doi: 10.12068/j.issn.1005-3026.2018.08.007

支持全操作的公共可验证外包数据库方案

王 强, 玄鹏开, 王红伟, 周福才
(东北大学 软件学院, 辽宁 沈阳 110169)

摘 要: 针对当前外包数据库完整性研究方案存在的时空开销大、查询和更新效率低、无法同时支持多种SQL查询结果的完整性验证等问题,提出一个支持全操作的公共可验证外包数据库模型,并给出该模型的形式化定义和安全性定义. 在模型的基础上利用双线性映射累加器和认证跳表实现了包含三方实体且支持全操作的公共可验证外包数据库方案,给出了方案中各算法的具体描述及实体间的交互过程. 最后分别对方案的安全性和效率进行分析,结果表明,该方案具有不可伪造性,并具有较高的效率.

关 键 词: 完整性验证;公共可验证;外包数据库;全操作;双线性映射累加器
中图分类号: TP 309 **文献标志码**: A **文章编号**: 1005-3026(2018)08-1098-06

A Publicly Verifiable Outsourced Database Scheme with Full Operations

WANG Qiang, *XUAN Peng-kai*, *WANG Hong-wei*, *ZHOU Fu-cai*
(School of Software, Northeastern University, Shenyang 110169, China. Corresponding author: ZHOU Fu-cai, E-mail: fczhou@mail.neu.edu.cn)

Abstract: Aiming at the existing shortcomings of outsourced database schemes including heavy temporal and spatial cost, low efficiency for query and update, and lack of the support for the complete verification of multiple SQL query results, this paper proposes a publicly verifiable outsourced database model supporting full operations. The formal definition and security definition are presented. On the basis of the model, a publicly verifiable outsourced database scheme, composed of three entities, is constructed using bilinear mapping accumulator and authenticated skip list. Also, the implement and the communication are described in detail. Finally, the security and efficiency are analyzed, respectively, which shows that the proposed scheme is with unforgeability and high efficiency.

Key words: integrity verification; public verification; outsourced database; full operation; bilinear-map accumulator

近年来,数据库外包作为云计算一种新兴的应用模式,因其成本低、效果好、操作易、灵活性高等特点,已成为用户或企业数据库外包的趋势. 然而,取消数据中心,将数据库外包至第三方数据库服务提供商进行管理和维护,用户将面临着服务提供商恶意篡改数据库内容、伪造查询结果和遭受外部攻击等风险,难以保证查询结果的完整性[1],因此实现支持查询结果完整性验证的数据库外包方案对云计算的发展具有重大意义.

目前,针对外包数据库查询结果完整性验证问题已展开大量研究,主要分为三类:基于数字签名、基于认证数据结构和基于可验证计算. 基于数字签名的方法[2-3]存在验证代价较高和更新代价高的问题. 基于认证数据结构的方法是对基于数字签名方法的改进,通常包含两类:基于Merkle哈希树[4-5]和基于累加器[6]. 基于Merkle哈希树虽不需要对每一元组进行签名,但由于叶子节点是排序好的元组的哈希值,任何一次更新都可能

收稿日期: 2017-04-24
基金项目: 中央高校基本科研业务费专项资金资助项目(N151704002); 国家自然科学基金资助项目(61772127, 61472184).
作者简介: 王 强(1991-),男,辽宁桓仁人,东北大学博士研究生; 周福才(1964-),男,辽宁沈阳人,东北大学教授,博士生导师.

破坏已有的认证数据结构,更新代价较高. 验证查询结果正确性的证据随元组的数量呈对数级增长,只支持单维范围和元素隶属关系的查询. 另一种基于累加器的方法,只支持集合交集、并集、差集及元素隶属关系的查询,不支持求和、计数、平均值、范围、最大值、最小值、连接和嵌套查询. 基于可验证计算的方法分为两类:基于电路[7-8]和基于 RAM[8-9]. 两者均是将外包数据库编译到程序中,在编译好的程序中输入查询语句,返回与之对应的结果. 这两种方法的计算代价较高,难以应用于实际. 基于电路的方法效率比基于 RAM 的更低,程序中电路的规模至少与数据库本身一样大,且不支持更新. 而基于 RAM 模型的系统虽然支持全部查询操作,但是编码的难度较高.

