ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, Supplement, December 2009, pp.154−164

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

DAS下一种基于生成检测查询的数据有效性验证方法*

闫巧芝[1,2], 王洁萍[1,2], 杜小勇[1,2+]

[1](教育部数据工程与知识工程重点实验室(中国人民大学),北京 100872)

[2](中国人民大学 信息学院,北京 100872)

Data Integrity Authentication Approach Based on Validating Queries in DAS Paradigm

YAN Qiao-Zhi[1,2], WANG Jie-Ping[1,2], DU Xiao-Yong[1,2+]

[1](Key Laboratory of Data Engineer and Knowledge Engineer (Remin University of China), MOE, Beijing 100872, China)

[2](School of Information, Renmin University of China, Beijing 100872, China)

+ Corresponding author: duyong@ruc.edu.cn

Yan QZ, Wang JP, Du XY. Data integrity authentication approach based on validating queries in DAS paradigm. *Journal of Software*, 2009,20(Suppl.):154−164. http://www.jos.org.cn/1000-9825/09019.htm

Abstract: In database-as-a-service (DAS) paradigm, data owners delegate their data to a third-party: Database service provider (DSP). Compared with traditional DBMS, DAS provides Web-based data access to relieve heavy database management routines. To guarantee quality of database service, most previous work has focus on data privacy and data integrity. This paper focuses on authentication of data integrity. All previous approaches for data integrity authentication require the DSP either to provide extra information or to store extra data. In dynamic scenario, the authentication data should be updated correspondingly, which is inefficient to deploy in real life. This paper proposes a data integrity auditing approach based on validating queries. In this approach, validating queries are generated from previous queries sent by the user. According to the result of validating queries and based on the relationship between previous queries and validating queries, the client can achieve the integrity auditing effectively and efficiently based on the probability. Experimental results confirm the effectiveness of the approach.

Key words: DAS; data security; data integrity; validating query; relationship auditing

摘 要: 在外包数据库服务(database-as-a-service,简称 DAS)中,数据拥有者将数据外包给第三方:服务提供商(database service provider,简称 DSP).与传统的 DBMS 相比,DAS 通过提供基于 Web 的数据访问来节省数据库管理开销.为了保证 DSP 的服务质量,之前大部分工作关注于对数据隐秘性和数据有效性的研究.目前验证数据有效性的方法均要求 DSP 提供额外信息或储存额外数据,且每次更新都需要验证数据做相应调整,这在实际部署中是很低效的.为此提出了一种基于生成检测查询的数据有效性验证方法:通过用户发出过的多个查询生成检测查询,客户端根据检测查询的执行结果,并利用多个查询与检测查询的关系高效、有效地完成基于概率的有效性验证.通过实验验证了该方法的可行性.

关键词: DAS;数据安全;数据有效性;检测查询;关系验证

* Supported by the Key Project of the Ministry of Education of China under Grant No.708004 (国家教育部重点项目); the Project of National Research Foundation for the Doctoral Program of Higher Education of China under Grant No.200800020001 (国家教育部博士点基金)

Received 2009-05-01; Accepted 2009-07-20

对于企业而言,数据库管理是一项关键且繁重的工作,企业需要硬件设备、网络和指定的技术人员来维护自己的数据库系统,为了降低数据维护成本,提高数据管理质量,产生了一种新的数据库管理模式——DAS 模式:在该模式下,企业作为数据拥有者将数据外包给服务提供商(DSP),所有用户的数据访问操作均转向 DSP,由 DSP 为用户提供远程的数据存储、更新和查询服务(如图 1 所示).

Fig.1 DAS system model

图 1 DAS 系统模式

数据安全问题是 DAS 模式下一个不可回避的问题,一方面,外包数据涉及企业的机密信息,外包之前需要将数据经过加密处理,保证数据的隐秘性.另一方面,DSP 可能会对数据进行篡改,为了防止该类行为,用户需要进行数据验证,确保查询结果的有效性.之前关于保证数据隐秘性的工作已较为深入,现在人们开始关注如何保证数据的有效性.

