软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(2):339−352 [doi: 10.3724/SP.J.1001.2011.03714]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种保护隐私的高效远程验证机制*

徐梓耀[1,2+], 贺也平[1], 邓灵莉[1,3]

[1](中国科学院 软件研究所,北京 100190)

[2](中国兵器工业信息中心,北京 100089)

[3](中国移动通信研究院 网络研究所,北京 100053)

Efficient Remote Attestation Mechanism with Privacy Protection

XU Zi-Yao[1,2+], HE Ye-Ping[1], DENG Ling-Li[1,3]

[1](Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Information Center, China North Industries Group Corporation, Beijing 100089, China)

[3](Department of Network Technology, China Mobile Research Institute, Beijing 100053, China)

+ Corresponding author: E-mail: ccxu@ercist.iscas.ac.cn, http://www.iscas.ac.cn

Xu ZY, He YP, Deng LL. Efficient remote attestation mechanism with privacy protection. *Journal of Software*, 2011,22(2):339−352. http://www.jos.org.cn/1000-9825/3714.htm

Abstract: A remote attestation mechanism, with high efficiency, flexibility and privacy protection based on Merkle hash tree is proposed in this paper. The problems of IMA (integrity measurement architecture) architecture are analyzed for a special target application scenario; followed by a detailed description of RAMT (remote attestation mechanism based on Merkle hash tree) architecture and its process of integrity measuring and verifying. The function and pseudo-code of command *TPM_HashTree*, which is a function enhancement to the existing TPM (trusted platform module), are presented for the newly proposed mechanism. The advantages of the new mechanism are analyzed and discussed.

Key words: trusted computing; remote attestation; Merkle hash tree; privacy protection; verification efficiency

摘 要: 基于 Merkle 哈希树提出了一种效率高、方式灵活并能保护平台隐私的远程验证机制.针对特定的目标应用场景,分析 IMA(integrity measurement architecture)体系架构的不足,详细描述基于 Merkle 哈希树的远程验证机制的体系架构和度量验证过程,阐述新机制对现有 TPM(trusted platform module)的功能增强即 *TPM_HashTree* 命令的功能及伪代码,并分析讨论新机制的优点.

关键词: 可信计算;远程验证;Merkle 哈希树;隐私保护;验证效率

中图法分类号: TP309 **文献标识码**: A

在传统的集中式计算环境中,涉及信息安全的计算机应用通常假设用户的机器是不可信的,用户终端只能通过与中心服务器的在线交互来建立信任.随着微型计算机(PC)与互联网技术的发展,近年来,网格计算(grid

* 基金项目: 国家自然科学基金(90818012); 中国科学院重大方向性项目(KGCX2-YW-125); 北京市科学技术委员会项目(Z08000102000801)

收稿时间: 2009-02-08; 修改时间: 2009-07-06; 定稿时间: 2009-08-19

computing)、对等网络(peer-to-peer network)等分布式计算环境受到广泛关注.与集中式计算环境相比,分布式计算环境一般没有中心服务器,其参与结点可随时加入或退出计算网络,且参与结点的计算平台也各不相同.这种开放性、动态性以及结点多样性的特点使得分布式计算环境下结点之间的信任难以建立,从而导致分布式应用的安全保障成为难题.因此,如何构建安全可靠的分布式应用成为计算机安全领域的研究热点.

可信计算组(Trusted Computing Group,简称 TCG)提出了可信计算(trusted computing,简称 TC)[1]技术,力图为分布式计算环境中参与结点的计算平台提供端点可信性.可信计算技术在计算平台的硬件层引入可信平台模块(trusted platform module,简称 TPM)[2],实际上为计算平台提供了基于硬件的可信根(root of trust,简称 RoT).从可信根出发,使用信任链传递机制,可信计算技术可对本地平台的硬件及软件实施逐层的完整性度量,并将度量结果可靠地保存在 TPM 的平台配置寄存器(platform configuration register,简称 PCR)中.此后,远程计算平台可通过远程验证机制(remote attestation)比对本地 PCR 中的度量结果,从而验证本地计算平台的可信性.可信计算技术让分布式应用的参与结点摆脱了对中心服务器的依赖,而直接通过用户机器上的 TPM 芯片来建立信任,使得创建扩展性更好、可靠性更高、可用性更强的安全分布式应用成为可能.

可信计算技术的核心机制是远程验证(remote attestation),分布式应用的参与结点正是通过远程验证机制来建立互信,从而保障应用的安全.可信计算规范[1]中的远程验证机制只验证到计算平台操作系统层的可信性,而并未定义应用程序层可信性的验证方法.事实上,应用程序层是否可信对许多分布式应用的安全至关重要,因此,研究人员提出了许多增强的远程验证机制,如 IMA(integrity measurement architecture)[3],PRIMA[4]等.这些机制虽然增强了 TCG 远程验证机制的功能,但是并没有解决普遍存在的隐私性保护问题,同时又带来验证效率低下的新问题.为此,本文提出一种基于 Merkle 哈希树[5,6]的远程验证机制,力图在保护待验证平台隐私的前提下实现高效的远程验证.

本文第 1 节首先介绍远程验证机制的基本框架,然后概述 IMA 增强型体系架构,再以某具体应用场景为例,指出 IMA 体系架构的不足.第 2 节首先介绍 Merkle 哈希树基本原理,然后提出基于 Merkle 哈希树的远程验证机制(remote attestation based on Merkle hash tree,简称 RAMT)并分析其体系架构;在此基础上,重点阐述新机制对 TPM 现有功能的增强及其远程度量验证过程.第 3 节结合 IMA 体系架构对比分析基于 Merkle 哈希树的远程验证机制的优点.第 4 节回顾远程验证和 Merkle 哈希树领域的相关工作.最后第 5 节总结全文.

1 现有远程验证机制及其问题

本节首先描述 TCG 远程验证机制及其基本实施过程;然后指出 TCG 远程验证机制的局限性并介绍 IMA 体系架构;最后以某具体分布式应用为例,指出 IMA 体系架构依然存在不足,以此作为本文的研究动机.

1.1 TCG远程验证机制与IMA增强型体系架构

1.1.1 TCG 远程验证机制

可信计算规范[1]给出的标准远程验证机制可分 3 步进行:(1) 完整性状态度量;(2) 度量结果质询通信;(3) 完整性状态验证.这 3 步共同构成了远程验证机制的基本框架,在计算平台实施远程验证时,它们缺一不可.

完整性状态度量是远程验证机制的第 1 步,由待验证计算平台在系统启动过程中实施,目的是收集待验证计算平台的硬件及软件栈完整性状态的信息,实施手段是基于信任传递模型(transitive trust model)的逐层完整性哈希值计算,图 1 展示了基于信任传递的完整性状态度量过程.由图 1 可知,待验证计算平台 TPM 的 PCR 中保存有本地平台硬件和软件的完整性哈希值.这些完整性哈希值在系统启动过程中,各层代码被执行前搜集,并被保存在能够防范恶意篡改的 PCR 中,它们如实地反映了系统从 BIOS 到操作系统各层次的完整性状态.

远程计算平台为了对待验证计算平台实施完整性状态验证,必须通过度量结果质询通信获得待验证计算平台在完整性状态度量过程中搜集的完整性哈希值的相关信息.度量结果质询通信是远程计算平台和待验证计算平台安全交换信息的过程.为保证远程计算平台能够获得正确可靠的完整性度量值,可信计算规范给出的度量结果质询通信协议使用 Nonce 和 AIK 签名机制防止了信息交换过程可能遇到的重放攻击(replay attacks)、篡改攻击(tampering)和伪造攻击(masquerading).

