(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 101477612 B
(45)授权公告日 2015.09.09

(21)申请号 200910002315.9
(22)申请日 2009.01.04
(30)优先权数据
12/003858 2008.01.02 US
(73)专利权人 ARM有限公司
地址 英国剑桥郡
(72)发明人 N·C·帕弗 D·克尔肖
(74)专利代理机构 北京东方亿思知识产权代理有限责任公司 11258
代理人 李晓冬
(51)Int.Cl.
G06F 21/53(2013.01)
G06F 21/74(2013.01)

(56)对比文件
CN 1711524 A,2005.12.21,
WO 2007091492 A1,2007.08.16,
JP 2006-33114 A,2006.02.02,
US 2005/0160210 A1,2005.07.21,

审查员 王阜东

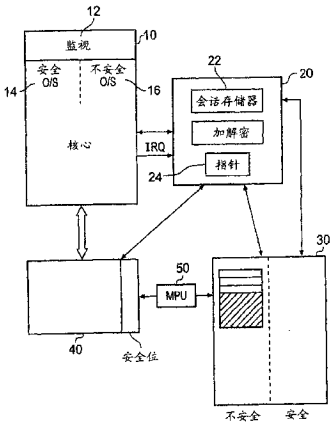
权利要求书3页 说明书10页 附图8页

(54)发明名称

保护从中央处理器发送的以供处理的安全数据的安全性

(57)摘要

本发明涉及保护从中央处理器发送的以供处理的安全数据的安全性。数据处理设备包括数据处理器和另一处理装置，在安全模式下处理数据的数据处理器可以访问安全数据，而在所述不安全模式下它不能访问安全数据，在安全模式下处理数据是在安全操作系统的控制下进行的，在不安全模式下处理数据是在不安全操作系统的控制下进行的；另一处理装置响应于来自数据处理器的请求执行任务；其响应于暂停任务的信号的接收以启动；使用安全密钥处理安全数据；将处理过的安全数据存储到不安全数据储存器；其还响应于接收到继续任务的信号来启动；从不安全数据储存器中检索处理过的安全数据；使用安全密钥恢复处理过的安全数据，安全密钥被安全存储。



CN 101477612 B

1. 一种数据处理设备，包括：

数据处理器，用于在安全模式和不安全模式下处理数据，在所述安全模式下处理数据的所述数据处理器能够访问安全数据，所述数据处理器在所述不安全模式下不能访问所述安全数据，并且所述数据处理器在所述安全模式下处理数据是在安全操作系统的控制下进行的，以及在所述不安全模式下处理数据是在不安全操作系统的控制下进行的；和

另一处理装置，其用来响应于来自所述数据处理器的请求而执行任务，所述任务包括处理至少其中一些是安全数据的数据；其中

所述另一处理装置响应于接收到暂停所述任务的信号，以判定所述信号是否指示延迟的进程状态切换，并且如果不指示则启动：

使用安全密钥处理所述安全数据；和

将所述处理过的安全数据存储到不安全数据储存器中；以及

如果暂停所述任务的所述信号包括延迟的进程状态切换，则暂停所述任务，并且不执行所述安全数据的所述处理和所述处理过的安全数据的所述存储，直到开始所述另一处理装置上的新任务的信号被接收；以及

所述另一处理装置还响应于接收到继续暂停的所述任务的信号，以启动：

从所述不安全数据储存器中检索所述处理过的安全数据；和

使用所述安全密钥恢复所述处理过的安全数据；其中所述安全密钥被安全地存储，使得其不能被操作在所述不安全模式下的其它进程访问。

2. 根据权利要求 1 所述的数据处理设备，其中所述安全数据的所述处理包括使用所述安全密钥加密所述安全数据，并且所述处理过的安全数据的所述恢复包括使用所述安全密钥解密所述加密的安全数据，所述安全密钥被安全地存储在执行所述加密和解密的装置中。

3. 根据权利要求 2 所述的数据处理设备，其中在启动所述数据的加密之前，所述另一处理装置启动从所述数据中生成安全签名，并启动所述安全签名的加密，以及连同所述安全数据一起存储所述安全签名；和

响应于接收到继续所述任务的所述信号，所述另一处理装置启动所述加密的安全签名的检索和所述加密的安全签名的解密，并且在所述加密的安全数据的解密之后，启动从所述解密的加密数据中生成所述安全签名，以及将所述生成的安全签名与所述解密的安全签名进行比较；和

响应于所述生成的安全签名与所述解密的安全签名不匹配，所述另一处理装置向所述数据处理器发出信号以表明所述数据已经被篡改；和

响应于所述生成的安全签名与所述解密的安全签名相匹配，所述另一处理装置使用所述解密的数据继续所述任务。

4. 根据权利要求 2 所述的数据处理设备，其中在加密所述数据之后，所述另一处理装置启动从所述加密的数据中生成安全签名，并启动所述安全签名以及所述加密的安全数据的加密和存储；和

响应于接收到继续所述任务的所述信号，所述另一处理装置启动所述加密的安全签名的检索和所述加密的安全签名的解密，并启动从所述加密的数据中生成所述安全签名，并将所述生成的安全签名和所述解密的安全签名进行比较；和

响应于所述生成的安全签名与所述解密的安全签名不匹配，所述另一处理装置向所述数据处理器发出信号以表明所述数据已经被篡改；和

响应于所述生成的安全签名与所述解密的安全签名相匹配，所述另一处理装置解密所述加密的数据，并使用所述解密的数据继续所述任务。

5. 根据权利要求 1 所述的数据处理设备，其中所述安全数据的所述处理包括从所述数据中生成安全签名以及使用所述安全密钥加密所述安全签名，并且所述处理过的安全数据的所述恢复包括解密所述加密的安全签名，以及从所述检索的数据中生成所述安全签名，并将所述生成的安全签名和所述解密的安全签名进行比较；和

响应于所述生成的安全签名与所述解密的安全签名不匹配，所述另一处理装置向所述数据处理器发出信号以表明所述数据已经被篡改；和

响应于所述生成的安全签名与所述解密的安全签名相匹配，所述另一处理装置使用所述检索的数据继续所述任务。

6. 根据权利要求 3 所述的数据处理设备，其中所述安全数据的所述加密和所述签名的所述生成是使用相同的算法进行的。

7. 根据权利要求 2 所述的数据处理设备，其中执行所述加密和解密的所述装置是所述另一处理装置。

8. 根据权利要求 7 所述的数据处理设备，其中所述另一处理装置包括加解密处理装置，所述加解密处理装置使用至少一个密钥执行数据的加解密处理，所述安全数据是由包括所述至少一个密钥的所述安全密钥加密的。

9. 根据权利要求 2 所述的数据处理设备，其中执行所述加密和解密装置的所述装置是通过专用数据通信路径连接到所述另一处理装置的加解密处理装置，该专用数据通信路径是其它装置不能访问的。

10. 根据权利要求 2 所述的数据处理设备，其中所述另一处理装置包括用来执行除加解密以外的功能的电路，所述另一处理装置包括附加的加解密电路以加密和解密所述安全数据，所述另一处理装置包括所述安全密钥的安全存储位置。

11. 根据权利要求 7 所述的数据处理设备，其中所述另一处理装置包括所述安全密钥的只写存储位置。

12. 根据权利要求 1 所述的数据处理设备，其中所述安全密钥在操作在安全模式下的所述数据处理器的控制下被存储在所述数据处理设备中，使得其只能由在安全模式下操作的所述数据处理器访问。

13. 根据权利要求 2 所述的数据处理设备，其中所述安全密钥是在制造时被设置的，并永久地安全存储在执行所述安全数据的所述加密和解密的所述装置中，使得其不能被其它装置访问。

14. 根据权利要求 2 所述的数据处理设备，其中执行所述安全数据的所述加密和解密的所述装置包括随机数生成器，所述安全密钥是由所述随机数生成器在复位时生成的。

15. 根据权利要求 1 所述的数据处理设备，其中所述不安全数据储存器包括存储器的不安全部分。

16. 根据权利要求 1 所述的数据处理设备，其中暂停所述任务的所述信号是由所述数据处理器响应于中断或异常而生成的。

17. 根据权利要求 1 所述的数据处理设备，其中暂停所述任务的所述信号是由所述数据处理器响应于信号而生成的，该信号表明所述数据处理器从操作在所述安全模式或不安全模式中的一种到所述安全模式或不安全模式中的另一种的上下文切换。

18. 根据权利要求 1 所述的数据处理设备，其中执行所述任务的所述请求是从在所述不安全模式下操作的所述数据处理器发出的。

19. 根据权利要求 18 所述的数据处理设备，其中所述数据处理器被配置成响应于所述数据处理器在不安全模式下执行的程序代码向所述另一处理装置发出执行所述任务的所述请求。

20. 根据权利要求 1 所述的数据处理设备，其中所述另一处理装置包括不可编程硬件。

21. 根据权利要求 1 所述的数据处理设备，其中所述另一处理装置被配置成不在操作系统的控制下处理程序代码。

22. 一种在数据处理设备上处理数据的方法，所述数据处理设备包括数据处理器和另一处理装置；所述数据处理器用于在安全模式和不安全模式下处理数据，在所述安全模式下处理数据的所述数据处理器能够访问安全数据，所述数据处理器在所述不安全模式下不能访问所述安全数据，所述数据处理器在所述安全模式下处理数据是在安全操作系统的控制下进行的，并且在所述不安全模式下处理数据是在不安全操作系统的控制下进行的；所述另一处理装置用来响应于来自所述数据处理器的请求而执行任务，所述任务包括处理至少其中一些是安全数据的数据；所述方法包括以下步骤：

