

可信计算环境证明方法研究

冯登国 秦 宇

(中国科学院软件研究所信息安全国家重点实验室 北京 100190)

摘 要 首先分析了可信计算环境下多远程证明实例执行的动态性、并发性、一致性问题,提出了一个完整的可信计算环境多远程证明实例动态更新证明方案,以保证通信双方终端计算环境的可信.然后阐述了主要由计算环境组件度量算法、会话组件树计算算法和多远程证明实例证明协议组成可信计算环境证明方法.最后对该证明方法的安全性和效率进行分析,并构建原型系统论证证明方案的可行性和高性能.

关键词 可信计算;远程证明实例;组件度量;会话组件树;更新证明

中图法分类号 TP309

Research on Attestation Method for Trust Computing Environment

FENG Deng-Guo QIN Yu

(State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100190)

Abstract At first, the authors analyze the problems of dynamic characteristic, concurrency and consistency for Multiple Remote Attestation Instance (Multi-RAI) in trust computing environment, and propose a complete dynamic update attestation scheme for Multi-RAI in trust computing environment, which guarantees the trustworthiness of endpoints' computing environment. Then the authors illustrate attestation method of trust computing environment which is comprised of measurement algorithm for computing environment, computing algorithm for session component tree and attestation protocol for Multi-RAI. At last the authors analyze the security and efficiency of Multi-RAI attestation method, and construct the prototype system for proving scheme's feasibility and high-performance.

Keywords trust computing; remote attestation instance; component measurement; session component tree; update attestation

1 引 言

分布式应用对开放系统环境下的计算平台的安全性要求越来越高,分布式应用中包括众多的利益和安全冲突方,因此建立平台间的相互信任,证明计算环境可信已经成为当前信息安全的一大迫切需

求.另一方面运行有恶意程序代码的计算平台表现出任意的攻击行为,例如拜占庭攻击,这使得平台之间无法保持长久的静态信任,目前通用的思路是使用远程代码证明来标识远程平台运行的软件配置和状态,乃至计算环境状态,检测出被损坏的参与平台,保证可信运行环境的平台间的正常通信. TCG组织制定了可信计算平台、可信存储和可信网络的一

系列工业标准^[1],其方法是在主机平台、移动平台和嵌入式平台上嵌入专用的安全芯片 TPM(Trusted Platform Module),以此为系统信任根解决可信计算平台信任的建立和证明问题.我国也制定了具有自主知识产权的 TCM(Trust Cryptographic Module)相关标准^[2],国内 IT 厂商相继研制出支持 TCM 标准的安全芯片,与 TPM 标准和芯片类似,TCM 标准和芯片同样支持可信计算平台计算环境的证明,本文将 TPM/TCM 的这种证明平台状态的安全功能通称为远程证明(remote attestation)^[3].

1.1 相关工作

TCG 框架下的远程证明方案得到了国内外众多学者、研究机构的广泛关注,众多的研究成果中较为典型的有 IBM 研究院提出的完整性度量框架 IMA^[4-5].这些远程证明方案中,基于属性的远程证明方案具有明显的发展优势,它克服了原有远程证明方案的复杂性、隐私泄漏和滥用证明结果等缺陷. IBM 研究院的 Poritz 等在文献^[6]中引入可信第三方转换属性,提出了基于属性的远程证明框架.随后该院的 Sadeghi 等在文献^[7]中给出了基于属性的远程证明方案的软硬件实现方法. Chen 提出了基于属性的远程证明协议(简称 PBA 协议)^[8].文献^[9]在可信引导器 TrustedGrub 的基础上,对基于属性的远程证明方法、属性验证和撤销等实现技术进行了研究.这些研究不断地推动了基于属性的远程证明方案的发展.

上述工作改进了 TCG 远程证明方案的各方面不足之处,扩展了远程证明方法,但对于远程证明的动态性和并发性研究很少.文献^[10]讨论了计算环境配置改变和封装数据不可用的问题,提出了基于属性封装的解决方法,实际上远程证明也存在同样的问题.当系统软硬件配置和状态发生改变时,原有的远程证明将无效,必须再次动态地进行更新证明,这就涉及了远程证明动态性的问题.可信计算平台远程信任建立实际应用过程中,系统中可能存在着多个远程证明会话实例(Multiple Remote Attestation Instance, Multi-RAI),这就要求 Multi-RAI 能够并发证明不同的 RAI 关联运行环境的可信性.与单个远程证明应用相比,又存在许多新的复杂问题.本文在 TPM/TCM 平台下主要针对可信计算环境证明的动态性、并发性等问题展开研究.

解决现有网络安全最常用的安全协议有 SSL 协议(或 TLS 协议)^[11]和 IPsec 协议^[12].SSL 协议和 IPsec 协议认证终端用户身份,保证网络通信数据的机密性和完整性,而远程证明保证终端运行环

境的可信性,将二者结合可以极大地增强网络应用的安全性,TCG 的体系结构互操作规范^[13]和 TNC 规范^[14]都涉及到了相关问题的讨论.文献^[15]提出了使用远程证明扩展 SSL 协议的方案,通信终端通过协商安全参数和在 SSL 协议上证明平台配置,以达到建立远程可信通道的目标. TCG 工作组也正着力于建立 TLS 协议上的远程证明扩展规范^[16]. Multi-RAI 安全连接的讨论也是建立在 TLS 协议扩展的远程证明基础之上.

系统环境复杂多变,需要采用多种安全机制来保护和标识软件运行环境,虚拟技术^①提供的强隔离机制为保护和维持计算环境的完整性,提供了良好的基础.无论是以 Xen^[17]为代表的虚拟技术、微内核技术^[18], Intel 的 LT 技术^②,还是 Microsoft 的 NGSCB 技术^③,都是在可信虚拟机监控器上将系统划分为不同安全级的计算隔间(Compartment),严格限制 Compartment 间的信息流,以达到控制和维护计算环境安全性的目的. Multi-RAI 证明的可信计算环境是以 Compartment 为基本单元,证明 Compartment 中组件构建的运行环境是否可信.

1.2 问题描述

系统打补丁、应用程序的升级,乃至恶意程序的篡改等致使可信计算平台的状态发生改变,状态的改变导致必须进行状态更新证明. Multi-RAI 的并发性和平台状态的动态性,极大地增加了远程证明的复杂度. TCG 规范提出的远程证明方法主要针对平台静态状态的远程证明,无法解决平台状态动态改变的远程证明问题. Multi-RAI 证明将对可信计算环境动态改变进行全面研究,重点解决 Multi-RAI 远程证明的动态性、一致性、并发性和防重发攻击等问题. Multi-RAI 证明最直观的解决方法便是扩展单个 RAI 证明,但这种简单扩展会引发许多新问题,图 1 归纳了简单 RAI 扩展用于 Multi-RAI 证明的常见问题.

图 1 中 Multi-RAI(a)描述的是非一致性证明,在第 4 步应该证明的 RAI_A 计算环境状态 β ,可是证明时 RAI_B 改变 Compartment 状态为 γ , RAI_A 实际证明的状态为 γ ,证明状态不一致,这是由于 Multi-RAI 计算环境状态相互影响造成的 TCG 规

① Intel. Virtualization technology, <http://www.intel.com/technology/computing/vptech/>, 2005

② Intel. LaGrande technology architectural overview, http://download.intel.com/technology/security/downloads/LT_Arch_Overview.pdf

③ Microsoft. Microsoft NGSCB home page, <http://www.microsoft.com/resources/ngscb>, 2003



图 1 Multi-RAI 证明常见问题

范采用服务器发送新鲜值 Nonce,防止远程证明的重放攻击,但是这种方法在 Multi-RAI 的更新证明中将不再适用,除首次证明外更新证明都是在 Requester 不知道的情形下发生,如果通知 Requester 发送 Nonce,Compartment 此时可能已经改变为另一状态,这将出现 Multi-RAI(a)的非一致性证明. Multi-RAI(b)描述的是 RAI 证明的重放攻击证明,第 4 步 RAI_A 计算环境状态已经改变为 γ ,而不诚实的用户可以重放状态 β 的证明. Multi-RAI 可信计算环境对于远程证明提出了新的安全要求.