目前的有效性验证方法主要有 3 类:基于数字签名的验证[8]、基于数据结构的验证[6,10]和基于概率的验证[11−13]:这些方法都是通过客户端对DSP返回的额外信息或额外数据的分析来验证查询结果的有效性,它们都要求DSP维护额外的数据结构或额外数据做到:第一,增加了DSP的空间开销和网络传输开销,随着并发处理的增多,DSP的额外负荷也在增加.第二,当用户进行更新操作时,服务端或客户端需要重新调整数据结构或数据以维持验证机制.第三,增加额外数据的方法还会对查询性能产生影响; 为了区分数据,需要元组增加属性列,这虽有助于保证数据的正确性,但也增加了客户端的计算开销和服务端的存储开销; 另外,数据分布会导致某些查询结果中不含额外数据,这样的查询便不能得到有效性验证,而对用户不需验证的查询也会返回额外数据,即用户的验证不是自主的.

在 DAS 应用中,很多企业将数据外包,并通过可信客户端完成加密、解密和过滤数据等工作,也就是说可信客户端需包含一个小的查询执行器.基于该应用和既有方法存在的问题,我们提出了一种基于生成检测查询的数据有效性验证方法.该方法的整体思路是:通过用户发出过的多个查询生成一个检测查询,客户端通过检测查询的执行结果,利用多个查询与检测查询的关系,高效有效的实现基于概率的有效性验证.我们需解决 3 个关键问题:首先:如何保证每条记录的正确性;第二:如何生成有效的检测查询;第三:如何通过生成的检测查询验证多个查询的完整性.本文对这些问题分别进行了解决.同时,对涉及到的检测查询发送形式和时机等问题在文中也进行了说明,最后,从概率上作了安全性说明.我们的贡献总结如下:

(1) 提出了一个新的有效性验证方法,使 DSP 不需要为提供验证做任何额外工作,客户端仅需花费较小的计算开销和可控的传输开销便可完成有效性验证.

(2) 我们对一定形式的查询(查询属性可以是加密属性也可以是公开属性)提供有效性验证,允许用户进行任何更新操作,且客户端与服务器端都不需要为此作任何调整.

(3) 提出了有效的检测查询生成算法,分析了其安全性,通过实验进行了验证.同时关系验证方法,保证了验证执行的有效性.

(4) 用户可以自主选择验证对象,验证时机以及验证频度.

本文第 1 节讨论相关工作.第 2 节对验证方法进行详细介绍.第 3 节对验证机制进行安全性说明.第 4 节通过实验模拟检测查询的生成算法,对实验结果进行分析.第 5 节对工作进行总结,提出后续研究的目标.

1 相关工作

在数据库服务外包中,主要解决的安全问题包括如何保证数据的隐秘性和有效性.针对前者已有很多研究工作,对保证数据有效性的研究也逐渐增多.

数据隐秘性方面的研究:文献[1]分析了使用普通加密方法加密时,加密粒度对系统效率的影响,文献[2]提出了一种使加密数据支持灵活查询的特殊索引方式,文献[3]研究了顺序保持加密带来的系统效率提升.文献[4,5]探讨了攻击者通过挖掘数据关联规则破坏数据隐秘性的问题.

验证数据有效性主要指用户在提交一个查询后,首先检查 DSP 返回的查询结果中每一个元组是否均属于原始数据库(正确性),然后还需要检查服务提供商是否返回了所有的结果(完整性)(也有研究开始关注验证数据是否是最新更新的(freshness)).我们从验证方法上对相关工作作如下介绍:

数据有效性,基于数字签名和数据结构的研究:文献[6]提出了利用Merkle Hash树[7]验证正确性的方法.文献[8]对几种基于数字签名验证数据正确性的方法进行了比较,其中有提到完整性问题,但并未给出完整解决方案.文献[9]采用链式数字签名的方式对简单查询提供完整性验证.文献[6,10]探讨了基于Merkle Hash树验证完整性的方法,由于Merkle Hash树根节点也是采用的数字签名,因此面临和数字签名方法同样的问题:空间、带宽消耗大,以及操作代价和更新维护代价也比较大.

数据有效性,基于概率的研究:Sion[11]提出了Chanlenge Token方法,用来检查DSP是否在整个数据库上完全执行了一组查询,但该方法不能解决恶意DSP执行完查询后,恶意删除查询结果的问题.基于该问题,谢敏和王海讯等人先后提出了不同形式的通过额外数据进行完整性验证的方法.文献[12,13]提出了向DSP插入伪数据的方法:用户通过验证查询结果中是否包含所有已知的伪数据来判断DSP是否对结果进行了恶意删除,同时支持并可以验证数据的更新操作.文献[14]提出了二重加密的方法:将原始数据用第 1 种加密模式加密后,从中抽取部分数据用第 2 种模式加密,将所有加密数据存放到DSP,用户的查询也可以有不同的加密模式,通过对查询结果中二重加密数据的分析,实现完整性验证(该方法需要查询属性都是加密的).

