ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.6, June 2009, pp.1625−1641
doi: 10.3724/SP.J.1001.2009.03343

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

基于组件属性的远程证明*

秦 宇[1,2+], 冯登国[1]

[1](中国科学院 软件研究所 信息安全国家重点实验室,北京 100190)

[2](中国科学院 研究生院,北京 100049)

Component Property Based Remote Attestation

QIN Yu[1,2+], FENG Deng-Guo[1]

[1](State Key Laboratory of Information Security, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: qin_yu@is.iscas.ac.cn, http://is.iscas.ac.cn

Qin Y, Feng DG. Component property based remote attestation. *Journal of Software*, 2009,20(6):1625−1641.
http://www.jos.org.cn/1000-9825/3343.htm

Abstract: A fine-grained property attestation based on the components is proposed to prove that the user platform satisfies the security property predefined by remote relying party. Compared with other remote attestation schemes, CPBA (component property based attestation) has the advantage of semantic and property expression to some extent. It is not only more fine-grained and extensive, but also easier to implement issuing, verifying and revoking the property certificate. CPBA guarantees the authenticity of attestation by component commitment, and protects the privacy of platform components by zero-knowledge proof. It is proved secure in Random Oracle Model under strong RSA Assumption. The experimental result of its prototype system indicates that CPBA is a flexible, usable, highly efficient attestation, and has no influence on system performance.

Key words: trusted computing; TPM (trusted platform module); remote attestation; property-base attestation; strong RSA assumption; security proof

摘 要: 提出了一个组件级的细粒度属性证明方案,用于向远程依赖方证明用户平台满足某种安全属性.与现有的远程证明方案相比,组件属性远程证明具有一定的语义和属性表述性等优势.该方案不但证明粒度细和扩展性强,而且属性证书的颁发、验证和撤销实现简单;本方案以组件承诺的方法保证属性证明的真实性,采用零知识证明实现平台组件的隐私性.基于强 RSA 假设,在 Random Oracle 模型下可被证明是安全的.实现的原型系统实验结果表明,组件属性证明是一种灵活、实用、高效的证明,对系统性能没有影响.

关键词: 可信计算;可信平台模块;远程证明;基于属性的证明;强 RSA 假设;安全性证明

中图法分类号: TP309 **文献标识码**: A

为了解决计算机和网络结构上的不安全,从根本上提高终端系统的安全性和提升计算机自身免疫力,必须

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z412 (国家高技术研究发展计划(863))

Received 2007-11-26; Accepted 2008-04-02

从芯片、硬件结构和操作系统等方面综合采取措施构建可信计算环境,由此产生了可信计算.可信计算技术是目前计算机安全领域的主要发展趋势之一,以欧美厂商为主体的国外厂商和科研机构(IBM,MicroSoft,SUN,Infineon,HP,Intel,AMD等)在国际可信计算组织(Trusted Computing Group,简称TCG)协调下,正在迅速推进和完善可信计算平台的TPM(trusted platform module)系列标准[1−3].

以证明计算平台可信为目标的远程证明是可信计算的重要特色功能之一,已经受到国内外科研机构的广泛关注.远程证明已经取得了众多研究成果,其中较为典型的研究成果有IBM的相互证明系统[4],并且开发出直接证明的原型系统tcglinux,该项工作已经成为了远程证明的研究基础.其他远程证明方案有Dartmouth大学的Enforcer/Bear隔间证明方案[5]、Carnegie Mellon大学的基于软件的证明方案(software-based attestation,简称SWATT)[6]、Stanford大学的Terra体系结构下基于可信虚拟机监控器的应用程序证明[7],还有基于语义的远程证明[8]和基于属性的远程证明[9,10].这些工作不同程度地对TCG远程证明[1]各个方面的不足进行了改进和扩展.在众多的远程证明方案中,基于属性的远程证明具有明显的发展优势,它克服了TCG远程证明的证明复杂、隐私泄漏和滥用证明结果等缺陷,易于实现远程证明扩展.IBM研究院针对tcglinux的缺陷,在文献[9]中提出了基于属性的远程证明框架,随后,文献[10]给出了属性远程证明的实现方法.Chen提出了基于属性的远程证明协议,简称PBA(property-based attestation)协议[11],这些研究不断地推动了基于属性远程证明的发展.

现有的基于属性的远程证明还存在很多缺陷:1) 现有的属性证明的粒度很粗,都是对整个平台的安全属性进行证明.由于计算平台的运行状态很复杂,对每个配置状态、运行状态给出安全属性评估和映射是非常困难的,粗粒度必然增大了属性的转换和映射的复杂度,这使得基于属性证明具有委托可信第三方验证的委托证明的缺陷;2) 属性的撤销很困难.以 PBA 方案为例,每次证明都要向 Issuer 申请属性证书(*cs*,*ps*),其中,*cs* 为平台配置,*ps* 为平台满足的安全属性.Issuer 为证明平台颁发属性证书容易,但是如果假设操作系统发现新的漏洞,那么原有的一系列配置状态{*cs*}都将不再满足属性 *ps*,由于{*cs*}状态非常复杂,Issuer 将无法撤销这些为不同平台{*cs*}状态颁发的属性证书,因此,PBA 方案验证该属性是否撤销是通过复杂的零知识证明来完成的.除了这两方面的主要问题外,还存在平台隐私保护、抗伪装攻击、实时证明等缺陷.

针对上述属性证明缺陷,我们基于原有的属性证明研究成果,设计了基于组件属性的远程证明方案(component property-based attestation,简称 CPBA).其特点是:1) 组件属性证明是一种具有一定语义和属性表述能力的组件级远程证明,具有细粒度、验证方便、扩展性强等优势;2) 无须平台证明前请求颁发属性证书,属性证书撤销和有效性验证比 PBA 协议简单得多;3) 用零知识证明组件类型和配置,保护平台组件的隐私;4) 组件属性证明能够防止 TPM 远程证明的伪装攻击.

1 组件属性证明模型

PBA 方案用 TPM 的平台配置寄存器(platform configuration register,简称 PCR)来描述系统的状态,然后证书颁发机构(issuer)为某个安全状态颁发属性证书,证明方(attestor)零知识证明当前配置状态的确是属性证书中所描述的状态,从而完成远程证明平台属性的目的.现实生活中,很多应用并不要求证明整个系统状态,而仅仅需要证明某几个组件或少量组件的属性.以网上银行交易为例,一般网银要求登录的客户端浏览器达到某安全属性.用户登录网银前必须向服务器证明其浏览器的安全属性,而没有必要证明与此无关的其他系统状态.组件属性证明将原有的整个系统证明转化为系统中各个硬件配置、系统服务、内核模块、应用程序等一个或多个组件的证明,属性证明的粒度将更细.描述单个组件的配置、颁发和撤销其安全属性都比较容易.组件出厂发布时就附上其安全属性证书,无须证明时由可信第三方临时颁发属性证书.某个组件的属性证书,用户还可以通过证书服务器查询,属性证书的撤销可以采用类似于 X.509 证书撤销的方式.

1.1 证明模型描述

组件属性证明包含组件生产厂商、用户平台、服务提供者、证书发布权威和验证中心这 5 个角色.除了组件生产厂商外,其他 4 个角色参与组件属性证明的远程证明过程.我们用下列符号来表示系统中的各个参与者:

$\mathcal{CA}$:证书发布权威机构,发布、撤销组件属性证书;

$\mathcal{U}$:用户平台,证明协议中的证明者,包括主机$\mathcal{H}$和可信平台模块$\mathcal{M}$两部分;

$\mathcal{SP}$:服务提供者(即证明验证者),提出属性证明要求,验证组件属性证明;

$\mathcal{VC}$:验证中心,验证组件属性是否已被撤销.

可信计算平台的基于组件属性的远程证明方案(CPBA)是属性权威机构为各种类型组件颁发的属性证书,属性证书与软、硬件绑定共同发布,平台证明者根据配置的组件属性证书和 TPM 的完整性度量向服务提供者证明其当前运行配置状态满足一定的安全属性.由图 1 的体系结构可以看出,CPBA 证明由 4 个步骤构成:*CPBA*= {*Setup*,*Attest*,*Verify*,*Check*}.

- Setup:系统各个参与者参数设置,主要是设置证书发布权威和验证中心的参数,证书发布权威为每个组件颁发属性证书;
- Attest:用户平台(包括 TPM 和主机)根据服务提供者的证明请求,计算平台组件属性签名,然后向服务提供者进行远程证明;
- Verify:服务提供者验证 TPM 签名和组件属性签名;
- Check:验证中心检查证明组件属性是否已经被撤销,证明的组件与属性证明承诺的组件是否一致.

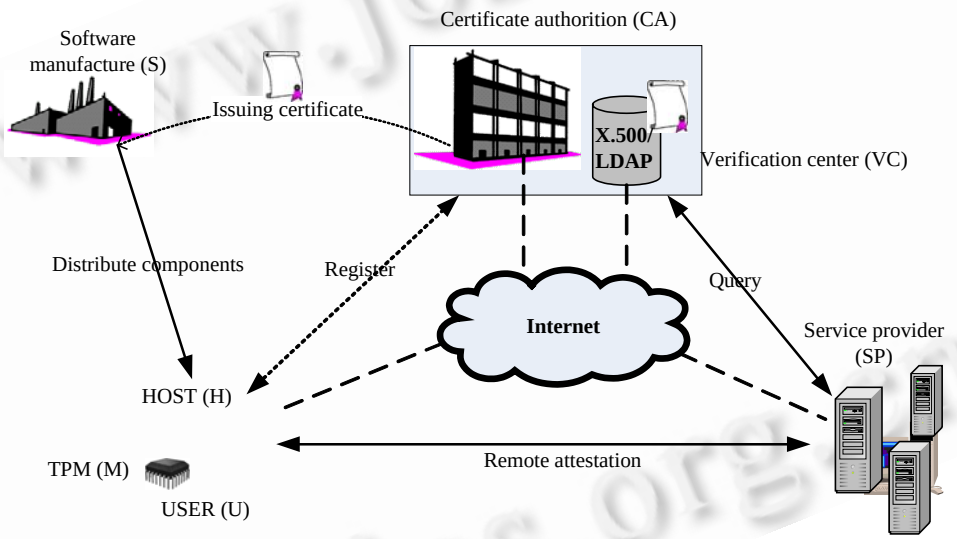


Fig.1 Architecture of component property attestation

图 1 组件属性证明体系结构

1.2 安全性要求

远程证明最基本的安全要求便是要求远程证明数据不可伪造.基于属性的远程证明由于引入了属性,因而要求达到安全需求属性的可撤销性.通过对各种类型的远程证明方案进行比较,将远程证明的安全性要求归纳总结如下:

(1) 不可伪造性:在可信平台模块未被篡改的情况下,任何主机不可能伪造远程证明的数据,使得能够通过远程依赖方的验证过程欺骗依赖方.不可伪造性是任何远程证明方案(包括 TCG 直接证明、委托证明、属性证明和语义证明等等)都必须满足的基本条件.

(2) 组件隐私性:证明某组件(如浏览器)满足一定安全属性,证明结束后,验证方除了知道该组件的安全属性外,无法知道组件的生产厂商、产品类型、版本等配置信息,我们称该远程证明满足组件的隐私性.更进一步地对隐私性进行划分,若证明结束后,验证方明确知道组件类型(如 IE 浏览器),但并不知道具体版本配置(IE 5.0 或 6.0),则称远程证明满足组件配置隐私性;若远程证明结束后,验证方无法知道

组件类型(如 IE 浏览器或 Firefox 浏览器),更不知道具体版本配置,则称远程证明满足组件类型隐私性.很容易看出,组件类型隐私性的安全要求高于组件配置隐私性.

