Journal of Computer Applications
计算机应用,2014,34(2):433 -437
ISSN 1001-9081
CODEN JYIIDU
2014-02-10
http://www.joca.cn

文章编号:1001-9081(2014)02-0433-05
doi:10.11772/j.issn.1001-9081.2014.02.0433

基于非平衡哈希树的平台完整性远程验证机制

翁晓康[1,2*],张　平[1,2],王　炜[1,2],朱　毅[1,2]
(1.信息工程大学,郑州 450001;　2.数学工程与先进计算国家重点实验室,郑州 450001)
(*通信作者电子邮箱 wxk_06@126.com)

摘　要:为提高计算平台完整性度量的远程验证效率,提出一种基于非平衡哈希树的平台远程验证机制。平台可信实体的散列值以非平衡哈希树叶子节点的结构存储,远程验证时,查找度量实体对应的叶子节点,记录该叶子节点到根节点的验证路径,然后传递根节点和验证路径给验证方,最后根据验证路径重新计算根节点来验证度量值的有效性。实验结果表明,该机制能够有效降低散列值存储的空间和时间开销,完整性度量验证的时间复杂度为$O(\mathrm{lb}\ N)$。

关键词:可信计算;完整性度量;远程验证;非平衡哈希树
中图分类号:TP309.1　　**文献标志码**:A

Remote attestation mechanism for platform integrity based on unbalanced-Hash tree

WENG Xiaokang[1,2*], ZHANG Ping[1,2], WANG Wei[1,2], ZHU Yi[1,2]
(1. *Information Engineering University, Zhengzhou Henan* 450001, *China*;
2. *State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou Henan* 450001, *China*)

Abstract: In order to improve the remote authentication efficiency for integrity measurement of computing platforms, this paper proposed a platform remote authentication mechanism based on unbalanced-Hash trees. Hash values of platform's trusted entities were stored in the structure of leaf nodes of unbalanced-Hash trees. Effectiveness of the metrics was verified through seeking corresponding leaf nodes of measured entities, recording the validation paths from leaf nodes to root nodes, passing from root nodes to the prover and finally recalculating the root nodes according to validation paths. The experimental results show that the proposed mechanism can effectively reduce time and space overhead of storing Hash values and the time complexity of integrity measurement authentication is $O(\mathrm{lb}\ N)$.

Key words: trusted computing; integrity measurement; remote attestation; unbalanced-Hash tree

0　引言

近几年,对等网络和分布式计算环境受到广泛关注,与集中式计算环境相比,分布式计算环境具有开放性、动态性及节点多样性等特点,这就使得参与计算的各节点之间的信任建立困难,相互之间地可信认证也成为较为复杂的问题。

可信计算组织(Trusted Computing Group, TCG)[1]在计算平台的硬件层引入可信平台模块(Trusted Platform Module, TPM),从可信根出发,使用信任链传递机制,可对本地平台的硬件以及软件层实施逐层的完整性度量。由于TCG提出的计算平台度量验证机制的度量验证层次较低,只验证到计算平台操作系统层的可信性,并未定义应用层可信性的验证方法,使得其应用范围较小;且在验证时,需要发送整个度量列表给验证方,容易造成其他度量值的泄露。所以,Sailer等[2]提出完整性度量体系结构(Integrity Measurement Architecture, IMA)。IMA通过在系统内核中维护一张度量列表(Measure List, ML),将表中所有度量值进行哈希运算后得到的聚合值作为验证度量列表的可信依据,从而将信任链延伸至应用层。IMA增加度量层次,扩展度量范围,但同样严重影响了度量验证效率。每一次验证,都需要将所有度量值重新进行哈希运算得到聚合值,大大影响系统的运行效率;并且IMA同样不能解决验证平台信息泄露问题。针对IMA的不足,徐梓耀等[3]提出一种保护隐私的高效远程验证机制RAMT(Remote Attestation based on Merkle hash Tree),该方法使用Merkle哈希树结构存储度量值,验证时只需要发送与度量值相关的验证路径给验证方,在一定程度上解决了平台信息泄露的问题。但由于Merkle哈希树是一棵平衡的二叉树,致使叶子节点平均查询路径长度较长,并且需要添加的中间节点数量过多,严重影响验证效率。针对Merkle哈希树中间节点过多的问题,本文提出一种基于非平衡哈希树的验证机制(Remote Attestation based on Unbalanced-Hash Tree, RAUT)。该机制使用非平衡的哈希树作为存储可信实体散列值的存储组织结构,在一定条件下,可以大大减少哈希树的中间节点,节省存储空间。在验证时,由于中间节点大幅减少,验证路径相应缩短,可以节省验证运算时间,提高远程验证效率;并且验证方得到的仅仅是与验证值相关的验证路径,并不能得到整个度