总的来看,目前的完整性验证方法都要求服务提供商维护额外的数据结构或额外数据才能进行验证,这就增加了 DSP 的空间开销和网络传输开销,随着并发处理的增多,DSP 的额外负荷也在增加.并且当用户进行更新操作时,服务端或客户端需要重新调整数据结构或数据来维持验证机制.

我们的有效性验证方法:相较于之前的方法我们方法最明显的优点是:服务提供商不需要为提供验证做任何额外工作且适用于任何数据存储模式;客户端与服务器端都不需要为用户进行的更新操作此做任何调整;客户端只需通过较少的计算开销和可控的传输开销便可实现数据有效性验证,并且用户可以自主选择验证对象,验证时机以及验证频度.但因为我们的方法是基于概率的,因此有一定的失败率,文中进行了安全性分析.(本文主要介绍数据正确性和完整性的验证,对验证数据是否是最新更新的即 DSP 是否即时完成更新操作的问题将在后续工作中介绍).

2 基于生成检测查询的数据有效性验证方法

在介绍我们的验证方法之前.首先讨论 DSP 可能存在的攻击,可将其分为:被动攻击.主动攻击.偷懒攻击.被动攻击指 DSP 试图非法访问未被授权的数据.主动攻击指 DSP 试图对存储数据进行非法的插入、修改和删除操作.偷懒攻击指不完全执行一组查询.只返回部分结果或延迟更新以减少开销.提高性能.防止被动攻击涉及的是数据隐秘性研究.后两种攻击涉及的是数据有效性(正确性和完整性)研究.

2.1 保证数据隐秘性和正确性

DAS情况下,为了防止DSP的被动攻击,数据拥有者可将数据进行元组级加密再存放到DSP[1,2]来保证数据的隐秘性.

为了能进一步防止DSP破解加密方法后对数据进行篡改(指主动攻击的非法插入、修改行为).保证数据的正确性,可通过把数字签名作为属性值添加到元组里的方法[12,13]来实现:给每条元组t增加一个属性值t_{info}:

$t_{info}=H(t,e)$;其中H为One-way哈希函数[15],e为仅数据拥有者和用户才知道的密钥,用户根据返回的元组属性值计算出的t_{info},并将它与元组内所含的t_{info}进行比较,如果不一致则说明存在数据篡改,即服务提供商对原始数据进行了修改或者是插入了伪造数据.

支持加密数据上灵活查询的研究工作[1−5]已比较深入.我们的有效性验证方法是建立在既有研究基础上的.既有研究已能保证数据的隐秘性和正确性,如何验证数据完整性,即如何防止恶意删除(主动攻击中的恶意删除数据的行为)和偷懒攻击,我们在第2.2节进行详细说明.

2.2 数据完整性验证

本文主要针对恶意删除行为来对我们的验证方法进行说明.该方法也支持偷懒攻击中的不完全执行情况:DSP只执行一组查询中的部分查询,造成查询结果集缺失元组(可将缺失元组看作被恶意删除的元组,同样通过本文验证方法检测结果的完整性.由于篇幅原因.本文不再单独详述).另外.如何防止偷懒攻击中的延迟更新将作为我们的后续工作进行研究.

2.2.1 基本假设、定义

假设1. DSP恶意删除的是查询结果集中的元组.不考虑对原始数据的删除.

假设1的依据:如果DSP删除的是原始数据,也就意味着记录的永久性删除,长期以来,数据拥有者或用户总可以通过一定手段发现,对于DSP来说这种被揭穿的风险很大,因此我们不考虑DSP删除原始数据的情况,考虑的是DSP在正确执行完一组查询后.从查询结果集中恶意删除元组的行为(下文中的"攻击"指的是该种行为).

假设2. DSP的恶意删除对象(元组)是随机的.

从DSP实际可操作性上考虑.假设DSP从查询结果集中恶意删除的元组是随机选择的(根据数据和查询的分布,选择删除对象的非随机情况,我们在后续工作中会做进一步研究).