(3) 属性可撤销性:一定配置的组件安全属性并不是固定不变的,随着黑客攻击方法的提高、攻击能力的增加、病毒破坏方式和攻击手段的增强,某一配置的组件就有可能不再满足原有的安全属性,因此就需要对原有颁发的组件属性证书进行撤销.

(4) 抗伪装攻击:远程证明中,平台 A 并不满足远程证明要求,然而平台 A 将证明请求转发给满足证明要求的平台 B,获取平台 B 的证明数据,平台 A 伪装成平台 B 进行欺骗性的远程证明.远程证明必须要求能够防止这种伪装攻击,特别是 TPM 直接匿名证明(direct anonymous attestation,简称 DAA)的情况下,这种伪装的远程证明更加难以觉察.

(5) 实时证明:远程证明存在着证明时刻t_1和事务处理时刻t_2时间间隔(time-to-check,time-to-use,简称 TOCTOU)问题,如果在完成证明后、事务处理之前(即t_1时刻后、t_2时刻之前),平台可信状态改变为不可信,那么远程证明将无效.解决这一问题的思路就是要尽量缩小时间窗口,尽量做到接近实时证明.

2 预备知识

定义 1(强 RSA 假设(strong RSA assumption)). 对于任何概率多项式时间算法,输入 RSA 模 n 和一个随机元素 $z\in Z_n^*$,找到$v>1$ 和u,使其满足$u^v\equiv z(\mathrm{mod}\ n)$,这在计算上是不可行的.更确切地说,对于概率多项式时间算法 $\mathcal{A}_k$,任意多项$p(\cdot)$,满足:

$$\Pr[n\leftarrow RSA\ \mathrm{mod}\ lus(1^k);z\leftarrow Z_n^*;(u,v)\leftarrow \mathcal{A}_k(n,z):v>1\wedge u^v\equiv z\ \mathrm{mod}\ n]<1/p(k),$$

二元组(n,z)则被称为灵活的 RSA 实例(flexible RSA instance).

定义 2(DDH假设(DDH assumption)). 给定群$G=\langle g\rangle,p=ord(g),\delta\in_R G,a,b,c\in_R[0,p-1]$,对于群的阶$p$足够大,则概率分布$\{(\delta,\delta^a,\delta^b,\delta^{ab})\}$和概率分布$\{(\delta,\delta^a,\delta^b,\delta^c)\}$在计算上不可区分.

定义 3(CL签名(camenisch and lysyanskaya signature))[12].

生成CL公私钥,选择强RSA模为$n=pq,p=2p'+1,q=2q'+1,p,q,p',q'$都是素数,且$n$的长度为$l_n$.随机选择$R_0,\ldots,R_{L-1},S,Z\in_R QR_n,(n,p,q)$为私钥,输出公钥$(n,R_0,\ldots,R_{L-1},S,Z)$.

输入消息$(m_0,\ldots,m_{L-1})$, $m_i\in\{0,1\}^{l_m}$.选择随机素数$e(l_e>l_m+2)$,随机数$v(l_v=l_n+l_m+l_k$,其中,l_k为安全参数)计算值A满足 $Z\equiv A^eR_0^{m_0}\ldots R_{L-1}^{m_{L-1}}S^v(\mathrm{mod}\ n)$,$(A,e,v)$就是消息$(m_0,\ldots,m_{L-1})$的CL签名.

CL签名验证时,只需检查消息$(m_0,\ldots,m_{L-1})$的签名(A,e,v)是否满足 $Z\equiv A^eR_0^{m_0}\ldots R_{L-1}^{m_{L-1}}S^v(\mathrm{mod}\,n)$,检查 $2^{l_e-1}<e<2^{l_e}$.

定理 1. CL 签名在 Strong RSA Assumption 下是抗适应性地选择消息攻击安全签名方案.

引理 2. n是强素数的RSA模,$Z\in QR_n$,令$r_i\in_R[n,n^2]$,e_i为没有小于 2^{l_e-1} 的因子, $x_i=z^{e_i r_i}$,$i=1,2,\ldots,k$.对于$(n,e_1,\ldots,x_k,e_1,\ldots,e_k)$,如果与$r_i$无关联的整数$t,c,s_i,|c|<2^{l_e-1}$,满足 $t^c=\prod_{i=1}^{k}x_i^{s_i}$,那么,要么$c|s_i$,要么以至少 1/2 的概率计算出整数$e>1,v,v^e=z$.

引理 2 成立的原因是 $x_i=z^{e_i r_i}$, $t^c=z^{\sum e_i r_i s_i}$,由于$c|e_i,r_i$随机选取,因此,c要么整除全部s_i,满足等式$t^c=z^u$,要么$c|u$必然能够计算出灵活RSA实例(n,z),破解Strong RSA Assumption.详细的证明请参照CL签名方案[12].

3 组件属性证明协议

3.1 组件属性转换

以用户登录网上银行为例,网银安全要求浏览器达到一定的安全属性,设 IE 浏览器标准配置 $\hat{c}_{i,1}$,Firefox 浏

览器标准配置 $\hat{c}_{i,2}$ 和 Netscape 浏览器标准配置 $\hat{c}_{i,3}$,都能达到网上银行要求的安全属性 $\hat{p}_1$.登录网上银行还要求网银客户端控件 $\hat{c}_j$ 必须达到安全属性 $\hat{p}_2$,Windows 系统必须打有 SP2 补丁,即补丁配置 $\hat{c}_k$ 达到属性 $\hat{p}_3$.因此,网银的属性证明要求(property attestation requirement,简称 PAR)可以表示为 $p:=PAR\{\hat{p}_1\wedge\hat{p}_2\wedge\hat{p}_3\}$,以组件属性证明(components property attestation,简称 CPA)形式表示为 $CPA\{(\hat{c}_{i,1}\vee\hat{c}_{i,2}\vee\hat{c}_{i,3})\wedge\hat{c}_j\wedge\hat{c}_k\Rightarrow\hat{p}\}$.

那么通用的组件属性转换表示为 $PAR\{\hat{p}_1\wedge\ldots\wedge\hat{p}_{\bar{n}}\Rightarrow\hat{p}\}$,简记为

$$PAR\left\{\mathop{\wedge}_{i=1}^{\bar{n}}\hat{p}_i\Rightarrow\hat{p}\right\} \tag{1}$$

$CPA\{\hat{c}_1\wedge\ldots\wedge\hat{c}_{\bar{n}}\Rightarrow\hat{p}\}$,简记为 $CPA\left\{\mathop{\wedge}_{i=1}^{\bar{n}}\hat{c}_i\Rightarrow\hat{p}\right\}$,其中 $\hat{c}_i=\hat{c}_{i,1}\vee\ldots\vee\hat{c}_{i,i_k}=\mathop{\vee}_{j=1}^{i_k}\hat{c}_{i,j}$,$\mathcal{J}_i\subseteq\{1,\ldots,i_k\}$表示 $\hat{c}_i$ 可选组件的子集,$\hat{c}_i=\mathop{\vee}_{j\in\mathcal{J}_i}\hat{c}_{i,j}$.

$\hat{c}_i=\mathop{\vee}_{j=1}^{i_k}\hat{c}_{i,j}$ 的含义是 $\hat{c}_i$ 能够满足指定安全属性的可选组件,用可选组件的逻辑析取范式表示.i_k=1 表示无其他可选的组件,包含其他可选组件则 $\hat{c}_i=\mathop{\vee}_{j=1}^{i_k}\hat{c}_{i,j}$ (i_k>1).不失一般性,组件属性转换用组件的逻辑合取范式形式表示,假定要求证明的前k个属性对应的组件由验证方指定,组件属性证明表示为

$$CPA\left\{\left(\mathop{\wedge}_{i=1}^{k}\hat{c}_i\right)\wedge\left(\mathop{\wedge}_{i=k+1}^{\bar{n}}\left(\mathop{\vee}_{j\in\mathcal{J}_i}\hat{c}_{i,j}\right)\right)\Leftrightarrow\hat{p}\right\} \tag{2}$$

组件属性证明就是根据不同的安全需求(以 $CPA\left\{\left(\mathop{\wedge}_{i=1}^{k}\hat{c}_i\right)\wedge\left(\mathop{\wedge}_{i=k+1}^{\bar{n}}\left(\mathop{\vee}_{j\in\mathcal{J}_i}\hat{c}_{i,j}\right)\right)\Rightarrow\hat{p}\right\}$ 表示),要求用户证明其平台运行在一定的可信计算环境,进一步缩小基于属性远程证明的证明范围,将整个平台状态和属性证明精确到了组件级,组件属性远程证明符合现有的软件发布机制、证书撤销和验证机制,契合 TCG 的信任扩展和传递模型、TCG 完整性发布框架,具有广泛的应用价值.

3.2 证明协议

组件属性证明(CPBA)协议相关安全参数为:l_χ(160)表示组件配置度量值的长度,$l_{\hat{p}}$(160) 表示组件属性值的长度,l_n(2048)表示RSA模的长度,l_e(368),l'_e(120) 表示权威机构C颁发的CL签名方案与e相关的安全参数长度,l_v(2536)是CL证书中v的长度,$l_\varnothing$(80)用于控制零知识证明统计属性的安全参数,l_H(160)是用于Fiat-Shamir启发式的 hash 函数的输出长度,H 是无碰撞散列函数 $H:\{0,1\}^*\to\{0,1\}^{l_H}$.这些安全参数要求满足:

$$l_e\geq\max\{l_\chi,l_{\hat{p}}\}+2,$$

$$l_v\geq l_n+\max\{l_\chi,l_{\hat{p}}\}+l_\varnothing.$$

TCG的TNC规范[13]为每个厂商的软件产品定义了一个ID,用 32 bits数表示.前 24 bits表示厂商ID(vendor ID),由厂商向TCG注册;后 8 bits表示产品子类型,由厂商自行定义具体的产品.证书中的id采用TCG的组件产品ID,l_{id}(32)为组件ID长度,也可以采用其他组件ID定义形式.下面依次阐述$CPBA$={$Setup$,$Attest$,$Verify$,$Check$}的各个操作步骤.

3.2.1 Setup

属性证书颁发(现采用 CL 签名):证书发布权威机构$\mathcal{CA}$的 RSA 模为 $n=pq$,$p=2p'+1$,$q=2q'+1$,p,q,p',q'都是素数,且n的长度为l_n.

令g_0是模n的二次剩余群QR_n的生成元,随机选择$x_0,x_1,x_2,x_g,x_h,x_s,x_z\in[1,p'q']$,计算:

$$g=g_0^{x_g}\mod n,$$

$$h=g_0^{x_h}\mod n,$$

$$S=h^{x_s}\mod n,$$

$$Z=h^{x_z}\mod n,$$

$$R_0 = S^{x_0} \bmod n,$$
$$R_1 = S^{x_1} \bmod n,$$
$$R_2 = S^{x_2} \bmod n.$$

因此$g,h,S,Z,R_0,R_1,R_2 \in \langle g_0 \rangle$,$\mathcal{CA}$的公开参数 $parm_{\mathcal{CA}} = (n, g_0, g, h, S, Z, R_0, R_1, R_2)$.设置 $parm_{\mathcal{VC}} = (g_0, g, h, n, y)$ 为验证中心$\mathcal{VC}$的公开参数,其中,$y=g^x \bmod n$是它的公钥,x是私钥.假设组件 $\hat{c}_i$ 配置 $(id_i, \chi_i, \hat{p}_i)$,$id_i$是组件 $\hat{c}_i$ 的组件ID,χ_i为组件 $\hat{c}_i$ 的度量值, $\hat{p}_i$ 为组件满足的安全属性.$\mathcal{CA}$为组件 $\hat{c}_i$ 颁发的CL签名属性证书 $(id_i, \chi_i, \hat{p}_i)$,$(A_i, e_i, v_i)$,满足:

$$Z \equiv A_i^{e_i} R_0^{id_i} R_1^{\chi_i} R_2^{\hat{p}_i} S^{v_i} \bmod n \tag{3}$$

$\mathcal{CA}$在循环群 $G=\langle g \rangle$上为$\mathcal{U}$分配 $(sk_{\mathcal{U}}, vk_{\mathcal{U}})$, $vk_{\mathcal{U}} = g^{sk_{\mathcal{U}}} \bmod n$;同样为$\mathcal{SP},\mathcal{VC}$分配公私钥对 $(sk_{\mathcal{SP}}, vk_{\mathcal{SP}})$, $(sk_{\mathcal{VC}}, vk_{\mathcal{VC}})$.其中, $sk_{\mathcal{VC}} = x$, $vk_{\mathcal{VC}} = y = g^x \bmod n$.私钥各自秘密保存,公钥公开发布.