收稿日期:2013-07-04;**修回日期**:2013-10-22。　**基金项目**:国家核高基项目(2013JH00103)。
作者简介:翁晓康(1989 -),男,浙江杭州人,硕士研究生,CCF会员,主要研究方向:网络信息安全、可信计算;　张平(1969 -),女,吉林长春人,副教授,博士,主要研究方向:高性能计算、信息安全、可信计算;　王炜(1975 -),男,湖北武汉人,讲师,博士,CCF会员,主要研究方向:计算机系统结构、信息安全、可信计算;　朱毅(1986 -),男,河北石家庄人,助理工程师,硕士研究生,主要研究方向:计算机系统结构、网络信息安全、可信计算。

量值列表,保证了平台信息的安全性。

1 相关研究

1.1 TCG 规范的度量验证机制

TCG 规范给出的标准远程验证机制可分为三步进行:1)完整性状态度量;2)度量结果质询通信;3)完整性状态验证。在平台启动时,系统由核心可信度量根(Core Root of Trust for Measurement, CRTM)作为信任的起点,逐级对可信实体进行完整性度量,并将度量值存入相应的程序控制寄存器(Program Control Register, PCR)中,一直将信任传递至 OS Loader 和 OS kernel。在度量结果质询阶段,待验证平台将 PCR 的度量值以及度量日志发送给远程验证方;在完整性验证阶段,验证方按照度量日志重新计算度量值,并与平台提供的度量值进行对比,以此证明平台可信性。理论上,信任链是可以扩展到应用层的。但实际上,由于应用层的软件数目庞大,并且加载具有随机性,操作系统需要频繁对其进行度量操作,导致这个方案在实际中难以得到实现。

1.2 IMA 验证机制

由于 TCG 度量的层次过低,只能验证平台从 BIOS 到操作系统层的完整性,度量粒度过粗,无法获得应用层的安全属性。为此,Sailer 等[2]提出了基于 TCG 的完整性度量体系结构,即 IMA 体系架构。IMA 通过在操作系统的内核空间维护一张度量列表,将信任链扩展至应用层,ML 中的每一个表项对应于某个特定软件的完整性哈希值。为了保证度量列表本身的完整性,IMA 将 ML 的所有度量值的聚合值扩展存储到 TPM 芯片相应的 PCR 中。在度量结果质询时,平台不但要提供 PCR 中的度量值及度量日志,而且要提供完整的度量列表信息。尽管 IMA 通过引入度量列表扩展可信链的度量范围,但同时也降低了验证效率。每次验证 ML 有效时,都需要对 ML 的每一个表项逐个重新计算度量值得到聚合值,而且验证需要将整个度量列表传送给验证方,没能解决 TCG 存在的隐私泄露问题。