(1)证明的动态性. 可信计算环境的软硬件配置和状态动态发生改变时,RAI 必须动态地向 Requester 证明更新后的新状态可信.

(2)证明的一致性. RAI_A 证明可信计算环境中它关联的运行状态时,不能由于其他 RAI 改变了可信计算环境的状态,而导致 RAI_A 错误的证明改变后的新状态,RAI_A 的证明必须与 RAI_A 的关联状态一致.

(3)证明的并发性. Multi-RAI 环境下同时存在多个并发运行的、向不同 Requester 证明计算环境的可信性的 RAI. 不能因为 RAI 证明而锁定当前计算环境,亦即 RAI 证明时,不允许平台状态更新,不允许其他 RAI 同时进行证明. 不管平台的运行环境经过了多少次改变,Multi-RAI 还能够并发证明其不同的最新关联状态.

(4)防重放攻击. RAI 的关联状态与 RAI 证明一一对应,但是当关联状态改变为不安全时,不诚实的用户,或恶意程序能够重用原有的证明数据,使 Requester 相信 RAI 仍然可信,这就是 Multi-RAI 的重放攻击.

1.3 我们的工作

我们提出了一个 Multi-RAI 可信计算环境证明方案,该方案主要包含可信计算环境组件度量描述、会话组件树构建和 Multi-RAI 证明流程,较好地解决了多个远程证明实例执行的并发性和一致性问题. 本文第 2 节介绍 TPM/TCM 的基础知识,概述

Multi-RAI 的证明模型和简要流程;第 3 节具体阐明详细的 Multi-RAI 证明方案;第 4 节介绍基于 Multi-RAI 证明方案的原型系统实现;第 5 节对全文进行总结.

2 证明模型

2.1 TPM/TCM 基础

TPM/TCM 是一个安全芯片(实质上是一个密码芯片),可提供密码操作、完整性管理、密钥管理、安全存储、远程证明等安全功能. 本文主要涉及 TPM/TCM 的完整性配置管理、单调计数器、TPM/TCM 身份密钥等功能.

TPM/TCM 内部包含一组不可篡改的平台配置寄存器(PCR),TPM/TCM 度量可信计算环境的软硬件状态,将度量结果保存在这些 PCR 寄存器中. PCR 寄存器的主要操作包括扩展、重置、签名等,当系统启动后,PCR 不能重置和篡改,只能进行杂凑值的扩展,其扩展方法为 $TPM_Extend(\alpha, \beta) = \mathcal{H}(\alpha \parallel \beta)$,其中 α 为 PCR 的旧值, β 为扩展值,输出 PCR 的新值,TCG 规范指定的 Hash 算法为 SHA1. 令 $\mathcal{R} = \{0, 1\}^{160}$,将 PCR 扩展操作定义为点积“ \cdot ”运算,则 $\mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}: (\alpha, \beta) \mapsto \mathcal{H}(\alpha \parallel \beta) = \alpha \cdot \beta$.

以可信计算平台的初始环境创建过程为例,主机平台的 BIOS 杂凑运算的度量结果为 β ,OS Loader 为 λ , OS kernel 为 κ , OS 加载的内核模块(例如特定硬件的驱动、内核服务模块等)度量结果为 χ_1, \dots, χ_m ,加载的用户空间的应用程序为 $\chi_{m+1}, \dots, \chi_{m+n}$,那么对整个信任链进行的扩展操作是

$$PCR = \mathcal{H}(\mathcal{H}(\dots \mathcal{H}(\mathcal{H}(\mathcal{H}(\mathcal{H}(0^{160} \parallel \beta) \parallel \lambda) \parallel \kappa) \parallel \chi_1) \parallel \dots) \parallel \chi_{m+n}) = 0^{160} \cdot \beta \cdot \lambda \cdot \kappa \cdot \chi_1 \cdot \dots \cdot \chi_{m+n} \tag{1}$$

TPM/TCM 单调计数器(Monotonic Counter)值保存在 TPM/TCM 内部的 NV 存储区域内,且计数器只能被增加. 创建一个新计数器需要 Owner 的授权,增加一个计数器值需要计数器相关授权口

令,删除计数器要么需要 Owner 授权,要么需要计数器相关的授权口令.创建一个新计数器后,计数器被赋予了一个初始值,通常初始值不从零开始.计数器可用来标识可信计算环境状态的更新顺序和防止证明数据的重放攻击.

2.2 体系结构

组件通过度量变量来衡量其安全属性,我们为组件颁发属性证书,来验证组件是否可信,图 2(a)

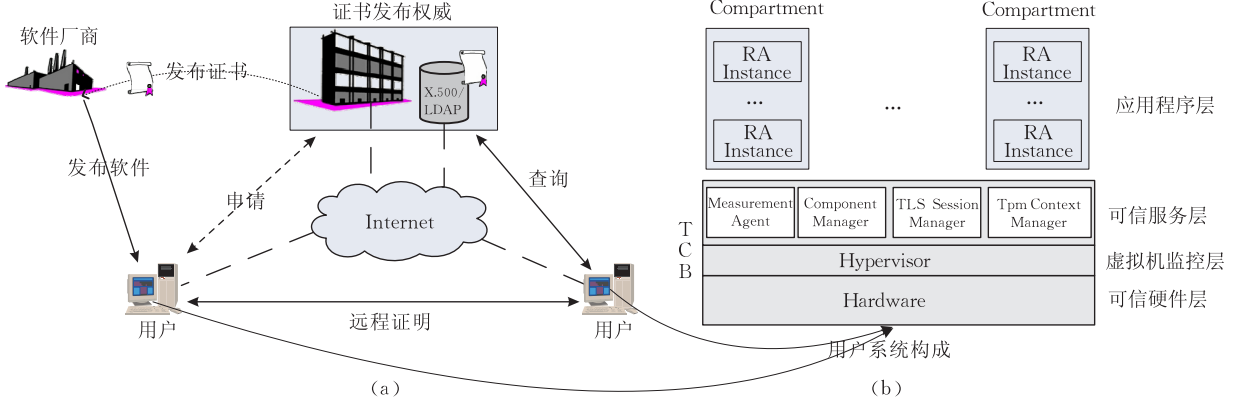


图 2 可信计算环境证明模型和体系结构

设 T 的公私钥为 (SK_T, PK_T) ,经过 T 评估认证的组件安全属性集合为 \mathcal{P} , \mathcal{P} 包含有长度为 l 的代表不同组件安全属性的字符串,具体形式为

$$\mathcal{P} = \{p_1, p_2, \dots, p_n\} \subset \{0, 1\}^l \setminus \{0^l\}.$$

组件用三元组 (id, χ, p) 表示, id 是组件的 ID, χ 是组件全部度量变量按照组件度量算法(参照 3.1.2 节)得到的度量值, p 是组件经过 T 评估满足的安全属性, $p \in \mathcal{P}$. T 为组件 (id, χ, p) 颁发的属性证书记为

$$cert(T, id, \chi, p) = (id, \chi, p, Sign(SK_T, (id, \chi, p))) \quad (2)$$

为了便于组件属性撤销查询, T 周期性颁发组件属性撤销列表 CRL,同时提供在线证书查询服务 OCSP 等.

图 2(b)为用户平台的体系结构,可信服务层、虚拟机监控层、可信硬件层共同构成了用户可信计算平台的可信计算基(TCB),TCB 的上面是运行各个相互隔离的隔间(Compartment),Compartment 可以采用 Xen 提供的隔离虚拟机,或 NGSCB 提供的保护分区,或微内核系统实现,我们的系统使用 Xen 可信平台上的虚拟机实现 Compartment. 远程证明时,首先证明 TCB 的可信,然后证明 Compartment 中 RAI 实例关联的组件安全属性. 可信服务层提供了组件证明必须的存储、会话、度量管理等服务.

