



[12] 发明专利申请公开说明书

[21] 申请号 02822219.9

[43] 公开日 2005 年 10 月 19 日

[11] 公开号 CN 1685297A

[22] 申请日 2002.10.30 [21] 申请号 02822219.9
 [30] 优先权
 [32] 2001.11.1 [33] US [31] 10/043,843
 [86] 国际申请 PCT/US2002/034487 2002.10.30
 [87] 国际公布 WO2003/038574 英 2003.5.8
 [85] 进入国家阶段日期 2004.5.8
 [71] 申请人 英特尔公司
 地址 美国加利福尼亚州
 [72] 发明人 劳伦斯·史密斯 三世
 詹姆斯·萨顿 二世
 戴维·波伊斯尼尔 克利福德·霍尔
 安德鲁·格洛伊 吉尔伯特·奈格
 理查德·乌利希 迈克尔·科祖克
 罗伯特·乔治 戴维·克劳罗克
 布拉德利·伯吉斯

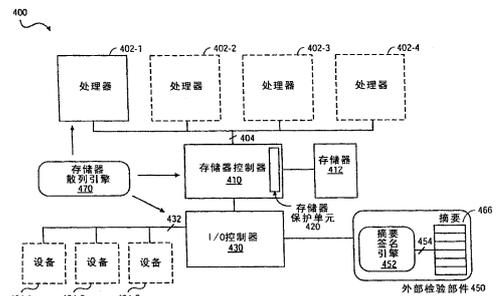
[74] 专利代理机构 北京东方亿思知识产权代理有限
 责任公司
 代理人 王 怡

权利要求书 9 页 说明书 17 页 附图 10 页

[54] 发明名称 在多处理器环境中单方面加载安全操作系统的装置和方法

[57] 摘要

本公开描述了一种用于在多处理器环境中单方面加载安全操作系统的装置和方法。该方法包括当检测到当前活动的加载安全区域操作时，不理睬收到的加载安全区域指令。否则，响应于收到的加载安全区域指令，存储器保护单元被引导以形成安全存储器环境。一旦被引导，则对一个或者多个受保护的存储器区域的未授权的读/写访问被禁止。最后，一个或者多个受保护的存储器区域的加密散列值作为安全软件标识值被存储到摘要信息库中。一旦被存储，则外部代理可以请求对被数字签名的软件标识值进行访问，以便建立对在安全存储器环境中的安全软件的安全检验。



1. 一种方法，包括：

当检测到当前活动的加载操作时，不理睬收到的加载指令；

- 5 否则的话，响应于所述收到的加载指令，引导存储器保护单元以形成包括了一个或者多个受保护的存储器区域的安全存储器环境，以使得对所述一个或者多个受保护的存储器区域的未授权的读/写访问被禁止；以及
- 在摘要信息库中存储所述一个或者多个受保护的存储器区域的加密散列值，从而使得能够建立所述安全存储器环境的安全检验。

- 10 2. 根据权利要求 1 所述的方法，其中，所述不理睬收到的加载指令的步骤还包括：

从安全启动软件接收作为所述加载指令的加载安全区域指令，以使得所述加载安全区域指令指示要被建立为所述安全存储器环境的所述一个或者多个存储器区域；

- 15 确认所述安全启动软件具有足够发出所述加载安全区域指令的特权等级；

一旦被确认，则确定是否存在当前活动的加载安全区域操作排他性互斥体；

- 20 当检测到当前活动的加载安全区域操作排他性互斥体时，不理睬所述收到的加载安全区域指令请求；以及

否则的话，建立加载安全区域指令排他性互斥体。

3. 根据权利要求 1 所述的方法，其中，所述不理睬收到的加载指令的步骤还包括：

确定是否已经收到安全复位命令；以及

- 25 当已经收到安全复位命令时，不理睬所述收到的加载指令请求。

4. 根据权利要求 3 所述的方法，其中，所述确定是否已经收到安全复位命令的步骤还包括：

查询所述摘要信息库中的值；以及

当所述摘要信息库中的值被提交的时候，检测复位命令的接收。

5. 根据权利要求 1 所述的方法，其中，所述引导存储器保护单元的步骤还包括：

从所述收到的加载指令请求确定所述一个或者多个受保护的存储器区域；以及

- 5 向存储器保护单元提供请求，该请求包括所述一个或者多个受保护的存储器区域的参数，以使得所述存储器保护单元根据所述加载指令，建立由加载安全区域操作所请求的所述安全存储器环境。

6. 根据权利要求 1 所述的方法，其中，所述引导存储器保护单元的步骤还包括：

- 10 清除所述摘要信息库；以及

将安全环境信息记录在所述摘要信息库中。

7. 根据权利要求 1 所述的方法，其中，所述存储所述一个或者多个受保护的存储器区域的加密散列值的步骤还包括：

- 15 散列所述安全存储器区域的内容，以为包含在所述安全环境中的软件生成作为安全软件标识值的加密散列标识值；以及

将所述安全软件标识值存储在所述摘要信息库中；以及
释放加载操作排他性互斥体。

8. 根据权利要求 1 所述的方法，还包括：

从外部代理接收安全验证请求；

- 20 通过安全通道访问安全软件标识值；

对所述软件标识值进行数字签名；以及

将所述被数字签名的安全软件标识值传送给所述外部代理，以使得所述外部代理可以检查在所述安全存储器环境中的安全软件的部件状态，并决定是否信任所述安全软件。

- 25 9. 根据权利要求 1 所述的方法，还包括：

向系统芯片组发出处理器复位请求；以及

响应于所述处理器复位请求，由所述芯片组复位耦合到所述芯片组的所有处理器。

10. 一种方法，包括：

响应于安全复位请求，复位系统中的一个或者多个处理器，所述安全复位请求在加载安全区域操作完成之后被发出；以及

使能对安全存储器环境的一个或者多个受保护的存储器区域的处理器读/写访问，所述安全存储器环境是按照用于建立安全环境的加载安全区域指令请求而形成的。

11. 根据权利要求 10 所述的方法，其中，所述复位系统中的一个或者多个处理器的步骤还包括：

向系统芯片组平台发出处理器复位命令；以及

由所述芯片组平台向所述一个或者多个处理器的每一个发出复位请求。

12. 根据权利要求 11 所述的方法，其中，所述发出处理器复位命令的步骤还包括：

进行对输入/输出端口地址的数据写入；以及

由所述芯片组平台译码所述数据写入，以检测处理器复位命令。

13. 根据权利要求 10 所述的方法，其中，所述重新使能处理器读/写访问的步骤还包括：

提交摘要信息库中的数据，以指示当前活动的复位命令；

使得耦合到所述系统的每个处理器进入已知的特权状态；以及

引导存储器保护单元重新使能对所述安全存储器环境的处理器读/写访问。

14. 根据权利要求 13 所述的方法，其中，所述使得耦合到所述系统的每个处理器进入已知的特权状态的步骤包括：

将所述一个或者多个处理器中的数据存储单元设置为常量或者由参数导出的值中的一个，所述常量是被存储在所述处理器中或者由所述处理器生成的，所述参数是在发出所述加载安全区域指令时由安全启动软件所规定的。

15. 一种计算机可读存储介质，包括程序指令，所述程序指令当被处理器执行的时候，引导计算机以规定方式运行，所述程序指令包括：

当检测到当前活动的加载指令时，不理睬收到的加载指令；

否则的话，响应于所述收到的加载指令，引导存储器保护单元以形成包括了一个或者多个受保护的存储器区域的安全存储器环境，以使得对所述一个或者多个受保护的存储器区域的未授权的读/写访问被禁止；以及

5 在摘要信息库中存储所述一个或者多个受保护的存储器区域的加密散列值，从而使得能够建立所述安全环境的安全检验。

16. 根据权利要求 15 所述的计算机可读存储介质，其中，所述不理睬收到的加载指令的步骤还包括：

10 从安全启动软件接收作为所述加载安全区域指令的指令，以使得所述加载安全区域指令指示要被建立为所述安全存储器环境的所述一个或者多个存储器区域，以及被加载在所述安全存储器区域中的安全软件；

确认所述安全启动软件具有足够发出所述加载安全区域指令的特权等级；

一旦被确认，则确定是否存在当前活动的加载操作排他性互斥体；

15 当检测到当前活动的加载操作排他性互斥体时，不理睬所述收到的加载安全区域指令请求；以及

否则的话，建立加载安全区域操作排他性互斥体。

17. 根据权利要求 15 所述的计算机可读存储介质，其中，所述不理睬收到的加载指令的步骤还包括：

确定是否已经收到安全复位命令；以及

20 当已经收到安全复位命令时，不理睬所述收到的加载指令请求。

18. 根据权利要求 17 所述的计算机可读存储介质，其中，所述确定是否已经收到安全复位命令的步骤还包括：

查询所述摘要信息库中的值；以及

当所述摘要信息库中的值被提交的时候，检测复位命令的接收。

25 19. 根据权利要求 15 所述的计算机可读存储介质，其中，所述引导存储器保护单元的步骤还包括：

从所述收到的加载指令请求确定所述一个或者多个受保护的存储器区域；以及

向存储器保护单元提供请求，该请求包括所述一个或者多个受保护的

存储器区域的参数，以使得所述存储器保护单元根据所述加载指令，建立由加载安全区域操作所请求的所述安全存储器区域。

20. 根据权利要求 15 所述的计算机可读存储介质，其中，所述程序还包括：

- 5 清除所述摘要信息库；以及
 将安全环境信息记录在所述摘要信息库中。

21. 根据权利要求 15 所述的计算机可读存储介质，其中，所述存储所述一个或者多个受保护的存储器区域的加密散列值的步骤还包括：

- 10 散列所述安全存储器环境中的内容，以为被加载在所述安全存储器环境中的软件生成作为安全软件标识值的加密散列标识值；
 将所述安全软件标识值存储在所述摘要信息库中；以及
 释放加载操作排他性互斥体。

22. 根据权利要求 15 所述的计算机可读存储介质，还包括：

- 15 从外部代理接收安全验证请求；
 通过安全通道访问软件标识值；
 对所述软件散列标识值进行数字签名；以及
 将所述被数字签名的软件标识值传送给所述外部代理，以使得所述外部代理可以检查在所述安全存储器环境中的安全软件的部件状态，并决定是否信任所述安全软件。

