软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(10):2782−2796 [doi: 10.13328/j.cnki.jos.005174]

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

虚拟可信平台模块动态信任扩展方法*

余发江[1,2], 陈 列[1], 张焕国[1,2]

[1](武汉大学 计算机学院,湖北 武汉 430072)

[2](软件工程国家重点实验室(武汉大学),湖北 武汉 430072)

通讯作者: 余发江, E-mail: fjyu@whu.edu.cn

摘 要: 将可信计算技术应用到虚拟计算系统中,可以在云计算、网络功能虚拟化(network function virtualization,简称NFV)等场景下,提供基于硬件的可信保护功能.软件实现的虚拟可信平台模块(virtual trused platform module,简称vTPM)基于一个物理TPM(physical TPM,简称pTPM),可让每个虚拟机拥有自己专属的TPM,但需要将对pTPM的信任扩展到vTPM上.现有方法主要采用证书链来进行扩展,但在虚拟机及其vTPM被迁移后,需要重新申请vTPM的身份密钥证书,可能会存在大量的短命证书,成本较高,且不能及时撤销旧pTPM对vTPM的信任扩展,也不能提供前向安全保证.提出了一种vTPM动态信任扩展(dynamic trust extension,简称DTE)方法,以满足虚拟机频繁迁移的需求.DTE将vTPM看作是pTPM的一个代理,vTPM每次进行远程证明时,需从一个认证服务器(authenticaiton server,简称AS)处获得一个有效的时间令牌.DTE在vTPM和pTPM之间建立了紧密的安全绑定关系,同时又能明显区分两种不同安全强度的TPM.在DTE里,vTPM被迁移后,无需重新获取身份秘钥证书,旧pTPM可及时撤销对vTPM的信任扩展,而且DTE可提供前向安全性.从原型系统及其性能测试与分析来看,DTE是可行的.

关键词: 可信计算;可信平台模块(TPM);虚拟可信平台模块(vTPM);信任扩展

中图法分类号: TP309

中文引用格式: 余发江,陈列,张焕国.虚拟可信平台模块动态信任扩展方法.软件学报,2017,28(10):2782−2796. http://www.jos.org.cn/1000-9825/5174.htm

英文引用格式: Yu FJ, Chen L, Zhang HG. Virtual trusted platform module dynamic trust extension. Ruan Jian Xue Bao/Journal of Software, 2017,28(10):2782−2796 (in Chinese). http://www.jos.org.cn/1000-9825/5174.htm

Virtual Trusted Platform Module Dynamic Trust Extension

YU Fa-Jiang[1,2], CHEN Lie[1], ZHANG Huan-Guo[1,2]

[1](School of Computer, Wuhan University, Wuhan 430072, China)

[2](State Key Laboratory of Software Engineering (Wuhan University), Wuhan 430072, China)

Abstract: The integration of trusted computing into virtual computing system can enable the hardware-based protection of trustworthiness in application areas such as cloud computing and network function virtualization (NFV). In a physical trusted platform module (pTPM) based virtual trusted platform module (vTPM), each virtual machine (VM) can be viewed as having its own private TPM. However, it is necessary to extend the trustworthiness of pTPM to vTPM so that a challenger can believe the vTPM is the root of trust of the VM. The existing techniques mainly use a certificate chain to build a trust link from pTPM to vTPM. But if these techniques were deployed in the scenario with frequent vTPM migrations, there would be very high cost of reacquiring new certificates for the migrated vTPM, moreover, pTPM couldn't revoke its trust extension in real time, and they couldn't provide forward security. This paper presents an approach of vTPM dynamic trust extension (DTE) to satisfy the requirements of frequent migrations. With DTE, vTPM is a delegation of the capability of signing attestation data from the underlying pTPM, with one valid time token issued by an authentication server (AS).

* 基金项目: 国家重点基础研究发展计划(973)(2014CB340600); 国家自然科学基金(61772384)

Foundation item: National Basic Research Program of China (973) (2014CB340600); National Natural Science Foundation of China (61772384)

收稿时间: 2016-07-25; 修改时间: 2016-09-29; 采用时间: 2016-10-27

DTE maintains a strong association between vTPM and its underlying pTPM, and has clear distinguishability between vTPM and pTPM because of the different security strength of the two types of TPM. In DTE, there is no need for vTPM to re-acquire identity key (IK) certificate(s) after migration, and pTPM can have a trust revocation in real time. Furthermore, DTE can provide forward security. Performance measurements and analysis of its prototype demonstrate that DTE is feasible.

Key words: trusted computing; trusted platform module (TPM); virtual trusted platform module (vTPM); trust extension

1 引 言

可信计算以一个硬件安全模块——可信平台模块(trusted platform module,简称 TPM)作为可信根,为宿主计算机提供平台身份认证、完整性保护和安全存储功能.除个人计算机外,TPM 被集成到服务器上,正变得越来越普遍,如 DELL PowerEdge R530 机架式服务器和 Cisco UCS B200 M4 刀片式服务器.将可信计算技术应用到基于服务器的虚拟计算系统中,可以在云计算、网络功能虚拟化(network function virtualization,简称 NFV)等场景下,提供基于硬件的可信保护功能.

在非虚拟化环境中,TPM 与宿主算机的比例是一比一[1].虚拟化技术允许多个虚拟计算机相互独立地运行在同一个物理平台上,由于资源有限,一个物理 TPM(physical TPM,简称 pTPM)不能同时供多个虚拟机使用.一种解决方案就是,为每个虚拟机提供一个软件实现的虚拟 TPM(virtual TPM,简称 vTPM)[2,3].对每个虚拟机而言,它们会觉得自己拥有一个专属的 TPM.pTPM 是整个平台的可信根,用户对 pTPM 的信任来源于对可信计算技术的认可,权威机构也对 pTPM 产品进行认证,确保其具体实现与技术规范相一致.软件实现的 vTPM 会被动态地创建和销毁,权威机构不可能对每个运行的 vTPM 实例进行一致性测评与认证,他们能做的只是对某个 vTPM 软件实现进行测评.对具体运行的 vTPM 实例的测评,可由配置有 pTPM 的宿主计算机来完成,一旦测评过,就需要将用户对 pTPM 的信任扩展到 vTPM 实例上,用户才会认可 vTPM 是虚拟机的可信根.

现有扩展方法主要是运用证书链来构建 pTPM 对 vTPM 的信任担保.如果将现有方法应用在 vTPM 频繁迁移的情景中,被迁移后的 vTPM 需重新获取证书,可能存在大量的短命证书,成本高昂.旧平台上的 pTPM 不能及时撤销它对 vTPM 的信任扩展,现有方法也不能提供前向安全保证.我们提出一种 vTPM 动态可信扩展(dynamic trust extension,简称 DTE)方法,以消除现有方法的弊端.

本文的主要贡献如下:

(1) 提出了一种适用于频繁迁移情况的 vTPM 动态可信拓展方法 DTE.DTE 在 vTPM 和 pTPM 之间建立了紧密的安全关联,DTE 将 vTPM 看作是 pTPM 进行虚拟机远程证明的代理,该代理权限的执行基于一个有效的时间令牌,令牌则需从一个认证服务器(authenticaiton server,简称 AS)处获取.DTE 能够明显地区分两种不同的 TPM,vTPM 和 pTPM 拥有不同的安全强度.当迁移发生时,旧 pTPM 可及时撤销对 vTPM 的信任授权,vTPM 无需重新获取身份秘钥(identity key,简称 IK)证书.同时,DTE 能够提供前向安全性.

(2) 基于 OpenStack 实现了针对 TPM 1.2 和 TPM 2.0 的 DTE 原型系统,从性能测试及分析结果来看,DTE 方法是可行的.

本文第 2 节介绍现有的 vTPM 信任扩展方法,分析其缺陷,提出 DTE 要达到的目标.第 3 节描述基于信任关系和可信测评的信任扩展模型.第 4 节描述 DTE 的详细过程和算法.第 5 节分析 DTE 的安全性.第 6 节介绍 DTE 原型系统的建立和性能测试与分析.第 7 节介绍相关工作.最后,第 8 节总结全文.

2 现有方法和 DTE 的目标

现有方法主要是运用证书链来对 vTPM 进行信任扩展.Berger 等人[2]用证书链将 vTPM 与 pTPM 相关联.他们提出的第 1 种机制,是在 vTPM 申请背书秘钥(endorsement key,简称 EK)证书时,用 pTPM 的身份秘钥 IK 证书进行担保.第 2 种机制,是使用 pTPM IK 证书担保 vTPM IK 证书.在一些场景里,为平衡计算资源,虚拟机可能会在不同的物理宿主机上进行迁移.当一个虚拟机被迁移到新的宿主机时,与之相关联的 vTPM 也被迁移.同时,也必须建立新 pTPM 到 vTPM 的信任扩展,而旧 pTPM 与 vTPM 之间的信任扩展需被撤销.若采用 Berger 等人

提出的第 1 种机制,迁移后的 vTPM 须基于新 pTPM 的 IK 证书重新申请 vTPM EK 证书,然后再申请 vTPM IK 证书.若使用第 2 种机制,迁移后的 vTPM 须基于新 pTPM 的 IK 证书,重新申请 vTPM IK 证书.目前还没有(certificate authority,简称 CA)专业提供平台 EK 和 IK 证书,我们假设平台证书的价格等同于(secure socket layer,简称 SSL)证书.在本文准备期间,我们查询了几个主要提供商的 SSL 证书价格,见表 1.若虚拟机频繁迁移,vTPM 也将频繁地获取 IK 证书,成本高昂.迁移之后,之前的证书将被遗弃,可能还远未到其过期时间,会有很多短命证书存在,是一种很大的浪费.另外,为了保护隐私,一个 vTPM 可能存在多个 IK 证书,这会带来更高的代价.

Table 1 The SSL certificate price comparison

表 1 SSL 证书价格对比

提供商及证书类型	1 年期价格	2 年期价格	3 年期价格
Rapid SSL 证书	$49 USD	$86 USD	$122 USD
Thawte SSL123 证书	$149 USD	$259 USD	$369 USD
DigiCert SSL Plus	$175 USD	$315 USD	$419 USD
GeoTrust True BusinessID	$199 USD	$348 USD	$498 USD
Symantec Secure Site	$399 USD	$695 USD	$995 USD

Danev 等人[4]提出了一种 vTPM 密钥层次结构,该结构有一个 pTPM 和 vTPM 之间的中间层,由一组 pTPM 的签名密钥(signing key,简称 SK)构成,用以连接 pTPM IK 和 vTPM IK.该方法也在 vTPM 和 pTPM 之间建立了绑定关系,但 pTPM SK 不能被迁移,故 vTPM 迁移之后,新的 pTPM 须重新生成 SK,vTPM 同样须重新获取 IK 证书.

