(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 102880826 A
(43)申请公布日 2013.01.16

(21)申请号 201210311760.5

(22)申请日 2012.08.29

(71)申请人 华南理工大学
地址 510640 广东省广州市天河区五山路 381 号

(72)发明人 刘发贵 王亮明 张浩 熊智

(74)专利代理机构 广州粤高专利商标代理有限公司 44102
代理人 邱奕才

(51)Int.Cl.
G06F 21/50(2013.01)
H04L 29/06(2006.01)
H04L 29/08(2006.01)

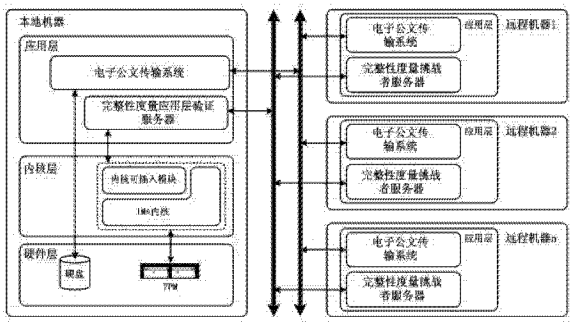
权利要求书 1 页 说明书 8 页 附图 2 页

(54)发明名称

一种电子政务云平台安全性的动态完整性度量方法

(57)摘要

本发明提供一种动态度量、安全性好的电子政务云平台安全性的动态完整性度量方法，包括完整性度量架构 IMA，针对指定进程和指定文件，分别在内核里设有双向链表，保存进程和文件的度量结果，TCM 芯片将度量结果扩展到相应的 PCR 中。这里所说的“指定”，是指需要度量的文件由权威部门确定，比如政府部门就需要对某些公文进行严格保护；度量指定的进程主要是提高计算机系统的安全性，辅助保护机密公文的完整性。



CN 102880826 A

1. 一种电子政务云平台安全性的动态完整性度量方法，包括完整性度量架构 IMA，其特征是，针对指定进程和指定文件，分别在内核里设有双向链表，保存进程和文件的度量结果，TCM 芯片将度量结果扩展到相应的 PCR 中。

2. 根据权利要求 1 所述的动态完整性度量方法，其特征是，所述文件度量结果扩展到第 10 号 PCR，进程度量结果扩展到第 11 号 PCR。

3. 根据权利要求 1 所述的动态完整性度量方法，其特征是，所述进程和文件的度量结果以日志形式保存。

4. 根据权利要求 1 所述的动态完整性度量方法，其特征是，所述方法需要完整性度量应用层验证服务器以及内核层的服务度量模块，所述完整性度量应用层验证服务器为指定文件和指定进程的度量模块，所述服务度量模块包括可插入模块实现的字符设备和修改 Linux 内核原有的完整性度量体系。

5. 根据权利要求 1 所述的动态完整性度量方法，其特征是，所述方法设有实时远程证明机制。

6. 根据权利要求 1 所述的动态完整性度量方法，其特征是，所述方法设有实时报告机制。

一种电子政务云平台安全性的动态完整性度量方法

技术领域

[0001] 本发明属于信息安全访问控制领域。

背景技术

[0002] 近年来电子政务发展步伐迅猛，给国家的经济发展做出了卓越的贡献，但是网络、信息技术的发展日益不能满足电子政务的建设需要。为深化应用电子政务，实现资源共享，保障信息数据安全，产生了将云计算跟电子政务相结合的方式。云计算可以为电子政务提高更高的安全性和性能，降低成本。不过由于云计算是基于互联网的，云计算平台本身就很容易受到非法的安全威胁。传统的保证平台状态安全的方法主要是基于操作系统的安全策略机制等其他一些第三方的杀毒软件、防火墙，但是如果操作系统、杀毒软件、防火墙等软件自身就存在被攻击的行为时，这些安全的措施或者部件就很难实现其本来的目的了。可信计算特别是完整性度量技术的研究与发展，使从根本上解决电子政务云平台的安全问题成为可能。

[0003] 可信计算中的可信平台模块 TPM 完成对数据的加密、解密和认证等安全功能，存储在 TPM 中的信息不能以软件的方式被随意获取，通过 TPM 能够保证软件层面的非法访问和恶意操作无法完成。在对可信计算的研究的基础上，2004 年，IBM 研究中心提出了基于可信平台模块 TPM 的完整性度量系统架构（Integrity Measurement Architecture，IMA）的设计，此架构是在 Linux 操作系统上设计并实现的。在系统将文件打开载入内存时，系统内核的 IMA 代码将对文件的完整性进行度量计算，然后将度量结果保存到度量列表中，同时把度量结果扩展到 TPM 芯片中。从而构建了一条底层 TPM 芯片——操作系统——应用程序的完整信任链。基于完整性度量架构的 IMA，为远程机器去验证本地机器平台状态完整性有没有遭受破坏，系统是否可信提供了可靠的理论依据和实践基础。