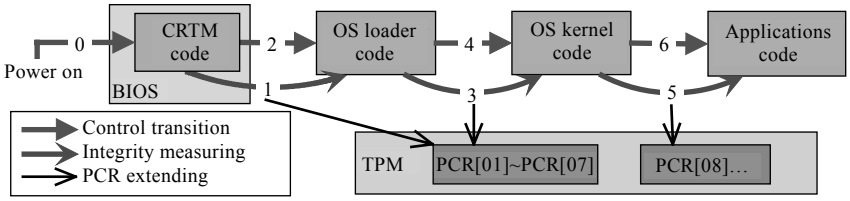


Fig.1 Integrity measurement based on trust transition

图 1 基于信任传递的完整性度量

远程验证机制的最后一步是完整性状态验证.该过程由远程计算平台实施,目标是获得待验证计算平台的安全属性,从而与之建立恰当的信任关系.完整性状态验证的结果直接决定远程计算平台与待验证计算平台的信任关系.事实上,该过程正是远程计算平台对待验证计算平台的信任评价过程.对同一个待验证计算平台,远程计算平台的真实完整性哈希值数据库或安全策略不同,均会导致其与待验证计算平台间不同的信任关系.

1.1.2 IMA 增强型体系架构

TCG 远程验证机制的完整性状态度量过程只进行到操作系统层(即图 1 的第 4 步结束),即只能验证某计算平台从 BIOS 到操作系统层的完整性,并以此建立信任关系.由于验证的层次较低、粒度较粗,TCG 远程验证机制无法获得待验证计算平台应用层软件栈准确的安全属性.为此,远程计算平台只能利用 TCG 远程验证机制提供的有限的完整性状态信息推测待验证计算平台未知的安全属性.显然,基于上述方法实施的远程验证机制是不可靠的,据此实施的分布式应用也是不安全的.为此,Sailer 等人提出了基于 TCG 的完整性度量体系架构,即 IMA[3]体系架构.

事实上,IMA 体系架构是对 TCG 远程验证机制功能的增强,将其完整性度量验证的范围拓展到应用层软件栈.在 IMA 体系架构中,待验证计算平台的操作系统在其内核空间维护一张度量列表,即 ML(measurement list).ML 的每一表项都对应于某特定软件的完整性哈希值,其中既包括操作系统启动管理器、内核模块等系统内核空间软件栈的完整性哈希值(由图 1 中的第 1 步、第 3 步获得),又包括用户空间运行的应用层软件的完整性哈希值(由图 1 中的第 5 步获得).与 TCG 远程验证机制相比,IMA 体系架构的完整性状态度量过程更进一步,其在 TCG 的可信启动结束后(即控制权已转到操作系统层),继续对待验证计算平台实施完整性度量,使用 ML 记录此后系统运行的全部应用程序的完整性哈希值,从而将完整性状态度量过程扩展到应用软件层.

IMA 体系架构使用存储在操作系统内核中的 ML 来保存完整性度量值.由于 ML 不具备 PCR 的防篡改特性,为保障其可靠性,度量所得的完整性哈希值在被添加到 ML 表项的同时还会被扩展到 TPM 的某特定 PCR 中.扩展操作在特定 PCR 中构建全部 ML 表项的有序聚合(aggregate),用于验证 ML 列表本身的完整性.

与 TCG 远程验证机制相比,IMA 体系架构完整性度量和验证的粒度更细、层次更高、更全面,因此,其反映的待验证计算平台的安全属性信息更加准确,据此建立的信任关系也更加可靠,相应应用的实施也更为安全.

1.2 IMA体系架构的不足

IMA 体系架构为安全分布式应用的实施提供了较好的手段.然而,TCG 远程验证机制的隐私保护问题在 IMA 体系架构上依然存在.此外,虽然 IMA 体系架构通过引入 ML 扩展了完整性状态度量的范围,但 ML 的引入同时降低了完整性状态验证的效率.为此,本节结合某具体的应用场景分析 IMA 体系架构依然存在的问题.

1.2.1 安全网上转账业务

假设用户要使用某银行的网上银行系统实施一次网上转账业务.为保障本次网上转账业务的安全,银行服务器有如下安全需求:在实施网上转账业务时,客户端系统必须运行在特定的计算平台下,同时必须安装运行由该银行制作发行的特定客户端软件;此外,服务器还要求用户能够使用指定的用户名、口令、USBkey 等信息来获得服务器端的认证以实施上述业务.

用户名、口令、USBkey 等认证信息是用户与银行事先协商好的,而客户端系统的可信性(是否运行在特定

计算平台下并安装运行特定客户端软件)则需要在每次实施交易前在线确认.因此,银行服务器需要在实施网上转账业务前对客户端系统进行远程验证,以确定其是否可信.

由于既要确认底层计算平台的可信性,又要确认上层应用层软件的可信性,TCG 远程验证机制不能满足需求.因此,银行服务器使用 IMA 体系架构来确认客户端系统的可信性.由第 1.1.2 节可知,IMA 体系架构能够为银行服务器提供客户端系统从底层硬件平台至上层全部运行应用程序的完整性哈希值,从而使银行服务器能够判断用户所在计算平台及其运行的客户端软件是否符合要求,最终确认网上转账业务可否实施.

1.2.2 安全网上转账业务实施 IMA 体系架构的隐患

IMA 体系架构虽然能够为上述银行业务提供可靠的完整性度量验证机制,然而其仍然存在自身局限,而这些不足导致其在隐私性保护、验证效率等方面均存在问题.

首先,与 TCG 远程验证机制相比,IMA 体系架构利用 ML 记录了客户端系统从下至上全部运行软件栈的完整性度量值.银行服务器通过 ML 可获得客户端系统的全面信息,既包括底层硬件、OS 内核及相关内核模块,又包括各种应用软件、各种静态或动态函数库以及各种系统配置文件等信息.利用这些信息,银行服务器能够对客户端系统的可信性作出准确的判断;但是,ML 在为银行服务器提供客户端系统全面信息的同时造成了对与网上转账业务不相关的客户端系统信息的泄漏.如果银行服务器滥用或者泄露这些信息,对客户端系统的安全无疑会造成较大的困扰[7].比如,银行服务器发现,某用户每次网上转账前,其 ML 中均含有某游戏程序的完整性度量值.银行服务器若有意或无意将此信息泄露给恶意第三方,那么用户很可能收到与该游戏相关的垃圾邮件,或者由于没有及时更新游戏安全补丁而被恶意攻击,甚至可能由于被攻击而感染木马、病毒等恶意软件,以致游戏帐号密码、信用卡信息以及公司商业机密等重要信息被窃取,从而造成重大经济损失.

事实上,依据银行服务器的安全需求,为保证网上转账业务的安全,其只需确认客户端系统计算平台和特定应用软件的可信性,而无须了解客户端系统的其他信息.但是,由于 IMA 体系架构基于 ML 的度量验证方式决定了其所提供的信息远超出保护网上转账业务的安全所需,因此,IMA 体系架构对客户端系统隐私信息的保护力度是不够的.

