第 36 卷 第 4 期 计 算 机 工 程 2010 年 2 月
Vol.36 No.4 Computer Engineering February 2010
·安全技术· 文章编号：1000—3428(2010)04—0115—03 文献标识码：A 中图分类号：TP309

# 基于 MH 树的外包数据库查询验证方法

袁多宝，王晓明
（暨南大学计算机系，广州 510632）

**摘 要：**分析 Merkle Hash(MH)树的结构特征，针对 MH 树的验证对象大、验证过程存在冗余、安全性低等不足，提出一种新的外包数据查询认证方法，使用部分物化中间节点的签名方法进行优化。分析结果表明，该方法具有网络附加负载小、验证快、安全性较高、能迅速实现篡改定位等优点。

**关键词：**外包数据库；查询验证；数据库安全；Merkle Hash 树

# Query Authentication Method Based on Merkle Hash Tree in Outsourced Database

YUAN Duo-bao, WANG Xiao-ming
(Department of Computer, Jinan University, Guangzhou 510632)

**【Abstract】**This paper analyzes the structural character of Merkle Hash(MH) tree. Aiming at that MH tree has disadvantages such as large verification objects, redundant verification procedures and relative low security and so on, this paper proposes a new authentication method in outsourced database and optimizes this method by using the partially materialize the signatures of some internal nodes. Analysis result show that the method has the advantages of small additional network load, fast verification, higher security, and being able to locate the tamper with the database at fine grain.

**【Key words】**outsourced database; query authentication; database security; Merkle Hash(MH) Tree

## 1 概述

外包数据库[1]模型主要由 3 个实体组成：数据所有者，数据库服务器和用户。在该模型中，数据所有者先将数据上传到外部数据库服务器，然后数据库服务器代表数据所有者向客户提供服务。外部服务器并非完全可信，因此，需要提供一种手段使用户能验证服务器返回的数据是否真实和完整。通常为了使客户能验证数据的正确性和完整性，服务器向用户返回查询结果的同时还要返回一些额外的信息，称这些额外的信息为验证对象(VO)。

目前已有不少外包数据库查询认证方案，其中有少数方案基于数字签名，这类技术需要给所有记录签名，并且只能逐个验证数据，效率较低，在实际应用中很难推广。为了减少签名次数和验证对象的大小、提高验证速度，多数查询验证技术使用文献[2]提出的 Merkle Hash(MH)树计算 VO。与基于数字签名的验证方法相比，MH 树有以下优点：只需对根节点签名，节省存储空间和签名计算，加快 MH 树的构建速度；hash 计算快且值较小，可减少通信成本和验证时间。

## 2 相关工作

目前实现数据验证的主要技术是添加攻击者不能控制的冗余验证信息。外包数据库的查询验证一般是通过综合运用消息认证码、数字签名及具有验证功能的数据结构等技术实现。文献[3]在提出了数据发布模式的查询验证问题，并给出了基于 MH 树的解决方案，这是目前主流查询验证算法的基础。文献[4]提出了 DSAC 查询认证技术。文献[5]在 MH 树的基础上提出了 MB 和 EMB 树认证模型，这 2 种模型支持外包数据的动态更新。文献[6]在 MH 树的基础上提出了 PMD 和 dPMD 技术，它们技术将认证结构和数据索引结构分开保存，其体系结构更加灵活。如果要为不支持认证功能的生产数据库添加认证功能，数据所有者只需要创建认证结构，不需要像以前的技术一样为了构建认证结构而重传所有记录。该技术支持需要验证和不需要验证 2 种需求，如果用户不需要验证数据，认证结构并不影响查询处理速度，其查询响应速度跟不支持认证的查询处理技术一样快。而且这两种认证技术比以往的技术能更好地支持外包数据库的动态更新、插入、删除等操作。