在所述数据处理器上处理数据；

将来自所述数据处理器的请求发送到所述另一处理装置以执行任务，所述任务包括处理至少其中一些是安全数据的数据；

在所述另一处理装置上开始所述任务；

向所述另一处理装置发送暂停所述任务的信号；

所述另一处理装置判定暂停所述任务的所述信号是否包括延迟的进程状态切换，并且如果不包括则：

暂停所述任务，并启动使用安全密钥对所述安全数据的处理及将所述处理过的安全数据存储在不安全数据储存器中；并且如果所述信号包括所述延迟的进程状态切换，则：

暂停所述任务并且不执行所述安全数据的所述处理和所述处理过的安全数据的所述存储，直到开始所述另一处理装置上的新任务的信号被接收；以及

从所述数据处理器生成指示所述另一处理装置继续所述任务的信号；

所述另一处理装置响应于所述信号从所述不安全数据储存器中检索所述处理过的安全数据，并使用所述安全密钥恢复所述检索的数据，所述安全密钥被安全地存储，使得其不能由其它装置访问。

保护从中央处理器发送的以供处理的安全数据的安全性

技术领域

[0001] 本发明的领域涉及数据处理，并且更具体地涉及使用附加的处理装置处理安全数据和不安全 (non-secure) 数据，以在第一数据处理器的控制下执行任务。

背景技术

[0002] 与主处理器相结合使用加速器或协处理器来增强性能是已知的，加速器或协处理器执行主处理器请求的任务。尽管通过提供附加的处理能力可以增强性能是显然的，但这类系统可能存在的缺陷是与处理器和加速器之间的通信相关的开销。例如在安全操作系统和不安全操作系统的控制下处理安全数据和不安全数据两者的情况下，与管理在处理器和加速器之间传送的数据的安全性状态相关的通信开销可能是非常高的。在这种情况下，将大任务发送给加速器来执行才可能是有利的，只占用几十个或甚至几百个周期 (cycle) 的任务所具有的与之关联的通信开销如同它们增进的处理性能一样多或更多。

发明内容

[0003] 本发明的第一方面提供了一种数据处理设备，包括数据处理器和另一 (further) 处理装置，所述数据处理器用于在安全模式和不安全模式下处理数据，在所述安全模式下处理数据的所述数据处理器可以访问安全数据，所述数据处理器在所述不安全模式下不可以访问所述安全数据，并且在所述安全模式下处理数据是在安全操作系统的控制下进行的，在所述不安全模式下处理数据是在不安全操作系统的控制下进行的；所述另一处理装置用于响应于来自所述数据处理器的请求而执行任务，所述任务包括处理至少其中一些是安全数据的数据；其中所述另一处理装置响应于接收到暂停所述任务的信号，以启动：使用安全密钥处理所述安全数据；和将所述处理过的安全数据存储到不安全数据储存器 (store) 中；所述另一处理装置还响应于接收到继续所述任务的信号，以启动：从所述不安全数据储存器中检索 (retrieval) 所述处理过的安全数据；以及使用所述安全密钥恢复所述处理过的安全数据；其中所述安全密钥是安全地存储的，使得其不能被工作在所述不安全模式下的其它进程访问。

[0004] 本发明认识到在另一处理装置处理的数据包括安全数据的情况下，使用该另一处理装置为可以在安全或不安全模式下操作的数据处理器执行特定任务的通信开销可能巨大。原因是该安全数据必须与数据处理器的不安全模式保持隔离，并且安全数据的管理通常是由数据处理器的安全操作系统完成的。然而，无论何时正由另一处理装置执行的任务被中断并且需要被暂停时，调用安全操作系统来管理由该另一处理装置正在处理的安全数据将需要许多操作周期，因此在性能方面代价会非常高。本发明通过提供所述另一处理装置解决了此问题，该另一处理装置本身具有的功能使得其能够使它正在处理的安全数据在存储到数据处理器可以访问的不安全数据储存器之前被处理。该处理是使用其它不安全进程不能访问的密钥完成的。通过提供本身具有这种功能的所述另一处理装置，提供了在无需安全操作系统介入的情况下维持安全数据的安全性的可能性。而且，在所述另一处理装

置原本处理的任务要被继续时，可以使用其它不安全进程不能访问的安全密钥在另一处理装置的控制下恢复安全数据，然后可以恢复所述另一处理装置的状态，并且继续该任务，其中可以在不安全模式下操作的数据处理器本身只能受控访问安全数据。

[0005] 因此，所述另一处理装置本身以有效方式维持了由所述另一处理装置处理的数据的安全性，并且与发送任务给另一处理装置相关的开销被降低，因此可以发送轻小的任务，获得性能益处 (performancebenefit)。

[0006] 在一些实施例中，所述安全数据的所述处理包括使用所述安全密钥加密所述安全数据，并且所述处理过的安全数据的所述恢复包括使用所述安全密钥解密所述加密的安全数据，所述安全密钥被安全地存储在执行所述加密和解密的装置中。

[0007] 保护安全数据的一种高效方式是在将其存储到不安全储存器之前在所述另一处理装置的控制下对其加密。这样数据处理器本身不能访问该安全数据。

[0008] 在一些实施例中，在启动所述数据的加密之前，所述另一处理装置启动从所述数据中生成安全签名，并启动所述安全签名的加密以及将所述安全签名连同所述安全数据一起存储；并且响应于接收到继续所述任务的所述信号，所述另一处理装置启动所述加密的安全签名的检索以及所述加密的安全签名的解密，并且在所述加密的安全数据的解密之后，启动从所述解密的加密数据中生成所述安全签名，并将所述生成的安全签名和所述解密的安全签名进行比较；并且响应于所述生成的安全签名与所述解密的安全签名不匹配，所述另一处理装置向所述数据处理器发出信号以表明所述数据已经被篡改；并且响应于所述生成的安全签名与所述解密的安全签名相匹配，所述另一处理装置使用所述解密的数据继续所述任务。

[0009] 尽管可以仅对安全数据加密以保护其不被不安全进程访问，但从所述数据中生成安全签名，然后用数据加密该安全签名可能是有利的。该签名然后可用在解密上，以确定所存储的加密数据是否已经被篡改。如果解密后生成的安全签名与存储的加密的安全签名不匹配，则可向数据处理器发出中止 (abort) 信号以通知数据处理器该数据已经被篡改。如果安全签名确实匹配，则处理可以继续。这样，不仅可以安全地存储安全数据，而且在进程被中断时，其完整性可被验证。

[0010] 在其它实施例中，安全签名是通过以下方式从加密的数据，而不是从解密的数据中生成的：加密所述数据之后，所述另一处理装置启动从所述加密的数据中生成安全签名，并且启动所述安全签名连同所述加密的安全数据的加密和存储；以及响应于接收到继续所述任务的所述信号，所述另一处理装置启动所述加密的安全签名的检索和所述加密的安全签名的解密，并启动从所述加密的数据中生成所述安全签名，将所述生成的安全签名和所述解密的安全签名进行比较；并且响应于所述生成的安全签名与所述解密的安全签名不匹配，所述另一处理装置向所述数据处理器发出信号以表明所述数据已经被篡改；以及响应于所述生成的安全签名与所述解密的安全签名相匹配，所述另一处理装置解密所述加密的数据，并使用所述解密的数据继续所述任务。

[0011] 在一些实施例中，所述安全数据的所述处理包括从所述数据中生成安全签名以及使用所述安全密钥加密所述安全签名，所述处理过的安全数据的所述恢复包括解密所述加密的安全签名，并且从所述检索的数据中生成所述安全签名以及将所述生成的安全签名和所述解密的安全签名进行比较；并且响应于所述生成的安全签名与所述解密的安全签名不

匹配，所述另一处理装置向所述数据处理器发出信号以表明所述数据已经被篡改；并且响应于所述生成的安全签名与所述解密的安全签名相匹配，所述另一处理装置使用所述检索的数据继续所述任务。

[0012] 由于在所述另一处理装置之外，安全数据是不能被理解的，所以正在被处理的安全数据可能不需要保护其自身不被查看。然而，安全数据不被篡改可能是重要的，因为篡改安全数据可能是试图访问安全信息的方式。在这种情况下，数据本身没有被加密，不过生成了安全签名，并且该安全签名可用来检查数据是否已经被篡改过。

[0013] 在一些实施例中，所述安全数据的所述加密和所述签名的所述生成是使用相同的算法进行的。

[0014] 在安全数据的加密是使用专门为此功能产生的加解密引擎 (crypto-engine) 完成的情况下，为了降低添加此引擎的开销，使用与用来加密安全数据相同的算法来产生签名（例如哈希）可能是有利的。

[0015] 在一些实施例中，执行所述加密和解密的所述装置是所述另一处理装置。

[0016] 如果所述另一处理装置本身执行加密和解密，则显然安全数据的安全性被增强，因为安全数据没有离开该装置，直到它本身被加密，因此对安全数据的访问是容易控制的。

[0017] 在一些实施例中，所述另一处理装置包括加解密处理 (cryptography processing) 装置，所述加解密处理装置使用至少一个密钥执行数据的加解密处理，所述安全数据是由包括所述至少一个密钥的所述安全密钥来加密的。

[0018] 该技术在应用于加解密处理装置时特别有用。这些装置使用密钥加密和解密数据，并且显然重要的是，这些密钥不能被不安全模式访问。因此，对这些另一处理装置来说，重要的是包括这些加解密密钥 (cryptography key) 的安全数据被保持安全。而且，这些装置已经包括加解密功能性，因此，可以使用这些装置及所述另一处理装置用于此功能的安全密钥来执行加密安全数据（在处理过程中用来加密数据的加解密密钥）。这样，可以使用所述另一处理装置本身的功能性，来加密和解密安全数据。因此，与包括该功能有关的面积开销很小。