其次,IMA 体系结构在进行完整性验证时,需要首先验证 ML 的完整性,再使用 ML 中的各表项来分析待验证计算平台的安全属性.验证 ML 的方法是重新计算其有序聚合,然后与客户端 TPM 签名的存储于特定 PCR 中的有序聚合进行匹配.为了计算 ML 的有序聚合,远程计算平台需要依次遍历 ML 全部表项,同时进行类似于 PCR 扩展操作的哈希值计算.由银行服务器的安全需求可知,银行服务器只关心客户端计算平台和特定应用软件的完整性,ML 中所包含的其他软件的完整性信息并不会对网上转账业务的安全产生影响,因此并不需要对这些完整性度量值进行验证.然而,IMA 体系架构为验证 ML 的完整性,必须遍历全部 ML 表项并计算其有序聚合,这等同于强制验证 ML 所包含的全部完整性度量值.对于需要处理大规模完整性验证的银行服务器来说,这种冗余的验证方法,其完整性验证效率是低下的.

通过上述分析可知,对于安全网上转账业务而言,IMA 体系架构虽然能够验证客户端系统可信性,但其同时存在隐私保护不足和验证效率低下的问题.本文针对这种情况,提出了基于 Merkle 哈希树的远程验证机制,很好地解决了上述问题.

2 RAMT:基于 Merkle 哈希树的远程验证机制

为了保护客户端系统的隐私同时提高完整性验证的效率,我们提出一种基于 Merkle 哈希树的远程验证机制——RAMT.下文首先介绍 Merkle 哈希树;然后阐述 RAMT 的体系结构;由于实施 RAMT 需要 TPM 提供新的支持,因此接下来具体描述新机制对 TPM 规范 1.2 版的增强;最后描述基于 Merkle 哈希树的远程验证机制完整性度量验证的具体过程.

2.1 Merkle哈希树

图 2 展示了包含 4 个叶子结点的二叉 Merkle 哈希树.由图可知,在 Merkle 哈希树中,数据对象被创建为叶子结点,而树上内部结点则是其子结点所包含内容的连接(concatenation)的哈希值.树的根结点称为根哈希(root

hash),它能代表所有数据对象,因为任意数据对象的改变都会引起根哈希值的变化.

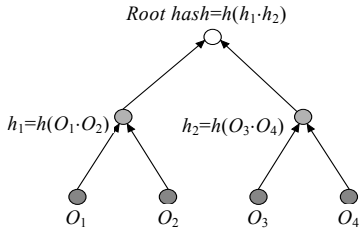


Fig.2 A binary Merkle hash tree

图 2 二叉 Merkle 哈希树

若要检查叶子结点的完整性,则需要:(1) 读取该结点及其兄弟结点的内容;(2) 连接它们的数据;(3) 计算连接后数据的哈希值;(4) 重复上述步骤至根哈希;(5) 检查计算结果与存储于根哈希结点中的内容是否一致.

若要更新叶子结点的值,则首先检查该叶子结点的完整性(过程如上),若该叶子结点完整,则:(1) 修改结点值;(2) 计算并更新其父结点的值;(3) 重复上述步骤,直到根哈希被更新完毕为止.

在信息安全领域,Merkle 哈希树常被用于解决“使用很小的可信存储空间保护存放在不可信存储器中的大量数据对象”的问题.其基本思想是,由可信构件维护 Merkle 哈希树,对任何数据对象的读取或者更新操作都通过可信构件实施,可信构件在任何操作前都先使用哈希树来验证数据对象的完整性.在实际应用中,只要保证根哈希被安全存放在可信存储器中,则即使哈希树的结点存放在不可信存储器中,也能实现对数据篡改的可知性.因为哈希树内部结点哈希值的计算采用的是非碰撞(collision-free)哈希算法,确保只要根哈希被可靠地保护,即使攻击者篡改了哈希树的某些结点,也无法利用这些结点构造一棵拥有原根哈希却拥有不同子结点的哈希树,这是计算不可行的.

2.2 RAMT的体系结构

Merkle 哈希树的安全特性使得它在实际应用中常被用来保护存储在不可信空间中的大量动态数据对象的完整性.RAMT 正是利用其上述特点.图 3 展示了其体系架构.

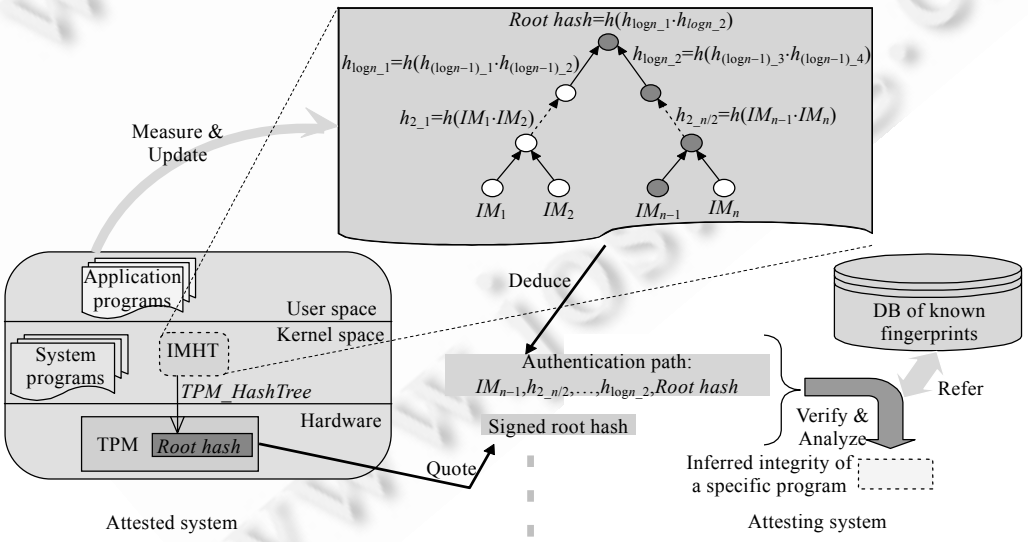


Fig.3 Architecture of remote attestation mechanism based on Merkle hash tree

图 3 基于 Merkle 哈希树的远程验证机制的体系架构

由图 3 可知,RAMT 的体系结构与 IMA 体系架构类似,但其在完整性哈希值的组织与保护、完整性验证过程以及对 TPM 的使用等方面又与 IMA 体系架构不同.

首先,RAMT 在内核中维护的不再是一张完整性度量值列表(ML),而是一棵完整性度量值哈希树(integrity measurement hash tree,简称 IMHT).其中,IMHT 的叶子结点存储的数据对象是待验证计算平台上被度量的各种程序的完整性哈希值,而其内部结点则依据 Merkle 哈希树的构建规则由子结点的连接的哈希值动态生成.

其次,为了维护 IMHT 叶子结点的完整性,RAMT 需要使用 TPM 中的一段存储器来保存 IMHT 可信根哈希的值.在 IMA 体系架构下,度量列表 ML 的完整性是通过将其各表项扩展到 TPM 中的特定 PCR,从而构建 ML 各表项的有序聚合来保障的.RAMT 不再需要构建 IMHT 叶子结点的有序聚合,只需保存 IMHT 的可信根哈希即可保障 IMHT 上全部叶子结点的完整性.IMHT 的叶子结点是随着完整性度量过程的实施而被不断添加的,这导致 IMHT 根哈希的值也随着叶子结点的加入而不断变化.因此,TPM 中保存的 IMHT 可信根哈希的值需要被不断替换更新以反映 IMHT 上各叶子结点当前的完整性状态.鉴于 TPM 的 PCR 只能对存储其中的值进行扩展或者清零操作[2]而不能替换更新,RAMT 不再使用 PCR 来保存 IMHT 可信根哈希的值,而使用 TPM 中的一块可更新的存储器来保存 IMHT 可信根哈希的值.同时,为了能够正确地更新 IMHT 可信根哈希的值,RAMT 对 TPM 规范 1.2 版的功能进行增强,增加名为 *TPM_HashTree* 的新的 TPM 接口命令(详见第 2.3 节).