在虚拟可信平台架构规范[3]中,可信计算组织(trusted computing group,简称 TCG)并没有明确定义如何创建 pTPM 和 vTPM 之间的连接关系,但 TCG 提出了一个“深度认证(deep attestation)”的概念.在远程实体完成对虚拟机的认证后,可能会继续认证其下层的虚拟机监控器(virtual machine monitor,简称 VMM)和虚拟机宿主平台,以确定它们是否足够可信,不会去破坏运行在其上的虚拟机.为了实现深度认证,远程实体需从 vTPM IK 证书扩展项中获取底层平台的信息,如 IP 地址或者统一资源标识符(uniform resource identifier,简称 URI).当虚拟机迁移发生时,vTPM 也需要获取新的 IK 证书,以包含新的宿主平台信息.

在前述的几种 vTPM 信任扩展方法中,vTPM 迁移后,它原来的 EK/IK 证书需被废除.废除证书最常用的方法就是证书撤销列表(certificate revocation list,简称 CRL).当 vTPM 被迁移时,与之关联的原 pTPM 便通知 CA,CA 将 vTPM 原有证书信息添加到 CRL 中,并对更新之后的 CRL 重新进行签名.在远程实体验证 vTPM 的远程证明报告时,先下载 CRL,检查 vTPM 当前使用的证书是否在 CRL 中.CRL 通常相当冗长,不会被频繁地下载.例如,1 周或者 1 个月被下载 1 次.因此,vTPM 的原有证书可能在迁移发生 1 个月之后才会被真正废除.在此期间,迁移后的 vTPM 仍然可以代表原 pTPM,生成虚拟机的证明报告,这是现有信任扩展方法的又一明显缺陷,pTPM 不能及时撤销它对 vTPM 的信任扩展.另外,一旦 vTPM 原有证书被加入到 CRL 中,即使在迁移之前由 vTPM 所生成的签名都将无效,故现有方法也不能提供前向安全性.

如果将现有扩展方法应用到虚拟机频繁迁移的场景中:(1) 将会造成大量的短命证书存在,成本高昂,浪费巨大;(2) pTPM 不能及时撤销它对 vTPM 的信任扩展;(3) 不能提供前向安全保证.我们提出的新方法——vTPM 动态信任扩展(DTE),应消除现有方法的这些弊端.DTE 的主要目标如下:(1) 在 vTPM 和 pTPM 之间建立紧密的安全关联;(2) 严格区分 vTPM 和 pTPM,这两种类型的 TPM 具有不同的安全强度;(3) 迁移后的 vTPM 无需重新获取 EK/IK 证书;(4) pTPM 可及时撤销对 vTPM 的信任扩展;(5) 提供前向安全性;(6) 是可行的.

3 基于可信测评的信任扩展模型

3.1 信任关系与可信测评

定义 1(信任关系). 实体 E1 信任 E2, 记作 E1 Tru E2. 信任关系具有如下性质.

(1) 自信任. E1 Tru E1.

(2) 传递性.若 E1 Tru E2, 且 E2 Tru E3, 则有 E1 Tru E3.

定义 2(可信测评). Evi_{E2} Con $\mathsf{Cri}_{E1,E2} \Rightarrow$ E1 Tru E2,其中,$\mathsf{Cri}_{E1,E2}$ 表示 E1 判断 E2 是否可信的标准,Evi_{E2} 表示与 E2 相关的证据,Con 表示证据与标准相符.

3.2 基于证书链的vTPM信任扩展模型

基于证书链的机制里所涉及的实体有权威机构 $\mathcal{I}$,认证中心 $\mathcal{CA}$,旧 pTPM $\mathcal{OM}$,新 pTPM $\mathcal{NM}$ 和 vTPM $\mathcal{VM}$ 以及验证者 $\mathcal{C}$. 初始时,$\mathcal{CA}$ Tru $\mathcal{I}$, $\mathcal{C}$ Tru $\mathcal{CA}$.

首先,权威机构 $\mathcal{I}$ 对旧/新 pTPM $\mathcal{OM}/\mathcal{NM}$ 进行可信测评.标准为可信计算相关技术规范 Spec $\mathsf{Cri}_{\mathcal{I},\mathcal{OM}}=\{\mathsf{Spec}\}/\mathsf{Cri}_{\mathcal{I},\mathcal{NM}}=\{\mathsf{Spec}\}$, $\mathsf{Evi}_{\mathcal{OM}}/\mathsf{Evi}_{\mathcal{NM}}$ 则由 $\mathcal{I}$ 对 $\mathcal{OM}/\mathcal{NM}$ 分析和测试得到.若有 $\mathsf{Evi}_{\mathcal{OM}}$ Con $\{\mathsf{Spec}\}\Rightarrow\mathcal{I}$ Tru $\mathcal{OM}/\mathsf{Evi}_{\mathcal{NM}}$ Con $\{\mathsf{Spec}\}\Rightarrow\mathcal{I}$ Tru $\mathcal{NM}$, $\mathcal{I}$ 向 $\mathcal{OM}/\mathcal{NM}$ 颁发符合性证书 $\mathsf{Cert}_{\mathcal{I},\mathcal{OM}}$ / $\mathsf{Cert}_{\mathcal{I},\mathcal{NM}}$.权威机构 $\mathcal{I}$ 对 vTPM $\mathcal{VM}$ 进行可信测评,标准为技术规范 Spec $\mathsf{Cri}_{\mathcal{I},\mathcal{VM}}=\{\mathsf{Spec}\}$, $\mathsf{Evi}_{\mathcal{VM}}$ 则由 $\mathcal{I}$ 对 $\mathcal{VM}$ 分析和测试得到.若有 $\mathsf{Evi}_{\mathcal{VM}}$ Con $\{\mathsf{Spec}\}\Rightarrow\mathcal{I}$ Tru $\mathcal{VM}$,则 $\mathcal{I}$ 颁发 $\mathcal{VM}$ 参考值 $\mathsf{Ref}_{\mathcal{I},\mathcal{VM}}$.

旧/新 pTPM 在宿主机上开始运行后,向认证中心 $\mathcal{CA}$ 申请 IK 证书,$\mathcal{CA}$ 对旧/新 pTPM 进行可信测评.因为 $\mathcal{CA}$ Tru $\mathcal{I}$,根据旧/新 pTPM 所提供的符合性证书 $\mathsf{Cert}_{\mathcal{I},\mathcal{OM}}$ / $\mathsf{Cert}_{\mathcal{I},\mathcal{NM}}$,可得 $\mathcal{CA}$ Tru $\mathcal{OM}$ / $\mathcal{CA}$ Tru $\mathcal{NM}$,颁发 IK 证书.

vTPM $\mathcal{VM}$ 在旧 pTPM $\mathcal{OM}$ 宿主机上运行,$\mathcal{OM}$ 通过代理对 $\mathcal{VM}$ 进行度量,度量值为 $\mathsf{Mea}_{\mathcal{OM},\mathcal{VM}}$. $\mathcal{VM}$ 向 $\mathcal{CA}$ 申请 IK 证书,$\mathcal{CA}$ 对 $\mathcal{VM}$ 进行可信测评.因为有 $\mathcal{CA}$ Tru $\mathcal{I}$, $\mathcal{CA}$ 认可 $\mathsf{Ref}_{\mathcal{I},\mathcal{VM}}$ 作为 $\mathcal{VM}$ 的可信标准.因为有 $\mathcal{CA}$ Tru $\mathcal{OM}$, $\mathcal{CA}$ 认可 $\mathsf{Mea}_{\mathcal{OM},\mathcal{VM}}$ 作为 $\mathcal{VM}$ 的证据.故若有 $\{\mathsf{Mea}_{\mathcal{OM},\mathcal{VM}}\}$ Con $\{\mathsf{Ref}_{\mathcal{I},\mathcal{VM}}\}\Rightarrow\mathcal{CA}$ Tru $\mathcal{VM}$,则颁发 IK 证书.$\mathcal{VM}$ 进行远程证明时,使用 IK 签名,可得 $\mathcal{C}$ Tru $\mathcal{CA}$ Tru $\mathcal{VM}$,验证者 $\mathcal{C}$ 认可 $\mathcal{VM}$ 为可信报告根.vTPM $\mathcal{VM}$ 被迁移到新 pTPM $\mathcal{NM}$ 宿主机后,通知 $\mathcal{CA}$ 取消旧 IK 证书,$\mathcal{VM}$ 向 $\mathcal{CA}$ 新申请 IK 证书,同样可得 $\mathcal{C}$ Tru $\mathcal{CA}$ Tru $\mathcal{VM}$, $\mathcal{C}$ 认可迁移后的 $\mathcal{VM}$ 为可信报告根.

3.3 vTPM动态信任扩展模型

与基于证书链的机制相比,vTPM 动态信任扩展 DTE 增加了一个新实体——认证服务器 $\mathcal{AS}$. 初始时,$\mathcal{CA}$ Tru $\mathcal{I}$, $\mathcal{OM}$ Tru $\mathcal{I}$, $\mathcal{AS}$ Tru $\mathcal{I}$, $\mathcal{C}$ Tru $\mathcal{CA}$, $\mathcal{C}$ Tru $\mathcal{AS}$.