3.2.2 Attest

组件属性证明协议:

- 用户($\mathcal{U}$)请求服务提供者($\mathcal{SP}$)提供服务,将 TPM 的 AIK 证书发送给 SP,证书中包含 TPM 的 AIK 公钥 $vk_{\mathcal{M}}$.
- $\mathcal{SP}$ 将证明属性请求 $PAR\left\{\bigwedge_{i=1}^{\bar{n}} \hat{p}_i \Rightarrow \hat{p}\right\}$ 和挑战随机数N_v发送给$\mathcal{U}$,要求$\mathcal{U}$证明计算平台达到安全属性 $\hat{p}$.
- $\mathcal{U}$进行组件属性证明 $CPA\left\{\bigwedge_{i=1}^{\bar{n}} \hat{c}_i \Rightarrow \hat{p}\right\} = CPA\left\{\left(\bigwedge_{i=1}^{k} \hat{c}_i\right) \wedge \left(\bigwedge_{i=k+1}^{\bar{n}} \left(\bigvee_{j \in \mathcal{J}_i} \hat{c}_{i,j}\right)\right) \Rightarrow \hat{p}\right\}$.

TPM 度量和签名

1. TPM进行系统组件度量,执行系统组件度量算法,组件 $\hat{c}_i$ (i=1,…,k)的度量值为 χ_i.对于组件 $\hat{c}_i$ (i=k+1,…,n),用户平台$\mathcal{U}$配备使用的可选组件为 $\hat{c}_{i,i^*}$,TPM对 $\hat{c}_{i,i^*}$ 的度量值为χ_i.由于平台没有配置,或未运行除 $\hat{c}_{i,i^*}$ 以外的其他 $\hat{c}_i$ 可选组件,TPM不度量 $\hat{c}_i$ 其他可选组件 $\hat{c}_{i,j}$, $j \in \mathcal{J}_i, j \neq i^*$;
2. TPM选择随机数 $N_t \in_R \{0,1\}^{l_\varnothing}$ 和随机数 $w_i \in \{0,1\}^{l_n + l_\varnothing}$,计算证明组件的承诺 $C_i = g_0^{id_i} g^{\chi_i} h^{w_i} \bmod n$ (i=1,…,n),计算 $K = DH(vk_{\mathcal{SP}}, sk_{\mathcal{U}})$, $h_i = h^{w_i}$, $g_i = g^{w_i}$,然后,TPM 用 sk_{TPM} 对组件承诺签名,$\sigma = Sign_{sk_{TPM}}(parm_C \| C_1 \| ... \| C_{\bar{n}} \| K \| N_v \| N_t)$;
3. TPM将计算结果C_i,h_i,g_i,K,σ发送给主机.

主机计算组件属性证明签名

4. 选择随机数 $u_i \in_R \{0,1\}^{l_n + l_\varnothing}$,计算 $a_i = g^{u_i} \bmod n$, $b_i = id_i \cdot y^{u_i} \bmod n$ ($i = 1,...,\bar{n}$).
5. $\mathcal{U}$执行知识签名协议(signature of knowledge protocol)

$$\left.\begin{aligned} SPK\{(id_i, \chi_i, v_i, e_i, w_i, r_i, e_i w_i, e_i e_i, e_i r_i)_{i=1,...,n} \mid \bigwedge_{i=1}^{\bar{n}} (Z / R_2^{\hat{p}_i} \equiv T_{i,1}^{e_i} R_0^{id_i} R_1^{\chi_i} S^{v_i} h^{-e_i w_i} \bmod n \wedge \\ T_{i,2} = g^{w_i} h^{e_i} g_0^{r_i} \bmod n \wedge 1 = T_{i,2}^{-e_i} g^{e_i w_i} h^{e_i e_i} g_0^{e_i r_i} \bmod n \wedge C_i = g_0^{id_i} g^{\chi_i} h^{w_i} \bmod n \wedge \\ \chi_i \in \{0,1\}^{l_\chi + l_\varnothing + l_{\mathcal{H}} + 2} \wedge (e_i - 2^{l_e}) \in \{0,1\}^{l'_e + l_\varnothing + l_{\mathcal{H}} + 1})\}(N_v, N_t) \end{aligned}\right\} \tag{4}$$

(a) 组件 $\hat{c}_i$ 的配置 $(id_i, \chi_i, \hat{p}_i)$,属性证书为 (A_i, e_i, v_i),主机 $\mathcal{H}$ 随机选择 $r_i \in \{0,1\}^{l_n + l_\varnothing}$,计算 $Z'_i = Z / R_2^{\hat{p}_i} \bmod n$, $T_{i,1} = A_i h_i \bmod n$, $T_{i,2} = g_i h^{e_i} g_0^{r_i} \bmod n$.

(b) TPM 选择随机数: $r_{id_i} \in_R \{0,1\}^{l_{id} + l_\varnothing + l_H}$, $r_{\chi_i} \in_R \{0,1\}^{l_\chi + l_\varnothing + l_H}$, $r_{w_i} \in_R \{0,1\}^{l_n + 2l_\varnothing + l_H}$,然后计算 $\tilde{C}_i = g_0^{r_{id_i}} g^{r_{\chi_i}} h^{r_{w_i}} \bmod n$,TPM 将 $\tilde{C}_i$ 发送给主机.

(c) $\mathcal{H}$选择随机数:

$$r_{v_i} \in_R \{0,1\}^{l_v+l_\varnothing+l_H},$$
$$r_{e_i} \in_R \{0,1\}^{l'_e+l_\varnothing+l_H},$$
$$r_{r_i} \in_R \{0,1\}^{l_n+2l_\varnothing+l_H},$$
$$r_{e_ie_i} \in_R \{0,1\}^{2l_e+l_\varnothing+l_H+1},$$
$$r_{e_iw_i}, r_{e_ir_i} \in_R \{0,1\}^{l_e+l_n+2l_\varnothing+l_H+1}.$$

$\mathcal{H}$ 计算:

$$\tilde{Z}'_i = T_{i,1}^{r_{e_i}} R_0^{r_{id_i}} R_1^{r_{\chi_i}} S^{r_{v_i}} h^{-r_{e_iw_i}} \bmod n,$$
$$\tilde{T}_{i,2} = g^{r_{w_i}} h^{r_{e_i}} g_0^{r_{r_i}} \bmod n,$$
$$\tilde{T}'_{i,2} = T_{i,2}^{-r_{e_i}} g^{r_{e_iw_i}} h^{r_{e_ie_i}} g_0^{r_{e_ir_i}} \bmod n.$$

$c_{\mathcal{H}} = H(parm_C \| (C_i \| Z'_i \| T_{i,1} \| T_{i,2})_{i=1,\ldots,\bar{n}} \| (\tilde{C}_i \| \tilde{Z}'_i \| \tilde{T}_{i,2} \| \tilde{T}'_{i,2})_{i=1,\ldots,\bar{n}} \| N_v)$,主机把 $c_{\mathcal{H}}$ 发送给 TPM.

(d) TPM 计算 $c = H(c_h \| N_t) \in [0, 2^{l_H-1}]$,然后 TPM 计算:

$$s_{id_i} = r_{id_i} + c \cdot id_i,$$
$$s_{\chi_i} = r_{\chi_i} + c \cdot \chi_i,$$
$$s_{w_i} = r_{w_i} + c \cdot w_i.$$

将c,N_t, s_{id_i}, s_{χ_i}, s_{w_i} 发送给主机

(e) $\mathcal{H}$计算知识签名:

$$s_{v_i} = r_{v_i} + c \cdot v_i,$$
$$s_{e_i} = r_{e_i} + c \cdot (e_i - 2^{l_e-1}),$$
$$s_{r_i} = r_{r_i} + c \cdot r_i,$$
$$s_{e_iw_i} = r_{e_iw_i} + c \cdot e_i \cdot w_i,$$
$$s_{e_ie_i} = r_{e_ie_i} + c \cdot e_i^2,$$
$$s_{e_ir_i} = r_{e_ir_i} + c \cdot e_i \cdot r_i.$$

6. $\mathcal{H}$计算出的组件属性签名为

$$\sigma_{CPBA} = (\sigma, N_t, K, c, (C_i, T_{i,1}, T_{i,2}, s_{id_i}, s_{\chi_i}, s_{v_i}, s_{e_i}, s_{w_i}, s_{r_i}, s_{e_iw_i}, s_{e_ie_i}, s_{e_ir_i})_{i=1,\ldots,\bar{n}}) \quad (5)$$

用户平台$\mathcal{U}$将组件属性签名σ_{CPBA}发送给$\mathcal{SP}$.

3.2.3 Verify

$\mathcal{SP}$验证 TPM 签名、知识签名,然后将数据 $(C_i, T_{i,1}, a_i, b_i, \hat{p}_i)_{i=1,\ldots,n}$ 发送给$\mathcal{VC}$验证组件属性证书是否被撤销,执行 Check 算法.如果组件属性证书未被撤销,最后验证协商密钥 K.

1. 验证 TPM 签名

检查AIK证书和新鲜性随机数N_v的有效性,然后 $Verify_{vk_{TPM}}(parm_C \| C_1 \| \ldots \| C_n \| K \| N_v \| N_t, \sigma)$.

2. 验证组件属性签名

$\mathcal{SP}$使用公钥$parm_C$=(n,g_0,g,h,S,Z,R_0,R_1,R_2),对消息N_v,N_t验证组件属性签名σ_{CPBA}.

(a) $\mathcal{SP}$把 $(C_i, T_{i,1}, a_i, b_i, \hat{p}_i)_{i=1,\ldots,n}$ 发送给$\mathcal{VC}$,验证组件属性证书是否被撤销,$\mathcal{VC}$执行 Check 过程.

(b) $\mathcal{SP}$计算 $Z'_i = Z / R_2^{\hat{p}_i} \bmod n$,然后计算:

$$\hat{C}_i = C_i^{-c} g_0^{s_{id_i}} g^{s_{\chi_i}} h^{s_{w_i}} \bmod n,$$
$$\hat{Z}'_i = Z'^{-c}_i T_{i,1}^{s_{e_i}+c\cdot 2^{l_e-1}} R_0^{s_{id_i}} R_1^{s_{\chi_i}} S^{s_{v_i}} h^{-s_{e_iw_i}} \bmod n,$$
$$\hat{T}_{i,2} = T_{i,2}^{-c} g^{s_{w_i}} h^{s_{e_i}+c\cdot 2^{l_e-1}} g_0^{s_{r_i}} \bmod n,$$

$$\hat{T}'_{i,2}=T_{i,2}^{-(s_{e_i}+c\cdot 2^{l_e-1})}g^{s_{e_i}w_i}h^{s_{e_i}e_i}g_0^{s_{e_i}r_i} \bmod n.$$

(c) 验证

$$c\overset{?}{=}\mathcal{H}(parm_C\|(C_i\|Z'_i\|T_{i,1}\|T_{i,2})_{i=1,\ldots,\bar{n}}\|(\hat{C}_i\|\hat{Z}'_i\|\hat{T}_{i,2}\|\hat{T}'_{i,2})_{i=1,\ldots,\bar{n}}\|N_v\|N_t),$$

$$C_i\overset{?}{\in}\langle g_0\rangle,$$

$$s_{\chi_i}\overset{?}{\in}\{0,1\}^{l_\chi+l_\varnothing+l_H+1},$$

$$s_{e_i}\in\{0,1\}^{l'_e+l_\varnothing+l_H+1}.$$

3. 验证协商密钥 K, $K\overset{?}{=}DH(vk_{\mathcal{U}},sk_{\mathcal{SP}})$,如果协商密钥相等,则说明没有受到$\mathcal{U}$的伪装攻击.