假设3. 与之前的研究[7,8]类似,我们验证的对象是具有统一模式的查询,具体形式为

$$\text{SELECT*FROM } T \text{ WHERE } precicate \tag{1}$$

T为储存在DSP上的任一张提供查询的用户表;$precicate$中每一个子条件均遵循a_i cond v_i的形式.其中a_i表示DSP存储表T的任一提供查询的属性;v_i表示该属性域内任一值;cond表示=,>,<,≥,≤.例:SELECT*FROM T_s WHERE $(1\le a_1\le 3)\vee(a_2=E_k(8))$(其中$a_1$为公开属性,$a_2$为加密属性.

注:允许用户进行任何更新操作和任何形式的查询,但仅对同一表上进行统一模式的查询(1)提供验证.

定义1. 逃逸率:DSP进行了攻击(对查询结果进行了恶意删除)且没被检测到的概率.

2.2.2 问题分析

对同一张表的任意两个简单查询q_1,q_2(满足(1)模式),其结果$\rho(q_1),\rho(q_2)$可能存在的关系为:包含、重合、相交、无交集,如图2所示.

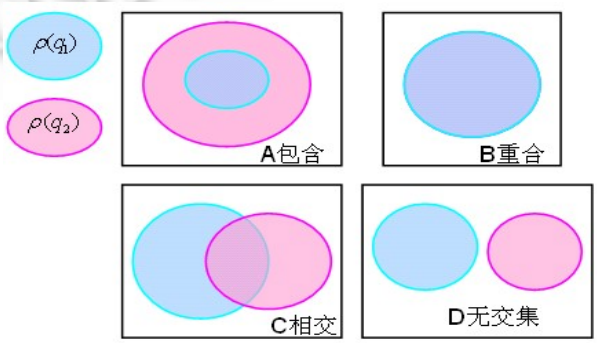


Fig.2 Relationships between results
图2 两结果间关系图

我们的完整性验证思路是:从用户的多个查询Q中生成一个检测查询q,因检测查询的结果生成多个子结果,每个子结果($\rho'(q_i)$)和其对应的验证对象($\rho(q_i)$)应当满足关系R(R为A,B,C,D中某个关系).通过验证两者间是否满足R,来验证Q结果的完整性.因此我们希望生成这样的检测查询:此查询下的R关系有着4种关系中较低的逃逸率和较高的验证效率.因此,我们首先比较A,B,C,D下的验证过程.

假设$\rho(q_i),\rho'(q_i)$中均被恶意删除了元组,且被删除元组数分别为m,n. $\rho'(q_i)$用来验证$\rho(q_i)$的完整,通过比较$\rho(q_i),\rho'(q_i)$是否满足应有关系R(R=〈A,B,C,D〉),来验证是否存在恶意删除行为.就图2中4种关系的逃逸率P_f分别说明如下:

关系 A:如果 $\rho(q_i),\rho'(q_i)$ 满足 A 关系,验证过程设计如图 3 所示.

```
Begin:
1   if(|ρ(q_i)| ≥ |ρ'(q_i)|)
2       不满足 A,存在攻击.终止
3   else
/*------------------------逐条验证------------------------*/
4       H←生成表,包含属性 id 和标记属性 a_mark
将 ρ(q_i) 中各元组 id 插入 H
5       While(逐条扫描 ρ'(q_i) 没有结束)
6               在 H 中查找该元组 id
7               if(存在)
8                   标记 a_mark 为 check
9       While(扫描 H)
10          if(存在未标记 check 的 a_mark)
11              不满足 A.终止
12      满足 A,验证通过.
End
```

Fig.3 Inclusion relation auditing algorithm

图 3 包含关系验证算法

由图 3 的验证过程可知: $\rho(q_i)$ 存在删除元组时, $\rho'(q_i)$ 对 $\rho(q_i)$ 验证失败的情形为 $\rho(q_i)$ 中被删除的元组包含于 $\rho'(q_i)$ 中被删除的元组,则逃逸率

$$P_f=\frac{1}{C^n_{|\rho'(q_i)|}}\times C^{n-m}_{|\rho'(q_i)|-m} \tag{2}$$

关系B:如果 $\rho(q_i),\rho'(q_i)$ 满足关系B,则验证过程如图 4 所示.ε为用户定义的完整可信度,ε_0为常数百分比,用户通过设定ε,可以决定是否终止验证,当用户设定为 $\varepsilon>\varepsilon_0$ 时,发生验证失败的情况为

$$\exists t_1,t_2,\text{满足}\begin{cases}t_1\in\rho(q_i)\wedge t_1\notin\rho'(q_i)\\ t_2\in\rho'(q_i)\wedge t_2\notin\rho(q_i)\end{cases} \tag{3}$$