[0019] 在一些实施例中，执行所述加密和解密的所述装置是通过专用数据通信路径连接到所述另一处理装置的加解密处理装置，其它装置不可以访问该专用数据通信路径。

[0020] 尽管在一些实施例中，该另一处理装置本身执行加解密功能，但在其它实施例中，可以由单独的加解密装置来执行此功能。在这种情况下，用来将安全数据传送到此加解密装置的数据路径必须是专用于此目的且不可被其它处理器访问的安全数据路径。

[0021] 在一些实施例中，所述另一处理装置包括用来执行除加解密以外的功能的电路，所述另一处理装置包括附加的加解密电路以加密和解密所述安全数据，所述另一处理装置包括所述安全密钥的安全存储位置。

[0022] 如果该另一处理装置不是加解密处理装置，则其可以象以前提到的那样使用单独的加解密处理装置，或者可以给予其附加的加解密电路，使得其可以在安全数据除了在加密状态外根本不离开所述另一处理装置的情况下加密和解密所述安全数据。在这种情况下，该另一处理装置包括安全密钥的存储位置，使得该安全密钥在该装置内可用于加密和解密，但不可用于任何其它装置。

[0023] 在一些实施例中，所述另一处理装置包括所述安全密钥的只写存储位置。

[0024] 将安全密钥存储在该另一处理装置上的只写存储位置可能是有利的。这样，安全密钥可被其它装置更新，但不能被其它装置读取。

[0025] 在一些实施例中，所述安全密钥是在操作在安全模式下的所述数据处理器的控制下，存储在所述数据处理设备中的，使得只有操作在安全模式下的所述数据处理器才可以访问所述安全密钥。

[0026] 例如可以在操作在安全模式下的数据处理器的控制下从数据处理设备的安全储存器中检索安全密钥。在希望不时地更新安全密钥的情况下这可能是有利的。从安全性的观点来看，这是可接受的，只要其存储在安全存储位置，使得其只能由其它装置在安全模式下操纵。在一些实施例中，可能只能在安全特权模式下访问安全密钥，这样安全性会更高。

[0027] 在一些实施例中，所述安全密钥是在制造时被设定的，并永久地安全存储在执行所述安全数据的所述加密和解密的所述装置中，使得其不可被其它装置访问。

[0028] 在安全模式下访问密钥的一种替代方式是将密钥永久地存储在加密安全数据的装置内，并使其不能被其它装置访问。这是保证安全密钥在该装置内保持安全并且不能被外界访问的一种简单方式。不过，这意味着密钥永远不能被改变或更新。

[0029] 生成并存储安全密钥的一种可替代方式是在加密和解密装置内具有随机数生成器，该随机数生成器在复位时 (at reset) 生成并存储安全密钥。在其它实施例中，可以使用伪随机数生成器，其中现有信号或值的秘密集合被组合，并从该组合中生成一数。

[0030] 尽管不安全数据储存器可包括许多不同的东西，但在一些实施例中，它包括存储器 (memory) 的不安全部分。

[0031] 在一些实施例中，暂停所述任务的所述信号是由所述数据处理器响应于中断或异常而生成的。

[0032] 尽管暂停任务的信号可以响应于多种情况而生成，但在一些实施例中，暂停任务的信号是响应于中断或异常而生成的。

[0033] 在一些实施例中，暂停所述任务的所述信号是由所述数据处理器响应于表示所述数据处理器从操作在所述安全模式或不安全模式中的一种切换到所述安全模式或不安全模式中的另一种的上下文切换的信号而生成的。

[0034] 数据处理器从安全模式到不安全模式的上下文切换涉及其它数据处理装置暂停另一处理装置的处理，并将其状态存储在存储器中，使得当数据处理器切换回它之前所在的模式时可以重新开始处理。

[0035] 在这种情况下，重要的是维持它之前正在处理的安全数据的安全性。这是通过使用加密和解密装置而不使用数据处理器的安全操作系统来完成的，所述加密和解密装置是由该另一处理装置本身控制的。因此，数据的安全性是通过该另一处理装置本身维持的，并且不需要调用安全操作系统来控制数据的安全性，调用安全操作系统在性能上是高代价的。

[0036] 在一些实施例中，执行所述任务的所述请求是从操作在所述不安全模式下的所述数据处理器中发出的。

[0037] 如果执行任务的请求是从不安全模式下的数据处理器中发出的，那么该另一处理装置对安全数据的处理必须明确地与数据处理器本身保持隔离，因为数据处理器本身操作在不安全模式。在这种情况下，响应于暂停处理的请求在将安全数据从该另一处理装置中

存储出去(storedout of the further processing means)之前对所述安全数据的加密在维持安全性而不涉及数据处理器本身方面是重要的。

[0038] 在一些实施例中,所述数据处理器被配置成响应于所述数据处理器在不安全模式下正执行的程序代码向所述另一处理装置发出执行所述任务的所述请求。

[0039] 本发明的实施例允许由数据处理器在诸如不安全用户模式之类的不安全模式下执行的诸如用户代码之类的代码,调用该另一处理装置本身来执行包括安全数据的代码,而不需要涉及安全操作系统。在允许以高效方式处理安全数据时这是非常有利的。这可用在例如使用安全密钥对数据进行的加解密处理中,而不需要涉及安全操作系统。

[0040] 在一些实施例中,所述另一处理装置包括不可编程硬件。

[0041] 该另一处理装置可包括简单的不可编程硬件装置,由于其不能被用户软件攻击,因而特别安全。因此,只要安全数据保持在该装置内,并且仅以加密形式离开该装置,其安全性可以被保证。

[0042] 在一些实施例中,所述另一处理装置被配置成不在操作系统的控制下处理程序代码。

[0043] 该另一处理装置可以是不包括操作系统的简单装置,这使得装置内的安全数据的控制更直接。它还使该装置成为能非常高效地执行预定处理步骤的高效装置。

[0044] 本发明的另一方面提供一种在数据处理设备上处理数据的方法,该数据处理设备包括数据处理器和另一处理装置;所述数据处理器用于在安全模式和不安全模式下处理数据,在所述安全模式下处理数据的所述数据处理器可以访问安全数据,所述数据处理器在所述不安全模式下是不能访问所述安全数据的,所述数据处理器在所述安全模式下处理数据是在安全操作系统的控制下进行的,在所述不安全模式下处理数据是在不安全操作系统的控制下进行的;所述另一处理装置用来响应于来自所述数据处理器的请求而执行任务,所述任务包括处理至少其中一些是安全数据的数据;所述方法包括以下步骤:在所述数据处理器上处理数据;将来自所述数据处理器的请求发送到所述另一处理装置以执行任务,所述任务包括处理至少其中一些是安全数据的数据;在所述另一处理装置上开始所述任务;向所述另一处理装置发送暂停所述任务的信号;响应于接收到暂停所述任务的所述信号,所述另一处理装置暂停所述任务,并且启动使用安全密钥对所述安全数据的加密以及将所述加密的安全数据存储到不安全数据储存器中;从所述数据处理器生成指示所述另一处理装置继续所述任务的信号;响应于所述信号,所述另一处理装置从所述不安全数据储存器中检索所述加密的安全数据,并使用所述安全密钥解密所述加密的安全数据,所述安全密钥被安全地存储在执行所述加密和解密的所述装置内,使得其不能由其它装置访问。

[0045] 结合附图阅读下文的示例性实施例的详细描述,本发明的上述目标、特征和优点以及其它的目标、特征和优点将变得显而易见。

附图说明

[0046] 图 1 示意性地显示了根据本发明的实施例的数据处理设备;

[0047] 图 2 示意性地显示了根据本发明的实施例的数据处理设备,该数据处理设备具有处理器,加速器和分离的加解密装置;

[0048] 图 3 示意性地显示了根据本发明另一实施例的数据处理设备;

[0049] 图 4 显示了图解说明根据本发明的实施例的方法的流程图；

[0050] 图 5 显示了用来在中断时暂停和继续硬件辅助 (hardware assist) 的控制流程；

[0051] 图 6 显示了用于上下文切出 (switch out) 的流程图；

[0052] 图 7 显示了用于将进程上下文切换到硬件辅助模块的流程图；和

[0053] 图 8 显示了图解说明延迟硬件辅助状态保存直到第一新进程访问的流程图。

具体实施方式

[0054] 图 1 示意性地显示了处理器核心 10 和加速器 20。该系统配备有监控程序 12，监控程序 12 至少部分地在监控模式下执行。安全性状态标志只在监控模式下是可写访问的，并且可以被监控程序写入。监控程序负责管理安全域和不安全域之间的在任一方向上的所有变化。从该核心的外部看，监控模式总是安全的，并且监控程序处于安全存储器中。监控模式 12 可以被认为是安全处理模式，因为安全状态标志在这种模式下可以被改变，并且监控模式 12 下的监控程序有能力自身设置安全性状态标志，这总体上有效地提供了系统内的最终安全性等级。

[0055] 监控模式在该系统中具有最高等级的安全性访问，而且是有权在不安全域和安全域之间在任一方向切换该系统的唯一模式。因此，所有的域切换都是通过切换到监控模式并且在监控模式内执行监控程序进行的。

[0056] 在不安全域中提供了不安全操作系统 14，它运行多个不安全应用程序，所述不安全应用程序与不安全操作系统 14 协同执行。在安全域中，提供了安全内核程序。可以认为安全内核程序 16 形成安全操作系统。一般地，这样的安全内核程序 16 被设计成只提供对必须提供在安全域中的处理活动是必需的那些功能，使得安全内核 16 可以尽可能地小和简单，因为这往往使其更加安全。