20 23. 根据权利要求 15 所述的计算机可读存储介质，还包括：

- 向系统芯片组发出处理器复位请求；以及
 响应于所述处理器复位请求，由所述芯片组复位耦合到所述芯片组的所有。

24. 一种计算机可读存储介质，包括：

- 25 响应于安全复位请求，复位系统中的一个或者多个处理器，所述安全复位请求在加载安全区域操作完成之后被发出；以及
 重新使能对安全存储器环境的一个或者多个受保护的存储器区域的处理器读/写访问，所述安全存储器环境是按照用于建立安全环境的加载安全区域指令而形成的。

25. 根据权利要求 24 所述的计算机可读存储介质，其中，所述复位系统中的一个或者多个处理器的步骤还包括：

向系统芯片组平台发出处理器复位命令；以及

5 由所述芯片组平台向所述一个或者多个处理器的每一个发出复位请求。

26. 根据权利要求 25 所述的计算机可读存储介质，其中，所述发出处理器复位命令的步骤还包括：

进行对输入/输出端口地址的数据写入；以及

由所述芯片组平台译码所述数据写入，以检测处理器复位命令。

10 27. 根据权利要求 24 所述的计算机可读存储介质，其中，所述重新使能处理器读/写访问的步骤还包括：

提交摘要信息库中的数据，以指示当前活动的复位命令；

使得耦合到所述系统的每个处理器进入已知的特权状态；以及

15 编程存储器保护单元，以重新使能对所述安全存储器环境的处理器读/写。

28. 根据权利要求 27 所述的计算机可读存储介质，其中，所述处理器中的每个还包括：

将所述处理器中的数据存储单元设置为常量或者由参数导出的值中的一个，所述常量是被存储在所述处理器中或者由所述处理器生成的，所述
20 参数是在发出所述加载安全区域指令时由安全启动软件所规定的。

29. 一种装置，包括：

存储器控制器；

多个处理器，所述多个处理器各自耦合到所述存储器控制器，并各自具有执行指令的电路；

25 耦合到所述存储器控制器的存储器保护单元，所述存储器保护单元阻止对一个或者多个受保护的存储器区域的未授权的存储器访问；

耦合到所述存储器控制器的摘要信息库，用于存储安全软件标识值；

和

耦合到所述存储器控制器的存储器设备，所述存储器设备具有存储于

其中的指令序列，所述指令序列包括至少一个加载指令，所述加载指令当被处理器执行时使得所述处理器：

当检测到当前活动的加载安全区域操作时，不理睬收到的加载安全区域指令；

5 否则的话，按照对应于所述收到的加载安全区域指令的加载安全区域操作，引导存储器保护单元以形成包括了一个或者多个受保护的存储器区域的安全存储器环境，以使得对所述一个或者多个受保护的存储器区域的未授权的读/写访问被禁止；以及

10 根据所述加载安全区域操作，在所述摘要信息库中存储所述一个或者多个受保护的存储器区域的加密散列值，作为所述安全软件标识值，从而使得能够建立由外部代理进行的所述安全存储器环境的安全检验。

30. 根据权利要求 29 所述的装置，其中，所述处理器还被使得：
从外部代理接收安全验证请求；

15 通过安全通道访问安全软件标识值；

对所述安全软件标识值进行数字签名；以及

将所述被数字签名的安全软件标识值传送给所述外部代理，以使得所述外部代理可以检查在所述安全存储器环境中的安全软件的部件状态，并决定是否信任所述安全软件。

20 31. 根据权利要求 29 所述的装置，其中，所述处理器还被使得：

响应于安全复位请求，复位所述多个处理器中的每一个，所述安全复位请求在所述加载指令完成之后被发出；以及

25 重新使能对所述安全存储器环境的所述一个或者多个受保护的存储器区域的处理器读/写访问，所述安全存储器环境是按照用于建立安全环境的所述加载安全区域操作而形成的。

32. 根据权利要求 29 所述的装置，其中，所述用于重新使能的指令还使得所述处理器：

提交摘要信息库中的数据，以指示当前活动的安全复位命令；
使得耦合到所述系统的每个处理器进入已知的特权状态；以及

引导存储器保护单元重新使能对所述安全存储器环境的处理器读/写访问。

33. 根据权利要求 32 所述的装置，其中，所述用于使得处理器进入特权状态的指令还使得所述处理器：

- 5 将所述多个处理器中的数据单元设置为常量或者由参数导出的值中的一个，所述常量是被存储在所述处理器中或者由所述处理器生成的，所述参数是在发出所述加载安全区域指令时由安全启动软件所规定的。

34. 一种系统，包括：

存储器控制器；

- 10 多个处理器，所述多个处理器各自耦合到所述存储器控制器，并各自具有执行指令的电路；

耦合到所述存储器控制器的存储器保护单元，所述存储器保护单元阻止对一个或者多个受保护的存储器区域的未授权的存储器访问；

耦合到所述存储器控制器的输入/输出控制器；

- 15 检验单元，所述检验单元耦合到所述输入/输出控制器，并包括用于存储安全软件标识值以及向外部代理提供安全验证的摘要信息库；和

耦合到所述存储器控制器的存储设备，所述存储设备具有存储于其中的指令序列，所述指令序列包括至少一个加载指令，所述加载指令当被处理器执行时使得所述处理器：

- 20 当检测到当前活动的加载指令时，不理睬收到的加载安全区域操作；

否则的话，按照对应于所述收到的加载指令的加载安全区域操作，引导存储器保护单元以形成包括了一个或者多个受保护的存储器区域的安全存储器环境，以使得对所述一个或者多个受保护的存储器区域的未授权的读/写访问被禁止；以及

- 25 在所述摘要信息库中存储所述一个或者多个受保护的存储器区域的加密散列值，从而使得能够建立由外部代理进行的所述安全存储器环境的安全检验。

35. 根据权利要求 34 所述的系统，其中，所述处理器还被使得：

从外部代理接收安全验证请求；
通过安全通道访问安全软件标识值；
对所述安全软件标识值进行数字签名；以及

5 将所述被数字签名的安全软件标识值传送给所述外部代理，以使得所述外部代理可以检查在所述安全存储器环境中的安全软件的部件状态，并决定是否信任所述安全软件。

36. 根据权利要求 34 所述的系统，其中，所述处理器还被使得：响应于安全复位请求，复位所述多个处理器中的每一个，所述安全复位请求在所述加载指令请求完成之后被发出；以及

10 重新使能对安全存储器环境的一个或者多个受保护的存储器区域的处理器读/写访问，所述安全存储器环境是按照用于建立安全环境的加载安全区域操作而形成的。

37. 根据权利要求 34 所述的系统，其中，所述用于重新使能的指令还使得所述处理器：

15 提交摘要信息库中的数据，以指示当前活动的安全复位命令；
使得所述多个处理器中的每一个进入已知的特权状态；以及
引导存储器保护单元重新使能对所述安全存储器环境的处理器读/写访问。

38. 根据权利要求 37 所述的系统，其中，所述用于使得处理器进入特权状态的指令还使得所述处理器：

20 将所述多个处理器中的数据存储单元设置为常量或者由参数导出的值中的一个，所述常量是被存储在所述处理器中或者由所述处理器生成的，所述参数是在发出所述加载安全区域指令时由安全启动软件所规定的。

在多处理器环境中单方地加载安全操作系统的装置和方法

5 技术领域

本发明一般地涉及计算机安全领域。更具体地说，本发明涉及用于在多处理器环境中单方地加载安全操作系统的方法和装置。

背景技术

10 随着计算机与我们的社会愈加紧密接合，计算机安全方面的需求猛烈增加。近来，互联网商务在全球计算机网络上经历了巨大的增长。不幸的是，除非互联网商务使用充分证明的计算机安全机制被充分地保护，计算机剽窃的潜在性可能有一天会侵害消费者的信心。换句话说，为了获得产品和服务而提供了保密信息的计算机用户必须具有足够的保障，以便信息
15 不会被计算机剽窃者截取。

因而，许多计算机系统现在加入了必不可少的安全特征，例如加密、来源检验、受信环境以及额外的安全特征。如此，现在的在线计算机系统通常依赖于可传递的信任关系。公钥基础设施（PKI）是这种可传递信任模型的一个例子。在公钥基础设施下，认证授权机构可以向个体提供只有
20 用户知道的私钥。

因此，当用户提供信息的时候，使用计算机用户的私钥可以对其加密。如此，被加密的信息的接收者可以通过联系认证授权机构获得公钥，以便解密该被加密的信息。另外，信息的来源也可以通过数字签名消息被鉴别，该数字签名消息也可以被解密以便检验信息的来源。

25 如可以看到的那样，PKI 提供了保证对于一对一关系的安全的机制。然而，关系可能很快增长而超出一对一的交互，这要求可传递的信任以保证安全。不幸的是，信任通常是不可传递的。例如，个体可能信任认证授权机构，并从该认证授权机构收到颁布的外部证书。证书颁布之后，认证授权机构可以决定信任另一个体，并授权该个体访问和控制所有颁布的证

书，包括最初的个体的证书。

不幸的是，最初的个体可能不信任被认证授权机构信任的后来的个体。因而，如果个体在证书颁布之前知道认证授权机构信任后来的个体，则该个体可能不会请求该证书。如此，该问题显示出可传递的信任既不对称，也不可传递，也不可分配。换句话说，唯一的可靠信任是自我信任，在最初的信任形成之后不会有被认证授权机构信任的未知的后来的个体。虽然对第三方的信任并不总是不可靠的，但是也不能总是被可靠地估计。

10 在一些计算机系统中，用户或者系统管理员可能希望加载可信操作系统。对于可信，其意思是用户或者第三方要求一种机制，用于检查系统并确定是否给定的操作系统被加载了。一旦完成了对操作系统加载的检验，外部代理可能也希望确定是否操作系统被加载在一个安全环境中。不幸的是，例如公钥基础设施的传统可传递信任模型不能支持这种能力。所以，存在克服上述的现有技术中的一个或者多个局限性的需要。