是以组件为证明单位的体系结构图,系统包含组件生产厂商、用户(简记为 U)和证书发布权威(T). 组件生产厂商生产和发布组件,提供组件度量类别和度量变量(参考 3.1.1 节);证书发布权威机构评估组件安全属性,发布、验证和撤销组件属性证书;用户平台由主机系统和 TPM/TCM 安全模块组成,组件属性证明和可信计算环境的更新证明发生在用户平台安全通信过程中.

(1) MA(Measurement Agent):进行平台组件的完整性度量,组件更新时对被更新的组件重新度量.

(2) CM(Component Manager):管理系统中各个组件的安装注册、删除卸载和更新升级等,保存有注册组件的组件属性证书. CM 负责处理 Compartment 中组件的更新请求,如果组件的安全属性发生变化,则触发 MA 对组件重新度量,确保组件满足属性证书中所声明的安全属性. CM 用组件列表来保存组件注册信息和组件属性证书信息,多个 Compartment 拥有相同的组件,CM 共享该组件列表项,若组件发生更新,CM 使用 COW(Copy On Write)机制修改组件列表项.

(3) TSMG(TLS Session ManaGer):管理系统中全部远程证明 TLS 会话实例 RA ,每个远程证明实例 $RAI \in RA$ 总是绑定一个会话关联组件集合 C ,这些会话关联组件为叶节点组成一棵会话组件树. Compartment 中 $\forall c \in C$,如果组件 c 发生更新,CM 会检查更新后的组件 c 的安全属性是否发生改变,如果改变则通知 TSMG 进行更新证明,验证成功则重用原有的 SSL 会话,但使用新的会话密钥.

(4) TCMG(TPM/TCM Context ManaGer):TPM/TCM 的上下文管理负责维护各个 Compartment 的 TPM/TCM 操作环境,包括 Compartment

关联的密钥、会话、数据以及各个 Compartment 绑定的物理 PCR, 各个 RAI 关联的 VPCR (Virtual PCR). Compartment 每创建一个 RAI 时, TCMG 在相应的 Compartment 的上下文中创建一个 VPCR 用来描述 RAI 关联的组件配置和状态, VPCR 值为组件会话树的根节点扩展物理 PCR 后的值.

2.3 证明流程概述

图 3 描述了可信计算环境证明的生命周期过程. 可信计算平台每创建一个 Compartment 时,

TCMG 就为该 Compartment 创建一个 TPM/TCM 上下文 (context). 当发起一个新的 TLS 会话时, TCMG 需要为该会话分配保存组件配置和状态的 VPCR, TSMG 则建立与该会话安全相关的会话组件树, 会话组件树的组件用来衡量 RAI 会话是否满足某种安全属性. 当一个 RAI 会话实例执行时, 将触发 MA 对会话关联组件进行度量, 度量完成后 TSMG 将计算会话组件树, 然后 TPM/TCM 对证明数据签名, 最后 RAI 运行扩展了远程证明的 TLS 协议完成证明.

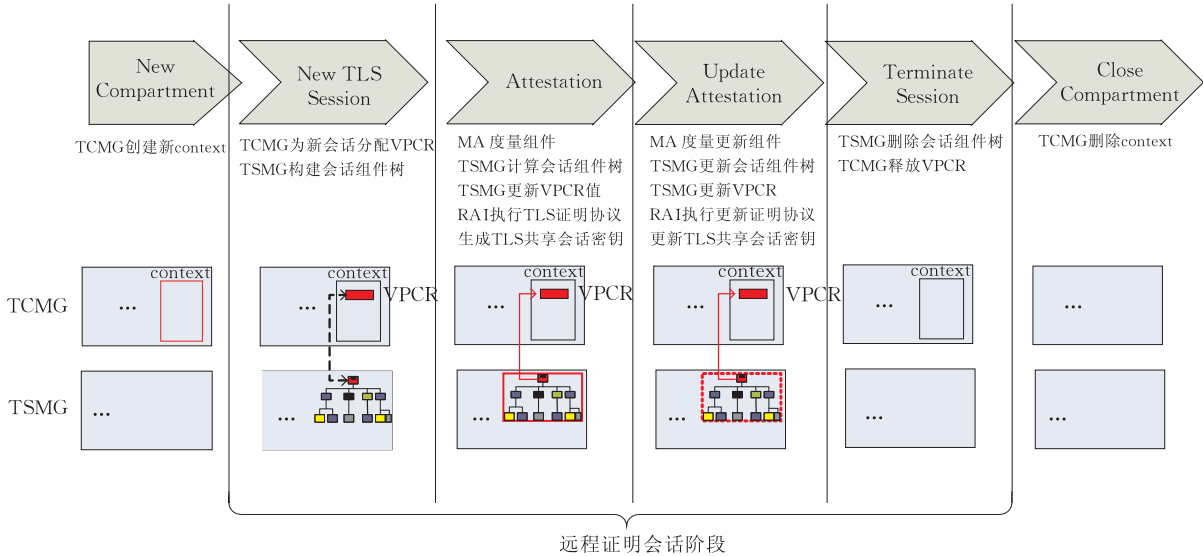


图 3 Multi-RAI 证明生命周期

可信计算环境的证明周期中, Multi-RAI 证明主要包括 3 个主要步骤:

(1) 组件度量. 由 MA 执行的对组件的度量变量进行度量、评估的算法. MA 度量的组件应在 CM 保存的组件属性证书包含的度量变量范围之内, MA 完成度量后使用组件度量属性证书验证组件配置状态, 如果二者不符, 则由 CM 通知系统管理员组件已经不符合组件属性证书声明的安全属性.

(2) 会话组件树计算. RAI 会话要求满足安全属性 p , CM 将会话安全属性映射为平台组件安全属性, 如 $p \Rightarrow \bigwedge_{i=1}^n p_i = p_1 \wedge \dots \wedge p_n$, p_i 是组件 c_i 的安全属性. TSMG 为这些与会话相关组件创建组件树, 被度量的组件经过验证后, 将用组件度量值 (或属性值) 扩展计算会话组件树.

(3) Multi-RAI 证明. 包括 RAI 证明和 RAI 更新证明.

RAI 证明是首次建立远程证明网络通信时必须进行计算环境可信性的证明. RAI 会话实例证明

协议建立在 TLS(SSL) 协议的基础上, 在证明用户身份、密钥协商的同时, 证明组件安全属性和 TPM/TCM 平台身份.

RAI 更新证明发生在 RAI 证明之后, 是对可信计算环境改变后的状态进行证明. 无论是用户合法的组件更新, 还是恶意攻击导致组件配置和状态的变化, 都会导致可信计算环境变化, 此时将无法保障原有 RAI 会话运行环境是否可信, 组件更新时 CM 将驱动 MA 重新度量更新组件, TSMG 进行更新组件证明.

3 具体方案

第 2.3 节描述了可信计算环境证明总体过程, 本节我们将提供具体的实现算法和协议, 来保证通信双方的计算环境可信. 可信计算环境证明方案 $\Gamma = (G, M, A, A_v, U, U_v)$ 是由一系列概率 Oracle^[19] 算法构成. 这些算法运行在组件度量、会话组件树计算、Multi-RAI 证明等主要步骤中, 其中密钥生成算

有正在运行的组件将退出,更新完毕后重新运行更新后的组件,IMA 度量方法就能对这类更新进行加载时度量. 组件动态更新(热更新)则是正在运行的组件不退出,运行时动态改变度量变量的状态,组件打补丁、安装插件、在线升级等是常见的组件动态更新方式.

组件动态更新时,TPM/TCM 已经完成了组件状态度量. 然而 TPM/TCM 作为一个防篡改硬件,不提供 TPM/TCM 度量状态回滚(roll back)机制,无法重新度量反映更新后的状态,目前 TCG 的安全框架也没有提出动态更新的相应解决方法.