## 3 MH 树分析

本文用到的符号及其说明如表 1 所示。

**表 1 符号说明**

| 符号名称 | 描述 |
|---|---|
| $r_i$ | 数据库中的一条记录，其中 $S_{root}$ 是正整数 |
| $N_i$ | MH 树的一个节点，其中 $i$ 是正整数 |
| $H(x)$ | 用具有无碰撞性的 hash 函数 $H$ 求 $x$ 的 hash 值 |
| $h_i$ | 节点 $N_i$ 的 hash 值 |
| $S(x)$ | 使用公钥签名算法对 $x$ 进行签名 |
| $S_i$ | 节点 $N_i$ 的 hash 值的签名 |
| $VO$ | 验证对象 |
| $R_q$ | 满足查询条件 $q$ 的所有记录 |

**基金项目：**国家自然科学基金资助项目(60773083)；广东省自然科学基金资助项目(81510632010000022)

**作者简介：**袁多宝(1984－)，男，硕士研究生，主研方向：数据库安全；王晓明，教授、博士

**收稿日期：**2009-07-20 **E-mail：**yuanduobao@163.com

### 3.1 MH 树介绍

目前，主流的外包数据查询验证技术一般使用文献[2]提出的 MH 树计算验证对象。首先对数据库中的所有记录按照某个字段值排序，然后在已排序的记录集上构造 MH 树。MH 树是一棵二叉树，每个叶子节点指向数据库中的一条记录并保存对应记录的 hash 值，中间节点存储由它的 2 个孩子节点的 hash 值串联后再做 hash 计算得到的结果。最后，数据所有者使用公钥签名方案和私钥对根节点的 hash 值签名，并公布数据所有者的公钥。因为数据所有者使用公钥数字签名方案给根节点签名，所以客户能够验证根节点的正确性。因为 hash 函数具有无碰撞性，所以要恶意地篡改某节点的 hash 值又使得根节点的 hash 值不会发生改变，这在计算上是不可行的。因此，要验证记录是否正确，只需验证根节点的签名即可。MH 树结构如图 1 所示。

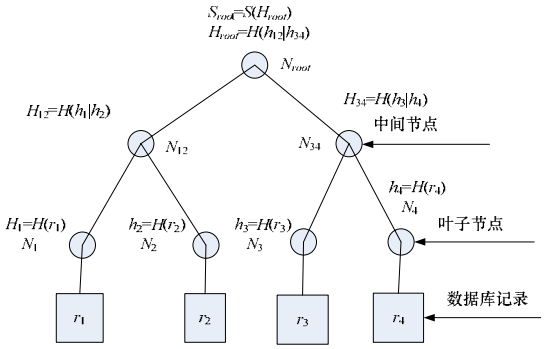


**图 1　MH 树结构**

具体查询验证过程如下：

(1)服务器找出满足条件 $q$ 的所有记录 $R_q$，设 $R_l$ , $R_r$ 分别是 $R_q$ 中值最小和最大的记录。

(2)服务器计算 *VO*。首先求从根节点到 $R_l$ ( $R_r$ )的路径上的所有节点，如果它们有左(右)兄弟节点，那么就将其左(右)兄弟节点的 *hash* 值保存到 *VO* 中。然后将 $S_{root}$ 保存到 *VO* 中。

(3)客户验证数据的正确性。由 $R_q$ 和 *VO* 从下到上迭代地构造 MH 树，并求出根节点的 hash 值，最后用求出的 hash 值和 $S_{root}$ 验证记录的正确性。

例如，在图 1 中，当满足条件 $q$ 的记录集 $R_q$={$r_1$,$r_2$,$r_3$} 时，数据库只需向用户返回 $R_q$ 和 *VO*={$h_4$,$S_{root}$}，用户就可以验证结果是否正确。其验证过程如下：首先由返回结果计算叶子节点 $N_1$ , $N_2$ , $N_3$ 的 hash 值(分别记作 $h_1$ , $h_2$ , $h_3$ )，然后计算 $N_1$ 和 $N_2$ 的父节点 $N_{12}$ 的 hash 值 $h_{12}$= $H$($h_1$|$h_2$) ，由 $h_3$ 和 *VO* 中的 $h_4$ 可计算 $h_{34}$ ，然后由 $h_{12}$ 和 $h_{34}$ 可计算出根节点的 hash 值 $h_{1234}$ ，最后使用根节点的签名 $S_{root}$ 就可验证 $h_{1234}$ 是否正确，从而可以验证返回结果是否正确。