15 附图说明

本发明通过示例的方式，而非限定的方式被示出于附图的图形中，其中：

图 1 描绘了一个框图，示出了本领域公知的网络计算机环境。

图 2 描绘了一个框图，示出了传统的计算机系统。

20 图 3 描绘了一个框图，示出了按照本发明一个实施例可以在其中实现本发明的系统。

图 4 描绘了一个框图，示出了按照本发明另一实施例的用于加载可信操作系统的多处理器计算机系统。

25 图 5A 和图 5B 描绘了一个框图，示出了按照本发明另一实施例的安全存储器环境。

图 6 描绘了一个流程图，示出了按照本发明的一个实施例，用于在多处理器环境中单方地加载安全操作系统的方法。

图 7 描绘了一个流程图，示出了按照本发明的一个实施例，用于在将会成为安全存储器环境的存储器区域中加载操作系统的另一方法。

图 8 描绘了一个流程图，示出了按照本发明的另一实施例，用于不理
会收到的 LSR 指令的另一方法。

图 9 描绘了一个流程图，示出了按照本发明的一个实施例，用于创建
安全存储器环境的另一方法。

5 图 10 描绘了一个流程图，示出了按照本发明的一个实施例，用于完
成安全存储器环境的形成的另一方法。

图 11 描绘了一个流程图，示出了按照本发明的另一实施例，对于外
部代理用于建立安全存储器环境安全检验的方法。

10 图 12 描绘了一个流程图，示出了按照本发明的一个实施例，响应于
SRESET 命令所进行的方法。

图 13 描绘了一个流程图，示出了按照本发明的另一实施例，响应于
收到 SRESET 命令用于复位一个或多个处理器的另一方法。

图 14 描绘了一个流程图，示出了按照本发明的示例性实施例，用于
重新使能对所形成的安全存储器环境的处理器读写访问的另一方法。

15

具体实施方式

用于在多处理器环境中单方地加载操作系统的方法和装置被描述。该
方法包括当检测到当前活动的加载安全区域指令的时候，不理睬收到的加
载安全区域指令。否则，响应于收到的加载安全区域指令，引导存储器保
20 护单元，以形成安全存储器环境。一旦被引导，对安全存储器环境的一个
或多个受保护的存储器区域的未授权的读/写访问被禁止。最后，一个或者
多个受保护存储器区域的加密散列（hash）值被存储在摘要信息库中。一
旦被存储，外部代理可以请求对被加密的标识信息的访问，以便建立安全
环境中的安全操作系统的安全检验。

25 在下面的描述中，为了说明的目的，提出了多种特定的细节，以便提
供对本发明的彻底的理解。然而，本发明可以不用这些特定细节中的某些
而被实现，这对于本领域的技术人员来说是清楚的。另外，为了举例说明
的目的，下面的说明提供了示例，并且附图示出了各种示例。然而，因为
这些示例仅仅是用来提供本发明的示例，而不是提供本发明所有可能的实

施方式的穷尽性列表，所以它们不应被认为是限定的含意。在其他情况中，公知的结构和设备以框图的形式被示出，以便避免使本发明的细节模糊。

下面的详细描述中的一些部分可能以对数据位操作的符号表达的形式和算法来表示。这些算法描述和表达被本领域的技术人员用于将他们的工作内容传达给本领域的其他技术人员。这里所描述的算法是指导致希望的结果的自恰的动作序列。动作是要求对物理量的物理操作的那些动作。这些量可以采用能够被存储、传输、组合、比较以及以其他方式被操作的电或磁信号的方式。此外，主要为了普通的用途，这些信号是指位、值、元素、符号、字符、项、数字等等。

然而，这些项以及类似的项将与适当的物理量相关联，并且仅仅是对这些量所使用的方便的标记。除非另外具体地说明，应当认识到使用例如“处理”或“计算”或“运算”或“确定”或“显示”等术语的讨论是指计算机系统或者类似的电子计算设备的动作和处理，该计算机系统或者电子计算设备将在计算机系统的设备中被表示为物理（电子）量的数据操作和转换为在计算机系统设备中被类似地表示为物理量的其他数据，这些计算机系统设备例如是存储器、寄存器或者其他这样的信息存储、传输、显示设备等。

这里所出现的算法和表示并非固有地涉及任何特定的计算机或者其他装置。各种通用系统可以按照这里的教导与程序一起被使用，或者可以证明构建更专用的装置来进行所需的方法是便利的。举例来说，根据本发明的任何一种方法可以通过编程通用处理器，在硬连线电路中被实现，或者通过硬件和软件的结合来实现。

本领域的技术人员将立即认识到，本发明可以用与下面所描述的那些结构不同的计算机系统结构来实现，包括手持设备、多处理器系统、基于微处理器的或者可编程序的消费电子产品、数字信号处理（DSP）设备、网络个人计算机、迷你计算机、大型计算机等。本发明还可以在分布式计算环境中实现，其中，任务由通过通信网络被链接的远程处理设备进行。多种这些系统所需的结构将从下面的描述中变得清楚。

应当理解，各种术语和技术被本领域的技术人员使用来描述通信、协议、应用、实施方式、机制等。一种这样的技术是根据算法或数学表达式来描述技术的实施方式。也就是说，虽然技术可以例如实现为执行计算机上的代码，但是作为公式、算法或数学表达式，该技术的表达可以被更适当和简洁地传达和传递。

因此，本领域的技术人员将会把表示 $A+B=C$ 的框认为是一个加法的函数，其在硬件和/或软件中的实施方式将是取两个输入（A 和 B），并产生一个求和输出（C）。因此，用作描述的公式、算法或者数学表达式的使用应被理解为至少具有以硬件和/或软件形成的物理的实施例（例如可以在其中将本发明的技术实践或者实施为实施例的计算机系统）。

在一个实施例中，本发明的方法体现在机器可执行指令中。这些指令可以被用于引起用指令被编程的通用或专用处理器进行本发明的步骤。可替代地，本发明的步骤可以通过含有用于进行这些步骤的硬连线逻辑的专用硬件部件来进行，或者通过被编程的计算机部件和定制硬件部件的任何组合来进行。

在一个实施例中，本发明可以被提供为计算机程序产品，其可以包括在其上存储有指令的机器或计算机可读介质，这些指令可以被用于编程计算机（或者其他电子设备）以进行按照本发明的处理。计算机可读介质可以包括但不限于软盘、光盘、压缩盘只读存储器（CD-ROM）和磁光盘、只读存储器（ROM）、随机存取存储器（RAM）、可擦除可编程只读存储器（EPROM）、电可擦除可编程只读存储器（EEPROM）、磁或光卡、闪存等。

因此，计算机可读介质包括适于存储电子指令的任何类型的媒体/机器可读介质。此外，本发明也可以作为计算机程序产品被下载。如此，程序可以从远程计算机（例如服务器）被传送到请求的计算机（例如客户端）。可以经由通信链路（例如，调制解调器、网络连接等）通过包含在载波或者其他传播介质中的数据信号的方式传输程序。

系统体系结构

现在参考图 1，图 1 描绘了网络环境 100，在其中可以实现本发明的

技术。如图所示，网络环境包括通过网络 102 互相连接的若干计算机系统，例如多个服务器 104 (104-1, ..., 104-M) 和多个客户端 108 (108-1, ..., 108-N)。网络 102 可以例如是互联网。注意，可代替地，网络 102 可以是或者包括以下的一个或者多个：局域网 (LAN)、广域网 (WAN)、卫星链路、光纤网、电缆网或者这些和/或其他组合。这里所描述的方法和装置实质上可以被应用于不论是本地还是远程的任何类型的通信装置或者设备，例如 LAN、WAN、系统总线、磁盘驱动器、存储装置等。

现在参考图 2，图 2 以框图的形式示出了传统的个人计算机 200，该个人计算机 200 可以代表图 1 中所示的客户端 108 和服务器 104 中的一个。框图是高级的概念表示，可以通过各种体系结构以多种方式被实现。计算机 200 包括总线系统 202，该总线系统 202 互连了中央处理单元 (CPU) 204、只读存储器 (ROM) 206、随机存取存储器 (RAM) 208、存储装置 210、显示器 220、音频设备 222、键盘 224、点选器 226、杂项输入/输出 (I/O) 设备 228 和通信设备 230。

总线系统 202 可以例如是系统总线、外围部件互连 (PCI)、加速图形接口 (AGP)、小型计算机系统接口 (SCSI)、固件等。CPU 204 可以是单个、多个、乃至分布式的计算资源。ROM 206 可以是任何类型的非易失性存储器，其可以是可编程的，例如掩膜可编程、快闪等。

另外，RAM 208 可以例如是静态的、动态的、同步的、异步的或者任何组合。存储装置 210 可以是压缩盘 (CD)、数字多功能盘 (DVD)、硬盘 (HDD)、光盘、磁带、闪存、记忆棒、录影机等。而显示器 220 可以例如是阴极射线管 (CRT)、液晶显示器 (LCD)、投影系统、电视机 (TV) 等。音频设备 222 可以是单声道的、立体声的、三维的声卡等。

键盘 224 可以是键盘、音乐键盘、小键盘、一系列的开关等。点选器 226 可以例如是鼠标、触摸板、轨迹球、手柄等。而 I/O 设备 228 可以是语音命令输入设备、拇指纹输入设备、智能卡插槽、个人计算机卡 (PC 卡) 接口、虚拟现实附件等，它们可以可选地通过输入/输出端口 229 连接到其他设备或系统。杂项 I/O 设备 228 的一个示例是用 I/O 端口 229 连接到乐器的乐器数字化接口 (MIDI) 卡。

通信设备 230 可以例如是用于局域网 (LAN) 连接的以太网适配器、卫星连接、机顶盒适配器、数字用户线路 (xDSL) 适配器、无线调制解调器、传统电话调制解调器、直接电话连接、混合光纤同轴电缆 (HFC) 连接、电缆调制解调器等。而外部连接端口 232 可以提供远程设备和总线系统 202 之间通过通信设备 230 的所需的任何互连。

举例来说, 通信设备 230 可是以太网适配器, 其通过连接端口 232 被连接到例如外部 DSL 调制解调器。注意, 取决于计算机系统的实际实施方式, 计算机系统可以包括框图中的部件的一些、全部、更多或者重新安排。举例来说, 瘦客户端可以由例如没有传统的键盘的无线手持设备构成。因而, 对图 2 的系统的许多变化是可能的。

返回参考图 1, 多个客户端 108 通过网络 102 被有效地连接到由诸如多个服务器 104 之类的服务器表示的网站、应用服务提供商、搜索引擎和/或数据库资源。网络浏览器和/或其他应用通常运行在多个客户端 108 上, 而信息通常驻留在多个服务器 104 上。为了便于说明, 将考虑单个服务器 104 或者单个客户端 108-1 来举例说明本发明的一个实施例。很明显, 这样的技术可以被容易地应用到多个客户端、服务器等。

现在参考图 3, 图 3 描绘了多处理器环境的计算机系统 300 的框图, 该计算机系统可以由如图 1 中所描绘的客户端或者服务器中的任何一个所利用。多处理器计算机系统 300 包括通过总线 304 被耦合到存储器控制器 310 的多个处理器 302 (302-1, ..., 302-N)。存储器控制器 310 包括存储器 320, 并通过第二总线 312 被耦合到 I/O 控制器 330。I/O 控制器 330 可以包括通过总线 332 被耦合的一个或者多个设备 340 (340-1、340-2 和 340-3)。