即两次查询均有被恶意删除了相同数量的元组.当 $\varepsilon<\varepsilon_0$ 时,由图 4 的算法可得: $\rho(q_i)$ 存在删除元组而 $\rho'(q_i)$ 未检测到的情况为 $\rho(q_1)$ 中被删除的元组和 $\rho(q_2)$ 中被删除的元组恰好相同,则

$$P_f=\frac{1}{C^n_{|\rho'(q_i)|}}=\frac{1}{C^m_{|\rho(q_i)|}} \tag{4}$$

关系 C,D:对于关系 D,显然无法通过 $\rho'(q_i)$ 验证 $\rho(q_i)$ 是否完整;而关系 C 的逃逸率显然是高于关系 B 的(推导较简单,篇幅原因,不再细述).

比较 A,B:由 P_f 可知,对于 A 因为 $|\rho'(q_i)|>|\rho(q_i)|$,则 $C^{n-m}_{|\rho'(q_i)|-m}>1$,故 A 的逃逸率高于 B,且对验证过程进行比较,可以看出 A 关系下,计算开销大于 B.因此,我们希望 R 为关系 B.由式(3)可得:

$$P_f=\frac{m(m-1)(m-2)...1}{|\rho(q_i)|(|\rho(q_i)|-1)...(|\rho(q_i)|-m+1)}\le\left(\frac{m}{|\rho(q_i)|}\right)^m \tag{5}$$

当查询结果元组数目较大,删除元组相对较少即: $|\rho(q_i)|>>m$ 时, P_f 趋于 0.

但上述重合关系验证算法(如图 4 所示)如果不能在行–5 终止,就需要逐条验证,这样验证效率是非常低的,因此我们希望生成这样的检测查询:通过该检测查询得到的验证对象之间均满足关系 B,在保证只有极低的逃逸率的条件下,只比较验证对象间元组个数是否相等,便能做出完整性判断.如何生成这种检测查询并完成验证,我们在下一节进行了详细介绍.

```
Begin:
1   if( |ρ(q_i)| ≠ |ρ'(q_i)| )
2       不满足 B.存在攻击.终止
3   else
4       if( ε > ε_0 )
5           满足 B,验证通过.终止
6       else
/*-------------------------逐条验证---------------------------*/
7           H←生成表,包含属性 id 和标记属性 a_mark
将 ρ(q_i) 中各元组 id 插入 H
8           While(逐条扫描 ρ'(q_i) 没有结束)
9                   在 H 中查找该元组 id
10                  if(存在)
11                      标记 a_mark 为 check
12                  else
13                      不满足 B,存在破坏.停止
12          While(扫描 H)
13                  if(存在未标记 check 的 a_mark )
14                  不满足 B.终止
15      满足 B,验证通过.
End
```

Fig.4 Equality relation auditing algorithm

图 4 重合关系验证算法

2.2.3 基于生成检测查询的完整性验证

定义 2. 查询结果大小:查询结果中所含的元组数目.

定义 3. 验证频度:单位时间内生成检测查询的次数.

检测查询是通过用户的多个查询来生成的,这多个查询可以来自用户的不同组查询,这就意味着,用户可以选择验证的对象,验证时机以及验证频度.我们假设验证对象来自同一组查询,对我们的完整性验证方法作如下介绍:

符号说明:

(1) $Q=\langle q_1,...,q_u\rangle$:用户向 DSP 发出的一组查询;

(2) $Q'=\langle s_1,...,s_k\rangle$:从 Q 中抽取出的部分查询;

(3) $C=\langle c(s_1),...,c(s_k)\rangle$:$Q'$ 中各查询对应的查询条件;

(4) q:生成的检测查询;

(5) $c(q)$:检测查询 q 所对应的查询条件;

(6) $\rho(Q)=\langle \rho(q_1),...,\rho(q_u)\rangle$:DSP 返回的 Q 的查询结果集;

(7) $\rho(Q')=\langle \rho(s_1),...,\rho(s_k)\rangle$:从 Q 中抽取出的查询对应的结果集;

(8) $\rho(q)$:DSP 返回的查询 q 的结果;

(9) t:用户自定义阈值,控制检测查询结果大小的上限.