3.2.4 Check

1. $\mathcal{VC}$计算组件ID, $id_i=b_i/a_i^x \bmod n$,以 $(id_i,\hat{p}_i)$ 为索引查询属性证书库,得到组件 $\hat{c}_i$ 的配置、属性证书: $(id_i,\chi_i,\hat{p}_i)$,(A_i,e_i,v_i);
2. 验证该组件属性和配置是否已经被撤销,如果属性证书未被撤销,则验证承诺是否与该组件匹配,验证中心$\mathcal{VC}$验证下列等式是否成立:

$$Z\times[C_i/(g_0^{id_i}g^{\chi_i})]^{e_i}\overset{?}{=}T_{i,1}^{e_i}R_0^{id_i}R_1^{\chi_i}R_2^{\hat{p}_i}S^{v_i} \tag{6}$$

若验证通过,则说明 U 的证明的组件属性未被撤销,且承诺消息是真实可靠的.

3. $\mathcal{VC}$将属性证书的验证结果传递给$\mathcal{SP}$.

4 安全性证明

4.1 安全模型

Random Oracle模型是由Bellare和Rogaway[14]于1993年提出来的,它是一种非标准化的计算模型.在这个模型中,任何具体的对象,例如哈希函数,都被当作随机对象.哈希函数被作为一个预言返回值,对每一个新的查询,将在均匀分布的输出域上得到一个随机应答.在RO模型下,采用问题归约的方法证明协议的安全性,证明只要存在攻击者以不可忽略的概率可以破解密码协议,我们就可以利用这个攻击者设计出另一种算法,使得这种算法也以不可忽略的概率来解决公开数论上的计算困难问题.

我们将采用random oracle模型描述的实际系统/理想系统(real system/idea system)[15]来证明协议的安全性.组件属性证明系统(real system)参与者:证书发布权威机构$\mathcal{CA}$,用户平台$\mathcal{U}$(主机$\mathcal{H}$和可信平台模块$\mathcal{M}$),服务提供者$\mathcal{SP}$(即证明验证者),验证中心$\mathcal{VC}$.这些参与者相互之间运行密码协议,还存在攻击者$\mathcal{A}$和环境$\mathcal{E}$,$\mathcal{E}$提供给这些参与者输入信息,以及与$\mathcal{A}$相互交互,最终获取参与者和$\mathcal{A}$的输出.而在理想系统(idea system)中,这些参与者不直接运行密码协议,而是将所有计算请求发送给一个理想的可信方(idea all-trust party)$\mathcal{T}$,$\mathcal{T}$根据参与者的输入计算协议输出,将输出结果返回给相应的参与者.如果对于每个$\mathcal{A}$和每个$\mathcal{E}$都存在一个协议模拟器$\mathcal{S}$,像实际系统中的攻击者$\mathcal{A}$一样在理想系统中控制一些参与者,使得环境$\mathcal{E}$无法区分是运行在实际系统中和$\mathcal{A}$交互,还是运行在理想系统中和模拟器$\mathcal{S}$交互.我们就可以说模拟器$\mathcal{S}$安全地模拟了密码协议的功能.

远程证明是远程依赖方能够验证证明方平台的真实运行状态,其安全基础和基本前提假设是 TPM uncorrupted,否则,证明方的一切证明都可以在 corrupted TPM 的基础上伪造.TPM 是否被 corrupted 研究是 TPM 身份证明的考虑范围,在本文的远程证明安全性证明模型中,$\mathcal{M}$始终固定为 honest.

远程证明除了攻击可信第三方外,还存在两类典型攻击者:一类是 TPM 真实未被篡改而主机已被损坏,或者主机安全但用户不诚实.在不可信运行状态下伪造远程证明数据,使得远程依赖方相信该可信计算平台处于某安全状态.另一类攻击是 TPM 和 Host 都是真实可信,但却被敌手控制的验证方,或者服务器管理员是恶意操作者,从证明数据中获取可信平台的安全配置,侵犯用户平台隐私.

4.2 安全性证明

在理想系统中,模拟器$\mathcal{S}$代表被破坏的参与者(corrupted party)与可信方$\mathcal{T}$进行交互,模拟实际系统环境$\mathcal{E}$中$\mathcal{A}$的行为,$\mathcal{S}$可以以黑盒的方式访问实际系统中的$\mathcal{A}$.对于被$\mathcal{S}$控制的参与者,模拟器将通知$\mathcal{E}$它们已经被破坏,用~表示已被破坏的参与者,例如,$\mathcal{H}$表示诚实的主机系统,$\tilde{\mathcal{H}}$表示已被敌手控制的主机系统.下面详细描述在理想系统中$\mathcal{S}$模拟协议执行的各步操作.

模拟 Setup

- $\{\widetilde{\mathcal{CA}}\}$情况:如果$\mathcal{CA}$已被敌手控制,那么,$\mathcal{S}$将从$\mathcal{A}$获取证书发布权威的公钥信息$parm_{\mathcal{CA}}=(n,g_0,g,h,S,Z,R_0,R_1,R_2)$.这样,用户平台无法知道运行的组件是否可信,获取的组件及其组件属性证书有可能被$\mathcal{A}$所伪造,而其他参与者也是无法知道这一情况的.$\mathcal{A}$可以为任何恶意程序颁发组件属性证书,严重危害整个系统的安全.
- $\{\mathcal{CA}\}$情况:如果$\mathcal{CA}$是honest party(诚实的参与者),那么,$\mathcal{S}$将模拟运行密钥产生协议,$\mathcal{S}$选择$\alpha,\beta,\gamma\in_R[1,p'q'],y\in_R QR_n$,然后生成$\mathcal{CA}$和$\mathcal{VC}$的公钥:

$$h=y^{\beta} \bmod n, g=h^{\alpha} \bmod n, g_0=g^{\gamma} \bmod n.$$

然后,按照证明协议的Setup过程计算出其他公钥参数S,Z,R_0,R_1,R_2,$\mathcal{S}$将$\mathcal{CA}$公钥$parm_{\mathcal{CA}}=(n,g_0,g,h,S,Z,R_0,R_1,R_2)$和$\mathcal{VC}$公钥$parm_{\mathcal{VC}}=(g_0,g,h,n,y)$发布给$\mathcal{A}$和其他参与者.软件厂商生产的组件经过$\mathcal{CA}$安全评估后,$\mathcal{CA}$为该类型的组件颁发组件属性证书.组件属性证书将随着软硬件的发布一起分发、配置到用户平台.

模拟 Attest

在实际系统中,如果 TPM 已被敌手攻占,那么基于 TPM 信任根的一切平台状态保证都将受到威胁,对于这种情况,任何类型的远程证明的安全性都将无法讨论.因此,模拟器在 Attest 过程中重点关注主机$\mathcal{H}$是 corrupted party 和$\mathcal{M}$是 honest party 的情形,而$\mathcal{H}$和$\mathcal{M}$都是 honest party,则$\mathcal{U}$完全遵循 CPBA 协议进行证明,不会伪造远程证明;而$\mathcal{M}$是 corrupted party,$\mathcal{H}$是 honest party,在实际系统中,我们很难保证 TPM 已经被破坏、而主机确实完全可信的.下面将模拟敌手$\mathcal{A}$存在、honest $\mathcal{M}$参与的 Attest 协议执行过程.

- $\{\tilde{\mathcal{H}},\mathcal{M}\}$情况:未损坏的$\mathcal{M}$收到来自corrupted $\mathcal{H}$的组件属性签名请求,$\mathcal{M}$请求$\mathcal{T}$进行计算,$\mathcal{M}$输入接收到来自$\mathcal{A}$的关于随机数N_v和$CPA\left\{\bigwedge_{i=1}^{\bar{n}}\hat{c}_i \Rightarrow p\right\}$,$\mathcal{S}$将计算组件属性签名,进行如下处理:

1. $\mathcal{S}$选择$h_i\in QR_n$,计算$g_i=h_i^{\alpha} \bmod n$,将h_i,g_i发送给$\mathcal{A}$;
2. $\mathcal{S}$根据$CPA\left\{\bigwedge_{i=1}^{\bar{n}}\hat{c}_i \Rightarrow p\right\}$查询承诺oracle,如果组件$\hat{c}_i$是新查询,则$\mathcal{S}$选择$C_i\in_R QR_n$;如果$\hat{c}_i$已查询过,上次询问承诺oracle输出的结果为$\hat{C}$,$\hat{h}$,则此次查询输出$C_i=\hat{C}\cdot h_i/\hat{h}$.$\mathcal{S}$询问TPM的签名oracle输出签名$\sigma$.$\mathcal{S}$将结果$C_i,\sigma$发送给$\mathcal{A}$(corrupted $\mathcal{H}$);
3. 现在,$\mathcal{S}$开始伪造部分组件属性签名,按照下列步骤进行:
   (a) $\mathcal{S}$选择随机整数:$s_{id_i}\in_R\{0,1\}^{l_{id}+l_{\varnothing}+l_H}$,$s_{\chi_i}\in_R\{0,1\}^{l_{\chi}+l_{\varnothing}+l_H}$,$s_{w_i}\in_R\{0,1\}^{l_{\chi}+2l_{\varnothing}+l_H}$;
   (b) $\mathcal{S}$选择随机数:$c\in_R\{0,1\}^{l_H}$;
   (c) $\mathcal{S}$计算$\tilde{C}_i=C_i^{-c}g_0^{s_{id_i}}g^{s_{\chi_i}}h^{s_{w_i}} \bmod n$,发送$\tilde{C}_i$给$\mathcal{A}$;
   (d) $\mathcal{S}$接收到来自$\mathcal{A}$的$c_{\mathcal{H}}$,使用签名时选择的随机数,请求 patched oracle(programmable random oracle),该 oracle 输出和随机数概率分布上不可区分,oracle 查询输出:$c=H(c_{\mathcal{H}}\|N_t)$;
4. $\mathcal{T}$通知代表$\tilde{\mathcal{H}}$完成关于N_v和$CPA\left\{\bigwedge_{i=1}^{\bar{n}}\hat{c}_i \Rightarrow p\right\}$的部分组件属性签名,如果$\mathcal{T}$告知$\mathcal{S}$:TPM正在准备签名和请求属性证明,则$\mathcal{S}$将推迟应答直至$\mathcal{T}$将部分属性签名$c,N_t,s_{id_i},s_{\chi_i},s_{w_i}$发送给$\mathcal{A}$.

模拟 Verify 和 Check

组件属性证明签名的验证将发生在下面两种情形:一种是$\mathcal{S}$(作为理想系统的$\mathcal{SP}$)接收到$\mathcal{T}$的通知,honest $\mathcal{H}$请求关于N_v和$CPA\left\{\bigwedge_{i=1}^{\bar{n}}\hat{c}_i \Rightarrow p\right\}$的组件属性证明验证,这种情况便是$\{\mathcal{H}\}$;另一种是$\mathcal{S}$(作为实际系统中的$\mathcal{SP}$)收到来自实际系统的敌手的组件属性证明,这种情况是$\{\tilde{\mathcal{H}}\}$.Verify和Check过程将不考虑$\mathcal{VC}$ corrupted的情况,这种情况下,corrupted Host完全可以用被撤销安全的组件进行组件属性远程证明,然后与$\mathcal{VC}$合谋,给出该组件属性未被撤销欺骗验证方$\mathcal{SP}$.并且在corrupted $\mathcal{VC}$的情况下,平台组件隐私性将完全得不到保护.