权威机构 $\mathcal{I}$ 对旧/新 pTPM $\mathcal{OM}$ / $\mathcal{NM}$ / $\mathcal{VM}$ 进行可信测评,与基于证书链的机制相同.$\mathcal{OM}$ / $\mathcal{NM}$ 向 $\mathcal{CA}$ 申请 IK 证书,$\mathcal{CA}$ 对旧/新 pTPM 进行可信测评,与基于证书链的机制相同.vTPM $\mathcal{VM}$ 第 1 次向 $\mathcal{CA}$ 申请 IK 证书,$\mathcal{CA}$ 对 $\mathcal{VM}$ 进行可信测评,与基于证书链的机制相同.

vTPM $\mathcal{VM}$ 在旧 pTPM $\mathcal{OM}$ 宿主机上运行,$\mathcal{OM}$ 通过代理对 $\mathcal{VM}$ 进行度量,度量值为 $\mathsf{Mea}_{\mathcal{OM},\mathcal{VM}}$. $\mathcal{OM}$ 对 $\mathcal{VM}$ 进行可信测评.因为有 $\mathcal{OM}$ Tru $\mathcal{I}$, $\mathcal{OM}$ 认可 $\mathsf{Ref}_{\mathcal{I},\mathcal{VM}}$ 作为 $\mathcal{VM}$ 的可信标准.故若有 $\{\mathsf{Mea}_{\mathcal{OM},\mathcal{VM}}\}$ Con $\{\mathsf{Ref}_{\mathcal{I},\mathcal{VM}}\}\Rightarrow\mathcal{OM}$ Tru $\mathcal{VM}$,则 $\mathcal{OM}$ 向 $\mathcal{VM}$ 颁发授权书 $w_{\mathcal{OM},\mathcal{VM}}$. $\mathcal{VM}$ 进行远程证明时,向 $\mathcal{AS}$ 申请时间令牌 $t_{\mathcal{AS},\mathcal{VM}}$, $\mathcal{AS}$ 对 $\mathcal{VM}$ 进行可信测评.因为 $\mathcal{AS}$ Tru $\mathcal{I}$, $\mathcal{I}$ Tru $\mathcal{OM}$,根据授权书 $w_{\mathcal{OM},\mathcal{VM}}$,可得 $\mathcal{AS}$ Tru $\mathcal{VM}$,颁发 $t_{\mathcal{AS},\mathcal{VM}}$.因为 $\mathcal{C}$ Tru $\mathcal{CA}$ Tru $\mathcal{OM}$, $\mathcal{C}$ Tru $\mathcal{AS}$ Tru $\mathcal{OM}$,验证者 $\mathcal{C}$ 根据正确的 $\mathcal{VM}$ IK 签名,$w_{\mathcal{OM},\mathcal{VM}}$ 和 $t_{\mathcal{AS},\mathcal{VM}}$,可得 $\mathcal{C}$ Tru $\mathcal{CA}$ Tru $\mathcal{OM}$ Tru $\mathcal{VM}$, $\mathcal{C}$ Tru $\mathcal{AS}$ Tru $\mathcal{OM}$ Tru $\mathcal{VM}$,只有两条路径同时存在,$\mathcal{C}$ 才认可 $\mathcal{VM}$ 为可信报告根.$\mathcal{VM}$ 被迁移时,$\mathcal{OM}$ 通知 $\mathcal{AS}$ 取消对 $\mathcal{VM}$ 的授权,$\mathcal{NM}$ 重新对 $\mathcal{VM}$ 授权,同样可得 $\mathcal{C}$ Tru $\mathcal{CA}$ Tru $\mathcal{NM}$ Tru $\mathcal{VM}$, $\mathcal{C}$ Tru $\mathcal{AS}$ Tru $\mathcal{NM}$ Tru $\mathcal{VM}$, $\mathcal{C}$ 认可 $\mathcal{VM}$ 为可信报告根.

4 vTPM 动态信任扩展 DTE

DTE 的基本思想如下:引入一个新实体——认证服务器 AS,在 DTE 的整体架构里包含 CA、AS、pTPM、vTPM 和远程验证者 Challenger 这 5 个实体.为了签署完整性报告,vTPM 须先从 AS 那里获取一个时间令牌,该令牌与 Challenger 发送的随机 nonce 相关.只有在 pTPM 签署授权给 vTPM 后,vTPM 才能获得有效的令牌.当 vTPM 发生迁移时,旧宿主机上的 pTPM 就通知 AS 停止颁发令牌给迁走的 vTPM.AS 维护着一个 pTPM 给 vTPM 签署的授权书列表.

为便于系统可扩展,一个 CA 可对应多个 AS,每个 AS 又为多个 pTPM 和 vTPM 提供服务.每个 vTPM 实例

所在虚拟机上还可能有多个用户,针对不同的应用场景,使用 vTPM 来进行远程证明.时间令牌与每一个远程证明相关,将时间令牌的发放设定为一个原子操作,确保其不被干扰.在 vTPM 频繁迁移时,DTE 里会出现大量的短命 pTPM 授权书,但授权书并不是 CA 证书,成本较低.AS 采用授权书白名单机制,大量撤销的短命授权书不会增加 AS 的负担,且能保证实时撤销和前向安全特性.

DTE 包含如下 5 个过程:(1) 参数设置;(2) 信任授权;(3) 令牌获取;(4) 远程证明;(5) 撤销授权.DTE 的总体过程及消息序列如图 1 所示,将在后续小节中描述其更多细节.

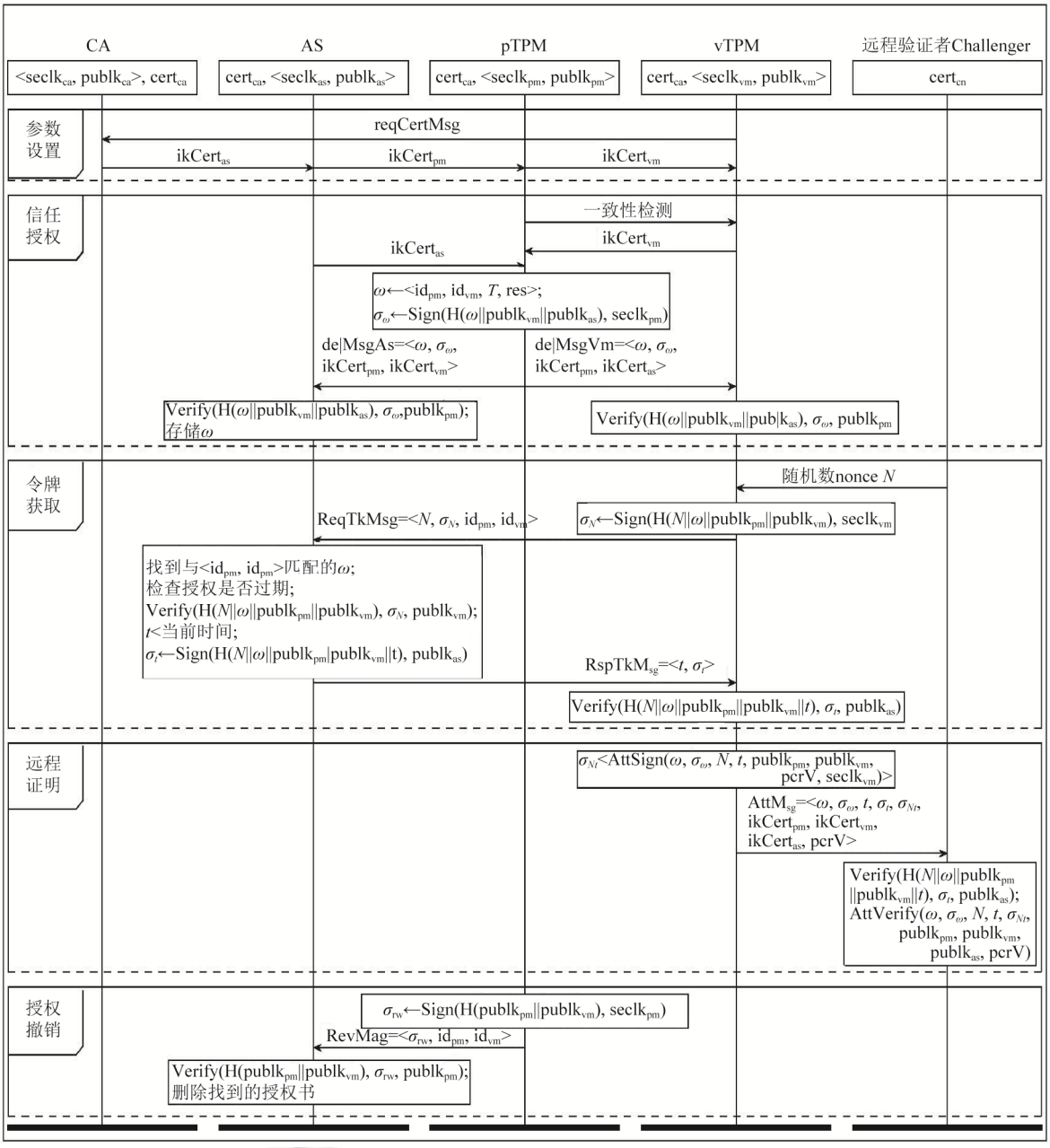

Fig.1 The procedures and message sequences of vTPM dynamic trust extension

图 1 DTE 的总体过程及消息序列

4.1 参数设置

系统参数主要包括 pTPM、vTPM 以及 AS 的参数.就非对称密码算法而言,TPM 1.2 支持 RSA,中国的可信密码模块(trusted cryptography module,简称 TCM)支持 SM2(一种椭圆曲线密码算法 ECC,elliptic curve cryptography),TPM 2.0 可灵活支持不同的算法.考虑到 RSA 和 ECC 的广泛应用,DTE 主要支持这两种算法.在模块同时支持 RSA 和 ECC 的情况下,具体使用哪种算法,视具体的部署环境而定.为了描述方便,除可能会导致混淆的特殊情况外,后文将 TPM 1.2、TPM 2.0 和 TCM 统称为 TPM.

1) pTPM 生成 IK, $\langle \mathsf{seclk}_{pm}, \mathsf{publk}_{pm} \rangle$, 从 CA 处获取对应的证书 ikCert_{pm}.

2) vTPM 生成 IK, $\langle \mathsf{seclk}_{vm}, \mathsf{publk}_{vm} \rangle$, 向 CA 申请对应的证书 ikCert_{vm}. CA 在颁发证书之前,首先验证其底层 pTPM 所签署的远程证明报告,判断 vTPM 实例与标准的一致性.

3) AS 生成 IK, $\langle \mathsf{seclk}_{as}, \mathsf{publk}_{as} \rangle$, 从 CA 获取对应的证书 ikCert_{as}. 我们假定 AS 也是一个可信计算平台(trusted computing platform,简称 TCP),AS 生成 IK 及获取证书的操作都和 pTPM 相同.

IK 的格式见公式(1).为了保护隐私,pTPM 和 vTPM 可能有多个 IK 及与之对应的证书.每个证书含有一个与之绑定的身份 id.

$$\begin{matrix}\langle \mathsf{seclk}_x, \mathsf{publk}_x \rangle \\ (x=\mathrm{pm},\mathrm{vm},\mathrm{as})\end{matrix} = \begin{cases}\langle\langle e_x, n_x\rangle, \langle d_x, n_x\rangle\rangle & \text{RSA} \\ \langle d_x, Q_x = d_x G\rangle & \text{ECC}\end{cases} \tag{1}$$

4.2 信任授权

为了将信任扩展到 vTPM 上,

1) pTPM 对 vTPM 实例进行一致性检测,获取 vTPM 的 IK 证书 ikCert_{vm}.

2) pTPM 创建授权书 w,包含 vTPM 的身份信息 id_{vm}、pTPM 的身份信息 id_{pm}、授权的有效期 T 以及其他限制性条件 res, 表示为 $w = \langle \mathsf{id}_{pm}, \mathsf{id}_{vm}, T, \mathsf{res} \rangle$, 其中, id_{pm} 来自于 ikCert_{pm}, id_{vm} 来自于 ikCert_{vm}.

3) pTPM 获取 AS 的 IK 证书 ikCert_{as}.

