡改性，并且在其内装入了提供有近程非接触通信功能的IC芯片。例如，信息处理设备100可被配置为在其内装入了这种IC芯片的IC卡、移动电话、移动终端等。此外，这种IC芯片可以通过与读写器200执行近程非接触通信，而与读写器200交换信息。

[0062] 提供读写器200，以便主要连接到车站内的自动检票机、收银机等，并且与其内装入了IC芯片的信息处理设备100执行近程非接触通信。通过读写器200与信息处理设备100执行近程非接触通信，例如，连接到读写器200的自动检票机可以开启闸机，或者连接到读写器200的收银机可以对商品进行支付。

[0063] 开发设备300是用于开发安装在装入信息处理设备100内的IC芯片中的计算机程序的设备。创建要安装在IC芯片内的计算机程序的用户可以利用开发设备300来创建要安装在IC芯片内并且要在IC芯片内执行的计算机程序的源代码。将由开发设备300创建的计算机程序的源代码经由连接到开发设备300并且安装在装入信息处理设备100内的IC芯片中的读写器200传送到信息处理设备100。信息处理设备100可以解释装入IC芯片的计算机程序的源代码，并且执行该计算机程序。

[0064] 装入根据实施例的信息处理设备100的IC芯片被配置为能够执行用于对该源代码定义的每个变量和函数独立进行安全设置的程序。通过这样做，装 入根据该实施例的信息处理设备100的IC芯片可以防止程序由对该装入的程序没有执行授权的应用等执行，因此，可以安全地执行该程序。

[0065] 图1示出由开发设备300创建的计算机程序的源代码通过连接到该开发设备300的读写器200传送到信息处理设备100的配置。然而，不必为了将来自开发设备的计算机程序安装到根据本公开的信息处理设备而通过读写器传送源代码，并且该计算机程序可以直接从开发设备传送到信息处理设备。

[0066] 上面参考图1描述了根据本公开实施例的信息处理系统1。接着，将描述根据本公开实施例的信息处理设备100的硬件配置示例。

[0067] [1-2.硬件配置示例]

[0068] 图2是示出根据本公开实施例的信息处理设备100的硬件配置的说明图。下面将参考图2描述根据本公开实施例的信息处理设备100的硬件配置。

[0069] 图2所示的信息处理设备100包括例如作为装入IC卡内的芯片的硬件配置，其总体具有抗篡改性。如图1所示，根据本公开实施例的信息处理设备100包括：NVM(非易失性存储器)110、CPU(中央处理单元)120、ROM(只读存储器)130、RAM(随机存取存储器)140、BASE 64模块150、密码(cryptography)处理模块160、随机数生成模块170以及串行I/O接口180。

[0070] 在NVM 110上，在初始化信息处理设备100时，事先写入ROM 130内的装入的函数被记录为码元。此外，NVM 110还存储用户定义的变量(用户定义变量)和函数(用户定义函

数)。由于即使电源被切断,NVM 110仍可以保持记录信息,所以即使当信息处理设备100的电源再一次激活时,也不再次进行初始化,并且注册的码元保持原样。

[0071] CPU 120控制信息处理设备100的操作,并且可以通过执行事先记录在ROM 130上的操作系统软件的读命令来执行操作系统。CPU 120可以将RAM140用作执行操作系统的工作区。在此,记录在ROM 130上的操作系统软件的示例包括通过解释过程编程语言而执行的软件,并且这种编程语言的示例包括:LISP、Ruby、Python等。

[0072] 根据该实施例的信息处理设备100的ROM 130存储添加到过程编程语言的基本函数的安全函数作为解释器。利用这种配置,当应用程序安装在信息处理设备100内时,不需要事先编译,并且因为对其附加了安全函数,所以使用该应用的信息处理设备100本身可以进行调试。这样缩短了开发步骤, 并且可以在短时间内开发应用程序。

[0073] BASE 64模块150是对用于与提供在信息处理设备100外部的设备(下面简称为“外部设备”)进行通信的二进制数据进行BASE 64转换和反转换的模块。根据该实施例的信息处理设备100通过串行I/O接口180执行串行数据通信,作为与外部设备的通信。根据该实施例,当从信息处理设备100的外部与列表处理模块进行通信时,BASE 64模块150将该二进制数据转换为BASE 64代码,以使得可以将所有输入数据作为ASCII字符序列进行通信。

[0074] 当在信息处理设备100与外部设备(例如,读写器)之间执行交叉验证并在之后进行通信时,二进制格式的交换编码数据经历BASE 64模块150执行的BASE 64转换。通过对报头附加BASE 64中未使用的符号(例如,“:”、“~”等),并且由通用列表处理公式判别该编码数据,可以交换该编码数据。

[0075] 对于通过串行I/O接口180的串行数据通信,回车(carriage return)符号用作输入的分隔符。在将ASCII码用作与外部设备进行通信使用的代码的情况下,信息处理设备100与该外部设备交换信息。然而,信息处理设备100利用BASE 64模块150对二进制数据执行BASE 64转换,用于交换诸如编码数据之类的二进制数据。然后,BASE 64模块150可以将BASE 64中未使用的字符(符号)添加到报头,用于对该二进制数据执行BASE 64转换,如上所述。当CPU 120执行程序时,通过对该报头添加BASE 64未使用的字符(符号),CPU 120可以将BASE 64模块150转换的数据识别为二进制数据。

[0076] 当CPU 120执行的程序锁定,并且诸如锁定使用的密钥的设置之类的码元公式和二进制数据同时出现时,通过对码元公式和二进制数据分配不同的字符(符号),当CPU 120执行该程序时,CPU 120可以识别码元公式与二进制数据之间的不同。

[0077] 密码处理模块160利用指定密钥对输入数据执行密码处理,并且输出输入数据,并且利用该指定密钥,对输入的编码数据执行解码处理并输出输入的编码数据。

[0078] 随机数发生模块170根据来自外部(例如,来自CPU 120)的随机数产生命令产生并输出适当的随机数。随机数发生模块170产生的随机数用于通过将随机数从信息处理设备100发送到要验证的相应装置(例如,读写器),允许相应装置产生密码,以及确定对相应装置(例如,读写器200)回复的密码的解码结果是否与信息处理设备100发送的随机数一致,来确定要验证 的相应装置是否正确。

[0079] 串行I/O接口180包括从外部设备识别串行数据作为分组并且提取适当的数据的功能和将从信息处理设备100内部输出的数据作为分组配置到外部设备并且将该分组数据作为串行数据输出的功能。

[0080] 由于具有如图2所示配置的信息处理设备100总体具有抗篡改性，所以难以从外部分析该装置和该电路的内部。关于CPU 120执行的计算机程序，对每个变量和函数分别进行安全设置。

[0081] 尽管图2中未示出，但是信息处理设备100提供有天线、调制与解调电路等，以用于与读写器200进行近程非接触通信。

[0082] 上面参考图2描述了根据本公开实施例的信息处理设备的硬件配置。接着，将参考图2描述信息处理设备100执行的计算机程序的结构。

[0083] [1-3.程序结构示例]

[0084] 图3至图6是示出根据本公开实施例的信息处理设备100执行的计算机程序的结构示例的说明图。下面将参考图3至图6描述根据本公开实施例的信息处理设备100执行的计算机程序的结构示例。

[0085] 尽管在假定信息处理设备100要执行的计算机程序的语言是LISP的情况下进行下面的描述，但是可以用作编程语言的语言不局限于这种示例，并且根据本公开，可以配置以使得对于作为扩展性能或者标准性能的每个变量和函数独立地进行安全设置的任何过程编程语言都是可用的。

[0086] CPU 120加载用于解释并且执行利用开发设备300开发的并且被安装在信息处理设备100内的程序的源代码的列表处理模块。图3是可以定义列表处理模块并且被称为码元的数据结构的说明图。

[0087] 如图3所示，能够定义列表处理模块的码元400包括：名称区401、变量定义区402、函数定义区403和安全属性区404。

[0088] 名称区401指示可打印字符表。在名称区401上，当该码元定义变量时，存储变量名称，并且当码元定义函数时，存储函数名称。在图3中，名称区401示为“pname”。

[0089] 变量定义区402在该码元定义简单变量时存储简单变量的值，并且当该码元定义列表变量时存储指示列表的值。在图3中，变量定义区402示为“value(值)”。

[0090] 当码元存储函数时，函数定义区403存储函数本身。在图3中，函数定义区403被示为“function(函数)”。

[0091] 安全属性区404存储与码元的安全属性相关的信息。例如，属性的示例包括：变量读取属性、变量改变属性以及函数执行属性。安全属性区404存储指示对码元的访问授权的访问标志和指示其内存储了访问码元的验证密钥的表的值。

[0092] 除了图3所示的码元400，顺序定义用于配置列表结构的被称为构成元素(cons cell)的元素。图4是示出用于配置列表结构的构成元素410的配置示例的说明图。如图4所示，构成元素410是包括被称为CAR时隙411和CDR时隙412的两个指针的对象。图4示出作为CAR时隙411的car 0至car9和作为CDR时隙412的cdr 0至cdr 9。当然，不用说，各时隙的数量并不局限于该示例。

[0093] 还设置了用于存储在码元400的名称区401内存储的名称的表。图5是示出用于存储在码元400的名称区401内存储的名称的名称存储表420的结构示例的说明图。图5所示的名称存储表420存储名称“eval”、“setq”、“cons”、“defun”和“osaifu”，在其实际情况下，它们分别与码元具有一对一的对应关系。附图标记421表示其内存储了名称“eval”的区，附图标记422表示其内存储了名称“setq”的区，附图标记423表示其内存储了名称“cons”的区，

附图标记424表示其内存储了名称“defun”的区，以及附图标记425表示其内存储了名称“osaifu”的区。当码元名称从名称存储表420的外部输入到名称存储表420时，指示并且评估存储在名称存储表420内的输入码元名称的码元。此外，“osa而”是当电子货币函数被装入信息处理设备100时表示电子货币的余额的变量。

[0094] 此外，还提供了存储在码元400的安全属性区404内的与指示存储验证密钥的表的值对应的表。图6是示出存储验证密钥的验证密钥表430的结构示例的说明图。图6示出在验证密钥表430内利用版本号(kv1至kv5)管理验证密钥的状态。附图标记431、432、433、434和435代表其内分别存储密钥“密钥1”、“密钥2”、“密钥3”、“密钥4”和“密钥5”的区域。

[0095] 在信息处理设备100内，在执行由开发设备300开发的计算机程序之前产生图3至图6所示的表，并且存储在NVM 110内。在这样做时，可以保持即使信息处理设备100的电源断开，表的内容也保持原样的状态。

