(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 105787372 A
(43)申请公布日 2016.07.20

(21)申请号 201610205761.X

(22)申请日 2010.08.11

(30)优先权数据
61/234604 2009.08.17 US
12/580891 2009.10.16 US
61/257043 2009.11.02 US
61/286369 2009.12.14 US
12/714547 2010.03.01 US

(62)分案原申请数据
201080046637.4 2010.08.11

(71)申请人 高通股份有限公司
地址 美国加利福尼亚州

(72)发明人 B.M.雅各布松 K-A.R.约翰松

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 张健 刘春元

(51)Int.Cl.
G06F 21/56(2013.01)
G06F 21/57(2013.01)
H04W 12/12(2009.01)

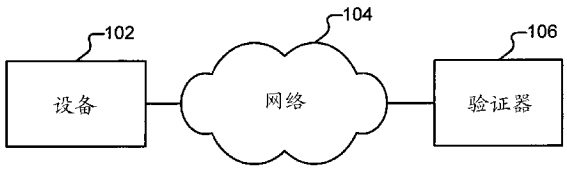
权利要求书2页 说明书22页 附图10页

(54)发明名称

审核设备

(57)摘要

公开了对包括物理存储器的设备的审核。接收与硬件配置对应的一个或者多个硬件参数。也接收初始化信息。选择性地读取物理存储器并且确定至少一个结果。向验证器提供结果。



CN 105787372 A

1.一种验证器,包括:

一个或多个处理器,被配置成:

至少部分地基于计算集合来获得至少一个结果,其中所述至少一个结果包括指示与所述计算集合相关联的处理持续时间的定时测量;

至少部分地基于所述至少一个结果和所述定时测量的评估来确定是否设备已受未授权程序影响,其中所述评估确定所述定时测量是否超过所确定的时间长度的容许;

基于所述设备是否已受未授权程序影响来生成安全状况的指示;并且

将所述指示提供给所述设备。

2.根据权利要求1所述的验证器,其中所述计算集合包括种子值和/或密钥值。

3.根据权利要求2所述的验证器,其中所述种子值和/或密钥值是在按照需要的基础上提供的。

4.根据权利要求2所述的验证器,其中,所述一个或多个处理器还被配置成存储一个或多个种子和/或密钥值直至被需要。

5.根据权利要求1所述的验证器,其中所述计算集合是对物理存储器执行的。

6.根据权利要求1所述的验证器,其中所述验证器处于所述设备外部且与外部服务提供商相关联。

7.根据权利要求1所述的验证器,其中所述指示包括对资源的访问的准予。

8.根据权利要求7所述的验证器,其中所述资源包括本地资源。

9.根据权利要求7所述的验证器,其中所述资源包括与外部服务提供商相关联的网络资源。

10.根据权利要求1所述的验证器,其中所述一个或多个处理器还被配置成执行审核。

11.根据权利要求1所述的验证器,其中所述一个或多个处理器还被配置成执行防病毒扫描。

12.根据权利要求1所述的验证器,其中所述一个或多个处理器还被配置成确定所述设备是否已被越狱、是否已被解锁或者是否包括盗版媒体。

13.一种验证器,包括:

一个或多个处理器,被配置成:

从用户标识模块(SIM)卡接收结果值,其中所述结果值至少部分地基于包括执行至少一个写入操作的计算集合的执行和挑战值;

至少部分地基于从SIM卡接收的结果值、预计结果值、指示与所述计算集合的执行相关联的处理持续时间的定时测量、以及所述定时测量是否超过所确定的时间长度的容许,来确定是否设备已受未授权程序影响;以及

至少部分地基于所述设备是否已受未授权程序影响的确定来生成安全状况的指示;

将指示提供给所述设备;以及

存储器,耦合到所述一个或多个处理器且被配置成给所述一个或多个处理器提供指令。

14.一种验证器,包括:

一个或多个处理器,被配置成:

接收设备的安全状况的指示;

从所述设备接收与包括执行至少一个写入操作的计算集合的执行相关联的结果值;并且

至少部分地基于所述结果值和指示与所述计算集合相关联的处理持续时间的定时测量的评估来确定所述设备是否已受未授权程序影响,其中所述评估确定所述定时测量是否超过所确定的时间长度的容许。

15.根据权利要求14所述的验证器,其中计算采用重排、混合和偏移使用中的一个或多个。

16.根据权利要求14所述的验证器,其中计算采用重排,其中计算中采用的位被重排以需要散列函数的集合的计算来重组验证值的集合中的任一个验证值。

17.根据权利要求14所述的验证器,其中计算采用混合,其中计算的输出的每一个位是计算的所有输入位的函数,每一个输入位需要散列函数评估以计算。

18.根据权利要求14所述的验证器,其中计算采用被应用于存储器的偏移使用,其中计算的每个线程与所述存储器的不同的部分对应以避免冲突。

19.根据权利要求14所述的验证器,其中所述结果值包括由代理提供的时间戳。

20.根据权利要求14所述的验证器,其中所述结果值包括由代理提供的时间戳。

审核设备

[0001] 本申请为分案申请，其母案的发明名称为“审核设备”，申请日为2010年8月11日，申请号为201080046637.4。

[0002] 对其他申请的交叉引用

本申请要求对于2009年8月17日提交的美国临时专利申请号61/234,604、标题为DETECTION OF MALWARE的优先权、于2009年11月2日提交的美国临时专利申请号61/257,043、标题为AUDITING A DEVICE的优先权以及于2009年12月14日提交美国临时专利申请号61/286,369、标题为AUDITING A DEVICE的优先权，出于所有目的通过引用将以上每个全部结合于此。

背景技术

[0003] 用于检测存在未授权程序现有技术通常是资源密集的。例如它们一般需要持续更新(例如黑名单)和针对问题定期或者连续扫描。如果受这样的技术保护的设备具有有限资源(诸如有限的存储器)或者由电池供电则使该情形恶化。作为一个示例，具有有限资源的设备可能不能存储用于检测所有已知未授权程序的定义。作为另一示例，扫描未授权程序通常是功率密集的动作并且可能快速耗尽电池供电的设备的电池。在一些环境中，中央机构用来促进发现未授权程序。这一方式的一个缺点在于它通常需要受保护的设备编译设备活动的详细日志。生成这样的日志是资源密集的(例如需要大量盘储存；用于汇编日志数据的处理能力；以及用于向中央机构递送日志数据的带宽)并且也可能出现隐私问题。

[0004] 用于检测存在未授权程序的现有技术一般还易受这样的程序尝试引起不正确报告的攻击。例如rootkit可以“监听”应用向操作系统的请求并且可以修改这些请求及其响应。如果应用请求关于什么进程在运行的信息，则恶意rootkit应用可以通过从操作系统返回的报告中去除关于其本身的信息来避免检测。

[0005] 也已知用于屏蔽安装或者执行未授权程序的现有技术易受恶意软件的如下新实例攻击，由于缺乏关于这些实例的结构和功能的信息而可能不可立即检测它们。因此并且无论可用于设备的资源如何，如果未授权程序充分复杂和/或前所未见，则它可以规避检测并且引起未检测到的损害。并且如果用户已经有意安装未授权程序以绕过检测(例如为了促进软件盗版)，则传统技术可能无法定位未授权程序或者任何其他未授权活动。

附图说明

[0006] 在下文详细描述和附图中公开本发明的各种实施例。

[0007] 图1图示了其中提供设备审核的环境的实施例。

[0008] 图2图示了设备的实施例。

[0009] 图3图示了用于执行设备审核的过程的实施例。

[0010] 图4图示了用于执行设备审核的过程的实施例。

[0011] 图5A图示了在执行图3中所示过程之前的存储器的表示。

[0012] 图5B图示了在图3中所示过程出现之时的存储器的表示。

[0013] 图6图示了用于执行设备审核的过程的实施例。

[0014] 图7图示了用于与审核设备结合使用的伪代码的示例。

[0015] 图8图示了用于执行设备审核的过程的示例。

[0016] 图9图示了其中提供设备审核的环境的实施例。

[0017] 图10图示了设备的一部分的实施例。

[0018] 图11图示了用于执行设备审核的过程的实施例。

[0019] 图12图示了根据步进读取存储器的一部分。

[0020] 图13图示了用于选择性地读取存储器的过程的实现的实施例。

[0021] 图14图示了用于对设备审核的一部分定时的过程的实现的实施例。

具体实施方式

[0022] 现在用多种方式实现本发明，包括实现为过程；装置；系统；物质组成；在计算机可读存储介质上体现的计算机程序产品；和/或处理器（诸如被配置成执行在耦合到处理器的存储器上存储和/或由该存储器提供的指令的处理器）。在本说明书中，这些实现或者本发明可以采用的任何其他形式可以称为技术。一般而言，可以在本发明的范围内变更公开的过程的步骤顺序。除非另有指明，描述为被配置成执行任务的部件（诸如处理器或者存储器）可以被实现为暂时被配置成在给定的时间执行该任务的一般部件或者被制造以执行该任务的具体部件。如这里所用，术语‘处理器’是指被配置成处理数据（诸如计算机程序指令）的一个或者多个设备、电路和/或处理核。

[0023] 下文与图示本发明原理的附图一起提供对本发明的一个或者多个实施例的详细描述。结合这样的实施例描述本发明，但是本发明并不限于任何实施例。本发明的范围仅受权利要求书限制，并且本发明涵盖诸多替代方案、修改和等效实施例。在下文描述中阐述诸多具体细节以便提供对本发明的透彻理解。出于举例的目的而提供这些细节，并且无这些具体细节中的一些或者所有细节也可以根据权利要求实现本发明。为求简洁，不详细地描述与本发明有关的技术领域中已知的技术材料，使得没有不必要地使本发明难以理解。

[0024] 图1图示了其中提供设备审核的环境的实施例。在所示示例中，设备102是蜂窝电话。设备102与验证器106通信（例如经由网络104）。在图1中，设备102经由3G网络来与验证器106通信。验证器106在电信公司（诸如设备102的电话服务提供商）的控制之下。验证器106包括硬件配置信息的数据库，包括与设备102和在设备102上包括的RAM数量对应的条目。

[0025] 如下文将更详细说明的那样，可以审核设备102，使得可以检测和/或去除设备上存在的任何规避程序（例如恶意软件）。在一些实施例中，这通过进行对设备102上包括的物理存储器的修改序列来实现。与存储器修改的执行关联的结果由验证器106验证。一旦确定设备102未受这样的规避程序影响，可以执行下文也更详细描述的附加扫描。例如除了检测恶意软件（例如在用户未知情和/或同意时安装的软件）之外，这里描述的技术也可以检测用户采取的“越狱”动作（例如权限提升）以诸如避开电信公司或者硬件制造商安装的数字版权管理。

[0026] 多种设备可以与这里描述的技术结合使用。例如在一些实施例中，设备102是视频游戏控制台。视频游戏控制台被配置成经由因特网（104）在游戏控制台的制造商的控制之

下与验证器通信。如果设备102的所有者对设备102做出未授权改变(例如通过使用修改芯片),则验证器将能够相应地检测修改。

[0027] 可以与这里描述的技术结合使用的设备的其他示例包括桌面型计算机、笔记本计算机、上网本、个人数字助理、视频回放设备(例如电视机、DVD播放器、便携视频播放器)、路由器、接入点、机顶盒、医疗设备以及包括处理器和存储器的实质上任何其他设备。

[0028] 在各种实施例中,验证器106由设备102的用户而不是单独实体控制。例如设备102的用户拥有的桌面型计算机可以被配置成向设备102提供验证服务。在该场景中,设备102可以被配置成经由本地网络来与验证器通信。设备102也可以被配置成直接与验证器106通信(例如经由专用线缆),并且如适用的那样省略网络104。

[0029] 在一些实施例中,验证器与设备102并置或者否则直接耦合到设备102。例如向蜂窝电话中插入的用户标识模块("SIM")卡可以被配置成向蜂窝电话提供验证器106的功能。作为另一示例,验证器106的功能可以集成到用来对蜂窝电话充电的电源线中。在这样的实施例中,可以省略外部验证器,或者除了并置/耦合的验证器提供的验证服务之外还可以使用外部验证器。作为示例,假设设备102是具有集成WiFi能力的个人视频播放器。用来对设备充电的电源线可以被配置成每当设备充电时向它提供验证服务。此外,如果WiFi无线电活跃,则设备可以被配置成与设备的制造商提供的验证器定期通信。作为另一示例,验证器106可以被包括在USB设备上,该USB设备由用户定期插入到膝上型计算机102上。此外,无论膝上型计算机102的用户何时尝试进行与在线银行的银行交易,银行也可以在准予访问用户的账户之前向膝上型计算机102提供验证服务。作为又一示例,网络运营商或者服务提供商可以要求用户让他的或者她的机器在他或者她在网络上被允许或者他或者她被允许访问服务之前被审核。用户也可以例如在认识到他或者她已经暴露于潜在风险的情形之后发起审核。用户可以发起审核的一种方式是选择设备上的菜单选项。另一示例方式是让用户请求来自验证器106的审核(例如通过web表单提交在线请求)。