综上所述,已有的外包数据库完整性验证方案存在时空开销大、查询和更新效率低和不支持全操作(支持多种 SQL 查询)等不足,难以适于实际应用. 针对此问题,本文提出一个支持全操作的公共可验证外包数据库模型(publicly verifiable outsourced database with full operations, PVODB-FO),并给出了该模型的形式化定义和安全性定义. 在模型的基础上,借助双线性映射累加器和认证跳表,本文方案不仅可以同时支持连接、多维范围、函数和嵌套等查询操作而且可高效更新.

1　双线性 q-SDH 假设

令 λ 为安全参数,G 和 G_T 为 p 阶循环群,g 为 G 的生成元,$e: G\times G\to G_T$ 为双线性映射,随机选取 $s\in \mathbf{Z}_p^*$,并给定 $(q+1)$ 个元素 $g, g^s, g^{s^2}, \cdots, g^{s^q}$,如果不存在一个概率多项式时间算法能以不可忽略的概率计算出 $(a, e(g,g)^{1/(a+s)})$,其中 $a\in \mathbf{Z}_p^*/\{-s\}$,则称双线性 q-SDH 假设是困难的.

2　PVODB-FO 模型

PVODB-FO 主要包含三方实体:数据拥有者、客户和云服务器. 系统架构如图 1 所示:数据拥有者首先生成公私钥对,并为外包数据库计算认证数据结构和摘要,然后将数据库和认证数据结构外包至云服务器,最后将公钥和摘要发送给客户. 客户向云服务器发送 SQL 查询请求,当云服务器接收到客户请求后,返回对应的查询结果及证据. 最后客户验证查询结果的完整性.

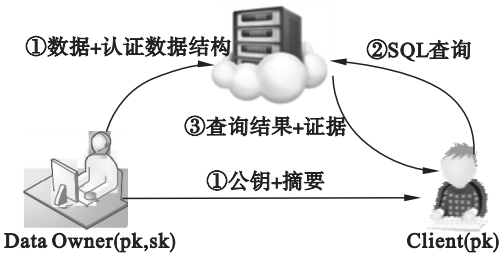

图 1　系统总体架构

Fig. 1　System architecture

2.1　模型定义

PVODB-FO 由 6 个概率多项式时间算法组成,PVODB-FO = {KeyGen, Setup, Query, Verify, UpdateO, UpdateS} 具体描述如下:

1) $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{KeyGen}(1^\lambda)$:数据拥有者输入安全参数 λ,算法输出公私钥对 $(\mathrm{pk}, \mathrm{sk})$.

2) $(D_{\mathrm{ADS}}, \delta) \leftarrow \mathrm{Setup}(D, \mathrm{sk})$:数据拥有者输入外包数据库 D 和私钥 sk,算法输出认证数据结构 D_{ADS} 和对应的摘要 δ.

3) $(R, \pi) \leftarrow \mathrm{Query}(Q, D, D_{\mathrm{ADS}})$:云服务器输入查询语句 Q、外包数据库 D 和认证数据结构 D_{ADS},算法输出查询结果 R 和证据 π.

4) $b \leftarrow \mathrm{Verify}(\mathrm{pk}, \delta, R, \pi)$:客户输入查询公钥 pk、结果 R、摘要 δ 和证据 π,算法输出比特 b,如果 $b=1$ 表示验证成功,否则表示验证失败.

5) $(\delta') \leftarrow \mathrm{UpdateO}(u, \delta, \mathrm{sk})$:数据拥有者输入更新语句 u、摘要 δ 和私钥 sk,算法输出更新后的摘要 δ'.

