doi:10.11959/j.issn.1000-436x.2017180

基于双层非平衡散列树的云平台远程验证方案

荣星[1,2]，沈昌祥[2]，江荣[3]，赵勇[2]
（1. 解放军信息工程大学三院，河南 郑州 450004；
2. 北京工业大学计算机学院，北京 100124；3. 国防科技大学六院，湖南 长沙 410073）

摘 要：为验证云服务的可信性，提出一种改进的基于非平衡散列树的云平台远程验证方案。通过引入层级构建双层非平衡散列树，将原先的单一树扩展为主树和子树，二者分别对应云服务平台中的虚拟机和虚拟机中的运行组件，证明时仅需要提供待度量组件和认证路径。分析表明，该方案进一步提高了云平台的证明效率，并且具有较好的隐私保护能力和可伸缩性，能够很好地用于云服务的可信性证明。
关键词：远程证明；云计算；非平衡散列树；虚拟机
中图分类号：TP309 **文献标识码**：A

Remote attestation scheme for cloud platform based on double-layer unbalanced hash tree

RONG Xing[1,2], SHEN Chang-xiang[2], JIANG Rong[3], ZHAO Yong[2]
(1. The 3rd Academy, PLA Information Engineering University, Zhengzhou 450004, China;
2. College of Computer Science, Beijing University of Technology, Beijing 100124, China;
3. The 6th Academy, National University of Defense Technology, Changsha 410073, China)

Abstract: In order to validate the service of cloud, an improved remote attestation scheme based on unbalanced hash tree of cloud platform was proposed. Double-layer unbalanced hash tree was built by introducing layer, original single tree was expanded to main tree and sub tree, which corresponded to virtual machine in cloud platform and the running components in virtual machine. Attestation needs no more than measurement component and authentication path. The analysis shows that this scheme can increase the attestation efficiency of cloud platform, and is good at protecting privacy and expandability, which is suitable for validating cloud platform service.
Key words: remote attestation, cloud computing, unbalanced hash tree, virtual machine

1 引言

2006 年，亚马逊推出的 AWS（Amazon Web service）服务正式拉开了全球云计算产业的大幕，经过 10 多年的发展，云计算颠覆了固有的传统架构，已经成为产业界发展的核心驱动力。服务外包是云计算服务的应用模式，在该模式下用户将数据全权委托给云端管理，优势在于：对于用户来说，能在数据维护和资源购置上节约大量的人力物力资源；对于云服务商来说，可以方便地统一管理高度集中的数据资源。然而，外包模式将用户数据的所有权、管理权及使用权进行了分离，使用户失去了对数据的直接控制，导致云中数据的机密性、完整性和服务可用性受到了威胁。恶意的云服务提供商（CSP, cloud service provider）可以直接获取数据而不会被用户发现，非恶意的 CSP 也可能由于内部人员的失职、黑客攻击或系统故障导致安全机制失效，系统和用户数据被非法获取。

收稿日期：2016-08-24；**修回日期**：2017-06-09
基金项目：国家高技术研究发展计划（“863”计划）基金资助项目（No.2015AA016002）；国家重点研发计划基金资助项目（No.2016YFB0800804，No.2016YFB0800303）
Foundation Items: The National High Technology Research and Development Program (863 Program) of China (No.2015AA016002), The National Key Research and Development Program (No.2016YFB0800804, No.2016YFB0800303)

当前，云计算在应用中面临的主要问题之一就是云服务的可信性证明问题，由 CSP 来证明自身服务的可信性缺乏说服力，如何针对云服务的外包化特点建立可信的监控机制保障用户数据安全，对外可以提供云服务的可信性证明，同时保证不会泄露云平台的隐私，是目前亟需解决的问题。Santos 等[1]首次提出了使用可信计算技术构建可信云计算平台（TCCP, trusted cloud computing platform），用户可通过第三方 TC 来验证云中虚拟机的安全机制。国内的 T-YUN[2]引入了一个专用的云平台作为可信第三方（TTP，trusted third party），TTP 通过验证云平台服务器内部的配置信息和度量信息，对云服务提供商的可信性进行验证。上述方法均通过平台完整性证明来验证 CSP 的可信性，证明采用的是二进制证明（binary attestation）方法[3]，如 IMA（integrity measurement architecture）体系架构[4]，效率和安全性方面均存在问题。

文献[5]指出 IMA 体系结构在度量时需要遍历系统所有组件，验证度量列表（ML，measurement list）的完整性需要将全部表项进行扩展操作然后计算聚合值，在验证大规模的计算环境时存在隐私泄露和效率低下的问题，从而提出了基于 Merkle 散列树（MH-Tree，Merkle hash tree）的远程验证机制 RAMT，使用 Merkle 散列树存储度量值，验证时只需要提供组件度量值及其相关的认证路径即可，从而解决了平台信息泄露的问题。文献[6]指出由于 Merkle 散列树是满二叉树，存在中间节点添加数量多和平均查询路径长的问题，使用 Merkle 散列树进行远程证明严重影响证明效率，提出了基于非平衡散列树（UH-Tree, unbalanced hash tree）的证明机制 RAUT，相比 RAMT 方案减少了中间节点的数量和认证路径长度，节省了存储空间和验证的运算时间，但是对于组件数量巨大和查询验证频繁的云计算环境，通过全局 ID 号查找虚拟机中组件的方法在证明效率上还有待提高。