[0057] 当在安全模式和不安全模式之间切换时，总是调用 (invoke) 监控模式以控制数据的安全性，并确保安全的数据永远不用于不安全侧。为了进入安全模式，首先要进入安全特权模式。除了响应于安全中断，从非特权安全模式不能进入监控模式。

[0058] 图 1 还显示了链接到核心 10 的加速器 20 它，并且在此实施例中，加速器 20 包括加解密引擎。加解密引擎 20 由核心 10 调用以为其执行加解密功能。可能即使在不安全模式下操作，仍希望对某些数据加密。在这种情况下，加解密引擎被调用以对数据加密，并且加解密引擎使用安全密钥执行加解密功能。显然，这些安全密钥不应为不安全侧可用，并且尽管加解密引擎可能是不可编程硬件装置，这使得其非常安全，但如果加解密过程出于某些原因被中断，那么如果来自加解密引擎的状态要被保存以便能够继续处理，就会存在安全性问题，因为如果当不安全操作系统正在控制核心时执行保存加解密过程的状态的操作，则保存加解密过程的状态可能会使其可用于不安全侧。然而，如果需要调用安全操作系统以便管理数据的安全性，则存在与中断该应用相关的很长的时延。在本实施例中，加解密引擎的功能性被用来使用存储在加解密引擎 20 中的会话密钥 22 对诸如在其处理过程中使用的密钥之类的安全数据进行加密。在此实施例中，会话密钥 22 被存储在加解密引擎 20 内，此时处理器核心在安全操作系统 14 的控制下在安全模式下操作，由此该核心指示加解密引擎从存储器 30 的的安全存储器部分加载会话密钥。会话密钥 22 存储在只写位置，使得试图攻击进入加解密引擎 20 的外部进程不能读取该会话密钥。

[0059] 因此，响应于来自核心 10 的中断，该中断表明加解密引擎正在处理的应用应该被暂停，加解密引擎 20 暂停其数据的处理，并使用其加解密功能性以利用会话密钥 22 对其之前正处理的安全数据进行加密。加解密引擎的状态连同被加密的安全数据和未被加密的不安全数据然后被存储到由指针 24 指示的位置处的存储器 30 的不安全部分中。

[0060] 当该核心希望继续它分派给加解密引擎 20 执行的应用时，它发送信号给加解密引擎 20 表明此意，由此加解密引擎 20 可以通过访问由指针 24 所指示的位置离开其状态和存储器 30 所存储的数据 (leave itsstate and the stored data from memory 30)。然后它可以使用会话密钥 22 解密该加密的数据，之后可以继续处理。这样，在核心 10 上的不安全操作系统的控制下运行的不安全应用可以给加解密引擎 10 分派任务以使用安全密钥执行加解密，并允许此应用被中断，而无需调用监控模式来控制安全数据的安全性状态。

[0061] 图 2 显示了可替代实施例，其中核心 10 给加速器 20 分派任务使其执行功能，在图 2 的情况下该功能不是加解密功能。在这种情况下，当核心 10 切换上下文时，例如从安全模式切换到不安全模式时，它分派给加速器 20 处理的安全数据需要被保存，使得通过该上下文切换而中断的任务可以在稍后继续。在这种情况下，加速器 20 不是加解密引擎，因此，它指示单独的加解密单元 50 来加密它正在处理的安全数据，加速器 20 通过专用的通信线路 52 链接到加解密单元 50。加解密单元 50 包括随机数生成器 52，加解密单元使用随机数生成器 52 以在复位时生成随机数。该随机数被存储为会话密钥 54，并用来加密加速器 20 发送给它的安全数据。在这个实施例中，加解密单元在加密数据之前还根据未加密的数据生成哈希值，该哈希值通过加解密单元使用会话密钥 54 连同加密的数据而被加密。在一些实施例中，可以使用伪随机数生成器，而不是具有真随机数生成器的开销。这获取一组现有信号，并根据这些信号的组合生成一数。信号的组合当然必须是秘密组合。

[0062] 该加密的数据连同加密的哈希值然后以与图 1 所示大致相同的方式与加速器 20 的状态以及不安全数据一起被存储在存储器中。

[0063] 尽管在这些实施例中，安全数据在存储之前被加密，但在一些实施例中，哈希可被产生并加密，并且未加密的数据连同加密的哈希一起被存储。这使得用户能够知道数据是否已经被篡改。在这种情况下，安全数据对于不安全侧是可访问的，但在系统不知道的情况下它是不能被更改的。在系统的安全性可能由于在另一处理装置中使用已经被篡改（但并不是通过查看该数据）的安全数据而受到威胁的情况下，这可能是有用的。

[0064] 当该核心切换回安全状态时，它给加速器 20 发信号指示其以前正在执行的任务应该继续。此时，从存储器 30 中检索存储的状态和数据，并且加密的数据和哈希值被发送到加解密单元 50。加密的数据然后连同哈希值一起被解密，并且根据解密的数据产生新的哈希值。如果这个新的哈希值与解密的哈希值匹配，则加解密单元 50 可以保证数据没有被篡改，然后解密的数据被发送到加速器 20。如果这些哈希值不匹配，则表明数据已经被篡改，给核心 10 发信号表明此结果，并中止该过程。

[0065] 图 3 显示了可替代实施例，其中加速器 20 再次不是加解密单元。在这种情况下，不是使用单独的加解密单元，加解密电路 25 被添加到加速器 20，因此，当核心 10 向加速器 20 表明它需要暂停某些安全数据的处理时，加解密电路 25 可以用来加密这些安全数据。在这个实施例中其还被用来根据未加密的数据生成哈希值，并加密该哈希值。在这种情况下，因为加解密电路 25 相对简单，哈希值和加解密使用相同的算法。用于该加解密的会话密钥

22 在制造时被永久地存储在加速器 20 中。

[0066] 在其它方面，该系统以与图 1 和图 2 所示的方式非常类似的方式进行工作。

[0067] 图 4 显示的流程图图解说明了根据本发明的实施例的方法的各步骤。在该方法中，执行任务的请求是在加速器处从主处理器核心接收的。加速器然后执行包括处理安全数据和不安全数据的该任务。在执行此操作时，它轮询 (poll) 中断，如果接收到一个中断，它暂停该任务的处理，并根据安全数据产生哈希值。然后它使用存储在加密器上的会话密钥加密安全数据和哈希值。应注意尽管在此实施例中，哈希值是根据未加密的安全数据产生的，但在其它实施例中，安全数据可以首先被加密，然后根据加密的数据产生哈希值。

[0068] 处理器的状态、加密的安全数据、哈希值和任何不安全数据然后被存储在存储器中，并且加速器执行由中断指示的另一任务。当这被完成并且接收到指示继续被中断的任务的信号时，从存储器中存储数据的地址检索数据以及处理器的状态，并且然后解密所加密的数据和哈希。然后根据解密的数据创建哈希值，并与存储的解密的哈希进行比较看它们是否匹配。如果匹配，则恢复处理器的状态，处理继续，如果不匹配，则中止任务并生成错误信号。

[0069] 上述技术与题为“Providing Secure Services to A Non-SecureApplication”的申请号为 12/003,857 以及题为“Controlling Cleaningof Data Values Within a Hardware Accelerator”的申请号为12/000,005的共同未决的美国专利申请中描述的技术有关。这两个共同未决申请的公开内容全部被并入本文。

[0070] 下面将公开与另一处理装置或ARM®数据处理设备中的硬件 (HW) 辅助一起使用的不同技术的其他细节。

[0071] 希望访问安全侧数据的用户模式 HW 辅助的一般运行模型是在 HW 辅助试图访问安全侧之前，安全会话已经由安全侧或由 Trustzone(TZ) 软件建立。

[0072] 存在四种管理需要从安全侧访问数据的 HW 辅助的安全会话的潜在方式。

[0073] 到安全侧的默认软件 (SW) 入口在 SW 中有所有安全性。每次要求安全服务时，进入特权模式，然后运行 SMC(调用安全侧) 以进入安全监视。然后提供安全服务，程序然后返回到用户模式应用。

[0074] 使安全监视中的入口直接试图访问 HW 辅助。在此模式下，每次调用 HW 辅助时，进入安全侧，并且安全会话可由安全侧直接管理。安全侧 SW 还可以运行所需的任何完整性检查，然后调用 HW 来执行该任务。这可以通过禁用对 HW 辅助的不安全访问来实现。

[0075] 建立全局安全会话，并给要求安全访问的每一进程提供可用于不安全侧的安全值的表格中的预定义索引。每个表格值与哈希值成对，以确保只有具有正确哈希的进程才可以访问特定的表格条目。安全会话是在进程创建时建立的，此时可用的表格条目和保护哈希值被计算。安全侧还将安全基表的地址和有效表格范围写入到 HW 辅助中的仅安全模式寄存器中 (secure mode only register)。为了从应用程序访问安全项，HW 辅助被用户编程有偏移和哈希值。HW 辅助模块然后使用这些值来访问安全信息，并检查哈希。在此模型中，安全会话仍然是全局视图，因此不需要在规则的操作系统上下文切换时调用安全监视 - 该偏移和哈希可以在上下文切换时被存储在进程应用空间中。

[0076] 每个进程建立一安全会话。在此模型中，每个进程实际上具有自己的安全会话。在进程被上下文切入时建立安全会话，并且在上下文结束时关闭该会话。并不是在每次访问

HW 辅助时，都要访问安全监视，只是在上下文开始和结束时访问安全监视。此模型的优点是存储在 HW 辅助中的任何安全状态也可以在上下文切换时由安全监视直接地或通过具有 CA_STATUS 寄存器的安全影子 (secure shadow) 而被保存。CA_STATUS 寄存器是为 HW 辅助模块提供状态，并具有特权访问的寄存器。