### 3.2 Merkle Hash 树的不足

MH 树具有以下不足：

(1)验证过程较冗余。必须通过根节点才能验证数据是否正确，因此，为了验证数据，客户需要计算许多中间节点的 *hash* 值，降低了验证速度。

(2)安全性较低。MH 树的安全性过于依赖根节点的安全性，容易出现“ 单点故障 ”，一旦根节点的签名被篡改，不管数据库中的记录是否正确，用户都无法验证数据的真伪。

(3)不能快速定位被篡改记录的位置。如果返回结果被篡改，用户验证后只知道结果有误，但不能找到错误记录的位置。因此，一旦数据被篡改，数据所有者必须重传所有记录，服务器必须重构整棵树。

## 4　改进的 MH 树模型

针对 MH 树的不足，本文提出了一种新的外包数据查询认证技术。

### 4.1　验证模型

对所有中间节点和根节点，数据所有者用自己的密钥和签名算法生成相应的签名，用户使用数据所有者的公钥和签名验证数据的正确性。在数据存储过程中，中间节点既要保存文献[2]中的 hash 值，同时又要存储数据所有者对该 hash 值的签名。新的 MH 树结构如图 2 所示。

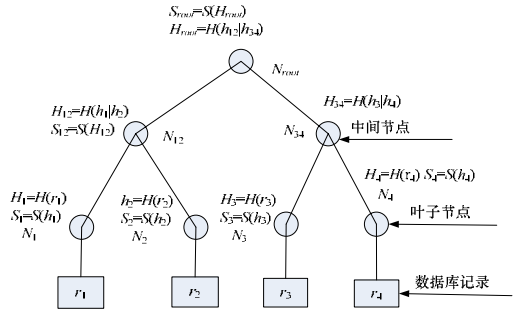


**图 2　新的 MH 树结构**

该方法的查询验证过程如下：

(1)搜索查询结果 $R_q$。设 $R_l$ , $R_r$ 分别是 $R_q$ 的左右边界值。

(2)计算包含 $R_q$ 中所有节点的最小子树。设该子树的根节点为 $N'_{root}$。

(3)计算 *VO*。首先将 $N'_{root}$ 的签名保存到 *VO* 中。然后求从 $N'_{root}$ 到 $R_l$ ($R_r$)的路径上的所有节点，如果它们有左(右)兄弟节点，则将其左(右)兄弟节点的 *hash* 保存到 VO 中。

(4)客户验证数据的正确性。从下到上迭代地构造子树并求出根节点的 hash 值，然后用求出的 hash 值和根节点 $N'_{root}$ 的签名来验证记录的正确性。

例如，在图 2 中，当满足条件 $q$ 的结果 $R_q$ = {$r_1$, $r_2$} , $R_l$ = $r_1$ , $R_r$ = $r_2$ 时，其验证过程如下：

(1)服务器计算包含 $R_q$ 中所有记录的最小子树。显然，$R_q$ 中所有记录共有的祖先是节点 $N_{12}$ 和 $N_{root}$ ，但距离叶节点最近的是 $N_{12}$ ；故最小子树的根为 $N'_{root}$ = $N_{12}$。

(2)服务器计算 *VO*。先将 $N'_{root}$ 所对应的签名 $S_{12}$ 存储到 *VO* 中。由于从 $N'_{root}$ 到 $R_l$ 的路径上和从 $N'_{root}$ 到 $R_q$ 路径上都没有节点，因而 *VO*= $S_{12}$。