3.1.2 组件度量

可信计算环境主要确保运行环境中硬件、固件和软件的真实性的,即组件的真实性. 要度量某个组件的完整性,只需要以组件 id 请求 MA, MA 对内存中运行的组件进程、依赖的内核模块和依赖的动态链接库进行度量,即对组件的全体度量变量进行度量,MA 度量输出组件度量值 χ 和度量日志 log .

对于某个标识为 id 的组件 c 而言,在组件发布时就确定好它的全部度量变量,设 $MC=\{mc_1, mc_2, \cdots, mc_k\}$ 是它的度量变量集合,则组件度量的算法描述如下.

算法. $M(id)$.
输入: $MC=\{mc_1, \cdots, mc_k\}$
输出: 如果执行成功,返回 log, χ
否则,返回错误代码
执行过程:
1. $log := \{ \}$
2. MA 查找组件进程是否运行,若没有运行,则返回错误.
3. MA 度量组件 c
a. $\chi := 0^{160}$
b. For each $j=1$ to k do
i. MA 检查度量变量 mc_j ,计算度量值 $\omega_j := \mathcal{H}(mc_j)$
ii. 添加度量日志, $log := log \cup \{(desc_j, \omega_j)\}$,
 $desc_j$ 为 mc_j 的具体描述信息
iii. 更新组件度量值, $\chi := \chi \cdot \omega_j$
c. return χ
4. return log

由组件度量算法可以知道,组件 c 的度量结果 $\chi := 0^{160} \cdot \omega_1 \cdot \omega_2 \cdot \cdots \cdot \omega_k$,而算法简记为 $M(id) := \{log, \chi\}$.

3.2 会话组件树计算

一般来说,TPM/TCM 芯片内部包含 24 个不可篡改的平台配置寄存器 PCR,PCR 只能通过扩展操作更新寄存器值. 我们的原型系统中,新启动一个

Compartment 时,TCMG 从 PCR16~23 中选择一个物理 PCR 与之关联,用来存储 Compartment 的组件度量结果. 当 $RAI[a]$ 实例进行远程证明时,将扩展 Compartment 的物理 PCR,由于多个 RAI 共享同一物理 PCR,好像为每个 RAI 虚拟一个 PCR 寄存器一样,因此我们称实例 $RAI[a]$ 扩展后的 PCR 为该 RAI 实例的 VPCR(Virtual PCR),VPCR 值为 $VPCR[a]$. 本方案中最多能够同时运行 8 个 Compartment,具体的物理 PCR 分配见表 1.

表 1 TPM/TCM 的物理 PCR 分配表

PCR Index	PCR Usage
0~7	CRTM, BIOS, Motherboard, BootLoader 等度量结果
8	Hypervisor 层度量结果
9	可信服务层(MA, CM, TSMG 等)度量结果
10~15	系统运行时度量
16~23	Compartment 组件度量

RAI 要求平台满足一定的安全属性,亦即要求与该会话相关的组件满足一定的安全属性. 我们将这些组件构建成一棵二叉杂凑树来表示 RAI 关联的计算环境,这棵树称之为会话组件树. 一个新的 RAI 会话发起时,通信双方就交换安全策略协商好要证明的 RAI 组件,然后 TSMG 创建 RAI 会话组件树,TCMG 新建 VPCR 用来保存会话证明结果. 当组件配置和状态发生变化时,会话组件树能够快速更新 RAI 组件状态实现更新证明.

3.2.1 会话组件树

会话组件树是以组件为叶节点的杂凑树(Merkle Hash Tree^[21], MHT),在二叉杂凑树中(如图 5 所示),叶节点是 RAI 关联的组件,包含组件的无碰撞杂凑值 χ 、安全属性值 $p \in \mathcal{P}$,叶节点的 χ 值按照上一节的组件度量算法 $M(id) := \{log, \chi\}$ 计算得到. TSMG 根据组件属性证书对 log 中 MC 度量值进行验证,验证通过赋予组件安全属性 p ,否则安全属性 $p = 0^{160}$. RAI 组件对象构成了树

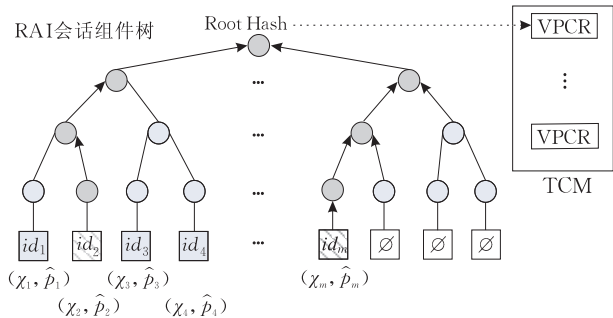


图 5 RAI 会话组件树

$MHT(id_1, \dots, id_m)$, 树节点 n 的存储的键值为 $k[n] = (\chi, p)$, 节点 n 的标识 $l[n]$ 递归定义为 $l[n] = k[n]$, 其中 $k[n]$ 是节点 $v[n] = (h_L, k[n], h_R)$ 的键值, $k[n] = \mathcal{H}(l_L \parallel l_R)$.

h_L 和 h_R 分别是节点 n 的左子女节点和右子女节点, 节点 n 的键值是左、右子树节点值连接的杂凑值, 组件树的根节点 RAI 所有组件对象的无碰撞杂凑值. 计算出组件会话树根节点值, 然后扩展 TPM/TCM 相应的 PCR. 会话组件树的节点、树根保存在 TSMG 运行空间中, 且只能够被 TSMG 操作. 如果 RAI 组件个数为 $m = 2^l$, 那么会话组件树是一个高度为 $l+1$ 的满二叉树, 若 $2^{l-1} < m < 2^l$, 会话组件树添加 $2^l - m$ 个空节点 \emptyset 构成满二叉树.

RAI 会话实例的终端安全属性和配置状态用会话组件树描述, 只要系统中 RAI 组件的任何度量变量发生改变, 都将会反映到这棵会话组件树上来. TCMG 再用根节点值扩展相应的 TPM/TCM 的物理 PCR, 那么 TPM/TCM 就能够标识各个 Compartment 的多个 RAI 会话状态. 会话组件树很好地解决了会话相关组件的状态表示, 并且便于组件动态更新状态标识. 会话组件树的组件更新仅仅需要更新组件节点到 Root Hash 路径上全部节点值, 就能反映出更新后的状态.

3.2.2 组件树计算

以组件为叶节点的 RAI 会话组件树计算记为 $\mathcal{H}(MHT(id_1, \dots, id_m))$, 本方案提供两种会话组件树的计算方法, 一是使用度量值 χ 计算会话组件树, 另一种是使用组件属性值 $p \in \mathcal{P}$ 计算组件会话树. 前者可用于直接二进制的远程证明, 用应用程序的二进制杂凑值来表示可信计算环境, 这种方法简单易行, 但存在冗余证明和平台配置隐私泄露的缺陷. 后者直接基于属性进行远程证明, 证明灵活易于扩展, 但需要颁发属性证书和验证属性是否撤销.

$$root = \mathcal{H}(MHT(id_1, \dots, id_m)) \Rightarrow$$

$$(\mathcal{H}(MHT(\chi_1, \dots, \chi_m)), \mathcal{H}(MHT(p_1, \dots, p_m))) \quad (3)$$

如果会话组件树 $MHT(id_1, \dots, id_m)$ 有 $m(2^{l-1} < m \leq 2^l)$ 个叶节点, 那么会话组件树的高度为 $h = l+1$, 有 $2^l - m$ 个空节点 \emptyset 构成满二叉树, 空节点的度量值为 ϕ , 定义空节点计算规则:

- (1) $\phi \cdot \phi = \phi$;
- (2) $\phi \cdot \chi = \chi \cdot \phi = \chi$.