[0030] 图2图示了设备的实施例。在所示示例中,设备102包括处理器202、第一存储器204、第二存储器206和通信接口208。作为一个示例,设备102包括528 Mhz ARM处理器(202)、128MB的RAM(204)、用户已经向其中插入1GB微SD卡的微SD卡槽(206)和3G调制器解调器(208)。存储器204这里也称为"快速"存储器。存储器206这里也称为"慢速"存储器。然而存储器204和206无需是不同速度。在设备102中也可以包括其他部件(诸如GPS接收器(未示出))。也可以如适用的那样省略元件,诸如第二存储器206。可以将可以包含活跃程序的RAM称为快速,并且将仅可以存储数据的RAM视为慢速。

[0031] 使用这里描述的审核技术,可以验证快速存储器中不存在活跃进程。并且在已经完成该验证之后,可以扫描所有存储器(例如快速和慢速)以标识、分类、报告和潜在修改快速和慢速存储器的内容或者其部分。可以用各种方式做出在快速与慢速存储器之间的区分。例如在具有RAM、闪存和硬盘驱动器的设备上,有可能仅将RAM视为快速存储器并且将闪存和硬盘驱动器视为慢速存储器。也有可能将RAM和闪存两者均视为快速存储器并且将硬盘驱动器视为慢速存储器。也有可能将物理上位于给定的设备上的所有存储器视为快速并且将设备可访问(或者潜在可访问)的所有外部存储器视为慢速。无论外部存储器的类型和实际本地访问速度如何,用于与外部部件通信的周转时间(turnaround time)将使这样的外部访问更慢。根据将什么类型的存储器视为快速比对慢速,将相应完成参数选择。

[0032] 如下文将更详细描述的那样，可以通过配置设备102运行对存储器204的一系列修改并且检查结果来检测存在对设备102的未授权修改。例如如果执行修改所花费的时间超过预定时间长度的容许或者如果与修改结合确定的结果与预计结果不匹配，则可以指示存在规避程序。在各种实施例中，跨设备上的所有存储器（例如存储器204和存储器206）执行而不是仅在快速存储器（诸如存储器204）上运行存储器修改。

[0033] 图3图示了用于执行设备审核的过程的实施例。在各种实施例中，图3中所示过程由设备102执行。可以用多种方式发起图3中所示过程。例如每当用户对设备充电时可以发起该过程（例如通过配置设备在它检测到电源时发起该过程）。也可以响应于出现特别大或者异常交易、响应于担心用户处于风险（例如响应于电信公司接收恶意个人已经发布新漏洞的通知）、响应于某一时间量的流逝等发起该过程。可以触发发起图3中所示过程的事件的附加示例包括设备102的用户尝试进行支付或者否则从事金融交易、认证尝试（例如设备的用户尝试访问银行账户）以及执行访问请求（例如请求向设备下载电影）。

[0034] 该过程在302始于接收与硬件配置对应的一个或者多个硬件参数时。示例硬件参数包括快速存储器204的量和速度。例如在图2中所示设备的情况下，硬件参数将包括“量=128M”和“速度=300 Mhz”。可以使用的附加参数包括核数量、总线类型等。

[0035] 可以用多种方式接收硬件参数。作为一个示例，蜂窝电话的SIM可以被配置成检测安装的存储器的量和速度。作为另一示例，如果专用线缆用来将设备102连接到电源（或者连接到计算机或者其他设备），则可以借助仅与具有具体量和速度的存储器的设备结合工作的线缆来知道（并且因此“接收”）参数。作为又一示例，设备的系列号可以指示设备上安装的快速存储器204的量和速度。在各种实施例中，要求用户（或者其代表）在web表单或者配置文件中输入存储器参数。也可以关于设备的可能存储器配置做出假设，并且可以运行基准测试程序以确认假设是否可能正确。

[0036] 在304执行对物理存储器的修改序列。下文更信息描述可以执行这样的修改的方式的示例。在一些实施例中，待执行的修改序列由验证器确定。可以用多种方式向设备提供待做出的修改集合。例如可以基于种子值在设备上构造序列。可以在制造时、在向供应商或者电信公司递送时或者在购买时向设备上预装序列。它也可以在购买之后的任何时间（例如作为额外更新（over-the-update）或者作为固件更新）或者在需要执行审核时由用户选择加载或者由服务提供商加载。参数化可以在给定的已知规范时由制造商或者供应商或者电信公司执行。也可以通过用户或者服务提供商例如系列号的查找来执行它。参数可以与机型或者设备名称关联。如果例如通过更换或者添加部件来重新配置设备，则这些新部件可以携带关于新的或者附加参数化的信息。部件也可以携带整个指令集而不是仅参数。替代地，部件系列号、名称或者类型可以指示所需参数改变。如果在安装算法或者新部件时认为客户端设备安全，则客户端机器也可以询问安装什么部件（如在引导系统时通常完成的那样）并且相应设置参数。

[0037] 在各种实施例中，设备制造商赋予无成本预装未激活的审核软件并且以后请求支付以激活下文更详细描述的审核服务（和/或附加扫描服务。审核软件可以随后在最终用户或者服务提供商请求时由电信公司激活。电信公司收取用于激活的支付并且可选地向手持设备制造商、审核软件提供商、附加扫描软件（例如防病毒检测服务）提供商以及在交易中涉及到的任何其他方转交支付的部分。

[0038] 在306，向验证器报告在304执行的过程部分的一个或者多个结果。在一些实施例中，向代理906提供结果，该代理906将结果加上时间戳并且向验证器提供它们。如将结合图5描述的那样，在一些实施例中，进行对存储器的多个修改迭代和与验证器的通信，并且相应适配图3和图4中所示过程。

[0039] 图4图示了用于执行设备审核的过程的实施例。在各种实施例中，图4中所示过程由验证器106执行。如上文说明的那样，在一些实施例中，图4中所示过程由与设备102分离的实体(诸如在电信公司控制的验证器上)执行。在其他实施例中，该过程由位于设备102上或者否则在物理上耦合到设备102的验证器执行。

[0040] 该过程在402始于接收结果时。例如当设备102在306报告结果时，那些结果在402由验证器接收。

[0041] 在404，关于在404接收的结果是否指示做出过预计的物理修改序列做出确定。利用信息配置验证器106，信息诸如执行存储器修改序列应当在设备106上花费的时间量(假设尚未做出已授权修改)。在一些实施例中，验证器106也被配置成存储附加信息，诸如种子值和设备102执行的计算结果。

[0042] 如果确定已经做出预计的物理存储器修改序列(例如设备106执行存储器修改序列)，则推断(406)尚未对设备做出未授权修改。并且已经使可能先前已经在设备102上活跃的任何规避进程无效。如果确定尚未做出预计的物理存储器修改序列(例如，因为用于执行该序列的时间量用完或者计算结果不正确)，则推断(406)已经对设备做出未授权修改(例如，规避进程存在于设备上并且尝试避免检测)。在各种实施例中，纠错代码用来避免由于网络噪声的错误。消息认证代码和其他认证技术可以用来避免主动的内容篡改。加密技术可以用来使内容模糊并且使窃听者不可能确定在传输明文消息。

[0043] 图5A图示了在执行图3中所示过程之前的存储器的表示。在所示示例中，内核502、已授权程序504、未授权程序(例如恶意软件代理)508和审核器程序506被加载到RAM中。通常为了保持驻留于设备上，规避程序需要完成两件事之一。它必须在RAM(或者交换空间)中保持活跃或者它必须修改设备的合法程序、数据或者配置以允许恶意软件代理在已经执行扫描之后获得控制。如下文将更详细说明的那样，使用这里描述的技术，无论恶意软件代理用来规避检测的技术如何都可以检测它的存在。此外，使用这里描述的技术，即使在恶意软件代理504之后加载审核器506，仍然可以检测恶意软件代理的存在。

[0044] 图5B图示了在图3中所示过程出现时的存储器的表示。如下文将更详细说明的那样，审核器506被配置成清空除了审核器506使用的空间之外的存储器RAM(和任何交换空间)。在各个实施例中，也允许最小其他服务集合以占用RAM。例如，如果设备102支持3G通信，则不清空3G驱动器/模块占用的RAM区域，使得审核器506可以使用3G调制解调器来与验证器106通信。作为另一示例，在一些实施例中，允许微内核占用RAM的部分而审核器506清空RAM的其余部分。

[0045] 图6图示了用于审核设备的过程的实施例。该过程在602始于在设备(诸如设备102)上运行的审核器过程清空未声明供审核器使用的存储器204(和任何交换空间)的所有部分时。在一些实施例中，这包括卸载内核、各种驱动和所有其他进程。在各种实施例中，未声明的存储器空间由序列改写而不是清空(例如改写为零)。一个示例序列是诸如通过使用XOR操作来与原始存储器内容组合的伪随机序列。这允许以后通过与补充或者等于先前所

用伪随机序列的伪随机序列的重复组合来重组未声明的存储器空间。也可以用如下方式用内容改写未声明的存储器空间，该方式清空它、但是不对应于设备的典型擦除操作。例如有可能通过向未声明的存储器写入序列01010101或者任何其他适当序列来清空它。

[0046] 在一些实施例中，审核器代码包括两个组成：加载器和变量算法段。加载器的任务是从非RAM储存（例如除了存储器204之外的储存）加载算法段并且向加载的算法段移交控制。在算法段已经完成之后，它向加载器移交回控制。

[0047] 在604，向验证器106报告存储器204的内容。在一些实施例中，报告全部内容。在其他实施例中，仅传送对自上次审核以来的改变的描述。

[0048] 在606，设备从验证器接收密码种子（cryptographic seed）。种子扩展成伪随机串，并且向RAM写入该串。下文提供用于根据过程600的部分606向RAM写入串的示例技术。

[0049] 在608，设备从验证器接收密码密钥。

[0050] 在610，设备使用接收的密钥来计算设备的RAM的全部内容的密钥化散列。

[0051] 在612，设备向验证器报告所得值。验证器106例如根据图4中所示过程来评估结果。

[0052] 在各种实施例中，设备102在验证器106设置的时间间隔处在606和610报告来自计算的状态信息。使用这样的间隔提供如下保证：在存储器204（而不是例如存储器206的部分）内执行设备102执行的计算。

[0053] 设备102在按照需要的基础上从验证器106获得种子和相应密钥的更新。使用更新提供设备102未向外部快速资源外包计算的保证。例如为了外包计算，规避程序将必须向外部设备转发种子和密钥更新，这将引入可测量延迟。

[0054] 验证器106验证最终函数值和部分结果正确并且在可接受时间界限内由设备102报告给验证器。下文提供用于评估审核器执行它的任务而花费的时间的示例技术。如上文提到的那样，在一些实施例中，验证器106在设备102外部并且由除了设备的所有者之外的一方操作。在其他实施例中，验证器106在设备102的用户的控制之下。

[0055] 在已经完成图6中所示过程之后，审核器506可以恢复设备的内容（无论是完全还是部分）并且向先前活跃进程或者向执行存储器内容的进一步扫描的进程返回控制。如果在执行定时计算之前向慢速存储器交换出快速存储器的内容，或者如果原始内容与串组合（后者允许执行相似组合），则可以恢复该内容，由此恢复先前状态。也有可能通过加载“开始状态”来重启设备。还有可能先向扫描、复核、报告和修改存储器内容或者这些操作的任何子集（下文更详细描述）的进程移交控制。可以向验证器106或者向第三方（诸如负责管理存储器内容处理的第三方）呈现报告。在后一种情况下，验证器106可以负责保证无活跃恶意进程，并且第二验证器可以负责处理设备的存储器以确定它是否顺应可以与恶意软件检测、数字版权管理或者标识什么设备存储器内容是期望的另一方针有关的特定方针。

[0056] 示例敌对策略

规避程序为了例如在图6中所示过程的部分604期间避免被检测到，它必须在RAM中活跃作为唯一进程（504）或者作为审核器506的受破坏版本的部分。下文是规避程序（诸如恶意软件代理504）可以尝试保持活跃的六种示例方式：

策略1：外包存储。

[0057] 恶意软件代理可以在RAM中保持活跃并且尝试通过使审核器106不清空适当空间

(例如在602)并且依赖于非RAM储存或者外部储存存储在606生成的伪随机串的对应部分来保持未被检测到。然后将修改在610的计算以使用外包储存而不是恶意软件代理驻留的空间。

[0058] 策略2:计算遗漏数据。

[0059] 代替外包存储伪随机串的部分的存储,恶意软件代理可以存储串的修改表示(例如压缩版本或者作为遗漏部分的版本)并且在610计算密钥化散列期间需要串的相关部分时重组它们。由于恶意软件代理具有用来生成伪随机串的种子,所以它可以使用这一种子——或者伪随机生成器的以后状态——来再生所需数据部分。

[0060] 策略3:外包计算

恶意软件代理可以向外部设备转发相关数据(假设仍然使能需要的通信基础设施(诸如WiFi连接))。外部设备从设备102接收数据并且计算为了向验证器106报告而需要的值,从而向设备102上的恶意软件代理馈送这些值。

[0061] 策略4:修改检测代码

恶意软件代理可以尝试将审核器506的代码替换为修改代码。可以将这一替换代码设计成抑制报告受危及的存储器内容或者包含用于在审核完成之后加载恶意代码的钩子(hook)。恶意软件代理可以尝试通过交换出或者压缩审核器代码的部分并且在需要它时再次加载或者解包它来并入对审核器506的这样的改变而未占用更多空间。