[0077] 在任务完成之前，HW 辅助可以在运行时被中断，然后需要时在稍后时间被重新启动。如果 HW 辅助正在运行时接收到中断，则以通常的ARM®方式进入中断，而不对 HW 辅助进行任何改变。在中断处理程序 (handler) 中，程序员具有以下选项：

[0078] 1. 不对 HW 辅助进行任何处理，让其继续运行。注意该核心的系统编程必须保证 HW 辅助仍能看到存储器的正确视图（即，页表没有被改变）。

[0079] 2. 暂时暂停 HW 辅助，但不使用 HW 辅助用于其它任何处理。(MSUSPEND)

[0080] 3. 暂停 HW 辅助，并将任何脏 (dirty) 状态保存回存储器中，这样 HW 辅助可以被其它方使用。(MSUSPENDC)

[0081] 在 (2) 和 (3) 的情况下，一旦中断处理程序完成，用继续命令重新启动 HW 辅助。对于 (2)，执行从它停止的地方重新开始，而在 (3) 中，在继续执行之前，中间状态必须首先被重新加载回 HW 辅助。图 5 显示了这两种情况的控制流程。在该图中，CA_STATE 寄存器是包含指向 HW 辅助模块的描述符的指针的寄存器，具有特权访问或用户访问。

[0082] 对于一般的中断处理程序，广播形式的 MSUSPEND 和 MRESUME 可以被用来暂停和继续所有 HW 辅助。对于更为专门的应用，各个 HW 辅助模块可以（通过将 HW 辅助的逻辑地址提供给控制）被独立地暂停和继续。

[0083] 存在需要被处理的三种基本类型的异常：

[0084] ●用户错误 --- 由用户模式 SW 处理

[0085] ●特权系统错误 --- 发信号通知操作系统以便处理的系统错误

[0086] ●安全错误 --- 由访问安全侧引起的错误，被发信号通知给安全监视。

[0087] 在默认情况下，特权错误和安全性错误经由中断被发信号通知核心以便进一步处理。期望用户模式错误由用户应用软件校检查 HW 辅助寄存器中的状态错误位（如 CA_Status 寄存器的 IUE 位）来处理。这种架构也支持在需要时允许用户模式错误给核心发信号，使核心中断，但这会招致处理操作系统的代价。

[0088] 在来自不同特权级的多个异常可能有效 (active) 的情况下，提供影子控制寄存器来存储异常信息。例如，如果发生安全故障和用户存储器故障，则 FAR 和 CA_STATUS 寄存器必须在安全侧有影子。FAR 寄存器具有特权访问，并提供出故障的存储器地址。

[0089] 支持上下文切换所需的基本操作类似于基本的中断处理，即，暂停和继续 HW 辅助操作以及从 HW 辅助模块中清除脏状态。上下文切换可以被分解成两部分：

[0090] ●切出旧进程

[0091] ●切入新进程

[0092] 对于每种情况，都有两种可能性：

[0093] ●严格 (strict)--- 立即进行切换

[0094] ●惰性 (lazy)--- 如果其它人确实想要使用 HW 辅助，则只保存状态。

[0095] 后者功率较低，因为其只在必要时保存状态，但实现起来更为复杂。

[0096] 注意：传统的惰性上下文切换（如用在 VFP 中的）并不切入新的上下文，直到 HW

被新的进程访问。在这种技术中，如果新的上下文具有以前被暂停的 HW 辅助，则只要上下文被恢复，其就需要重新启动，而不等到在新的上下文中对 HW 的首次新访问。

[0097] 图 6 显示了为 HW 辅助模块 (CHA) 切换上下文的一般流程。如果上下文切换不是惰性的，则通过使用 MSUSPENDC 和到描述符的指针暂停并清除 HW 辅助，每个启用的 HW 辅助的状态寄存器和 FAR 被保存。在惰性上下文切换的情况下，仅暂停 HW 辅助，而不保存任何状态。在两种情况下，操作系统都可以继续进行其它上下文切换清除，同时等待 HW 辅助停止（以及潜在清除）。在页表条目需要被更改之前，操作系统执行数据加速器屏障 (data accelerator barrier，DAN) 以确保所有的 HW 辅助都已经完成暂停，然后在 HACR 中禁用所有的 HW 辅助。上下文切出然后象通常一样继续。

[0098] 可以细化一般的上下文切出，代价是操作系统进行更多的分析。例如：

[0099] 如果 HW 辅助模块当前没有被启用，则没有进程上下文需保存，也不需要进行任何操作。

[0100] 如果 HW 辅助被启用，但没有运行 --- 没有脏状态潜在地需要保存。

[0101] 可以选择允许 HW 辅助将其完成而不是暂停 --- 确保了所有的脏数据回到存储器中。

[0102] 在将新的进程切换到 HW 辅助模块中时，第一步是启用 HW 辅助模块。在这之后，描述符和状态寄存器被重新装载到 HW 辅助模块中，并且继续所有命令被发出以继续所有被启用的 HW 辅助模块的执行。这显示在图 7 中。

[0103] 如果采用的是惰性方案（即旧进程的状态还没有被切出），则操作系统应当确定新的进程是否已经暂停了 HW 辅助模块。如果新的上下文正在使用 HW 辅助，则旧的状态应该在上下文切换时从 HW 辅助中清除出去（而不是延迟到其之后被访问）。

[0104] 延迟的上下文切入

[0105] 延迟的进程状态切换在保存任何状态之前等待，直到新的进程尝试访问 HW 辅助模块。如果新进程不使用 HW 辅助模块，则没有状态需要被保存。

[0106] 为了检测新进程何时尝试访问 HW 辅助模块，在上下文切换时禁用该模块，使得尝试使用该模块会触发操作系统中的事件，操作系统可以保存并恢复所需状态（未定义的指令陷阱）。一旦旧状态已经被保存并且新状态被加载，访问 HW 辅助的命令可以被重新运行。该程序显示在图 8 中。假设暂停的 HW 辅助是在上下文切入时被检测的，并被自动重新启动，而不是等待访问它们的命令。

[0107] 注意操作系统必须能够访问以前的 HW 辅助上下文的用户空间描述符以将 HW 辅助状态保存到正确位置。

[0108] 尽管本文已经参考附图详细描述了本发明的示例性实施例，但应理解本发明并不局限于这些确切的实施例，在不偏离由所附权利要求书限定的本发明的范围和精神的情况下，本领域技术人员可以实施各种改变和改进。

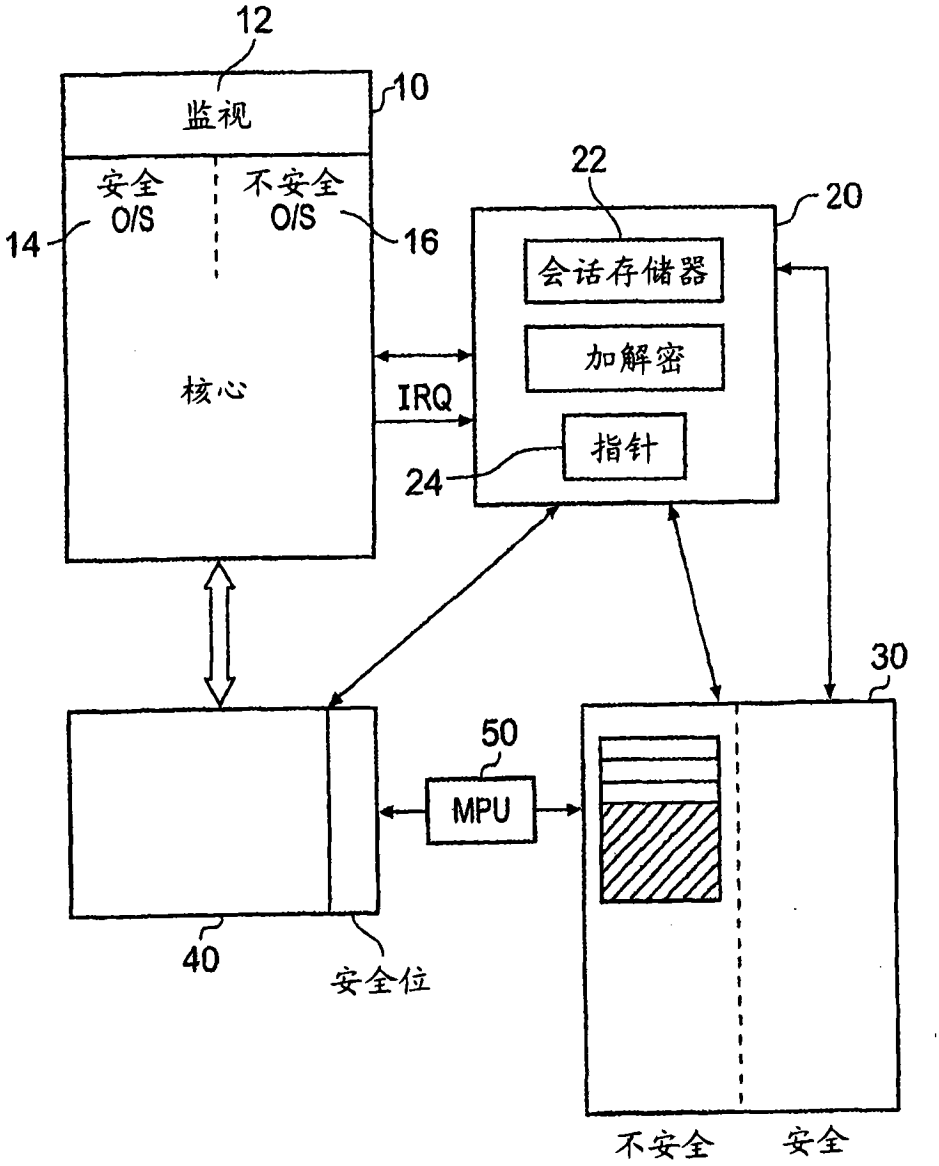

12
监视
10
安全
O/S
不安全
O/S
14
16
22
20
会话存储器
加解密
核心
IRQ
指针
24
30
50
MPU
40
安全位
不安全
安全