再次,RAMT 的完整性验证过程基于认证路径(authentication path)实施.认证路径是指 IMHT 上从待验证叶子结点到根哈希的路径.在 IMA 体系架构中,为实施完整性验证,其需要重构 ML 各表项的有序聚合;而 RAMT 则需重构 IMHT 的根哈希.事实上,IMHT 根哈希的重构过程正是通过自底向上依次计算认证路径上各个结点的值来实现的.

总之,基于 Merkle 哈希树的远程验证体系结构的核心是完整性度量值哈希树,其由图 3 所展示的完整性度量和验证过程均围绕 IMHT 展开.

2.3 RAMT对TPM功能的增强

RAMT 要求 TPM 能够可靠地存储并替换更新 IMHT 可信根哈希的值,然而,依据可信计算规范[2]1.2 版本设计的 TPM 并不具备此功能,因此需要为 TPM 增加名为 *TPM_HashTree* 的新接口命令,其负责在 TPM 内部维护 IMHT 可信根哈希的值.图 4 展示了 TPM 内部与 *TPM_HashTree* 命令相关的数据结构以及该命令执行过程的一个例子.图中所示的 IMHT 拥有 8 个叶子结点,示例哈希树刻画了创建叶子结点 h_{1011} 后使用 *TPM_HashTree* 命令更新 TPM 中存储的 IMHT 可信根哈希值的过程.其中,浅色结点勾画出认证路径,它们沿着箭头所示方向被依次计算;而深色结点则代表为计算浅色结点各步的相应输入,图中用 step#标注其输入顺序.

如图 4 所示,IMHT 根哈希所保护的叶子结点中存储的数据对象是计算平台上各种程序的完整性度量值(IM),可用数据结构 TPM_IM_BLOB 表示.该数据结构包含 3 个域:域 address,代表该完整性度量值在 IMHT 中所处的位置,表示为从根到该叶子节点的二进制路径(根节点为“1”,路径向左延伸则附加“0”,否则附加“1”);域 name,代表被度量的程序代码的名称;域 hashValue,用来存储被度量的程序代码的完整性哈希值.

新增加的 TPM 接口命令 *TPM_HashTree* 共有 5 个参数,其中,参数 aikhandle 代表调用该命令的 AIK 句柄;参数 mode 指明本次命令调用执行的具体操作(READ 或 CREATE);参数 nonce 是用于防止重放攻击的随机数;参数 newIM 为 TPM_IM_BLOB 数据结构,代表本次命令执行的目标程序的完整性度量结果;参数 stepInputs 是一个数组,包含本次命令执行时计算根哈希各步所需的中间结点哈希值.上述这些参数均由命令的调用者(计算平台)提供,来自非 TPM 内部的不可信存储空间.利用计算平台提供的参数值,TPM 使用 *TPM_HashTree* 命令实现对其存储的 IMHT 可信根哈希值的可靠更新.

图 5 给出了 *TPM_HashTree* 命令的伪代码.由图 5 可知,*TPM_HashTree* 命令执行时需要沿认证路径依次计算哈希树上各结点的值.由于这些计算需要在 TPM 内部完成,因此 TPM 在其内部开辟若干空间,以保存哈希树的当前状态,即图 4 所示的 Hash Tree State 中包含的内容.此外,为了防止哈希树叶子结点被恶意篡改,*TPM_HashTree* 命令使用 TPM 中的单调计数器(leaf address)来保存下一个新增叶子结点在哈希树上的地址.

在初始状态-即尚未有程序的被实施完整性度量时,待验证计算平台及其 TPM 假设 IMHT 的叶子结点均为

特殊的空值(如全零),并据此建立初始 IMHT 并确定可信根哈希的初始值.显然,由于假设初始叶子结点均为空值,初始 IMHT 是完全对称的,即处在相同深度的结点是相同的.故对于拥有 N 个叶子结点的 IMHT,只需预先计算 $\log_2 N$ 个的哈希值,即可构造初始 IMHT,其中的每个哈希值代表了本层结点的初始值,而最高层的哈希值正是可信根哈希的初始值.在待验证计算平台上,这些预先计算的哈希值可用于构造初始 IMHT.同样,TPM 通过在其内部保存上述预先计算的哈希值也可初始化可信根哈希的值.

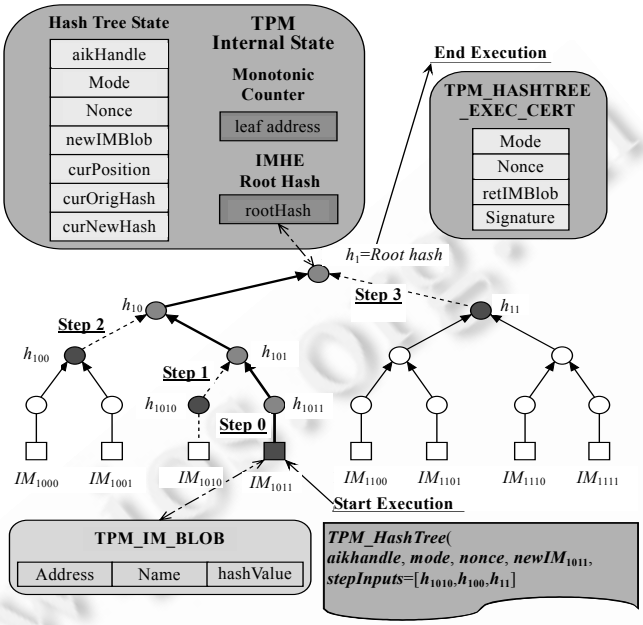


Fig.4 Related data structures of command *TPM_HashTree* and an example of its execution

图 4 *TPM_HashTree* 命令相关数据结构及其执行过程的一个例子

此后,待验证计算平台的内核负责维护 IMHT,而 IMHT 可信根哈希的值则由 TPM 在其内部维护,两者通过 *TPM_HashTree* 命令来通信.由图 5 给出的 *TPM_HashTree* 命令的伪代码可知,计算平台在创建新叶子结点时可替换更新 TPM 中的可信根哈希的值,而在读取某指定叶子结点时则不会改变可信根哈希的值.为防止用户使用 *TPM_HashTree* 命令恶意篡改可信根哈希的值,TPM 在其内部维护一个单调计数器 leaf address,用来保存下一个新叶子结点的在 IMHT 中的地址.每次计算平台向 IMHT 中成功添加新叶子结点,单调计数器 leaf address 的值便会加“1”,这使得计算平台使用 *TPM_HashTree* 命令创建新的叶子结点时,其只可向 IMHT 中增加新的结点,而不可覆盖已有的叶子结点,从而保障 TPM 中的可信 IMHT 根哈希的值不能被恶意篡改.

由此可知,初始 IMHT 和初始可信根哈希建立后,随着待验证计算平台完整性状态度量过程的进行,待验证计算平台在其内核中依次增加新叶子结点.与此同时,TPM 通过使用 *TPM_HashTree* 命令将保存其中的可信根哈希的值不断更新.此外,如图 5 所示 *TPM_HashTree* 命令的伪代码在执行时,还会检查 AIK 句柄,随机数 nonce 以及叶子结点的地址是否正确,同时会根据命令的输入计算根哈希的旧值,只有计算所得根哈希的值与 TPM 中保存的可信根哈希的值相等才会继续执行 READ 或者 CREATE 操作.上述这些检查保证可信根哈希的值被正确更新,并且在命令执行成功时返回值 TPM_HashTree_Exec_Cert 中保存的 retIMBlob 是可信的.