[0062] 填充快速存储器

这一节描述可以与图6中所示过程的部分606结合使用的示例技术。

[0063] 图7图示了用于与审核设备结合使用的伪代码示例。在所示示例中,子例程get_permutation返回矢量,该矢量指示范围从0到number_blocks-1的number_blocks个项目的随机排列,其中number_blocks是大小与RAM包括的闪存块相等的部分减去审核器需要的那些部分的数量。子例程next_string_chunk返回伪随机生成的位块;术语块用来指代可以在存储器总线上发送的数据量。作为一个示例,对于Android G1电话,块是32位。

[0064] get_permutation和next_string_chunk均使用最近提供的种子作为输入。可以将伪随机串计算为$segment_i \leftarrow hash(segment_{i-1})$(即以不能使用随机访问来计算的方式)。一个示例是在给定散列函数的非同态(non-homomorphic)性质时基于散列函数迭代应用的函数。可以使用多种散列函数。一个示例是在512位模式中的MD6。

[0065] 常数rounds是使用函数modify_memory向单元的内容中异或(XOR)伪随机块的次数。rounds的选择控制敌对方为了执行第二敌对策略(包括计算遗漏数据)而必须执行的工作量而又也给针对大值的诚实执行算法带来增加的成本。在所示示例中,rounds=2,这给敌对方造成比rounds=1明显更大的成本,因为每个单元的值将变得依赖于两个其他单元。这可以使敌对方的存储器管理策略混乱。在所示示例中,chunks_per_block是闪存块中包含的块数量,对于示例G1电话而言等于32768(=128kB/32位)而number_blocks=1024(=128MB/128kB)。

[0066] 函数modify_memory(pos, string)将位置pos的内容与值string异或,其中pos=0描述待操作的RAM的第一个块,并且pos=number_blocks×chunks_per_block-1是最后一个块。

[0067] 结合图7描述的存储器访问结构如果被迫使用闪存(例如存储器206)而不是RAM

(204)则引起对随机排序块的个别页面的访问。这将引起以巨大概率清空闪存，并且伪随机访问顺序防止敌对方调度存储器访问以避免这一缺点。闪存界限的计算与可用于诚实执行算法的RAM界限的替代方案相比的成本是明显更耗费时间。

[0068] 在一些实施例中，一种散列函数应用用来生成next_string_chunk的若干调用。这减少与审核过程关联的计算负担，该过程强调存储器访问在用于执行任务的时间方面的贡献。

[0069] 在各种实施例中，向散列函数的输入是恒定数量的先前输出；这使得用于希望对伪随机生成器的给定部分的状态进行重组的恶意软件代理的存储变复杂，并且因此对进一步挫败对使用策略2(计算遗漏数据)的任何尝试有用。

[0070] 执行定时

这一节描述可以用于对执行审核任务定时的示例技术。例如在一些实施例中，该技术如结合与图6对应的文字描述的那样由验证器106运用。

[0071] 验证器106被配置成对图6中所示过程的部分606和610的执行定时以例如标识对外包存储、计算遗漏数据以及外包计算的尝试。

[0072] 在一些实施例中，验证器106被配置成以频繁间隔(例如由验证器106设置)从设备102获得状态信息。状态信息的一个示例是上次更新的存储器块的存储器内容，这证明设备102已经到达计算的这一阶段。验证器106以规律间隔向设备102发送更新请求。在一些实施例中，更新请求对应于用来计算子例程next_string_chunk的输出的伪随机生成器的状态更新。如果通过从已经生成的伪随机串选择未用部分来生成子例程next_string_chunk的输出，则可以同时清空串，因此迫使新种子立即影响状态。

[0073] 运用敌对策略3(即外包计算)的规避程序必须向执行计算的外部设备传输伪随机串的更新，此后外部设备必须计算设备102将向验证器106报告的所得下一个值并且向规避程序传输这一个值。这带来了往返延迟。如果往返延迟超过在定时检查点之间的时间，则将检测到欺骗。这里做出种子和密钥与其他状态信息一起在客户端设备与验证器之间安全传送的假设。各种密码技术可以用来实现这一点。

[0074] 在各种实施例中，选择在检查点之间的设备特有时间，使得无足够时间使用在设备102上包括的通信设备(例如WiFi)来外包计算，从而悲观地假设无拥塞环境。

[0075] modify_memory的执行时间由上文描述的参数选择和使用什么散列函数来计算next_string_chunk确定。例如MD6散列函数可以被配置成从224到512位的不同输出大小。如上文说明的那样，在一些实施例中，使用512位版本。modify_memory的每次调用时间明显少于上文确定的在检查点之间的时间。

[0076] 检测各种规避程序的示例

下节提供可以如何使用这里描述的技术来检测运用上述各种策略的规避程序的示例。

[0077] 防范敌对策略1——外包存储。

[0078] 假设已经向设备102中插入空SD卡。对应写入速度可以到达直至5MB/s。如上文描述的modify_memory所处理的块的大小在这一示例中选择为128kB。用于向SD卡写入数据的时间将是25ms。作为比较，假设设备102上的RAM具有100MB/s的写入速度。对应写入时间将是1.25ms。可以容易检测到附加延迟。并且如果在两个检查点之间进行对SD卡的多次访问，则将甚至更容易检测到附加延迟。

[0079] 防范敌对策略2——计算遗漏数据。

[0080] 如上文提到的那样，可以用不能使用随机访问来计算的方式来计算伪随机串。为了计算某个输出的值，需要根据存储的数据计算对应输入。由于rounds>1，所以RAM中存储的数据并非这一所需状态而是两个或者更多回合的状态的组合。状态需要由恶意软件代理显式存储(在RAM中)作为它的代码空间的部分和据此计算的所需状态。这迫使恶意软件代理除了它为了执行“合法”计算而需要的计算之外还在执行图6中所示过程的部分610期间计算至少rounds×number_blocks×chunks_per_block次(并且事实上比这一数量多得多的)散列操作。对于所选参数选择，该数量将多于100百万次散列操作。这在给定用于计算散列函数调用的近似时间为10μs时约为1000s，该时间近似为比预计的并且可以相应检测的时间多3个量级。

[0081] 现在将提供与防范策略2对应的样本证明。假设规避程序驻留于存储器204中并且占用一些c个32位块的至少部分用于它本身及其变量。可以做出所有这一空间可以有效用来存储变量这样的悲观假设，这并不可能，但是给出关于恶意软件为了保持未被检测到而必须执行的工作量的下界。在现实中，它的付出更大，因为并非所有c个块都可以用于存储而是需要一些块来存储它的代码。

[0082] 对于未包含为了计算函数而需要的值的RAM块的c次命中中的每次命中，恶意软件代理必须计算预计内容。在这一示例中假设原始内容——在执行RAM填充之前——是零。如果并非如此，则恶意软件代理的付出将更大，因而做出这一假设建立关于恶意软件代理的付出的下界。为了计算RAM填充算法将执行的对这一单元的预计更新，恶意软件代理需要计算用于在讨论的存储器块上的所有rounds遍(pass)的值。异或到存储器中的值来自伪随机序列。并且仅有可能通过根据恶意软件代理在c个存储块的部分上存储的值计算next_string_chunk来重组链在如下单元中的状态，它在该单元遗漏。假设变量仅存储于RAM中或者恶意软件代理需要还有策略1(外包存储)成功。

[0083] 如上文说明的那样，不能使用随机访问方式来计算伪随机生成器。情况是在给定32位的块大小和512位的状态大小(=MD6输出大小)时需要L=16个块来存储状态。恶意软件代理必须根据与这一状态关联的RAM位置(该位置不必是恶意软件代理存储这一状态的位置)重新计算散列函数调用序列。

[0084] 在给定在存储器写入期间单元内的随机排列(恶意软件代理不能预期的顺序)时，运行到与存储的状态对应的串位置的预计长度至少为rounds×n×(c/L)，其中n=number_blocks×chunks_per_block对应于构成RAM的块的数量，rounds×n是伪随机串的长度，并且其中c/L是恶意软件代理存储的伪随机状态的数量。因此对于“坏”单元的每次命中，恶意软件代理必须执行next_string_chunk的预计rounds×n×L/c次调用，这对应于rounds×n×/c。有c次这样的命中，未对在恶意软件代理尝试计算预计状态之一时出现的对“坏”单元的命中进行计数。因此，恶意软件代理必须执行至少rounds×n次散列操作以根据存储的内容计算c个坏块的内容。用于这样做的近似时间(根据示例实现)至少在比合法客户端慢100,000-1,000,000倍之间，这指示将检测到对计算遗漏数据的任何尝试。

[0085] 如果链的计算引起访问如下单元，该单元已经用来存储用于计算的另一遍的值，则这引起招致另一命中。如上文描述的那样，它将以针对每个存储访问的近似概率 $(c-c/rounds)/c\times c/number\_blocks=(c-c/number\_blocks)/number\_blocks\approx c/number\_blocks$ 并且因此以

针对给定的第一坏单元命中的近似概率 $1-(1-c/number\_blocks)^{number\_blocks*rounds^2/c}$ 发生。这一数量的粗略近似是 $1-e^{-rounds^2}$。对于rounds=2，这多于98%的概率。这一附加成本将随着c的值增加而增加。敌对方因此将尽力使c小。

[0086] 在下文中假设敌对方仅使用c=L=16个单元(所有16个单元用于存储一个值)。利用这一配置，敌对方将无法在链在如下那些情况中计算值(除非使用外部存储器)，在那些情况中，链通往如下方向，该方向不使根据“存储单元”中的值计算与“编程单元”对应的值成为可能。对于rounds=2，这一失效以概率75发生。在其余25%的情况下，将简单地减缓敌对方。(为了总是成功计算值，敌对方需要存储至少rounds=2个值，每个值为512位长)。

[0087] 防范敌对策略3——外包计算。

[0088] 在一些实施例中，选择在检查点之间的时间，使得无时间使用设备上的通信设备来外包计算。校验器106可以用如下方式选择在检查点之间的时间，该方式使这立即可检测。涉及到外包计算的策略将失败，因为必须在两个检查点之间完成往返以便敌对方提供正确值。这独立于在客户端设备与验证方之间的通信速度。

[0089] 防范敌对策略4-修改检测代码

假设未授权程序508破坏(例如如结合图6描述的)一些步骤的执行，然后自愿加载合法代码并且去除它本身。这样的敌对方可以潜在破坏该过程的部分602和604，但是将不能破坏部分606。具体而言，它需要破坏该过程的部分602(清空交换空间和RAM)以便维持活跃。它然后可以引起在604的错报状态。然而这将在计算存储器内容的密钥化散列时被检测到(610)。这既归因于使用的散列函数的假设无冲突，并且又归因于直至608才向设备公开密钥的事实。如上文描述的那样，在606期间不活跃(这又将引起检测)的情况下不能破坏部分608。并且当未执行图6中所示过程的部分606时规避程序将不能计算将在612报告的正确值。

[0090] 四个敌对策略的组合也将失败，因为将检测它们中的每个策略并且它们的组合未改变底层设备特有限制。

[0091] 附加处理

图8图示了用于执行设备审核的过程的示例。在各种实施例中，上文描述的审核过程形成两个(或者更多)阶段过程的一个阶段(802)。一旦已经关于设备(诸如设备102)运用上文描述的技术，可以做出无规避软件在设备的RAM中活跃的假设。然后可以在设备上执行任意附加处理(804)。下文描述可以执行的附加处理的示例。

[0092] 示例:恶意软件

在执行802的处理之后，设备102在804执行传统防病毒软件以标识诸如可以存储于存储器206中的已知坏软件。设备102在804也可以被配置成向验证器106或者向另一设备报告存储器206的全部内容或者存储器的部分。

[0093] 示例:越狱

在执行802的处理之后，设备102在804确定它的操作系统加载器是否具有特定散列和/或否则确定是否已经变更操作系统加载器离开希望的状态。

[0094] 示例:电话解锁

在执行802的处理之后，设备102在804确定它的操作系统加载器是否已经变更并且也

确定与服务提供商关联的任何信息是否已经变更。

[0095] 示例:软件盗版

在执行802的处理之后,设备102在804确定是否已经从预计配置修改存储器206中包括的任何软件、断定用于软件的任何关联系列号和/或否则确定是否在以未授权/未许可方式使用包括的软件。在一些实施例中,设备102向验证器106报告存储器206或者其部分的内容。

[0096] 示例:媒体盗版

假设在分发期间使用水印来定制媒体文件(例如音乐、视频或者图像文件)并且例如使用MAC或者数字签名来密码认证这些水印。在804可以确定设备102上存在的哪些文件具有合法水印并且这些是否包含有效认证方。可以在设备102本地或者集中(例如在验证器106上)做出该确定。

[0097] 在各种实施例中,应用(诸如设备102上安装的音乐播放器)记录使用和其他数据(形成活动日志)并且将信息与适当媒体(例如歌曲文件)关联。日志可以在804由验证器106读取。