图 1

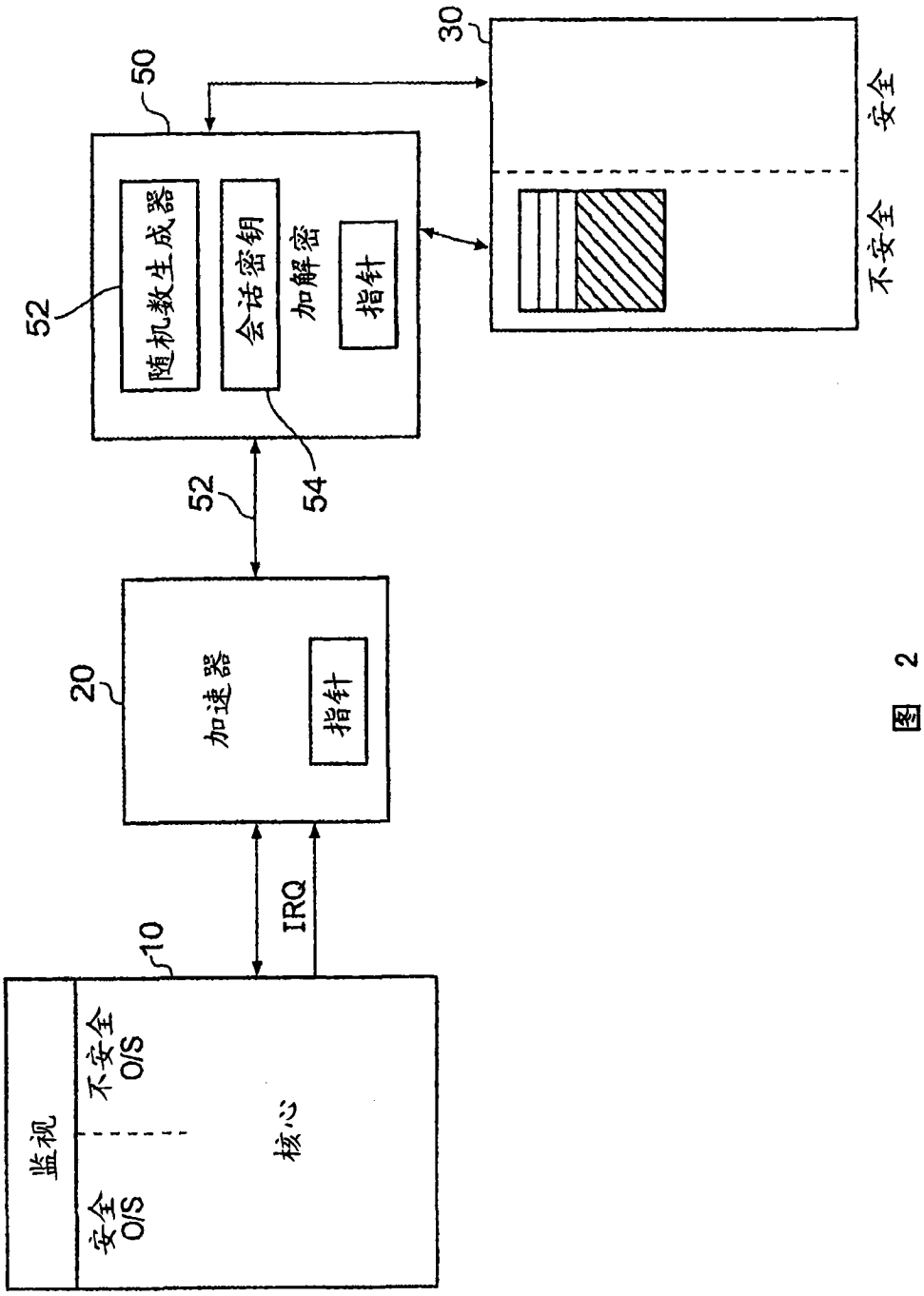

30
50
安全
不安全
52
随机数生成器
会话密钥
加解密
指针
54
52
20
加速器
指针
IRQ
10
监视
不安全 O/S
安全 O/S
核心

图 2

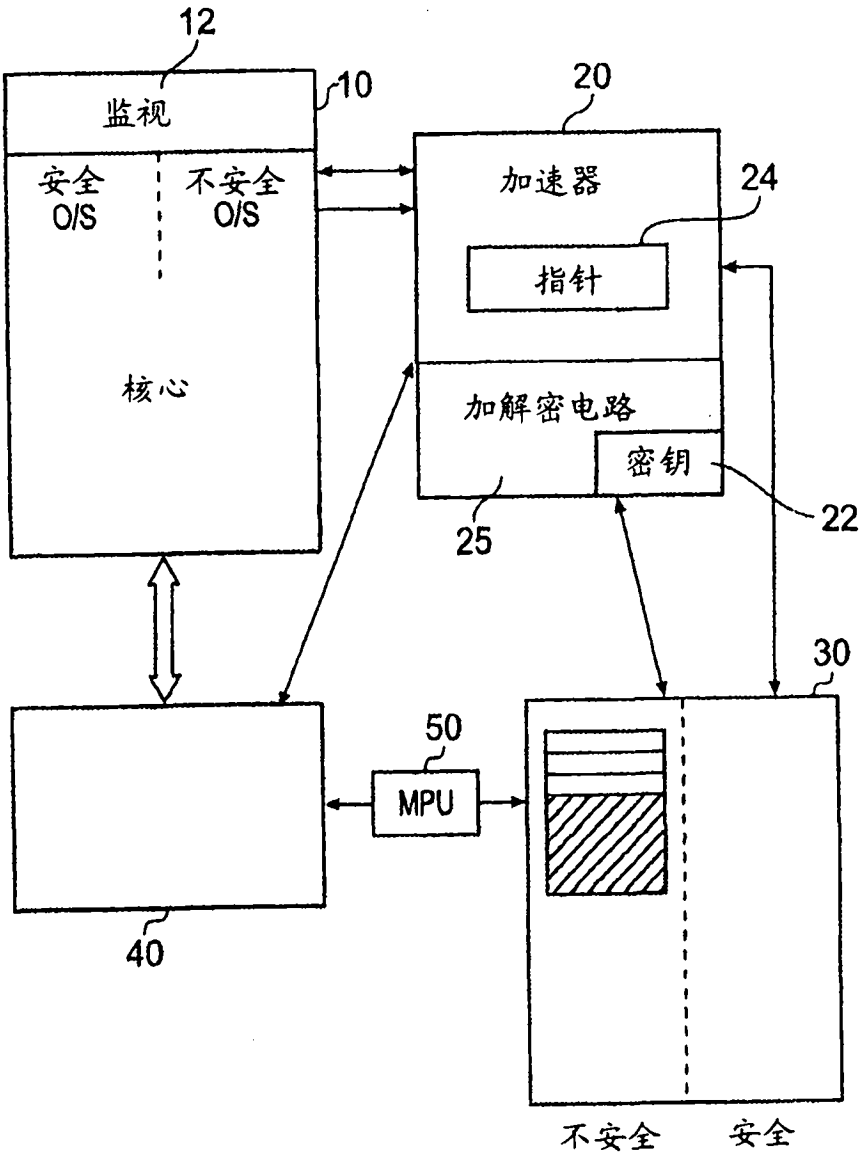

12
监视
10
安全
O/S
不安全
O/S
核心
20
加速器
24
指针
加解密电路
密钥
22
25
30
50
MPU
40
不安全
安全

图 3

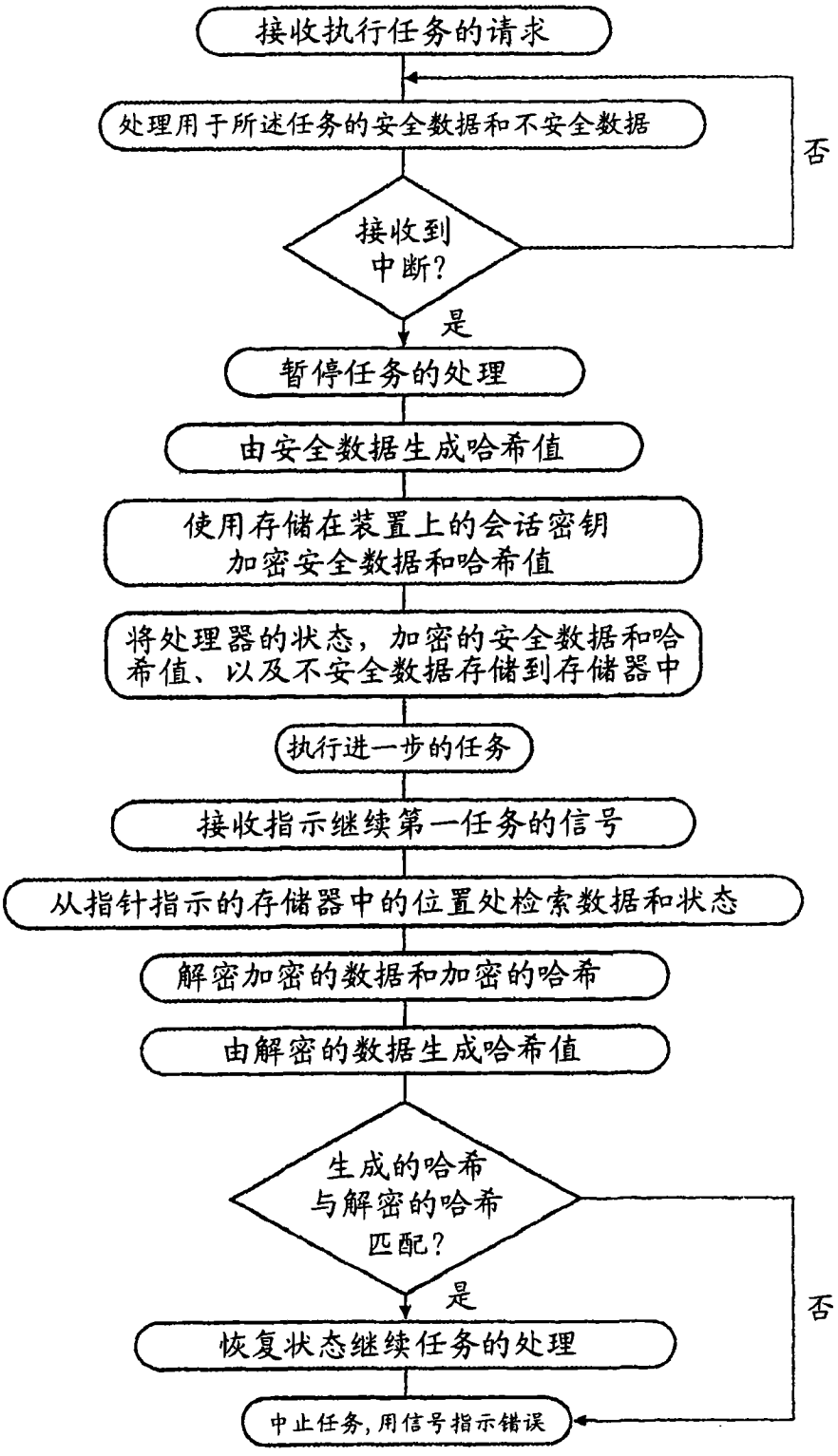

接收执行任务的请求
处理用于所述任务的安全数据和不安全数据
否
接收到中断?
是
暂停任务的处理
由安全数据生成哈希值
使用存储在装置上的会话密钥加密安全数据和哈希值
将处理器的状态，加密的安全数据和哈希值、以及不安全数据存储到存储器中
执行进一步的任务
接收指示继续第一任务的信号
从指针指示的存储器中的位置处检索数据和状态
解密加密的数据和加密的哈希
由解密的数据生成哈希值