6) $(D', D_{\mathrm{ADS}}') \leftarrow \mathrm{UpdateS}(u, D, D_{\mathrm{ADS}})$:云服务器输入更新语句 u、外包数据库 D 和认证数据结构 D_{ADS},算法输出更新后的数据库 D' 和认证数据结构 D'_{ADS}.

2.2　安全性定义

给定安全参数 λ 和外包数据库 D,敌手 A 伪造的查询结果及证据始终不能通过验证. 下面通过安全性实验给出该方案安全性的形式化定义,安全性实验步骤如下.

初始化阶段:

1) 挑战者 C 执行算法 KeyGen,生成公私钥对,将公钥 pk 发送给敌手 A,私钥 sk 保留.

2) 挑战者 C 执行算法 Setup,输出认证数据结构 D_{ADS} 和摘要 δ,并将 D_{ADS} 和 δ 发送给 A.

3) 敌手 A 向 C 进行多项式 poly (λ) 次查询或更新操作:①查询——A 向 C 发送查询请求 Q,C 执行算法 Query 并将 (R, π) 返回至 A;②更新——A 向 C 发送更新请求 u, C 执行算法

UpdateO 和 UpdateS,最后将更新后的 δ', D' 和 D'_{ADS} 发送至 A.

挑战阶段:当敌手 A 决定结束步骤 3),选择一个目标查询请求 q^*,A 伪造对应的查询结果 R^* 和证据,且算法 Verify(pk,δ,R^*,π^*)输出 1.

如果对于任意具有概率多项式时间计算能力的敌手 A,都不能以不可忽略的概率 negl(λ)赢得上述安全性实验,则称 PVODB-FO 是安全的.

3 PVODB-FO 方案

3.1 密钥生成

算法 KeyGen 首先输入安全参数 λ,调用双线性映射算法生成(p,g,G,G_{T},e),其中 p 是 λ 位的素数,g 为 G 的生成元,G 和 G_{T} 为两个 p 阶循环群,双线性映射 $e: G\times G\rightarrow G_{\mathrm{T}}$. 随机选择 $s\in \mathbf{Z}_p^*$,计算$(g^s, g^{s^2}, \cdots, g^{s^q})$,其中 $q=\mathrm{poly}(\lambda)$. 选取对称加密方案 ε 的密钥 sk_ε.

最终输出公钥 $\mathrm{pk}=(g^s, g^{s^2}, \cdots, g^{s^q})$,私钥 $\mathrm{sk}=(s,\mathrm{sk}_\varepsilon)$.

3.2 系统初始化

对外包数据库 D 中每个表 T,按如下方式构建认证数据结构 D_{ADS} 和摘要 δ:不失一般性,假定 T 有 $n+1$ 行,对表中任意 2 列 i,j 进行组合得到键值对集合 $S_{i\times j}^T=\{x_0,\cdots,x_n\}$,其中 $x_k=(x_{ki}, x_{kj})$, x_{ki} 为 key, x_{kj} 为 value. 构建认证跳表(authenticated skip list, ASL)$\mathrm{ASL}_{i\times j}^T$ 如图 2 所示,其存储了集合 $S_{i\times j}^T$ 的元素,S_0 按 x_{ki} 递增的方式存储集合 $S_{i\times j}^T$ 中所有元素,以及 $-\infty$ 和 $+\infty$ 两个哨兵节点. 最高层 S_l 中的 $-\infty$ 节点为跳表搜索的起始节点. S_i 中的每个节点 v 存储 v. elem, v. down 和 v. right. v. elem 表示该节点内存储的元素, v. down表示下层 S_{i-1} 中对应的节点, v. right 表示 v 右侧直接相连的节点. 内部节点 u 的子树 T_u 对应的叶子节点集合 N,满足 $\cup T_u=N$. 内部节点 u 的 key 为子树 T_u 中叶子节点中最小的 key,节点 u 的 value 为 $\varepsilon_{\mathrm{sk}_\varepsilon}\left(\prod_{x_{kj}\in T_u}(x_{kj}^{-1}+s)\right)\| g_{x_{kj}\in T_u}^{\prod(x_{kj}^{-1}+s)}$. ASL 构建完毕,输出其摘要 $\delta_{i\times j}^T$. 最后使用可交换哈希函数 H 对表 T 每行计算哈希值 $H(x_{k1},\cdots,x_{km})$,其中 m 表示表 T 列的数量.