针对云服务的可信性证明问题，本文提出一种基于双层非平衡散列树（DUH-Tree, double-layer unbalanced hash tree）的远程证明方法（RADUT，remote attestation based on double-layer unbalanced hash tree），在单层 UH-Tree 之上引入了一棵主树，采用分层的结构来细化证明的粒度，分层方案通过子树来区分不同虚拟机，减少了对外证明时的查询开销，在确保隐私安全的前提下有效降低了云服务平台完整性验证的延迟。

2 平台完整性证明

2.1 二进制证明

可信计算组织[3]（TCG, trusted computing group）提出的远程验证机制在系统启动过程中实施完整性状态度量，如图 1 所示，通过逐次计算硬件及软件栈的完整性散列值，保存在外部操作不能更改的平台配置寄存器（PCR, platform configuration register）中，这些值能够真实反映系统从 BIOS 到操作系统 OS 层的完整性状态。远程计算平台通过挑战质询获得本地的完整性度量值，通过完整性状态验证本地平台的安全属性，从而建立相应的信任关系。TCG 远程验证机制度量的完整性状态有限，只能验证平台从 BIOS 到 OS 层的完整性，无法实施应用层软件栈的完整性验证。

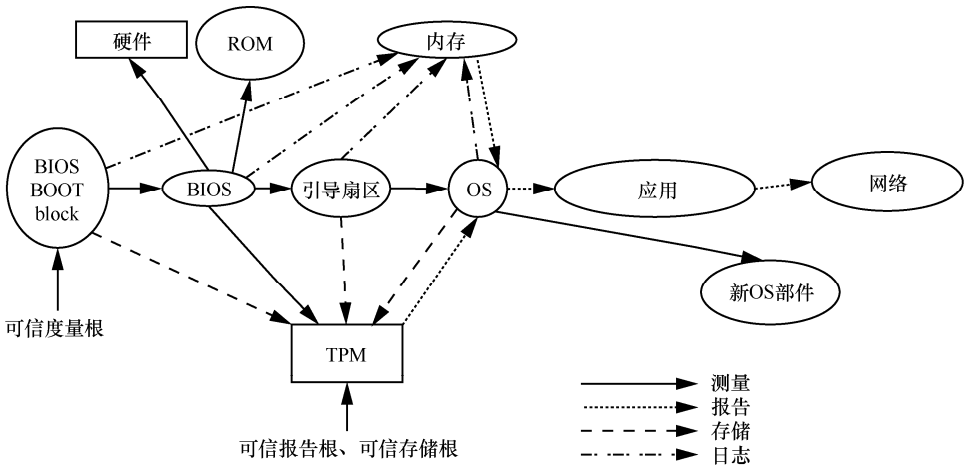


图 1 TCG 可信度量过程

IMA 体系架构对 TCG 体系做了进一步扩展，将其完整性验证的范围扩大到应用层软件栈，在完成 TCG 可信启动过程以后，由待验证平台的 OS 在内核空间维护一张度量列表，记录包括 OS 启动管理器、内核等系统内核空间软件栈和用户态应用层软件的完整性散列值，并且将值扩展到可信平台模块（TPM）特定的 PCR 中，从而在 PCR 中构建出全部 ML 表项的有序聚合，保证了 ML 列表自身的完整性，提高了验证的粒度和全面性。IMA 在验证时需要掌握全部组件的散列值和 PCR 的扩展顺序，按顺序重构 PCR 的值，存在两方面的不足：1) 只适用于步骤固定的系统启动过程的可信证明，由于应用程序运行或启动的随机性较强，且相互之间没有严格的依赖关系，不适于应用程序间的证明；2) 平台需要发送的 ML 中含有全部组件的散列值，完整性度量值中包含有系统平台全部的信息，如硬件、OS、应用软件、函数库等摘要值，完全将平台的配置信息暴露给验证方，存在隐私泄露问题。

在云服务中用户多关心的是某个组件的完整性，如客户机 OS 镜像有没有使用修改过的操作系统版本，此时为了验证云服务的完整性，只需要验证虚拟机特定部分组件的可信性，无需提供整个虚拟机环境证明信息，从而避免不必要的平台信息泄露，防止恶意用户利用泄露的平台信息对虚拟机进行攻击。