生成的哈希与解密的哈希匹配?
否
是
恢复状态继续任务的处理
中止任务，用信号指示错误

图 4

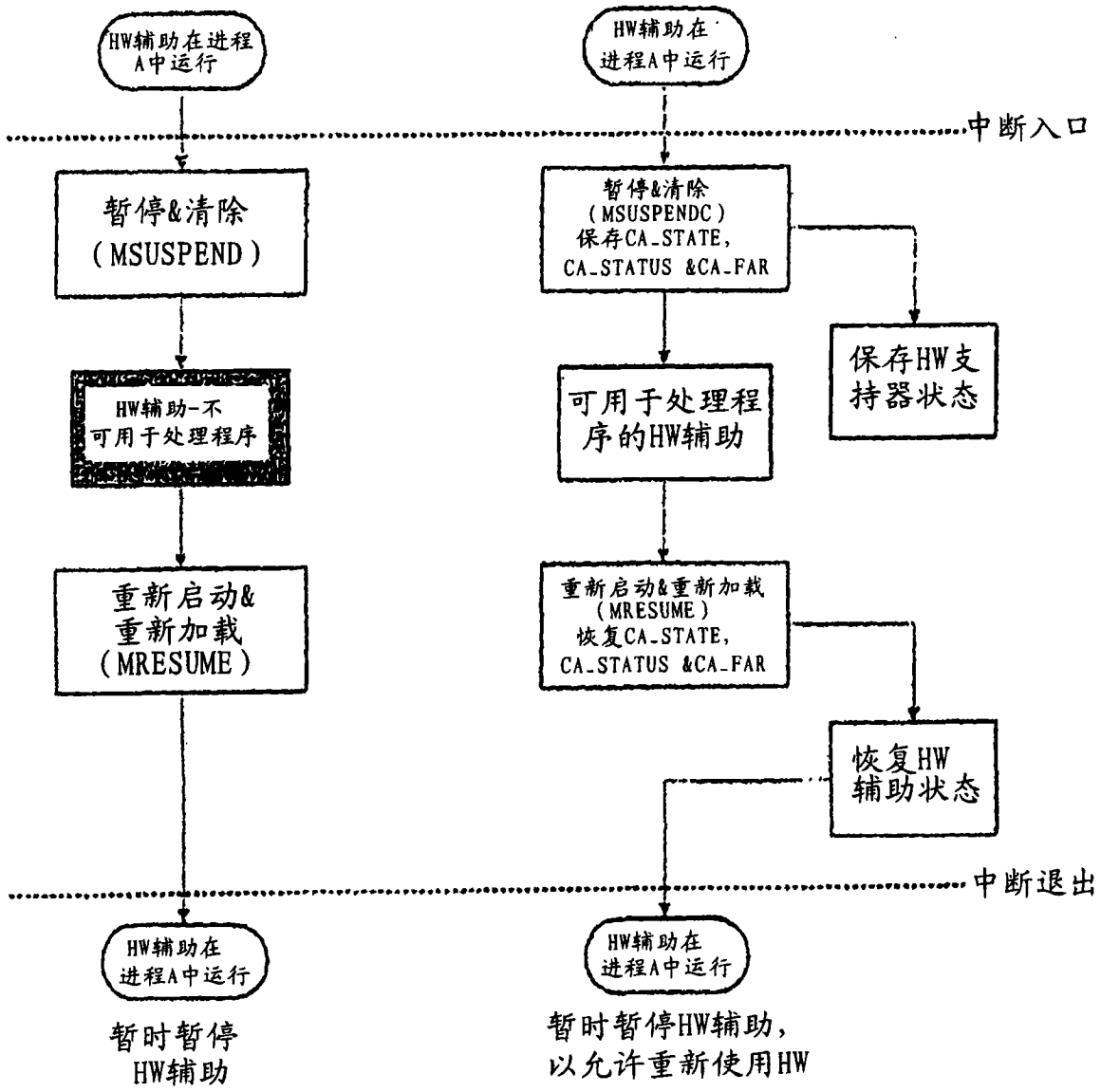

HW辅助在进程A中运行
HW辅助在进程A中运行
中断入口
暂停&清除（MSUSPEND）
暂停&清除（MSUSPENDC）保存CA_STATE，CA_STATUS &CA_FAR
保存HW支持器状态
HW辅助-不可用于处理程序
可用于处理程序的HW辅助
重新启动&重新加载（MRESUME）
重新启动&重新加载（MRESUME）恢复CA_STATE，CA_STATUS &CA_FAR
恢复HW辅助状态
中断退出
HW辅助在进程A中运行
HW辅助在进程A中运行
暂时暂停HW辅助
暂时暂停HW辅助，以允许重新使用HW

图 5

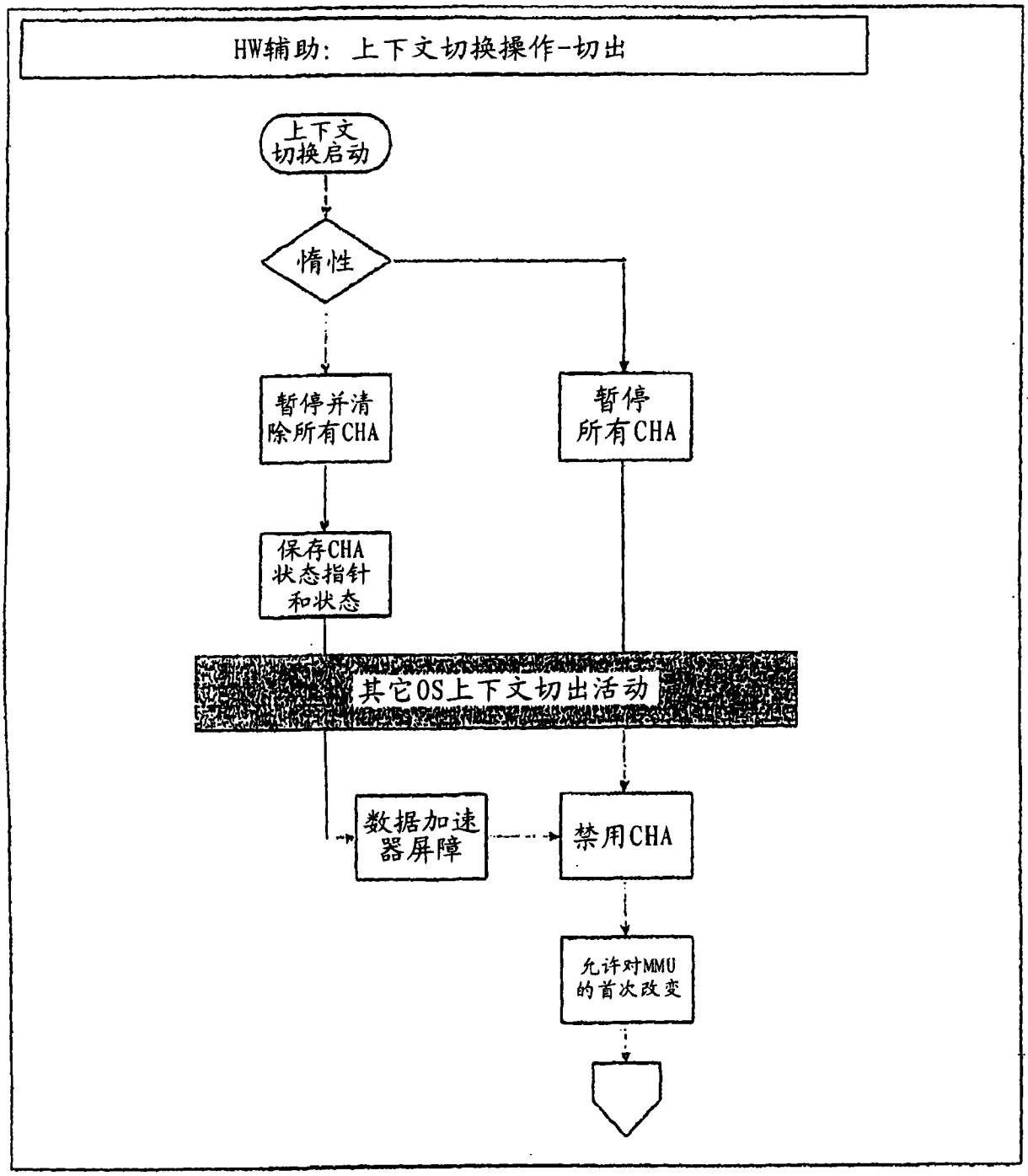

HW辅助：上下文切换操作-切出
上下文切换启动
惰性
暂停并清除所有CHA
暂停所有CHA
保存CHA状态指针和状态
其它OS上下文切出活动
数据加速器屏障
禁用CHA
允许对MMU的首次改变