最后输出认证数据结构 D_{ADS} 包含全部 $\mathrm{ASL}_{i\times j}^T$,摘要 δ 包含全部 $\delta_{i\times j}^T$ 和哈希值 $H(x_{k1},\cdots,x_{km})$.

ε 选择 AES 加密方案实现,连接符右侧部分为叶子节点中 value 的累加值,用于查询时生成证据,左侧部分是对右侧指数部分的加密结果,用于更新数据库,因此只有拥有私钥 sk_ε 的数据拥有者才能更新.

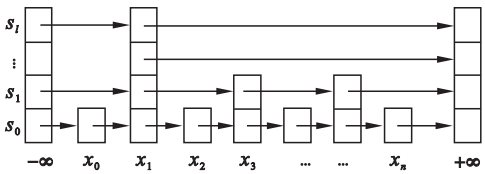

图 2 认证跳表结构图

Fig. 2 Structure of authenticated skip list

3.3 查询与验证

$(R,\pi)\leftarrow\mathrm{Query}(q, D, D_{\mathrm{ADS}})$ 以查询语句 $q\in Q$、数据集 D 和认证数据结构 D_{ADS} 为输入,输出查询结果 R 和证据 π. $b\leftarrow\mathrm{Verify}(\mathrm{pk},\delta,R,\pi)$ 输入查询公钥 pk、结果 R、摘要 δ 和证据 π,输出一个比特 b:$b=1$ 表示验证通过,$b=0$ 表示验证未通过. 不同的 SQL 查询,生成的证据也不同.

3.3.1 常规查询与验证

查找查询(Search):x 为待查找元素,从 ASL 根节点搜索,令 v 表示当前节点,比较 x 与 v. elem 大小. 若 $x>v$. elem,向右搜索,若 v 为右侧哨兵节点,搜索结束,查找失败;否则令 $v=v$. right. 若 $x\leqslant v$. elem,向下搜索,若 v. down = null 且 $x=v$. elem,则找到元素 x;若 v. down $\neq$ null 且 $x\leqslant v$. elem,令 $x=v$. down;若 v. down = null 且 $x\neq v$. elem,查找失败.

范围查询(RangeCover):给定 x_{L}, x_{R} ($x_{\mathrm{L}}\leqslant x_{\mathrm{R}}$),查询 $x_{\mathrm{L}}\sim x_{\mathrm{R}}$ 范围内的所有元素 x. 将 x_{L}, x_{R} 视为待查找元素,分别调用 Search 查询,找到两元素,进而得到 x_{L}, x_{R} 所在范围的查询结果 N.

上述两种查询操作,通过比对摘要值就可验证查询结果的正确性.

3.3.2 连接查询与验证

查询:连接查询 q 涉及表 T 的第 i 列和表 T' 的第 j 列,令 C_i 和 C_j 表示每一列中元素的集合. 使用 $k_{\mathrm{L}}=-\infty$ 和 $k_{\mathrm{R}}=\infty$,在 $S_{i\times i}^T$ 上执行范围查询,返回 $S_{i\times i}^T$ 对应的 $\mathrm{ASL}_{i\times i}^T$ 的根节点值,根节点值中包含 $\mathrm{acc}(C_i)$. 然后,在 $S_{j\times j}^{T'}$ 上执行相同操作得到 $\mathrm{acc}(C_j)$. 利用可验证集合交集运算得到 C_i 和 C_j 的交集 C^*. 最后云服务器返回交集 C^* 中元素所在的行的数据作为查询结果,将该行的哈希值和交集计算的证据作为查询的证据.

验证:客户利用集合交集的证据来验证集合交集计算的正确性. 若验证失败,终止并拒绝查询结果;否则验证元素哈希值的正确性,若验证失

败，终止并拒绝查询结果，否则接受查询结果.