4) pTPM 用其 IK 私钥对授权书 w 签名, $\sigma_w = \mathtt{Sign}(\mathtt{H}(w \| \mathsf{publk}_{vm} \| \mathsf{publk}_{as}), \mathsf{seclk}_{pm})$, 函数 Sign 的定义见公式(2). publk_{vm} 来自于 ikCert_{vm}, publk_{as} 来自于 ikCert_{as}. 若 seclk_{pm} 是 ECC 私钥,则 σ_w 是 ECC Schnorr 签名,表示为 $\sigma_w = \langle r_w, s_w \rangle$.

5) pTPM 发送消息 delMsgVm 给 vTPM, $\mathsf{delMsgVm} = \langle w, \sigma_w, \mathsf{ikCert}_{pm}, \mathsf{ikCert}_{as} \rangle$;pTPM 发送消息 delMsgAs 给 AS, $\mathsf{delMsgAs} = \langle w, \sigma_w, \mathsf{ikCert}_{pm}, \mathsf{ikCert}_{vm} \rangle$.

$$\mathtt{Sign} = \begin{cases}\mathtt{RSASign} & \mathsf{seclk}_x \text{是RSA私钥} \\ \mathtt{ECCSchSign} & \mathsf{seclk}_x \text{是ECC私钥}\end{cases} \tag{2}$$

接收到消息 delMsgVm 后,

1) vTPM 检查 ikCert_{pm} 和 ikCert_{as}, 从 ikCert_{pm} 中提取 publk_{pm}, 从 ikCert_{as} 中提取 publk_{as}.

2) vTPM 检查授权书签名, $\mathsf{Y/N} = \mathtt{Verify}(\mathtt{H}(w \| \mathsf{publk}_{vm} \| \mathsf{publk}_{as}), \sigma_w, \mathsf{publk}_{pm})$, Verify 的定义见公式(3).

$$\mathtt{Verify} = \begin{cases}\mathtt{RSAVerify} & \mathsf{publk}_x \text{是RSA公钥} \\ \mathtt{ECCSchVerify} & \mathsf{publk}_x \text{是ECC公钥}\end{cases} \tag{3}$$

接收到消息 delMsgAs 后,

1) AS 检查 ikCert_{pm} 和 ikCert_{vm}, 从 ikCert_{pm} 中提取 publk_{pm}, 从 ikCert_{vm} 中提取 publk_{vm}.

2) vTPM 检查授权书签名, $\mathsf{Y/N} = \mathtt{Verify}(\mathtt{H}(w \| \mathsf{publk}_{vm} \| \mathsf{publk}_{as}), \sigma_w, \mathsf{publk}_{pm})$; 若 σ_w 是正确的,则 AS 就存储授权书 w.一旦授权到期,AS 可删除其列表中的授权书,故 DTE 中 AS 授权书列表不会无限增加,这有别于其他普通的白名单或者黑名单方法.

4.3 令牌获取

如果远程验证者无法确定 vTPM 签名的准确时间,那么授权书的有效期也没有用处.DTE 用时间令牌来解

决这个问题，并且使 pTPM 能够及时撤销其对 vTPM 的信任扩展.若 vTPM 收到一个来自于远程验证者带随机数 nonce N 的证明请求,在其对报告数据进行签名之前,须先从 AS 处获取一个针对该 nonce 的特定时间令牌.

1) vTPM 生成一个与随机数 nonce 相关的签名, $\sigma_N=\mathtt{Sign}(\mathtt{H}(N\|w\|\mathsf{publk}_{\mathrm{pm}}\|\mathsf{publk}_{\mathrm{vm}}),\mathsf{seclk}_{\mathrm{vm}})$.

2) vTPM 向 AS 发送令牌请求消息, $\mathsf{ReqTkMsg}=\langle N,\sigma_N,\mathsf{id}_{\mathrm{pm}},\mathsf{id}_{\mathrm{vm}}\rangle$, 其中, $\mathsf{id}_{\mathrm{pm}}$ 从 $\mathsf{ikCert}_{\mathrm{pm}}$ 中提取.

接收到消息 ReqTkMsg 后,

1) AS 查询是否存在与 $\langle\mathsf{id}_{\mathrm{pm}},\mathsf{id}_{\mathrm{vm}}\rangle$ 相匹配的授权书,如果找到授权书 w,则 AS 确定它是否在有效期内.

2) AS 找到与 $\mathsf{id}_{\mathrm{pm}}$ 对应的 $\mathsf{ikCert}_{\mathrm{pm}}$, 从中提取出 $\mathsf{publk}_{\mathrm{pm}}$; 找到与 $\mathsf{id}_{\mathrm{vm}}$ 对应的 $\mathsf{ikCert}_{\mathrm{vm}}$, 从中提取出 $\mathsf{publk}_{\mathrm{vm}}$.

3) AS 验证令牌请求消息的签名 σ_N, $\mathsf{Y/N}=\mathtt{Verify}(\mathtt{H}(N\|w\|\mathsf{publk}_{\mathrm{pm}}\|\mathsf{publk}_{\mathrm{vm}}),\sigma_N,\mathsf{publk}_{\mathrm{vm}})$.

4) AS 计算 $\sigma_t=\mathtt{Sign}(\mathtt{H}(N\|w\|\mathsf{publk}_{\mathrm{pm}}\|\mathsf{publk}_{\mathrm{vm}}\|t),\mathsf{publk}_{\mathrm{as}})$, 其中, t 是当前时间.

5) AS 发送令牌响应消息给 vTPM, $\mathsf{RspTkMsg}=\langle t,\sigma_t\rangle$.

接收到消息 RspTkMsg 后,

vTPM 验证 σ_t, $\mathsf{Y/N}=\mathtt{Verify}(\mathtt{H}(N\|w\|\mathsf{publk}_{\mathrm{pm}}\|\mathsf{publk}_{\mathrm{vm}}\|t),\sigma_t,\mathsf{publk}_{\mathrm{as}})$; 若 σ_t 是正确的,vTPM 就可代表 pTPM 对虚拟机报告数据签名,进行远程证明.

4.4 远程证明

接收到正确的时间令牌后,

1) vTPM 计算 $\sigma_{Nt}=\mathtt{AttSign}(w,\sigma_w,N,t,\mathsf{publk}_{\mathrm{pm}},\mathsf{publk}_{\mathrm{vm}},\mathsf{pcrV},\mathsf{seclk}_{\mathrm{vm}})$, 函数 AttSign 的定义见算法 1,其中, pcrV 为所选定的 vTPM 中平台配置寄存器(platform configuraion register,简称 PCR)的当前值.

2) vTPM 将证明消息发送给远程验证者, $\mathsf{AttMsg}=\langle w,\sigma_w,t,\sigma_t,\sigma_{Nt},\mathsf{ikCert}_{\mathrm{pm}},\mathsf{ikCert}_{\mathrm{vm}},\mathsf{ikCert}_{\mathrm{as}},\mathsf{pcrV}\rangle$.

接收到证明消息 AttMsg 后,

1) 远程验证者首先检查 $\mathsf{ikCert}_{\mathrm{pm}}$、$\mathsf{ikCert}_{\mathrm{vm}}$ 和 $\mathsf{ikCert}_{\mathrm{as}}$, 从 $\mathsf{ikCert}_{\mathrm{pm}}$ 中提取 $\mathsf{publk}_{\mathrm{pm}}$, 从 $\mathsf{ikCert}_{\mathrm{vm}}$ 中提取 $\mathsf{publk}_{\mathrm{vm}}$, 从 $\mathsf{ikCert}_{\mathrm{as}}$ 中提取 $\mathsf{publk}_{\mathrm{as}}$.

2) 验证者检查 σ_t 的正确性, $\mathsf{Y/N}=\mathtt{Verify}(\mathtt{H}(N\|w\|\mathsf{publk}_{\mathrm{pm}}\|\mathsf{publk}_{\mathrm{vm}}\|t),\sigma_t,\mathsf{publk}_{\mathrm{as}})$.

3) 验证者检查 σ_{Nt} 的正确性, $\mathsf{Y/N}=\mathtt{AttVerify}(w,\sigma_w,N,t,\sigma_{Nt},\mathsf{publk}_{\mathrm{pm}},\mathsf{publk}_{\mathrm{vm}},\mathsf{publk}_{\mathrm{as}},\mathsf{pcrV})$, 函数 AttVerify 的定义见算法 2.

算法 1. $\sigma_{Nt}=\mathtt{AttSign}(w,\sigma_w,N,t,\mathsf{publk}_{\mathrm{pm}},\mathsf{publk}_{\mathrm{vm}},\mathsf{pcrV},\mathsf{seclk}_{\mathrm{vm}})$.

1: **if** $\mathsf{seclk}_{\mathrm{vm}}$ 是一个 RSA 私钥
  & σ_w 是一个 RSA 签名 **then**
2: $\sigma_{Nt}\leftarrow\mathtt{RSAPrivateEnc}(\sigma_w\oplus\mathtt{Encode}(\mathtt{H}(N\|w\|\mathsf{publk}_{\mathrm{pm}}\|\mathsf{publk}_{\mathrm{vm}}\|t\|\mathsf{pcrV})),\mathsf{seclk}_{\mathrm{vm}})$
3: **else**
4: **if** $\mathsf{seclk}_{\mathrm{vm}}$ 是一个 ECC 私钥
  & σ_w 是一个 ECC Schnorr 签名 **then**
5: $\mathsf{seclk}_{\mathrm{vm}}\leftarrow s_w+r_w\mathsf{seclk}_{\mathrm{vm}}\bmod n$
6: $\mathsf{publk}_{\mathrm{vm}'}\leftarrow\mathsf{seclk}_{\mathrm{vm}'}G$
  G 是所选定椭圆曲线上的一个加法循环子群的生成元,n 是该子群的阶
7: $\langle r_{Nt},s_{Nt}\rangle\leftarrow\mathtt{ECCSchSign}(\mathtt{H}(N\|w\|\mathsf{publk}_{\mathrm{pm}}\|\mathsf{publk}_{\mathrm{vm}}\|t\|\mathsf{pcrV}\|\mathsf{publk}_{\mathrm{vm}}),\mathsf{seclk}_{\mathrm{vm}})$
8: $\sigma_{Nt}=\langle r_{Nt},s_{Nt},\mathsf{publk}_{\mathrm{vm}}\rangle$
9: **end if**
10: **end if**

算法 2. $\mathsf{Y/N}=\mathtt{AttVerify}(w,\sigma_w,N,t,\sigma_{Nt},\mathsf{publk}_{\mathrm{pm}},\mathsf{publk}_{\mathrm{vm}},\mathsf{publk}_{\mathrm{as}},\mathsf{pcrV})$.