[0096] 图7是示出图3所示码元、图4所示构成元素、图5所示名称存储表以及图6所示验证密钥表的对应关系的说明图。如上所述，该码元具有指示可打印名称表的区域、指示值或者值列表、函数属性和安全属性的区域。函数属性具有指示函数的类型和函数的实际条件的指针，而安全属性具有指示安全标志、密钥版本和密钥的指针。此外，图7示出其中采用图6所示验证密钥表430中的附图标记431所示的密钥“key1”和附图标记432所示的密钥“key2”的状态。

[0097] 通过采用具有防篡改功能的诸如图2所示信息处理设备100的硬件进行保护，可以防止图3所示的码元、图4所示的构成元素、图5所示的名称存储表以及图6所示的验证密钥表全部不被不适当地获取或者误用值。

[0098] 如上所述，列表处理模块的通用结构被称为码元，并且包括数值或者保持数值的列表的指针、定义函数情况下的函数的指针以及指示存储可打印字符序列的表的指针。

[0099] 除了这些组成部分，根据该实施例，对码元附加用于保持安全属性和两种编码密钥信息的表的指针。一个密钥的指针指示主密钥，而另一密钥的指针指示码元的访问密钥(验证密钥)。主密钥指示当码元的安全属性或者访问密钥改变时事先要利用交叉验证函数验证的密钥。当在评估或者改变码元保持的信息内容时或者在执行该函数时添加对码元设置的安全标志时，需要利用对码元添加的一个密钥来进行验证以使用该码元。另一密钥指示当码元的密钥改变时用于确认授权的授权验证。为了改变访问信息，需要利用授权验证密钥进行验证。

[0100] 此外，有被称为构成元素的两对指针，它们表示码元之间的关系，如图4所示，每个指针分别具有指示码元的结构或者指示另一码元的构成元素。

[0101] 装入的函数被写入ROM 130，并且写入ROM 130的装入的函数被定义为在第一次激活信息处理设备100的电源时在NVM 110内创建的码元。在之后激活电源时，已经被激活的码元不被初始化。

[0102] 上述配置具有即使在用户注册新函数时仍具有相同功能的结构。

[0103] CPU 120执行的列表处理模块具有可以自由注册码元并且可以将数值、列表和函数自由注册到该码元的配置。为了使注册的码元具有安全函数，代码密钥和访问标志被注册在该码元内。对于CPU 120执行的列表处理模块，首先设置被称为系统密钥的编码密钥。在对应于在利用系统密钥已经进行了 交叉验证的模式的状态下(对应于下面描述的模式2

的状态)，可以仅对新注册的码元设置唯一的密钥和访问标志。此外，CPU 120执行的列表处理模块执行的计算机程序被配置为能够改变仅在对应于利用系统密钥已经进行了交叉验证的模式的状态下使用的变量和函数的定义。

[0104] CPU 120执行的列表处理模块的条件是，已经利用在用于注册函数码元的函数中要使用的码元的所有密钥进行了验证。此后，列表处理模块被配置以使得当使用注册的函数时，仅利用函数执行密钥进行验证。

[0105] 接着，将描述在启动列表处理模块和CPU 120执行的列表处理模块模式转移时的处理。图8是示出在起动CPU 120执行的列表处理模块时的处理的流程图。此外，图9是示出CPU 120执行的列表处理模块的模式转移的说明图。

[0106] 首先，参考图9描述在启动CPU 120执行的列表处理模块时的处理。首先，当开启信息处理设备100的电源时，列表处理模块被加载在CPU 120上。然后，CPU 120执行的列表处理模块首先确定初始化完成标志是否已经开启(步骤S101)。

[0107] 当步骤S101的确定结果是初始化完成标志未开启时，列表处理模块将来自ROM 130的函数和变量装入NVM 110，并且定义码元。当码元的定义完成时，列表处理模块开启初始化标志(步骤S102)。

[0108] 当步骤S102的处理完成时，或者当步骤S101的确定结果是初始化完成标志已经开启时，列表处理模块读取从信息处理设备100的外部输入的字符，产生码元，并且转换为列表结构(步骤S103)。

[0109] 当步骤S103的处理完成时，列表处理模块评估列表公式并且将评估结果看作列表结构(步骤S104)。然后，当列表公式的评估结果除了码元定义无错误时，列表处理模块将码元定义写入NVM 110(步骤S105)。此后，列表处理模块将列表结构结果变更为列表公式，并且输出该列表公式(步骤S106)。

[0110] 在步骤S106将列表结构结果变更为列表公式并且输出了该列表公式之后，列表处理模块返回步骤S103，读取从信息处理设备100的外部输入的字符，产生码元，以及执行向列表结构的转换。通过上面的一系列操作，CPU 120执行的列表处理模块可以利用装入NVM 110的函数和变量来执行开发设备300创建的计算机程序。

[0111] 接着，参考图9描述CPU 120执行的列表处理模块的模式转移。如图9 所示，当在模式0、模式1和模式2之间转移时，CPU 120执行的列表处理模块操作。

[0112] 模式0是利用未锁定码元可以参考并且改变变量以及执行函数，并且以明文(plain text)进行通信的模式。

[0113] 模式1是从模式0转移到模式2的过程阶段，即，验证处理阶段，并且处于列表处理模式已经验证为通信对方的状态下。在这种模式下，列表处理模块和通信对方还没有进入交叉验证状态，并且列表处理模块需要验证通信对方，以进入交叉验证状态。此外，在图9中将“auth1”公式用作用于验证列表处理模块作为通信对方的公式。

[0114] 此外，模式2是指示以下状态的模式，其中在列表处理模块与通信对方之间已经利用事先设置的编码密钥(被称为系统密钥)进行了交叉验证，并且可以根据已经验证的并未锁定的码元的验证标志参考变量和执行函数，并且可以使模式0下的函数可操作。此外，CPU 120执行的列表处理模块中执行的计算机程序配置以使得仅在与已经利用系统密钥进行交叉验证的模式对应的状态下改变要使用的变量和函数的定义。尽管利用对话密钥以编码文

本进行通信，但是在模式0下还可以接收明文。此外，在图9中，将“auth2”公式用作用于验证通信对方的列表处理模块的公式。

[0115] 如果在模式2的状态下，列表处理模块执行复位操作，则该状态返回可以利用未锁定码元参考和改变变量并且执行函数的模式0。此外，在图9中，“复位”公式用作用于从模式2转移到模式0的公式。

[0116] CPU 120执行的列表处理模块可以执行未锁定码元，并且利用包括模式0、模式1和模式2的三种模式下的操作来释放并且执行锁定码元。

[0117] 上面参考图9描述了CPU 120执行的列表处理模块的模式转移。接着，将描述根据本公开实施例的信息处理设备100内的码元注册序列和安全函数激活序列。图10和图11是示出根据本公开实施例的信息处理设备100内的码元注册序列和安全函数激活序列的流程图。

[0118] 首先，参考图10描述开发应用时的码元注册序列。首先，利用开发设备300开发应用，在开发时注册应用中要使用的变量和函数(步骤S111)。在注册了应用中要使用的变量和函数后，开发设备300执行调试处理，以检验注册的变量和函数是否正确地工作(步骤S112)。

[0119] 如果通过步骤S112的调试处理，应用中没有问题，则将应用从开发设备 300安装到信息处理设备100。当应用安装在信息处理设备100内时，首先使得列表处理模块利用系统密钥执行交叉验证(步骤S113)。如上所述，可以仅在模式2的状态下对码元设置唯一的密钥和访问标志，在该模式2中，利用系统密钥进行交叉验证。

[0120] 当利用系统密钥的交叉验证完成时，将应用从开发设备300安装到信息处理设备100，并且所需安全属性设置为函数和变量的码元(步骤S114)。在这样做时，可以锁定应用中要使用的函数和变量的码元，并且可以仅允许具有执行函数和参考变量的授权的人执行该函数、参考该变量以及改变值。

[0121] 接着，将参考图11描述当为了发布信息处理设备100而将开发设备300开发的应用安装在信息处理设备100内时的流程。

[0122] 当为了发布信息处理设备100而将开发设备300开发的应用安装在信息处理设备100内时，首先使得列表处理模块利用系统密钥执行交叉验证(步骤S121)。当利用系统密钥的交叉验证完成时，则使得列表处理模块读取开发设备300开发的程序、定义码元、注册变量以及注册函数(步骤S122)。在不执行步骤S121的交叉验证的情况下，列表处理模块不读取程序。通过在步骤S121执行交叉验证，列表处理模块可以定义对其已经进行了安全设置的码元、注册变量以及注册函数，从而执行函数、参考变量以及改变值。

[0123] 接着，将描述根据本公开实施例的信息处理设备100执行的列表处理模块可以参考的、对其进行了安全设置的变量的列表结构示例。图12是示出已经对其进行了安全设置的变量的列表结构示例的说明图。

[0124] 在利用系统密钥进行了验证后，产生码元，并且设置安全标志、密钥版本和密钥。可以指定三种安全属性(S-标志)。在图12中，X表示函数执行锁定，M表示内容更新锁定，E表示内容读取锁定。函数执行锁定表示在未利用验证密钥进行验证的情况下函数的执行不可用。内容更新锁定表示在未利用验证密钥进行验证的情况下变量的更新不可用，内容读取锁定表示在未利用验证密钥进行验证的情况下参考该变量的值不可用。图12示出包括

“osaifu”和“log”两个变量的结构示例。下面将描述两个变量“osaifu”和“log”的列表结构。

[0125] 图12所示的变量“osaifu”是存储电子货币的余额的变量。对变量“osaifu”指定变量名称(pname)“osaifu”，对用于存储电子货币的余额的值(value)指定0。此外，内容更新锁定(M)和内容读取锁定(E)指定为变量“osaifu” 的安全属性。在这样做时，在不利用与变量“osaifu”的值相关的验证密钥进行验证的情况下，不能参考和改变该内容。

[0126] 此外，与对主密钥的指针、用于访问变量“osaifu”的访问密钥的版本以及访问密钥的指针相关的信息存储在变量“osaifu”上。如上所述，主密钥是在码元的安全属性和访问密钥(在该示例中是变量“osaifu”)时需要事先利用交叉验证进行验证的密钥。图12示出用于访问变量“osaifu”的访问密钥的版本是1，而存储与访问密钥的指针相关的信息的状态。

[0127] 图12所示的变量“log”是存储函数的执行结果的变量。“log”指定为变量“log”的变量名称(pname)。此外，存储函数执行结果的值(value)具有循环配置。在图12所示的配置中，5次执行函数的结果存储在变量“log”上。此外，利用装入列表处理函数内的函数来设置循环文件的产生。