- $\{\mathcal{H}\}$情况:主机平台是诚实的.根据平台假设主机是honest,那么TPM也是honest.这种情况下,$\mathcal{S}$(作为$\mathcal{H}$)将收到来自$\mathcal{T}$的通知:某个TPM已经完成了对N_v和$CPA\left\{\bigwedge_{i=1}^{\bar{n}}\hat{c}_i \Rightarrow p\right\}$的部分属性签名,$\mathcal{S}$将使用random oracle模拟实际系统主机的组件属性签名,具体步骤如下:

1. $\mathcal{S}$用id_i查询oracle,如果id_i是新查询,oracle选择$b_i \in_R Z_n, a_i=(b_i/id_i)^{\alpha\beta}$,输出$a_i,b_i$作为询问结果;如果$id_i$已经查询过,上次查询结果为$\hat{a},\hat{b}$,则oracle随机选择$b_i \in_R Z_n$,计算$a_i=\hat{a}\cdot(b_i/\hat{b})^{\alpha\beta}$,输出$a_i,b_i$;
2. $\mathcal{S}$随机选择$T_{i,1} \in_R \langle h\rangle, T_{i,2} \in_R \langle g_0\rangle$;
3. $\mathcal{S}$按照如下步骤伪造关于$N_v, CPA\left\{\bigwedge_{i=1}^{\bar{n}}\hat{c}_i \Rightarrow p\right\}$,$T_{i,1}$和$T_{i,2}$的组件属性签名:

   (a) $\mathcal{S}$选择随机数

$$s_{id_i} \in_R \{0,1\}^{l_{id}+l_\varnothing+l_H},$$
$$s_{\chi_i} \in_R \{0,1\}^{l_\chi+l_\varnothing+l_H},$$
$$s_{v_i} \in_R \{0,1\}^{l_v+l_\varnothing+l_H},$$
$$s_{e_i} \in_R \{0,1\}^{l'_e+l_\varnothing+l_H},$$
$$s_{w_i} \in_R \{0,1\}^{l_\chi+2l_\varnothing+l_H},$$
$$s_{r_i} \in_R \{0,1\}^{l_n+2l_\varnothing+l_H},$$
$$s_{e_ie_i} \in_R \{0,1\}^{2l_e+l_\varnothing+l_H+1},$$
$$s_{e_iw_i}, s_{e_ir_i} \in_R \{0,1\}^{l_e+l_n+2l_\varnothing+l_H+1}.$$

   (b) $\mathcal{S}$选择随机数$c \in \{0,1\}^{l_H}$

   (c) $\mathcal{S}$计算

$$\tilde{C}_i = C_i^{-c} g_0^{s_{id_i}} g^{s_{\chi_i}} h^{s_{w_i}} \bmod n,$$
$$\tilde{Z}'_i = Z_i'^{-c} T_{i,1}^{s_{e_i}+c\cdot 2^{l_e-1}} R_0^{s_{id_i}} R_1^{s_{\chi_i}} S^{s_{v_i}} h^{-s_{e_iw_i}} \bmod n.$$

   其中,$Z'_i = Z / R_2^{\hat{p}_i} \bmod n$

$$\tilde{T}_{i,2} = T_{i,2}^{-c} g^{s_{w_i}} h^{s_{e_i}+c\cdot 2^{l_e-1}} g_0^{s_{r_i}} \bmod n,$$
$$\tilde{T}'_{i,2} = T_{i,2}^{-(s_{e_i}+c\cdot 2^{l_e-1})} g^{s_{e_iw_i}} h^{s_{e_ie_i}} g_0^{s_{e_ir_i}} \bmod n.$$

   (d) $\mathcal{S}$选择随机数N_t,请求patched oracle得到

$$c = H(H(parm_C \| (C_i \| Z'_i \| T_{i,1} \| T_{i,2})_{i=1,\ldots,\bar{n}} \| (\tilde{C}_i \| \tilde{Z}'_i \| \tilde{T}_{i,2} \| \tilde{T}'_{i,2})_{i=1,\ldots,\bar{n}} \| N_v) \| N_t).$$

4. $\mathcal{S}$发送$\sigma_{CPBA}=(\sigma, N_t, K, c, (C_i, T_{i,1}, T_{i,2}, s_{id_i}, s_{\chi_i}, s_{v_i}, s_{e_i}, s_{w_i}, s_{r_i}, s_{e_iw_i}, s_{e_ie_i}, s_{e_ir_i})_{i=1,\ldots,\bar{n}})$作为组件证明签名给$\mathcal{A}$.

- $\{\mathcal{H},\mathcal{VC}\}$情况下,如果$\mathcal{SP}$也是 honest,那么$\{\mathcal{H},\mathcal{SP},\mathcal{VC}\}$都 uncorrupted,协议按照正确的方式执行,协议操作将与$\mathcal{S}$毫无关系,即没有任何协议操作会触发$\mathcal{T}$调用$\mathcal{S}$.
- 如果是$\{\mathcal{H},\widetilde{\mathcal{SP}},\mathcal{VC}\}$的情形,敌手控制$\mathcal{SP}$,其攻击目标是通过与$\mathcal{H}$交互获取$\mathcal{H}$的平台配置,实施针对组

件隐私性攻击.$\mathcal{S}$(作为 $\widetilde{\mathcal{SP}}$)从$\mathcal{T}$获得组件属性签名σ_{CPBA},从$\mathcal{A}$得到组件id_i,χ_i,由于$\mathcal{CA}$将所有组件属性公开发布,$\mathcal{S}$可以从环境$\mathcal{E}$中获取组件id_i的相关属性信息 $(id_i,\chi_i,\hat{p}_i,A_i,e_i,v_i)$.

如果 $Z_i' = A_i^{e_i} R_0^{id_i} R_1^{\chi_i} S^{v_i} \bmod n$,$\mathcal{S}$从 $CPA\left\{\bigwedge_{i=1}^{\bar{n}} \hat{c}_i \Rightarrow p\right\}$ 查找到包含组件 $\hat{c}_i$ 的请求,且 $C_i = g_0^{id_i} g^{\chi_i} h_i \bmod n$,则$\mathcal{S}$停止输出“Failure 1”;若 $C_i \neq g_0^{id_i} g^{\chi_i} h_i \bmod n$,则$\mathcal{S}$忽略这些值.如果$\mathcal{S}$从 $CPA\left\{\bigwedge_{i=1}^{\bar{n}} \hat{c}_i \Rightarrow p\right\}$ 没有查找到包含组件 $\hat{c}_i$ 的请求,则$\mathcal{S}$停止输出“Failure 2”.

如果 $Z_i' \neq A_i^{e_i} R_0^{id_i} R_1^{\chi_i} S^{v_i} \bmod n$,$\mathcal{S}$将忽略这些值.

◆ $\{\tilde{\mathcal{H}}\}$ 情况:$\mathcal{S}$(作为$\mathcal{SP},\mathcal{VC}$)获得一个新的来自$\mathcal{A}$的关于$N_v, CPA\left\{\bigwedge_{i=1}^{\bar{n}} \hat{c}_i \Rightarrow p\right\}$ 的组件属性签名,$\mathcal{S}$首先检查TPM AIK证书、TPM签名σ和组件属性知识签名σ_{CPBA}的有效性,$\mathcal{S}$按照公式(6)进行验证,如果验证失败,则忽略该组件的属性签名;如果验证成功,则$\mathcal{S}$查找与σ_{CPBA}相关联的TPM平台,检查C_i是否出现过TPM承诺和签名过程.

- 如果承诺C_i使用过,则$\mathcal{S}$进行如下处理:

* $\mathcal{S}$使用过C_i作为某个$\mathcal{M}$模拟过Attest过程(参照前面$\{\tilde{\mathcal{H}},\mathcal{M}\}$的情况),该承诺是$\mathcal{T}$请求$\mathcal{S}$(作为 $\tilde{\mathcal{H}}$)关于$N_v, CPA\left\{\bigwedge_{i=1}^{\bar{n}} \hat{c}_i \Rightarrow p\right\}$ 的组件属性签名,$\mathcal{M}$输出C_i,g_i,h_i.

$\mathcal{S}$计算$A_i=T_{i,1}/h_i$,以 $A_i,\hat{p}_i$ 为索引从$\mathcal{E}$处查询得到属性证书 $(id_i,\chi_i,\hat{p}_i,A_i,e_i,v_i)$,如果$(b_i/id_i)^{\alpha\beta}$与a_i不相等,则说明主机能够用另一组件(id^*,χ^*)伪造TPM的承诺,$\mathcal{S}$停止然后输出“Failure 3”.

如果$a_i=(b_i/id_i)^{\alpha\beta}$,但 $g_0^{id_i} g^{\chi_i} h_i$ 与TPM承诺的C_i不相等,则说明在$\mathcal{CA}$未被损坏的情况下,主机能够伪造组件CL属性签名,$\mathcal{S}$停止然后输出“Failure 4”.

* 否则,$\mathcal{S}$查找以前$\mathcal{M},\mathcal{H}$执行的数据,赋予以前的承诺值给C_i,$\mathcal{S}$代表 $\tilde{\mathcal{H}}$ 与$\mathcal{T}$进行交互,初始化执行关于$N_v, CPA\left\{\bigwedge_{i=1}^{\bar{n}} \hat{c}_i \Rightarrow \hat{p}\right\}$ 的组件属性签名.

- C_i未使用过,对于$\mathcal{S}$是新的,则说明攻击者$\mathcal{A}$选择了一个corrupted $\mathcal{M}$进行组件证明,远程证明的前提假设是 TPM 是 honest,这种情况不在远程证明的讨论范围之内.

如果模拟器按照$\mathcal{S}$执行的操作进行工作,则在 DDH 假设和 Strong RSA Assumption 下,$\mathcal{S}$不会停止输出“Failure$X$”,并且,环境和敌手无法区分协议是运行于实际系统还是理想系统.

- Failure 1:出现这种情况的原因是敌手$\mathcal{A}$能够从TPM承诺C_i中得到组件id_i.因为承诺C_i是关于$id_i,\chi_i,\ w_i$的离散对数表示的知识,$C_i = g_0^{id_i + x_g \chi_i + x_h w_i} \bmod n$,我们可以通过回卷(rewinding)敌手和random oracle提取出 $\log_{g_0} C_i$,所以我们可以把能够产生“Failure 1”的敌手归约为能够破解离散对数困难问题的敌手.
- Failure 2:这种情况将发生在主机$\mathcal{H},\mathcal{CA}$都是 honest、但是对于组件属性 $\hat{p}$,$\mathcal{A}$能够找到一个不存在的组件或者安全属性并不满足 $\hat{p}$ 的组件,伪造该组件满足属性 $\hat{p}$ 的组件 CL 签名.
- Failure 3:TPM公正、客观地描述了平台组件配置状态并承诺C_i,但是corrupted $\mathcal{H}$能够伪造出TPM承诺$C_i=commit(id^*,\chi^*,w^*),id^*\neq id_i$,且证书权威机构为组件$id^*$颁发过满足属性 $\hat{p}_i$ 的组件证书,因此,该伪造承诺能够通过$\mathcal{SP},\mathcal{VC}$的验证.如果$id^*$也满足安全属性 $\hat{p}_i$,则主机仅仅欺骗组件配置;如果id^*不满足安全属性,则主机不仅欺骗组件配置,还欺骗组件安全属性.
- Failure 4:这种情况与Failure 2 相同,对于组件 $(id_i,\chi_i,\hat{p}_i,A_i,e_i,v_i)$ 满足安全属性 $\hat{p}_i$,$\mathcal{A}$能伪造组件 $(id_i,\chi^*,\hat{p}_i)$ 的CL签名(例如(A^*,e^*,v^*))并能够成功地实施证明.$\mathcal{A}$伪造的签名满足 $T_{i,1} = A^* h_i^* = A_i h_i$, $C_i = g_0^{id_i} g^{\chi_i} h_i = g_0^{id_i} g^{\chi^*} h_i^*$,由于 $Z = A_i^{e_i} R_0^{id_i} R_1^{\chi_i} R_2^{\hat{p}_i} S^{v_i} = A^{*e^*} R_0^{id_i} R_1^{\chi^*} R_2^{\hat{p}_i} S^{v^*}$,伪造的远程证明满足验证公式

(6),使得$\mathcal{V}$无法检测出这种CL签名伪造.