在例如图 3 中所描绘的计算机系统中, 用户或者系统管理员可能希望加载可信操作系统。如这里所引用的, 术语可信是指被提供给用户或者第三方用于以后检查系统 300 并确定给定的操作系统是否被加载的能力。一旦确定出给定的操作系统是否被加载, 则用户或者第三方可以进一步确定操作系统是否已经被加载进安全存储器环境中。这种机制在传统的操作系统中没有被提供。

然而，因为所描述的机制允许可信操作系统在非受信软件部件已经运行之后被加载，所以它是特别有益的。而且，在存在恶意软件的情况下，这种能力是鲁棒的，所述恶意软件在系统 300 可能正在尝试注册可信软件部件的同时，试图从例如多处理器系统中的另一处理器破坏计算机系统 5 300 的安全。

如此，在典型的计算机系统中，处理器 302 强制实施特权级别。这些特权级别限定了特定的软件部件可以访问哪些系统资源。因此，在例如图 3 所描绘的传统系统中，用户或者第三方当前不可通过对处理器 302 中的一个的单方请求而加载受信操作系统。如下面进一步详细描述，提供了 10 加载（加载安全区域）指令，响应于该指令，进行加载（加载安全区域）操作，以创建受保护的或者安全的存储器环境，在其中所选的操作系统可以作为受信操作系统具有完全的特权而运行。相应地，如这里所描述的，可互换地提到加载指令或者加载安全区域（LSR）指令。另外，可互换地提到响应于加载/LSR 指令的发出而执行的加载操作或者加载安全区域 15 （LSR）操作。

相应地，图 4 描绘了包含在安全存储器环境中单方地加载可信操作系统的能力的计算机系统 400。所提供的机制在存在恶意软件的情况下是鲁棒的，该恶意软件在系统正在注册可信软件部件的同时试图从多处理器系统的另一处理器破坏计算机系统 400 的安全。因此，计算机系统 400 差不多如图 4 中所描述的那样被配置，但是，在所描述的实施例中，存储器控制器包括存储器保护单元 402，以及第三方检验部件 450，以及存储器散列引擎 470。虽然被提供在一个特定实施例中，但是图 4 中所描绘的计算机系统 400 代表了对图 1 中所描绘的客户端 108 或服务器 104 中的任何一个所利用的计算机系统的抽象。不论怎样，本领域的技术人员将认识到， 20 计算机系统 400 的各种部件之间的划分仅仅是逻辑划分。实际上，任何部件可以被集成为被分为多个电路片（die）的同一硅电路片或者它们的组合。另外，各种部件可以在微代码、软件或者设备专用部件中被完成。

如这里所描述的，特定存储器区域被认为是包含了用户或者系统管理员希望加载的可信操作系统。在一个实施例中，该存储器区域包含了以完

全特权执行的可信操作系统的一部分。因而，计算机系统 400 需要一种机制用于防止非受信软件和设备破坏安全存储器环境。因此，在典型的计算机系统中，用于将处理器或设备存储器事务调整到安全存储器区域中的机制是必需的。

- 5 在图 4 所描绘的实施例中，计算机系统 400 的存储器控制器 410 负责将收到的存储器事务转发到适当的目标。但是，与传统的计算机系统相对比，存储器控制器 410 包括存储器保护单元 420。在所描述的实施例中，存储器保护单元 420 是一种特殊结构，其可以阻止某些存储器访问前进到存储器 412。被该结构阻止的写入操作被放弃而不影响存储器。另外，取
10 决于系统需求，读取操作可以返回这样的值：其中所有的位被清除，或者其中的位被全部设置，或者随机值。

 因而，存储器保护单元可以由启动软件使用，以便建立一个或者多个存储器区域作为上面所描述的安全存储器环境。另外，启动软件可以在安全存储器环境的指定存储器区域中加载希望的软件或者操作系统。一旦被
15 加载，只要含有该软件或者操作系统的安全存储器环境被建立，则该软件或者操作系统就作为安全软件或者可信操作系统而运行。

 从而，一旦形成安全存储器环境的存储器区域被建立，存储器保护单元就将保护该环境不被破坏。因此，一旦建立了安全存储器环境，其中所含有的软件或者操作系统就被认为是可信的。除了存储器保护单元 420 之
20 外，计算机系统 400 还包括外部检验部件 450。在一个实施例中，外部检验部件被使用，以便对于外部代理建立被包含在安全存储器区域中的操作系统或者安全软件的信任检验。

 因此，在一个实施例中，通过被提供了检查驻留在安全存储器环境中的软件的能力，外部代理可以建立信任或者决定信任被包含在安全存储器
25 环境中的操作系统。在一个实施例中，计算机系统 400 向使代表了安全存储器区域的加密散列值被检查，而不是整个区域。该值在该区域被保护之后由存储器散列引擎 470 生成，并被存储在摘要 460 中。在一个实施例中，安全存储器环境的加密散列值代表其中所包含的安全环境的软件标识值。

如此，一旦生成安全软件标识值，系统 400 就可以将安全软件标识值存储在摘要 460 中。在一个实施例中，摘要是信息库，其可以被发送给外部代理用于建立信任。在所描述的实施例中，这些值可以被读取，并在被发送给外部代理之前，由散列签名引擎 452 加密地签名。如此，在所描述的一个实施例中，摘要 460 代表摘要寄存器库，这些寄存器可以被摘要签名引擎 452 通过安全通道 454 访问。

因此，摘要签名引擎 452 利用安全通道 454 来访问摘要 460，并且会在从外部代理收到对内容签名的请求后对该内容签名。通过请求这种签名，外部代理可以观察由被签名的软件标识值所报告的规定部件，并决定是否信任计算机系统 400。在一个实施例中，被数字签名的安全软件标识值可以被存储在硬件令牌中，该令牌可以被外部代理访问。这被提供以便避免在存在试图破坏计算机系统安全的恶意软件的情况下可能容易地绕开的软件到软件的检验。

现在参考图 5A，图 5A 描绘了按照本发明实施例所形成的安全存储器环境 500 和 550。在一个实施例中，响应于加载安全区域指令的发出，形成安全存储器环境。在一个实施例中，响应于由安全启动软件发出加载安全区域（LSR）指令，进行加载安全区域（LSR）操作。从而，安全启动软件将选择希望成为安全软件的软件或者操作系统。接着，安全软件将引导所选择的软件在指定的存储器区域中的加载。

一旦软件或者操作系统被加载，在其中加载了操作系统的存储器区域就被启动软件通过 LSR 指令指定成为安全存储器环境。因此，LSR 指令将被提供给处理器 402（图 4），该处理器将执行 LSR 操作，引导处理器单方面地开始形成安全存储器环境，而不与任何一个其他的处理器（402-2 到 402-4）协作。这样做时，处理器 402 将建立 LSR 排他性互斥体，以保证没有正在进行的当前 LSR 指令。

如此，处理器将进行一系列动作，包括对容许启动软件发出这种指令的检验（例如，特权级别检查），接着引导存储器保护单元保护指定的安全存储器环境。这一旦完成，处理器就可以作为启动处理器而采集关于安全存储器环境以及标识符的信息，并将该信息记录在摘要中。如此，处理

器将提交（commit）摘要中的全部的值，并引导散列引擎计算安全软件标识值，该值将被存储在摘要中，并可以被提供给外部代理，以便建立检验。

再次参考图 5A，初始软件可以请求安全存储器环境 550，该安全存储器环境 550 包括固定基址 554 和固定偏移或者入口点 552，在其中，一旦安全存储器区域被形成，每个处理器将随着下面所进一步详细描述的安全复位（SRESET）的发出而被引导进行操作。另外，安全存储器环境 500 可以包括可变基址 504 和固定偏移入口点，在其中，一旦 SRESET 命令被发出，则引导处理器中的每一个。

因此，如下面所进一步详细描述，SRESET 命令将迫使每个处理器清除所有的未完成事务、存储器信息、任务、代码状态等。另外，处理器被引导到如图 5A 中所描绘的特定入口点，使得在 SRESET 之后进行的一个操作是在安全存储器环境中所包含的可信操作系统中。另外，在 SRESET 的时候，系统中的所有处理器都进入多个已知的特权状态 S_1 ， S_2 ，...， S_N 中的一个中。

在一个实施例中，这些状态中的每一个都可以通过一组参数被描述（例如，初始特权等级、操作模式、寄存器值、控制值等），这些参数在 SRESET 的时候从平台被读取（这种情况中，参数的值被直接或者间接地记录在摘要日志中），或者被作为处理器配置的一部分。举例来说，在一个实施例中，SRESET 之后的处理器的特权模式是固定的；该特权模式可以被编码进处理器的 SRESET 微代码序列中。然而，在 SRESET 操作之后处理器从其开始执行指令的地址可以在 SRESET 的时候从平台被读取。

在一个实施例中，处理器中的数据单元被设置为常量。相应地，这些常量或者可以被存储在例如通用代码（ucode）ROM 中（例如，将值 $0x55AA$ 存储到寄存器 N 中）。或者，这些常量可以被硬连线为芯片的操作（例如，一旦收到 RESET（复位）信号，芯片的一段中的所有触发器就被复位/设置）。最后，这些值可以从在通用代码 ROM 中的所表达的算法被导出（例如，对于高速缓存中的每个地址，将单元的地址存储到单元中）。

在另一实施例中，处理器中的数据存储单元被设置为规定的参数值。相应地，这些参数可以在发出 LSR 指令的时候由启动软件规定。这些参数可以不是被存储在寄存器中的确切的值。举例来说，一个特定参数可以例如表明“允许分页”。处理器会知道如何将参数译成将值 A、B 和 C 存储到寄存器 X、Y 和 Z 中。在发出 LSR 的时候，这些参数或者（1）被存储在平台中的预定位置中（例如，摘要、芯片组或者启动处理器），或者（2）被传送给所有处理器。于是，在发出 SRESET 的时候，每个处理器或者（1）从一个（多个）存储位置取得参数，或者（2）已经具有所有感兴趣的参数的拷贝。开始代码地址就是一个这样的参数。