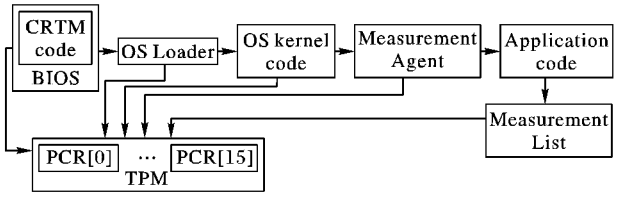


图 1 IMA 体系结构

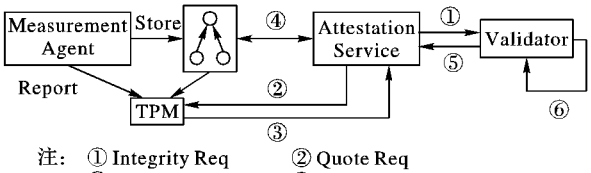
1.3 RAMT 验证机制

RAMT 在 IMA 方案的基础上进行改进,利用 Merkel 哈希树存储可信实体的完整性度量值。所有度量值都存储在 Merkel 哈希树的叶子节点之中,非叶子节点存储的是由其孩子节点度量值连接后形成的哈希散列值。为了保证 Merkel 哈希树的完整性,树的根节点被扩展存储到 TPM 芯片相应的 PCR 中。在质询阶段,平台只需要提供验证实体的散列值,验证路径和哈希树根节点散列值,验证方不能得到其他可信实体的完整性度量值,在一定程度上弥补了 IMA 方案泄露隐私的缺陷,并且提高了验证效率。IMA 验证的时间复杂度为 $O(N^2)$,而 RAMT 验证的时间复杂度为 $O(N \text{ lb } N)$。尽管 RAMT 机制在验证效率及隐私保护方面比 TCG 和 IMA 都有很大的提高,但由于使用的 Merkel 哈希树是满二叉树结构,需要添加的中间节点过多,导致节点验证路径过长,在一定程度上影响验证效率。

2 基于非平衡哈希树的平台远程验证机制

2.1 RAUT 体系结构

RAUT 的体系结构由度量代理(Measurement Agent, MA)、验证服务(Attestation Service, AS)和验证者(Validator)三个部分组成。MA 首先初始化一棵空的非平衡哈希树,随后根据度量的不断进行,增加新的度量节点,并同时更新根节点在 TPM 相应 PCR 中的散列值。AS 部件在验证者发出挑战请求后,将待验证叶子节点到根节点之间的所有节点以及 TPM 签名的根节点散列值发送给验证者。验证者在受到挑战响应后,根据验证路径节点信息重新计算根节点的散列值,并与 AS 部件发送的根节点散列值进行对比,从而判断证明方的度量值是否被修改。



注: ① Integrity Req ② Quote Req
③ Quote Res ④ Retrieve
⑤ Integrity Res ⑥ Validate Res

图 2 RAUMT 体系结构

2.2 非平衡哈希树

RAMT 使用 Merkle 哈希树结构,将度量对象哈希后的值作为叶子节点,而分支节点则是其子节点内容连接后的哈希值。任意数据对象的改变都将引起根哈希值的变化,只要保证根哈希被安全存放,就能保证所有叶子节点的完整性。虽然 Merkle 哈希树能够有效地以较小代价保护大量数据,但由于其使用的是二叉满树结构,构造树会存在着大量中间节点,需要付出额外的空间与时间开销,还会延长验证路径,很大程度上影响完整性验证效率。本文在 Merkle 哈希树基础上提出一种非平衡结构的哈希树(Unbalanced-Hash Tree, UH-Tree),以避免过多中间节点填充引起额外开销。

所谓非平衡哈希树是一棵任意节点的左子树都是满二叉树结构的二叉哈希树。树中的叶子节点对应一个可信实体度量值,每个内部节点都是该节点的两个子节点信息连接后的哈希值,根节点信息用于描述全部可信实体的完整性。

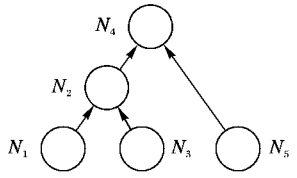


图 3 非平衡哈希树

在 UH-Tree 中,树节点记为 $N_x(1 \leqslant x \leqslant R)$,从 1 开始顺序编号,节点最大编号为 R。每增加一个叶子节点,将增加两个节点,其中一个为内部节点,记为 N_{r+1};另一个为叶子节点,记为 N_{r+2}。N_{r+2} 将作为 N_{r+1} 的右孩子连入树中,而 N_{r+1} 将作为根节点或者其他节点的右节点连入树中,N_{r+1} 的左孩子将由当前节点的最大编号 $r+2$ 决定。UH-Tree 叶节点编号均为奇数,而内部节点编号均为偶数。非平衡哈希树的叶子节点从 1 开始编号,每增加一个叶子节点,会增加两个节点,由定义可

知,编号 $r+1$ 为中间节点,编号 $r+2$ 为叶子节点。若度量的可信实体数目为 M,则树中最大节点编号为 $2M-1$;根节点的编号为 $2^{(\lceil \mathrm{lb}\, M \rceil -1)}$。

2.3 完整性度量算法

由于采用非平衡树作为存储可信实体散列值的结构,RAUMT 的完整性度量算法在度量值的增加和维护方面与 RAMT 存在较大差异。

UH-Tree 中每个节点都由三个参数组成:id、inode 和 value。如图 4 所示,叶子节点中的 id 代表可信实体的 id 号,中间节点的 id 号从 1 递增,inode 为节点在树中的编号,value 记录该节点的散列值。根节点的散列值扩展至 PCR 寄存器中,用于保护 UH-Tree 的完整性。