[0128] 然后，内容更新锁定(M)和内容读取锁定(E)指定为变量“log”的安全属性。在这样做时，在未利用验证密钥进行验证的情况下，对于变量“log”的值，不能参考和改变内容。图12示出用于访问变量“log”的访问密钥的版本是2，并且存储与访问密钥的指针相关的信息的状态。

[0129] 此外，图12所示的“System”具有该模块的原始授权。如果对“System”设置密钥，则对于产生每个函数和变量对象都需要进行预验证。另一方面，如果未对“System”设置密钥，则不需要预验证，并且例如，在Felica(注册商标)中，“System”变成用于保持系统代码的变量和模块信息，Felica是非接触式IC卡的一种技术。

[0130] 接着，将描述根据本公开实施例的信息处理设备执行的列表处理模块可以参考的、已经对其进行了安全设置的函数的列表结构。图13是对其进行了安全设置的函数的列表结构示例的说明图。

[0131] 图13示出函数“charge”的列表结构示例。下面将描述函数“charge”的列表结构。

[0132] 用于定义函数“charge”的S公式如图13所示。在此，S公式基于LISP内使用的逻辑描述方法，并且用于定义码元。利用S公式，函数“charge”执行将自变量x指定的值与图12所示的变量“osaifu”的值相加的处理。此外，S公式使用图12所示的两个安全锁定变量“osaifu”和“log”。

[0133] 图13所示的“charge”、“osaifu”和“log”是安全锁定函数和变量。因此，为了执行函数“charge”，需要对所有安全锁定函数执行预验证。

[0134] 如上所述，为了执行函数“charge”，需要对两个安全锁定变量“osaifu”和“log”执行预验证。然而，被授权执行函数“charge”的人被认为已经被授权参考和改变变量“osaifu”和“log”的值，以及归因于授权的转移，可以通过利用对函数“charge”设置的码元进行验证来执行函数“charge”，且无需对变量“osaifu”和“log”重复验证。

[0135] 上面描述了已经对其进行了安全设置的函数的列表结构示例。如上所述，根据该实施例的信息处理设备100的CPU 120执行的列表处理模块可以参考安全锁定变量并且执

行安全锁定函数。此外，可以对每个所参考的变量和信息处理设备100执行的列表处理模块执行的函数分别唯一地设置安全属性，并且安全锁定该变量和函数。

[0136] 在这样做时，可以对信息处理设备100执行的列表处理模块执行的程序中使用的变量和函数分别执行唯一的安全设置，这样获得严密安全实现。关于信息处理设备100执行的列表处理模块，不需要预编译，因为除了解释器的基本函数之外，还添加了安全函数。此外，在实际使用的信息处理设备100中可以进行调试。此外，在NVM 110内，可以直接和动态地定义变量和函数，通过在NVM 110内动态和直接地定义函数和变量，可以灵活地开发程序。此外，通过设置安全属性和函数执行的限制，也可以限定定义信息的函数的读取和改变，并且可以防止程序因为外界的攻击而泄漏或者误用定义信息的函数。

[0137] 安全锁定变量和函数不可用，直到利用相应验证密钥进行验证为止。关于函数，对于定义函数需要对函数在内部用作自变量的安全锁定变量和函数进行预验证。然而，当使用定义之后的函数时，用作在内部使用的自变量的变量和函数无条件地可用。因此，不需要准备所有验证密钥并且不需要每次执行函数时都执行验证，并且有助于其操作。

[0138] 将描述具体示例。当对变量“osaifu”执行安全设置时，仅在利用对变量“osaifu”指定的密钥执行交叉验证的状态下，在变量“osaifu”中存储的值的读取和改变可用。

[0139] 另一方面，例如，当对改变关于变量“osaifu”存储的值的变量“charge”执行安全设置时，仅在利用对函数“charge”指定的密钥执行交叉验证的状态下，可以执行函数“charge”。在这种情况下，如果在验证函数“charge”的执行时，未验证读取和改变在变量“osaifu”中存储的值，则不能读取和改 变在变量“osaifu”中存储的值。通过对变量和函数分别执行唯一安全设置，灵活操作是可行的。此外，不需要对内部使用的所有变量准备验证密钥，并且不需要在每次执行函数时执行验证，且因此有助于操作。

[0140] 可以在任意定时实现信息处理设备100执行的列表处理模块执行的程序内使用的变量和函数的安全功能。因此，在对开发设备300开发的程序进行了充分调试后，在开发设备300内，安全功能可以安装在变量和函数中，并且在可以进一步执行在安装了安全功能的状态下的调试。

[0141] 在传输和交换之前，密码处理模块160编码的二进制数据被转换为可打印字符。在这样做时，可以由诸如网络应用等的另一应用操作密码处理模块160编码的二进制数据。

[0142] [1-4.信息处理设备的修改例]

[0143] 接着，将描述根据本公开实施例的信息处理设备的修改例。图14是示出根据本公开实施例的信息处理设备的修改例的信息处理设备1100的硬件配置的说明图。此后，将参考图14描述信息处理设备1100的硬件配置。

[0144] 作为设备的整体，图2所示的信息处理设备100具有抗篡改性。如图14所示，信息处理设备1100包括：NVM 1110、安全CPU 1120、ROM 1130、RAM 1140和串行I/O 1180。信息处理设备1100与图2所示的信息处理设备100的不同之处在于，作为整个设备，信息处理设备1100没有抗篡改性，并且只有安全CPU 1120。如上所述，包括CPU的部分由抗篡改函数保护并且利用密码技术保护与其他函数通信和交换数据的配置也适用。此外，图2所示信息处理设备100提供在具有抗篡改环境的位置、参考变量并且执行函数属性的另一配置也适用。

[0145] 当初始化信息处理设备400时，在NVM 1110上，事先写入ROM 1130的装入的函数记录为码元。此外，NVM 1110还存储用户定义的变量(用户定义变量)和函数(用户定义函数)。

由于即使在电源被切断时，NVM 1110仍可以保持记录信息，所以当信息处理设备1100的电源激活时，不重复初始化，并且原样地保持该注册码元。

[0146] 安全CPU 1120控制信息处理设备1100的操作，并且可以通过执行事先记录在ROM 1130内的操作系统软件的读命令，来执行操作系统。当执行操作系统时，CPU 1120可以将RAM 1140用作工作区域。在此，例如，记录在 ROM 1130内的操作系统软件是可以解释并且执行过程编程语言的软件，并且这种编程语言的示例包括LISP、Ruby、Python等，如上所述。

[0147] 此外，安全CPU 1120还包括图2所示的密码处理模块160的功能。安全CPU 1120包括在其内保持编码密钥并且对要写入ROM 1130的程序代码和要记录在NVM 1110和RAM 1140上的数据进行编码的功能。此外，安全CPU1120还包括图2所示随机数生成模块170的功能，并且可以根据随机数生成命令来产生适当的随机数。

[0148] 串行I/O接口1180包括将来自外部设备的串行数据识别为分组并且提取正确数据以及将从信息处理设备1100的内部输出到外部设备的数据配置为分组数据并将该数据作为串行数据输出的功能。

[0149] 上面参考图14描述了信息处理设备400的硬件配置。如果不是整个设备而是该设备的一部分包括抗篡改性，则在通过使得具有抗篡改性的部分执行上述列表处理模块来分别保护变量和函数的同时，诸如信息处理设备1100的设备可以执行程序。

[0150] 上面描述了通过对列表处理模块添加安全属性并且进一步保护具有抗篡改性功能的设备，而能够进行开发并执行安全应用的信息处理设备。在此，将描述现有技术中的应用开发模型与根据本公开实施例的信息处理系统的应用开发模型之间的差别。

[0151] [1-5.应用开发模型的比较]

[0152] 图15是示出现有技术中的应用开发模型的说明图。下面将参考图15描述现有技术中的应用开发模型。首先，当创建IC卡要执行的程序时，需要将该程序进行编译以产生类文件。此后，利用模拟器调试该类文件，并且如果操作错误，则需要校正创建的程序。

[0153] 完成调试的类文件与库文件一起被送到转换器，以创建由在其内装入了诸如IC卡等的IC芯片的装置执行的应用文件。然后，创建的应用文件被装入用于模拟IC卡的运行环境的卡模拟器，并且在该卡模拟器上进行调试。如果在该阶段发生错误，则需要校正该程序、编译为类文件、以及转换为应用文件。

[0154] 当卡模拟器上的调试完成时，应用文件被最终装入该IC卡中。然后，执行对IC卡的调试，并且如果该阶段存在错误，则需要校正该程序、编译为类 文件、转换为应用文件、以及在卡模拟器上进行调试。

[0155] 如上所述，需要根据现有技术的应用开发模式，在多个阶段重复创建和调试程序，并且在诸如IC芯片等的防篡改环境下实现竞争程序(competedprogram)需要大量步骤。

[0156] 图16是示出根据本公开实施例的信息处理系统的应用开发模式的说明图。下面将参考图16描述根据本公开实施例的信息处理系统的应用开发模式。首先，利用开发设备300开发信息处理设备100要执行的程序，然后，将该程序原样地装入信息处理设备100中。在利用开发设备300开发期间，可以事先对该程序使用的变量和函数进行安全设置，或者，也可以对该程序之后使用的变量和函数设置这种安全函数。

[0157] 利用防篡改环境下执行的列表处理模块，程序装入其内的信息处理设备100执行开发设备300开发的程序。此时，不需要对程序代码执行预编译，而在现有技术的应用开发

模型中需要预编译，并且利用信息处理设备100可以直接执行该程序。在这样做时，在没有从外部窥探该程序使用的程序代码或者数据的情况下，信息处理设备100可以安全地执行该程序。

[0158] 当在信息处理设备100执行该程序中观察到存在错误时，开发设备300对该程序执行调试。然而，应当明白，所增加的调试步骤数量显著少于图15所示的现有技术的应用开发模式中的步骤数量。因此，与现有技术中的应用开发模式相比，可以显著减少在诸如IC芯片等的防篡改环境下实现所完成的程序开发步骤，因此，基于根据本公开实施例的信息处理系统的应用开发模型，可以容易地并安全地开发应用。

[0159] 基于根据本公开实施例的信息处理系统的应用开发模型，在创建程序时，也可以事先对变量和函数执行安全设置，并且使得信息处理设备100执行该程序。然而，另一种操作也适用，其中在对变量和函数未执行安全设置的状态下，使得信息处理设备100执行程序、如果该操作中没有问题，则该程序安装在信息处理设备100中、以及对所安装的程序执行安全设置。