3.3.3　多维范围查询与验证

查询：多维范围查询结果可通过 2 维范围查询交集得到. 不失一般性，以 2 维范围查询为例，假定所查询范围分别是$[w^-, w^+]$和$[z^-, z^+]$，在$S^T_{w\times 1}$对应的$\mathrm{ASL}^T_{w\times 1}$上使用边界值$w^-$和$w^+$作为输入，调用 RangeCover 查询，输出查询结果R_w. 同样，将边界值z^-和z^+作为输入，在$\mathrm{ASL}^T_{z\times 1}$上执行 RangeCover 查询，输出查询结果$R_z$. 利用可验证集合交集运算对集合$R_w$和$R_z$做交集，进而得到 2 维范围查询的结果$R^*$. 最后云服务器返回$R^*$中元素作为查询结果，哈希值和交集计算的证据作为查询的证据.

验证：用集合交集的证据来验证集合交集计算的正确性，若验证失败，终止并拒绝查询结果；否则验证元素哈希值的正确性，若验证失败，终止并拒绝查询结果，否则接受查询结果.

3.3.4　函数型查询与验证

为满足统计需求，巧妙地设计双线性映射累加器，实现了集合求和运算查询结果的完整性验证，并在此基础上实现了计数和平局值查询.

假定求和查询所在列对应的集合$S_j = \{x_{j1}, \cdots, x_{jn}\}$，集合中元素的累加值$\mathrm{acc}(S_j) = g^{\prod_{i=1}^{n}(x_{ji}+s)}$. 累加值指数部分对应的多项式为$\prod_{i=1}^{n}(x_i^{-1}+s) = s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0$，其中$a_0 = \prod_{i=1}^{n} x_i^{-1}$，$a_1 = \sum_{i=1}^{n}(\prod_{j\neq i} x_j^{-1})$. 集合$S_j$中元素的和$\mathrm{sum} = \sum_{i=1}^{n} x_i = a_1 a_0^{-1} \bmod p$. 求和运算的证据为$w_1, w_2$，其中$w_1 = g^{s^{n-1}+\cdots+a_2 s+a_1}$，$w_2 = g^{s^{n-2}+\cdots+a_3 s+a_2}$. 若$e(g^s, w_1) = e(\mathrm{acc}(S)/g^{a_0}, g)$，$\mathrm{sum} = a_1 a_0^{-1} \bmod p$和$e(g^s, w_2) = e(w_1/g^{a_1}, g)$同时成立，则接受求和计算结果.

求和查询过程如下.

查询：对表T的第j列进行求和查询，云服务器利用多项式快速傅里叶变换算法求得a_0和a_1，计算$a_1 a_0^{-1} \bmod p$即为求和结果 sum. 返回证据w_1和w_2.

验证：判断$e(g^s, w_1)$与$e(\mathrm{acc}(S)/g^{a_0}, g)$是否相等，若不相等，则$a_0$系伪造，终止并拒绝查询结果，否则判断$e(g^s, w_1)$与$e(\mathrm{acc}(S)/g^{a_0}, g)$是否相等，若不相等，$a_1$系伪造，终止并拒绝查询结果，否则判断 sum 与$a_1 a_0^{-1} \bmod p$是否相等，若不相等则拒绝查询结果，否则接受查询结果 sum.

计数查询过程如下.

查询：云服务器通过 SQL 计数查询可直接获得计数值R. 构造证据时，云服务器在表T中需要计数的第j列后增加 1 列j'，j'列中每行元素等于j列对应行元素加 1，返回第j列和j'列求和查询的证据作为计数查询的证据.

验证：首先验证求和查询$\mathrm{sum}(j)$和$\mathrm{sum}(j')$的正确性，再验证计数值$\mathrm{count}(j)$是否等于$\mathrm{sum}(j') - \mathrm{sum}(j)$；若相等则接受计数结果，否则拒绝计数结果.

平均值查询过程如下.