[0098] 示例:监管(custody)链/使用日志

假设应用(或者数据文件)具有用来记录交易的关联日志文件。一个示例是日志文件,该文件记录金融交易的出现(包括存储值信息)。可以验证对日志文件做出的改变的合法性如下。先执行802的处理。然后在804可以关于是否已经变更应用(或者数据文件)并且因此关于日志文件是否真实做出确定(例如通过比较程序映像的散列)。

[0099] 在这一示例中的一种用于在804执行的处理的方式如下:先扫描存储器206并且创建应用的列表和与应用关联的数据文件。接着确定用于应用和数据文件的描述符的列表。描述符的示例是文件的散列以及文件的名称和类型以及声称它与(一个或多个)什么串序列匹配的标识符。接着产生对尚未在第一列表中报告的应用或者数据的任何描述的第二列表。这里创建的描述可以包括用于应用的代码或者对它处理什么类型的输入文件和它产生什么类型的输出文件的描述中的全部或者部分。向外部方(诸如验证器106)传输第二列表,它在该验证器被验证。也可以使用从方针验证服务器获得的任何方针来本地处理第二列表。

[0100] 验证的结果可以用来影响对应用和数据的准许并且可以用来控制外部服务器如何与设备交互,包括是否准予它访问网络资源(诸如因特网、3G网络、公司网络等)。作为另一示例,被允许在设备上运行的软件可以被限制并且向用户通知缺乏顺应、尝试去除或者修复或者否则修改文件等。

[0101] 示例:亲代(parental)控制过滤器和其他监视特征

在执行802的处理之后,在各种实施例中,安装附加中间件,这些中间件可以被配置成记下(和/或阻止)与设备关联的各种事件。示例包括:

(a)确定在设备上生成并且以后传出什么照片(以例如防止“性短信”)。

[0102] (b)确定(例如基于设备活动和GPS改变)在以大于20英里每小时的速度行进时是否使用设备(例如用于发短信或者观看视频剪辑)。

[0103] (c)确定(例如基于安装活动)是否已经安装替代应用(诸如除了默认程序之外的第二即使消息接发程序),并且然后为替代应用创建日志文件。

[0104] (d)确定(例如基于浏览器历史信息)用户已经拜访什么URL,包括人工导航到哪些URL以及在访问的其他HTML文档中引用哪些URL。这一记日志的一个益处是标识个人是否可能已经沦为网络钓鱼(phishing)的受害者;已经拜访已知分发不希望内容(包括恶意软件)的web站点;以及设备是否可能卷入点击-欺诈。在未传染设备本身时有可能实现这样的滥用(诸如通过使用JavaScript、联结样式表和/或其他有关脚本编写语言)。

[0105] 示例:附加应用

除了上述示例之外,这里描述的技术的更多使用是可能的。例如设备审核可以在车载黑匣子中用于计量使用、保险、关税、税务、通行费等用途——均为了标识恶意软件和有意篡改。

[0106] 可以在其他应用中包括设备审核技术作为组成,从而允许这些应用暂时挂起它们本身以在已知干净状态中执行扫描并且以后再次被给予控制。

[0107] 作为又一示例,技术可以在医疗设备中用来确定它们是否未受感染、被正确配置和维护并且以便在知道谁有权访问数据和设备变得有价值时审核在特殊情况下的使用。讨论的设备可以用预装应用不能干扰的方式一直记下使用信息;审核过程将包括存储器打印扫描以断言预装应用仍然在良好状态下并且不存在冲突应用或者配置。

[0108] 最后,技术可以用于在无需补救或者这并非主要目标的情形中检测恶意软件。一种这样的背景对于在线游戏而言是在游戏中检测不存在用于作弊的模块。

[0109] 保留隐私

在一些实施例中,向验证器106传送对所有状态(例如存储器204的内容)的描述。然而应当优选地未从设备102传走一些数据,诸如私有密钥和不可执行的私有数据。在下一节中描述保留这样的数据的隐私的技术。

[0110] 假设第一随机数称为x并且它选自于可能值1..max_x的某些空间。可能的是x除了向它针对的审核过程提供输入之外还对恶意软件编码。合法程序根据输入数据x和一些系统参数计算称为(g_1,n_1)的单向函数值y。这样做的一种示例方式是通过计算以模n_1的$y=g_1^x$,其中g_1生成G_{n1}的大子群。

[0111] 让程序然后根据值y和一些系统参数计算称为(g_2,n_2)的第二单向函数值z。这样做的一种示例方式是通过计算以模n2的$z=g_2^y$,其中g_2生成G_{n2}的大子群。

[0112] 接着假设客户端机器证明(例如使用零知识证据)有值x使得$z=g_2^{g_1^x \text{ modulo } n_1}$模n2,其中验证器已知(z, g_1, g_2, n_1, n_2),但是(z,x)未知。设备(“证明器”) 然后擦除值x,但是存储(y,z)和参数(g_1, g_2, n_1, n_2)。

[0113] 在以后时间,设备必须证明它存储的、但是是密码的值y对应于值z。(这里z可以存储于设备102上,但是也可以由验证器106存储。)可以使用的一个示例证据是零知识证据。

[0114] 如果第二证据结束并且验证器106接受它,则验证器知道客户端存储的未知值z是如下格式,该格式不能用来向恶意软件代理隐藏大量值数据。

[0115] 这里,z可以用来加密某些其他数据,该数据称为m并且其密文称为c。因此$c=E_z(m)$用于加密算法E。假设对称加密,$m=D_z(c)$用于某些解密算法D。可以验证设备内容,但是m保持不为接收c的一方所知。这一方将不知道z、但是仅知道z是不能隐藏大量恶意软件数据的某些可接受形式。由于这里描述的审核过程允许向验证器方保证仅合法程序存在于客户端

设备的RAM中，所以在给定c时可以知道程序——使用秘密值z——可以访问m。然而验证器不能。

[0116] 由于已知访问程序合法，所以也已知将仅以批准的方式访问m。例如如果m是数据而不是代码，则情况是访问程序将不试图执行数据。

[0117] 使用伪随机串生成器

图9图示了其中提供设备审核的环境的实施例。在所示示例中，设备902除了图2中所示部件之外还包括SIM，该SIM被配置成作为用于外部验证器904的代理(906)。如下文将更详细描述的那样，设备102的指令高速缓存中存储的单片(monolith)内核(在它完全适配时)在它被激活时交换出所有其他进程(除了它例外选择的任何进程之外)并且执行审核过程。单片内核具有位于数据高速缓存(和寄存器)中的关联工作空间。高速缓存通常使用RAM来实施并且这里被视为是它的部分。如这里使用的那样，“自由RAM”是在已经交换出所有应用——包括普通内核——之后应当自由的RAM部分。在一些实施例中，“自由RAM”定义为批准的例程和数据集未占用的RAM段。例如普通内核可以是批准的例程，其可能是常用和列入白名单的应用。另外，批准的数据可以对应于外部验证器已知的数据并且可以是任何格式，只要它列入白名单(即视为安全)即可。在这样的情况下，批准的程序无需交换到(如下文更详细描述的)次级存储而是可以代之以在审核的存储器读取部分期间保持驻留(例如1108)。

[0118] 在一些示例中，单片内核对应于针对已知执行环境ε参数化的程序F_ε。如上文说明的那样，执行环境对应于设备的硬件配置。对输入x执行F_ε产生输出序列$F_{\varepsilon i}(F_\varepsilon, x)$，每个输出在从执行开始起的时间$t_i(F_\varepsilon, x)$内并且产生结束状态$s(F_\varepsilon, x)$。在这一示例中，$x \in X$，其中X是所有合法输入的集合。

[0119] 代理906用来减少来自于设备的延时差异并且在各种实施例中取代SIM或者除了SIM之外还实施为有绳蜂窝电话、蜂窝电话塔等。在一些实施例中，外部验证器904执行(下文更详细描述的)初始计算并且使用设备902作为媒介向代理906传送(例如经由安全信道)信息的部分。代理906对单片内核执行的计算定时并且向外部验证器904回报定时测量。在一些实施例中，取代代理906或者处理代理906之外还使用外部设备，诸如有绳蜂窝电话或者计算机、基站或者附加外部验证器。也有可能使用被视为防篡改的软件代理或者使用专用硬件代理。

[0120] 图10图示了设备的一部分的实施例。如上文提到的那样，“自由”RAM定义为是在已经交换出所有应用和标准内核之后应当自由的RAM部分。总线的宽度是字。存储器大小也可用字为单位来描述。例如如图10中所示512字节存储器页面在标准手持社保上具有128个字的大小，其中字是32位。如这里使用的那样，“块”是高速缓存线的长度。在所示示例中，高速缓存线对应于8个字(每个字是32位)，并且块相应地是256位。

[0121] 图11图示了用于执行设备审核的过程的实施例。在各种实施例中，图11中所示过程由设备902执行。配置该过程使得它的计算有望在特定时间量内完成。对评估的自由RAM量的任何改变和对访问次级存储1004的任何尝试将造成计算完成所花费的时间量的可观测延长。类似地，改变任何列入白名单的程序或者关联数据的内容将引起延迟或者不正确响应的计算。

[0122] 可以用多种方式(诸如结合图3中所示过程描述的方式)发起图11中所示过程。作

为附加示例，可以在关机或者引导路线中包括审核过程。也有可能让应用发起审核过程。将解激活应用，将执行处理，并且将在完成时向应用移交回控制。在一些实施例中，应用向中央机构(或者设备)查询关于多久前执行最后扫描的信息。SIM卡可以存储关于何时执行扫描的信息。如果SIM卡具有允许它持续测量时间的功能，则它可以给出实际时间作为答案。否则，它可以基于所见事务(将已知其中许多事务为定期的)的数量给出时间估计。这样的信息可以用来评价从最后扫描起的持续时间。

[0123] 该过程在1102始于向次级存储1004交换除了单片内核1006之外的存储器1002的内容(以及它认为可接受保留的任何进程)时。在一些实施例中，该过程的部分1102包括交换出正常内核或者其部分。关键特征(诸如用于串行通信的设备驱动器)重新实施于单片内核1106中。在各种实施例中，逐字交换出内容，或者交换出或者在代理、外部验证器或者其他受信任设备上存储或者在处于不能用于活跃代码的状态中的RAM中存储对内容的紧凑描述。(例如有可能在未用于指令而是仅用于数据的高速缓存部分中存储指令)。在一些实施例中，无“自由”空间存在，并且省略图11中所示过程的部分1102。

[0124] 在1104，接收与硬件配置对应的一个或者多个硬件参数。该过程的这一部分与图3中所示过程的部分302相似。也在1104接收初始化信息(诸如可以用来生成伪随机串的种子)。初始化信息的其他示例包括如下文更详细描述的步进值和密钥值。

[0125] 在1106改写自由RAM。在一些实施例中，使用种子生成的伪随机串的输出用来改写自由RAM。一种用于改写自由RAM的技术是生成具有特定性质的n^2位伪随机串，该性质是计算输出位中的任何一位将花费与计算整个位块或者MD6在512位模式中的至少512个应用所花费的时间的至少一半那么长。该技术使用三个阶段并且重复(用不同aux值)直至输出串填充整个自由RAM：

1.生成：使用具有“n”位输出大小的散列函数h(例如MD6)针对$0 \le i \le n-1$生成值$x_i = h(seed, i, aux)$和某些值aux。这生成n^2个随机位。

[0126] 2.重排(shuffle)：计算$y_j = \prod_{i=0} 2^i BIT_j(x_i), 0 \le j \le n-1$，其中$BIT_j$是返回输入第j个最高有效位的函数。这以如下方式重排位，该方式需要计算所有n个散列函数应用以重组任何一个值。

[0127] 3.混合：针对$0 \le j \le n-1$计算$z_j = h(y_j)$。这保证输出的每位是所有n个输入位的函数，每个输入位需要一个散列函数赋值(evaluation)来计算。

[0128] 在各种实施例中，执行附加重排和混合以进一步增加计算最终串的任何部分的成本。此外，可以使用其他用于改写自由RAM的技术代替结合图11中所示过程的部分1106描述的示例技术。

[0129] 在1108以“步进”值确定的方式读取存储器1002(或者其部分)。累加结果并且使用密钥来密钥化计算。在各种实施例中，部分1108的处理由存储器访问调度器和累加器执行，现在将更详细描述其中每个。

[0130] 存储器访问调度器

令“sRAM”为RAM 1002的以块为单位的按其整体测量的大小。外部验证器904将在页面<步进<sRAM-页面范围中选择随机步进，使得“步进”是奇数值。这里，“页面”表示也以块为单位测量的在次级存储中的一个存储器页面的大小。在有若干页面大小的情况下(例如如果

有构成次级存储的若干部件)，则在各种实施例中使用页面大小中的最大页面大小。

[0131] 执行1108的处理包括如下循环，在该循环中访问存储器并且组合结果以形成密钥化存储器校验和。对于循环的每次迭代，将访问位置增加步进值模sRAM。由于“步进”和sRAM互质，所以将访问所有RAM存储器位置确切一次。另外，访问顺序将不为敌对方所知直至公开值“步进”。在图12中提供根据“步进”读取存储器1002的图示。