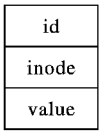


图 4　节点参数

每一次可信实体的度量请求到达,会先计算该可信实体的散列值,以该散列值为索引搜索整棵 UH-Tree,可分为两种情况:如果度量值已经在树中,找到度量值相对应的节点,记录从该节点到根节点的验证路径,并记录该路径上的所有节点;如果度量值不在树中,就需要将该该度量值作为叶子节点增加到 UH-Tree 中。

由于 UH-Tree 结构的特殊性,完整性度量算法增加新节点到树中与 RAMT 存在着很大不同。每增加一个可信实体的度量值到 UH-Tree,需要增加两个节点,其中一个节点为叶子节点,另一个节点为中间节点。根据树中叶子节点的数目不同,新增加一个叶子节点可以分为三种情况:1)当增加操作之前叶子节点数目为 2^n(n 为正整数)时,将新添加的中间节点作为根节点,保存旧的根节点散列值,并将新叶子节点的散列值与旧根节点的散列值连接后的散列值存到新的根节点中,记录最后一个叶子节点的散列值;2)当叶子节点数目为奇数时,添加的叶子节点为偶数节点,将中间节点作为叶子节点的父节点,另一个子节点为上个叶子节点,其散列值为叶子节点与上一个叶子节点散列值连接后的哈希值,并从中间节点开始迭代向上更新树中相关节点的散列值,直到根节点的散列值被更新,这时树中所有相关节点都已更新完毕;3)当叶子节点数目为偶数时,添加的叶子节点为奇数节点,将中间节点作为叶子节点的父节点,另一个子节点为上一个叶子节点的父节点,其散列值为其两个子节点的散列值连接后的哈希值,并从中间节点开始迭代向上更新树中的相关节点,直到根节点被更新。

UH-Tree 的完整性度量算法:

初始化 UH-Tree,创建根节点记为 root,将第一个可信实体的散列值存储到 root. value,并将 root. value 扩展至 TPM 的 PCR 寄存器,number 记录其中叶子节点的个数,maxinode 为树中的最大编号。

```
Step 1:
  temp = travel(id,tree[node]);                    //遍历 UH-Tree
  if(temp)
    path[n] = getpath(id, tree[node]);             //记录验证路径
Step 2:
  Ln = setinode(id, inode1, value);
  Rn = setinode(id, inode2, value);
  Rn. value = SHA1(e);
  inode1 = maxinode + 1;
  inode2 = maxinode + 2;
  maxinode = inode2;
  number = number + 1;
  if(number > 2n)
    N++;
  else
    goto Step 3;
Step 3:
  if(number%2n ! =1)
    goto Step 4;
  else
  {
    old_root. value = root. value;
    root = Ln;
    Ln. value = SHA1(Rn. value || old_root. value);
    last. value = Rn. value;
  }
goto Step 5;
Step 4:
  if((number%2) ==1)
    goto Step 5;
  else
  {
    N1. inode = Ln. inode;
    N2. inode = Ln. inode - 2m;
  }
  while(N1. inode ! = root. inode)
  {
    m = m + 1;
    N1. value = SHA1(N1. value || N2. value);
    N1. inode = (N1. inode + N2. inode)/2;
    N2. inode = (N1. inode - 2m);
  }
  Root. value = SHA1(N1. value || N2. value)
  goto Step 6;
Step 5:
  N1. inode = Rn. inode,
  N2. inode = N1 - 3, m = 3,
  while(N1. inode! = root. inode)
  {
    N1. value = SHA1(N1. value || N2. value);
    N1. inode = (N1. inode + N2. inode)/2;
    N2. inode = (N1. inode - 2m);
    m = m + 1;
    }
  root. value = SHA1(N1. value || N2. value);
  goto Step 6;
Step 6:
  setPCR(root. vaule);
                    //将根节点散列值扩展到 TPM 的 PCR 寄存器
```

为使算法更加容易理解,在下面的描述过程中,将采用算法示例的方式进行展示。在示例中,假设需要度量验证的可信实体分别为 e1,e2,e3,e4 和 e5。5 次度量验证都是新的度量请求,需要向 UH-Tree 中增加新的节点。如图 5 所示。