1: **if** $\mathsf{publk}_{\mathrm{vm}}$ 是一个 RSA 公钥
  & $\mathsf{publk}_{\mathrm{pm}}$ 是一个 RSA 公钥 **then**

2: $\sigma_{w'} \leftarrow$ RSAPublicDec(σ_{Nt},publk$_{vm}$)$\oplus$Encode(H(N‖ w‖ publk$_{pm}$‖ publk$_{vm}$‖ t‖ pcrV)

3: R$\leftarrow$RSAVerify(H(w‖ publk$_{vm}$‖ publk$_{as}$),$\sigma_{w'}$,publk$_{pm}$)

4: **return** R

5: **else**

6: **if** publk$_{vm}$ 是一个 ECC 公钥 & publk$_{pm}$ 是一个 ECC 公钥 **then**

7: $(x_r, y_r) \leftarrow$ publk$_{vm} - r_w$(publk$_{pm}$ + publk$_{vm}$)

8: **if** H(H(w‖ publk$_{vm}$‖ publk$_{as}$)‖ x_r)!= r_w

9: **return** R

10: **end if**

11: R$\leftarrow$ECCSchVerify(H(N‖ w‖ publk$_{pm}$‖ publk$_{vm}$‖ t‖ pcrV‖ publk$_{vm}$),$\langle r_{Nt}, s_{Nt}\rangle$,publk$_{vm}$)

12: **return** R

13: **end if**

14: **end if**

4.5 撤销授权

当 vTPM 及其对应的虚拟机被迁移到新的宿主机之后,旧主机上的 pTPM 即可撤销对 vTPM 的信任授权.

1) pTPM 计算 σ_{rw} = Sign(H(publk$_{pm}$‖ publk$_{vm}$),seclk$_{pm}$).

2) pTPM 发送消息 RevMsg 给 AS,请求删除相应的授权书, RevMsg = $\langle\sigma_{rw}$,id$_{pm}$,id$_{vm}\rangle$.

接收到消息 RevMsg 后,

1) AS 查询其列表中是否存在与 $\langle$id$_{pm}$,id$_{vm}\rangle$ 相匹配的授权书,若存在,则有

2) AS 找到与 id$_{pm}$ 对应的 ikCert$_{pm}$, 从中提取出 publk$_{pm}$; 找到与 id$_{vm}$ 对应的 ikCert$_{vm}$, 从中提取出 publk$_{vm}$.

3) AS 验证 σ_{rw} 的正确性, Y/N = Verify(H(publk$_{pm}$‖ publk$_{vm}$),σ_{rw},publk$_{pm}$).

4) AS 删除找到的授权书.

当 AS 收到一个 vTPM 的时间令牌请求时,若没有可匹配的授权书,则该 vTPM 没有被授权、授权已过期或已经被撤销,AS 会拒绝该请求.没有时间令牌,vTPM 就不能对报告数据进行签名,故 pTPM 可及时撤销对 vTPM 的信任扩展.

5 安全性分析

本节我们将分析 DTE 可以确保正确性和满足如下安全特征:秘密依赖、不可伪造、可验证、可识别、不可否认、可撤销、严格区分 vTPM 和 pTPM 这两种不同安全强度的 TPM 和前向安全.

5.1 正确性

DTE 的正确性定义:若 CA、AS、pTPM、vTPM 和远程验证者 Challenger 都是诚实的,那么,对于 vTPM 签名的完整性报告,Challenger 会以百分百的概率通过验证.

DTE 除使用了 AttSign、AttVerify 两个特殊函数外,其余都是普通的 RSA 和 ECC 签名、验证算法.普通 RSA、ECC 算法的正确性可以得到保证,我们着重分析 AttSign、AttVerify 的正确性.

若选用 RSA 算法,验证者在执行 AttVerify 函数时,首先会计算 RSAPublicDec(σ_{nt},publk$_{vm}$). 其中, σ_{nt} 是由 vTPM 在执行 AttSign 时所计算出来的, σ_{nt} = RSAPrivateEnc($\sigma_w \oplus$ Encode(H(n‖ w‖ publk$_{pm}$‖ publk$_{vm}$‖ t‖ pcrV)),seclk$_{vm}$), 故有

$$\text{RSAPublicDec}(\sigma_{nt},\text{publk}_{vm}) = \sigma_w \oplus \text{Encode}(H(n\| w\| \text{publk}_{pm}\| \text{publk}_{vm}\| t\| \text{pcrV})),$$

$$\text{RSAPublicDec}(\sigma_{nt},\text{publk}_{vm}) \oplus \text{Encode}(H(n\| w\| \text{publk}_{pm}\| \text{publk}_{vm}\| t\| \text{pcrV}) = \sigma_w.$$

因而等式 $\sigma_{w'} = \sigma_w$ 成立, RSAVerify(H(w‖ publk$_{vm}$‖ publk$_{as}$),$\sigma_{w'}$,publk$_{pm}$) 的返回值应为 Y.

若选用 ECC 算法,验证者在执行 AttVerify 函数时,首先会计算 $\mathsf{publk}_{\mathrm{vm}}-r_w(\mathsf{publk}_{\mathrm{pm}}+\mathsf{publk}_{\mathrm{vm}})$, $\mathsf{publk}_{\mathrm{vm}}$ 是由 vTPM 在执行 AttSign 时所计算出来的,$\mathsf{publk}_{\mathrm{vm}}=\mathsf{seclk}_{\mathrm{vm}}G=s_wG+r_w\mathsf{seclk}_{\mathrm{vm}}G=s_wG+r_w\mathsf{publk}_{\mathrm{vm}}$, 故有

$$\mathsf{publk}_{\mathrm{vm}}-r_w(\mathsf{publk}_{\mathrm{pm}}+\mathsf{publk}_{\mathrm{vm}})=s_wG+r_w\mathsf{publk}_{\mathrm{vm}}-r_w(\mathsf{publk}_{\mathrm{pm}}+\mathsf{publk}_{\mathrm{vm}})=s_wG-r_w\mathsf{publk}_{\mathrm{pm}}.$$

从 ECC Schnorr 签名算法可以得知,授权书签名 $\sigma_w=\langle r_w,s_w\rangle$, $r_w=\mathrm{H}(\mathrm{H}(w\|\ \mathsf{publk}_{\mathrm{vm}}\|\ \mathsf{publk}_{\mathrm{as}})\|\ x_w)$, $s_w=k_w+r_w\mathsf{seclk}_{\mathrm{pm}} \bmod n$. 故有

$$\mathsf{publk}_{\mathrm{vm}}-r_w(\mathsf{publk}_{\mathrm{pm}}+\mathsf{publk}_{\mathrm{vm}})=k_wG+r_w\mathsf{seclk}_{\mathrm{pm}}G-r_w\mathsf{publk}_{\mathrm{pm}}=k_wG.$$

因而 $(x_r,y_r)=(x_w,y_w)$, 等式 $\mathrm{H}(\mathrm{H}(w\|\ \mathsf{publk}_{\mathrm{vm}}\|\ \mathsf{publk}_{\mathrm{as}})\|\ x_r)=r_w$ 成立, AttVerify 的返回值应为 Y.

5.2 安全性

秘密依赖.基于 pTPM 的 IK 私钥,生成一个新秘密,再生成远程证明签名 σ_{Nt}. 从计算远程证明签名 σ_{Nt} 的算法 1 AttSign 中可见,若 DTE 采用 RSA,在计算 RsaPrivateEnc 时,会用到授权书签名 σ_w; 若 DTE 采用 ECC,在计算临时私钥 $\mathsf{seclk}'_{\mathrm{vm}}$ 时,会用到授权书签名 $\sigma_w=\langle r_w,s_w\rangle$. σ_w 的生成都需要用到 pTPM 的 IK 私钥 $\mathsf{seclk}_{\mathrm{pm}}$.

不可伪造.一个 vTPM 只有在获得一个 pTPM 授权经过信任扩展之后,才可生成一个合法的远程证明签名 σ_{Nt}. 从计算远程证明签名 σ_{Nt} 的算法 1 AttSign 中可见,若 DTE 采用 RSA,在计算 RsaPrivateEnc 时,会用到授权书签名 σ_w 和 vTPM IK 私钥 $\mathsf{seclk}_{\mathrm{vm}}$; 若 DTE 采用 ECC,在计算临时私钥 $\mathsf{seclk}'_{\mathrm{vm}}$ 时,会用到授权书签名 $\sigma_w=\langle r_w,s_w\rangle$ 和 vTPM IK 私钥 $\mathsf{seclk}_{\mathrm{vm}}$. σ_w 的生成都需要用到 pTPM 的 IK 私钥 $\mathsf{seclk}_{\mathrm{pm}}$. 任何实体,包括 pTPM,在不知道 $\mathsf{seclk}_{\mathrm{vm}}$ 和 $\mathsf{seclk}_{\mathrm{pm}}$ 的情况下,不能伪造一个有效的认证签名 σ_{Nt}. 无论 RSA 还是 ECC,从公钥计算出私钥,都是困难问题.

可验证.验证者收到一个 vTPM 发送的远程证明消息 AttMsg, AttMsg 中含有签名 σ_{Nt}, 若验证其合法,可确信远程证明经过了一个 pTPM 的同意.DTE 远程证明消息 AttMsg 含有授权书 w 和 pTPM 对其的签名 σ_w, w 中有 pTPM 的身份标识 $\mathsf{id}_{\mathrm{pm}}$, σ_w 的生成都需要用到 pTPM 的 IK 私钥 $\mathsf{seclk}_{\mathrm{pm}}$. 验证者在执行 AttVerify 时,会用到 pTPM 的 IK 公钥 $\mathsf{publk}_{\mathrm{pm}}$. 这些都可让验证者确信 vTPM 所发起的远程证明经过了 pTPM 的授权同意.

可识别.验证者依据一个合法的 DTE 远程证明消息,可确定一个 vTPM 身份.DTE 远程证明消息 AttMsg 含有授权书 w 和 vTPM IK 公钥证书 $\mathsf{ikCert}_{\mathrm{vm}}$, w 和 $\mathsf{ikCert}_{\mathrm{vm}}$ 中都含有 vTPM 的身份标识 $\mathsf{id}_{\mathrm{vm}}$. 在算法 1 AttSign 中计算远程证明签名 σ_{Nt} 时,会用到 vTPM IK 私钥 $\mathsf{seclk}_{\mathrm{vm}}$. 验证者在执行 AttVerify 时,需使用 $\mathsf{ikCert}_{\mathrm{vm}}$ 中的 vTPM IK 公钥 $\mathsf{publk}_{\mathrm{vm}}$. 这些都可让验证者确定 vTPM 的身份.