[0004] 由于 IMA 定义的完整性是基于简单度量装载的代码以及一些系统静态数据，加上 IMA 在进行完整性度量的时候插入了大量的度量点，从而增加了度量的不准确性和度量的冗余性。2006 年 Jaeger T 等人在 IMA 的基础上提出了采用策略规约的方法减少冗余度量的 PRIMA 原型，其中的策略规约的方法是借鉴了 SELinux 的策略。不过 IMA 和 PRIMA 从本质上说仍然属于静态度量系统，也就是说它们实现的完整性度量仅仅发生在程序加载前，而不是贯穿程序的整个生命周期。Peter A. Loscocco 等人提出的 LKIM 在原来静态度量的基础上定义了一系列能够表示系统状态的变量，监控这些变量是否发生变化，一旦变量发生变化即触发重新度量的动作，以此实现其动态度量的目的。但是仍然存在以下三个问题：文件度量工作冗杂、对指定进程的度量缺乏具体的实践应用、不支持实时的完整性汇报机制。

发明内容

[0005] 本发明提供一种动态度量、安全性好的电子政务云平台安全性的动态完整性度量方法。

[0006] 为解决上述技术问题，本发明采用的技术方案是：提供一种电子政务云平台安全性的动态完整性度量方法，包括完整性度量架构 IMA，其特征是，针对指定进程和指定文件，分别在内核里设有双向链表，保存进程和文件的度量结果，TCM 芯片将度量结果扩展到相应的 PCR 中。这里所说的“指定”，是指需要度量的文件由权威部门确定，比如政府部门就需要对某些公文进行严格保护；度量指定的进程主要是提高计算机系统的安全性，辅助保护机密公文的完整性。

[0007] 进一步地，所述文件度量结果扩展到第 10 号 PCR，进程度量结果扩展到第 11 号 PCR。

[0008] 进一步地，所述进程和文件的度量结果以日志形式保存。

[0009] 进一步地，所述方法需要完整性度量应用层验证服务器以及内核层的服务度量模块，所述完整性度量应用层验证服务器为指定文件和指定进程的度量模块，所述服务度量模块包括可插入模块实现的字符设备和修改 Linux 内核原有的完整性度量体系。

[0010] 对于指定文件进行度量的设计主要集中在以下两个方面的内容：1、度量点的选取。结合电子政务云平台的特点，本方法旨在保护电子公文内容的完整性不受恶意篡改。电子公文的完整性发生变化，也通常是由于电子公文的内容发生了变化，所以在本方法的设计中，只要指定电子公文内容被修改了，就对电子公文的完整性进行一次度量。要实时检测到指点电子公文的内容是否发生变化，必须对 Linux 内核对于文件内容修改是如何处理的进行深入研究，然后在各个可以体现文件内容可以被修改的地点安放度量点。2、指定文件的标记。在原有的 IMA 实现中，对于计算机系统中每一个打开的文件都进行了度量，所以本方法对于指定公文进行的度量的设计与实现中，必须解决的一个问题就是怎样在所有已经打开并且被修改的文件中去筛选指定的文件，同时又怎么尽量使筛选所需要的时间降到最低。在本文的设计中，将把那些能够表示指定文件的信息存储在内核中，并结合 Linux 内核的设计与实现，以某种特定的数据结构来存储这些信息。

[0011] 对于支持对指定进程的度量：使用电子政务云平台进行诸如公文传输、公文阅读等方面的政务服务时，如果仅仅一味地追求电子公文在本地系统存储时的完整性不受破坏，但是传输系统的进程受到了攻击，是文件在传输的过程中被篡改了，这样由于 IMA 的设计与实现中不支持对指定进程的度量，完全察觉不到公文已经遭受破坏。为保证电子政务云平台中某些重要的进程（比如本地系统应用层中的完整性汇报服务器进程）的完整性免受恶意破坏，本方法在电子政务云平台中添加了对指定进程的度量策略，提高了完整性度量架构的度量精度，从而进一步提高电子政务云平台的安全性。

[0012] 进一步地，所述方法设有实时远程证明机制，即在任何适当的、需要的时候，都可以进行远程验证，从而实时保证机器的完整性未早篡改。当本地机器跟远程某些机器需要进行电子公文的传输（包括电子公文的接收、发送与查询等操作）之前，远程机器挑战者服务器首先对本地机器的完整性提出验证申请，以确保本地机器上的相应硬件（主板设置、相关的 ROM 等）以及指定的具有非篡改性的电子公文的完整性没有遭受破坏。本地机器上的验证服务器接着向系统内核中获取相应的完整性信息，发送到远程机器，在成功完成远程验证之后，才开始对电子公文的各种操作。