[0132] 在图9中所示示例中，设备902包括单个单核CPU。在诸如包括多核处理器和/或多个处理器的膝上型计算机的系统中，可以用如下方式构造1108的处理，该方式固有地串行(并且因此将妨碍使用多个处理器)或者适于利用多个处理器。作为后者的一个示例，可以用偏移开始若干计算，使得每个线程对应于存储器的不同部分并且其中无冲突。

[0133] 累加器

可以使用简单非线性函数在寄存器中累加存储器内容，该函数逐个组合先前寄存器内容(这里称为“状态”)与新读取的存储器内容(数据)。累加函数的示例包括散列函数(例如MD6)、非线性往回移位(shift-back)寄存器以及更简单的函数。

[0134] 更简单函数的一个示例是状态←ROR(状态+数据)。后一个函数对应于函数ROR(…(ROR(状态$_0$+数据$_1$)+数据$_2$)…+数据$_n$)，其中“+”是指普通加法，并且“ROR”将寄存器的内容向右旋转一位。在这一情况下，函数本身可能不是非线性，但是当与先验未知步进大小和严密定时要求组合时，它却足以满足所需处理要求。

[0135] 如上文提到的那样，在各种实施例中，密钥化累加过程。一种用于实现这一点的方式是以规律间隔用新值 “密钥”(从外部验证器或者代理获得的)偏移值“状态”。可以通过将当前值状态与新值密钥相加来执行偏移。

[0136] 另外，尽管结合1108描述的过程基于读取存储器，但是在一些实施例中，包括写入操作以引起进一步基于闪存的减速。作为一个示例，如果数据存储于闪存中，当引起擦除整个块时则写入“1”的序列。为了简化对何处写入的调度(并且单片内核利用它)，可以在获得新密钥值的同时从代理获得位置。

[0137] 也可以在1108执行各种其他存储器访问序列。例如有可能使用两个步进值而不是一个，其中这些步进值可以均为偶数，但是其中它们主要引起覆盖所有空间。也有可能使用对选择位置序列的函数进行确定的数字或者参数汇集。有可能将这视为最大长度序列，其中输出是位置，并且最大长度序列包括与存储器位置对应的在给定的范围中的所有值。如果希望则有可能偏移这样的值以避免访问某些区域(例如单片内核)。在最大长度序列的情况下，外部验证器或者代理提供的密钥可以是初始状态或者是与LFSR的各种单元关联的权值。

[0138] 在1110向外部验证器904提供密钥化计算。如果外部验证器批准结果，则认为设备902在安全状态中。

[0139] 在1112，设备902执行将在安全状态中执行的任何函数。示例包括针对恶意/不希望程序建立SSL连接、播送投票、录入口令、扫描次级存储等。在各种实施例中，如果安全状态函数的代码在次级存储中(即它并非单片内核的部分)，则比较函数的摘要与单片内核中(或者代理上)存储的值。如果值匹配则才激活函数。在各种实施例中，如果代理可以执行消息摘要计算，则单片内核无需包含用于这样做的代码。

[0140] 在1114通过加载次级存储1004的(在1102交换出的)内容来恢复RAM 1002的状态。

[0141] 图1中所示过程的潜在负荷的大部分涉及到从RAM向次级存储交换出应用和数据并且将它交换回。有可能避免这样做以便例如节省时间。这可以通过取消应用来完成。如果外部验证器或者其他资源知道什么应用在运行并且潜在也知道它们或者其部分的状态，则有可能让这一方在审核过程已经运行之后辅助重启所选应用。有可能让次级存储或者SIM卡或者其他板上单元维护这一信息中的一些信息。有可能不通过应用和数据的全串而是通过更短标识符标识它们以节省空间和时间。有可能具有主要在检测算法已经运行之后再生相同状态的近似算法。例如这可以重启浏览器、但是无法恢复浏览器内容。

[0142] 另外，如果活跃应用仅占用RAM的某些部分，则不必交换出它们。例如假设它们仅占用RAM的下一半。例如自由RAM中的每个单元(编号i)将该单元的内容复制到向上更高位置(位置2i)。这优选地从末尾开始执行(更高编号位置)。这有效切分应用并且使它们仅驻留于偶数位置。现在伪随机值仅需写入到奇数编号位置中并且仅需执行奇数编号单元的非线性累加。注意不可能让任何工作恶意软件保持活跃。有可能让恶意软件存在，但是仅在它的指令是“向下一开放空间跳转”并且那是下一指令所在空间时。由于伪随机材料未改写的所有空间将仅为跳转(无空间用于连续空间中的更多空间)，则可知恶意软件不能实现任何事情。尤其如果空间不可由敌对方预测，则也有可能使分片之间的距离更大。该距离例如可以由序列生成器预测，其中不同距离是不同长度。在RAM内展开数据和程序可以与用随机串偏移这些数据和程序组合。微内核(负责恶意软件检测的程序)——将不以该方式展开，因为它需要保持于允许它执行的状态。

[0143] 结合图11，关于如何可以使用并非先验已知的步进大小来读取RAM做出描述。也有可能使用除了一之外的步进大小来写入(自由)RAM，其中步进大小可以是先验未知的。这一点的一个效果在于它针对希望在次级存储中存储信息的攻击者引起向次级存储写入。如果次级存储使用闪存，则这些延迟大于读取延迟。有可能使用以与待写入的范围对应简单步进增量模整数——如果单片内核未驻留于高地址则加上偏移。也有可能未将特定模数用于算术—这对应于使用与可寻址RAM空间对应的模数———并且标识待写入的单元是否在不应写入的范围中。

[0144] 图13图示了用于选择性地读取存储器的过程的实现的实施例。在一些实施例中，图13用来实施图11中所示过程的部分1108的一部分。

[0145] 在一些实施例中，进行对数据高速缓存的全利用。具体而言，按照高速缓存线的宽度(该宽度的大小通常范围从8到512字节)以块为单位处理数据。在下文中，参数“α”表示高速缓存线的以字为单位的宽度。参数“周期”对应于定时周期的长度。将这设置成对应于比从设备902到潜在外部帮助者的乐观往返延时更短的时间。报告的值和报告它们的时间由代理906记录。

[0146] 检测各种规避程序的示例

下节提供如何可以使用这里描述的技术来检测运用上述各种策略的规避程序的示例。为了成功规避检测，敌对方必须在预计时间 $t_1(F_c,x)+\delta$ 内提供 $F_{c'}(x)$。有敌对方可以尝试这一点的各种方式，现在将描述每种方式。

[0147] 防范敌对策略1——外部支持。

[0148] 通过安全信道向代理传送值“密钥”和“步进”。针对外循环的每次迭代报告“状态”

值。这一值的计算依赖于值“密钥”。因此，如果(从发出值密钥到接收计算的结果)往返延时超过时间阈值$t_i(F_\varepsilon,x)+\delta$(其中是$t_i(F_\varepsilon,x)$是用于$F_\varepsilon$生成响应的时间)，则不可能让外部设备帮助生成响应。

[0149] 防范敌对策略2——使用次级存储。

[0150] 假设敌对方已经修改RAM中的一些“k”个数据字并且这一区域的“既定”内容的映像保持于次级存储中。当F_ε将尝试访问修改的单元之一时，F’代之以访问次级存储中的映像。

[0151] 这可以用三种方式来完成：(a)整个RAM可以在次级存储中被映像，并且在此进行每次访问；(b)F_ε的每次存储器访问替换为比较地址与范围而所选访问尝试仅向次级存储重新路由；以及(c)转译后备缓冲器(TLB)用来将具有所选范围的访问自动重新路由。

[0152] 在第一情况下，每次存储器访问将遭受与访问次级存储而不是RAM关联的延迟。在第二情况下，仅重新路由的访问遭受这一延迟，但是对于每次存储器访问必须做出比较。在第三情况下，仅有针对重新路由的访问施加的延迟。然而由于TLB的有些粗略的粒度，所以有最小数量的如下单元，可以针对这些单元重新路由访问。

[0153] 防范敌对策略3——使用RAM

由于构造用来填充自由RAM的伪随机串，所以计算任何一个单元将需要计算至少512个散列函数赋值，由于每个单元中的值(在一些实施例中)是第二层散列函数输出的一部分，该函数的输入是512个第一级散列函数赋值的函数。

[0154] 假设有用于每次散列函数计算的设置成本并且这是至少1156个时钟周期。同时，用于MD6-512的散列计算——一旦完成设置——花费每字节155个周期或者用于64位输入的9920个周期。这二者均假设优化代码并且针对典型32位处理器。在这一设置中的每个散列函数赋值的总成本因此是11076个周期。计算RAM中的仅一个单元的内容将花费至少$(512+1)*11076$个周期。

[0155] 将代理用于定时

如上文提到的那样，在一些实施例中，外部验证器用来分发新种子并且对来自定时检查点的值的到达定时。在审核的设备与外部验证器之间的示例通信模型是：(a)物理连接；以及(b)在外部验证器与SIM卡之间的VPN。加密向下数据、认证向上数据和/或出现双向认证。

[0156] 当物理连接未使用于设备与验证器之间时，延时差异可以阻挠对计算的定时。因而在一些实施例中，代理(具有更少延时差异)用来支持定时。可以用作代理的一款示例硬件是SIM卡。

[0157] 代理可以用来减少与计算的开始关联的差异。这用于于整个计算任务并且应用于子任务。可以通过发送计算所需要的密钥或者种子来发起任务和子任务，从而确保计算不开始直至接收密钥或者种子。

[0158] SIM卡接收加密种子、解密它们并且向设备902提供值。可以使这一配发(dispensal)时间相对于在设备上观测的其他事件。考虑三元组(位置，数据，种子)并且假设验证器产生这样的三元组，并且以加密和认证方式向SIM卡发送它们。这里，位置描述程序内的计算步骤(或者阶段)，并且数据描述与这一位置关联的某些状态。假设外部验证器

可以预期计算或者可以预期少量很可能的计算路径。

[0159] 这使得有可能让外部验证器计算这些三元组，从而预测什么数据(或者状态)将在给定位置(或者阶段)在设备上可观测。三元组的第三元素，种子，表示如果设备到达与位置和数据关联的给定状态则向它配发的值。

[0160] 除非“值”位置可以由设备计算或者预测或者从别处接收，否则SIM卡将向设备发送它。当计算已经到达与值位置(这可以是循环迭代值)关联的阶段时，设备上的软件向SIM卡发送某些预定类型的最近计算的值——或者其部分。这可以是最近计算的值、给定的寄存器的内容或者任何其他适当值。SIM卡比较这一个值和与位置关联的值数据，并且如果它们相同或者等效，则将以种子值做出响应。设备将它的当前种子值替换为用于种子的这一新值。这允许设备在它已经计算某些值时替换种子，该值保证它已经到达给定的计算阶段。此前未公开新种子，这具有安全益处。替代地，一旦报告新结果，无论这一报告值是否正确都可以公开新种子；将把任何报告值与接收它的时间一起记录，从而允许验证器以后确定是否接收不正确结果或者是否有任何明显延迟。可以如适用的那样压缩或者概括代理发送的报告以便节省带宽。

[0161] 一旦做出匹配并且配发关联种子值，计算已经到达与三元组(位置，数据，种子)关联的给定检查点。

[0162] 在一些实施例中，SIM卡具有这样的三元组的长矢量并且选择线中的第一三元组(在直接计算中)或者与输入匹配的任何一个三元组(在有外部验证器单元不可预见的分支的情况下)。如果使用可预测位置，则有可能省去值位置。SIM卡可以获得长的三元组列表而不可能让这些三元组中的任何三元组由手持社保或者其他方截获；这通过使用在外部验证器(或者关联代理)与SIM卡之间的点到点加密来变得有可能。

[0163] 也有可能使用验证器发起的事件而不是值数据。因此一旦SIM卡观测到事件，就释放关联种子值，并且用这一值开始设备计算。这可以用来开始钟控。这也可以由必须呈现给定条数据的设备实施，该数据可以例如由外部验证器向手持设备递送以便开始计算。

[0164] 有可能一些种子值为空，在该情况下未向设备发送种子值。这可以用来关于手持设备已经到达通信的什么点执行校验。

[0165] SIM卡不必被钟控(并且因此知道时间)，并且它可以操作于敌对环境中，(只要值“数据”充分长以排除手持设备上的恶意代码的穷尽搜索，这是以用过早方式获得种子值为目标的。

[0166] 可以如上文所述通过具有检查点来标识计算的结束，此后SIM卡向外部验证器报告计算的时间或者报告如下信息，该信息允许外部验证器计算对在计算的开始与结束之间的时间的估计。在前一种情况下，本地时间值的汇集可以由SIM卡记录，一个时间值用于到达的每个检查点。这可以记录为(位置，时间)的列表，其中值时间可以是无需与外部验证器的时间概念同步的本地时间。替代地，可以记录报告值和关联时间对。SIM卡可以认证这一列表并且向外部验证器传输它，该验证器将验证认证正确，此后它可以做出对各种计算任务花费多长时间的确定。这可以用来推断设备的安全状况。替代地，SIM卡可以做出这一安全确定并且向外部验证器发送该确定的认证结果。

[0167] 即使无能力告知时间，SIM卡仍然可以对事件排序。在它从验证器(或者与验证器而不是与恶意软件协作的任何实体)接收分组时，它将确定最近发生设备报告的什么事件。