2.2　基于散列树的证明方法

散列树具有利用小可信空间实现保护存放在不可信存储器中的海量动态数据对象的优良特性，只要保证根散列值的安全存储，就能保护数据对象的完整性[7]。

RAMT 机制并未按链式结构来扩展得到 PCR 值，而是将各可信实体完整性度量值的散列值构建成一棵 Merkle 树，叶子节点对应组件的散列值，对外证明某个组件的完整性时，只需要叶子节点和对应的认证路径序列即可，从而有效保护了平台的隐私信息。RAMT 在内核中维护的是散列值，叶子节点是待验证平台上各种组件完整性度量值的散列值，内部节点是对应的子节点连接后生成的散列值。Merkle 散列树使用的是满二叉树，在节点数不接近 2 的指数倍时，需要添加大量的中间节点，造成了时间和空间上额外的开销。另外，Merkle 散列树可扩展性差，建立后无法继续扩展，当平台中有新的组件运行或关闭时都需要重新构造整棵二叉树。在动态性很强的云计算环境下，存在大量的虚拟机和用户应用程序，程序的注销和建立频繁，按照 Merkle 树证明方法中每个用户程序与二叉树的一个叶子节点对应的话，构造出来的二叉树不仅形态庞大，耗费时间长，而且动态性不能满足云环境的需求。

RAUT 是基于非平衡散列树的远程验证机制，主要用来验证物理平台的组件完整性，利用非平衡散列树存储组件散列值，减少了空间开销，同时继承了散列树证明的隐私保护性。然而，云平台中运行的是虚拟机实例，虚拟机按照模板方式批量部署，虚拟机和程序组件的启动又具有很强的随机性，使云平台中的组件数量随着虚拟机的增加成倍地增长，加之 RAUT 采用的非平衡散列树是动态无序排列的，对外证明时遍历速度变慢，影响查询验证的效率，因此，用 RAUT 验证云平台服务可信时会存在一定的局限性。

3　非平衡散列树

非平衡散列树[8]是一棵能够在线动态构建的二叉树，满足最左子树原则。构建方法如下：将叶子节点从左至右顺序排列，从左到右每次选取最多的节点依次构建出多棵完全二叉树，再通过从右到左每次连接最小的 2 棵独立树的方法合并出最终的树。非平衡散列树满足唯一性，验证者能够构建出与证明者完全相同的树来进行验证，且具有平衡二叉树的对数特性。

假定 T 是棵非平衡散列树，根是 $root$，T 的左、右子树分别记作 T_{l}、 T_{r}，左、右子树的根分别记作 $root_{\mathrm{l}}$、 $root_{\mathrm{r}}$，T_{l} 一定符合满二叉树结构。树的深度用函数 $depth(\ )$表示。

节点添加规则如下所示。

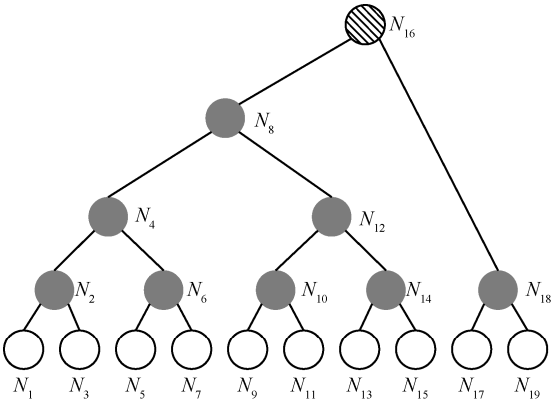
规则 1　当 T 是满二叉树结构，子树中有 $2^{k-1}-1$ 个节点（$k \in N^+$），新增一个度量组件时，增加一个中间节点 $N_{2^{k-1}}$ 并作为新树 T'的根，原子树 T 作为 $root'$的左孩子，增加一个叶子节点 $N_{2^{k-1}+1}$，作为 $root'$ 的右孩子，并更新根节点 $N_{2^{k-1}}$ 的值，$N_{2^{k-1}} = hash(N_{2^{k-2}} \| N_{2^{k-1}+1})$。

规则 2　当 T 不是满二叉树结构且 T_r 是满二叉树结构，$depth(T_{\mathrm{l}}) > depth(T_r)$，以 T_r 为操作对象，按照规则 1 进行节点添加操作，T 的新右子树为 $T_{r'}$，根为 $root_{r'}$，更新根节点 $root$ 的值，$root = hash(root_{\mathrm{l}} \| root_{r'})$。

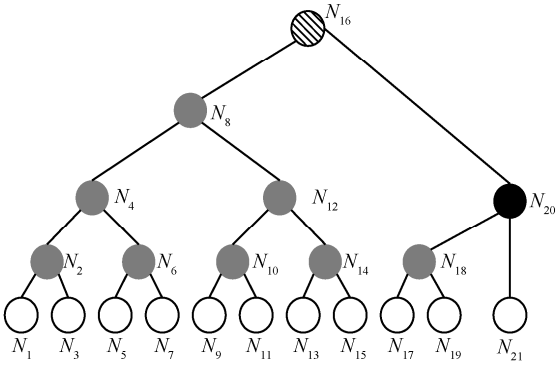
规则 3　当 T 不是满二叉树结构且 T_r 不是满二叉树结构，$depth(T_l) = depth(T_r)$，当前编号最大的叶子节点为 N_t，添加中间节点 N_{t+1} 和叶子节点 N_{t+2}，移动节点 N_t 并作为 N_{t+1} 的左孩子，N_{t+2} 作为 N_{t+1} 的右孩子，将根为 N_{t+1} 的子树连接到节点 N_t 原来的位置，更新中间节点的值，$N_{t+1} = hash(N_t \| N_{t+2})$，迭代往上更新 T_r 中相关节点和根 *root* 的值。