2.4 完整性验证算法

由于 UH-Tree 结构上的改变,完整性验证过程也相应改变,需要新的验证算法与之配合。设远程的度量平台为 VF,

本地计算平台为证明方 AS,具体的算法如下:

步骤 1 VF 产生一个 160 bit 的随机数 nonce,然后将所请求的验证对象和随机数以挑战的形式发送给 AS,即:VF→ChReq(nonce,PCR,e)。

步骤 2 AS 从 TPM 芯片中读取 UH-Tree 的根节点的散列值,并使用私钥 AIKpriv 生成应答 Root = Sig(PCR,nonce)AIK_{priv},获取 VF 所请求的可信实体的散列值对应的叶子节点 N_i, 到根节点 root 的认证路径: $n_i,n_j,\cdots,n_k$, root。

步骤 3 AS 将 Root、认证路径以及 $n_i,n_j,\cdots,n_k$, root 以挑战相应的格式发送给 VF,即 AS→VF:Res(Root,($n_i,n_j,\cdots,n_k$, root))。

步骤 4 VF 验证 AS 对 Root 的签名和随机数 nonce;同时,依据验证路径 $n_i,n_j,\cdots,n_k$, root 重新计算得到根节点的散列值,并与 AS 发送的 PCR 进行对比。

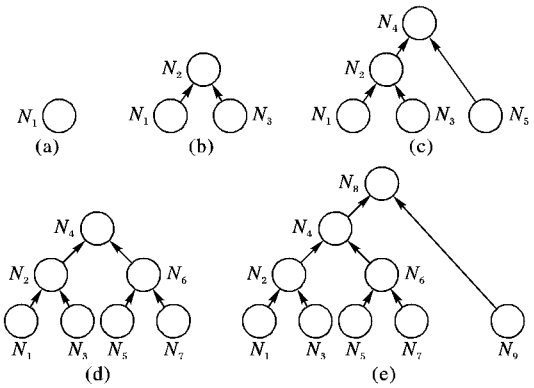


图 5 UH-Tree 建立过程

3 性能分析

3.1 隐私保护

IMA 体系为了将可信扩展到应用层,采用了基于度量列表 ML 的完整性验证机制。在度量阶段,需要将可信实体的度量值依次放入 ML 中,并将 ML 的完整性扩展到 TPM 的 PCR 中;在验证阶段,需要将整张 ML 发送给验证方,重新计算聚合值。这样就会将平台所有的可信实体的信息都完全暴露给验证方,可能造成隐私泄露的问题。

而 RAMT 采用基于 Merkle 树的可信实体完整性度量验证机制。在度量阶段,将哈希树的完整性扩展到 TPM 的 PCR 中;在验证阶段,本地平台只需要将验证的可信实体对应的散列值、认证路径序列以及 TPM 中存储的根节点散列值经过签名发送给验证方。因此,RAMT 增强了对本地平台的隐私保护。而 RAUT 延续了 RAMT 隐私保护的优点,在一次完整性验证过程中,验证方只能取得这次验证的可信实体的散列值和认证路径,并不会取得其他可信实体的散列值,从而保证了本地平台隐私的安全性。

3.2 性能分析

在远程度量验证过程中,哈希树构建开销和单个可信实体查询验证对于效率影响最大,本文将从这两个方面对比 RAMT 和 RAUT 的验证效率。

实验环境为 Lenovo Think Centre M8200t, Intel Core i5 CPU 2.6 GHz,4 GB 物理内存,Windows XP SP3 系统。实际测试结果显示,当树节点小于 216 时,时间的消耗大约都在 100 μs左右,两种结构之间的差距不明显,为了使结果更直观,采用两者的时间比作为实验的参考数据。

3.2.1 构建开销

设需要度量的实体相同,且数目为 M。在空间开销方面,由于实体的数目为 M, 则 Merkle 哈希树(Merkle-Hash Tree, MH-Tree)和 UH-Tree 的叶子节点均为 M, 两者存储空间都由构造 M 个叶子节点所需要建立的总节点数决定。设 MH-Tree 所需要的总节点数为 O_{MH}, UH-Tree 所需的总节点数为 O_{UH}, 由两类模型的构建规则可知:

$$O_{UH} = 2M - 1$$

两树所需要的存储空间与可信实体的数目 M 的关系如图 6 所示。

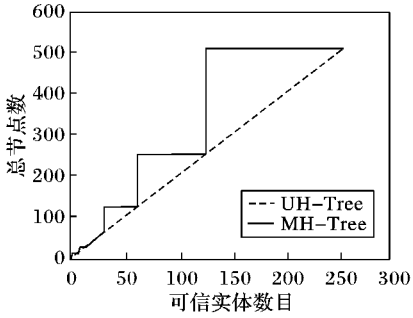


图 6 空间开销对比

通过图 6 可以看出,UH-Tree 在空间开销上要优于 MH-Tree,只有当度量的可信实体数目恰为 2^n(n 为整数)时,两者的空间开销相等。在可信实体数目恰好为 2^n+1 时,UH-Tree 相对于 MH-Tree 的效率最高。

在时间复杂度方面,建立 UH-Tree 与 MH-Tree 的复杂度与树的节点数目相关,节点的数目越多,构建树消耗的时间越多。在理论上,度量的可信实体数目越多,UH-Tree 在构建树的开销上优势越大,但 UH-Tree 的构建算法比 MH-Tree 复杂,在遍历节点和确定节点间关系上需要耗费更大的开销。故两者之间的开销需要通过具体的实验评估。

在实际测试中,UH-Tree 和 MH-Tree 都用数组作为存储结构,MH-Tree 需要填充节点时,直接填充空节点,以此提高测试的效率。可信实体的数目选取了几个典型的数据点:2^n+1、2^n 和 $2^{n-1}+1$。

如图 7,UH-Tree 和 MH-Tree 在可信实体数目为 2^n 时,由于总节点的数目相同,构建的时间基本相同;可信实体的数目恰好为 2^n+1 时,MH-Tree 相对于 UH-Tree 的空间消耗最大,耗费的时间也较长。

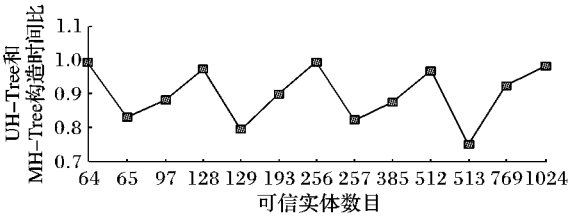


图 7 时间复杂度对比

3.2.2 查询验证开销

从理论上分析,当度量的可信实体数目为 2^n(n 为整数)时,UH-Tree 和 MH-Tree 的总节点数目相同,并且结构上均是满二叉树,验证其中的任意叶子节点消耗的时间应该是相同的。如果可信实体的数目小于 2^n, 这时可以分为两种情况,第一种情况是验证的叶子节点属于左子树,由于 UH-Tree 的

任意左子树都是满树,即左子树为二叉满树,这时 UH-Tree 和 MH-Tree 左子树上叶子节点的验证时间复杂度是相同的,均为 $O(N \text{ lb } N)$;第二种情况是验证的叶子节点属于不平衡的右子树,这时 UH-Tree 的右子树不是二叉满树,叶子的验证路径相对于 MH-Tree 较短,需要计算的散列值较少,验证效率更高。

在实际测试中,两种结构都采用相等的可信实体数目,分别验证以上三种情况。

如图 8 所示,当叶子节点的数目为 2^n 时,UH-Tree 和 MH-Tree 的验证效率基本相同,两者的验证时间比均在 1 周围呈无规则的浮动。当所取得验证节点均是第一个叶子节点时(如图 9 所示),属于左子树满树,UH-Tree 和 MH-Tree 的验证效率基本相同。

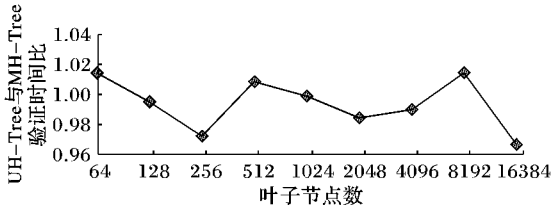


图 8　满二叉树验证效率对比

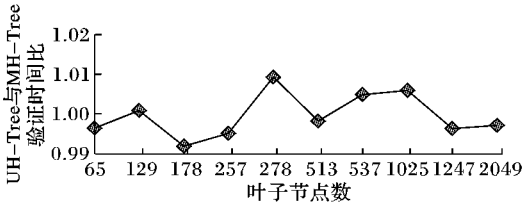


图 9　左子树验证效率对比

在图 10 中,本文选取的验证节点均为最后一个叶子节点,可以发现当验证节点属于不平衡的右子树中时,UH-Tree 的验证效率相对 MH-Tree 较高;当验证节点是 $2^n + 1$ 时,验证的效率达到最高,并且随着叶子节点数目的增长,验证效率逐渐提高,验证的时间复杂度为 $O(\text{lb } N)$。