生成检测查询:

检测查询 q 的生成过程可分两步理解.

第 1 步: $Q=\langle q_1,...,q_u\rangle \xrightarrow{\text{St}} Q'=\langle s_1,...,s_k\rangle$.

$$\text{St}=\begin{cases} |\rho(Q')| \le t|\rho(Q)| & (6)\\ |\rho(Q)| = \sum_{i=1}^{u}|\rho(q_i)| & (7)\\ |\rho(Q')| = \sum_{i=1}^{k}|\rho(s_i)| & (8)\\ k\text{尽可能大} & (9)\\ t\text{为用户自定义阈值,且}t\in\left(\frac{1}{u},1\right] & (10)\end{cases}$$

第 2 步: $Q'\to q$.

由 Q' 得各查询对应的查询条件; $C=\langle c(s_1),...,c(s_k)\rangle$ 得:$c= c(s_1)\vee...\vee c(s_k)$.

$$c(q)=Merge(c(s_1)\vee...\vee c(s_k)) \quad (11)$$

进而生成检测查询 q.

生成算法的理论说明:

满足条件 St 的检测查询生成算法设计如图 5 所示.

函数 Generate_query ($\rho(Q)$,t)
输入:用户的一组查询 Q 的返回结果集 $\rho(Q)$;
输入:用户自定义阈值 t.
输出:$c(q)$, $\rho(Q')$.

Begin:
1 $n\leftarrow 0, i\leftarrow 1$
2 求得 $|\rho(q_i)|$ (i=1,…,u)
3. 求得 $|\rho(Q)|$
/*----------生成 1 到 u 的一个随机序列---------*/
4. $Randperm(u)=\langle x1,...,xu\rangle$
/*----------查找 $\langle s_1,...,s_k\rangle$ 中 s_1 ----------*/
5 while($i\le u$)
6 if($|\rho(q_i)|\ge t|\rho(Q)|$)
7 i++
8 $Q'\leftarrow q_{xi}$
9 $c=c(q_{x1})$
10 $n=|\rho(q_{xi})|$
/*----------查找 $\langle s_2,...,s_k\rangle$ ----------*/
11 i++
12 while($i\le u$)
13 if($(n+|\rho(q_{xi})|)\ge t|\rho(Q)|$)
14 终止
15 $Q'\leftarrow q_{xi}$
16 n+=$|\rho(q_{xi})|$
17 $c=c\vee c(q_{xi})$
18 $c(q)$ = $Merge(c)$
End

Fig.5 Generating validating query algorithm

图 5 检测查询生成算法

这样做是考虑到如下几点:

(1) 传输和查询开销角度:通过一个检测查询 q 验证多个查询的完整性,减小了验证开销;用户通过自定义

阈值 t 可以调节检测查询结果的大小,控制传输和查询开销.

(2) 安全角度:q 由 Q 中抽取的查询生成,而抽取是随机的(通过图 5 中的第 5 行:Randperm()生成随机序列).随机抽取的查询集 Q'跟原查询序列无特定关系,从而防止了泄密.例如:假设将原查询集按结果大小排列,Q'由大到小顺序选择部分查询,那么 DSP 在长期执行过程中,就会探测到检测查询的生成跟 Q 的特定关系,进而只攻击那些结果小的查询,则验证时攻击将难以被探测到.

检测查询的执行时机:

检测查询 q 是由 Q 中某些查询生成的,查询条件上存在一定关系,为防止 DSP 探测出 q 为检测查询,用户可根据实际情况自主决定验证时间,可将 q 和其他查询一起作为新一组查询 Q_t 发送到 DSP,q 的返回结果集在对 Q 验证的同时,也能一定程度上验证 Q_t 的执行是否存在攻击,例如:DSP 正确执行了 Q,但对 Q_t 中的查询进行了攻击,如果 q 属于被攻击查询,则进行验证时会被探测到.因此,保持 q 的隐秘性有助于双向验证.

另外,因为 DSP 的破坏是随机的,即使探测出 q 为检测查询,对验证机制的影响并不大.因为 DSP 要使对 q 的攻击对应于之前对 Q 的攻击是难以操作的(第 4 节进行了说明).但从双向验证角度考虑,也为了最大限度的降低逃逸率,我们希望保持 q 的隐秘性,因此我们通过 *Merge*()规则(式(8),图 5 中的第 18 行)对 q 的查询条件进行了合并.