[0160] [1-6.开发设备的硬件配置]

[0161] 接着，将参考图17详细描述根据本公开实施例的开发设备300的硬件配置。图17是示出根据本公开实施例的开发设备300的硬件配置的框图。

[0162] 开发设备300主要包括：CPU 901、ROM 903、RAM 905、主总线907、桥接器909、外部总线911、接口913、输入设备915、输出设备917、成像设备918、存储设备919、驱动器921、连接端口923以及通信设备925。

[0163] CPU 901用作计算处理设备和控制设备，并且根据记录在ROM 903、RAM 905、存储设备919或者可拆卸记录介质927上的各种程序，来控制开发设备300的全部或者部分操作。ROM 903存储CPU 901执行的程序、计算参数等。RAM 905临时存储CPU 901的执行使用的程序和在该程序的执行过程中适当改变的参数等。这些部件通过由诸如CPU总线等的内部总线配置的主总线907互相连接。

[0164] 主总线907通过桥接器909连接到诸如PCI(外围部件互连/接口)之类的外部总线911。

[0165] 输入设备915是用户可以操作的诸如鼠标、键盘、触摸面板、按钮、开关、操纵杆等的操作单元。此外，输入设备915可以是采用红外线或者其他电波的遥控单元(所谓遥控器)，也可以是对应于开发设备300的操作的诸如移动电话、PDA等的外部连接装置。此外，输入设备915包括根据用户利用操作单元输入的信息产生输入信号并且将该输入信号输出到CPU 901的输入控制电路等。开发设备300的用户可以通过操作输入设备915关于开发设备300输入各种数据项目或者命令处理操作。

[0166] 输出设备917由可视地或者利用声音通知获得的信息的使用的诸如CRT显示设备、液晶显示设备、等离子显示设备、EL显示设备、灯等的显示设备，诸如扬声器、头戴耳机等的声音输出设备、打印机、移动电话、传真机等的设备予以配置。输出设备917输出例如开发设备300执行的各种处理获得的结果。具体地说，显示设备将开发设备300执行各种处理获得的结果显示为文本或者图像。另一方面，声音输出设备将包括再现的声音数据、声学数据等的音频信号转换为模拟信号，并且输出该模拟信号。

[0167] 成像设备918提供在例如显示设备的上部，并且可以拍摄开发设备300的用户的静

止图像或者运动图像。成像设备918提供有CCD(电荷耦合器件)图像传感器或者CMOS(互补金属氧化物半导体)图像传感器,并且通过将由镜头采集的光转换为电信号,可以拍摄静止图像或者运动图像。

[0168] 存储设备919是用于存储数据的设备,其被配置为开发设备300的存储单元的示例,并且例如由诸如HDD(硬盘驱动器)的磁存储装置、半导体存储装置、光学存储装置、磁光存储装置等配置。存储设备919存储CPU 901 执行的程序和各种数据项目以及从外部获得的声学信号数据、图像信号数据等。

[0169] 驱动器921是记录介质的读写器,并且安装在开发设备300之内或者附加在其外部。驱动器921读取记录在安装在其上的诸如磁盘、光盘、磁光盘、半导体存储器等的可拆卸记录介质927上的信息,并且将该信息输出到RAM905。此外,驱动器921还可以将记录写入安装在其上的诸如磁盘、光盘、磁光盘、半导体存储器等的可拆卸记录介质927。可拆卸记录介质927是例如DVD介质、蓝光介质、CompactFlash(CF)(注册商标)、存储棒、SD存储卡(安全数字存储卡)等。此外,可拆卸记录介质927可以是例如非接触式IC芯片安装在其上的IC卡(集成电路卡)、电子装置等。

[0170] 连接端口923是用于将装置直接连接到开发设备300的端口,并且其示例包括:USB(通用串行总线)端口、诸如i.Link等的IEEE 1394端口、SCSI(小型计算机系统接口)端口、RS-232端口、光声端子、HDMI(高清晰度多媒体接口)端口等。通过将外部连接装置929连接到连接端口923,开发设备300直接从外部连接装置929获得声学信号数据或者图像信号数据,或者将该声学信号数据或者图像信号数据直接提供到外部连接装置929。

[0171] 通信设备925是例如由连接到通信网络931的通信装置等配置的通信接口。通信设备925的示例包括:有线或者无线LAN(局域网)、蓝牙、WUSB通信卡(无线USB)、用于光通信的路由器、ADSL(非对称数字用户线)的路由器、各种通信的调制解调器等。基于例如诸如TCP/IP等的预定协议,通信设备925可以与因特网或者另一通信装置交换信号等。此外,连接到通信设备925的通信网络931可以由以有线或者无线方式连接的网络等配置,并且可以是因特网、国内LAN、红外通信、无线电波通信、无线通信等。

[0172] 用户可以利用这种开发设备300开发信息处理设备100要执行的计算机程序。利用信息处理设备100创建的计算机程序可以通过例如连接到开发设备300的读写器200而安装在信息处理设备100内。

[0173] <2.结论>

[0174] 如上所述,根据本公开实施例,通过以对变量和函数分别独立设置安全性的方式,配置信息处理设备100要执行的列表处理模块,并且利用防篡改函数保护该列表处理模块,可以实现有助于开发安全应用的编程操作系统。由于可以对该程序要使用的变量和函数分别进行唯一的安全设置,所以可以轻而易举地实现严密安全性的实现。

[0175] 该列表处理模块提供有如解释器的基本功能,并且当使得信息处理设备100执行程序时,不需要预编译,并且通过进一步添加安全函数,可以对执行该程序要使用的IC芯片本身进行调试。因此,开发步骤的数量小于现有技术中的应用开发模型中的开发步骤的数量,并且可以在较短的时间内开发应用。此外,可以在信息处理设备100内提供的非易失性存储器内动态并直接地定义变量和函数,这样可以灵活地开发应用。

[0176] 在利用分别设置的验证密钥进行验证之前,根据本公开实施例的信息处理设备

100执行的应用使用的安全锁定变量和函数不可用。因此，在定义函数时，需要验证在定义函数的过程中由函数用作自变量的安全锁定变量和函数。然而，在定义了函数之后使用函数时，仅通过对定义函数执行安全验证，就可以无条件地使用在内部用作自变量的变量和函数。利用这种配置，不需要准备所有验证密钥并且不需要每次都执行验证，因此，可以方便应用的操作。

[0177] 在下载该程序后，容易重写以脚本语言编写的程序。然而，该特征使得在开放式平台上关心误用问题。利用根据本公开实施例的信息处理设备100的配置，可以防止程序被误用。

[0178] 尽管参考附图详细描述了本公开的优选实施例，但是本公开并不局限于该示例。显而易见，本技术领域内的技术人员可以进行各种修改和校正，而不脱离所附权利要求书中描述的技术思想的范围，并且应当明白，所有这些修改和校正都属于本公开的技术范围。

[0179] 本公开含有与于2011年2月10日向日本专利局提交的第JP2011-027652号日本优先权专利申请公开的主题有关的主题，在此通过引用包括该专利申请的全部内容。