这对应于接收的并且验证为正确的值“数据”。因此值数据可以由设备生成(在外部验证器设置或者已知给定的某些计算任务时)或者可以是来自外部验证器的通信结果。

[0168] 为了应对向SIM卡的有意延迟的报告,一旦接收事件分组,就有可能需要SIM卡向外部验证器的立即确认。这些消息被认证和/或具有恶意代理不能预期的格式。一种示例方式是通过让上述值“数据”对应于来自外部验证器的事件来构造这一消息并且一旦从SIM卡接收关联值“种子”就让设备向外部验证器报告它。外部验证器然后可以在先前分组被发送之后对这一确认的到达定时。替代地,可以允许设备向外部验证器直接报告与“数据”对应的值。这可以针对所有检查点的某些部分或者针对附加检查点来完成。

[0169] 假设串S标识的分组在事件E发生之后、但是在任何其他事件发生之前由SIM卡接收(并且由SIM卡正确验证)。然后向日志L添加对(S,E)。替代地,向日志添加标识串S和E的部分。在通信结束时向验证器传送日志。替代地,向外部验证器传送排序信息。

[0170] 在一些实施例中,向矢量添加三元组(位置,数据,报告),其中“报告”是指示向验证器发送报告的值。这也可以使用普通种子值来实现,其中释放的最后种子值是设备为了停止定时而向验证器传送的值。也可以通过让设备执行某些附加密钥化任务并且在交换种子中回报值来停止定时,在该情况下具有串S的下一分组的到达将标识定时何时停止(根据在既定计算结束之后表观计算的周期数量来推断)。

[0171] 一旦定时已经结束,SIM卡加密(并且潜在认证)日志并且向设备传递这一日志用于向验证器传输。验证器确定它是否被正确认证、解密它并且然后在给定与包含值(诸如S)的分组的传输时间有关的信息时根据日志确定通信的部分步骤的完成时间是什么。它可以知道这些分组由与手持设备最近的基站传输的时间或者可以知道它们何时由网络代理或者由分组本身的始发方处理。

[0172] 在一些实施例中,值S由受信任信标生成并且在外部验证器在设备上开始计算之前潜在地不为它所知。

[0173] 一些类型的分组(诸如包含S的分组)在它们通过网络传输之前无需加密。在一些实施例中,认证它们。然而SIM卡对分组是否具有正确认证的验证无需在编译预备日志条目之前完成。如果认证验证对于已经编译的日志条目而言失败,则可以擦除这一日志条目。

[0174] SIM卡是半双工(即不能同时接收和发送数据)。SIM卡作为从属设备来操作,即仅将在已经被请求向附着设备(我们的手持设备)发送数据之后才这样做(有一些特殊例外,诸如当它们上电时)。然而一些智能卡可以在来自它们的关联手持设备的查询之间独立操作而并非仅作为对被查询的反应来操作。

[0175] 如果SIM卡支持多线程,则有可能让一个线程执行简单计数(在定时开始时开始)并且向其他线程提供这一计数器,该其他线程每当接收正确数据值时记录这一个值。计数器可以连同数据值一起存储关联位置或者索引,该索引指示什么数据值与计数器关联。在一些情形中,诸如如果确保一个数据值不能被接受多于一次并且有待接收的值的确定顺序,则我们可以记录计数器值而并非数据值或者其他状态信息。

[0176] 一些SIM卡(诸如典型Java卡)仅在从SIM卡对接设备(CAD)接收消息之后支持计算。该计算通常在生成并且向CAD传输来自SIM卡的响应时结束。

[0177] 如果SIM卡即使在从手持设备接收消息之前和在传输响应之后仍然允许针对每个时钟周期(或者另一确定周期)增加计数器,则即使未支持多线程仍然有可能维持恰当计

数。

[0178] 有可能在SIM卡中保持基于时间的状态;也有可能让SIM卡认证事件(包括它们发生的时间)并且导出这样的已认证事件的列表。

[0179] 图14图示了用于对设备审核的一部分定时的过程的实现的实施例。在所示示例中,修改的Java卡用作代理。该修改允许该过程在对请求做出响应之后保持活跃。代理从外部验证器接收值输入和输出的矢量并且产生在执行完成时向外部验证器传输的矢量持续时间。(假设认证和加密在代理和外部验证器之间的所有通信)。在所示示例中,值“∞”对应于指示客户端上的恶意软件代理尝试欺骗的错误消息。在接收时,外部验证器将确定矢量持续时间中的值是否都落入严密界限内(这暗示成功完成)并且如果这一点成立则才将推断客户端在安全状态中。

[0180] 将代理用于定时的附加方式

除了使用SIM或者相似一款硬件作为代理之外,也有可能使用视为在安全状态中的另一设备作为代理。有可能通过先让一个设备(诸如电话)验证为安全来引导安全性,并且然后使用该设备执行本地定时和在验证第二设备的安全性的过程中的其他验证任务。

[0181] 设备可以是不同类型。例如一个可以使用SIM卡、本地蜂窝电话塔或者本地计算机在外部验证器请求时辅助对手持设备的安全验证。有可能让这一外部验证器已经预备在验证中使用的数据的部分,诸如上文描述的(数据,种子)。也有可能让第三方执行这一预先计算。在各种实施例中,种子由代理生成。一个示例是让代理和设备均包含传感器(诸如加速度计或者光电传感器)。设备和代理均观测相同现象(例如通过保持在一起并且一起摇晃)并且计算种子方式。在这一情况下,设备使用观测的种子,并且代理向外部验证器发送(它独立体验的)种子。

[0182] 只要确定第一设备在安全状态中,它就可以用来对第二设备(诸如汽车的信息娱乐系统、另一手持设备、笔记本、膝上型或者桌面型计算机或者其他设备)的安全评价定时或者否则辅助该安全评价。这一安全评价可以是相同类型(例如基于用于执行计算任务的时间),或者它可以是在第一设备上引导的替选安全评价方法。类似地,第一安全评价可以是不同类型而以后可以在第一设备上引导安全评价并且可以使用我们基于定时的方式。

[0183] 在一些实施例中,第一已知“安全”设备用来产生将在以后点由其他设备消耗的种子值集合的汇集。有可能让这样的集合被认证(例如使用PKI或者通过基于对等的值认证)。另外,如果外部验证器支持在线代理,则有可能让外部验证器以离线方式操作。例如外部验证器可以在审核过程之前很久的时间发送加密和认证的数据;可以一次发送若干这样的副本(transcript)。它们可以发送到代理,可以在该代理保持它们;或者它们可以发送到审核的设备或者其代理,可以在该设备或者代理保持它们直至需要它们。代理由于存储器打印而生成的副本也可以由设备缓冲并且以后无论是在请求它们时还是在有通信信道可用时发送到外部验证器。在一些实施例中,所有记录在它们被认证并且可能加密之前用时间戳和系列号标记。

[0184] 有可能将这实施于小节点的网络中,其中一些节点先验受信任或者被评价为安全;此后这些节点用来辅助其他节点的安全评价。这是潜在递归方式并且可以是循环的,即用来评价其他设备的安全性的先前受信任设备可以以后由这些设备中的一些设备或者以其他方式评价为安全的设备验证。外部验证器仍然可以包含于该环境中并且可以帮助启动

验证事件链，并且帮助调度什么节点应当何时以及由谁验证。

[0185] 使用压缩的访问表

在一些实施例中，存储器访问的位置由矢量“位置”的内容确定，该矢量的内容对应于自由RAM的所有单元的排列。如果保持在那里，则这一矢量可以占用所有自由RAM。在一些实施例中，它存储于次级存储(诸如闪存)中，并且按照需要换入部分。现在将描述如下替选方式，该方式维持伪随机访问顺序、但是使在主循环期间最小化计算工作量。

[0186] 考虑locationH和locationL这两个矢量，其中两个均为矢量，每个矢量包含部分存储器访问位置的排列。这里，实际存储器访问位置是两个部分位置的组合，例如一个locationH元素的位与一个locationL元素的位连结。这里假设locationH元素包含更高阶位而locationL元素包含更低接位。这些元素可以是相同或者不同大小、但是在组合时将是对一个存储器位置寻址的大小。如果每个包含范围中的所有可能元素，则所有组合的汇集将对应于所有存储器地址的汇集。(可以通过比较组合结果与阈值从这一汇集去除未在自由RAM中的那些并且如果结果落在这一阈值以下则丢弃它)。这一表示仅占用为了存储而寻址的空间大小的平方根。有可能使用三个分量，在该情况下它们占用寻址空间的立方根。有可能也使用大量分量。一个示例组合函数是连结。

[0187] 在一些实施例中，矢量元的访问顺序遵循几何图案，该图案确保将以巨大可能性使用所有组合。在给定增加存储器访问图案时，没有对一个矢量内的相同一项的若干次访问可以是有益的，因为这减少对于敌对方而言不可预测的程度。有可能覆盖一个组合多于一次而概率可忽略不计，但是在确保进行所有访问的同时限制对存储器的访问总次数是有益的。

[0188] 有可能访问在位置x的locationH矢量和在位置y的locationL矢量，并且访问沿着对角线的x-y位置。这里可以在位置(x,y)=(0,0)开始第一序列，此后针对循环的每次迭代将x和y均同时加一。当增加一个坐标超出矢量的大小时，将该坐标再次设置成0。然后 当位置再次变成(0,0)时，可以修改它为在位置(x,y)=(1,0)开始，此后重复增量序列直至它回到(1,0)，这时将它改变成(2,0)。这并非存储器访问的位置：它是对何处进行存储器访问进行描述的矢量中的位置。

[0189] 也有可能通过具有位置元素矢量(其中每个这样的位置仅描述地址的部分而以另一方式计算或者根据在计算时的程序状态来推断地址的其余位)来否则压缩对访问什么单元的描述。另外，这两种方式可以组合并且与至少部分预先生成的对访问位置的更多其他有关描述组合。

[0190] 关于定时的附加信息

在各种计算中，对计算的定时和部分计算出现如下。(A)一旦向审核器提供所有必需值(无论是来自外部验证器还是其代理)，就启动定时器。这些值通常包括种子值。(B)当审核器向外部验证器或者其代理提交正确值“状态”时停止定时器(并且记录从它启动起的时间)。

[0191] 有可能在旧时间间隔已经结束时立即开始新时间间隔(其中开始由上述步骤A表示而结束由上述步骤B表示)。也有可能在这些间隔之间实施“休息”；在这些休息期间，可以不对计算定时，并且算法可以执行例行维护(诸如与外部各方通信、读取或者向次级存储写入或者其他功能)。可以在算法请求开始下一定时间隔(例如步骤A)时结束休息；一种可以

这样做的方式是通过用信令通知外部验证器或者其代理开始下一间隔;或者它可以由选择开始新间隔的外部验证器或者其代理完成。

[0192] 也有可能实施休息作为标准定时间隔,这些间隔的长度对于最终确定审核的设备的安全状况而言无足轻重。

[0193] 伪随机访问

在一些实施例中,利用对访问位置的读取和写入序列通过伪随机顺序的访问来实现作为审核过程的部分而执行的选择性的读取。现在描述使用伪随机访问的替代实施例。首先将提供对存储器填充示例的描述。然后将提供对定期定时示例的描述。

[0194] 填充快速存储器

以下存储器打印功能可以用来填充自由RAM。在其他类型的存储器在访问时间方面可与RAM比较的情况下,它也可以用来填充这样的其他类型的快速存储器。以伪随机顺序向自由RAM中异或伪随机序列;以后计算RAM的全部内容的密钥化散列。即使RAM未使用块和页面,它也可以划分成与闪存的块和页面对应的"虚拟"块和页面。未在页面或者块中访问闪存的连续块。这使访问在闪存中缓慢、但是在RAM中仍然快速。

[0195] 为了向自由RAM填充伪随机串,有两个主要步骤。第一,运行设置功能。这使用从验证器获得的种子生成伪随机值来确定存储器打印功能将进行的存储器访问的随机顺序。表存储于闪存中,并且设置功能使用的程序空间在设置完成之后清空。第二,存储器打印功能用来填充所有自由RAM。从开始到结束并且在更短间隔中对它的执行定时。

[0196] 处置网络延迟

可以从如下设备测量传染所引起的延迟,该设备通过内部接线、标准网络端口(诸如USB)、通过有线接口、通过WiFi网络、通过LAN、通过因特网、通过分组交换网络、通过通信网络或者其组合来连接到客户端设备。这些通信介质中的一些通信介质可能引入可以使用统计方法从测量中分离的延迟和差异。

[0197] 验证由如下设备做出,该设备使用线缆、LAN、WAN、蓝牙、Wifi、因特网、另一网络或者网络组合来连接到审核的设备。通过比较接收的结果与计算的结果并且验证是否在恰当时间界限内接收它(和在它之前的序列)来做出验证。所有这些通信介质可能带来延时,并且一些通信介质可能漏掉分组。

[0198] 暂时假设"良好"事件占用10个时间单位加上在1与5个时间单位之间的时间(鉴于网络差异)。

[0199] 然后假设"坏"事件占用15个时间单位加上鉴于网络差异的1到5个时间单位。

[0200] 考虑在这些时间的部分结果的接收:

序列a:0,12,25,(遗漏分组),50——这一序列尽管有遗漏分组单仍然可能良好,因为最后部分结果"保证"丢失的分组。

[0201] 序列b:0,11,30,35,50——这一序列尽管有在第二与第三分组之间的长延迟仍然可能良好,因为"太早"接收第四分组。

[0202] 序列c:0,11,30,45,57——这一序列由于在第二分组之后的长延迟并且无说明该延迟的事件而可能是坏的。

[0203] 虽然已经出于理解清楚的目的而以一些细节描述前述实施例,但是本发明并不限于提供的细节。有实施本发明的多个替选方式。公开的实施例为示例而非限制。

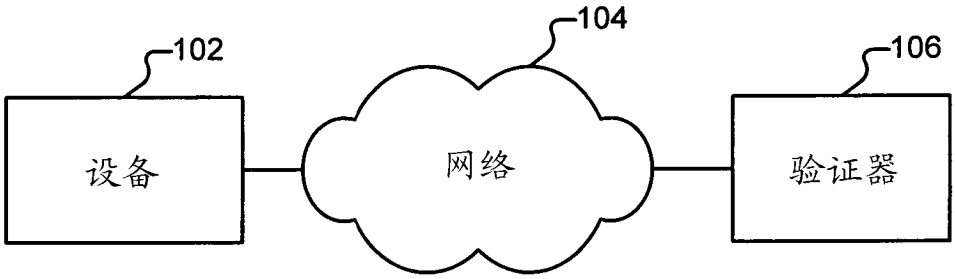

102
104
106
设备
网络
验证器

图 1

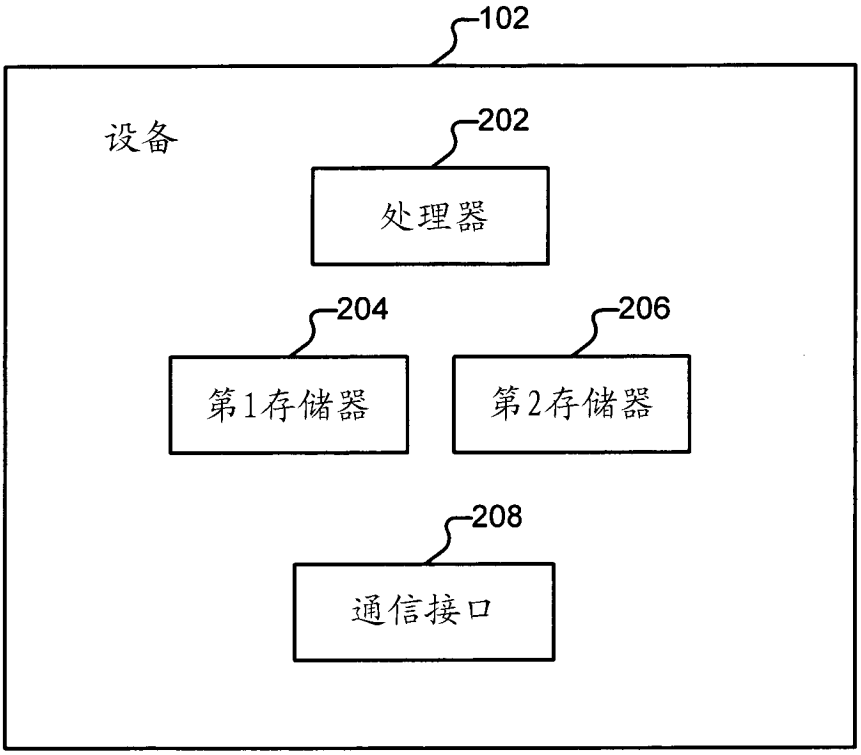

102
设备
202
处理器
204
206
第1存储器
第2存储器
208
通信接口

图 2

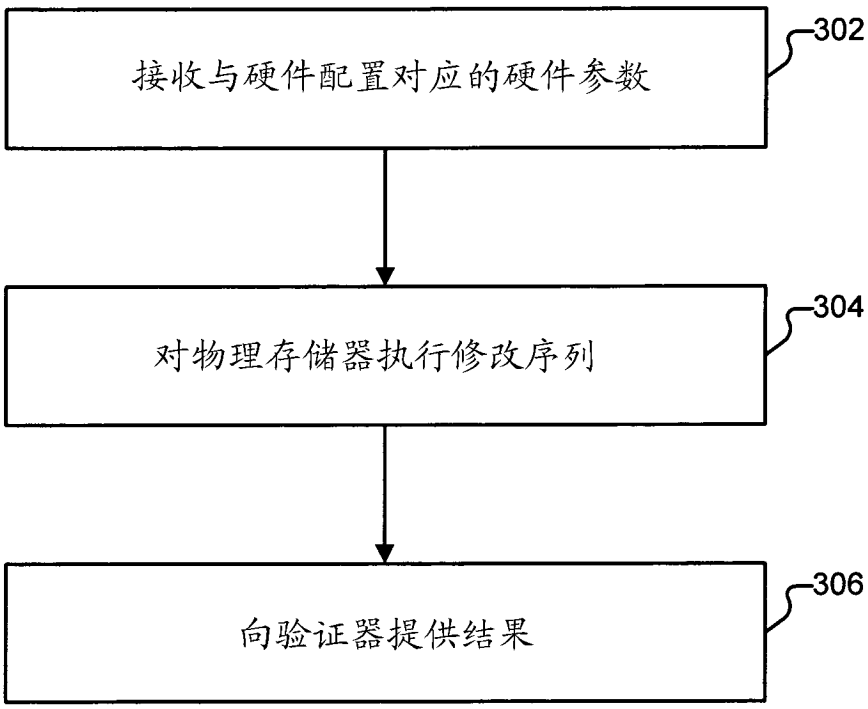

接收与硬件配置对应的硬件参数
302
对物理存储器执行修改序列
304
向验证器提供结果
306

图 3

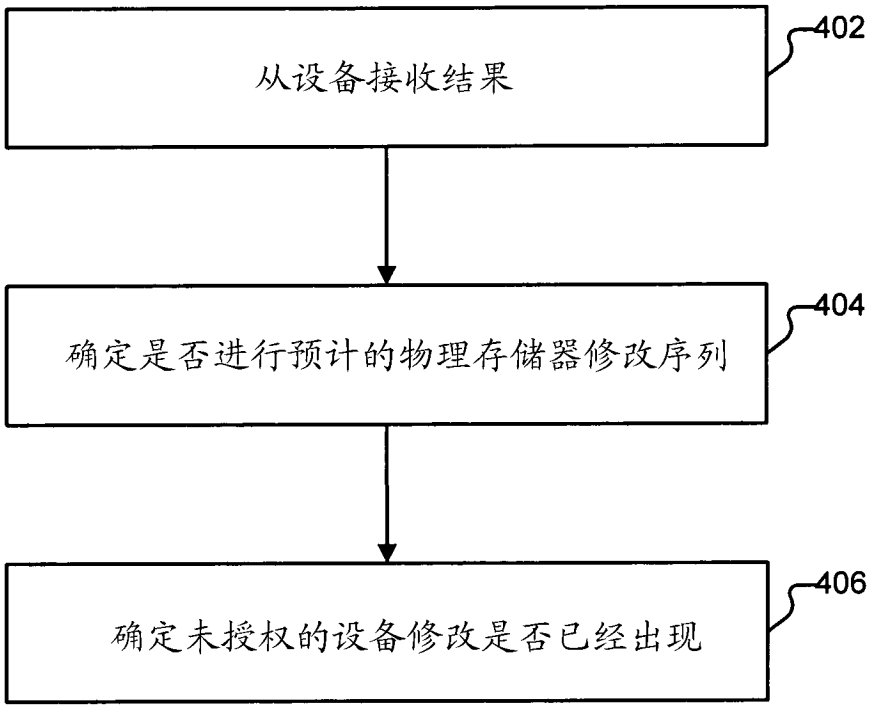

从设备接收结果
402
确定是否进行预计的物理存储器修改序列
404
确定未授权的设备修改是否已经出现
406

图 4

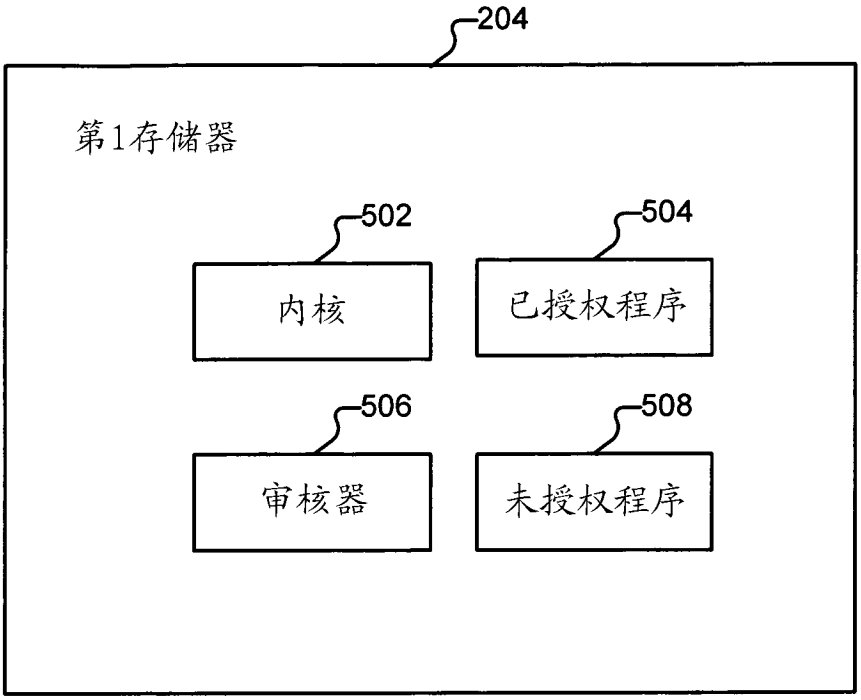


图 5A

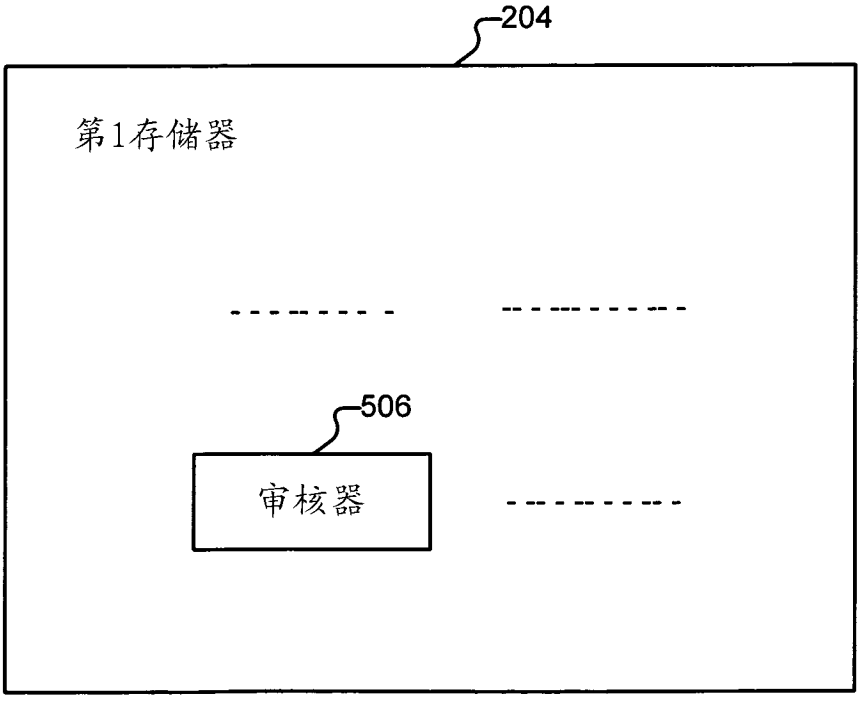


图 5B

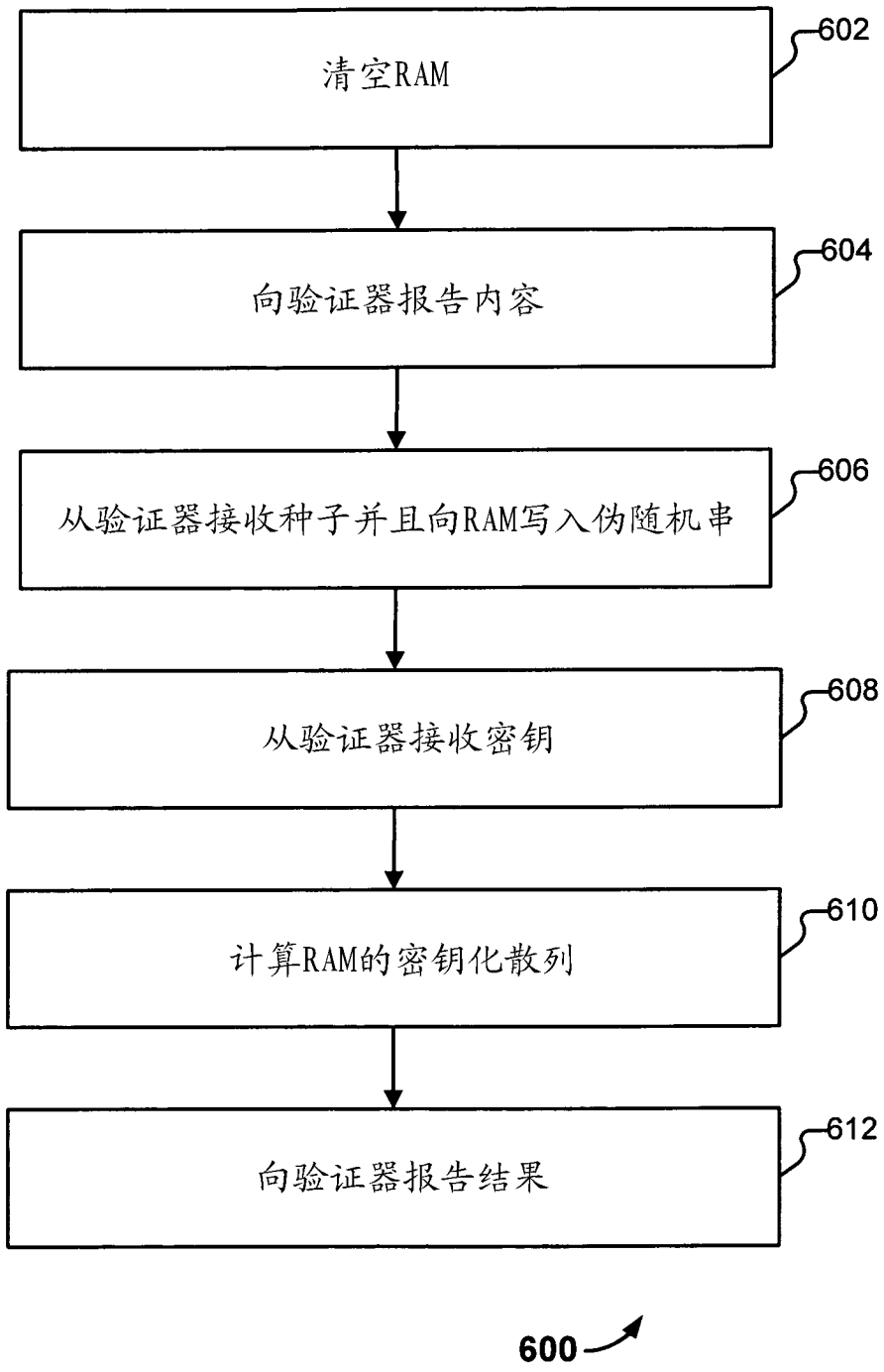


图 6

```
for i:=1 to rounds
  for k:=0 to chunks_per_block -1
    permutation[1  number_blocks] := get_permutation
    for j:= 1 to number_blocks
      modify_memory(permutation[j]*chunks_per_block+k, next_string_chunk)
```

图 7

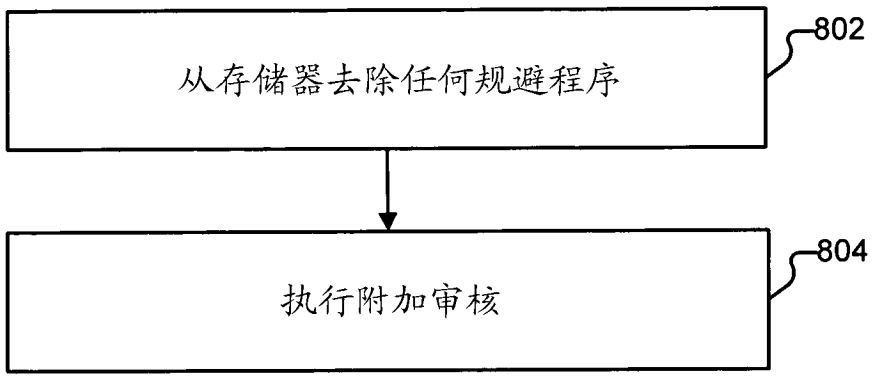


图 8

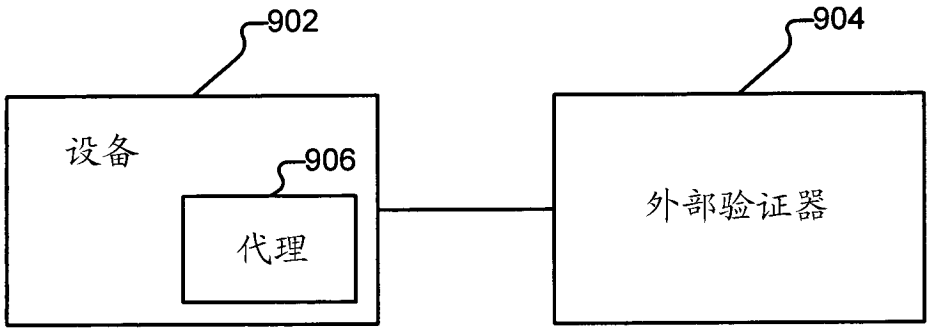


图 9

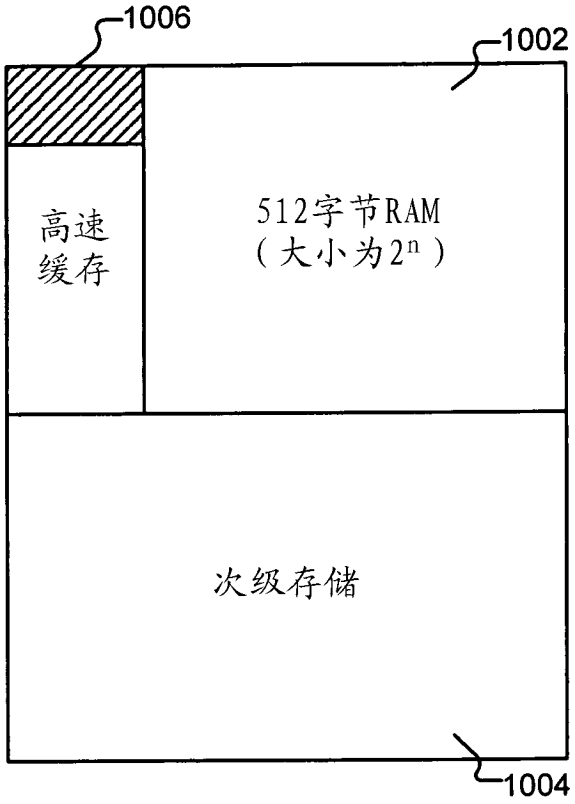


图 10