[0013] 进一步地，所述方法设有实时报告机制。实时的完整性度量汇报机制旨在实现对每个指定文件的实时监控与汇报——对于指定文件，每一次恶意篡改都会触发一次完整性

度量操作，每一次度量操作都触发一次完整性汇报。要实现实时汇报机制，考虑到程序的鲁棒性和扩展性，本方法的设计与实现从 Linux 内核源码中关于完整性度量架构部分开始，没捕捉到一次度量操作，都想用户汇报。基于此机制，远程机器便可以实时地获取本地系统平台的完整性变化，而不是仅仅做一次完整性验证就一劳永逸了，这样更进一步提高了 IMA 在远程证明中的灵活性，提高了系统的安全性。

[0014] 与现有技术相比，有益效果是：本发明建立了一种支持远程证明的电子政务平台安全框架，在保证安全方面主要通过动态完整性度量方法，对完整性度量架构 IMA 进行优化改进的方案，此方案通过对指定文件进行度量降低了验证过程中的验证次数并提高了验证性能，通过支持对指定进程代码段完整性进行度量提高了对系统平台度量的精度，通过加入实时报告机制解决了在验证过程中遇到的 TOC-TOU 问题。

附图说明

[0015] 图 1 为电子政务云平台完整性度量系统结构示意图；

图 2 为三大对象的通信过程图；

图 3 为度量指定文件的流程图；

图 4 为度量指定进程的流程图。

具体实施方式

[0016] 下面结合附图和具体实施方式对本发明作进一步地详细说明。

[0017] 如图 1 所示，本发明总体结构分为硬件层、内核层及应用层。

[0018] 在硬件层，可信计算模块（TPM）防止了系统中的部分硬件和软件层次（当然也包括电子公文）的完整性不受恶意代码或者是其他恶意操作的篡改。在本文中，电子公文的完整性信息都被存储在平台配置寄存器（PCR）中。默认地，PCR 值是无法被解封的，这样电子公文的完整性信息也是无法从 PCR 的值进行逆计算得到的。

[0019] 在内核层中，在前面介绍过，为实现对指定文件以及指定进程的度量，修改完善了原有的 Linux 内核中的完整性度量架构（IMA）以及相应的 read/write 系统调用。应用到电子政务中，为方便应用层服务器能够跟 Linux 内核进行可靠通信，设计并实现了内核可插入模块，并基于此实现了一个虚拟的字符设备，扮演了度量代理的角色。依靠度量代理，可实现获取保存在内核态的指定公文和指定进程的度量列表，还可以获取 TPM 的值，最后将这些获取的信息交由应用层进行远程证明。除了这些，这个自定义的度量代理还可以依照客户的意愿进行完整性度量工作，不过，这些操作最终都是通过 Linux 内核来完成的。所以，此度量代理扮演了应用层跟 Linux 内核的通信兵的角色。

[0020] 在应用层中，一些用户或者恶意的代码对指定的公文读写之后，篡改了指定公文的内容，由此造成对指定公文完整性的篡改，不过最终这些操作都会反映到指定电子公文完整性度量列表 TPM 的 PCR 中。应用层的验证服务器负责跟远程机器完成对本地机器的完整性验证。通过读取 Linux 内核中的度量列表，来获取指定公文的完整性度量列表，实现远程证明。本方法采用从应用层想内核可插入模块发送不同的命令，内核模块根据每个命令对应的功能，完成操作，向应用层提供数据。应用层获得度量列表，即可以跟远程机器完成远程证明。同时，为保证电子政务系统中某个指定进程代码段的完整性，应用层还可以直接

指定进程，完成对它的完整性度量，在本方法的设计中，主要是对进程的代码段的度量。

[0021] 1) 通信协议的设计

在本发明的设计中，远程挑战者机器，本地应用层服务器以及本地内核可插入模块三者相互间通信采用统一的通信协议，应用层服务器不仅需要获取文件度量信息，还需要获取进程度量信息。同时考虑到协议的可扩展性与伸缩性，在本方法的设计中，在协议结构体里面适当地添加一个命令字域，以表达不同的请求或者应答信息。本方法设计的命令字域为一个 8 比特的域，同时还添加了随机数域。

[0022]
```
struct emos_ima_request:
    uint8_t cReqCommand;
    uint32_t dwNonce;
```

其中命令字域 cReqComnand 在本方法的设计中暂时只用到了三个情况 :请求获取文件度量列表，请求对传输进程进行度量并且返回进程度量列表 ;这三者对应的命令字域分别是 0x01，0x03。对于在 dwNonce 域，是一个 32 位的域，在对文件度量的时候是一个由 TPM 生成的随机数，对于进程度量，此域被赋为进程 pid。

[0023] 对应于验证请求结构体，在本方法的设计中，根据不同的命令请求，分别设计了文件度量应答结构体以及进程度量结构体。首先先来看看文件度量应答结构体 emos_ima_response。