组件会话树的度量值和属性值计算方法为

$$\mathcal{H}(MHT(id_1, \dots, id_m)) = \mathcal{H}(MHT(\chi_1, \dots, \chi_m)) =$$

$$\mathcal{H}(MHT(\chi_1, \dots, \chi_m, \underbrace{\phi, \dots, \phi}_{2^l - m})) = f(\chi_1, \dots, \chi_m, \underbrace{\phi, \dots, \phi}_{2^l - m}; 2^l) \quad (4)$$

其中 $f(x_1, \dots, x_{2^l}; 2^l)$ 为一递归计算函数, 其函数定义如下:

$$f(x_1, \dots, x_{2^l}; 2^l) = \begin{cases} f(x_1, \dots, x_{2^{l-1}}; 2^{l-1}) \cdot f(x_{2^{l-1}+1}, \dots, x_{2^l}; 2^{l-1}), & t > 1 \\ x_1 \cdot x_2, & t = 1 \end{cases} \quad (5)$$

由式(3)~(5)可以计算得到根节点度量值, 同样会话组件树根节点属性值可以由式(3)~(5)计算得到

$$\mathcal{H}(MHT(id_1, \dots, id_m)) = f(p_1, \dots, p_m, \underbrace{\phi, \dots, \phi}_{2^l - m}; 2^l),$$

计算结果最终更新相应的 VPCR, $VPCR = PCR \cdot root$, 组件度量值和属性值验证也按照相同的方法进行. RAI 某个组件发生改变, 不用重新计算整个会话组件树, 只需要更新该组件节点到根节点路径上的节点值, 亦即更新序列 s 所描述的从节点 $v = (h_L, k, h_R)$ 到根节点路径上的全部节点值.

$$s = (h_L, k, h_R; k_1, h_1; k_2, h_2; \dots; k_r, h_r) \quad (6)$$

除了使用普通的杂凑树构建会话组件树外, 可以借鉴虚拟单调计数器认证查找树^[22]的构造方法, 采用认证查找树 (Authenticated Search Tree^[23]) 构建 RAI 会话组件树 $AST(id_1, \dots, id_m)$. 以组件 ID 为关键字 (或称键值) 建立认证查找树, 认证查找树的每个节点 n 都有唯一一个查找键值 $k[n] = (id, \chi, p)$, 如果 n_L 是节点 n 的左子女节点, 则 $k[n_L] < k[n]$, 如果 n_R 是节点 n 的右子女节点, 则 $k[n] < k[n_R]$. $AST(id_1, \dots, id_m)$ 树的节点 n 的标识 $l[n]$ 递归定义为 $l[n] = H(v[n])$, $v[n] = (h_L, k[n], h_R)$. 如果节点 n 的左子女节点不存在, 则 $h_L = \text{nil}$, 右子女节点也一样. $AST(id_1, \dots, id_m)$ 树节点 $v = (h_L, k, h_R)$ 到根结点 v_r 按照序列式(6)进行扩展计算. 令 $l_0 = \mathcal{H}(v)$, 对于 $0 < j \leq r$, 认证查找树节点可以进行如下递归计算:

$$l_j = \mathcal{H}(v_j), \quad v_j = \begin{cases} (l_{j-1}, k_j, h_j), & k < k_j \\ (h_j, k_j, l_{j-1}), & k_j < k \end{cases} \quad (7)$$

认证查找树是以组件 ID 为节点键值构建, 而杂凑二叉树是以组件度量值 (或属性值) 为叶结点构建, 两种组件会话树方式都可以描述 RAI 会话相关组件的运行状态. RAI 证明会话开始建立时, 双方交换证明策略, 其中包含了 RAI 要证明的组件 ID

(或安全属性), TSMG 根据组件 ID 建立组件会话树, 验证时, Requester 则通过证明策略中的组件 ID 构建组件会话树进行组件运行状态和属性的验证.

3.3 Multi-RAI 证明

Compartment 启动时, TCMG 就分配一个物理 PCR 与之关联, TCMG 必须能够完成 Multi-RAI 的并行证明, 在未使用隔离技术的单个系统 (例如 Windows 或 Linux 系统), 多个实例并发证明的问题将更为突出. Multi-RAI 证明除了满足并发性外, 还必须满足证明的一致性和防重放攻击.

RAI 组件状态最终以组件树的形式聚集为 VPCR 值, 设实例 $RAI[a]$ 的组件状态聚集到组件树根节点值为 θ , TCMG 扩展相应的物理 PCR, 得到 $RAI[a]$ 的 VPCR 值, $VPCR[a] = PCR \cdot \theta$. 组件度量日志、RAI 会话组件树、根节点值 θ 和 VPCR 值共同构成了证明状态数据 D . TCMG 进行物理

PCR 扩展后, 会相应地增加单调计数器的值, 用于保证证明状态的一致性和防止恶意攻击者和程序的重放攻击.

RAI 实例建立时, TCMG 会为 RAI 生成证明凭证, 以后每一次 RAI 组件状态更新, TCMG 会生成相应的更新凭证. 证明凭证和更新凭证都是 TCMG 使用算法 $A(i, a, D, t)$ 生成, 凭证格式为 $[\theta_i, PCR, n, t, Sign(sk, H(n \parallel t \parallel PCR))]$, 证明凭证和更新凭证的区别是在于随机数 n , 证明凭证中 n 是 Requester 发送的 Nonce, 而更新凭证为 TPM/TCM 内部生成的随机数. Compartment 的多个 RAI 并发的扩展同一物理 PCR, 这样就构成一条与单调计数器关联的更新凭证链 (如图 6 所示), 对于 RAI 实例 a 建立初始会话时, 只需要提供时刻 t_0 时的证明凭证和它之前 $t_0 - 1$ 的更新凭证 (凭证 $RAI_ID \neq a$) 就能证明 RAI 实例运行环境的可信.

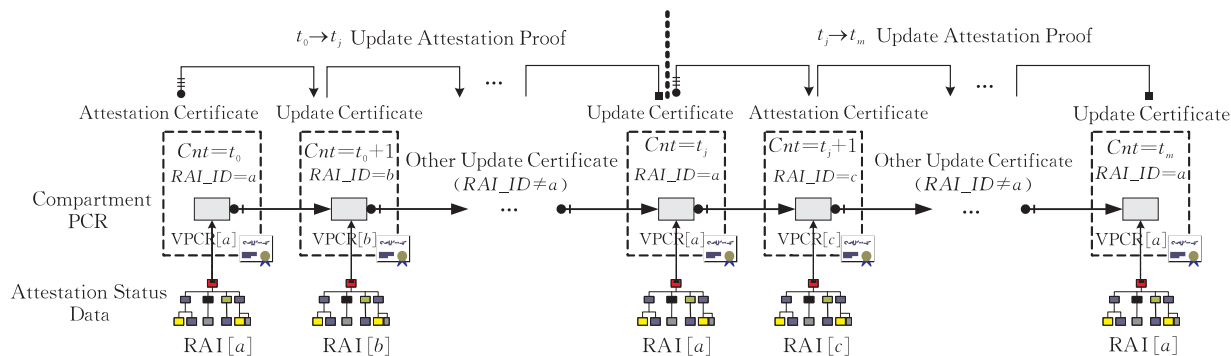


图 6 Multi-RAI 更新证明证据

当需要证明 t_m 时刻组件更新后的状态可信, TSMG 只需查找最近的一次 $RAI[a]$ 凭证 u_j , 提供更新凭证证据 $f = (u_j, u_{j+1}, \dots, u_m) = U(i, a, D_j, D_m)$. 验证时 Requester 首先重构时刻 t_m 的证明状态数据, 主要是根据证明策略请求的组件 ID 建立会话组件树, 验证根节点 θ_m 可信, 然后验证更新凭证链有效. 这种方法有效地防止了 $RAI[a]$ 重放攻击, 因为攻击者重放 t_k 时刻的更新证明, 攻击者必须满足如下两个条件才能成功: (1) 伪造时刻 t_k ($t_k = t_m$) 的更新凭证; (2) 伪造一条从 $t_j \rightarrow t_m$ 的更新凭证链. 这就要求攻击者能够破解无碰撞杂凑函数和 TPM/TCM 的 RSA 签名.