下面首先讨论环境和敌手无法区分协议是运行实际系统还是理想系统.除了 Attest 过程外,模拟器都是代表单个参与者执行所有的操作,并且$\mathcal{S}$采用的是patch random oracle,这种random oracle输出均匀分布的随机数.当模拟器$\mathcal{S}$模拟honest TPM时,输出C_i, $\tilde{C}_i$, s_{id_i}, s_{χ_i}, s_{w_i},模拟honest主机时,$\mathcal{S}$随机选择$T_{i,1},T_{i,2},s_{id_i},s_{\chi_i},s_{v_i},s_{e_i},s_{w_i}$, $s_{r_i},s_{e_iw_i},s_{e_ie_i},s_{e_ir_i}$.下面将讨论在组件属性远程证明Attest过程中,敌手和环境无法区分这些值是由模拟器选择,还是实际系统中Attest协议指定.

所有的s_X输出的区分:如果$l_\varnothing$足够大,则很容易看出实际系统的s_X输出和$\mathcal{S}$模拟输出统计上非常接近,两者的统计距离最大为$2^{-(l_\varnothing-1)}$.

然后讨论C_i的区分:模拟器$\mathcal{S}$选择的C_i与实际系统一样是随机的,$C_i=g_0^{id_i}g^{\chi_i}h_i\in\langle g_0\rangle$,对于给定的TPM和随机数$w_i$, $\log_{g_0}C_i$是和实际系统相同的.在DDH假设下,没有敌手能够区分位于Z_n^*子群$\langle g_0\rangle$上的这两种分布.

讨论$\tilde{C}_i$的区分:实际协议执行中选择的$(r_{id_i},r_{\chi_i},r_{w_i})$和模拟器选择的$(s_{id_i},s_{\chi_i},s_{w_i})$在概率分布统计上相近,模拟器输出$\tilde{C}_i=C_i^{-c}g_0^{s_{id_i}}g^{s_{\chi_i}}h^{s_{w_i}} \bmod n$,实际系统中输出的$\tilde{C}_i=g_0^{r_{id_i}}g^{r_{\chi_i}}h^{r_{w_i}} \bmod n$,由于$C_i\in\langle g_0\rangle$且在实际系统和理想系统中不可区分,则$\tilde{C}_i$在概率分布上不可区分.

讨论$T_{i,1},T_{i,2}$的区分:模拟器$\mathcal{S}$随机选择的$T_{i,1},T_{i,2}$各自来自群$\langle h\rangle$和$\langle g_0\rangle$.实际系统中计算$T_{i,1}=A_ih_i \bmod n$, $T_{i,2}=g_ih^{e_i}g_0^{r_i} \bmod n$,对于$T_{i,1}$而言,$h_i\in\langle h\rangle$,由Setup时$\mathcal{CA}$公钥参数选择和组件属性证书$Z\equiv A_i^{e_i}R_0^{id_i}R_1^{\chi_i}R_2^{\hat{p}_i}S^{v_i} \bmod n$可以知道$A_i\in\langle h\rangle$,因而实际系统中$T_{i,1}$的选择和群$\langle h\rangle$上随机元素概率上不可区分;同样地,$T_{i,2}$的选择和群$\langle g_0\rangle$上的随机元素概率上不可区分.因而实际系统和理想系统中$T_{i,1},T_{i,2}$不可区分.

由上面的分析可知,敌手和环境无法通过协议输出区分组件属性证明协议是运行于实际系统还是理想系统.

对于模拟器$\mathcal{S}$执行上述操作输出"Failure X"的情况,我们将构造算法$\mathcal{S}'$,在敌手无法控制可信权威机构$\mathcal{CA}$和$\mathcal{V}$,但可以控制$\mathcal{SP}$和$\mathcal{H}$的前提下,算法$\mathcal{S}'$仍然可以解决任意flexible RSA instance$(n,z\in QR_n)$.算法将按照如下方式与$\mathcal{A}$,$\mathcal{E}$和$\mathcal{F}$进行交互:

模拟实际系统的 Setup 过程

对于给定的flexible RSA instance,$\mathcal{S}'$将产生$\mathcal{CA}$的公钥.$\mathcal{S}'$随机选择$r_{g_0},r_g,r_h\in_R[n,n^2]$,计算$g_0=z^{r_{g_0}} \bmod n$, $g=z^{r_g} \bmod n$, $h=z^{r_h} \bmod n$,按照实际系统的计算方法得到g_0,g,h,R_0,R_1,R_2,S,Z,最后,$\mathcal{S}'$将产生的$\mathcal{CA}$的公钥$parm_{\mathcal{CA}}=(n,g_0,g,h,S,Z,R_0,R_1,R_2)$发送给$\mathcal{A}$.

引理 3. 在 Strong RSA Assumption 下,不存在这样的 adversary:在不能控制属性证书权威机构($\mathcal{CA}$)和验证中心($\mathcal{V}$)的情形下,能够使模拟器输出"Failure 3".

模拟器执行组件属性远程证明输出"Failure 3",说明敌手能够伪造承诺$C_i=g_0^{id_i}g^{\chi_i}h^{w_i}=g_0^{id^*}g^{\chi^*}h^{w^*} \bmod n$,带入$\mathcal{CA}$公钥参数$g_0,g,h$得到$z^{r_{g_0}id_i+r_g\chi_i+r_hw_i}=z^{r_{g_0}id^*+r_g\chi^*+r_hw^*}$,因此$z^f=1 \bmod n$,其中,

$$f=r_{g_0}(id_i-id^*)+r_g(\chi_i-\chi^*)+r_h(w_i-w^*).$$

由于$(id_i,\chi_i,w_i)\neq(id^*,\chi^*,w^*)$,$f\neq0$,$ord(z)|f$,任意选择素数$v(0<v<\phi(n))$,$gcd(v,f)=1$,则存在$d\in Z_n^*$,$vd\equiv1(\bmod f)$,$d$可以由扩展的欧几里德算法求得.令$u=z^d \bmod n$,$z=z^{vd}=(z^d)^v=u^v \bmod n$.对于任意灵活的RSA实例$n,z\in QR_n$,$\mathcal{S}'$计算求得$u,v$,满足$z=u^v \bmod n$.也即是,使得模拟器$\mathcal{S}$输出"Failure 3"的敌手,对于任意灵活的RSA实例$n,z\in QR_n$,我们可以构造算法$\mathcal{S}'$,破解Strong RSA Assumption.故引理 3 得证.

模拟实际系统的 Attest 过程:$\mathcal{S}'$模拟实际系统的 Attest 过程与$\mathcal{S}$的模拟相同.

引理 4. 在 Strong RSA Assumption 下,不存在这样的 adversary:在不能控制属性证书权威机构($\mathcal{CA}$)和验证中心($\mathcal{V}$)的情形下,能够使模拟器输出"Failure 2"或"Failure 4".下面通过$\mathcal{S}'$模拟 Verify,Check 过程证明引理 4.

模拟实际系统的 Verify,Check 过程

如果$\mathcal{S}'$收到来自敌手$\mathcal{A}$的有效的组件属性签名,$\mathcal{S}'$将验证TPM的组件承诺和$\mathcal{CA}$为该组件颁发的属性证书id_i,χ_i是否一致.一旦$\mathcal{S}'$没有发现$\mathcal{CA}$为组件元组id_i,χ_i颁发过组件属性证书(此种情况下模拟器$\mathcal{S}$模拟Verify,Check操作将输出“Failure 2”或“Failure 4”),$\mathcal{S}'$将回卷(rewind)敌手$\mathcal{A}$到产生组件属性签名时的状态,$\mathcal{A}$调用random oracle输出组件属性签名,oracle提供给敌手不同的c,这将获得第 2 个关于id_i,χ_i的组件属性签名.$\mathcal{S}'$从这两个签名能够提取出相同的$(C_i,\tilde{C}_i,T_{i,1},T_{i,2},Z_i',\tilde{T}_{i,2},\tilde{T}_{i,2}')$,但是这两个签名有不同的$c$和$(s_{id_i},s_{\chi_i},s_{v_i},s_{e_i},s_{w_i},s_{r_i},s_{e_iw_i},s_{e_ie_i},s_{e_ir_i})$这两个签名分别表示为

$$((C_i,\tilde{C}_i,T_{i,1},T_{i,2},Z_i',\tilde{T}_{i,2},\tilde{T}_{i,2}'),c,(s_{id_i},s_{\chi_i},s_{v_i},s_{e_i},s_{w_i},s_{r_i},s_{e_iw_i},s_{e_ie_i},s_{e_ir_i})),$$

$$((C_i,\tilde{C}_i,T_{i,1},T_{i,2},Z_i',\tilde{T}_{i,2},\tilde{T}_{i,2}'),\overline{c},(\overline{s}_{id_i},\overline{s}_{\chi_i},\overline{s}_{v_i},\overline{s}_{e_i},\overline{s}_{w_i},\overline{s}_{r_i},\overline{s}_{e_iw_i},\overline{s}_{e_ie_i},\overline{s}_{e_ir_i})).$$

因为$c\neq\overline{c}$,可以令$\Delta c=c-\overline{c}$,$\Delta s_X=s_X-\overline{s}_X$,从实际系统的 Verify 过程的第 2(b)步的验证式可以得到:

$$C_i^{\Delta c}=g_0^{\Delta s_{id_i}}g^{\Delta s_{\chi_i}}h^{\Delta s_{w_i}} \bmod n \tag{7}$$

$Z_i'^{\Delta c}=T_{i,1}^{\Delta s_{e_i}+\Delta c\cdot 2^{l_e-1}}R_0^{\Delta s_{id_i}}R_1^{\Delta s_{\chi_i}}S^{\Delta s_{v_i}}h^{-\Delta s_{e_iw_i}} \bmod n$,其中,

$$Z_i'=Z/R_2^{\hat{p}_i} \bmod n \tag{8}$$

$$T_{i,2}^{\Delta c}=g^{\Delta s_{w_i}}h^{\Delta s_{e_i}+c\cdot 2^{l_e-1}}g_0^{\Delta s_{r_i}} \bmod n \tag{9}$$

$$T_{i,2}^{\Delta s_{e_i}+\Delta c\cdot 2^{l_e-1}}=g^{\Delta s_{e_iw_i}}h^{\Delta s_{e_ie_i}}g_0^{\Delta s_{e_ir_i}} \bmod n \tag{10}$$

从引理 2 和等式(9)可知,Δc 可以同时整除Δs_{w_i}和Δs_{e_i},否则可以以至少 1/2 的概率计算出 flexible RSA Instance(n,z).令$\hat{s}_{w_i}=\Delta s_{w_i}/\Delta c$,$\hat{s}_{e_i}=\Delta s_{e_i}/\Delta c$.对等式(10)两边进行$\Delta c$次方,使用等式(9)我们可以得到:

$$(g^{\Delta s_{w_i}}h^{\Delta s_{e_i}+c\cdot 2^{l_e-1}}g_0^{\Delta s_{r_i}})^{\Delta s_{e_i}+c\cdot 2^{l_e-1}}=g^{\Delta c\Delta s_{e_iw_i}}h^{\Delta c\Delta s_{e_ie_i}}g_0^{\Delta c\Delta s_{e_ir_i}} \bmod n \tag{11}$$