[0024]
```
typedef struct emos_ima_response :
uint8_t cResCommand :应答域，固定为 0x02
uint32_t dwNonce :跟 emos_ima_request 中的 dwNonce 相同
emos_pcr_info stPcrInfo :描述 TPM 设备某个寄存器的内容，具体见下文描述
uint16_t wInfoNums :在文件度量列表中的已经度量过的文件数目，用于指示此结构体中下一域数组 stImaInfo 中实际的元素个数
emos_ima_info stImaInfo[64] :从内核中获取的文件度量列表，实际度量过的数目由 wInfoNums 指示
```

依照远程证明模型，在挑战者向本机提出完整性度量请求之后，为了防止攻击者的重放攻击等等，必须将 emos_ima_request 结构体中的随机数域拷贝回去进行判断。出于只是实验性的考虑，在本文的设计中，指定的度量文件个数不会太多，度量次数也不会过于夸张庞大，所以数组 stImaInfo 的上限暂时设为 64，以后由于扩展的需要，可以适时调整这个上限值。

[0025] 下面的结构体 emos_ima_task 为进程度量应答结构体。

[0026]
```
typedef struct emos_ima_task :
uint8_t cResCommand :应答域，固定为 0x04
uint32_t dwNonce :跟 emos_ima_request 中的 dwNonce 相同，表示进程 pid
emos_pcr_info stPcrInfo :描述 TPM 设备某个寄存器的内容，具体见下文描述
uint8_t acHashVal[16][20] :从内核中获取的指定进程度量列表
```

在前面的两个数据结构中，都涉及到了关于 TPM 中某个寄存器的值，其结构体为 emos_pcr_info。

[0027]
```
typedef struct emos_pcr_info :
```

uint8_t cPcrIndex :描述此结构体取的是第几号 PCR 的值，在 emos_ima_response 中，此值为 10，在 emos_ima_task 中，此值为 11

uint8_t acPcrVal[20] :描述在 TPM 设备中的第 cPcrIndex 号的寄存器的内容，此值通过 TPM 的哈希扩展算法生成

另外，文件度量应答结构体中还有一个域 stImaInfo，它是类型为 emos_ima_info 的数组，用来描述某个指定文件在度量之后的信息。在前面章节介绍过，在本文的设计中，给远程挑战者的文件度量列表中，仅仅包含公文文件名而已，另外由于某些公文可以更新版本，所以还需提供版本号。

[0028] typedef struct emos_ima_info :

uint8_t acFileName[IMA_EVENT_NAME_LEN_MAX + 1] :描述此指定公文的文件名，IMA_EVENT_NAME_LEN_MAX 为 255

int8_t cFileVersion :描述此指定公文的文件版本号

2) 内核交互模块的设计

在本发明的设计中，内核可插入模块是整个系统的中间层，接收来自应用层服务器的命令并对其进行解析，根据不同的命令完成获取度量列表、调用内核提供的函数度量指定进程或者指定文件、实时报告度量结果、添加指定文件等功能。当然，要跟应用层服务器进行数据交互，是要通过此内核模块来实现一个设备来进行的，在本文的设计中，实现了一个简单的字符设备，作为度量代理。

[0029] (1) 模块初始化的工作

在本发明的设计中，内核可插入模块在加载的时候，就需要对那些固定的文件进行度量。对于红黑树 emos_ima_file_check_tree，是用来专门组织指定文件的，如果在内核源码中就对指定文件的信息事先写好，假如需要指定的文件数目相当大，就需要在内核源码中要么维护很大的一块空间来事先保存这些信息，要么就使用相当冗余的代码来进行红黑树的构建，显然，这种做法极大程度地损失了系统的扩展性，伸缩性甚至性能。考虑到系统的这些扩展性与伸缩性等问题，本发明没有在内核源码中指定死需要度量的文件，而是采取配置文件的方式来指定文件。也就是将需要指定的文件路径信息在配置文件中配置好，在加载内核可插入模块的时候去读取配置文件，从中获取需要指定文件的路径信息，进而组织红黑树 emos_ima_file_check_tree 以及度量这些指定文件。这样，既不用在内核源码中专门维护缓存也避免了使用冗余的程序来构建红黑树 emos_ima_file_check_tree 以及度量文件，而且对于以后要添加指定文件或者撤销对某些文件的指定话，直接更新配置文件就可以，而不是编译整个系统内核。由于指定的文件在通常情况下是不应该改变的，包括文件名、文件路径等等，也就是说，配置文件里面的内容通常情况下不能改变。所以，为了防止配置文件被恶意篡改，在本发明的设计中，首先对此配置文件进行完整性度量。