规则 4　当 T 不是满二叉树结构且 T_r 不是满二叉树结构，$depth(T_l) > depth(T_r)$，以 T_r 的右子树为对象，按照规则 2 和规则 3 对 T_r 的右子树进行节点添加操作，生成新的右子树 $T_{r'}$，根为 $root_{r'}$，更新根节点 *root* 的值，$root = hash(root_l \| root_{r'})$。

以叶子节点数从 10 增加到 11 为例，更新过程描述如下。当叶子节点数为 10 时，构建出的非平衡散列树如图 2(a)所示，白色的是叶子节点，灰色的是中间节点，花纹填充的是根节点，新增节点时，如图 2(b)所示，先引入一个新的中间节点，用黑色节点表示，然后将右子树与新增的叶子节点分别作为中间节点新的左右孩子，再连接到根节点下，最后完成中间节点和根节点值的更新。



(a) 叶子节点数为10时的UH-Tree



(b) 叶子节点数为11时的UH-Tree

图 2　非平衡散列树

4　基于 DUH-Tree 的云平台远程证明机制

4.1　DUH-Tree 的更新

定义 1　双层非平衡散列树 DUH-Tree。采用分层结构，分为主树和子树，主树、子树均是非平衡散列树结构，主树对应云平台，子树对应虚拟机，每棵子树的根对应主树的一个叶子节点，非平衡散列树是一棵任意节点的左子树都是满二叉树的二叉散列树。

图 3 所示的是一棵包含 N 个虚拟机的双层非平衡散列树，主树有 N 个叶子节点，N 个叶子节点分别对应 N 棵子树，每棵子树有若干个叶子节点，子树叶子节点数目等于虚拟机中度量组件的数目，子树的叶子节点值是虚拟机中组件可信度量值的散列值，中间节点是其 2 个孩子节点值连接后的散列值，主树的根节点信息用于描述整个云平台的完整性。DUH-Tree 树的更新规则如下。

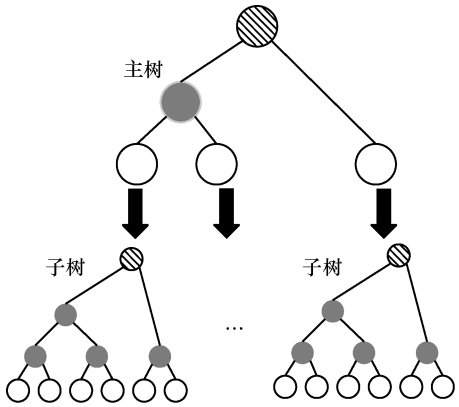


图 3　双层非平衡散列树

1) 当云平台中有虚拟机注销时，更新采取懒惰算法，直接删除虚拟机对应的子树，子树对应的主树叶子节点值不变，更新主树中该叶子节点的状态为空闲。虚拟机中组件停止服务时，对子树叶子节点也采取类似的方法。

2) 当云平台中有新的虚拟机实例启动时，主树中没有空闲的叶子节点，新增一个叶子节点，对应增加一棵子树，根据虚拟机组件的度量值完成子树的构建，完成子树对应的主树叶子节点及其到主树根节点路径上所有节点值的更新，然后更新主树根节点在相应 PCR 中的值；当主树中存在空闲的叶子节点时，直接在该节点下增加子树，构造方法同上。

3) 当虚拟机中的度量组件值发生变化时，更新

对应子树叶子节点及其到子树根节点路径上所有节点的值，并更新子树根节点到主树根节点路径上所有节点的值。

4) 当虚拟机中有新的程序组件运行时，查找对应的子树是否存在空闲叶子节点，若存在，直接更新对应该空闲节点及其到主树根节点路径上的所有节点值；若不存在，新增一个叶子节点，更新新加入的叶子节点及其到主树根节点的所有节点值。

4.2 RADUT 体系结构

RADUT 体系结构由 CSP、可信第三方 TTP 和验证者 verifier 三方组成，验证过程中需要隐私 CA 参与平台身份认证密钥（AIK, attestation identity key）证书的验证，如图 4 所示。CSP 的云平台有物理 TPM（trusted platform module）为整个云平台提供硬件上的可信支撑，可信度量服务初始化生成一棵双层非平衡散列树，根据服务的运行情况动态更新树。TTP 在收到 verifier 的云服务可信验证请求后，向 CSP 发出挑战，CSP 通过可信证明服务将对应虚拟机的子树叶子节点、位于认证路径上的节点及 TPM 签名过的主树根节点值一起发送给 TTP，TTP 根据认证路径的节点信息重新计算主树根节点的散列值，并与签名值进行比较，从而判断出云服务的可信性，并将结果返回给 verifier。