(3)服务器向客户返回 $R_q$ 和 *VO* = {$S_{12}$} 。当用户接收到服务器返回的数据后，即可验证返回的查询结果是否正确完整。

如果根节点被破坏而其他节点没有被破坏，该模型仍可验证数据的正确性。此时，客户如果发现根节点被破坏，就会向服务器请求根节点的 2 个孩子节点的签名，接到返回的数据后即可验证数据。验证过程分为 2 步：(1)验证 $r_1$ 和 $r_2$ ；(2)验证 $r_2$，其验证过程同文献[2]中的 MH 树验证过程。

如果数据所有者发现数据库中有记录被恶意修改了，就可以按照以下过程精确地定位被修改的记录，减少上传的数据量：

(1)向服务器请求根节点的左、右孩子节点(分别是 $N_l$ , $N_r$ )的签名，以及根节点的左右子树所对应的记录集。

(2)用 $N_l$ 验证左子树中的记录。如果验证正确表示该子树中的记录没有被修改，该子树的定位过程结束。如果验证错误则表示该子树中有记录被修改了，则需要返回步骤(1)在以 $N_l$ 为根节点的子树中精确定位被修改的记录。然后使用 $N_r$ 验证右子树中的记录。

(3)如果 $N_l$ 和 $N_r$ 是叶子节点，则表示它们对应的记录已被修改，定位过程结束。显然，在搜索被篡改了的记录时，本方案充分利用了二叉树查找很快的特点，定位效率较高。

### 4.2 MH 树新模型的优化

给所有的中间节点签名增加了服务器的存储开销和 MH 树的构建时间。因此，为了减少服务器的存储成本，可以利用文献[6]提出的部分物化(Partially Materialize)的思想优化本文的模型。该优化算法的基本思想是：只对 MH 树的部分中间节点签名，减少 MH 树的构建时间和服务器端的存储开销，同时减少数据所有者的通信成本。

具体优化过程如下：(1)对所有的记录排序，按照每组 $\alpha$ 个节点对叶子节点分组，数据拥有者可根据实际需要设置 $\alpha$ 的值。(2)为每一组记录集构建一颗普通的 MH 树，称这些 MH 树为第一层子树，只给这些子树的根节点签名。(3)按照每组 $\alpha$ 个节点对第 $i$ 层子树的根节点分组；为每组构建 MH 子树，称这些树为第 $i$+1 层子树，最后只给子树的根节点签名。(4)重复步骤(3)使得最后只有一颗 MH 子树。

当 $\alpha = 4$ 时，优化后的一颗 MH 树的结构如图 3 所示，其中，长方形表示节点分组；小圆圈表示节点；实心圆圈表示该节点既要保存 *hash* 值又要保存签名；空心圆圈表示该节点只需保存 *hash* 值；大圆圈表示一颗 MH 子树。

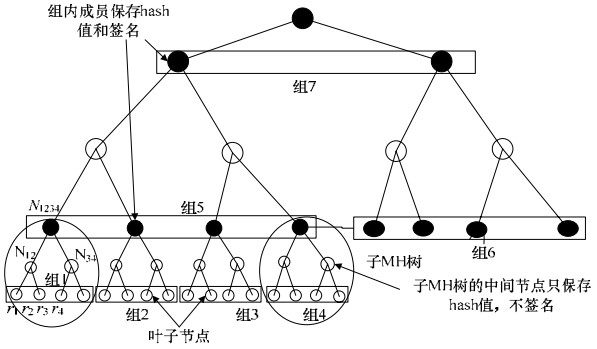


**图 3 优化后的 MH 树结构**

验证过程如下：(1)计算查询结果 $R_q$。假设 $R_l$、$R_r$ 分别是是 $R_q$ 中的最小值和最大值。(2)构建 MH 子树。首先求包含 $R_q$ 中所有节点的最小子树，如果该子树的根节点保存了签名则返回该子树，否则就以该根节点的第一个保存了签名的祖先节点为根节点构建 MH 子树。假设最后构建的 MH 子树的根节点是 $N'_{root}$。(3)在计算出的 MH 子树上计算 *VO*，计算方法与优化前一样。(4)客户验证数据。