[0030] (2) 实时报告机制的设计

原来在 IMA 体系架构中缺乏实时报告机制，这样就很容易给恶意攻击者留出一个相当优良的条件 :本地机器跟远程挑战者机器在进行远程证明的时候，本地的完整性信息没有遭受任何的破坏，但是在远程证明之后，远程机器跟本地机器在进行诸如公文阅读、公文传输的过程中，本地的公文信息遭受恶意的攻击，随之本地的完整性信息已经遭遇篡改，虽然在本地机器能够实时度量完整性，但是没有一套比较可靠的实时报告机制，远程挑战者对

此毫不知情，继续对已经被篡改的公文进行“可靠”的操作。

[0031] 这种情况，也是一种 TOC/TOU 问题。基于这种 TOC/TOU 问题，在本发明的设计中，设计了一套实时报告机制。在经过了本地机器跟远程机器的两端的远程证明之后，如果本地公文的完整性信息遭到破坏，不仅本地机器需要检测度量出此完整性信息，更重要的是还得向远程挑战者机器报告这一变化，在远程机器那边重新进行一次远程验证。从另一个角度看，这种设计也可以看成是多次验证机制，也就是远程挑战者对本地机器的验证不仅仅局限于一次。

[0032] 由于实时报告涉及到三大对象 :Linux 内核、自定义度量代理、应用层服务器。其中，Linux 内核在实时报告中完成完整性度量工作，自定义度量代理完成对完整性变化的实时监测工作，应用层服务器就通过度量代理从内核中提取实时的完整性度量信息，并及时报告给远程挑战者机器。这三大对象的通信过程如图 2 所示。

[0033] 3) 应用层服务器的设计

(1) 执行对指定文件度量的总体程序流程

在本发明的实现中，对指定文件的度量的总体程序流程如图 3 所示。

[0034] 在对一个文件进行度量之前，需要对其判断是不是指定的特殊文件，这里使用函数 emos_ima_iint_find() 来进行判断。如果是指定文件，则调用本文涉及的函数 emos_ima_file_write_check() 来对此文件进行完整性度量。最后，系统会使用函数 ima_add_digest_entry() 把对文件的哈希度量值写进度量列表，这个度量列表是内核维护的一个双向链表 ;同时系统调用 ima_pcr_extend() (进而调用 tpm_pcr_extend()) 将文件哈希值扩展到 TPM 设备的 10 号寄存器中。

[0035] 在本发明的设计中，把指定文件的文件名都放进一棵红黑树中，此红黑树是由 Linux 内核定义的 rb_root。红黑树是平衡二叉树的一种，它上面的每个结点都是有序排列的，同时基于红黑树本身是平衡二叉树，所以对于红黑树的查找时间复杂度是 O（lgN）的，在 Linux 内核中，使用结构体 vm_area_struct 来管理红黑树的。emos_ima_iint_find() 在判断一个文件是否为指定文件的时候，就在这棵红黑树上去查找相应的节点对应的文件名是否跟当前文件名相同，如果相同，则证明当前文件为指定的需要进行度量的文件，否则不是。

[0036] 为了确保 emos_ima_iint_find() 函数的可操作性，需要事先将指定的文件名插入到红黑树中。在本文的设计中，链接到红黑树节点的数据结构如下设计，

```
typedef struct emos_ima_iint_cache :
    struct rb_node rb_node :用于链接到指定文件红黑树中 ;
     u8 emos_filename[128]; 指定文件的文件名 ;
    struct mutex mutex; 确保操作的原子性 ;
```

为了便于结构体 emos_ima_iint_cache 的频繁分配和回收，在本文的实现中，使用 slab 分配器来管理此结构体。在 linux 内核中，slab 分配器扮演的是通用数据结构缓存层的角色，以此来管理内核中的空闲链表。slab 层把 emos_ima_iint_cache 划分为高速缓存组，通过这些高速缓存，系统就可以高效地对 emos_ima_iint_cache 进行分配和回收。高速缓存 emos_iint_cache 是使用 kmem_cache_create 函数创建的，其中创建的采用 SLAB_PANIC 的方式。

[0037] 对文件的完整性度量的真实入口函数是 process_measurement()。此函数的返回值为 0 时表示对文件 file 成功度量。参数 function 表示对文件进行度量采取的方式，mask 表示文件进行度量的因由，可能是由于读文件引起，可能是由于写文件引起，亦或者是其他原因。在进行最终的度量之前，会将这两个参数传递给函数 ima_must_measure() 进行判断是不是一定要度量，这个函数的作用是根据 mask 以及 function 去度量策略里面查找有没有相对应的度量策略。在 IMA 体系中，度量策略都被默认维护在一个结构体为 ima_measure_rule_entry 的数组 default_rules 中，这个结构体的主要描述如下所示，

```
struct ima_measure_rule_entry:
    struct list_head list;  //用于链接到链表的域
      enum ima_action action;  //指示是否需要进行度量，DONT_MEASURE/MEASURE，前者指示不需要度量，后者相反
    unsigned int flags;
enum ima_hooks func;  //枚举类型，指示度量的动作，目前有四种动作(包括本文设计中添加的 EMOS_FILE_WRITE_CHECK)
     int mask;  //引起文件进行完整性度量的因由，MAY_READ/MAY_WRITE……
     unsigned long fsmagic;  //魔数
    uid_t uid;  //用户 id，在本文添加的策略中为 0
```