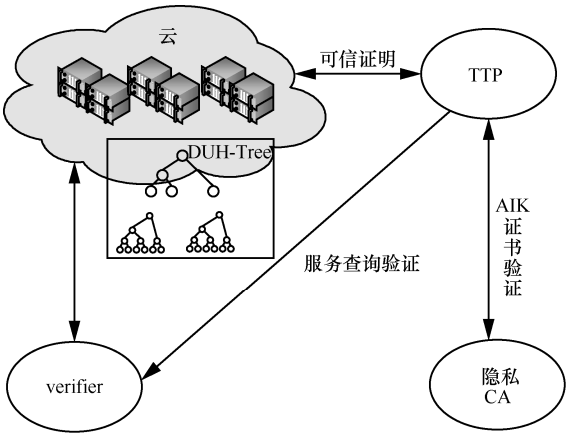


图 4　RADUT 体系组成

4.3 云服务可信证明过程

步骤 1　用户 $\rightarrow$ TTP : $\{obj\}_{\text{request}}$。用户向 TTP 发送验证云服务某个组件对象 *obj* 可信性的请求。

步骤 2　TTP $\rightarrow$ CSP : $\{nonce, obj\}_{\text{challenge}}$。TTP 向提供服务的 CSP 发出挑战，并向其发送待验证的组件对象 *obj* 和生成的随机数。

步骤 3　CSP 在云平台中查找 *obj* 所在的虚拟机，在 DUH-Tree 中找到相应的叶子节点 N_i 及其到主树根节点的认证路径 $path=<N_j \cdots N_k>$，TPM 在 PCR 中读取 DUH-Tree 的主树根节点的散列值和程序的度量列表，利用 TPM_Quote 功能对包含主树根节点的 PCR 值和随机数签名。

步骤 4　CSP $\rightarrow$ TTP : $N_i, path, N_j \cdots N_k, sig_{\text{AIK}}\{nonce, PCR_{\text{root}}\}, ML, Cert_{\text{AIK}}$。CSP 将对象节点的值、认证路径及路径上所有节点的值、主树根节点的签名值、度量列表，连同平台 AIK 证书发送给 TTP。

步骤 5　TTP 接收到消息后，向隐私 CA 查询 AIK 证书及确认数字签名的真实性，比较随机数的正确性，根据认证路径中的节点值重新计算得到主树的根节点值，并与 PCR 中的值比较，验证组件度量散列值的完整性，再利用度量散列值和度量列表判定云服务的可信性。

步骤 6　TTP 向用户返回验证的结果。

5 性能分析

本文采用的是基于二叉树的可信完整性验证方法，摒弃了基于度量列表的链式结构，在隐私性保护和验证效率方面均优于传统的二进制证明方法[5]，与文献[5,6]不同的是采用的二叉树结构不同，本文采用的是双层非平衡散列树 DUH-Tree，文献[5]采用的平衡二叉树 MH-Tree，文献[6]采用的是单层非平衡散列树 UH-Tree，下面，从隐私保护及构建、验证更新开销方面进行比较分析。

5.1 隐私保护

本文采用的是基于二叉树的可信完整性验证方法，云服务平台度量时只需要将主树的根节点值扩展到 TPM 的 PCR 中，CSP 对外只提供度量对象的散列值、认证路径序列及 PCR 中的根节点值，验证时由可信第三方对 CSP 的服务可信性进行验证，TTP 给出的结果具有一定的权威性。在验证过程中，CSP 只提供了度量对象的散列值和认证路径，方案继承了树型证明方法的隐私性，避免了整个云平台信息的泄露，有效保护了云服务平台的隐私。

5.2 构建空间和时间开销

假设云服务平台中的虚拟机以相同的镜像模板部署，各虚拟机的运行状态相同，平台中共有 k_1 个虚拟机，每个虚拟机均有 k_2 个组件，共有 k_1k_2 个组件，

存储空间由构造含有 k_1k_2 个叶子节点的树的大小决定。设 MH-Tree、UH-Tree 和 DUH-Tree 所需要的总节点数分别为 S_{MH}、S_{UH} 和 S_{DUH}，根据满二叉树和非平衡散列树的构建规则可得