TPM Command: *TPM_HashTree*
Inputs: int *aikhandle*, byte *mode*;
TPM_NONCE *nonce*;
TPM_IM_BLOB *newIMBlob*;
TPM_DIGEST *stepInputs*[];
Outputs: If executes successfully, returns TPM_HASHTREE_EXEC_CERT
Else return error code.
Action:
1. Check authorization for the AIK and for command, and ABORT on failure
2. Check *mode* and ABORT if illegal
3. Check *newIMBlob.address* according to *mode*, and ABORT if illegal
4. HASHTREE_START routine:
Initialize and setup TPM's internal Hash Tree State for leaf node
   i. Let *hts* be the TPM's internal Hash Tree State
   ii. Set *hts.aikHandle=aikhandle*
   iii. Set *hts.mode=mode*
   iv. Set *hts.nonce=nonce*
   v. Set *hts.newIMBlob=newIMBlob*
   vi. Set *hts.curPosition=newIMBLob.address*
   vii. If *mode* is equal to CREATE
   Then *hts.curOrigHash*=KnownNullHashes
   hts.curNewHash=Hash(*newIMBlob*)
   viii. If mode is equal to READ
   Then *hts.curOrigHash*=Hash(*newIMBlob*)
   hts.curNewHash=KnowNullHashes //No use for READ op
5. HASHTREE_STEP loop:
FOR *i*=0 TO *stepInputs.length* Do
   i. *sliblingHash=stepInputs*[*i*]
   ii. isRight=*hts.curPosition* & 1 //get lowest bit, "1" means on the right branch, "0" means left
   iii. If (isRight) //node is on the right branch
   Then //compute the parent of the node
   hts.curOrigHash=Hash(*sliblingHash*||*hts.curOrigHash*)
   If (*hts.mode* is CREATE)
   Then *hts.curNewHash*=Hash(*sliblingHash*||*hts.curNewHash*)
   iv. Else //node is on the left branch
   hts.curOrigHash=Hash(*hts.curOrigHash*||*sliblingHash*)
   If (*hts.mode* is CREATE)
   Then *hts.curNewHash=Hash*(*hts.curNewHash*||*sliblingHash*)
   v. *hts.curPosition=hts.curPosition*>>1 //right shift
6. Check if the computed original root hash is the same as the trusted root hash stored in TPM in advance
   i. If (*hts.curPosition*!=1)
   Then ABORT //not enough stepInputs
   ii. If (*hts.curOrigHash*!=*TPM.rootHash*)
   Then ABORT //wrong newIMBlob or stepInputs
7. Execute the operation specified by mode
   i. Create new TPM_HASHTREE_EXEC_CERT *execCert*
   execCert.mode=hts.mode
   execCert.nonce=hts.nonce
   ii. If (*hts.mode* is CREATE)
   Then *TPM.rootHash=hts.curNewHash* //update the trusted root hash stored in TPM
   Increase Monotonic Counter "leaf address"
   iii. *execCert.retIMBlob=hts.newIMBlob*
   iv. *execCert.signature*=Sign(*execCert.mode*||*execCert.nonce*
   ||*execCert.retIMBlob*)using AIK specified by *hts.aikHandle*
   v. return *execCert*

Notes: 1. Mode can be READ or CREATE
2. Successful creation of a new node will increase monotonic counter "leaf address"

Fig.5 Pseudo-Code of command *TPM_HashTree*

图 5 *TPM_HashTree* 命令的伪代码

2.4 RAMT的完整性度量验证过程

与 IMA 体系架构类似,待验证计算平台启动时,从 BIOS 层到 OS 层的完整性度量依然由 TCG 远程验证机制负责;当执行控制权转到操作系统内核后,RAMT 被启动实施,并负责对应用层的各程序实施完整性度量验证,其度量验证过程围绕 IMHT 展开,下文针对第 1.1.1 节所述的远程验证的 3 个基本步骤依次比较说明.

(1) 完整性状态度量阶段

IMA 体系结构将度量所得的完整性哈希值依次添加到 ML,同时在 TPM 的指定 PCR 中构建 ML 的有序聚合.而 RAMT 则将度量所得的完整性哈希值作为叶子结点依次添加到 IMHT,同时使用 *TPM_HashTree* 命令不断更新保存在 TPM 中的 IMHT 可信根哈希的值.

(2) 度量结果质询通信阶段

IMA 体系架构将待验证计算平台的 ML 和经 TPM 签名的指定 PCR 的值传送给远程计算平台.而 RAMT 有两种选择,采取哪种方式主要取决于远程计算平台的验证能力.若远程计算平台能够计算哈希树认证路径各结点哈希值,那么待验证计算平台只要将某叶子结点到 IMHT 根哈希的认证路径上的输入哈希值序列(图 4 所示 IMHT 中的深色节点)、某叶子结点以及 TPM 签名的可信根哈希传送给远程计算平台.若远程计算平台不具备认证路径结点哈希值计算能力,那么待验证计算平台则执行 *TPM_HashTree* 命令的读取(READ)操作,获得包含某叶子结点 IMBlob 的 TPM_HashTree_Exec_Cert,然后将 TPM_HashTree_Exec_Cert 传送给远程计算平台.

(3) 完整性状态验证阶段

IMA 体系结构首先重新计算 ML 的有序聚合,再与 TPM 签名的指定 PCR 值相比较,从而验证 ML 的完整性,最后使用 ML 中各程序的完整性哈希值来判定待验证计算平台的安全属性.

根据远程计算平台的验证能力,RAMT 有两种不同的完整性验证方式.若远程计算平台具备计算哈希树认证路径结点哈希值的能力,那么其首先使用待验证计算平台返回的输入哈希值序列和叶子结点计算 IMHT 根哈希的值,之后与 TPM 签名的可信根哈希的值进行比较,在确认两者相符合后,再使用叶子结点存储的程序完整性哈希值来判断相关应用的安全需求是否得到满足.若远程计算平台不能计算结点哈希值,那么其直接从待验证计算平台返回的 TPM_HashTree_Exec_Cert 中取得相应的叶子结点,并使用其中的程序完整性哈希值即可,因为待验证计算平台的 TPM 在生成 TPM_HashTree_Exec_Cert 的过程中已经验证 IMHT 的根哈希.

3 RAMT 的优点分析与讨论

3.1 隐私性保护增强

传统的 TCG 远程验证机制将度量所得的完整性哈希值依次扩展到 PCR 中,并在进行完整性验证时重构 PCR 的值,这种机制的优点是便于构建完整性信任链,缺点则是需要了解度量所得的全部完整性哈希值以及其向 PCR 的扩展次序,因此其适于度量验证从机器加电到操作系统启动的整个过程.然而对应用程序而言,其互相之间并不信任且一般没有严格的执行顺序关系,而 IMA 基于 ML 的度量验证机制依然需要对应用程序的完整性哈希值从头到尾按顺序扩展指定 PCR 以实施度量验证,这是引起 IMA 对计算平台隐私保护不足的根本原因.在 IMA 体系架构下,无论远程计算平台是否需要,为实施度量验证,本地计算平台总是发送全部应用程序的完整性哈希值(ML),因此,本地平台的隐私信息总是完全暴露给远程计算平台.诚然,通过对远程计算平台实施认证能够在一定程度上保证远程计算平台不再将这些隐私信息主动外泄,但是如果远程计算平台被漏洞挖掘成功,那么这些隐私信息仍然存在被动泄漏的风险,而这一切都是由于 IMA 体系架构采取基于 ML 并扩展特定 PCR 的度量验证手段引起的.因此,IMA 无法保证待验证计算平台的隐私.