用户可以根据需要自己在里面添加或者删除某些度量策略。在本文的实现中，添加了自己定义的度量策略，具体如下：

```
.flag = MEASURE, .func = EMOS_FILE_WRITE_CHECK, .mask = MAY_WRITE, .uid = 0, .flags = IMA_FUNC | IMA_MASK | IMA_UID
```

在本文的实现中，传递给入口函数 process_measurement() 的 mask 和 function 分别指定为 MAY_WRITE 和 EMOS_FILE_WRITE_CHECK，表明是由于篡改文件内容引起的文件度量动作。

[0038] (2) 对指定进程的度量方案的实现

在本发明文的实现中，对指定进程的度量的总体程序流程如图 4 所示

首先需要根据进程 pid 去获取对应的进程结构体，描述了进程地址空间的各种信息。函数 emos_ima_task_check(pid) 中的参数 pid 是指定进程的进程 id，在用户态指定，然后通过内核模块传递过来。emos_ima_task_check() 函数作为内核符号使用宏 EXPORT_SYMBOL_GPL 导出，如此，其他的内核性质的代码就可以直接使用此函数，包括后面章节中的可插入内核模块部分。其实 emos_ima_task_check() 函数也仅仅作为一个供其他代码使用的符号而已，真正的对进程度量的入口是此函数中调用的 emos_ima_task_measurement() 函数。emos_ima_task_measurement() 函数随即调用了两个关键函数 :emos_ima_task_collect_measurement() 以及 emos_ima_task_store_measurement()。前者完成对进程代码段的哈希度量计算，后者完成对此哈希度量结果的存储 ;而前者是使用了 emos_ima_task_calc_hash() 函数来完成哈希计算，后者是一方面使用 emos_ima_add_task_digest_entry() 来将哈希计算结果存储到度量列表中去，另一方面使用 tpm_pcr_extend() 函数将此哈希计算结果扩展到 TPM 设备的第 11 号寄存器中去。

[0039] 在本发明的设计中，对于给定的一个进程 pid，要对其进程代码段进行完整性度

量，首先要根据此 pid 来获取此进程结构体。获取此进程结构体通过遍历任务队列来实现。接着调用函数 emos_ima_task_calc_hash() 对进程的代码段进程度量的时候，要先取得进程的代码段的开始地址 start_code 以及结束地址 end_code，整个代码段空间大小为他们之间的差值。下面便是对进程代码段进行度量的主体流程，

```
int emos_ima_task_calc_hash(pid_t pid, char *digest)
{
struct hash_desc desc;  //记录哈希中间结果
struct scatterlist sg[1];  //页向量
char *rbuf;  //用于缓存进程代码段部分内容，以供哈希计算
struct task_struct *st;  //指向指定进程
loff_t i_size, offset;
遍历任务队列，根据 pid 获取相应的进程结构体，将 st 指向此结构体；
使用 init_desc() 函数初始化 desc;
为 rbuf 分配适当空间；
获取进程 st 代码段的开始地址 offset 以及空间长度 i_size;
while( 如果 offset 还没到达进程 st 代码段的结束地址 ){
        从 offset 开始读取适当长度 len 的内容到 rbuf 中；
        offset += len；
        sg_init_one(sg, rbuf, len);
        调用 crypto_hash_update() 函数对页向量 sg 里面的内容进行哈希编码，结果存入 desc;
}
释放 rbuf 的空间；
调用 crypto_hash_final() 函数将暂存在 desc 结构体的中间值计算到数组 digest 中去；
}
```

在对进程代码段进行哈希计算的时候，采用的算法是 Linux 内核自带的 crypto 编码方法。在本发明的实现中，对进程代码段度量的结果的存储通过函数 emos_ima_task_store_measurement() 来实现，包括两个方面：更新度量列表以及扩展 TPM 设备的寄存器。为了跟文件度量部分区分开来，这里的度量列表是自行设计的，而在扩展 TPM 设备时也是扩展到第 11 号寄存器。进程的度量列表是一个类型为 list_head 的双向链表。链接到这个链表的结构体描述如下，

```
struct emos_ima_task_queue_entry:
    struct list_head later:  //用于度量列表对接
    pid_t pid:  //描述指定进程的 pid
   u8 digest[IMA_DIGEST_SIZE]:  //指定进程代码段的哈希度量结果。
```