$$S_{\mathrm{MH}} = 2^{\lceil \mathrm{lb}k_1k_2 \rceil + 1} - 1 \tag{1}$$

$$S_{\mathrm{UH}} = 2k_1k_2 - 1 \tag{2}$$

$$S_{\mathrm{DUH}} = (2k_1 - 1) + k_1(2k_2 - 1) = 2k_1k_2 + k_1 - 1 \tag{3}$$

从式(1)~式(3)中可以发现，DUH-Tree 和 UH-Tree 二者的空间开销与总节点数呈线性关系，DUH-Tree 需要比 UH-Tree 多 k_1 个节点的空间。由于主树的节点数是虚拟机的数量，目前，一台物理服务器上大概可运行数十个虚拟机，系统度量组件包含系统服务、驱动程序、应用程序及配置，仅驱动程序一般就有 2 000 多个[9]，k_1 与 S_{UH} 相比基本可以忽略，S_{DUH} 接近于 S_{UH}。由于 MH-Tree 是满二叉树，在组件数目是 2^n−1 时，不需要进行叶子节点填充操作，此时，S_{MH} 与 S_{UH} 相同，最差的情况是当组件数目是 2^n+1，需要填充 2^n−1 个叶子节点，以及由此带来的中间节点，此时，DUH-Tree、UH-Tree 比 MH-Tree 相对所需的空间开销最小。

在 Intel Core2 i5 2.50 GHz、内存为 8 GB、操作系统为 Centos 7 的物理机上，用 KVM 虚拟 2 个 2 GB 内存的 Ubuntu12.04 虚拟机环境，分别作为 TTP 和客户虚拟机，用 Python2.7 作为编程实现语言，散列函数使用 SHA-256 算法，进行 3 种不同树的构建时间开销实验。叶子节点值是客户虚拟机中度量组件值的映射，对需要填充的节点值使用随机数散列，构建树的时间开销与总节点数和树的生成算法有关，在保持主树叶子节点为 33 的情况下，选择不同数量的子树叶子节点，构建 3 种不同的二叉树，如图 5 所示，MH-Tree 的时间开销最大，DUH-Tree 比 UH-Tree 的花费时间略大，这主要是由于双层结构的 DUH-Tree 的节点比 UH-Tree 多，但随着节点数的增加开销时间几乎相同。

5.3 查询验证和更新开销

虽然 MH-Tree、UH-Tree 的最长认证路径均为 $\lceil \mathrm{lb}k_1k_2 \rceil$，它们的左子树查询验证、更新开销相同，但是 UH-Tree 是非平衡的，其右子树的节点数和认证路径长度都比满二叉树 MH-Tree 的右子树小，显然，UH-Tree 的查询验证更新开销要优于 MH-Tree。下面，对 DUH-Tree、UH-Tree 的查询验证、更新开销进行分析比较。

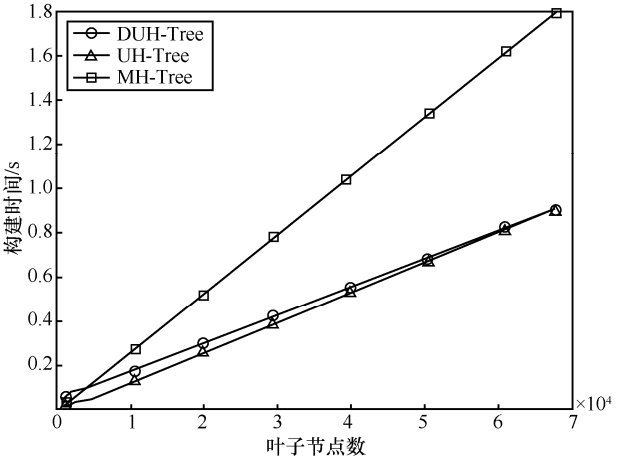


图 5 3 种散列树的构建时间对比

查询验证和更新开销与认证路径长度直接相关，图 6 是在主树节点数为 65、子树节点数为 257 时最后一棵子树的第一个叶子节点的认证路径，如图 6 所示，DUH-Tree 中子树的根节点值 e61642574990142 34b9faa935fef01a7993209805b55fdf53b6b8d4ae20fe440 正是主树的叶子节点值，认证路径由子树认证路径和主树认证路径联合构成，此时，子树的认证路径长度为 9，主树的认证路径长度为 1。

```
--- audit proof
m:    0
n:    257
leaf: b784b9136024ba914f0a081bf0c351231352aacc2956be20d8606d7e7c3c68e3
path:
    - 7882e380fce99f27f1ab963e4ffb97349e143d1e4744c59e522eee8d20cf3de7
    - 9f0b3364346073dadbfc439c9527d36b5fec700d2577cf18053a6a8414dbeb96
    - 5db867698b0bceb2576318e582f33f17faa1ecfeaf04ac227a84a9a5205c7de6
    - 2227ecf40b65fde818cbde4f3c27f23631b9ed7fc282c10176d6a999e13bbed10
    - 0cae9d96591ce2a26462dbf5c80bd096d9e70fcd79df24a7ed7a69df047a962f
    - df4f8050735216a2ce030b1a0c3ca195c0197a4f1d2c4b907c063fb2b3a16891
    - ebce970202ebf524b6c269b4feff1734fae91f9fd6dc3cfd3cfe2b23e11ba2e9
    - a1ee467b25093a6d14aa154f3697526c2f06c6b025714d5115b62566c05672a7
    - 88d47656ff03d34c4c908e76908f5ef15f0927d29aa67cb51a0345b3f034d02a
root: e61642574990142 34b9faa935fef01a7993209805b55fdf53b6b8d4ae20fc440
--- audit proof
m:    64
n:    65
leaf: e61642574990142 34b9faa935fef01a7993209805b55fdf53b6b8d4ae20fc440
path:
    - 80c98f4072f3f1ef67bb9bf85307008fc8d3b03088276b375ca049cc5a2c4c8d
root: b736bfa25d602c667032ed55d41871c00af04fffcc43e8a68fb9e281f6ce5228
.
--------------------------------------------------------------------------
```