查询：第j列平均值查询的结果可通过求和查询结果除以计数查询结果得到. 返回的证据为求和查询的证据和计数查询的证据.

验证：首先验证求和查询与计数查询的正确性，再验证平均值查询结果$\mathrm{avg}(j)$是否等于$\mathrm{sum}(j)/\mathrm{count}(j)$；若相等则接受平均值结果，否则拒绝平均值结果.

最大值与最小值查询过程如下.

查询：对第j列作最大值和最小值查询，可直接通过 SQL 查询得到；构建证据时，返回在认证跳表$\mathrm{ASL}^T_{j\times j}$(key 和 value 均由第$j$列元素构成)上对最大值和最小值做 Search 查询的证据.

验证：验证与 Search 查询的验证相同.

3.3.5　嵌套查询与验证

查询：假定执行嵌套查询$Q_1 \circ Q_2$，云服务器相当于分别执行查询Q_1和Q_2，然后对Q_1和Q_2查询结果进行可验证的集合交集或并集操作. 云服务器返回的证据为中间结果的摘要和验证最终结果的证据；因不需要返回中间结果的全部证据，从而极大地减小了存储与验证的开销.

验证：首先验证中间结果的摘要值与对应结果的摘要值是否相等，若不相等，终止并拒绝查询结果，否则验证结果的正确性. 根据查询类型在上文中找到相应的验证方法，若验证失败，终止并拒绝查询结果，否则接受查询结果.

3.4　更新

更新过程由数据拥有者和云服务器交互实现，更新操作包括插入、删除、修改数据库中的一行(或多行)数据. 云服务器更新外包数据库对应的数据和认证跳表，数据拥有者更新摘要值. 插入时，首先通过 Search 查询找到低于插入元素$x = (\mathrm{key}_x, \mathrm{value}_x)$且最接近$\mathrm{key}_x$的位置，记录下搜索路径. 找到最底层的位置后，新建节点，连入底层跳表. 随机生成元素的最高层，由低到高在每一层新建节点，通过搜索路径进行相应的连接操作. 删除时，从起始节点开始搜索删除节点x，找到x前一个位置的节点v，则$v.\mathrm{right}$即为x所在位置，记

录搜索路径. 沿路径向上依次删除含有元素 value_x 的节点. 修改时,可先通过删除、再插入来实现. 更新数据后,数据拥有者输出更新后的摘要 δ',并从底层节点一直更新至根节点. 每个节点的更新过程:使用私钥解密 $\varepsilon_{\text{sk}_\varepsilon}(\prod_{x_{kj}\in T_u}(x_{kj}^{-1}+s))$ 得到 $\prod_{x_{kj}\in T_u}(x_{kj}^{-1}+s)$,更新得到新的 $\prod_{x_{kj}\in T_u}(x_{kj}^{-1}+s)'$,计算更新后的累加器值 $g_{\prod_{x_{kj}\in T_u}(x_{kj}^{-1}+s)'}$,然后将 $\varepsilon_{\text{sk}_\varepsilon}(\prod_{x_{kj}\in T_u}(x_{kj}^{-1}+s)')\parallel g_{\prod_{x_{kj}\in T_u}(x_{kj}^{-1}+s)'}$ 发送给云服务器. 最终云服务器输出更新数据 D'和 ASL.

4 安全性分析和性能分析

4.1 安全性分析

由于认证跳表弥补了双线性映射累加器不能实现范围查询、最大值和最小值查询的不足,并且认证跳表的安全性在已有方案[8]中已得证明,这里不在赘述. 其他类型的查询都是基于求和查询的,因此只要证明求和查询的安全性即可保证该方案的安全性. 安全性证明如下:

如果存在一敌手 A 攻破了求和查询的安全性,即伪造了一个结果及证据并通过了算法 Verify 的验证,那么敌手 A 可构造一算法 B,以不可忽略的概率解决双线性 q-SDH 难题. 证明过程如下.