[0040] 以上所述仅为本发明的一个实例，并非因此限制本发明的专利范围，凡是利用本发明说明书及附图内容所作的等效结构或流程变换，或直接或间接运用在其他相关的技术领域，均同理包括在本发明的专利保护范围内。

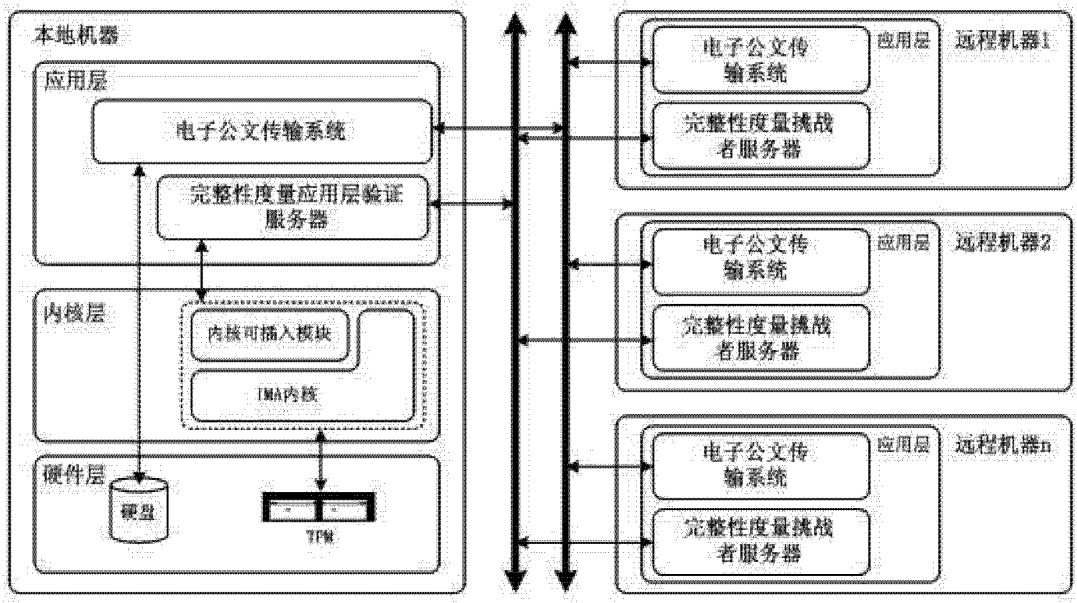

本地机器
应用层
电子公文传输系统
完整性度量应用层验证服务器
内核层
内核可插入模块
IMA内核
硬件层
硬盘
TPM
电子公文传输系统
应用层
远程机器1
完整性度量挑战者服务器
电子公文传输系统
应用层
远程机器2
完整性度量挑战者服务器
电子公文传输系统
应用层
远程机器n
完整性度量挑战者服务器