不可否认.在 pTPM 授权后,一旦 vTPM 创建了一个合法的远程证明签名 σ_{Nt}, pTPM 和 vTPM 都不可否认. 在算法 1 AttSign 中计算远程证明签名 σ_{Nt} 时,会用到授权书签名 σ_w 和 vTPM IK 私钥 $\mathsf{seclk}_{\mathrm{vm}}$, σ_w 的生成需要用到 pTPM IK 私钥 $\mathsf{seclk}_{\mathrm{pm}}$. 验证者在执行 AttVerify 时,需使用 $\mathsf{ikCert}_{\mathrm{pm}}$ 中的 vTPM IK 公钥 $\mathsf{publk}_{\mathrm{pm}}$ 和 $\mathsf{ikCert}_{\mathrm{vm}}$ 中的 vTPM IK 公钥 $\mathsf{publk}_{\mathrm{vm}}$. pTPM 和 vTPM 皆无法否认一个合法的远程证明签名 σ_{Nt}.

可撤销.pTPM 可实时撤销对 vTPM 的信任扩展,在撤销之后,vTPM 不能再进行合法的远程证明.DTE 远程证明消息 AttMsg 含有时间令牌 t 和 AS 对其的签名 σ_t, 在算法 1 AttSign 中计算远程证明签名 σ_{Nt} 时,会用到时间令牌 t.只有在 pTPM 对 vTPM 的授权书 w 未过期、未被撤销的情况下,AS 才会向 vTPM 发送时间令牌,且 σ_t 与远程证明请求 nonce N 相关.pTPM 可通知 AS 撤销对 vTPM 的授权,即 pTPM 可实时撤销对 vTPM 的信任扩展.

区分.在 DTE 中,vTPM 所提交的远程证明信息包括授权书及其签名、时间令牌及其签名、平台配置完整性值、DTE 认证签名以及 pTPM、vTPM 和 AS 的 IK 证书.TCG pTPM 所提交的远程证明信息包括平台配置完整性值、引用签名以及 pTPM IK 证书.验证这两种不同类型的报告,需要使用不同的方法.验证 vTPM 的报告,需要 vTPM 的 IK 公钥 $\mathsf{publk}_{\mathrm{vm}}$、pTPM 的 IK 公钥 $\mathsf{publk}_{\mathrm{pm}}$ 和授权书 w;而验证 pTPM 的报告,只需要它的 IK 公钥 $\mathsf{publk}_{\mathrm{pm}}$. 根据报告的不同格式和验证的不同程序,远程验证者就可区分出 vTPM 和 pTPM.CA 在给 vTPM 颁发 IK 证书时,可加注其是 vTPM 的说明.

前向安全.假设 vTPM 在迁移之前,已生成有一个远程证明签名 $O\sigma_{Nt}$, 经过了旧宿主机 pTPM 的授权.出于某

种原因,远程验证者在 vTPM 迁移完成之后,才收到了其提交的证明信息, $O\mathrm{AttMsg}=\langle Ow,O\sigma_w,Ot,O\sigma_t,O\sigma_{Nt},\mathrm{ikCert}_{Opm},\mathrm{ikCert}_{vm},\mathrm{ikCert}_{as}\rangle$. 由于在 DTE 中,无需旧的 pTPM 通知 CA 将 ikCert_{vm} 加入到 CRL 中,以撤销信任扩展,故 ikCert_{vm} 不在 CA 的 CRL 中,验证者仍然可以从 ikCert_{vm} 中提取出 publk_{vm}. 验证者也可以获取 publk_{pm} 和 publk_{as}, 可以正常完成对 $O\sigma_t$ 和 $O\sigma_{Nt}$ 的验证.因此,DTE 可提供前向安全保证.

6 原型系统与性能测试

为了测试 DTE 的性能,我们在 OpenStack 基础上建立了一个 DTE 原型系统,包含一个控制节点和两个计算节点.控制节点和计算节点的系统都是 Ubuntu 14.04.2 LTS Server (64 位).在控制节点上安装了网络时间协议(network time protocol,简称 NTP)服务、MySQL 数据库、RabbitMQ 消息代理服务、身份服务、镜像服务、计算服务和传统 nova 网络模块.在计算节点上安装了 NTP 服务、计算服务、传统 nova 网络模块、QEMU-KVM 虚拟机管理程序(支持 KVM 全虚拟化的 QEMU).NTP 服务用来在 AS、pTPM、vTPM 和验证者之间同步时间.

6.1 原型系统

计算节点被配置为一个可信虚拟平台(trusted virtualized platform,简称 TVP),包含一个 pTPM 代理和 vTPM 代理.AS 被部署在控制节点上,为了方便使用 TPM 的签名和验签功能,控制节点被配置为一个可信计算平台 TCP.为了简单起见,计算节点的物理平台充当验证者的角色.另外,在原型系统中没有 CA,不存在证书操作,我们假设公钥不会被篡改.

为了测试对不同非对称密码算法的兼容性,我们分别建立了针对 TPM 1.2 和 TPM 2.0 两个不同版本的原型系统,其中,基于 TPM 1.2 的原型系统如图 2 所示.

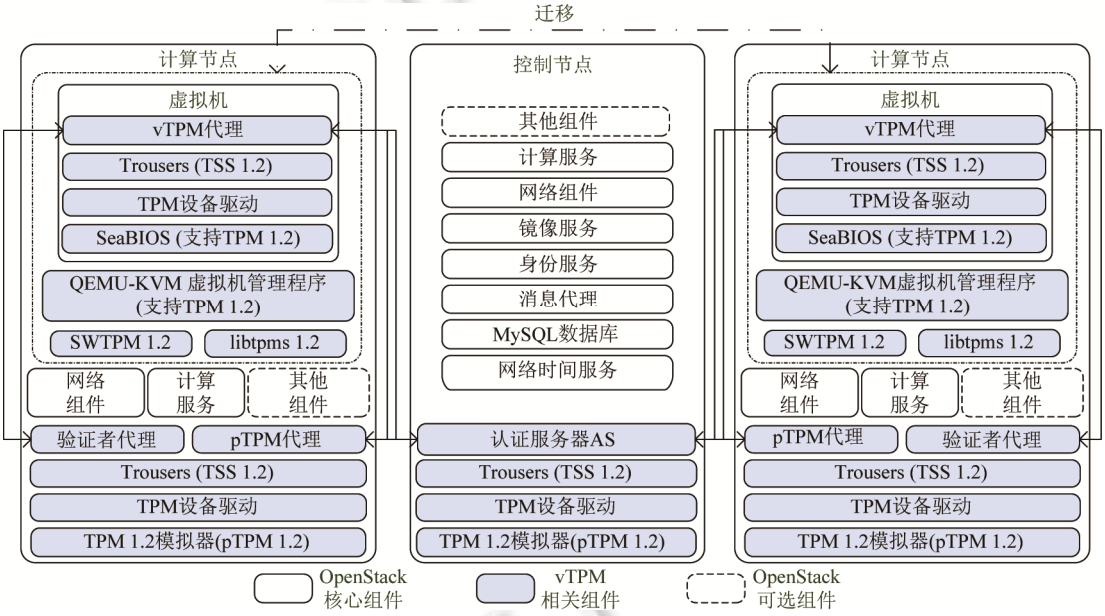

Fig.2 The architecture of the DTE prototype for TPM 1.2

图 2 针对 TPM 1.2 的 DTE 原型系统

在控制节点和计算节点上,都配置有 TPM 1.2 模拟器[5]、设备驱动和 TSS 1.2 软件栈 TrouSerS[6].在计算节点上还配置有 libtpms[7]、基于软件的 TPM 模拟器(software-based TPM emulator,简称 SWTPM)[8]和 SeaBIOS,设定 QEMU-KVM,以支持 vTPM 1.2.我们创建了一个集成有 TPM 1.2 设备驱动和 TSS 1.2 软件栈 TrouSerS 的虚拟机镜像,系统为 Ubuntu 12.04.5 LTS Server(64 位).在构建针对 TPM 2.0 的原型系统时,我们遇到了一些困难.

(1) TSS 2.0[9]的实现尚不完整,只包含资源管理器层(resource manager,简称 RM)、TPM 访问代理层(TPM

access broker,简称 TAB)、TPM 命令传输接口层(TPM command transmission interface,简称 TCTI)和系统 API(system application program interface,简称 SysAPI),缺少特征 API(feature API).

(2) 没有用以创建 QEMU TPM 实例的 TPM 2.0 软件实现.

(3) QEMU-KVM 不支持 TPM 2.0 设备.

(4) 虚拟机中的 SeaBIOS 不支持 TPM 2.0 扩展.

考虑到这些困难,我们直接在控制节点、计算节点物理平台和虚拟机上运行 Linux 版的 TPM 2.0 模拟器[10].AS 程序、pTPM 代理和 vTPM 代理都使用 TSS 2.0 SysAPI 接口,通过套接字驱动访问各自对应的 TPM 2.0 模拟器.针对 TPM 2.0 的原型系统如图 3 所示.