RAI 证明与 TLS/SSL 协议的认证方式相同, 也分为单向证明和双向证明, 双向证明不过是 RAI 在 Requester 一方的重用, 本文主要探讨 RAI 单向证明. RAI 证明时, 首先证明系统 TCB 的可信, 采用二进制证明就能满足要求, IMA 度量体系^[4]和 TLC 原型系统^[1]实现了这种方法, 本系统中, 即证

明 TCB 启动时各个部件及 PCR0~9 的安全性. 然后再证明与 RAI 直接关联的用组件表示的可信计算环境的安全性, 证明时 TCMG 执行 $A(i, a, D, t)$ 算法为 RAI 实例生成证明凭证, TSMG 借助证明凭证和证明状态数据 D 进行证明. 由于操作系统、应用软件自身缺陷和漏洞的暴露以及攻击技术的提高, 原有的组件配置将不再满足某一安全属性, 原有颁发的属性证书将无效, 证书发布权威机构将予以撤销. 一般来说, 验证组件属性证书是否被撤销有两种方法: TCB 本地验证, CM 验证管理的属性证书是否有效, 如果运行的组件的属性证书已经被撤销, 立即通知系统管理员, 通知 Requester 某组件运行状态不安全. 另一种是远程验证, Requester 以 (id, p) 查询证书权威机构, 验证属性证书是否被撤销. 由于存储在本地用于验证的 CRL 存在被替换和版本回滚 (Version-Rollback) 的危险, 文献^[9]提供了一种 TPM 封装存储 CRL, 定期更新 CRL 的本地验证属性证书的方法. 为了降低本地系统的复杂性,

我们采用远程验证组件属性的方法,其缺陷是在仅进行证明组件属性的 RAI 证明中,降低了平台组件的隐私性。

3.3.1 RAI 证明

前面对 RAI 证明的相关问题及解决方法进行了概述,下面具体阐述 RAI 证明流程,为简化起见将忽略对 TCB 状态的证明。

1. Requester \rightarrow RAI: $nonce, attestpolicy$.

2. RAI 处理:解析 $attestpolicy$

(1) MA 度量组件: $M(id) := \{log, \chi\}$;

(2) TSMG 构建 RAI 会话组件树,形成证明状态数据 D ;

(3) TCMG 请求 TPM/TCM 生成证明凭证, $s \leftarrow A(i, a, D_j, t_j)$, PCR 初值为 $t_j - 1$ 的 VPCR 值,记为 PCR_0 .

$s = [\theta_j, VPCR_j, nonce, t_j, Sign(sk, H(nonce \parallel t_j \parallel VPCR_j))]$

(8)

3. RAI \rightarrow Requester: s, log, PCR_0 .

4. Requester 验证

(1) Requester 验证证明凭证, $(VPCR_j, t_j) \leftarrow A_V(s, i, a, pk)$;

(2) 验证组件属性是否被撤销,然后重构会话组件树,计算根节点 θ'_j ,验证 $\theta_j \stackrel{?}{=} \theta'_j$;

(3) 验证 $VPCR_j \stackrel{?}{=} PCR_0 \cdot \theta'_j$;

(4) 前两步都验证成功,Requester 对 t_j 和 $VPCR_j$ 颁发确认凭证。

RAI 实例收到 Requester 的确认凭证后, TSMG 会将 t_j 和 $VPCR_j$ 标识的那个证明凭证作为最近的一次更新凭证,也就是说这个证明凭证所描述的状态是 RAI 实例的计算环境的最新状态,这样以便以后的更新证明。

3.3.2 RAI 更新证明

Multi-RAI 证明建立安全会话后,随着正常的软件更新、升级,或者恶意病毒和黑客的攻击,导致会话关联组件的配置和运行状态发生改变,此时原有建立的信任关系将被破坏,因此需要重新证明。

1. RAI:更新证明状态数据和产生更新凭证证据

(1) TSMG 更新实例 $RAI[a]$ 的最近一次证明状态数据, $D_j \rightarrow D_m$,接着计算会话组件树根节点 θ_m ;

(2) TCMG 扩展 PCR, $VPCR_m = VPCR_{m-1} \cdot \theta_m$,然后生成更新凭证 $u_m \leftarrow A(i, a, D_m, t_m)$,

$u_m = [\theta_i, VPCR_m, n, t_m, Sign(sk, H(n \parallel t_m \parallel VPCR_m))]$;

(3) TSMG 生成更新凭证证据 $f \leftarrow U(i, a, D_j, D_m)$,

$f = (u_i, u_{i+1}, \dots, u_m)$.

2. RAI \rightarrow Requester: f, log .

3. Requester 验证

(1) Requester 验证更新凭证 u_m , $(VPCR_m, t_m) \leftarrow A_V(u_m, i, a, pk)$

(2) 验证更新的组件属性是否被撤销,计算出会话组件树根节点 θ'_m ,验证 $\theta_m \stackrel{?}{=} \theta'_m$ 和 $VPCR_m \stackrel{?}{=} VPCR_{m-1} \cdot \theta'_m$.

(3) 验证更新凭证证据, $\{(VPCR_j, t_j), (VPCR_m, t_m)\} \leftarrow U_V(f, i, a, pk)$.

(4) 如果都验证成功,Requester 对 t_m 和 $VPCR_m$ 颁发确认凭证。

4. RAI:收到 Requester 的确认凭证后,将凭证 u_m 作为最近的一次更新凭证。

上述过程中如果 RAI 更新证明验证失败, Requester 将切断 RAI 的 SSL 网络连接. 如果更新证明仅仅是证明属性,RAI 实例的会话组件尽管配置发生改变,但安全属性没有发生改变,此时 RAI 没有必要进行更新证明。

3.3.3 安全性分析

在 \mathcal{H} 是无碰撞杂凑函数、 $Sign$ 是安全的数字签名方案的前提下,下面简要的说明可信计算环境证明方案 $\Gamma = (G, M, A, A_V, U, U_V)$ 是安全的。

首先,令 u 和 u' 都是更新凭证,满足 $t \leftarrow A_V(u, i, a, pk)$ 和 $t \leftarrow A_V(u', i, a', pk)$, 且 $a \neq a'$, 由于 TPM/TCM 签名方案假定是安全的,Compartment i 在时刻 t 只能返回一个签名,因此这两个凭证 u 和 u' 是完全相同的,凭证包含时刻 t 的 PCR 值 $VPCR_t$. 对于 Compartment i 而言 $t - 1$ 存在唯一更新凭证 u_{t-1} , 凭证包含 $VPCR_{t-1}$. $VPCR_t = VPCR_{t-1} \cdot \theta_t$, 所以更新凭证 u 和 u' 关联的会话组件树的根节点 θ_t 相等,唯一的可能便是 RAI 实例 a' 可以破解无碰撞杂凑函数 \mathcal{H} .

其次,令 f 是更新凭证证据, u 为更新凭证,满足 $\{(VPCR_j, t_j), (VPCR_m, t_m)\} \leftarrow U_V(f, i, a, pk)$, $t_h \leftarrow A_V(u, i, a, pk)$, $t_j < t_h \leq t_m$. $f = (u_j, \dots, u_m)$ 包含了时刻 t_m 的 $VPCR_m$ 不可伪造签名凭证 u_m , 更新 Oracle 将 PCR 值由 $VPCR_{m-1}$ 扩展为 $VPCR_m$, $VPCR_m = VPCR_{m-1} \cdot \theta_m$. 重复上述过程, f 包含时刻 t_h 的 $VPCR_h$ 不可伪造签名凭证 u_h , $VPCR_m = VPCR_h \cdot \theta_{h+1} \cdot \dots \cdot \theta_m$, 由于 \mathcal{H} 为无碰撞散列函数,在时刻 t_h , TPM/TCM 仅仅只能返回唯一的更新凭证,所以凭证 u_h 和 u 必定存在一个是伪造的。

由上述讨论可知,在 \mathcal{H} 是无碰撞杂凑函数、 $Sign$ 是安全的数字签名方案下,Multi-RAI 证明和更新证明是安全的。

3.3.4 效率分析

假定 Compartment 中共计有 N 个组件,同时有 I 个 RAI 会话实例运行于 Compartment 中,在最坏的情况下,每个 RAI 实例平均拥有 N/I 个 RAI

会话组件. RAI 会话树的每个节点计算、PCR 扩展等等, TPM/TCM 都必须作为一个原子操作, 不允许被中断. 这样对于实例 $RAI[a]$ 的一个更新凭证证据而言, TPM/TCM 要执行 $O(I(1+\log(N/I)))$ 个原子操作.