因此，如图 5B 所描绘的，安全存储器区域 602 将包含如上面所描述的受保护的软件/操作系统代码 610，其将与普通环境的应用程序 640 形成对比。如此，一旦安全存储器环境 602 被形成，安全软件 610 可以运行至结束，或者启动可以执行所希望的进一步的功能的受信应用程序（小应用程序）620。因此，如上面所描述的，可信操作系统的身份被记录在摘要 460 中，并在 SRESET 命令之后，所有的处理器将开始执行属于该可信操作系统的代码。

从而，外部实体以后可以请求摘要的被签名的版本，并且能够评价系统是否在可信的状态中。在一个实施例中，上面所描述的 LSR 和 SRESER 操作可以以多种方式被公开给启动软件，例如，通过 I/O 端口访问、存储器访问、指令等。在一个实施例中，LSR 操作被实现为一个指令，SRESET 被实现为对平台芯片组的一个命令。

因此，LSR 操作可以在处理器微代码中被实现。另外，SRESET 命令可以通过芯片组逻辑被实现。此外，处理器资源可以被用于实现存储器散列引擎 470（图 4）的功能。也就是说，存储器散列引擎 470 可以是利用处理器 402 的计算序列资源的微代码序列。在另一实施例中，存储器散列引擎 470 可以在存储器控制器 410 或者 I/O 控制器 430 中被实现。

然而，由于 LSR 功能被实现在处理器中的事实，平台将能够区分由散列引擎 470 生成的消息和由通用启动软件生成的消息。在一个实施例中，引入了仅对 LSR 存储器散列微代码可用的专用消息类型。因此，微代码软

件将缺乏生成这些类型的消息的机制。在一个实施例中，SRESET 命令可以是对某个 I/O 端口地址的简单的写入，该命令被芯片组译码。从而，芯片组然后可以进行 SRESET 命令所描述的动作。

5 虽然存储器保护单元 420 被描绘为集成到如图 4 中所描绘的存储器控制器 410 中，但是本领域的技术人员将认识到，这种机制可以在 I/O 控制器和处理器中出现的功能在整个系统 400 中被实现和分配。此外，摘要和摘要签名代理可以被集成到所描述的芯片组部件中的任何一个中，或者可以被实现为分立的平台设备。在所描述的实施例中，这些部件被示为连接到 I/O 控制器 430 的分立设备。现在描述用于实现本发明教导的过程的方法。

操作

现在参考图 6，图 6 描绘了一个流程图，示出了用于在例如如图 4 所描绘的多处理器环境中单方地加载安全操作系统的方法 700。在处理框 15 710，确定加载安全区域（LSR）指令（加载指令）是否已经被收到。一旦收到了 LSR 指令，则进行处理框 712。在处理框 712，确定是否检测到了当前活动的 LSR 操作（加载操作）。进行该检测是因为任何处理器都可以各自从启动软件接收 LSR 指令。

换句话说，如上所述，这里所描述的机制提供了一种技术，用于由启动处理器进行单方的创建，以响应于例如安全启动程序而形成安全存储器环境。如此，当收到 LSR 指令时，接收处理器确定是否另外的处理器已经收到了当前正在被执行的 LSR 指令（当前 LSR 操作）。从而，当检测到了当前的 LSR 操作时，所接收的 LSR 指令将被放弃。否则，进行处理框 730。

25 从而，一旦检验出没有另外的处理器正在执行 LSR 指令，则进行处理框 730。在处理框 730，接收处理器将引导存储器保护单元 420 形成安全存储器环境，如 LSR 指令所指示。最后，在处理框 740，处理器会将安全存储器环境的加密散列值存储到摘要信息库 460 中。如所描述的，对散列值或者安全软件标识值的存储被用来使得外部代理能够进行环境的安全

检验。

现在参考图 7，图 7 描绘了一个流程图，示出了用于通过安全启动软件发起 LSR 指令的另一方法 702。在处理框 704，安全启动软件将选择一个或者多个存储器区域，用于被包括在一起作为安全存储器环境。一旦选定，则在处理框 706，启动软件将引导处理器将所选的软件或者操作系统加载到所述一个或者多个存储器区域中的一个存储器区域中。最后，在处理框 708，启动软件将向所选择的处理器发出 LSR 指令，用于将一个或者多个存储器区域形成为安全存储器区域。一旦完成，控制流转移到图 6 的处理框 710。

现在参考图 8，图 8 描绘了一个流程图，示出了响应于收到来自安全启动软件的 LSR 指令，由接收处理器进行的动作。在处理框 716，处理器将接收来自安全启动软件的 LSR 指令。一旦收到，则在处理框 718，处理器将确定安全启动软件是否被授权以发出 LSR 指令。当启动软件是未被授权的时候，LSR 指令被放弃。否则，在处理框 720，处理器将确定是否检测到了 LSR 操作的排他性互斥体。如上面所描述的，排他性互斥体使得启动处理器能够通知所有其他处理器，告知当前正在执行 LSR 指令。

当检测到了排他性互斥体的时候，收到的 LSR 指令被放弃。否则，在处理框 722，确定是否检测到了安全复位（SRESET）命令。如下面将进一步详细描述，启动处理器利用 SRESET 命令迫使每个处理器接受安全存储器区域/环境，并开始安全存储器环境的操作系统中进行操作。当检测到 SRESET 命令时，收到的 LSR 指令被放弃。否则，在处理框 724，处理器将由于上述的原因而建立 LSR 操作排他性互斥体。一旦完成，控制流转移到图 6 中所描绘的处理框 730。

现在参考图 9，图 9 描绘了另一方法 732，用于引导存储器保护单元 420 形成如图 6 所描绘的处理框 730 的安全存储器环境。在处理框 734，存储器保护单元 420 将清除摘要信息库。一旦被清除，在处理框 736，存储器保护单元 420 将采集安全存储器环境信息。在一个实施例中，安全存储器环境信息可以包括安全软件标识信息、处理器型号、状态信息等。如此，安全存储器信息的采集提供了被启动软件最初加载的软件或者操作系

统的注册信息。最后，在处理框 738，存储器保护单元 420 会将安全存储器环境信息存储到摘要信息库 460 中。一旦被存储，软件或者操作系统就在安全存储器环境中被注册为安全软件或者安全操作系统。

现在参考图 10，图 10 描绘了一个流程图，示出了用于生成图 6 中所描绘的处理框 740 的加密散列值或者安全软件标识值的另一方法 742。在处理框 744，存储器散列引擎 470 将散列安全存储器环境的内容，以生成作为安全软件标识值的加密散列值。接着，在处理框 746，散列引擎 470 会将安全软件标识值存储在硬件摘要信息库 460 中。这一旦完成，安全存储器环境以及安全软件就已经被标识和注册了。因此，在处理框 748，处理器将释放 LSR 操作排他性互斥体，以使得系统中的其他处理器能够执行 LSR 指令。

现在参考图 11，图 11 描绘了对于外部代理用于安全存储器环境的安全检验的方法 750。在处理框 752，确定是否从外部代理收到了安全验证请求。一旦收到，则在处理框 754，摘要签名引擎 452 将通过安全通道 754 从硬件摘要 460 中访问安全软件标识值。一旦进行了访问，则在处理框 756，摘要签名引擎 452 将对安全软件标识值进行数字签名。最后，在处理框 758，被签名的标识值被传送给外部代理。

现在参考图 12，图 12 描绘了方法 800，说明了如上面所描述的例如在如图 4 所描绘的计算机系统 400 中的安全复位 (SRESET) 命令。在处理框 802，确定安全存储器区域的形成是否完成了。在一个实施例中，这可以通过检验摘要 460 中的信息是否在完成 LSR 指令之后经由启动处理器已经被提交来确定。一旦完成了形成，则进行处理框 804。在处理框 804，启动处理器将向系统 400 中的多个处理器 402 发出复位命令。一旦被复位，则在处理框 820，处理器将使每个处理器能够对安全存储器环境的一个或者多个受保护的存储器区域进行读/写访问。如此，一旦读/写访问被使能，则所有处理器将开始执行属于可信操作系统的代码。

现在参考图 13，图 13 描绘了一个流程图，示出了响应于如图 12 所示的处理框 804 的复位命令所进行的另一方法 806。在处理框 808，处理器将向系统平台芯片组发出处理器复位命令。一旦发出，则在处理器 810，

芯片组平台或者平台芯片组将向计算机系统 400 的多个处理器中的每一个发出复位请求。一旦发出，则在处理框 812，各处理器将清除所有未完成的任务、事务、高速缓存存储器等，使得外部动作不会造成例如系统存储器中断。

5 最后，参考图 14，图 14 描绘了一个流程图，示出了用于使能如图 12 所描绘的处理框 820 的处理器读/写访问的另一方法 822。在处理框 824，处理器将提交摘要信息库 460 中的全部信息，以指示活动的 SRESET 命令。一旦提交了，则在处理框 826，系统 400 中的多个处理器 402 中的每一个被置于计算机系统 400 中可用的高级特权状态或者已知的特权状态
10 中。

 接着，在处理框 828，各处理器将被引导到已知的状态，可以包括例如多个已知特权状态 (S_1, S_2, \dots, S_n) 中的一个、操作模式、寄存器和控制值以及例如图 5A 中所描绘的安全存储器区域中的预定入口点。最后，在处理框 830，存储器保护单元 420 将被引导以重新使能对于安全存储器
15 环境的处理器读/写访问。以这种方式，可信操作系统的身份被记录在摘要中，并在 SRESET 之后，所有处理器将开始执行属于该可信操作系统的代码。此外，外部代理可以在以后请求摘要的经签名的版本，并能够评价该系统是否在可信状态中。

20 替代实施例

 已经描述了用于提供在多处理器环境中单方地加载安全操作系统的安全存储器环境多处理器系统的一个实施方式的若干方面。但是，安全存储器环境多处理器系统的各种实施方式提供了包括补充、增加和/或替换上述特征的许多种特征。特征可以在不同的实施方式中被实现为同一硅片的一
25 部分、多个电路片、它们的组合，或者微代码的一部分，或者专用硬件或软件部件。另外，为了说明的目的，前面的描述使用了特定的术语来提供对本发明的彻底的理解。然而，为了实现本发明，这些特定细节并不是必需的，这对于本领域的技术人员是清楚的。

 另外，虽然这里所描述的实施例是针对安全存储器环境多处理器系

5 统，但是本领域的技术人员应当认识到，本发明的教导可以被应用于其他系统。实际上，用于单方地加载可检验的可信操作系统的系统是在本发明的教导之内，而不脱离本发明的范围和精神。上述实施例被选择和描述以便最佳地解释本发明的原理及其实际应用。因此，这些实施例被选择以使得本领域的技术人员能够最佳地利用本发明和适合于设想的具体用途的具有各种修改的各种实施例。