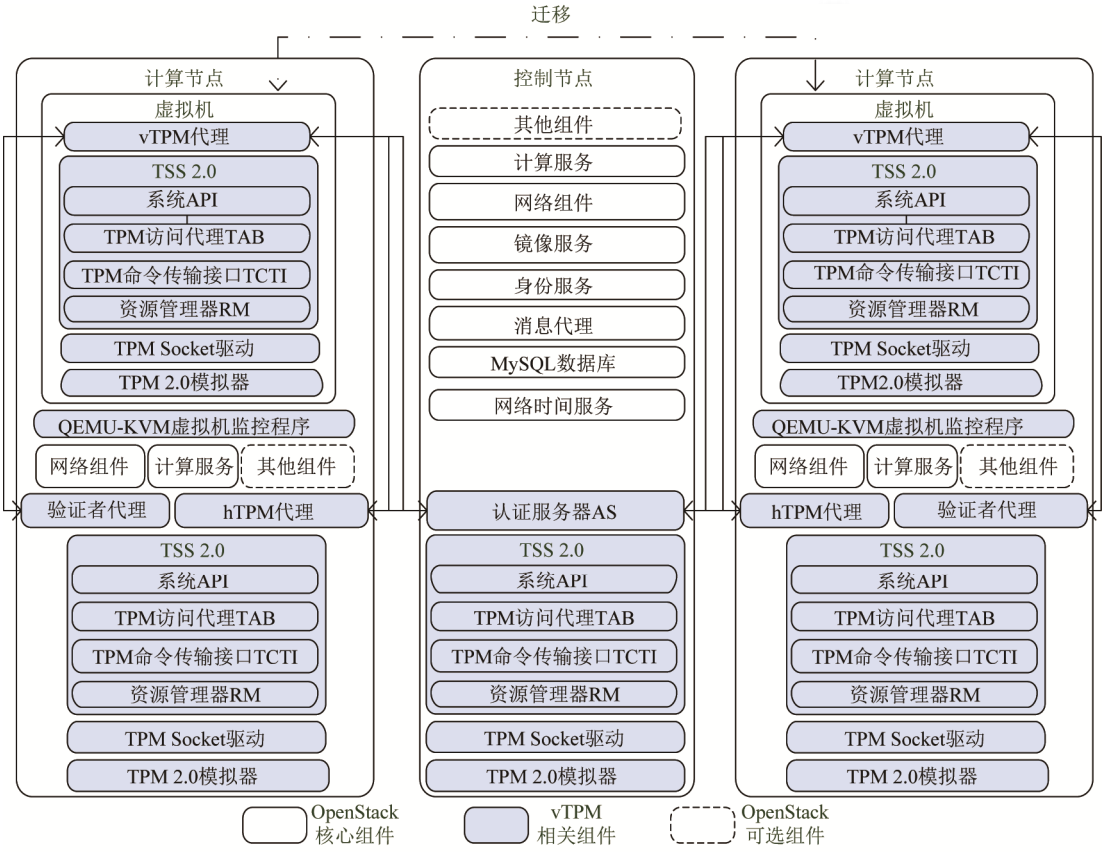

Fig.3 The architecture of the DTE prototype for TPM 2.0

图 3 针对 TPM 2.0 的 DTE 原型系统

6.2 功能接口调用性能测试

AS、pTPM 代理、vTPM 代理和验证者的大多数功能,都可通过调用 TSS1.2 TSPI 或 TSS2.0 SysAPI 来实现,但有些功能没有现成的 TSS API 可用,我们转而调用 OpenSSL 接口,例如 `RSA_Verify`,或者直接在代理程序中用代码实现,例如 `SchnorrEcc`. DTE 的性能测试,可以通过这些调用接口或函数所花费的时间来衡量,测试结果见表 2.表中的时间是 10 次测试结果的平均值.在我们的实验中,控制节点和计算节点都是运行在 VMWare 上的虚拟机,每个虚拟机都被分配一个处理器和 1GB(giga bytes)内存空间.运行 VMWar 的计算机,处理器是 Intel Core i3-4020Y@1.50GHZ,内存是 4GB,系统是 Windows 8.1.计算节点上的虚拟机所分配的内存大小为 768MB(mega bytes).

Table 2 The performance of AS, pTPM, vTPM and Challenger

表 2 AS、pTPM、vTPM 和验证者 Challenger 的性能测试结果

实体	过程	TSS 1.2 TSPI	时间(μs) (RSA)	TSS 2.0 SysAPI	时间(μs) (RSA)	时间(μs) (ECC)
AS	创建 IK	Tspi_TPM_Collate IdentityRequest	199 832	Tss2_Sys_Create	810 336	120 692
pTPM			196 143		997 353	125 436
vTPM			459 412		1 030 989	130 336
pTPM	授权书签名	Tspi_Context_LoadKeyByUUID	32 998	Tss2_Sys_Load	278 956	289 147
		Tspi_TPM_Quote§	13 722			
		Tspi_Context_LoadKeyByUUID	38 579	Tss2_Sys_Quote§	430 269	425 078
		Tspi_TPM_Quote2§	11 334			
AS	验证授权书签名	RSA_verify†	253	Tss2_Sys_LoadExternal	280 937	280 103
				Tss2_Sys_VerifySignature	417 147	419 750
vTPM		RSA_verify†	336	Tss2_Sys_LoadExternal	302 661	298 289
				Tss2_Sys_VerifySignature	457 471	418 122
vTPM	令牌请求消息签名	Tspi_Context_LoadKeyByUUID	789 352	Tss2_Sys_Load	364 291	291 580
		Tspi_TPM_Quote§	207 373			
		Tspi_Context_LoadKeyByUUID	710 007	Tss2_Sys_Quote§	450 861	436 086
		Tspi_TPM_Quote2§	201 887			
AS	验证令牌请求签名	RSA_verify†	191	Tss2_Sys_LoadExternal	293 320	284 083
				Tss2_Sys_VerifySignature	419 880	417 992
AS	时间令牌签名	Tspi_Context_LoadKeyByUUID	43 015	Tss2_Sys_Load	281 112	279 623
		Tspi_TPM_Quote§	6 361			
		Tspi_Context_LoadKeyByUUID	28 668	Tss2_Sys_Quote§	422 678	432 271
		Tspi_TPM_Quote2§	21 279			
vTPM	验证令牌签名	RSA_verify†	441	Tss2_Sys_LoadExternal	293 898	299 854
				Tss2_Sys_VerifySignature	418 649	436 756
vTPM	远程证明签名	RSA_private_encrypt†	12 911	RSA_private_encrypt†	66 458	
				EcDaa‡		48
				EcPointMultiply‡		819
				SchnorrEcc‡		4 955
验证者	验证证明签名	RSA_public_decrypt†	127	RSA_public_decrypt†	139	
		RSA_verify†	152	Tss2_Sys_LoadExternal	355 003	
				Tss2_Sys_VerifySignature	441 208	
				ValidateSignatureEcSchnorr‡		2 201
				ValidateSignatureEcSchnorr1‡		3 235

§ 我们使用 *Quote* 对授权书、时间令牌及其请求消息进行签名,把签名对象当作外部数据,选定相对固定的 PCR 值

† OpenSSL 接口

‡ 直接在代理程序中实现的函数,参考了部分 TPM 2.0 模拟器的代码

从表 2 可见,调用 TSS 2.0 SysAPI 所用时间大约是 TSS 1.2 TSPI 的 10 倍,主要原因是 TSS 2.0 有一个中间层 RM,RM 管理着上下文,负责 TPM 对象、会话和一些队列的现场缓存与恢复,以扩展 TPM 非常有限的内存,花费了额外时间.RM 也几乎消除了 TPM 2.0 中 RSA 和 ECC 的效率差异.另外,由于 swtpm 1.2 实例直接运行在宿主机上,vTPM 代理和 swtpm 1.2 之间的交互必须经过 QEMU-KVM,故 vTPM 代理调用 `Tspi_Conext_LoadKeyByUUID` 及 `Tspi_TPM_Quote` 的时间是 pTPM 和 AS 所用时间的 20 倍左右.另外,还可以看到,调用 OpenSSL 接口和在代理程序中直接编程实现的函数,所用时间比 TSS API 要少.

6.3 与TCG可信虚拟平台两层深度认证的性能对比

在 DTE 中,pTPM 对 vTPM 进行一次信任扩展,vTPM 即可签署多次远程证明,故我们主要考虑令牌获取和远程证明的时间,以此来衡量 DTE 的性能,并与 TCG TVP 的两层深度认证作对比,结果见表 3.TCG TVP 进行一次深度认证,验证者可确定虚拟机及其宿主机的安全状况.在 DTE 里,若将 pTPM 的平台配置寄存器(platform configure register,简称 PCR)映射到 vTPM 中,vTPM 进行一次远程证明,则验证者也可确定虚拟机及其宿主平台的安全状况.

Table 3 The comparison of DTE performance with TCG two layers deep attestation

表 3 DTE 和 TCG 可信虚拟平台两层深度认证性能对比

	动态信任扩展 DTE				TCG 可信虚拟平台两层深度认证			
过程	子过程	时间(μs) (1.2 RSA)	时间(μs) (2.0 RSA)	时间(μs) (2.0 ECC)	子过程	时间(μs) (1.2 RSA)	时间(μs) (2.0 RSA)	时间(μs) (2.0 ECC)
生成完整性报告	vTPM 签名令牌请求消息	954 309§	815 152	727 666	生成第 *N* 层完整性报告†	954 309	815 152	727 666
	AS 验签令牌请求消息	191	713 199	702 075				
	AS 签名时间令牌	49 661	703 790	711 894				
	vTPM 验签时间令牌	441	712 547	736 610	生成第 *N*–1 层完整性报告†	48 317	709 225	714 225
	vTPM 签名完整性报告	12 911	66 458	5 822				
	合计	1 017 513	3 011 146	2 884 067	合计	1 002 626	1 524 377	1 441 891
验证报告	验签时间令牌	109	718 726	713 452	验证第 *N* 层报告‡	109	718 726	713 452
	验签完整性报告	279	796 349	5 436	验证第 *N*–1 层报告‡	109	718 726	713 452
	合计	388	1 515 075	718 888	合计	218	1 437 452	1 426 904

§ 等于表 2 中调用“`...UUID,..._Quote`”和“`...UUID,..._Quote2`”所用时间的平均值,DTE 其他时间为表 2 中对应接口或函数所用时间之和

† 生成第 *N* 层完整性报告所用时间等同于表 2 中 vTPM 签名令牌请求消息所用时间,生成第 *N*–1 层完整性报告所用时间等同于表 2 中 pTPM 签名授权书所用时间

‡ 等同于表 2 中验证者验签令牌所用时间

从表 3 可见,针对 TPM 1.2 RSA,DTE 生成完整性报告所用时间与 TCG TVP 深度认证大约相同.针对 TPM 2.0 RSA 和 ECC,DTE 生成完整性报告所用时间约是 TCG TVP 深度认证的 2 倍.若 vTPM 对完整性报告的签名由 vTPM 实例自己完成,而非像原型系统由 vTPM 代理程序完成,那么生成报告需要使用更多的时间.但在 AS 验签时间令牌请求消息和 vTPM 验签时间令牌的操作,可不调用 TSS API,而是由代理程序直接使用公钥在外部完成,以加速操作.针对 TPM 1.2 和 2.0 RSA,DTE 验证完整性报告所用时间与 TCG TVP 深度认证大约相同.针对 TPM 2.0 ECC,DTE 验证完整性报告所用时间甚至比 TCG TVP 还少,这主要是因为在 DTE 原型系统中,代理程序直接使用公钥在外部完成验签操作.

6.4 vTPM频繁迁移时的性能分析

增加 vTPM 状态管理模块和迁移模块,在基于 KVM 的两个计算节点之间,实现 vTPM 与虚拟机相绑定的动态迁移,保证迁移后 vTPM 的可用性和安全状态的一致性.为了分析 DTE 在 vTPM 频繁迁移时的性能状况,我们假设 vTPM 在两个计算节点之间迁移了 m 次,每次迁移完成后,vTPM 进行 p 次远程证明,并与基于传统证书链的信任扩展进行对比分析,结果见表 4.

每次迁移发生时,在 DTE 里,有 vTPM 获取新 pTPM 授权书的过程,尽管基于证书链的信任扩展机制没有这一过程,但需要重新向 CA 获取 IK 证书.获取 CA 证书所用时间受到 CA 服务器性能、网络等诸多因素的影响,在此,我们假设 CA 签署 IK 证书所用时间与 pTPM 对授权书签名相同,vTPM 验证 IK 证书所用时间与验证授权书相同.故获取新 pTPM 授权,并不会增加 DTE 的时间负担.此外,在迁移时,若旧的授权书未过期,则 DTE 需要旧 pTPM 告知 AS 撤销授权,但基于证书链的信任扩展机制也需要告知 CA 将旧的 IK 证书加入到 CRL 中.故撤销旧 pTPM 授权,也不会增加 DTE 的时间负担.