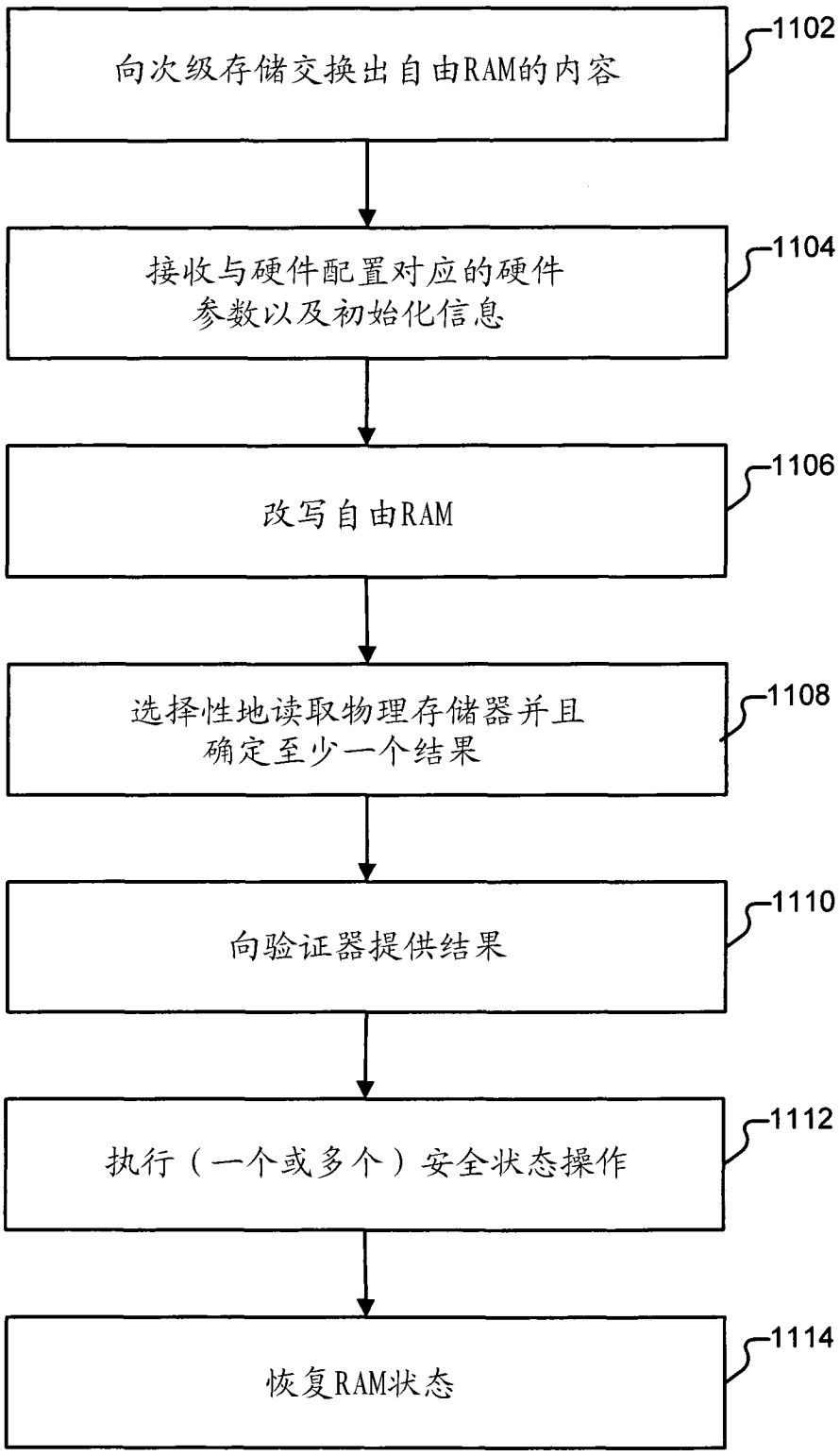

向次级存储交换出自由RAM的内容
1102
接收与硬件配置对应的硬件参数以及初始化信息
1104
改写自由RAM
1106
选择性地读取物理存储器并且确定至少一个结果
1108
向验证器提供结果
1110
执行（一个或多个）安全状态操作
1112
恢复RAM状态
1114

图 11

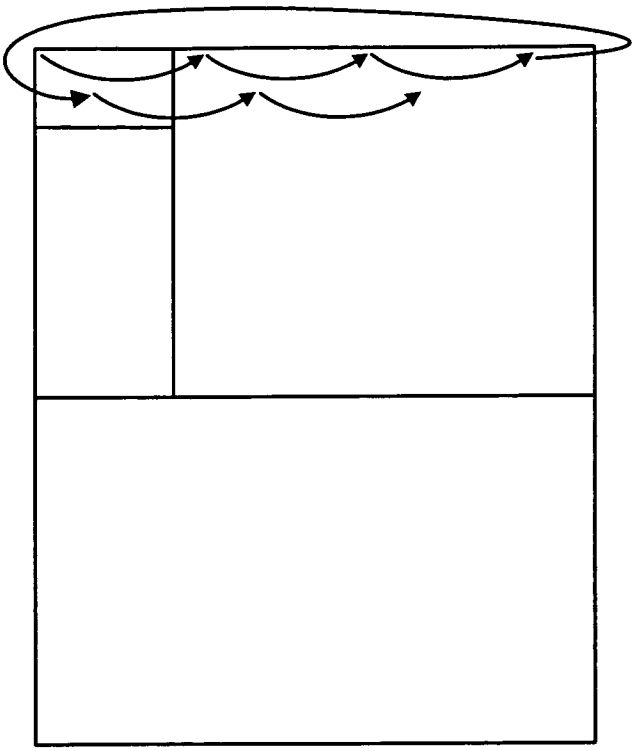


图 12

```
state ← 0
location ← 0
repeat (s_RAM/period/a) times:
    receive value key
    state ← state XOR key
    repeat period times:
        data ← RAM[location]                    %读取高速缓存线
        state ← ROR(state XOR data)             %累加
        data ← RAM[location + 1]                %从高速缓存读取
        state ← ROR(state XOR data)             %累加
        ...
        data ← RAM[location + a − 1]            %累加
        state ← ROR(state XOR data)             %从高速缓存读取
        location ← (location + step) mod s_RAM  %下一位置?
    report value state
```

图 13

```
counter ← 0
time ← 0
repeat period times:
    counter ← counter + 1
    repeat until value state is received:
        time ← time + 1
        if time = maxtime then
            duration(counter) ← ⊥
            counter ← counter + 1
    if state = input(counter) then
        duration(counter) ← time
        respond with value output(counter)
    else
        duration(counter) ← ⊥
```

图 14