应当理解，尽管在上面的描述中已经连同本发明各种实施例的结构和功能的细节，提出了本发明各种实施例的许多特性和优点，但是被公开仅是示例性的。在一些情况中，某些子部件仅仅用一个这样的实施例被详细
10 描述。然而，应当认识到并且其意思是这样的子部件也可以被用在本发明的其他实施例中。可以在本发明的原理之内对细节，尤其是部件的管理和结构，在最大可能的范围内作出变化，该范围由所附权利要求被表达于其中的项目的更宽广的一般含意所指示。

本发明提供了优于已知技术的许多优点。本发明包括在多处理器环境
15 中单方地加载安全操作系统的能力。本发明提供了一种机制，通过该机制，系统可以在安全存储器环境中安装可信操作系统。因而，用户或者系统管理员可以利用如上面所描述的其后跟随 SRESET 指令的 LSR 指令来加载可信操作系统。一旦完成，外部代理以后就可以检查系统，并确定给定的操作系统是否被加载，并且如果是的话，它是否已经被加载到了安全环境
20 中。因此，由于所描述的这种机制允许可信操作系统在非受信的软件部件已经运行之后被加载，所以它是特别有益的。另外，在存在恶意软件的情况下该机制是鲁棒的，所述恶意软件在系统正在注册可信软件部件的时候，试图从多处理器系统中的另一处理器破坏计算机系统的安全。