在迁移完成后,vTPM 每次进行远程证明时,DTE 需要向 AS 获取时间令牌,而基于证书链的信任扩展机制没有这一过程.DTE 获取令牌与 vTPM 签署证明报告所用时间之和,与基于证书链的信任扩展机制里签署两层完整性报告所用时间的对比,以及两种机制里验证报告所用时间的对比,在第 6.3 节中已有分析.从功能接口调用性能测试结果、与 TCG TVP 深度认证的对比以及对 vTPM 进行频繁迁移时的性能分析来看,DTE 是可行的.另外,还可通过简化 TPM 上下文缓存管理来提高 DTE 的性能.

Table 4 The DTE performance with frequent vTPM migrations

表 4 vTPM 频繁迁移时 DTE 的性能表现

动态信任扩展 DTE					基于证书链的信任扩展				
过程	次数	时间(μs) (1.2 RSA)	时间(μs) (2.0 RSA)	时间(μs) (2.0 ECC)	过程	次数	时间(μs) (1.2 RSA)	时间(μs) (2.0 RSA)	时间(μs) (2.0 ECC)
授权书签名	*m*	48 317§	709 225	714 225	CA 签署 IK 证书	*m*	48 317	709 225	714 225
vTPM 验证授权书	*m*	336	760 132	716 411	验证 IK 证书	*m*	336	760 132	716 411
令牌获取†	*mp*	1 004 602	2 944 688	2 878 245	无	0	0	0	0
vTPM 签署报告	*mp*	12 911	66 458	5 822	生成两层报告‡	*mp*	1 002 626	1 524 377	1 441 891
验证报告*	*mp*	388	1 515 075	718 888	验证两层报告‡	*mp*	218	1 437 452	1 426 904

§ 等于表 2 中 pTPM 对授权书签名调用“`...UUID,..._Quote`”和“`...UUID,..._Quote2`”所用时间的平均值

† 为表 3 中 vTPM 签名令牌请求消息、AS 验签令牌请求消息、AS 签名时间令牌、vTPM 验签时间令牌所用时间之和

‡ 为表 3 中生成第 *N* 层、第 *N*–1 层完整性报告以及验证第 *N* 层、第 *N*–1 层完整性报告所用时间之和

* 为表 3 中验签时间令牌、验签完整性报告所用时间之和

7 相关工作

Berger 等人[2]还提出了第 3 种机制,建立一个本地 CA,用以给 vTPM 颁发证书.TCG 在可信虚拟平台 TVP 规范[3]里也提出了一种类似的机制,在特权虚拟机或者虚拟机监控程序里建立本地 CA,用以给 vTPM 颁发证书.但是,这两种机制都没有描述本地 CA 与全局 CA 和 pTPM 之间的关系,vTPM 和 pTPM 之间并没有建立可信关联.

IBM 公司定义过一些 TPM 虚拟化命令[11],使用这些命令以基于 pTPM 创建、删除、挂起、还原、迁移 vTPM 实例.England 等人[12]和 Stumpf 等人[13]都提出了相似的方法,让多个虚拟机通过一个控制结构体安全地共享一个 pTPM.Sadeghi 等人[14]提出了一种基于属性设计的 vTPM,以提高其可维护性和适用性.Aziz 等人[15]提出通过一个远程服务器,为虚拟机提供 TPM 功能.Liu 等人[16]提出了一种 vTPM 云架构,没有 TPM 模块的平台可以访问 TPM 云,以获得 TPM 功能.这些工作都主要集中在如何为虚拟机提供 TPM 功能上,并没有考虑如何将 pTPM 的信任扩展到 vTPM 或者在没有 pTPM 的情况下如何构建可信根.

8 总 结

本文提出了一种适用于 vTPM 频繁迁移的动态可信拓展方法 DTE.DTE 将 vTPM 看作是 pTPM 的一个代理,利用从 AS 处获得的时间令牌,进行虚拟机的远程证明.DTE 在 vTPM 和 pTPM 之间建立了紧密的安全关联,但又能明显区分这两种不同安全强度的 TPM.当 vTPM 发生迁移时,旧宿主机上的 pTPM 通知 AS,停止授予令牌给 vTPM,可及时撤销信任扩展.迁移完成之后,新宿主机上的 pTPM 会对 vTPM 进行新的信任授权,vTPM 无需生成新的 IK、申请新的 IK 证书,而且在旧宿主机上生成的完整性报告签名依然有效.从原型系统的性能测试结果来看,DTE 远程证明的签名与验签操作均可在可接受的时间内完成.

在 DTE 中,vTPM 是 pTPM 对认证数据进行签名的一个代理.DTE 是代理签名的一种实际应用,代理签名最早由 Masahiro 等人[17,18]在 1996 年提出.尽管 DTE 的过程和算法已在本文中有详细的阐述,但仍有一些问题值得探讨.例如:(1) 若 vTPM 已被迁移到新的宿主机上,该主机与旧主机对应不同的 AS,那么 DTE 在这种跨 AS 的情况下应如何工作;(2) 如何将 DTE 与 PCR 扩展事件日志传输、验证者检查平台部件完整性与标准值符合性相结合;(3) DTE 如何支持直接匿名验证 DAA(direct anonymous attestation).这些都是我们后续的工作.

References:

[1] Trusted Computing Group (TCG). TCG Specification Architecture Overview pecification Revision 1.4. 2007. https://www.trustedcomputinggroup.org/wp-content/uploads/TCG_1_4_Architecture_Overview.pdf

[2] Berger S, Cáceres R, Goldman KA, Perez R, Sailer R, van Doorn L. VTPM: Virtualizing the trusted platform module. In: Proc. of the 15th USENIX Security Symp., Security 2006. Berkeley: USENIX Association, 2006. 305–320. https://www.usenix.org/legacy/event/sec06/tech/full_papers/berger/berger.pdf

[3] Trusted Computing Group (TCG). Virtualized trusted platform architecture specification version 1.0 revision 0.26. 2011. http://www.trustedcomputinggroup.org/wp-content/uploads/TCG_VPWG_Architecture_V1-0_R0-26_FINAL.pdf

[4] Danev B, Masti RJ, Karame GO, Capkun S. Enabling secure VM-VTPM migration in private clouds. In: Proc. of the 27th Annual Computer Security Applications Conf., ACSAC 2011. New York: ACM, 2011. 187–196. [doi: 10.1145/2076732.2076759]

[5] Strasser M. PeterHuewe tpm-emulator v0.7.4. The famous TPM-emulator. 2014. https://github.com/PeterHuewe/tpm-emulator

[6] Int'l Business Machines Corp (IBM). TrouSerS 0.3.13. An open-source TCG software stack implementation. 2014. https://sourceforge.net/projects/trousers/

[7] Int'l Business Machines Corp (IBM). LibTPMs v0.5.2.1. A library that targets the integration of TPM functionality into hypervisors, primarily into qemu. 2015. https://github.com/stefanberger/libtpms

[8] Int'l Business Machines Corp (IBM). Software-Based TPM Emulator (SWTPM) 0.7.4. The package provides socket interfaces and the Linux cuse interface to LibTPMs for the creation of mulitple native/dev/vtpm* devices. 2011. https://sourceforge.net/projects/tpm-emulator.berlios/

[9] Intel. TPM2.0-TSS 0.97. Trusted platform module 2.0 software stack. 2015. https://github.com/01org/TPM2.0-TSS

[10] Int'l Business Machines Corp (IBM). IBM's Software TPM 2.0 477. It is based on the TPM specification parts 3 and 4 source code donated by microsoft, with additional files to complete the implementation. 2015. https://sourceforge.net/projects/ibmswtpm2/

[11] International Business Machines Corporation (IBM). TPM main part 3 IBM commands specification version 1.2 revision 36. 2008. http://researcher.watson.ibm.com/researcher/files/us-kgoldman/mainP3IBMCommandsrev36.pdf

[12] England P, Loeser J. Para-Virtualized TPM sharing. In: Lipp P, Sadeghi AR, Koch KM, eds. Trusted Computing-Challenges and Applications, the Proc. of the 1st Int'l Conf. on Trusted Computing and Trust in Information Technologies, Trust 2008. LNCS 4968, Berlin, Heidelberg: Springer-Verlag, 2008. 119–132. [doi: 10.1007/978-3-540-68979-9_9]

[13] Stumpf F, Eckert C. Enhancing trusted platform modules with hardware-based virtualization techniques. In: Proc. of the 2nd Int'l Conf. on Emerging Security Information, Systems and Technologies (SECUWARE 2008). IEEE Computer Society, 2008. 1–9. [doi: 10.1109/SECURWARE.2008.23]

[14] Sadeghi AR, Stüble C, Winandy M. Property-Based TPM virtualization. In: Wu TC, Lei CL, Rijmen V, Lee DT, eds. Proc. of the 11th Int'l Conf., ISC 2008. LNCS 5222, Berlin, Heidelberg: Springer-Verlag, 2008. 1–16. [doi: 10.1007/978-3-540-85886-7_1]

[15] Aziz NA, Khalid PS. Utilizing TPM functionalities on remote server. In: Yeo SS, Pan Y, Lee YS, Chang HB, eds. Computer Science and its Applications, CSA 2012. LNEE 203, Springer Netherlands, 2012. 3–12. [doi: 10.1007/978-94-007-5699-1_1]

[16] Liu D, Lee J, Jang J, Nepal S, Zic J. A cloud architecture of virtual trusted platform modules. In: Proc. of the 8th IEEE/IFIP Int'l Conf. on Embedded and Ubiquitous Computing (EUC). Washington: IEEE Computer Society, 2010. 804–811. [doi: 10.1109/EUC.2010.125]

[17] Masahiro M, Usuda K, Okamoto E. Proxy signatures: Delegation of the power to sign messages. IEICE Trans. on Fundamentals of Electronics Communications and Computer Sciences, 1996,E79-A(9):1338–1354.

[18] Masahiro M, Usuda K, Okamoto E. Proxy signatures for delegating signing operation. In: Proc. of the 3rd ACM Conf. on Computer and Communications Security, CCS'96. New York: ACM, 1996. 48–57. [doi: 10.1145/238168.238185]

余发江(1980－),男,重庆人,博士,副教授,CCF 专业会员,主要研究领域为可信计算,系统安全,密码应用.

张焕国(1945－),男,教授,博士生导师,CCF 高级会员,主要研究领域为信息安全,可信计算,密码学.

陈列(1991－),男,学士,主要研究领域为可信计算,云计算.