结合应用程序之间彼此并不依赖的并行特点,RAMT 摒弃了基于 ML 的链式结构,将待验证计算平台的各程序的完整性度量值组成一棵 Merkle 哈希树,并在树的叶子结点中保存程序的完整性哈希值,而树中的非叶子结点均根据 Merkle 哈希树的规则自动生成.在进行完整性验证时,RAMT 针对某具体叶子结点实施,即每次完整性验证的对象均为具体的特定应用程序.虽然 RAMT 有两种完整性验证方法可选择(参见第 2.4 节),但是无论采用哪种验证方法,远程计算平台一次只能获得一个特定叶子结点的内容(图 4 中的深色方块节点),并从中取得其中所包含的特定程序的完整性哈希值,而不能获得其他程序的完整性哈希值(注:远程平台所获得的输入哈希值序列中的哈希值是 Merkle 哈希树生成的,不是待验证计算平台上的程序的完整性哈希值).

事实上,如果远程计算平台是恶意的,那么通过实施一次 IMA 远程验证,利用所获得的本地平台的隐私信息,很容易对其进行漏洞挖掘.与之相对,RAMT 利用哈希函数“位承诺”的特点,可以灵活地构造验证路径,在保

证待验证应用程序完整性不可篡改的同时,屏蔽不相关应用程序的完整性信息.RAMT 体系架构在实施远程验证时,让远程计算平台一次只能获得 1 个特定应用程序的完整性哈希值,保护平台上其他应用程序的完整性信息不随意泄漏.即使远程计算平台是恶意的,实施一次 RAMT 远程验证也只能让其获得一个特定程序的完整性信息,它很难利用这有限的信息对本地平台进行漏洞挖掘.如果恶意的远程计算平台试图实施多次 RAMT 远程验证以获得大量系统中运行的应用程序的完整性信息,那么本地平台通过设置远程验证超时限制,可以察觉远程计算平台的这种恶意行为,进而终止远程验证的继续执行,达到防止自身隐私信息泄漏的目的.

综上所述,与 IMA 体系架构相比,RAMT 在度量、验证待验证计算平台上的特定程序的完整性过程中不会将其上面更多程序的完整性哈希值暴露给远程计算平台,从而确保待验证计算平台的隐私得到保护.

3.2 验证效率提高

为保障第 1.2.1 节所述的目标应用场景的成功实施,即银行服务器要在网上转账前先确认客户端系统计算平台及特定应用软件的可信性,首先假设客户端系统共有 N 个程序的完整性状态被度量,即 ML 的长度或 IMHT 的叶子结点的个数均为 N.

对 IMA 体系架构而言,度量列表 ML 的每一项均包含某特定应用程序的完整性哈希值,假设其占用的内存空间为 K 字节($20<K$),那么 IMA 体系架构需要在内存中维护 $N\times K$ 字节的信息.与之相对,如果 RAMT 体系架构的 Merkle 哈希树采用完全二叉树,并假设其叶子结点占用的内容空间也为 K 字节($20<K$),那么计算可知,RAMT 体系架构需要在内存中维护不超过 $2\times N\times K$ 字节的信息.由此可见,RAMT 体系架构在内存中维护 IMHT 所占用的空间和 IMA 体系架构维护度量列表 ML 所占用的内存空间是相当的.因此,只要是 IMA 体系架构能够实施的系统,RAMT 体系架构的实施就不存在由内存空间不足所引起的困难.

下面通过分析和比较基于 IMA 体系架构和 RAMT 两种远程验证机制在进行完整性验证时哈希值计算的次数来分析 RAMT 的验证效率.

IMA 体系架构在进行完整性验证时,为了验证度量列表 ML 的完整性,需要计算其有序聚合,然后就可以使用 ML 中的各程序的完整性哈希值.因此,IMA 体系架构的哈希值计算主要在计算度量列表 ML 的有序聚合时产生.由有序聚合的计算方式可知,需要进行 N 次哈希值计算.

RAMT 的两种验证方法均针对特定叶子节点实施,其哈希值计算主要在认证路径节点哈希值计算过程中产生.结合图 4 可知,认证路径节点哈希值计算的次数等于认证路径深度+1.对拥有 N 个叶子结点的 IMHT,其认证路径的深度为 $\log_2 N$,因此.针对特定叶子节点的验证共需要进行($\log_2 N$+1)次哈希值计算.

对第 1.2.1 节所述的目标应用场景而言,验证客户端系统计算平台的完整性均采用 TCG 远程验证机制,验证效率相同;而验证特定应用软件的完整性,RAMT 的验证效率要高于 IMA 体系架构,并且这种效率的提升会随着 N 值的增大愈加明显.这种验证效率的提升对目标应用中的银行服务器尤为重要,因为银行服务器要同时处理大量客户端请求,因此需要验证的客户端数目庞大,假设银行服务器同时验证 N 个客户端,那么使用 IMA 体系架构的验证效率的数量级为 $O(N^2)$,而使用 Merkle 哈希树的验证效率的数量级为 $O(N\log_2 N)$.可见,验证效率的提升被进一步放大.

3.3 验证方式灵活

IMA 体系架构要求远程计算平台必须能够计算度量列表 ML 的有序聚合以验证 ML 的完整性,而 RAMT 则可以根据远程计算平台的验证能力选取不同的验证方法.若远程计算平台功能较强,则可以采用让其计算认证路径各结点哈希值的验证方法;若远程计算平台功能较弱,则可以直接发送 TPM 签名的 TPM_HashTree_Exec_Cert.对于大规模的分布式计算环境而言,各结点的计算能力各不相同.显然,这种灵活的验证方式可以满足不同远程计算平台的验证能力需求.

此外,两种验证方式将计算负载分别放置在待验证计算平台或远程计算平台上,因此,可以利用不同的验证方式来平衡计算平台间的计算负载.若远程计算平台的本身的计算负载较重,则可以采用让待验证计算平台生成 TPM_HashTree_Exec_Cert 的验证方法.反之,若待验证计算平台当前的计算负载较重,则可以将验证计算负

载交给远程验证方处理.

4 相关工作

4.1 远程验证相关领域

远程验证领域的研究最早可追溯到 AEGIS[8],即 Arbaugh 等人提出的一个基于 FreeBSD 实现的原型系统.AEGIS 将系统启动过程分成 5 个级别,在系统启动时,每当控制从一个级别传递到后一个级别前,AEGIS 首先对后一级别的程序代码进行完整性验证,只有验证通过控制才会向后传递.如果某个级别的验证失效,AEGIS 则通过预先备份的数据实施强制恢复.AEGIS 实施的逐级验证启动过程是安全引导(secure booting)的概念,它为可信引导(trusted booting)的提出奠定了基础.

TCG 成立后,可信计算规范[1]正式提出了远程验证机制.TCG 远程验证机制立足于 TPM,通过可信引导机制对计算平台的软硬件基础设施进行完整性度量,使用 PCR 可靠存储完整性度量值,利用度量结果报告机制实施远程验证.TCG 远程验证机制存在度量粒度较粗、层次较低、隐私保护不够、对可信硬件依赖程度高等问题.