图 6

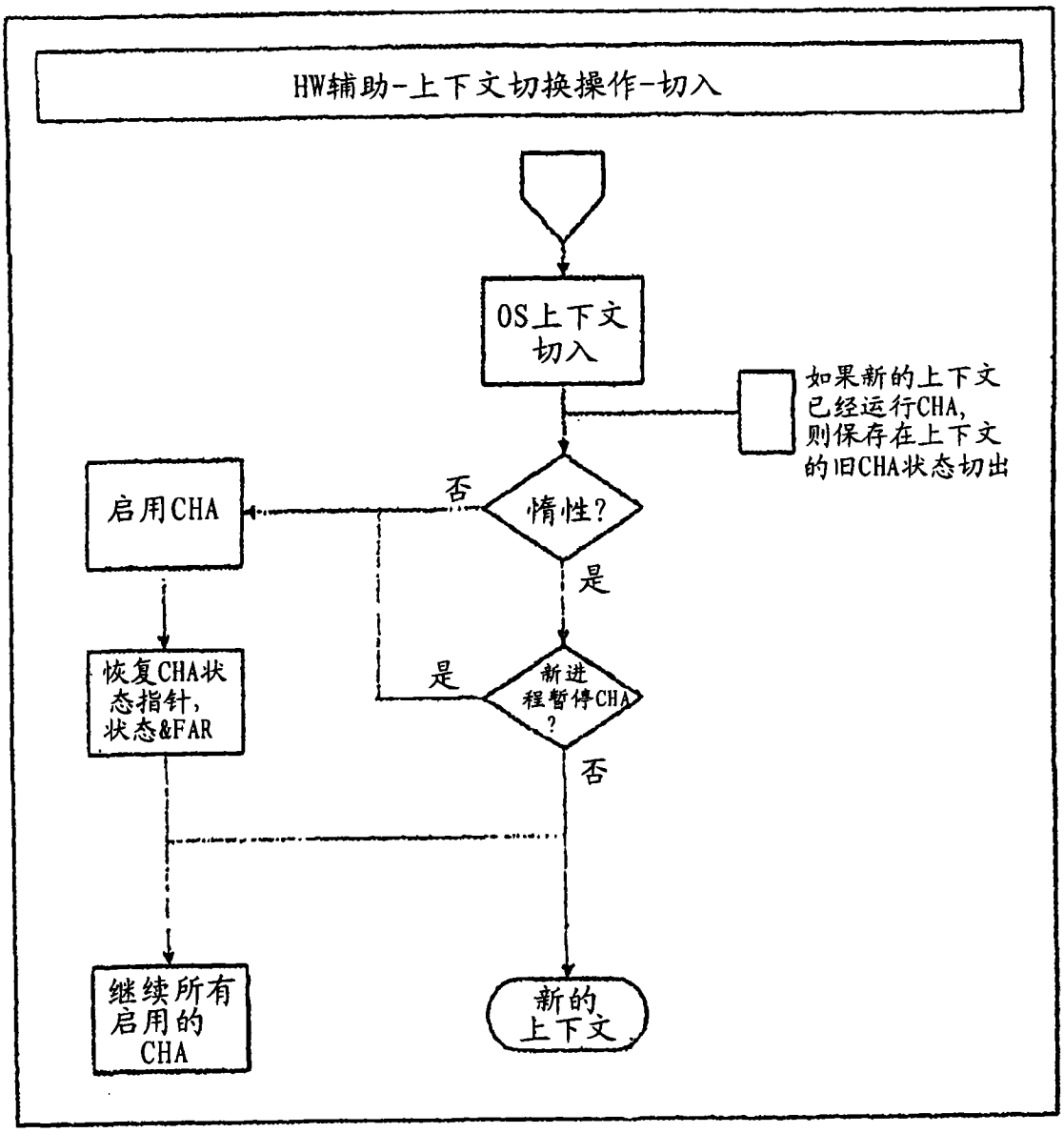

HW辅助-上下文切换操作-切入
OS上下文切入
如果新的上下文已经运行CHA，则保存在上下文的旧CHA状态切出
惰性?
否
是
启用CHA
恢复CHA状态指针，状态&FAR
新进程暂停CHA?
是
否
继续所有启用的CHA
新的上下文

图 7

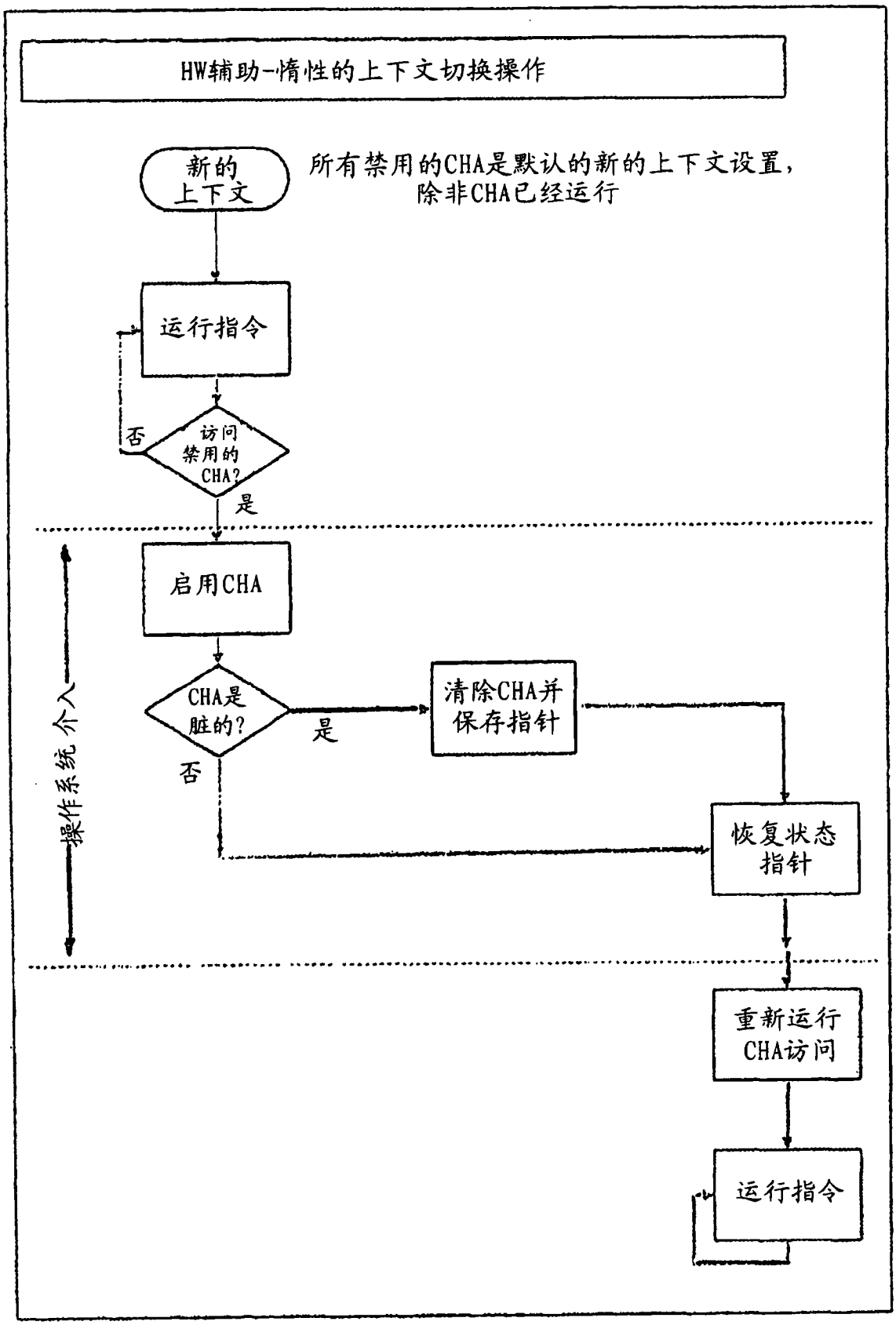

HW辅助-惰性的上下文切换操作
新的上下文
所有禁用的CHA是默认的新的上下文设置，除非CHA已经运行
运行指令
否
访问禁用的CHA?
是
启用CHA
CHA是脏的?
是
清除CHA并保存指针
否
恢复状态指针
操作系统介入
重新运行CHA访问
运行指令

图 8