图 6 双层非平衡二叉树的认证路径

5.3.1 查询验证开销

叶子节点的查询验证可分为查询和验证 2 个步骤。对于查询操作，UH-Tree 查询叶子节点直接在非平衡散列树的叶子节点中进行遍历，DUH-Tree 则在双层非平衡散列树中首先确定叶子节点所属虚拟机对应的子树，对主树的叶子节点进行查找后在子树中进行遍历；验证时，UH-Tree 找出该节点的认证路径进行验证，DUH-Tree 则先找出子树的认证路径，再找出子树根节点在主树中的认证路径后进行验证。非平衡散列树的查询时间与叶子节点

的个数有关，验证时间与树的高度有关，记叶子节点个数为 N 时的查询时间复杂度为 $t(N)$，验证时间复杂度为 $O(\mathrm{lb}N)$，则当叶子节点个数为 k_1k_2 时，UH-Tree、DUH-Tree 的叶子节点查询验证时间复杂度 O_{UH}、O_{DUH} 分别为

$$O_{\mathrm{UH}} = t(k_1k_2) + O(\mathrm{lb}k_1k_2)$$
$$= t(k_1k_2) + O(\mathrm{lb}k_1) + O(\mathrm{lb}k_2) \quad (4)$$

$$O_{\mathrm{DUH}} = t(k_1) + t(k_2) + O(\mathrm{lb}k_1) + O(\mathrm{lb}k_2) \quad (5)$$

显然 $O_{\mathrm{UH}} > O_{\mathrm{DUH}}$，即 UH-Tree 的叶子节点查询验证时间复杂度比 DUH-Tree 大。下面，进行实验对比，树中所有叶子节点的总查询时间能够更好地反映出查询验证开销，随机选择 k_1 值为 40，k_2 的个数从 100 到 1 000 递增，实验结果如图 7 所示，DUH-Tree 的总查询时间明显少于 UH-Tree，查询验证效率更好。

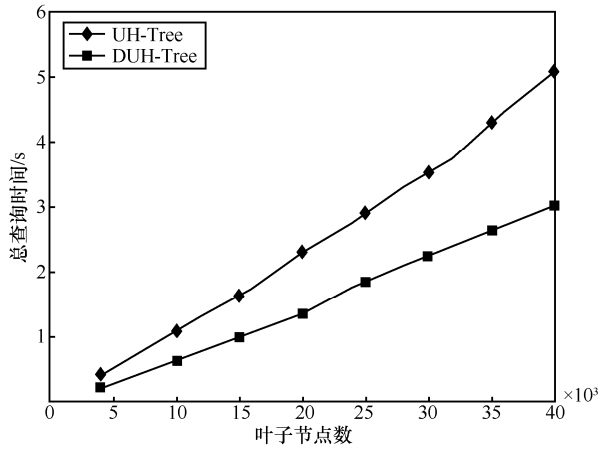


图 7　叶子节点的总查询验证时间比较

5.3.2　更新开销

由于节点的更新采用懒惰算法，删除节点只改变节点的状态值，并不改变节点本身的散列值，在删除相同个数的叶子节点时，DUH-Tree、UH-Tree 的更新开销相同；但在叶子节点发生变化时，除了更新叶子节点值外，还需要对该叶子节点到根节点上所有路径的节点值进行更新，即根据认证路径进行散列运算并对其上一层的节点进行修改，因而更新时需要查找出对应认证路径上的节点，更新的开销与认证路径的长短直接相关。当 k_1=40，k_2=1 000 时，在 UH-Tree 中随机选择数量为 M 的节点进行更新测试，在 DUH-Tree 中随机选择 5 棵子树，对每棵子树中 $\frac{M}{5}$ 个随机节点值进行更新，当 M 值为 400、600、800、1 000、2 000、3 000、5 000 时，更新实验结果如图 8 所示，DUH-Tree 的更新时间明显少于 UH-Tree，效率更好，一方面是由于 DUH-Tree 的整个查询验证开销优于 UH-Tree，另一方面是当子树中大量节点需要批量更新时，DUH-Tree 采取先更新子树再更新主树的方式，避免了对主树中节点的重复操作。