由引理 2 的证明很容易得到$\Delta s_{w_i}(\Delta s_{e_i}+\Delta c2^{l_e-1})=\Delta c\Delta s_{e_iw_i}$,否则能够解决给定的 flexible RSA Instance.等式变形得到:

$$\Delta s_{e_iw_i}=\hat{s}_{w_i}(\Delta s_{e_i}+\Delta c2^{l_e-1})=\hat{s}_{w_i}\Delta c(\hat{s}_{e_i}+2^{l_e-1}) \tag{12}$$

将等式(12)带入等式(8)变形后得到:

$$Z_i'^{\Delta c}=\left(\frac{T_{i,1}}{h^{\hat{s}_{w_i}}}\right)^{\Delta c(\hat{s}_{e_i}+2^{l_e-1})}R_0^{\Delta s_{id_i}}R_1^{\Delta s_{\chi_i}}S^{\Delta s_{v_i}} \bmod n \tag{13}$$

由引理 2 可知,Δc 可以整除$\Delta s_{id_i},\Delta s_{\chi_i},\Delta s_{v_i}$,令$\hat{s}_{id_i}=\Delta s_{id_i}/\Delta c$,$\hat{s}_{\chi_i}=\Delta s_{\chi_i}/\Delta c$,$\hat{s}_{v_i}=\Delta s_{v_i}/\Delta c$,我们重写等式(13)得到:

$$Z=\left(\frac{T_{i,1}}{h^{\hat{s}_{w_i}}}\right)^{(\hat{s}_{e_i}+2^{l_e-1})}R_0^{\hat{s}_{id_i}}R_1^{\hat{s}_{\chi_i}}R_2^{\hat{p}_i}S^{\hat{s}_{v_i}} \bmod n \tag{14}$$

由于Δc 可以整除$\Delta s_{id_i},\Delta s_{\chi_i},\Delta s_{v_i}$,变形等式(7)可以得到:

$$C_i=g_0^{\hat{s}_{id_i}}g^{\hat{s}_{\chi_i}}h^{\hat{s}_{w_i}} \bmod n,$$

其中,

$$(id_i,\chi_i)\neq(\hat{s}_{id_i},\hat{s}_{\chi_i}) \tag{15}$$

因此对于组件(id_i,χ_i,w_i)的属性证明,若模拟器$\mathcal{S}$输出“Failure 2”或“Failure 4”,则算法$\mathcal{S}$能够伪造消息$(\hat{s}_{id_i},\hat{s}_{\chi_i},\hat{p}_i)$的CL签名,伪造的签名为$\left(\left(\frac{T_{i,1}}{h^{\hat{s}_{w_i}}}\right),\hat{s}_{e_i}+2^{l_e-1},\hat{s}_{v_i}\right)$,这与CL签名在Strong RSA Assumption下是安全的相矛盾,故引理 4 得证.

综上所述,在理想系统中存在模拟器$\mathcal{S}$模拟实际系统证明协议操作,使得敌手和环境无法区分协议是运行于实际系统还是理想系统,并且引理 3 和引理 4 成立,因此我们可以得到如下结论(定理 5):

定理 5. 在 DDH 假设和 Strong RSA Assumption 下,组件属性证明方案在 Random Oracle 模型下可被证明是安全的.

4.3 安全性分析

4.3.1 平台隐私性分析

组件属性证明协议能够保护平台运行组件的隐私.远程证明中隐私性的保护是一个非常矛盾的问题:一方面,验证方需要了解平台的真实运行状态,就不得不知道证明方平台配置的硬件、运行的软件等平台隐私信息;另一方面,可信计算平台又不愿意暴露平台隐私,或仅仅暴露少量平台隐私.远程证明一般都采用引入可信第三方进行验证来实现平台隐私保护,PBA 方案采用颁发属性证书和远程证明承诺的方式来保护隐私.

本文的平台隐私性保护结合了这两种方法的优点,用组件配置承诺来防止恶意主机平台欺骗,用可信第三方验证来保护组件隐私.由引理 2 可知,恶意的验证方无法获取组件的配置信息(id_i,χ_i),也就是CPBA协议保证了不存在能够在理想系统中输出“Failure 1”和“Failure 3”的敌手,从而保护了平台隐私性.CPBA协议根据证明请求:如果证明请求中要求证明具体组件,那么对于这些组件,CPBA协议满足组件配置隐私性;如果有可选组件供用户选择证明,那么CPBA协议满足组件类型隐私性.

4.3.2 属性可撤销性分析

随着黑客攻击方法的提高、攻击能力的增加、病毒破坏方式和攻击手段的增强,某一配置的组件不再满足原有的安全属性,此时就需要对原有颁发的组件属性证书进行撤销.在Chen的PBA方案中,属性的撤销检查是通过复杂的零知识证明来实现的,计算量大且非常复杂.本文采用可信第三方验证属性的撤销,完全可以套用 X.509 证书撤销机制.如果(id_i,χ_i)组件的安全属性已经从 $\hat{p}_i$ 降低到 $\hat{p}_i'$,则只需发布撤销信息 $(id_i,\chi_i,\hat{p}_i)$ 给验证中心 $\mathcal{VC}$,$\mathcal{CA}$再重新为该组件颁发组件属性证书 $(id_i,\chi_i,\hat{p}_i')$,无须以复杂的零知识证明来判断属性是否已经被撤销.

4.3.3 抗伪装攻击分析

CPBA 协议为了防止伪装的远程证明,在 TPM 签名中包含了 TPM 与远程依赖方的协商密钥 K.如果平台 A 实施伪装攻击,则在 TPM 匿名身份证明的情况下,平台 A 伪装为平台 B 固然能够通过远程依赖方的验证,但是由于协商密钥 K 只有真正进行证明的平台 B 的 TPM 才知道,平台 A 无法知道远程证明建立的协商会话密钥 K.本文只采用了简单的 Diffie-Hellman 密钥协商抵抗伪装攻击,这方面还可以根据实际的安全需求进行更深入的探讨.

5 系统实现

根据基于组件属性的远程证明原理,我们在配置有符合 TPM 1.2 规范的 National Semiconductor TPM 安全芯片的安全 PC、系统为 Federal Core 5 Linux 平台上实现了组件证明原型系统.该原型系统在可信引导程序、可信操作系统度量的基础上实现了对 Linux 系统的组件的度量和证明.

在原型系统TPM要证明一个组件是否可信,TPM可以度量其加载到内存中可执行文件镜像,可以度量组件的代码段(text section),度量组件所要调用的内核模块,度量其运行期间需要加载到内存中的系统动态链接库等等.这些从各个不同侧面证明组件是否可信的分类称为度量类别(measurement class,简称 MC).而每个度量类别下描述组件运行状态的具体度量项,定义为组件度量变量(measurement variable,简称 MV).目前,我们实现原型系统中包括的度量类别有:

- 可执行文件镜像:程序运行时加载到进程空间的可执行文件的内存镜像.
- 组件动态链接库:随组件一起发布的动态链接库.当组件程序运行需要某个链接库时,在其动态加载时对其进行度量.组件运行时,无须对那些未加载的组件动态链接库度量,只需对可执行程序调用的动态链接库进行动态度量.
- 系统动态链接库:对组件运行所需要的系统动态链接库进行度量.该度量类别作为组件运行的系统依赖,由于各个系统软硬件配置差异非常大,组件属性证书并不包含系统动态链接库的评估.但是,组

件属性证明的验证服务可以将这一类别作为额外的系统环境来进行验证.

- 系统内核模块:组件所依赖的内核模块也要进行度量,这一度量是由可信操作系统度量子系统完成.

除了我们实现的这些度量类别外,还可以从组件配置数据、组件代码段数据、组件运行的关键数据结构等度量类别来描述组件的运行状态.原型系统中,证书发布权威机构根据软件厂商组件的可执行文件镜像和组件动态链接库这两个度量类别的数据进行属性评估,然后颁发属性证书.TPM 对上述组件类别进行度量,然后遵循组件属性证明协议进行远程证明.图 2 为基于组件属性证明服务的系统结构和浏览器 Firefox 组件度量和证明结果数据.

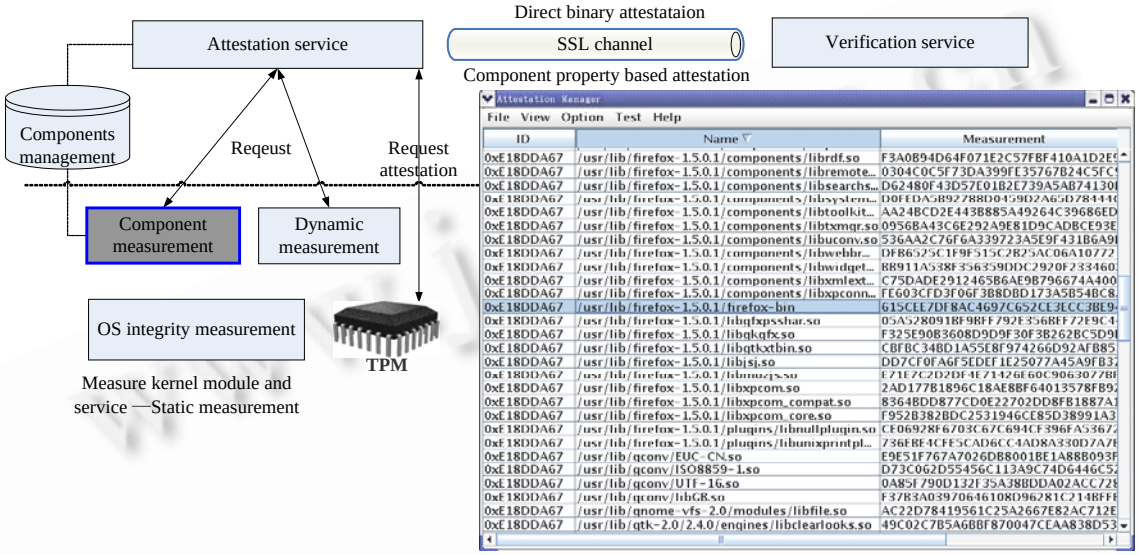
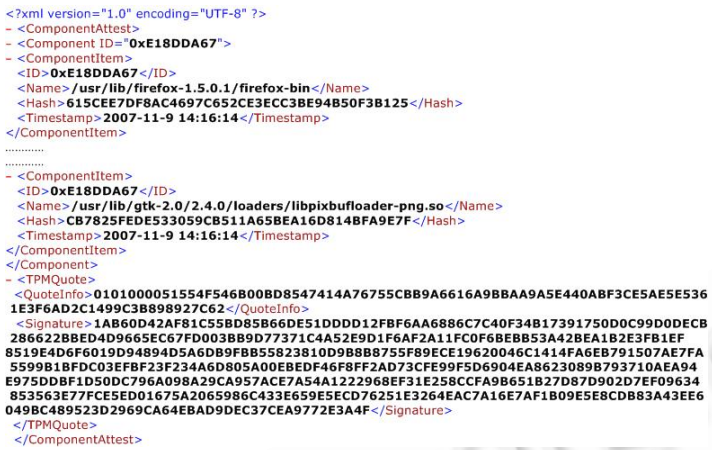


Fig.2 System implementation of component property attestation

图 2 组件属性证明系统实现

图 2 给出了基于组件属性证明的主要体系结构,证明服务负责组件属性证明数据的管理.它驱使组件度量代理对系统的组件进行分类度量,然后,TPM 对度量数据进行签名.除了基本的组件属性度量和证明外,我们还设计并实现了动态度量代理,能够实时地对系统中运行的程序进行度量并进行证明,这增强了基于组件属性证明的实时性.Firefox 组件证明包括了浏览器 Firefox 可执行文件 firefox-bin、Firefox 组件动态链接库/usr/lib/firefox-1.5.0.1/components/libmork.so,/usr/lib/firefox-1.5.0.1/components/libeditor.so,…;所依赖的系统动态链接库/usr/lib/libssl3.so,/lib/libpthread-2.4.so,/lib/libdl-2.4.so,…;还有其他所依赖的动态链接库,如 GTK 库、GNOME 库等,这些度量变量的完整性确保了浏览器组件 Firefox 的安全.最后证明服务创建证明数据,通过 SSL 安全通道向服务器方证明平台运行组件满足某种安全属性,图 3 是证明服务提供的 Firefox 证明数据模板实例.