定义 ω 为实例 $RAI[a]$ 在一个时间单位内发生组件更新, 执行更新证明的概率, 简化起见, 假定每个组件更新概率都为 ω . γ 为 TPM/TCM 执行一个原子操作指令的平均计算时间. J 为一个 RAI 组件时间在同一个时间单位上请求更新的组件个数期望值 ($0 \leq J \leq N/I$), T 为同一个 RAI 实例连续两次请求更新证明的时间间隔期望值.

对于 Multi-RAI 证明, 更新证据计算过程 $f \leftarrow U(i, a, D_j, D_m)$ 是 $[t_j, t_m]$ 时间段内计算量最大、最耗时的步骤. 产生更新证明证据的每个更新凭证 $u_k (j \leq k \leq m)$ 计算上要耗费 $1 + J + \log(N/I)$ 个原子操作, 其中 J 为组件 Hash 值更新计算原子操作时间, 1 为更新物理 PCR 的原子操作时间.

$$T = \gamma I(1 + J + \log(N/I)) \tag{9}$$

同一 Compartment 中, RAI 实例在时间 T 内组件不更新, 不执行更新证明的概率为 $(1 - \omega)^T$, 所以 RAI 实例在同一时间有组件更新的概率为 $1 - (1 - \omega)^T$, 则平均有 $\frac{N}{I}(1 - (1 - \omega)^T)$ 个组件同时更新, 因此

$$J = \frac{N}{I}(1 - (1 - \omega)^T) \approx \omega NT/I \tag{10}$$

同一个 Compartment 有多个 RAI 同时请求更新证明, 即同时有多个 RAI 的会话关联组件发生更新, 同时更新时, TCMG 将按顺序依次更新 PCR 进行证明, 在同一单位时间内, TCMG 只能生成一个 RAI 实例的更新凭证, 其他的 RAI 将进入等待队列. 若 X 为执行更新证明概率为 ω 的时间随机变量, X 服从几何分布, $E(X) = 1/\omega$. 那么平均 $1/\omega$ 个单位时间, RAI 执行一次更新协议, 平均 $(1 - \omega)/\omega$ 个单位时间没有 RAI 执行更新协议. 因此要求同一个 RAI 两次更新请求间隔时间期望满足

$$T \leq (1 - \omega)/\omega \tag{11}$$

将式(10)代入式(9)得到 $T = \gamma I + \omega \gamma NT + \gamma I \log(N/I)$, 因此

$$T = \frac{\gamma I(1 + \log(N/I))}{1 - \omega \gamma N} \tag{12}$$

然后对式(12)进行求导计算 T 的最大值, $\frac{dT}{dI} = \frac{\gamma \log(N/I)}{1 - \omega \gamma N}$, 因此 I 满足 $\frac{\gamma \log(N/I)}{1 - \omega \gamma N} = 0, I = N$. 代

入式(12)中得到

$$T = \frac{\gamma N}{1 - \omega \gamma N} \tag{13}$$

同一 RAI 实例更新证明的平均间隔时间与组件更新的概率、TPM/TCM 的原子操作处理时间和 Compartment 组件的个数密切相关, 等式(13)给出了它们之间的相互关系. TPM/TCM 原子操作受 TPM/TCM 物理芯片和硬件 I/O 效率影响, 系统的组件更新概率是一个不可预知的因素, 可以通过分隔 Compartment 的组件, 使系统满足更新证明效率要求. 将式(13)代入不等式(11)进行化简得到

$$N \leq \frac{1 - \omega}{\omega \gamma (2 - \omega)} \tag{14}$$

在系统组件更新概率和 TPM/TCM 原子操作速度固定的前提下, 我们可以按照不定式(14)限制 Compartment 中运行的组件个数, 以避免同一 Compartment 中 Multi-RAI 并发更新证明, 而影响可信服务层(包括 TSMG, TCMG, CM 等)的处理效率.

4 系统实现

我们在 Xen 隔离 Compartment 的平台上实现了 Multi-RAI 原型系统, 其体系结构参照图 2(b). 原型系统中可信硬件层使用的是符合 TPM 1.2 规范的 Nation SemiConductor TPM 安全芯片(采用国产的 TCM 安全芯片也能实现同样的功能). 系统使用开源虚拟机监控器 Xen-3.0.2 实现虚拟机监控层, Xen Hypervisor 实现各个虚拟机的强隔离. 可信服务层的 TCMG、TSMG、MA 等模块运行于虚拟机操作系统内核, 处理 Multi-RAI 的证明. 可信服务层上各个 Compartment 以相互隔离的虚拟机实现. 下面详细地介绍可信服务层的 MA、TCMG、TSMG 等服务模块.

度量代理 MA 使用 Linux 系统的 LSM(Linux Security Module^[24])机制实现, 当有可执行程序运行, 加载进入内存时, MA 内核模块度量可执行程序和相关动态连接库程序. 当 Compartment 的组件发生更新时, MA 再次度量执行的可执行程序 and 动态连接库程序. Xen Hypervisor 提供了 TPM 驱动程序, 包含 Compartment 的前端驱动(tpm-front)和虚拟机监控器的后端驱动(tpm-backend), 我们构建了 TCMG 内核模块负责与 tpm-backend 通信, 管理各个 Compartment 的 TPM 请求的 context. TCMG

模块具有扩展物理 PCR, 维护 Compartment 的 VPCR, 生成 RAI 的更新凭证等功能, TCMG 发送 TPM 命令给物理 TPM 执行实现上述功能, 例如, 发送 TPM_LoadKey, PCR_Extend, TPM_Quote 等命令给 TPM, 实现生成 RAI 更新凭证的功能. 我们进行了实验测试, 对比了可信虚拟体系结构下 TCMG 模块处理 TPM 命令和 Linux 系统下 TPM 命令执行效率, 其实验结果见图 7.

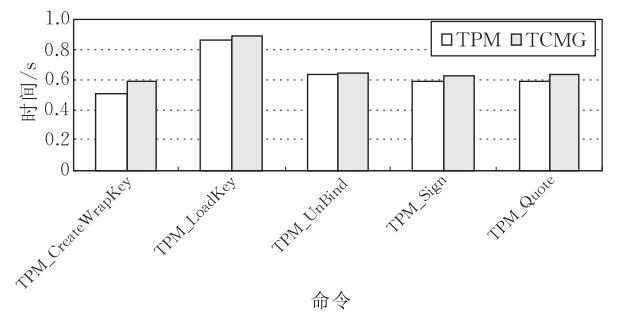


图 7 可信虚拟体系结构下 TCMG 模块处理 TPM 命令和 Linux 系统下 TPM 命令执行效率对比

RAI 的关联计算环境的组件表示是在 TSMG 中实现, TSMG 维护着各个 RAI 会话组件树数据. RAI 实例与 Requester 进行通信协商时, TSMG 新建会话组件树, 首次证明时将计算整个会话组件树, 用根节点扩展 PCR, TPM 签名生成证明凭证. 当与 RAI 关联组件发生更新, 组件管理部件将触发 TSMG 对 RAI 会话树进行更新, 通知 TCMG 生成更新凭证. 我们实现的会话组件树方法, 比 TPM 直接表示组件状态的证明方法效率要高, 图 8 为实验对比结果.