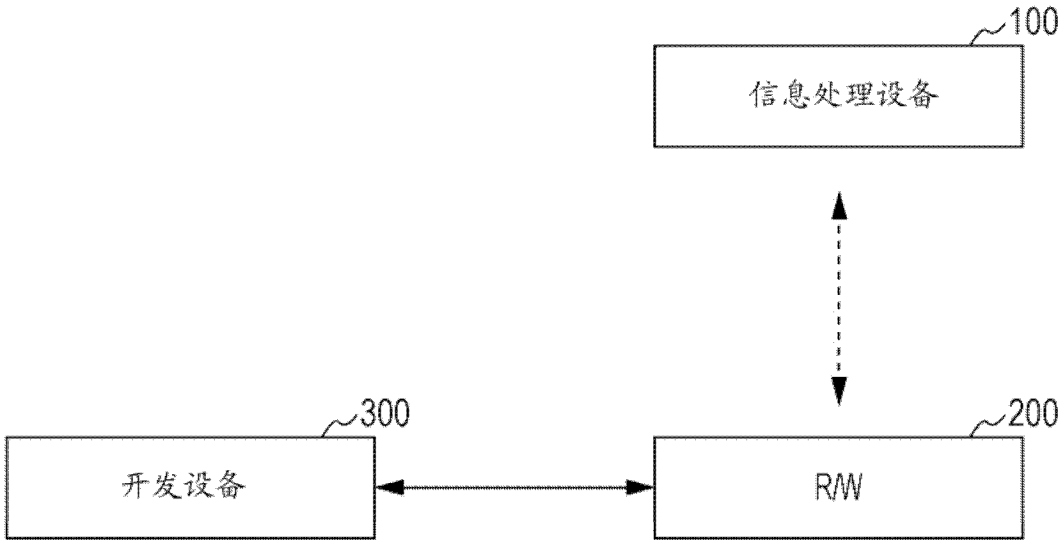


图1

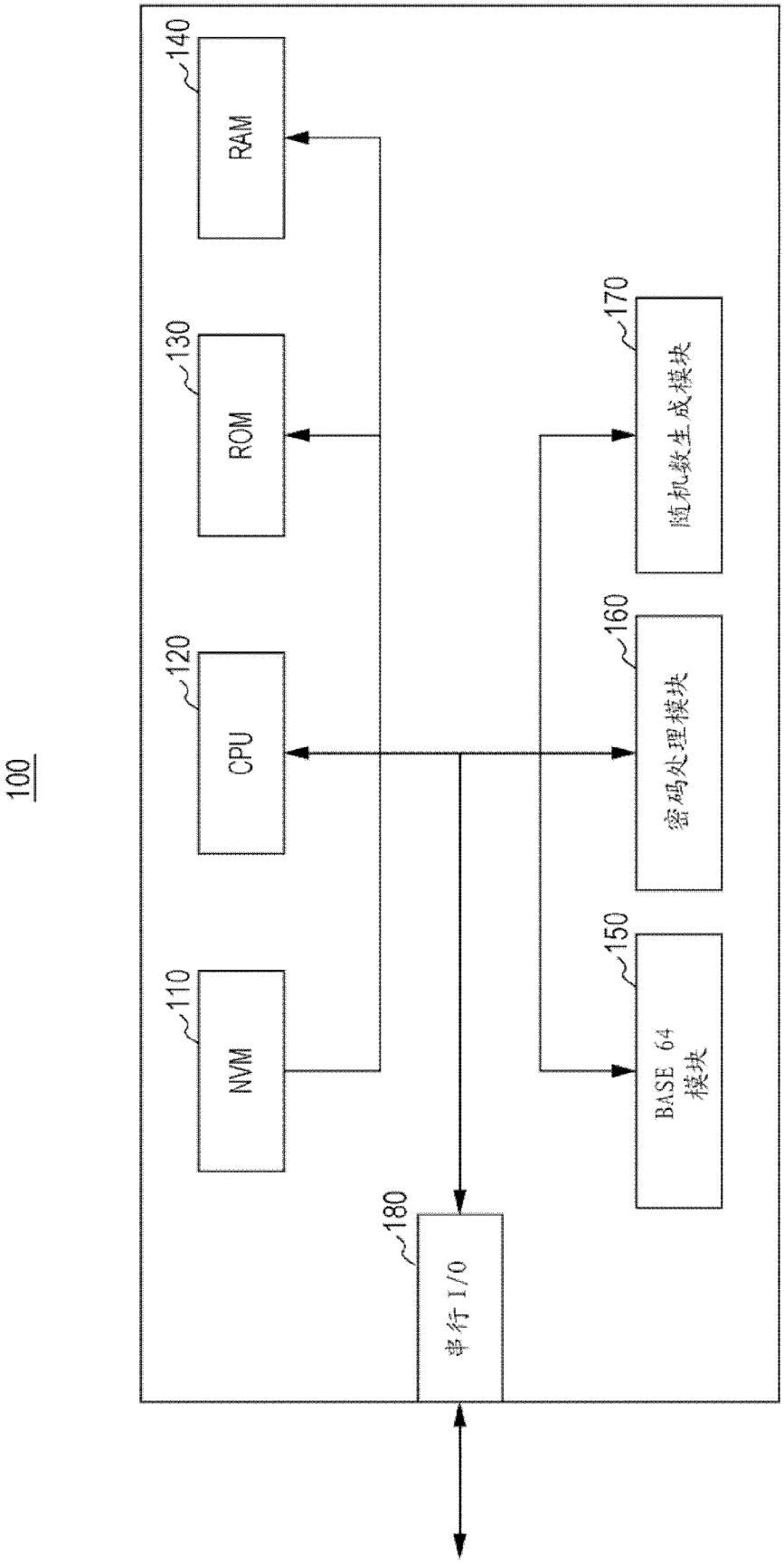

100
140
RAM
130
ROM
120
CPU
110
NVM
170
随机数生成模块
160
密码处理模块
150
BASE 64
模块
180
串行 I/O

图2

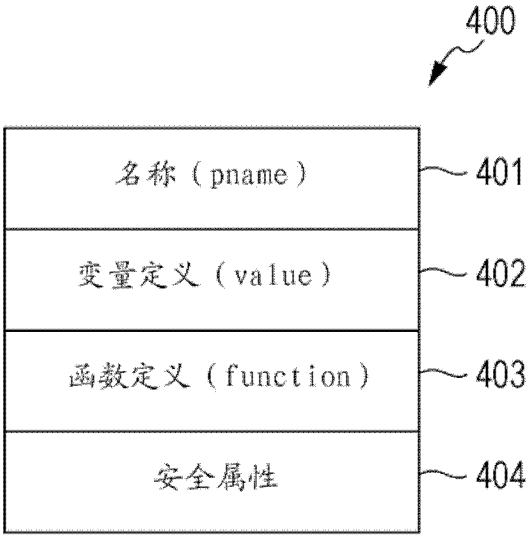

400
名称（pname）
401
变量定义（value）
402
函数定义（function）
403
安全属性
404

图3

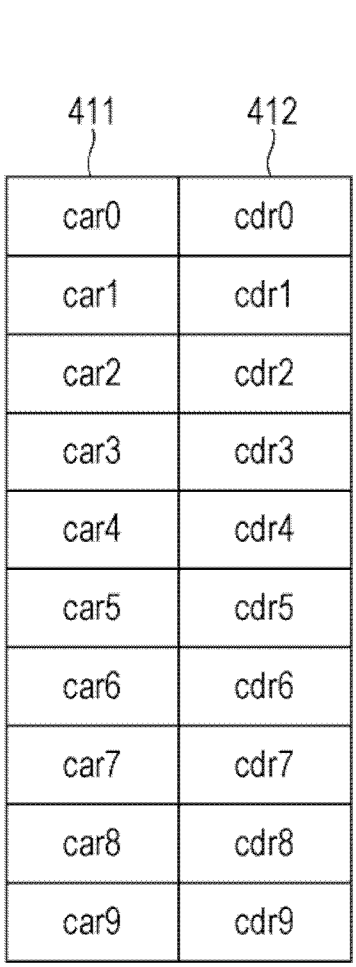

410
411
412
car0
cdr0
car1
cdr1
car2
cdr2
car3
cdr3
car4
cdr4
car5
cdr5
car6
cdr6
car7
cdr7
car8
cdr8
car9
cdr9

图4

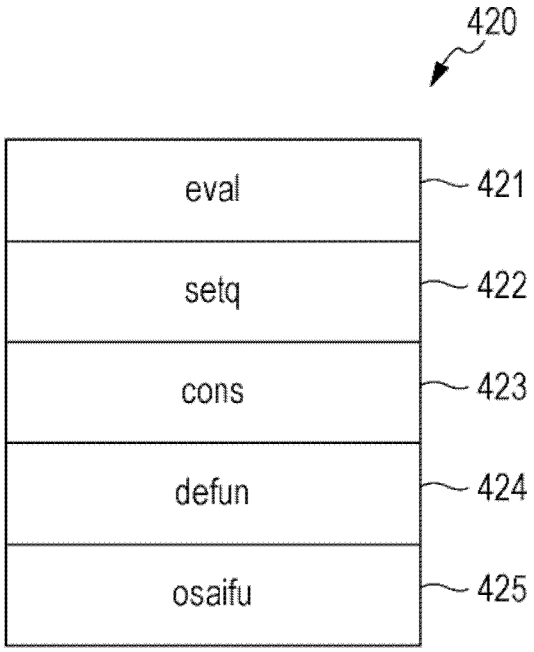

420
eval
421
setq
422
cons
423
defun
424
osaifu
425

图5

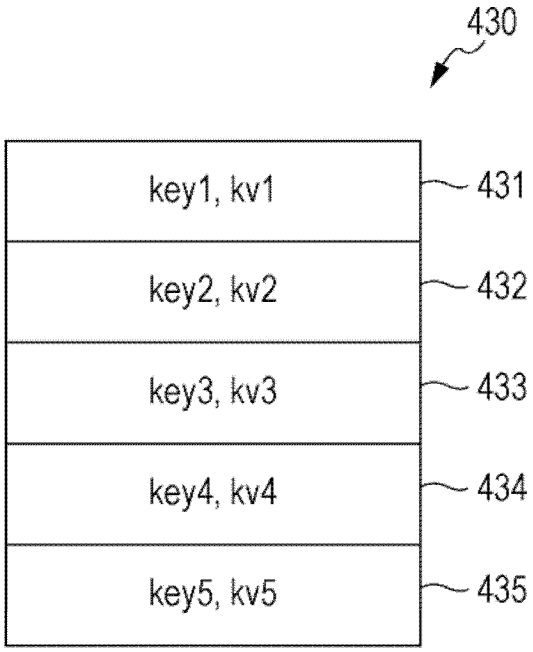

430
key1, kv1
431
key2, kv2
432
key3, kv3
433
key4, kv4
434
key5, kv5
435

图6

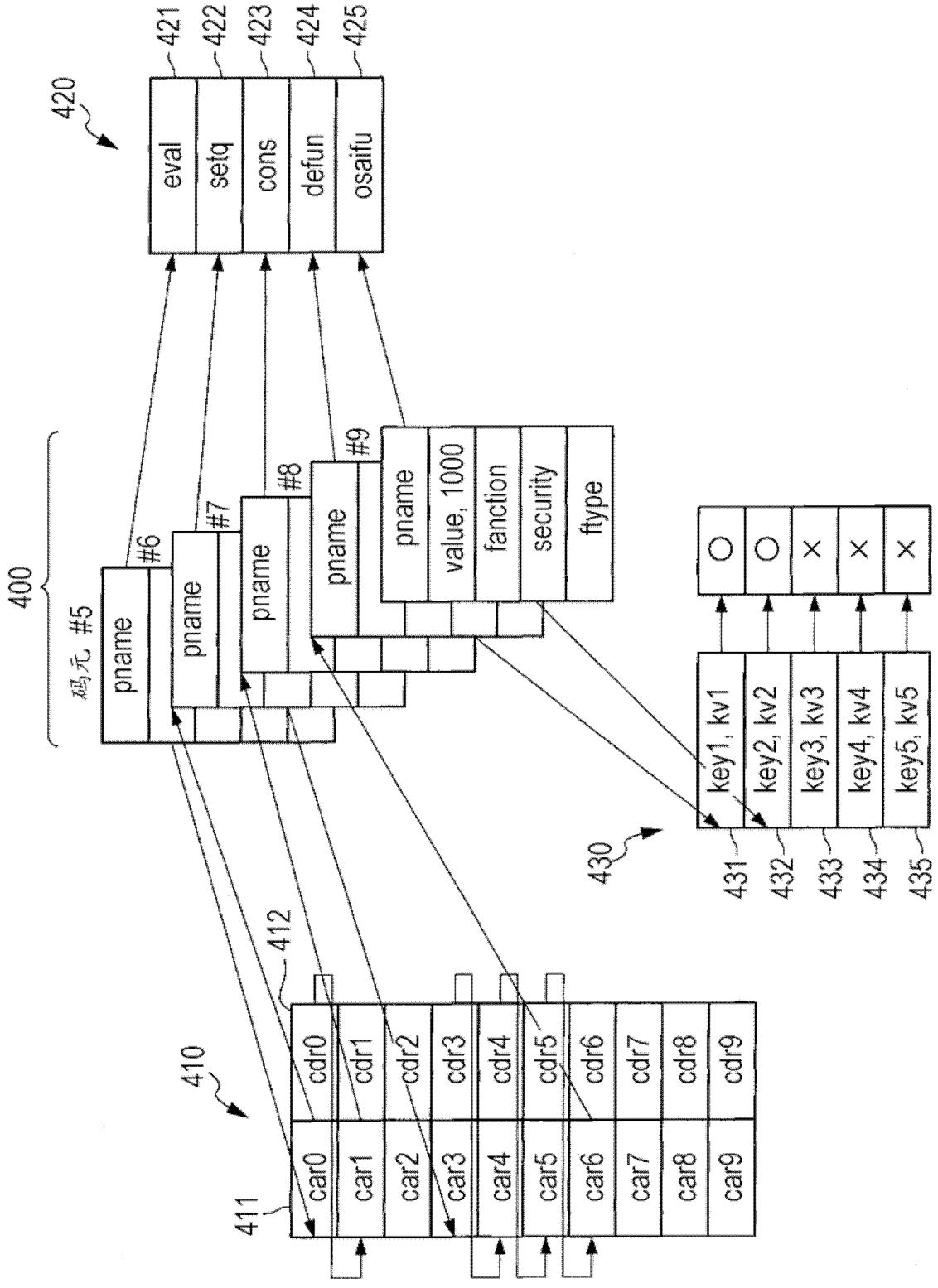

420
421
422
423
424
425
eval
setq
cons
defun
osaifu
400
码元 #5
#6
#7
#8
#9
pname
value, 1000
fanction
security
ftype
○
○
×
×
×
key1, kv1
key2, kv2
key3, kv3
key4, kv4
key5, kv5
430
431
432
433
434
435
410
411
412
car0
car1
car2
car3
car4
car5
car6
car7
car8
car9
cdr0
cdr1
cdr2
cdr3
cdr4
cdr5
cdr6
cdr7
cdr8
cdr9