已经公开了示例实施例和最佳模式之后，可以对所公开的实施例作出
25 修改和变化，而保持在由所附权利要求所定义的本发明的范围之内。

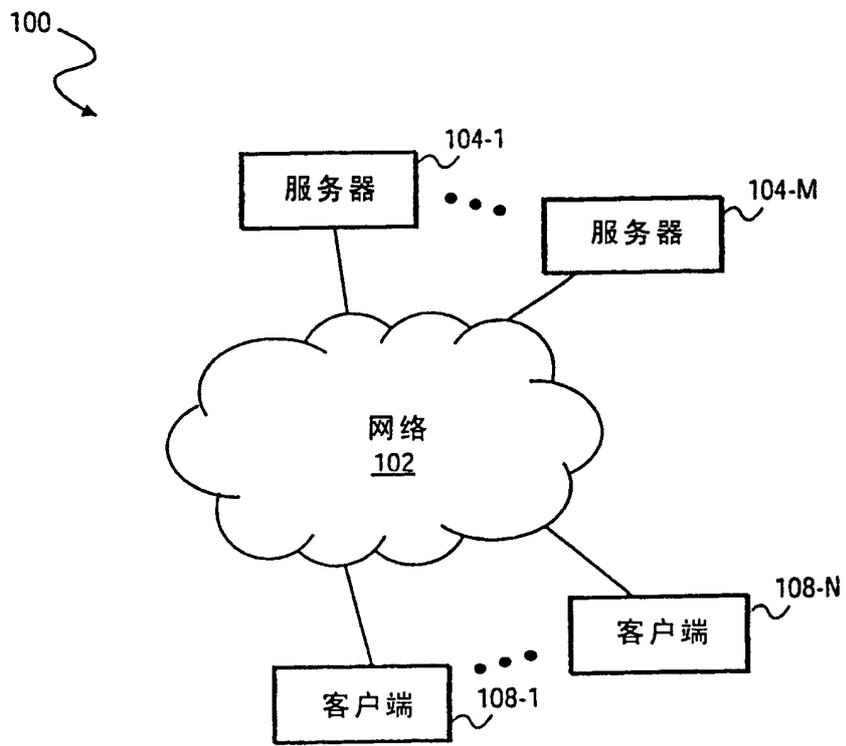


图1

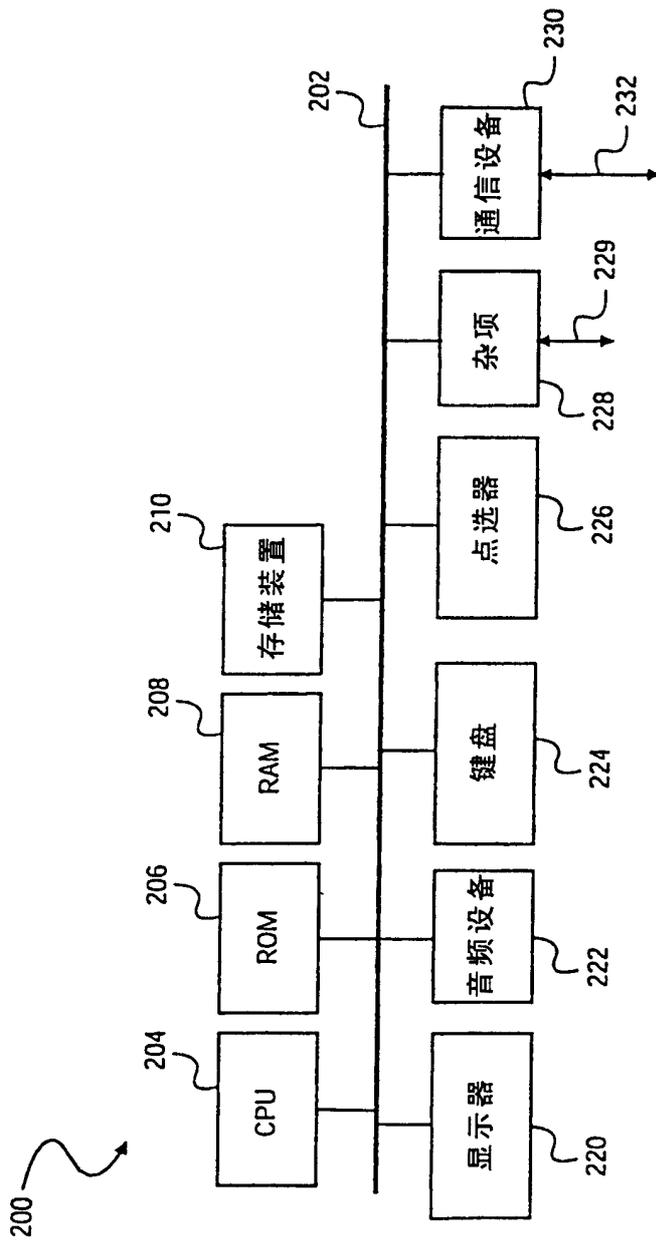


图2

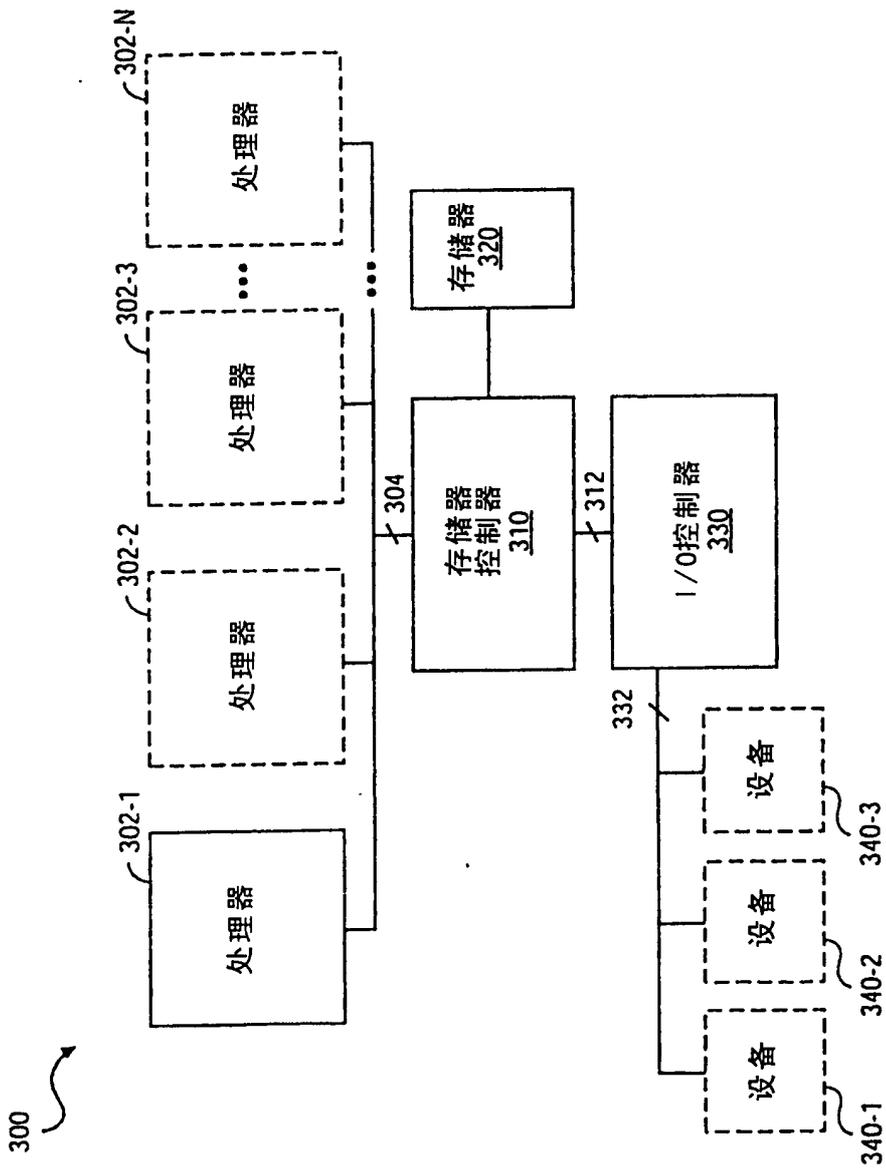


图3

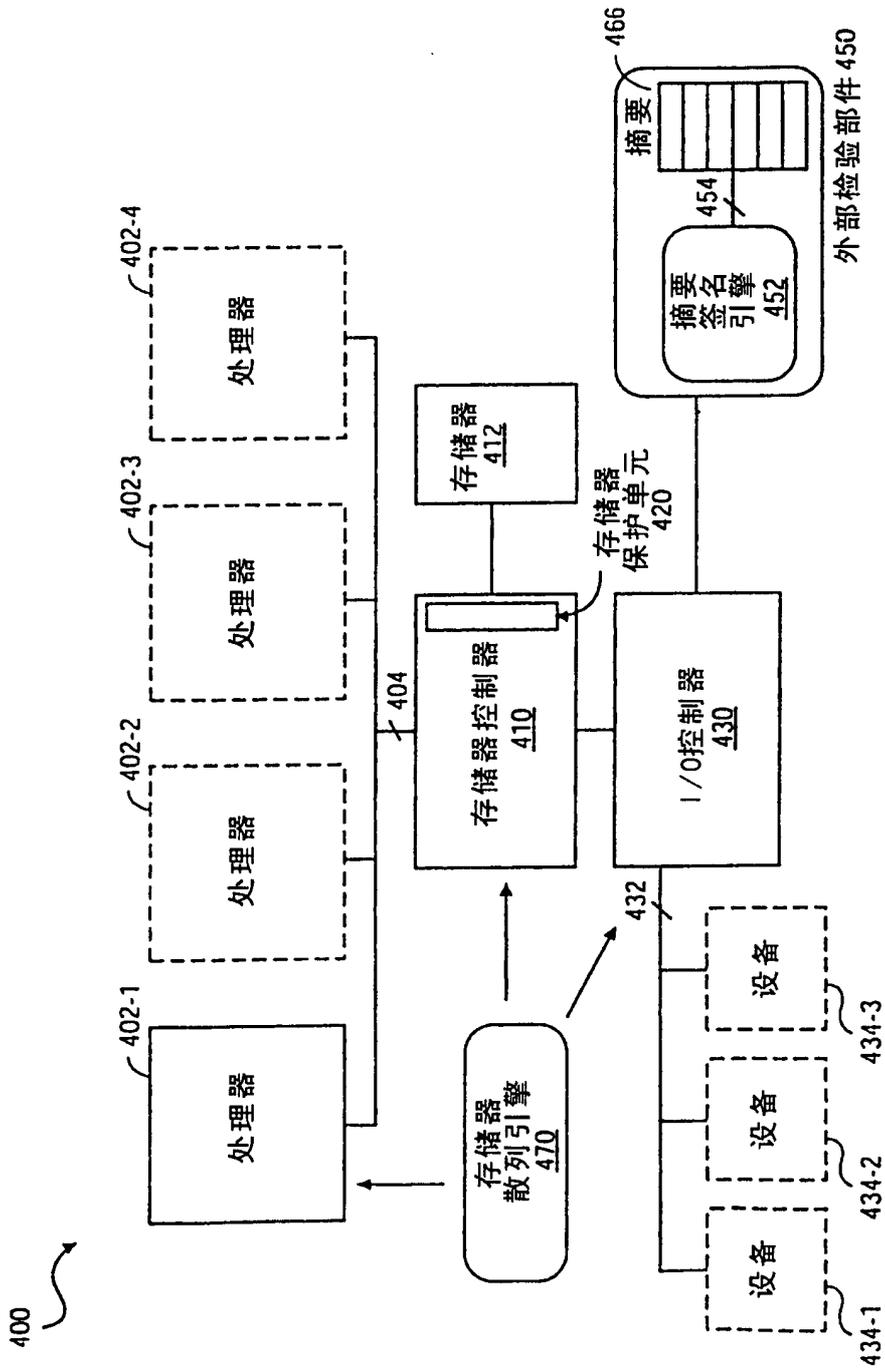


图4

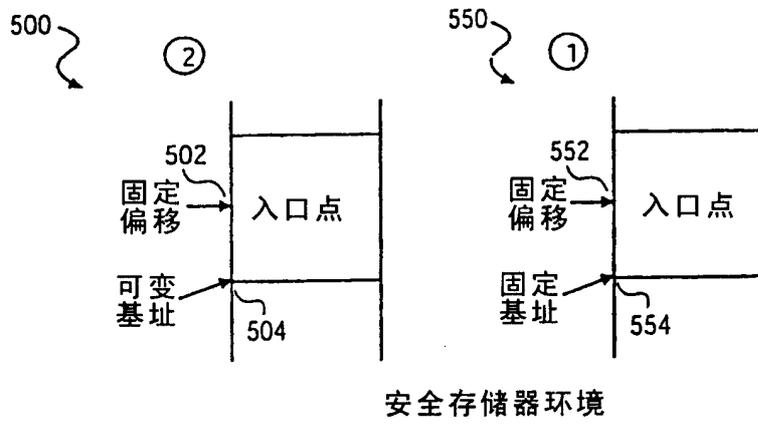


图5A

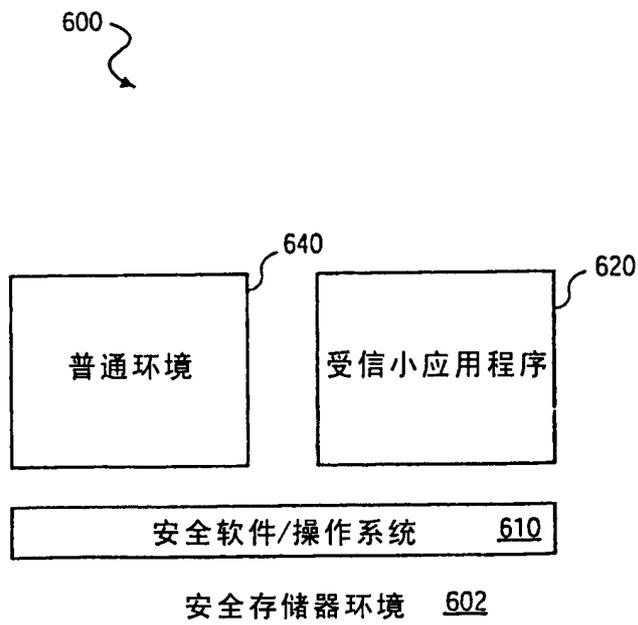
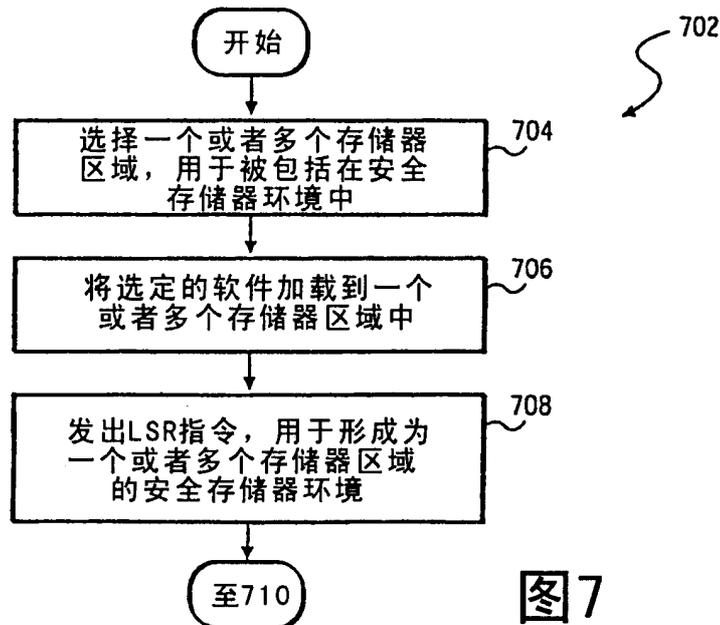
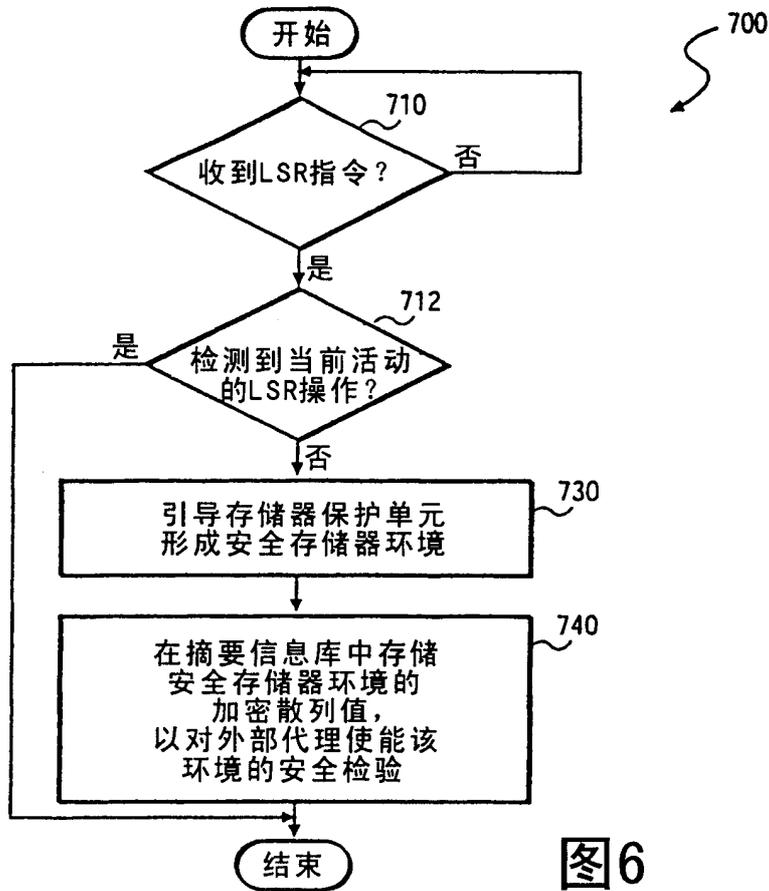
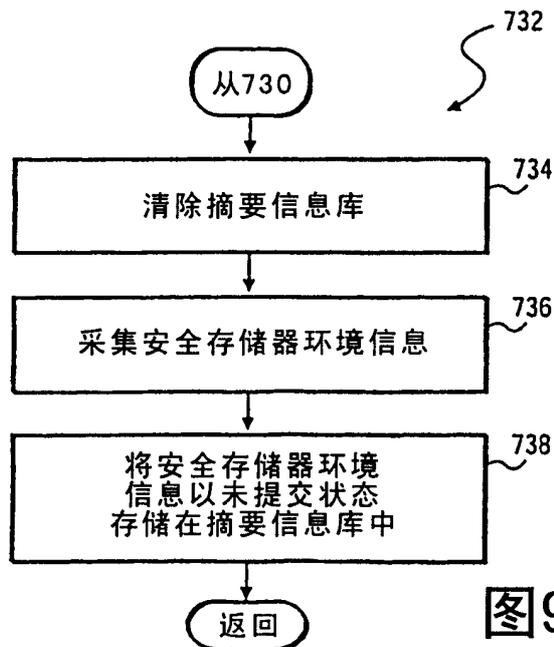
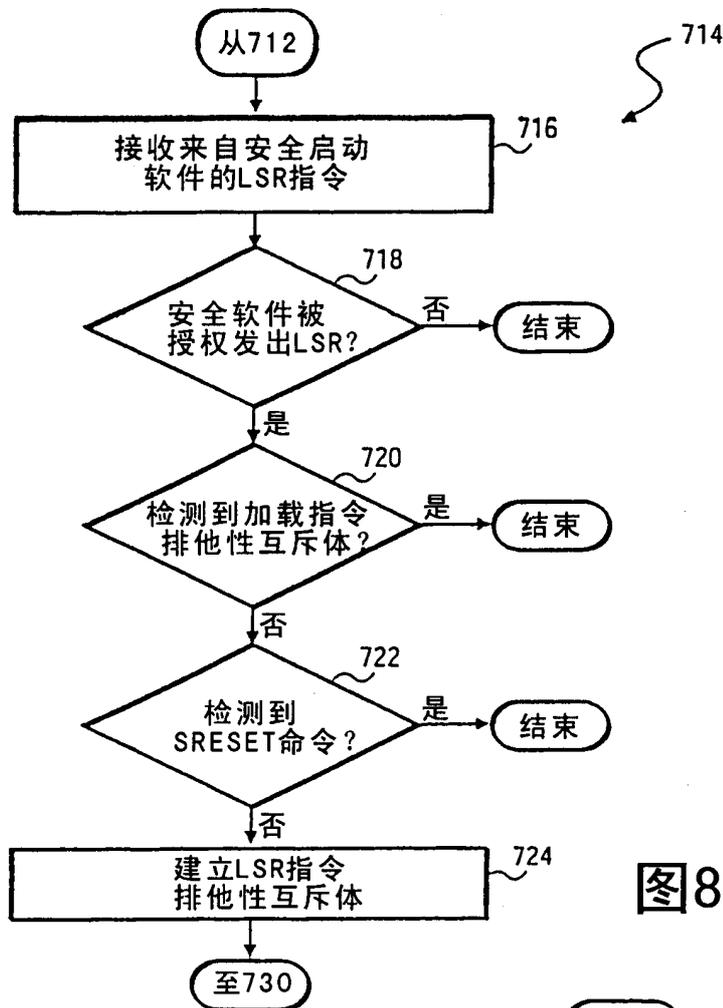