Sailer 等人提出的基于 TCG 的完整性度量体系架构,即 IMA[3]体系架构,通过在 OS 内核中维护一张度量列表,记录系统运行过的应用程序的完整性度量值,IMA 体系架构将远程验证扩展到应用程序层.在此基础上,Sailer 等人又提出了基于远程验证的访问策略实施框架[9].通过该框架,客户端平台能够在本地可靠地实施服务器端平台的访问控制策略.IMA 体系架构的度量列表记录全部运行应用程序的完整性度量值,这种大而全的设计导致其完整性验证效率低下.为此,Jaeger 等人提出了 PRIMA[4],将 CW-Lite 模型[10]融入到度量验证过程中,利用 SELinux 安全策略来确定待验证系统中的可信主体及其信息流,从而精简 IMA 体系架构;同时,PRIMA 通过关注代码运行过程中信息流的完整性还解决了由错误输入数据引起的代码运行安全问题.

Shi 等人提出了基于可信计算和现有微处理器架构实现的细粒度验证服务——BIND[11].BIND 在关键代码即将执行前对其进行完整性度量,然后使用沙盒(sand-box)机制保护其运行环境,最后将关键代码的执行结果和度量结果绑定,作为输入数据,在下次使用时被验证.通过将完整性度量对象的范围缩小到关键代码,BIND 提高了度量验证的粒度,而代码度量后马上执行的机制也缓解了度量-使用时间差问题[3].

为了摆脱远程验证机制对可信硬件的依赖,Seshadri 等人提出了纯软件的远程验证方案——Pioneer[12]和 SWATT[13].这种方案通过能够实施伪随机内存遍历的自校验代码来动态地建立远程验证的可信根.由于自校验代码是精心设计并经过优化的,可保证任何对它的篡改都会导致校验和计算时间显著增加,通过比较本地和远程两次计算自校验代码的时间可确定远程计算平台是否可信.其优点是为不具备 TPM 的机器提供了实施远程验证的方法,但是由于需要猜测待验证平台的计算能力且易受网络传输时间影响,因此验证结果的可靠性较低.

远程验证领域的其他研究还包括:Halda 等人提出的语义远程验证[14],试图增强远程验证的语义表达能力;Sadeghi 等人[7]和 Poritz 等人[15]分别提出的基于属性的远程验证方案,试图通过引入第三方代理来解决远程验证中的隐私保护问题;Maruyama 等人提出的 Tpod[16],Sandhu 等人提出的基于可信计算的对等网络访问控制框架[17],Zhang 等人提出的网格环境下的安全数据共享[18]等.这些研究试图为网格等分布式计算环境设计远程验证机制的实施方案.

4.2 Merkle哈希树相关领域

哈希树,又称 Merkle 哈希树[5,6],由 Merkle 在 1980 年作为一种通过使用公钥基础设施在两实体间建立共享秘密的方法提出,其目标是取代“基于证书机制在两实体间建立共享秘密”的方法,因为后者需要一个可信证书权威来确认某实体与其公钥证书之间的正确绑定关系.该方法思路如下:首先计算包含全部实体-公钥证书绑定关系的文件的哈希值并将其公布;如果实体 B 试图验证某公钥是否绑定到实体 A,则只需重新计算包含全部绑定关系的文件的哈希值,然后与其已知公布值比对即可.然而,要获得全部绑定关系是困难的,因为包含全部绑定关系的文件可能很大.为此,Merkle 提出使用哈希树来组织包含全部绑定关系的文件.通过使用哈希树,只需从绑定关系结点到根哈希结点的认证路径上的中间结点即可重构绑定关系文件的哈希值,进而验证特定公钥

与实体的绑定关系.

由于 Merkle 哈希树的根哈希能够反映全部叶子结点的完整性,因此只要 Merkle 哈希树的根哈希被可靠地保存,就能利用它保护存放于不可信空间的大量数据对象叶子结点.

鉴于 Merkle 哈希树的上述特点,Blum 等人在文献[19]中最早提出使用 Merkle 哈希树来保护内存中的数据块.该文献分析并指出,以 Merkle 哈希树形式构建的内存的访问效率是 $O(\log N)$,其中,N 为内存中数据块的个数.然而,该文献并没有指出在实际应用中,内存数据块的大小如何取值.

只可执行内存(the execute only memory,简称 XOM)体系架构[20]为应用程序提供了一种隔离的、安全可靠的执行环境.在 XOM 体系架构下,内存中的数据按照不同区域(compartment),被不同的密钥加密存储,当数据被读入处理器时,内存中的数据被解密同时加标记(tagging),然后在程序在执行时,XOM 体系架构要求其不可读写非自身区域内的数据,通过限制程序读写数据的区域 XOM 保证程序的正确执行.然而,XOM 体系架构在内存中数据的保护上做得并不好,容易遭受重放攻击.为此,Gassend 等人提出基于 Merkle 哈希树来组织并验证内存中数据的方法[21].文献[21]分析并指出,基于哈希树构建的内存的访问开销很大.为了提高内存数据的访问效率,该文献提出了缓存哈希树内部节点的优化方法(CHTree),并通过实验证明这种优化方法效果显著.

此后,Suh 等人对内存完整性验证的方法进一步研究,提出了基于日志哈希(LHash)[22]的内存完整性验证方法,并给出其改进方案——层次状日志哈希(H-LHash),还比较了上述几种基于哈希值构建的内存完整性验证方法的性能.在上述研究工作的基础上,Suh 等人最终提出一种抗篡改的体系架构——AEGIS[23](这里的 AEGIS 与 Arbaugh 等人提出的用于实施安全引导的 AEGIS[8]是两个不同的概念).AEGIS 使用安全处理器(secure processor),它能够加密内存中的数据并在运行时通过处理器内部解密数据并使用;为防范重放攻击,AEGIS 将内存中数据组织成 Merkle 哈希树,并将哈希树的根哈希保存在安全处理器中.

在可信计算领域,Sarmenta 等人提出了一种不依赖可信 OS 而是基于 TPM 或者类似的设备来实现海量的虚拟单调计数器的机制[24],并给出两种具体实施方案:一种是基于日志的方案(log-based scheme),该方案基于 TPM 规范 1.2 版本实施;另一种是基于 Merkle 哈希树的方案(hash-tree-based scheme),这种方案需要扩展现有 TPM 的功能.基于哈希树实施的方案不仅可以改善原有方案的性能和扩展性,而且能够用于实现次数受限对象(count-limited objects).次数受限对象包括:N 次可用加密密钥、只限 N 次拷贝的可移动数据对象等只可以在单调计数器某规定数值范围内使用的对象.

此外,Merkle 哈希树领域的相关研究还包括:Maheshwari 等人提出的利用 Merkle 哈希树在不可信存储空间中构建可信数据库的方法[25],该方法还讨论了如何利用内存局部性(locality)来降低磁盘访问的带宽;Williams 等人讨论的基于 Merkle 哈希树的内存完整性验证机制的数据块参数的选择最优化问题[26],等等.

5 结 论

本文分析了 TCG 远程验证机制及 IMA 扩展体系架构的不足,提出了基于 Merkle 哈希树的远程验证机制——RAMT.本文阐述了 RAMT 的体系架构,并详细描述了新增加到 TPM 中 *TPM_HashTree* 命令的接口、功能以及伪代码实现方案;在此基础上,本文按照远程验证的 3 个基本步骤阐述了 RAMT 的完整性度量验证过程,并分析讨论了 RAMT 体系架构的优点.