图7

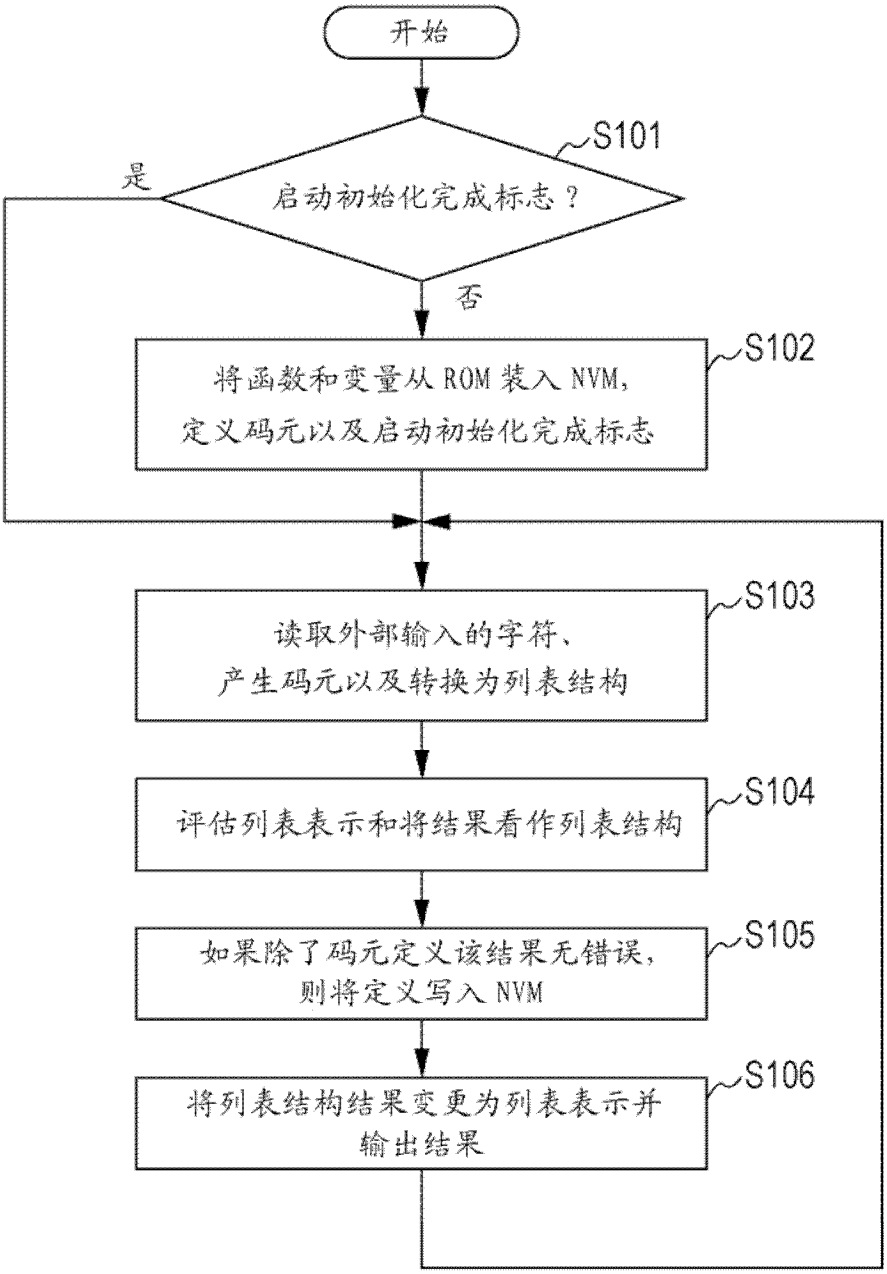

开始
S101
启动初始化完成标志？
是
否
S102
将函数和变量从 ROM 装入 NVM，
定义码元以及启动初始化完成标志
S103
读取外部输入的字符、
产生码元以及转换为列表结构
S104
评估列表表示和将结果看作列表结构
S105
如果除了码元定义该结果无错误，
则将定义写入 NVM
S106
将列表结构结果变更为列表表示并
输出结果

图8

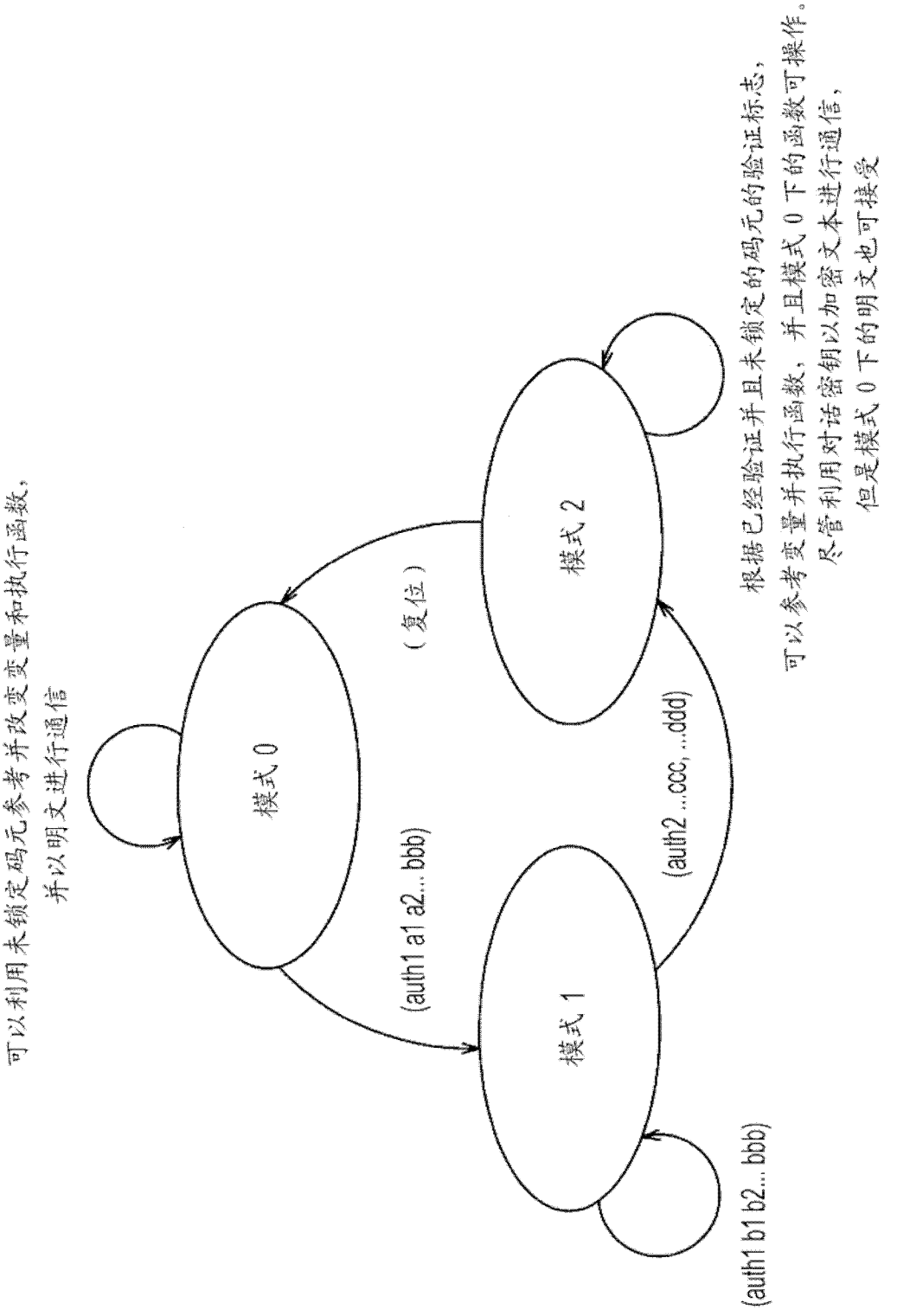

可以利用未锁定码元参考并改变变量和执行函数，
并以明文进行通信
模式 0
模式 1
模式 2
(auth1 a1 a2... bbb)
(复位)
(auth2 ...ccc, ...ddd)
(auth1 b1 b2... bbb)
根据已经验证并且未锁定的码元的验证标志，
可以参考变量并执行函数，并且模式 0 下的函数可操作。
尽管利用对话密钥以加密文本进行通信，
但是模式 0 下的明文也可接受