## 5 性能分析

改进的 MH 树保留了文献[2]中 MH 树的所有特点。但改进后由于部分中间节点保存了签名，通常客户通过中间节点就可以验证数据的正确性，减少了中间节点的 hash 计算，减少了 *VO* 的大小，加快了服务器的查询处理速度和客户的验证速度，减少了客户和服务器间的通信成本。

由于某些中间节点和根节点保存了数据所有者签名，根节点被破坏后，用户通过中间节点仍能验证数据的正确性，从而减少了对根节点的依赖性，增强了其抗攻击的能力。

如果有记录被篡改了，数据所有者可以精确的定位被篡改的记录，然后更新其值，而不需要重新上传所有记录。而文献[2]中的 MH 树，只要数据库中有记录被篡改了，数据所有者就必须重新上传数据库中的所有记录。因为需要对中间节点进行签名，所以增加了服务器端的存储开销、构建 MH 树的成本以及数据所有者向服务器发送数据的通信成本。本文利用文献[6]提出的部分物化的技术对该方案进行优化。优化后，在可接受的条件下，本方案适当增加了服务器的存储成本，减少了通信量，增强了系统的安全性，提高了查询处理速度和验证速度，同时还能细粒度地定位被篡改的记录。表 2 从几个方面对比了 MH 树模型改进前后的性能。

**表 2 性能比较结果**

| 性能和安全性 | 改进前的 MH 树 | 改进后的 MH 树 |
|---|---|---|
| 服务器与数据所有者间的通信成本 | 低 | 高 |
| 服务器端的构建成本 | 低 | 高 |
| 服务器端的存储开销 | 低 | 高 |
| 验证对象 *VO* 的大小 | 大 | 小 |
| 服务器与客户间的通信成本 | 高 | 低 |
| 服务器处理查询请求的速度 | 慢 | 快 |
| 客户的验证速度 | 慢 | 快 |
| MH 树的安全性 | 低 | 高 |

## 6 结束语

本文针对 MH 树的不足提出一种新的认证模型。主要做了以下工作：(1)增强了 MH 树的安全性；(2)加快了服务器的查询响应速度和客户的验证速度；(3)实现了篡改定位功能，当有记录被破坏但签名仍然完好时，可以迅速找到被篡改记录的具体位置，减少需要重传的数据，加快 MH 树的重构速度。虽然只是对 MH 树进行了改进，但由于目前已有的外包数据库查询认证技术大部分都是基于 MH 树，因此可以用本文的模型改进某些现有的外包数据库查询认证技术。

## 参考文献


[1] Hacigumus H, Mehrotra S, Iyer B, et al. Providing Database as a Service[C]//Proc. of the 18th International Conference on Data Engineering. San Jose, CA, USA: IEEE Computer Society, 2002.

[2] Merkle R C. Protocols for Public Key Cryptosystems[C]//Proc. of IEEE Symposium on Research in Security and Privacy. [S. l.]: IEEE Press, 1980.

[3] Devanbu P, Gertz M, Martel C, et al. Authentic Third-party Data Publication[C]//Proc. of the 14th IFIP TC11/WG11.3 Annual Working Conference on Database Security. Schoorl, Netherlands: [s. n.], 2000.

[4] Mykletun E, Narasimha M, Tsudik G. DSAC: Integrity for Outsourced Databases with Signature Aggregation and Chaining[C]//Proc. of ACM CIKM'05. New York, USA: ACM Press, 2005.

[5] Li Feifei, Marios H, George K, et al. Dynamic Authenticated Index Structures for Outsourced Database[C]//Proc. of ACM SIGMOD'06. Chicago, Illinois, USA: ACM Press, 2006.

[6] Kyriakos M, Dimitris S, HweeHwa Pang. Partially Materialized Digest Scheme: An Efficient Verification Method for Outsourced Databases[J]. VLDB, 2009, 18(1): 345-362.


编辑 金胡考