组件属性证明仅仅关注于系统的部分组件的安全属性证明,对系统性能影响不大.当一个组件运行时,TPM 对组件的各个类别中的每个度量变量进行完整性度量.表 1 是组件度量和证明过程中对系统效率影响最大的组件完整性收集的性能测试结果.以浏览器 Firefox 为例,证明系统选择了 142 个度量变量对 firefox 进行度量,度量过程耗费时间为 3.950s,firefox 可执行代码度量结果(TPM 执行 SHA-1 算法的度量值)为“615CEE7DF8AC4697C652CE3ECC3BE94B50F3B125”.将表中的测试结果进行统计,组件证明系统平均耗费 0.029s 检查一个组件度量变量,这丝毫不影响系统效率.



```
<?xml version="1.0" encoding="UTF-8" ?>
- <ComponentAttest>
- <Component ID="0xE18DDA67">
- <ComponentItem>
  <ID>0xE18DDA67</ID>
  <Name>/usr/lib/firefox-1.5.0.1/firefox-bin</Name>
  <Hash>615CEE7DF8AC4697C652CE3ECC3BE94B50F3B125</Hash>
  <Timestamp>2007-11-9 14:16:14</Timestamp>
</ComponentItem>
.....
.....
- <ComponentItem>
  <ID>0xE18DDA67</ID>
  <Name>/usr/lib/gtk-2.0/2.4.0/loaders/libpixbufloader-png.so</Name>
  <Hash>CB7825FEDE533059CB511A65BEA16D814BFA9E7F</Hash>
  <Timestamp>2007-11-9 14:16:14</Timestamp>
</ComponentItem>
</Component>
- <TPMQuote>
  <QuoteInfo>0101000051554F546B00BD8547414A76755CBB9A6616A9BBAA9A5E440ABF3CE5AE5E536
  1E3F6AD2C1499C3B898927C62</QuoteInfo>
  <Signature>1AB60D42AF81C55BD85B66DE51DDDD12FBF6AA6886C7C40F34B17391750D0C99D0DECB
  286622BBED4D9665EC67FD003BB9D77371C4A52E9D1F6AF2A11FC0F6BEBB53A42BEA1B2E3FB1EF
  8519E4D6F6019D94894D5A6DB9FBB55823810D9B8B8755F89ECE19620046C1414FA6EB791507AE7FA
  5599B1BFDC03EFBF23F234A6D805A00EBEDF46F8FF2AD73CFE99F5D6904EA8623089B793710AEA94
  E975DDBF1D50DC796A098A29CA957ACE7A54A1222968EF31E258CCFA9B651B27D87D902D7EF09634
  853563E77FCE5ED01675A2065986C433E659E5ECD76251E3264EAC7A16E7AF1B09E5E8CDB83A43EE6
  049BC489523D2969CA64EBAD9DEC37CEA9772E3A4F</Signature>
  </TPMQuote>
  </ComponentAttest>
```


Fig.3 Data template instance of component property attestation

图 3 组件属性证明数据模板实例

Table 1 Performance analysis of component attestation

表 1 组件证明性能分析表

Component name	Measurement variable	Measurement time (s)	Measurement value of executive image
Text editor: kedit	15	0.479	F3503233927BE1A3A4F1210A10711CEF454CD3C1
Graphic viewer: kview	21	0.642	C9C8C1B2C4B1CEBBECDD1B5A14ACA6F07103597B
PDF reader: kpdf	58	1.444	F4E180D058515BC44A14E4D410DBF490A023FED5
Communicator: gaim	130	4.116	CE259289120D872F4B1DFF5D9331B1BFE48E8F9D
Browser: firefox	142	3.950	615CEE7DF8AC4697C652CE3ECC3BE94B50F3B125

6 进一步的讨论

CPBA方案和PBA方案在TPM签名和证明过程中引入交互知识签名证明,两者对于每个知识签名的计算量基本相等,假定每个知识签名计算量为τ,那么PBA方案交互知识签名证明的计算量为$N_{PCR}\tau$(N_{PCR}为平台PCR的个数,一般为 16 个),CPBA方案的知识签名计算量为$N_{component}\tau$($N_{component}$为要证明组件的个数).只有当要证明的组件个数超过平台的PCR个数时,CPBA方案的零知识计算量才会大于PBA方案.对于像网上银行这样的实例,最多只需证明不超过 8 个组件就能满足安全要求.对于数量更大的组件证明,一种方法是可以采用直接的二进制证明,另一种方法是采用临时聚合组件证明的方法减少计算量和通信量.验证属性是否撤销,PBA方案也采用知识签名的方法,计算量为$N_{PCR}\tau$,而CPBA采用组件属性证书查询的方法,这方面效率将大幅度提高.

从组件证明请求中不难看出,证明组件数量增大,如证明可信计算平台的全部组件,组件属性证明将会生成大量的证明数据,增大计算量,影响证明协议的通信效率.对于大数量组件,证明将进一步进行改进,可采用颁发聚合组件临时属性证书的方法.用户平台将证明请求和组件度量结果发送给组件属性权威,由它为用户平台颁发有效周期很短的临时聚合组件属性证书,用户平台使用临时属性证书向服务提供者进行证明.

令 Hash 函数的连接操作(TPM 的扩展操作)定义为 $H(a||b)=a\cdot b$.用户平台关于组件证明属性请求 $CPA\left\{\left(\underset{i=1}{\overset{k}{\wedge}}\hat{c}_i\right)\wedge\left(\underset{i=k+1}{\overset{\bar{n}}{\wedge}}\left(\underset{j\in\mathcal{J}_i}{\vee}\hat{c}_{i,j}\right)\right)\Rightarrow\hat{p}\right\}$ 的组件度量值为$\chi_i(i=1,...,\bar{n})$.

1. TPM 聚合(aggregating)组件度量值 $\chi=\prod_{i=1}^{\bar{n}}\chi_i=\chi_1\cdot\chi_2\cdot...\cdot\chi_{\bar{n}}$,用户平台将组件证明请求和度量值,TPM 签名发送给$\mathcal{CA}$;
2. $\mathcal{CA}$验证组件度量和 TPM 签名,根据组件属性请求评估用户平台是否满足属性 $\hat{p}$,如果满足则颁发临时属性证书 $(id,\chi,\hat{p},A,e,v)$, $Z\equiv A^e R_0^{id} R_1^{\chi} R_2^{\hat{p}} S^v \bmod n$,其中,$id$ 为临时分配的组件 ID;
3. 用户平台按照 Attest 过程为聚合组件χ生成承诺 C,然后零知识证明临时聚合组件属性;

4. $\mathcal{SP}$验证关于临时组件证书$(id,\chi,\hat{p},A,e,v)$的组件属性证明.

上述过程将大数量的组件转化为一个聚合组件进行组件属性证明,证书颁发权威为聚合组件颁发临时属性证书,这样,用户平台只需证明聚合组件,大大减少了证明数据量,提高了证明效率.与原有的组件属性证明相比,大数量组件的证明每次都得请求证书颁发权威颁发聚合组件属性证书,证书颁发权威可能成为通信瓶颈.

7 总 结

本文提出了一种组件级的细粒度属性远程证明方法.该方法兼容现有的 TCG 完整性管理框架,能够精确地描述平台的安全属性,特别是对于组件数量不大的属性远程证明,其优势非常明显.该方案摆脱了纷繁复杂的平台配置状态,以组件为基础描述平台配置的安全属性,采用组件配置表示和安全属性的逻辑范式(与、或)来表示平台安全属性.组件属性证明方案所颁发的组件属性证书易于撤销和验证.CPBA 方案能够根据证明请求保护被证明组件的类型、版本等组件隐私信息,具有抵抗远程证明伪装攻击的能力.下一步研究将集中在改进组件属性证明的证明效率上,特别是对大数量组件的属性证明将进一步研究,提出更为实用的远程证明方法.

References:

[1] TCG Group. TPM main part 1, design principles specification. Version 1.2. 2003. https://www.trustedcomputinggroup.org/home

[2] TCG Group. TCG architecture overview specification. 2004. https://www.trustedcomputinggroup.org/home

[3] TCG Group. TCG software stack (TSS) specification. Version 1.10. 2003. https://www.trustedcomputinggroup.org

[4] Sailer R, Zhang XL, Jaeger T, Doorn LV. Design and implementation of a TCG-based integrity measurement architecture. In: Proc. of the 13th Usenix Security Symp. San Diego: Usenix Press, 2004. 16−16.

[5] Smith S. Trusted Computing Platforms—Design and Applications. New York: Springer-Verlag, 2005. 193−194.

[6] Seshadri A, Perrig A, Doorn LV, Khosla P. SWATT: Software-Based attestation for embedded devices. In: Proc. of the IEEE Security & Privacy Conf. Oakland: IEEE Press, 2004. 272−282.

[7] Garfinkel T, Rosenblum M, Boneh D. Flexible OS support and applications for trusted computing. In: Proc. of the 9th Workshop on Hot Topics in, Operating Systems (HotOS IX). Hawaii: Usenix Association, 2003. 25−25.

[8] Haldar V, Chandra D, Franz M. Semantic remote attestation: A virtual machine directed approach to trusted computing. In: Proc. of the USENIX Virtual Machine Research and Technology Symp. San Jose: Usenix Press, 2004. 29−41.

[9] Poritz J, Schunter M, Herreweghen EV, Waidner M. Property attestation—Scalable and privacy-friendly security assessment of peer computers. IBM Research Report, RZ 3548, 2004.

[10] Sadeghi A, Stüble C. Property-Based attestation for computing platforms: Caring about properties, not mechanisms. In: Proc. of the New Security Paradigms Workshop. Nova Scotia: ACM Press, 2004. 67−77.

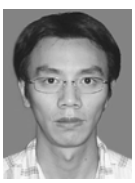
[11] Chen LQ, Landfermann R, Löhr H, Stüble C. A protocol for property-based attestation. In: Proc. of the 1st ACM Workshop on Scalable Trusted Computing. Nova Scotia: ACM Press, 2006. 7−16.

[12] Camenisch J, Lysyanskaya A. A signature scheme with efficient protocols. In: Proc. of the Security in Communication Networks, the 3rd Int'l Conf., SCN 2002. Amalfi: Springer-Verlag, 2002. 268−289.

[13] TCG Group. TCG trusted network connect TNC IF-TNCCS specification. Version 1.1. 2007. https://www.trustedcomputinggroup.org/home

[14] Bellare M, Rogaway P. Random oracles are practical: A paradigm for designing efficient protocols. In: Proc. of the 1st CCS. New York: ACM Press, 1993. 62−73.

[15] Canetti R. Security and composition of multi-party cryptographic protocols. Journal of Cryptology, 2000,13(1):143−202.



秦宇(1979－),男,重庆人,博士生,主要研究领域为信息与网络安全.

冯登国(1965－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为信息与网络安全.