图9

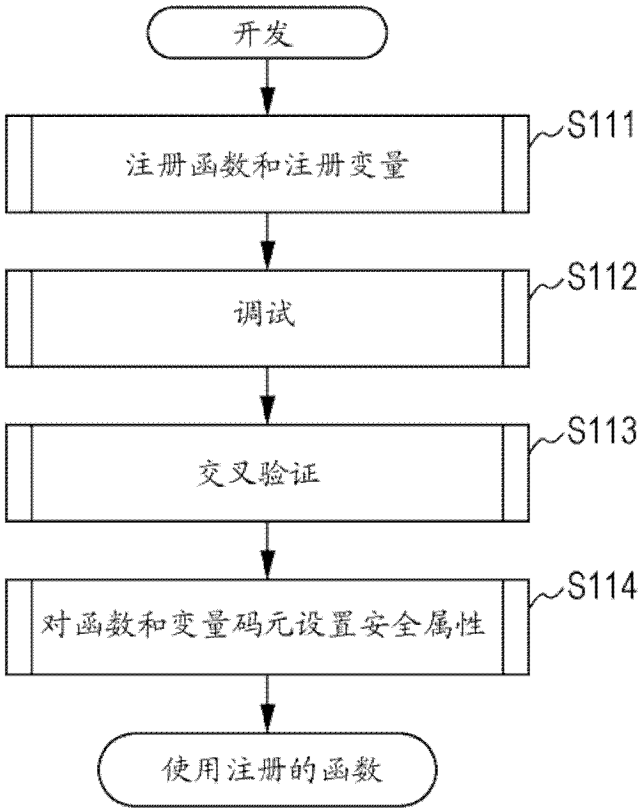

开发
注册函数和注册变量
S111
调试
S112
交叉验证
S113
对函数和变量码元设置安全属性
S114
使用注册的函数

图10

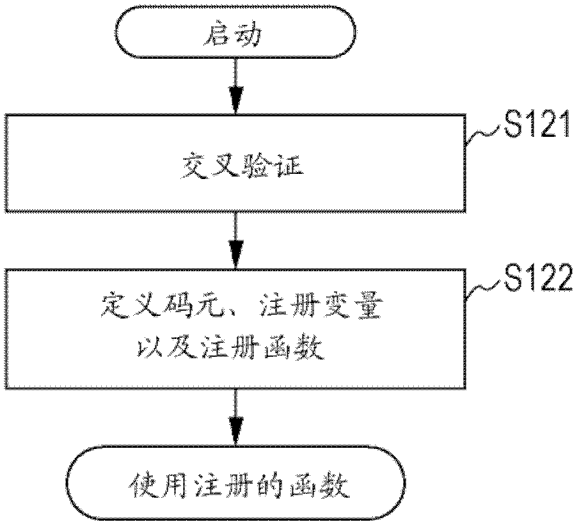

启动
交叉验证
S121
定义码元、注册变量
以及注册函数
S122
使用注册的函数

图11

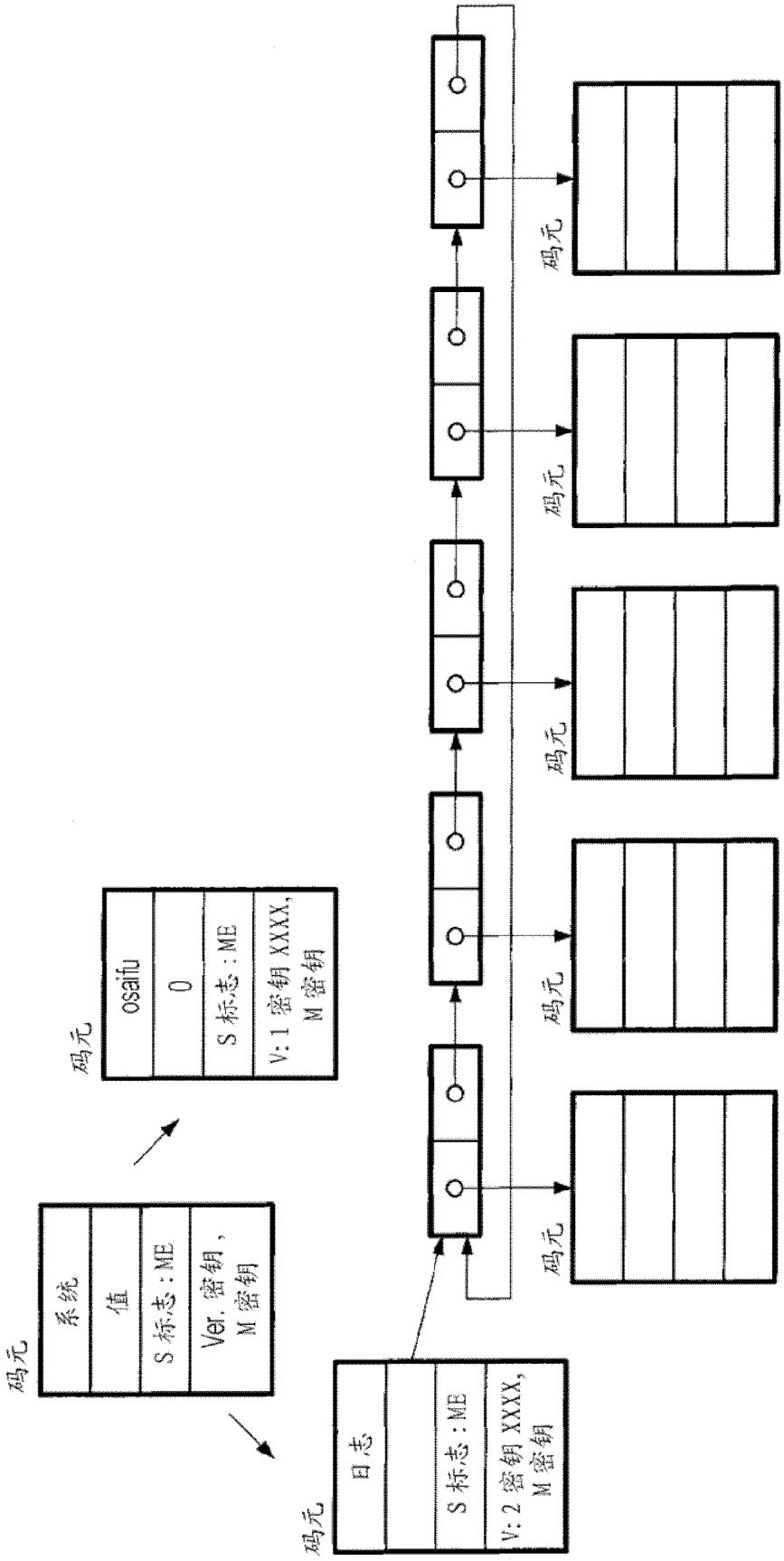

码元
系统
值
S 标志：ME
Ver, 密钥，M 密钥
码元
osaifu
0
S 标志：ME
V：1 密钥 XXXX，M 密钥
码元
日志
S 标志：ME
V：2 密钥 XXXX，M 密钥
码元
码元
码元
码元
码元
码元

图12

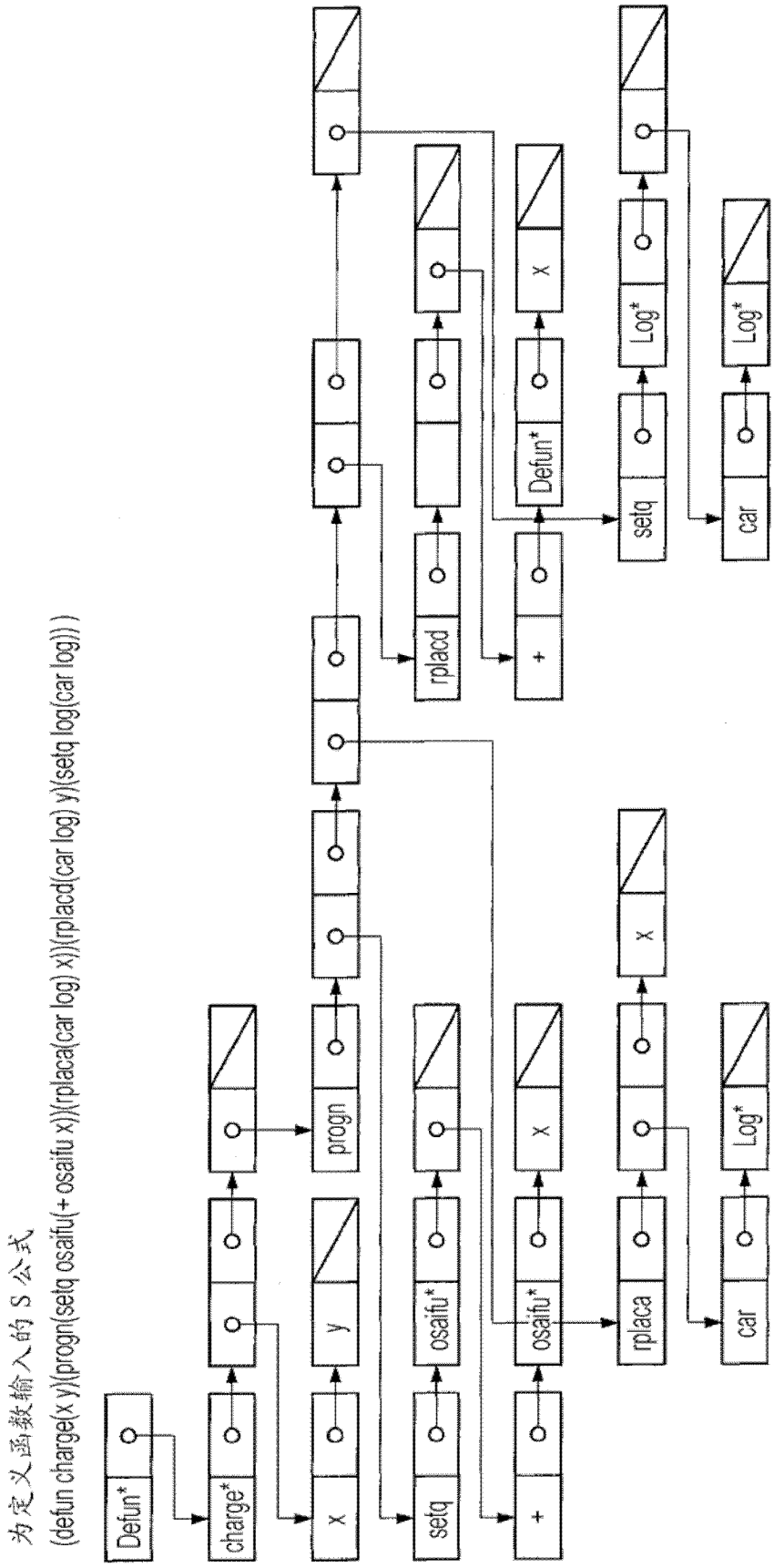

为定义函数输入的 S 公式
(defun charge(x y)(progn(setq osaifu(+ osaifu x))(rplaca(car log) x))(rplacd(car log) y)(setq log(car log)))
Defun*
charge*
x
y
progn
setq
osaifu*
+
rplacd
Defun*
x
setq
Log*
car
Log*
x
rplaca
car
Log*