图 1

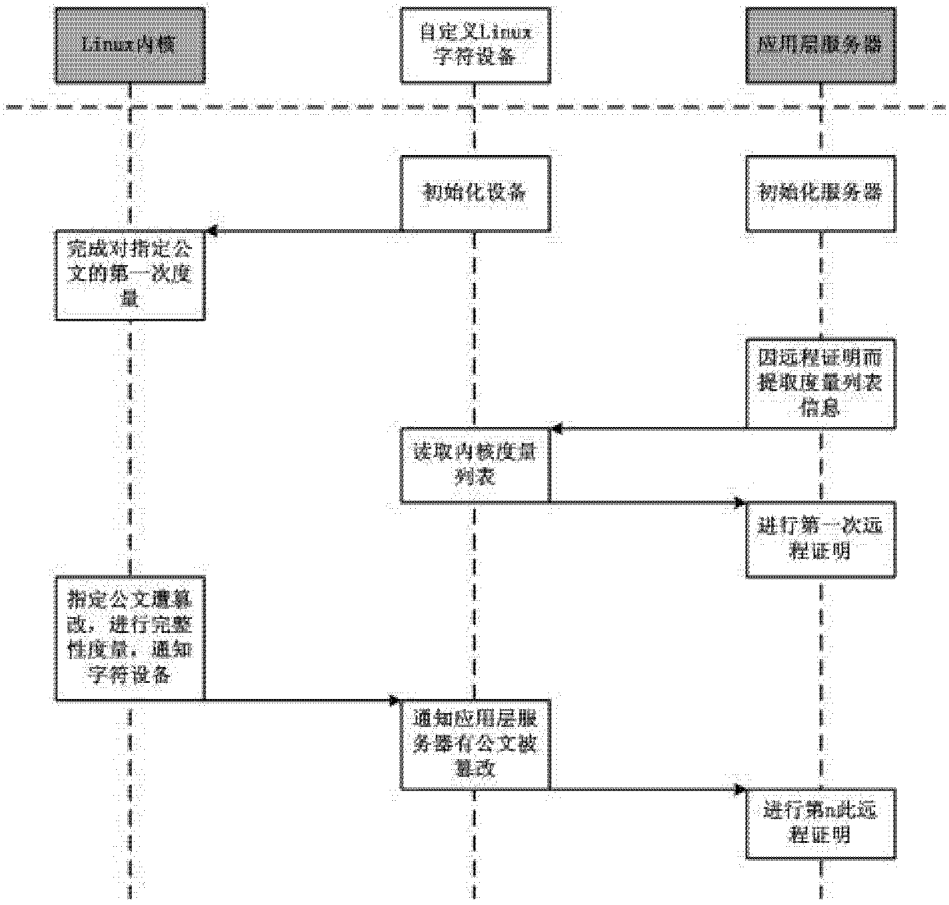

Linux内核
自定义Linux字符设备
应用层服务器
初始化设备
初始化服务器
完成对指定公文的第一次度量
因远程证明而提取度量列表信息
读取内核度量列表
进行第一次远程证明
指定公文遭篡改，进行完整性度量，通知字符设备
通知应用层服务器有公文被篡改
进行第n此远程证明

图 2

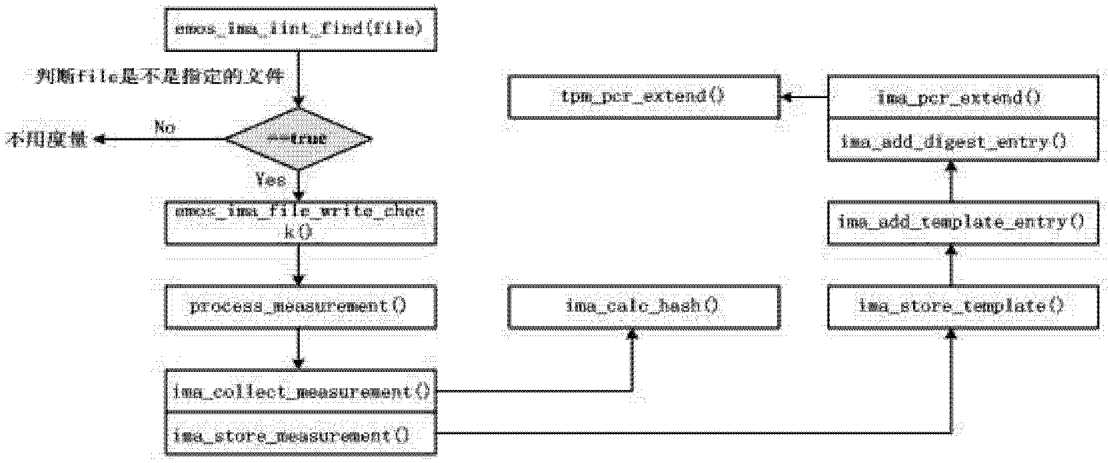

emos_ima_iint_find(file)
判断file是不是指定的文件
No
不用度量
==true
Yes
emos_ima_file_write_chec
k()
process_measurement()
ima_collect_measurement()
ima_store_measurement()
tpm_pcr_extend()
ima_pcr_extend()
ima_add_digest_entry()
ima_add_template_entry()
ima_calc_hash()
ima_store_template()

图 3

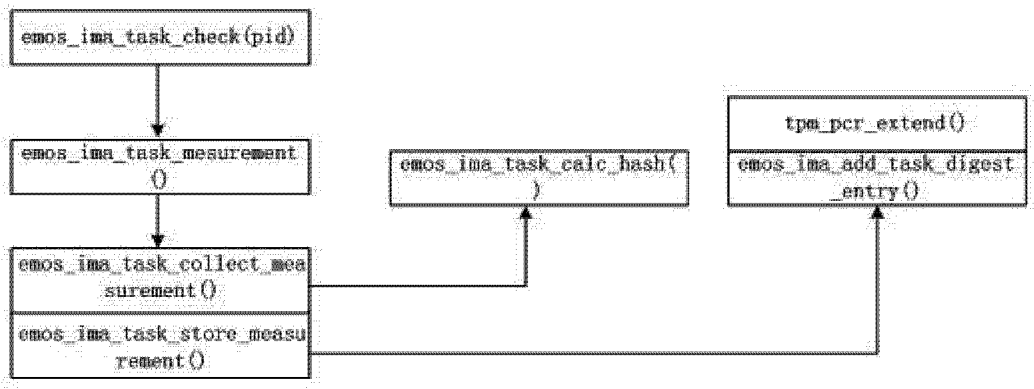

emos_ima_task_check(pid)
emos_ima_task_mesurement
()
emos_ima_task_collect_mea
surement()
emos_ima_task_store_measu
rement()
emos_ima_task_calc_hash(
)
tpm_pcr_extend()
emos_ima_add_task_digest
_entry()

图 4