图5B





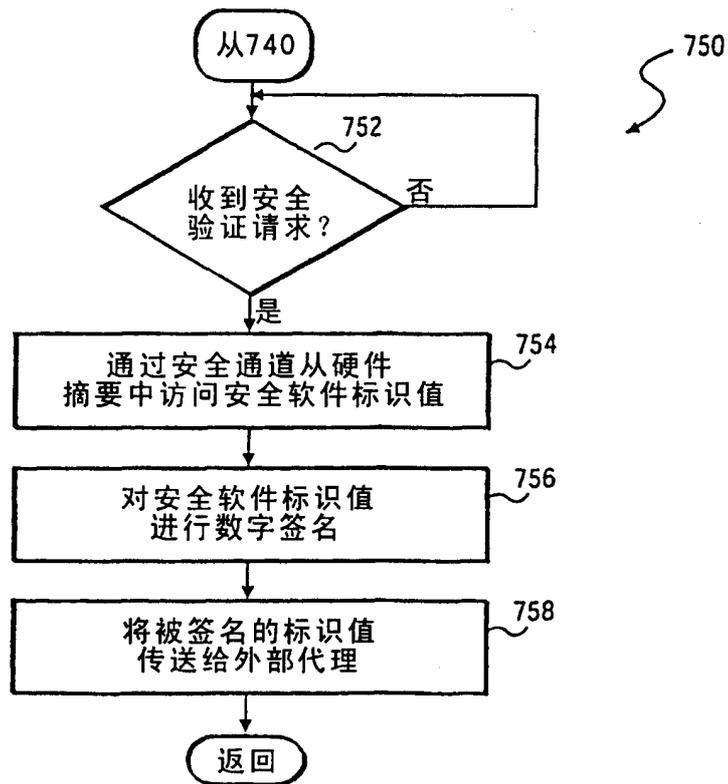
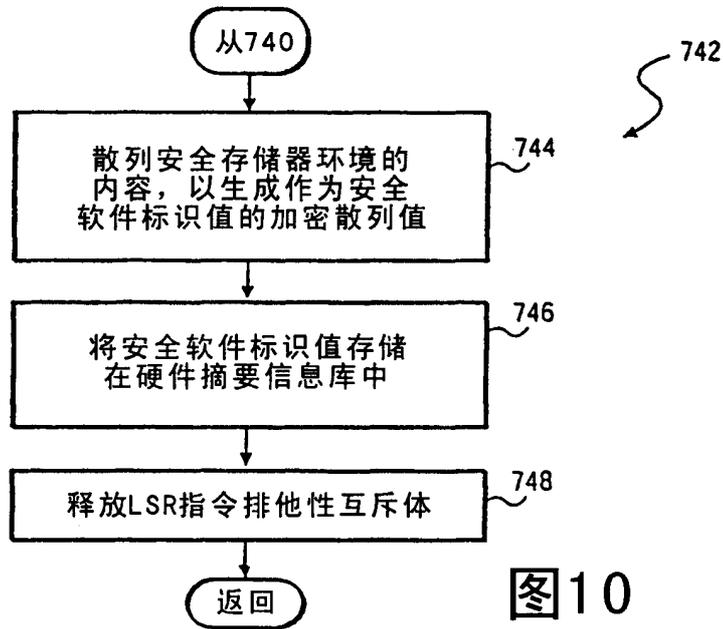
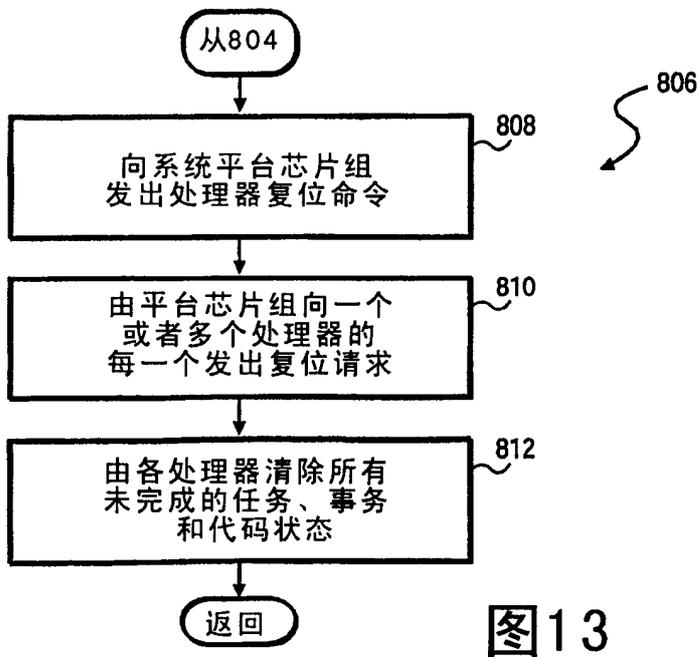
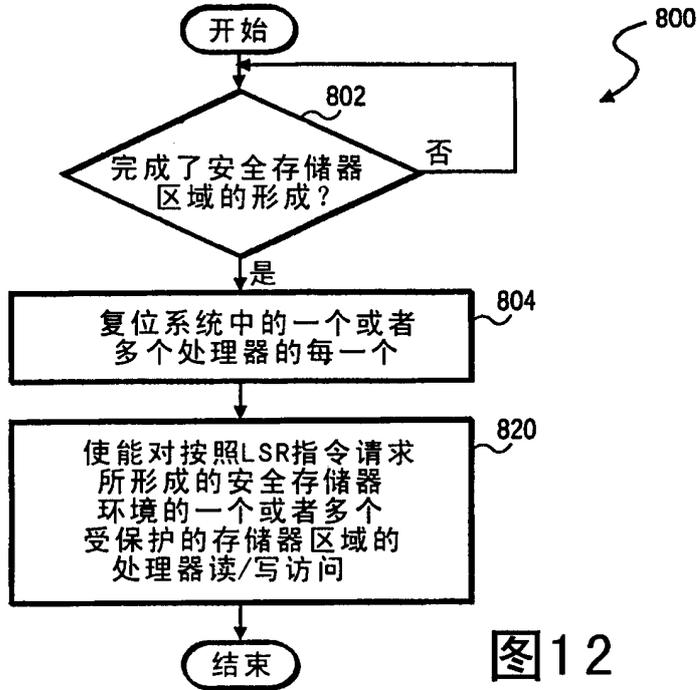


图11



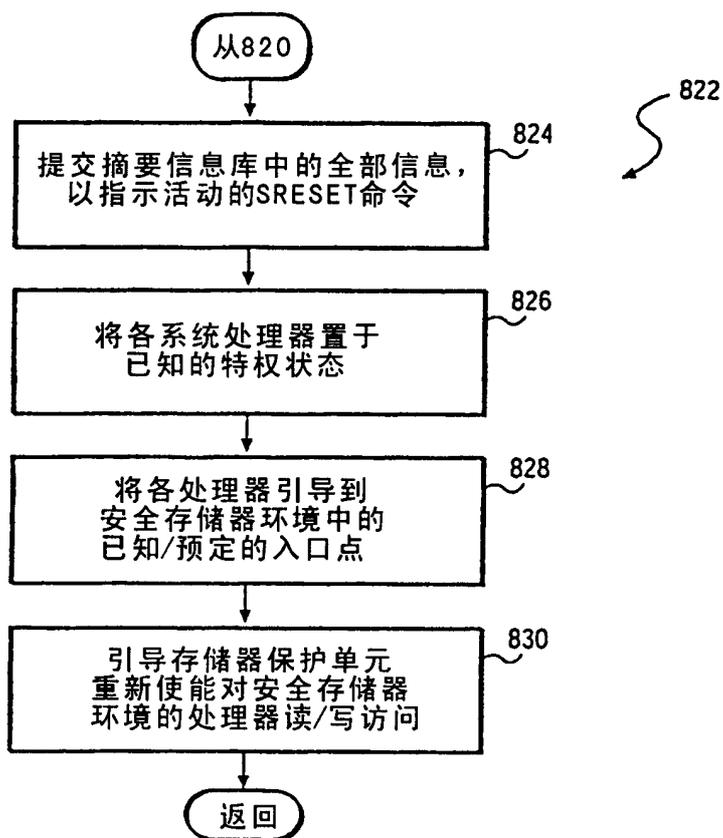


图14