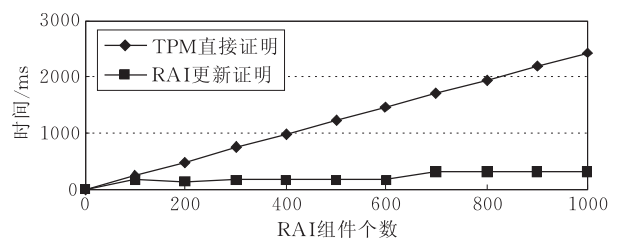


图 8 会话组件树方法与 TPM 直接证明方法效率对比

原型系统在 SSL 协议的基础上, 实现 Compartment 可信计算环境的多种证明. 最基本的证明方式是平台证明 (tcglinux 原型系统实现的证明), tcglinux 在 IMA 度量架构下, 对整个平台的配置进行证明, 这种直接证明平均通信时间为 4.932s. 根据 Multi-RAI 证明方法, 我们实现了 Multi-RAI 组件证明和 Multi-RAI 更新证明, Multi-RAI 组件证明平均通信时间 1.915s, 而更新证明平均时间为 1.677s. 由此可见, Multi-RAI 与直接证明相比, 在

SSL 通信过程中平均通信时间降低 61.2%.

5 总 结

本文深入研究了可信计算平台的状态改变引发 RAI 更新证明和多 RAI 实例的并发证明问题, 提出了 Multi-RAI 动态更新证明方法. 只要 RAI 关联组件配置和状态发生改变, 都会触发 RAI 会话组件树描述的计算环境的改变. 通过维护 VPCR, 构建更新凭证证据, 解决 Multi-RAI 的一致性、并发性和重放攻击等问题. 目前我们没有过多地关注计算平台的配置隐私性问题, 即使会话组件树使用属性值进行计算, 但在通信双方协商证明安全策略阶段还是会暴露计算平台组件构成, 这会导致攻击者对相关组件的漏洞、缺陷实施针对性攻击. 因此我们下一步的工作目标是在证明可信计算环境的同时, 保护双方计算环境的组件构成.

参 考 文 献

[1] Trusted Computing Group. TCG Architecture Overview. Specification, Revision 1.2. 28 April 2004

[2] State Password Administration Committee in China. Functionality and Interface Specification of Cryptographic Support Platform for Trusted Computing. December, 2007 (in Chinese) (中国国家密码管理局. 可信计算密码支撑平台功能与接口规范. 2007 年 12 月)

[3] Trusted Computing Group. TPM Main Part 1, Design Principles. Specification Version 1.2, Revision 62. 2 October 2003

[4] Sailer Reiner, Zhang Xiao-Lan, Jaeger Trent, van Doorn Leendert. Design and implementation of a TCG-based integrity measurement architecture//Proceedings of the 13th Unix Security Symposium. San Diego, California, 2004; 223-238

[5] Sailer Reiner, van Doorn Leendert, James P. Ward. The role of TPM in enterprise security. IBM Research Report RC 23368, October 2004

[6] Poritz Jonathan, Schunter Matthias, van Herreweghen Els, Waidner Michael. Property attestation — Scalable and privacy-friendly security assessment of peer computers. IBM Research Report RZ 3548, October 5, 2004; 223-238

[7] Sadeghi A, Stübke C. Property-based attestation for computing platforms: Caring about properties, not mechanisms//Proceedings of the New Security Paradigms Workshop, 2004; 67-77

[8] Chen Li-Qun, Landfermann Rainer, Löhr Hans et al. A protocol for property-based attestation//Proceedings of the 1st ACM Workshop on Scalable Trusted Computing. Nova Scotia Canada, 2006; 7-16

[9] Kühn Ulrich, Selhorst Marcel, Stueble Christian. Realizing

- property-based attestation and sealing with commonly available hard- and software//Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing. Alexandria, Virginia, USA, 2007
- [10] Kühn Ulrich, Kursawe Klaus, Lucks Stefan, Sadeghi Ahmad-Reza, Stübke Christian. Secure data management in trusted computing//Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES). LNCS 3659. Springer, 2005; 324-338
- [11] Dierks T, Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.1. Network Working Group RFC 4346. Internet Engineering Task Force. <http://www.ietf.org/rfc/rfc4346.txt>, Apr. 2006
- [12] Kent S, Seo K. Security Architecture for the Internet Protocol. Network Working Group RFC 4346. Obsoletes: RCF2401, Internet Engineering Task Force. <http://www.ietf.org/rfc/rfc4301.txt>, Dec. 2005
- [13] TCG Infrastructure Working Group. Reference Architecture for Interoperability (Part D). Specification Version 1.0, Revision 1. 16 June 2005
- [14] TCG Trusted Network Connect, TNC Architecture for Interoperability. Specification Version 1.0, Revision 4. 3 May 2005
- [15] Gasmi Yacine, Sadeghi Ahmad-Reza, Stewin Patrick et al. Beyond secure channels//Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing. Alexandria, Virginia, USA, 2007
- [16] Trusted Computing Group. TLS Extensions for Attestation. Specification Version 1.0, Revision 0.8. July 2004, Work in Progress
- [17] Barham P, Dragovic B, Fraser K et al. Xen and the art of virtualization//Proceedings of the 19th ACM Symposium on Operating Systems Principles. Bolton Landing, NY, 2003
- [18] Hand Steven, Warfield Andrew, Fraser Keir et al. Are virtual machine monitors microkernels done right//Proceedings of the 10th Workshop on Hot Topics in Operating Systems (HotOS-X). 2005
- [19] Bellare M, Rogaway P. Random oracles are practical: A paradigm for designing efficient protocols//Proceedings of the 1st Annual Conference on Computer and Communications Security. 1993
- [20] TCG Infrastructure Working Group (IWG). TCG Infrastructure Workgroup Subject Key Attestation Evidence Extension. Specification Version 1.0, Revision 7, June 2005
- [21] Merkle Ralph C. A certified digital signature//Proceedings on Advances in Cryptology 1989. Santa Barbara, California, USA, 1989
- [22] van Dijk M, Sarmenta L F G, O'Donnell C W, Devadas S. Offline untrusted storage with immediate detection of forking and replay attacks//Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing (STC'07). Alexandria, Virginia, USA, 2007
- [23] Buldas A, Laud P, Lipmaa H. Accountable certificate management using undeniable attestations//Proceedings of the 7th ACM Conference on Computer and Communications Security. Athens, Greece, 2000; 9-17
- [24] Chris Wright, Crispin Cowan, Stephen Smalley, James Morris, Greg Kroah-Hartman. Linux security modules: General security support for the Linux kernel//Proceedings of the USENIX Security Symposium. San Francisco, CA, 2002



FENG Deng-Guo, born in 1965, Ph. D., professor, Ph. D. supervisor. His mainly engaged in the research and development of network and information security.

QIN Yu, born in 1979, Ph. D. candidate. His research interests include information and network security, trust computing.

Background

The work attributes to the project "Trust Chain Establishment in Trust Platform", which is supported by the National Basic Research Program (973 Program) of China under grant No. 2007CB311202, the National Science Foundation of China under grant No. 60673083, and the National High Technology Research and Development Program of China (863 Plan) under grant No. 2007AA01Z412.

The attestation on trust computing environment becomes one of the imminence requirements in the distribution application security, which is widely focused on by so many research centers and institutes. The recent methods like TCG attestation, direct attestation, and property-based attestation etc. solve the static attestation on the running state of computing

environment, and also protect the platform configuration privacy, but the research about dynamic update attestation and concurrent attestation is not covered recently. In this paper, the authors analyze the problems of dynamic characteristic, concurrency and consistency for multiple remote attestation instances (Multi-RAI), and then propose a Multi-RAI dynamic and concurrent attestation scheme in trust computing environment which is comprised of component integrity algorithm, session component tree calculation and Multi-RAI attestation protocol. At last the prototype for the proof-of-concept is implemented to prove the feasibility and performance, and the method security and efficiency are also analyzed.