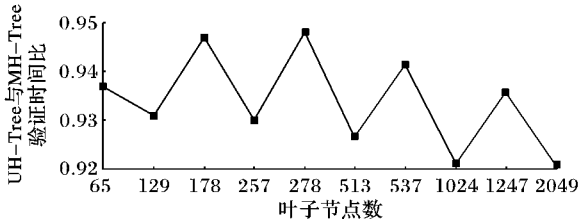


图 10　右子树验证效率对比

4　结语

可信计算是当前的研究热点,远程度量验证是其中的关键技术。徐梓耀等提出的 RAMT 方案采用 Merkle 树来存储可信实体的散列值,由于构造 Merkle 树需要的中间节点过多,造成了验证路径过长,影响验证效率。本文在此基础上基于非平衡哈希树提出新的完整性度量验证机制 RAUT,使用非平衡哈希树存储可信实体的度量值,减少哈希树中间节点数目,缩短叶子节点与根节点之间度量验证路径的长度,大大提高度量验证效率。实验结果表明,本方案在构建哈希树方面,时间复杂度和空间复杂度均优于 RAMT;在验证节点方面,RAMT 的时间复杂度 $O(N \text{ lb } N)$,而 RAUT 为 $O(\text{ lb } N)$,进一步提升了验证效率,且度量实体数目越大,提高的性能越明显。但随着用户计算机软件的不断增长,UH-Tree 的体积会不断增大,将会导致验证效率的降低。因此,下一步的主要工作是研究如何减少 UH-Tree 的节点个数,进一步提高验证效率。

参考文献:

[1] Trusted Computing Group. TCG specification architecture overview revision1.2[EB/OL].[2013-01-20]. http://www.Trusted computing group.org/.

[2] SAILER R, ZHANG X L, JAEGER T. Design and implementation of a TCG-based integrity measurement architecture[C]// Proceedings of the 13th USENIX Security Symposium. Berkley: USENIX Association, 2004:23 - 28.

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

[4] JAEGER T, SAILER R, SHANKAR U. PRIMA: Policy-Reduced integrity measurement architecture[C]// Proceedings of the 11th ACM Symposium on Access Control Models and Technologies. New York: ACM, 2006:19 - 28.

[5] LOSCOCCO P A, WILSON P W, PENDERGRASS J A. Linux kernel integrity measurement using contextual inspection[C]// Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing. New York: ACM, 2007:21 - 29.

[6] 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.

[7] SUH G E, CLARKE D, GASSEND B. AEGIS: Architecture for tamper-evident and tamper-resistant[C]// Proceedings of the 17th Annual International Conference on Supercomputing. New York: ACM, 2003:23 - 26.

[8] CHEN L, LANDFERMANN R, LOHR H. A protocol for property-based attestation[C]// Proceedings of the 1st ACM Workshop on Scalable Trusted Computing. New York: ACM, 2006:7 - 16.

[9] WANG Q, WANG C, REN K. Enabling public auditability and data dynamics for storage security in cloud computing[J]. IEEE Transactions on Parallel and Distributed System, 2011,22(5): 847 - 859.

[10] SLANEY M, CASEY M. Locality-sensitive hashing for finding nearest neighbors[J]. IEEE Signal Processing Magazine, 2008, 25(2):128 - 131.

[11] QIN Y, FENG D. Component property based remote attestation [J]. Journal of Software, 2009,20(6):1625 - 1641.(秦宇,冯等国.基于组件属性的远程证明[J].软件学报, 2009:20(6):1625 - 1664.)

[12] MAN J, WANG Z, LI C. Software behavior analysis at runtime in open network environment[J]. Journal of Computational Information System, 2011,7(1):127 - 134.

[13] LI X, ZUO X, SHEN C. System behavior based trustworthiness attestation for computing platform[J]. Acta Electronica Sinica, 2007,35(7):1234 - 1239.(李晓勇,左晓栋,沈昌祥.基于系统行为的计算平台可信证明[J].电子学报, 2007,35(7):1234 - 1239.)