Merge()规则

定义动机:除了减少开销上的原因外,参与生成 q 的查询 $\langle s_1,...,s_k\rangle$ 的查询条件取'或'操作后,可能存在 q 身份的泄密.例如:假设表 T 上提供查询属性 a; $c(s_1)$ 表示 a=2, $c(s_2)$ 表示 a=2$\vee b$<5;则生成条件 $C=(a=2)\vee(a=2)\vee(b<5)...$,则由于 a=2 出现了两次,即使查询经过了加密,加密后也会出现两次,DSP 可能从查询条件上分辨出 q 为检测查询,破坏 q 的隐秘性.

定义规则:对 C 的各只含$\vee$操作的 $c(s_i)$ 中每个 a_i cond v_i (式(1)),查找同种属性,对每种属性均作如下处理:

(1) a cond v 出现多次,则合并为 1 次.

(2) a cond v_1 和 a cond v_2 同时出现,且 $v_2>v_1$,

(i) cond 为>或≥,则合并为 a cond v_2; (ii) cond 为<或≤,则合并为 a cond v_1;

(3) a cond$_1$ v_1 和 a cond$_2$ v_2 同时出现且 v_2=v_1,

(i) cond$_1$为>且cond$_2$为<;(ii) cond$_1$为<且cond$_2$为>,两种情况均合并为 a!=v_1;

(4) a cond$_1$ v_1 和 a cond$_2$ v_2 同时出现,$v_2\geq v_1$,cond$_1$为>或≥且cond$_2$为<或≤(情形 3 除外),则删除所有 a cond v.

(5) a=v_1 和 a cond$_2$ v_2 同时出现,(i) cond$_2$>或≥且$v_2<v_1$,(ii) cond$_2$为<或≤且$v_2>v_1$,两种情况均合并为 a cond$_2$ v_2.

(6) v_1 cond$_l$ a cond$_r$ v_2 和 v_3 cond$_l$ a cond$_r$ v_4 同时出现,cond$_l$ 为 <或≤, cond$_r$ 为 >或≥.$[v_1,v_2]$,$[v_3,v_4]$ 两区间有交集,则对 v_1 cond$_l$ a 和 v_3 cond$_l$ a 用规则 2.(i) 合并;对 a cond$_r$ v_2 和 a cond$_r$ v_4 用规则 2.(ii)合并.

验证过程:

根据第 2.2.2 节问题分析中的结论,我们通过验证对象间所满足的关系进行验证:Q' 中被抽取的查询序列记为 $\langle s_1,...,s_k\rangle$,其对应的结果集为 $\rho(Q')=\langle\rho(s_1),...,\rho(s_k)\rangle$,因为 $c(q)=Merge(c(s_1)\vee...\vee c(s_k))$ 则 $\rho(q)$ 与 $\rho(Q')$ 非严格满足关系 B,因为 $\langle\rho(s_1),...,\rho(s_k)\rangle$ 结果集间可能存在相同元组,故 $|\rho(q)|\leq|\rho(Q')|=\sum_{i=1}^{k}|\rho(q_i')|$ 客户端利用小的查询执行器,将 $\langle s_1,...,s_k\rangle$ 在 $\rho(q)$ 上执行,得到 $\langle\rho'(s_1),...,\rho'(s_k)\rangle$,则 $\rho'(s_i)$ 与 $\rho(s_i)$ 严格满足关系 B.

用户设定ε使 $\varepsilon>\varepsilon_0$,对每对<$\rho'(s_i)$, $\rho(s_i)$>执行图 4 算法(行 1~5),只需验证每对结果集大小是否都相等便可验证完整性.这样做只需花费较小的计算开销,提高了验证效率并且是安全的.如果个别用户想最大限度的验证结果集的完整性,则可通过调整ε值,使得 $\varepsilon<\varepsilon_0$,对<$\rho'(s_i)$, $\rho(s_i)$>执行图 4 算法.完整性验证方法的安全性在第 4 节进行了说明.

生成算法的有效性分析:

定义 4. 无效检测查询:如果 Q 被攻击,但被攻击的查询均未被 Q' 抽取,则生成的为无效检测查询.