与 IMA 体系架构等现有远程验证机制相比,本文提出的远程验证机制对待验证计算平台的隐私性保护更好,对待验证计算平台的完整性验证效率更高.同时,本文提出的新远程验证机制的完整性验证方式更加灵活,便于应用于分布式计算环境,与网格、P2P 等分布式应用相结合.

References:

[1] Trusted Computing Group. TCG specification architecture overview revision 1.4. 2007. http://www.trustedcomputinggroup.org/

[2] Trusted Computing Group. TPM main specification version 1.2 revision 103 part 1 & 2 & 3. 2007. http://www.trustedcomputinggroup.org/

[3] Sailer R, Zhang XL, Jaeger T, van Doorn L. Design and implementation of a TCG-based integrity measurement architecture. In: Proc. of the 13th USENIX Security Symp. Berkley: USENIX Association, 2004. 223−238.

[4] Jaeger T, Sailer R, Shankar U. PRIMA: Policy-Reduced integrity measurement architecture. In: Ferraiolo D, *et al*., eds. Proc. of the 11th ACM Symp. on Access Control Models and Technologies. New York: ACM, 2006. 19−28.

[5] Merkle RC. Protocols for public key cryptosystems. In: Proc. of the IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society, 1980. 122−134.

[6] Merkle RC. A certified digital signature. In: Brassard G, ed. Proc. of the 9th Annual Int'l Cryptology Conf. on Advances in Cryptology. Heidelberg: Springer-Verlag, 1989. 218−238. [doi: 10.1007/0-387-34805-0_21]

[7] Sadeghi A, Stüble C. Property-Based attestation for computing platforms: caring about properties, not mechanisms. In: Raskin V, ed. Proc. of the 2004 Workshop on New Security Paradigms. New York: ACM, 2004. 67−77. [doi: 10.1145/1065907.1066038]

[8] Arbaugh WA, Farber DJ, Smith JM. A secure and reliable bootstrap architecture. In: Proc. of the IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society, 1997. 65−71.

[9] Sailer R, Jaeger T, Zhang XL, van Doorn L. Attestation-Based policy enforcement for remote access. In: Aturi V, *et al*., eds. Proc. of the 11th ACM Conf. on Computer and Communications Security. New York: ACM, 2004. 308−317. [doi: 10.1145/1030083.1030125]

[10] Shankar U, Jaeger T, Sailer R. Toward automated information-flow integrity for security-critical applications. In: Proc. of the 13th Annual Network and Distributed Systems Security Symp. Internet Society, 2006.

[11] Shi E, Perrig A, Van Doorn L. BIND: A fine-grained attestation service for secure distributed systems. In: Proc. of the IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society, 2005. 154−168. [doi: 10.1109/SP.2005.4]

[12] Seshadri A, Luk M, Shi E, Perrig A, van Doorn L, Khosla P. Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems. In: Herbert A, ed. Proc. of the ACM Symp. on Operating Systems Principles. New York: ACM, 2005. 1−16. [doi: 10.1145/1095810.1095812]

[13] Seshadri A, Perrig A, van Doorn L, Khosla P. SWATT: SoftWare-Based ATTestation for embedded devices. In: Proc. of the IEEE Symp. on Security and privacy. Washington: IEEE Computer Society, 2004. 272−282. [doi: 10.1109/SECPRI.2004.1301329]

[14] Haldar V, Chandra D, Franz M. Semantic remote attestation—A virtual machine directed approach to trusted computing. In: Proc. of the 3rd Virtual Machine Research & Technology Symp. Berkley: USENIX Association, 2004. 29−41.

[15] Poritz J, Schunter M, Herreweghen EV, Waidner M. Property attestation–scalable and privacy-friendly security assessment of peer computers. Technical Report, RZ3548, New York: IBM, 2004.

[16] Maruyama H, Seliger F, Nagaratnam N, Ebringer T, Munetoh S, Yoshihama S. Trusted platform on demand. Technical Report, RT0564, Tokyo: IBM Tokyo Research Laboratory, 2004.

[17] Sandhu R, Zhang XW. Peer-to-Peer access control architecture using trusted computing technology. In: Ferrari E, ed. Proc. of the 10th ACM Symp. on Access Control Models and Technologies. New York: ACM, 2005. 147−158. [doi: 10.1145/1063979.1064005]

[18] Sandhu R, Ranganathan K, Zhang XW. Secure information sharing enabled by trusted computing and PEI models. In: Lin FC, ed. Proc. of the 2006 ACM Symp. on Information, Computer and Communications Security. New York: ACM, 2006. 2−12. [doi: 10.1145/1128817. 1128820]

[19] Blum M, Evans W, Gemmell P, Kannan S, Naor M. Checking the correctness of memories. In: Proc. of the 32nd Annual Symp. on Foundations of Computer Science. Washington: IEEE Computer Society, 1991. 90−99.

[20] Lie D, Thekkath C, Mitchell M, Lincoln P, Boneh D, Mitchell J, Horowitz M. Architectural support for copy and tamper resistant software. ACM SIGPLAN Notices, 2000,35(11):168−177. [doi: 10.1145/356989.357005]

[21] Gassend B, Suh GE, Clarke D, van Dijk M, Devadas S. Caches and hash trees for efficient memory integrity verification. In: Proc. of the 9th Int'l Symp. on High-Performance Computer Architecture. Washington: IEEE Computer Society, 2003. 295−306. [doi: 10.1109/HPCA.2003.1183547]

[22] Suh GE, Clarke D, Gassend B, van Dijk M, Devadas S. Hardware mechanism for memory integrity checking. Technical Report, MIT-LCS-TR-872, Cambridge: Massachusetts Institute of Technology, 2002.

[23] Suh GE, Clarke D, Gassend B, van Dijk M, Devadas S. AEGIS: Architecture for tamper-evident and tamper-resistant processing. In: Banerjee U, ed. Proc. of the 17th Annual Int'l Conf. on Supercomputing. New York: ACM, 2003. 160−171. [doi: 10.1145/782814.782838]

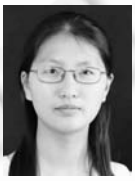
[24] Sarmenta LFG, van Dijk M, O'Donnell CW, Rhodes J, Devadas S. Virtual monotonic counters and count-limited objects using a TPM without a trusted OS. In: Juels A, ed. Proc. of the 1st ACM Workshop on Scalable Trusted Computing. New York: ACM, 2006. 27−42. [doi: 10.1145/1179474.1179485]

[25] Maheshwari U, Vingralek R, Shapiro W. How to build a trusted database system on untrusted storage. In: Proc. of the 4th Symp. on Operating System Design & Implementation. Berkley: USENIX Association, 2000. 135−150.

[26] Williams D, Sirer EG. Optimal parameter selection for efficient memory integrity verification using Merkle hash trees. In: Proc. of the 3rd IEEE Int'l Symp. on Network Computing and Applications. Washington: IEEE Computer Society, 2004. 383−388. [doi: 10.1109/NCA.2004.1347805]

徐梓耀(1980－),男,吉林省吉林市人,博士,工程师,主要研究领域为安全操作系统,可信计算技术,虚拟机技术.

贺也平(1962－),男,博士,研究员,博士生导师,主要研究领域为密码协议,安全操作系统,可信计算技术.



邓灵莉(1980－),女,博士,工程师,主要研究领域为对等网络安全,分布式访问控制及互操作,可信计算技术.