图13

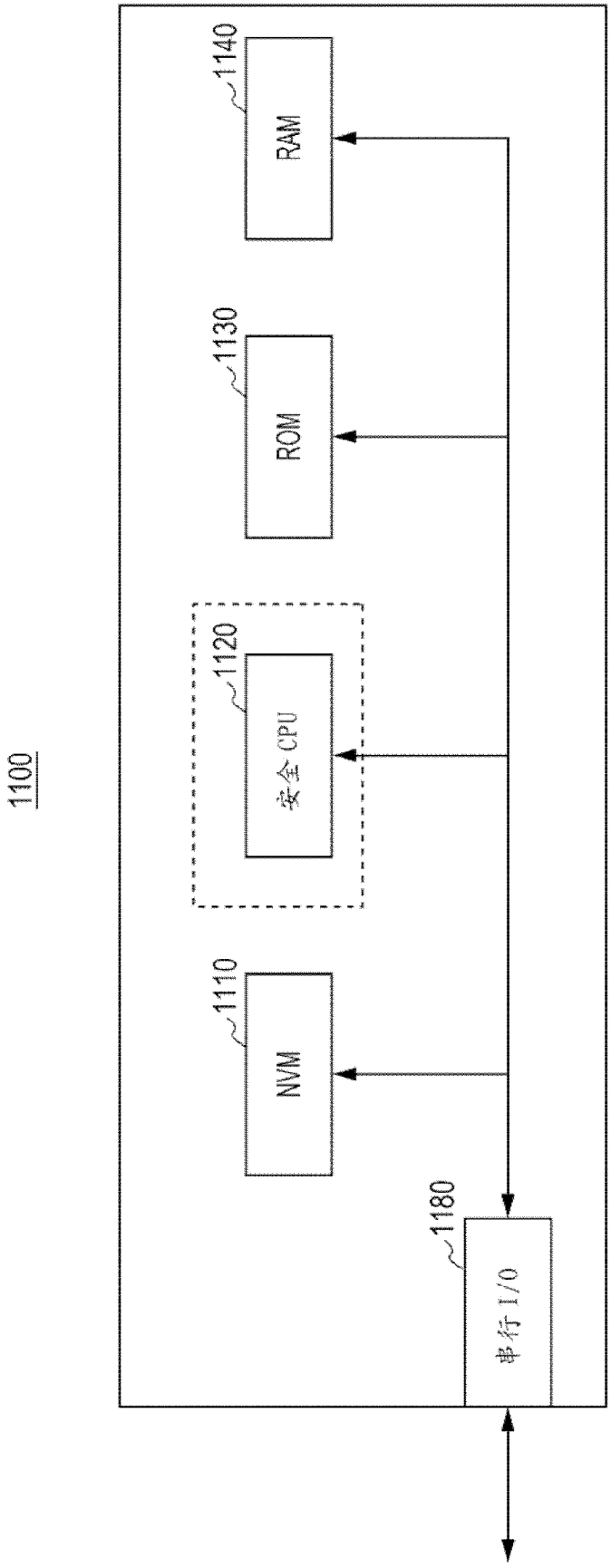


图14

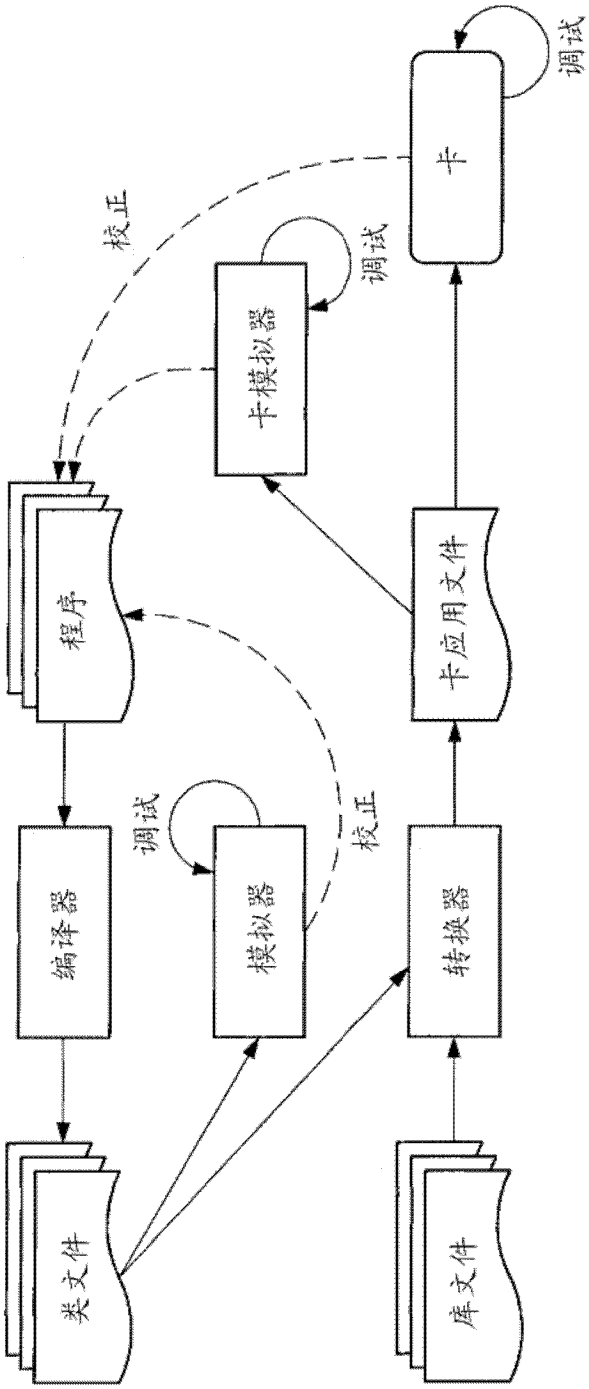


图15

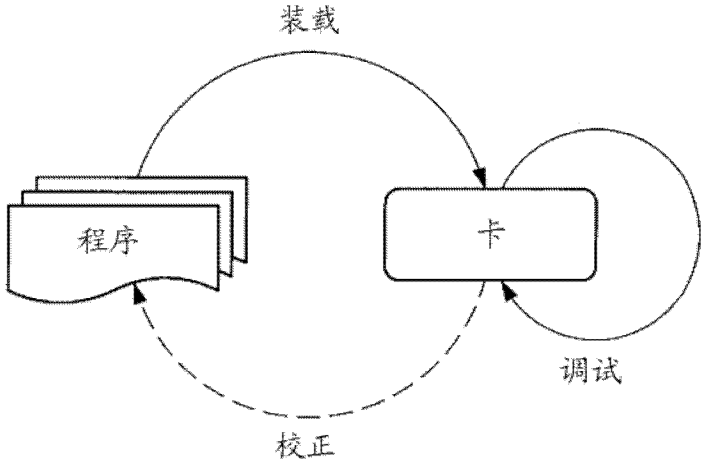


图16

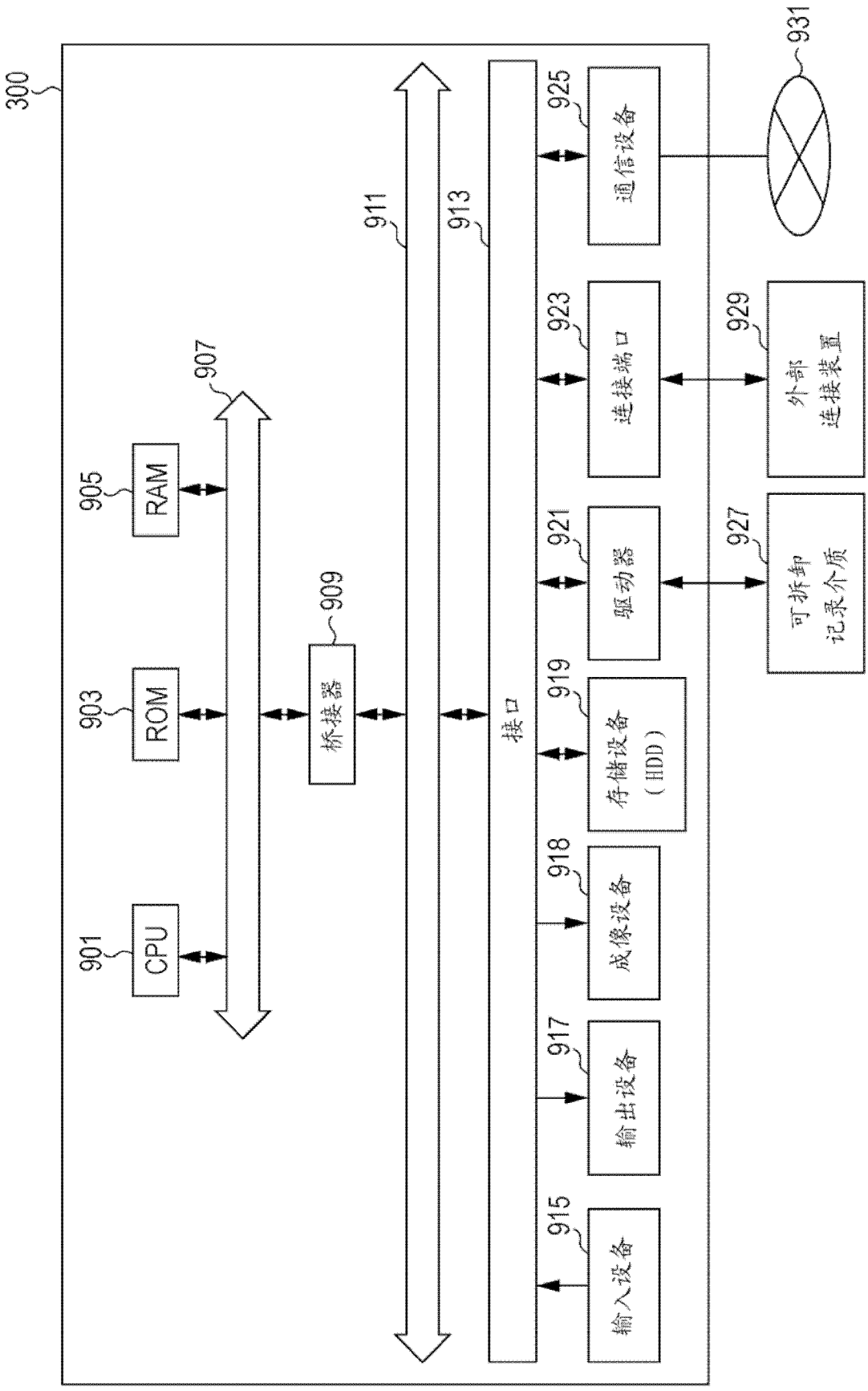

300
901
CPU
903
ROM
905
RAM
907
909
桥接器
911
913
接口
915
输入设备
917
输出设备
918
成像设备
919
存储设备
(HDD)
921
驱动器
923
连接端口
925
通信设备
927
可拆卸
记录介质
929
外部
连接装置
931

图17