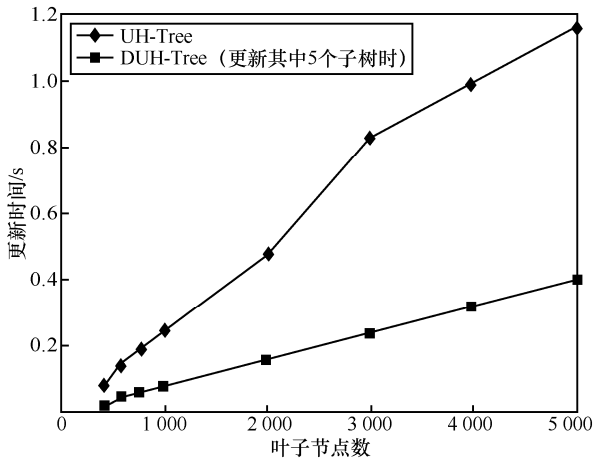


图 8　叶子节点的更新时间比较

6　结束语

RADUH 方案缩短了认证路径的长度，减少了查询验证的时间，方案的动态性还体现在主树和子树的更新过程上，采用双层非平衡散列树减少了子树节点活动对整棵树带来的影响，降低了更新开销。主树与子树分别对应云平台中的虚拟机、虚拟机中的度量组件，可根据云服务平台中虚拟机服务的动态变化而相应调整，伸缩性强，分层结构可以有效区分不同虚拟机环境，查询验证效率更高，因而比单一的非平衡二叉树更适用于云计算环境。在验证云服务可信时，平台仅向可信第三方提供待度量的组件节点及认证路径上，避免了云平台隐私的泄露，判定结果由可信第三方提供，比 CSP 单方面证明自身服务的可信更具有权威性。

参考文献：

[1] SANTOS N, GUMMADI K P, RODRIGUES R. Towards trusted cloud computing[C]//Conference on Hot Topics in Cloud Computing. 2009: 1-5.

[2] 刘川意, 方滨兴. T-YUN: 云提供商可信性审计与验证[J]. 信息网络安全, 2012, 8: 97-100.

LIU C Y, FANG B X. T-YUN: trustworthiness audit and verification of the cloud[J]. Netinfo Security, 2012, 8: 97-100.

[3] 张焕国, 罗捷, 金刚, 等. 可信计算研究进展[J]. 武汉大学学报（理学版）. 2006, 52(5): 513-518.
ZHANG H G, LUO J, JIN G, et al. Research progress of trusted computing environment[J]. Journal of Wuhan University: Natural Science Edition. 2006, 52(5): 513-518.

[4] SAILER R, ZHANG X, JAEGER T, et al. Design and implementation of a TCG-based integrity measurement architecture[C]//USENIX Security Symposium. 2004: 223-238.

[5] 徐梓耀, 贺也平, 邓灵莉. 一种保护隐私的高效远程验证机制[J]. 软件学报, 2011, 22(2): 339-352.
XU Z Y, HE Y P, DENG L L. Efficient remote attestation mechanism with privacy protection[J]. Journal of Software, 2011, 22(2): 339-352.

[6] 翁晓康, 张平, 王炜, 等. 基于非平衡散列树的平台完整性远程验证机制[J]. 计算机应用, 2014, 34(02): 433-437.
WENG X K, ZHANG P, WANG W, et al. Remote attestation mechanism for platform integrity based on unbalanced-hash tree[J]. Journal of Computer Applications, 2014, 34(2): 433-437.

[7] YAN J, ZHAO Y. Trusted attestation of behavior measurement based on Merkle hash tree[J]. Journal of Computational Information Systems, 2013, 9(9): 3443-3451.

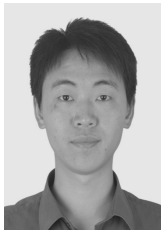
[8] 朱毅, 李清宝, 钟春丽,等. 用于细粒度完整性度量的非平衡二叉散列树模型[J]. 小型微型计算机系统, 2014,35(7):1604-1609.
ZHU Y, LI Q B, ZHONG C L, et al. Non-balanced binary hash-tree model for fine-grained integrity measurement[J].Journal of Chinese Computer Systems,2014,35(7): 1604-1609.

[9] ENGLAND P. Practical techniques for operating system attestation[C]//Trusted Computing Challenges and Applications, First International Conference on Trusted Computing and Trust in Information Technologies, Trust 2008. 2008: 1-13.

作者简介：

荣星（1986-），男，安徽合肥人，解放军信息工程大学博士生，主要研究方向为网络安全、云计算。

沈昌祥（1940-），男，浙江奉化人，中国工程院院士，北京工业大学教授、博士生导师，主要研究方向为计算机信息系统、密码学、信息安全架构、系统软件安全及网络安全。



江荣（1984-），男，福建连城人，博士，国防科技大学助理研究员，主要研究方向为大数据隐私保护和网络空间安全。

赵勇（1980-），男，山西左权人，博士，北京工业大学讲师，主要研究方向为可信计算、安全操作系统。