算法 B 执行算法 KeyGen 生成公私钥对(pk, sk),将 pk 发送给 A,sk 私密保存. A 选定一个集合 S,算法 B 计算累加值 acc(S)并公开,敌手伪造求和的结果 sum^*,并且证据 $a_0^*, a_1^*, w_0^*, w_1^*$ 通过验证,那么一定满足 $a_0^*\neq a_0$ 或 $a_1^*\neq a_1$.

情形 1:当 $a_0^*\neq a_0$ 时,因为 $e(g^s, w_1^*)=e(\text{acc}(S)/g^{a_0^*}, g)$ 且 $e(g^s, w_1)=e(\text{acc}(S)/g^{a_0}, g)$,敌手 A 可计算出 $g^{1/s}=(w_1^*/(g^{s^{n-1}+\cdots+a_2s+a_1}))^{(a_0-a_0^*)^{-1}}$,这与双线性 q-SDH难题相矛盾.

情形 2:当 $a_0^*\neq a_0$ 且 $a_1^*\neq a_1$ 时,因为 $e(g^s, w_1^*)=e(\text{acc}(S)/g^{a_0^*}, g)$,可得 $w_1^*=g^{s^{n-1}}+\cdots+a_2s+a_1$. 又因为 $e(g^s, w_2)=e(w_1/g^{a_1^*}, g)$,敌手 A 可计算出 $g^{1/s}=(w_2^*/(g^{s^{n-2}+\cdots+a_2}))^{(a_1-a_1^*)^{-1}}$,这与双线性 q-SDH 难题相矛盾.

4.2 性能分析

从查询类型的丰富性和算法效率两方面,将本文方案与其他方案进行了对比.

在查询类型丰富性方面,将本文方案与基于 RAM、基于电路和基于树或签名的方案进行对比. 由表 1 可以看出,本方案和基于电路的方案支持的查询类型更加丰富.

表 1 方案对比

Table 1 Comparison with prior works

方案	连接查询	多维查询	多维连接查询	函数查询	更新
基于树	×	×	×	×	×
基于签名	×	×	×	×	×
基于电路	✓	✓	✓	✓	×
基于 RAM	✓	✓	✓	✓	✓
本文方案	✓	✓	✓	✓	✓

注:×不支持,✓支持.

在算法效率方面,从初始化、查询与验证、更新及认证跳表这 4 个方面对方案的性能进行了分析,如表 2 所示. 其中,m_i 表示表 i 的列数,n_i 表示表 i 的行数,d 为多维范围查询的维数,R 为查询结果,n 为认证跳表叶子节点数量,u 表示更新的行数,O 表示时间复杂度的上界.

表 2 性能分析

Table 2 Performance analysis

认证跳表	查找		$O(\text{lb}n)$
	范围查询		$O(\text{lb}n)$
	插入		$O(\text{lb}n)$
	删除		$O(\text{lb}n)$
初始化	私钥大小		$O(1)$
	算法复杂性		$O(\sum_i m_i^2 n_i)$
	摘要大小		$O(\sum m_i^2+\sum n_i)$
查询与验证	多维范围查询	证据大小	$O(d\cdot\text{lb}n)$
		验证时间	$O(d\cdot\text{lb}n+\lvert R\rvert)$
	连接查询	证据大小	$O(\lvert R\rvert)$
		验证时间	$O(\lvert R\rvert)$
	求和查询	证据大小	$O(1)$
		验证时间	$O(1)$
	嵌套查询(多维范维+求和)	证据大小	$O(d)$
		验证时间	$O(d+\lvert R\rvert)$
更新操作			$O(m_i^2\text{lb}n)$

5 结 论

1) 提出一个支持全操作的公共可验证外包数据库模型,并给出了模型的安全性定义.

2) 在模型基础上,利用双线性映射累加器和认证跳表实现了一个不仅能同时支持多种 SQL 查询,还支持数据动态更新的公共可验证外包数据库方案,给出了方案中各算法的具体实现.

3) 证明了方案的安全性,其安全性归约至双线性 q-SDH 问题,并分析了方案中各算法的性能.

(下转第 1113 页)