为了说明检测查询生成算法的有效性,我们从概率的角度进行说明.由假设 2 我们知道 DSP 的恶意删除对象是随机的,则从长时间和整体上看,可做如下假设:

(1) 假设查询对象表 T 上提供的查询属性的数值满足均匀分布;

(2) Q 为一组等宽查询,Q 执行后的结果集大小记为 N,即 $|\rho(Q)|=N$,被删除元组数记为 δN (δ为由 DSP 的攻击程度决定的比例),被删元组在均匀分布,则 $\delta N \ge u$ 时,认为每个查询均存在被删除元组,即各查询均被攻击. $\delta N < u$ 时,被攻击查询个数为 δN. 生成无效检测查询的概率记为 P_0,则:

① 当 $\delta N \ge u$ 时, $P_0 = 0$;

② 当 $\delta N < u$, $k > (u - \delta N)$ 时, $P_0 = 0$;

③ 当 $\delta N < u$, $k \le (u - \delta N)$ 时,

$$P_0 = \frac{C_{u-\delta N}^{k}}{C_u^k} = \frac{(u-\delta N)(u-\delta N-1)...(u-\delta N-k+1)}{u(u-1)...(u-k+1)} \le \left(\frac{u-\delta N}{u}\right)^k.$$

由 $|\rho(Q')| = \sum_{i=1}^{k}|\rho(s_i)| \le tN$ (等宽查询时 $k|\rho(s_i)| \le tN$),并且 k 为最大值(式(6)~式(9))可知:t 越大,k 越大,因此,通过调节 t 的大小可减小 P_0,且当 $k > (u - \delta N)$ 时 $P_0 = 0$.

通过以上分析可知:我们的检测查询生成算法生成无效查询的几率是很低,并且是可以通过用户调节保证检测查询生成的有效性.我们通过实验模拟实现了该算法,并对结果进行了分析(见第 5 节).

3 安全性证明

我们通过基于概率的分析,来证明验证方法的安全性:即在生成了有效检测查询的情况下,根据查询结果($\rho(q)$),应用我们的验证方法,可以对验证对象($\rho(Q)$)的完整性做出正确判断.

首先,假设 Q 被攻击,并生成了有效的检测查询 q,为验证完整性,将 q 同其他查询一起发到 DSP,记为 $Q_t = \langle q, q_2', ..., q_u' \rangle$,则用我们的验证方法,攻击成功时需满足以下条件:

(1) Q_t 被攻击且 q 被攻击;

(2) 设置 $\varepsilon > \varepsilon_0$ 时,$\langle s_1, ..., s_k \rangle$ 中每个 $\rho(s_i)$ 被删除的元组数目与各查询在 $\rho(q)$ 上执行后对应的 $\rho'(s_i)$ 缺少的元组数目相同(参见第 2.2.2 节中式(2)的情形),此情况概率记为 p_3 (注:删除的元组数目可能为 0,即对于没被攻击的查询 $\rho'(s_i)$ 应为完整的结果).在设置 $\varepsilon < \varepsilon_0$ 时,每个 $\rho(s_i)$ 缺少的元组(即被恶意删除的元组),与执行 q 时被恶意删除的元组完全相同,此情况概率记为 p_4.

假设任意一组查询被攻击的概率为 p_1,且其中单个查询被攻击的概率设为 p_2,则满足条件 1 时的概率为 p_1 p_2 (p_1, p_2 均由 DSP 的攻击行为决定). $p_3 = \sum_{i=1}^{k} p(|\rho'(s_i)| = |\rho(s_i)|)$ ($\rho'(s_i)$ 为 s_i 在 $\rho(q)$ 上执行后的结果集).当 $\varepsilon > \varepsilon_0$ 时,逃逸率 $P = p_1 p_2 p_3$;

由于对每组查询来说,删除是随机的,对 DSP 来说,使 $\langle s_1, ..., s_k \rangle$ 中每个查询都满足 $|\rho'(s_i)| = |\rho(s_i)|$ 是难以操作的, p_3 值越小,P 越小至趋于 0.即便如此,如果用户认为 P 不在安全范围内,则可调整ε使 $\varepsilon < \varepsilon_0$,进一步降低逃逸率:由第 2.2.2 节中的分析及式(3)、式(4)可知 $p_4 = \frac{1}{C_{|\rho(q)|}^{m}} \le \left(\frac{m}{|\rho(q)|}\right)^m$ (m 为删除元组数);当 $|\rho(q)| >> m$ 时, p_4 趋于 0,由 $P = p_1 p_2 p_4$ 可知,P 趋于 0.

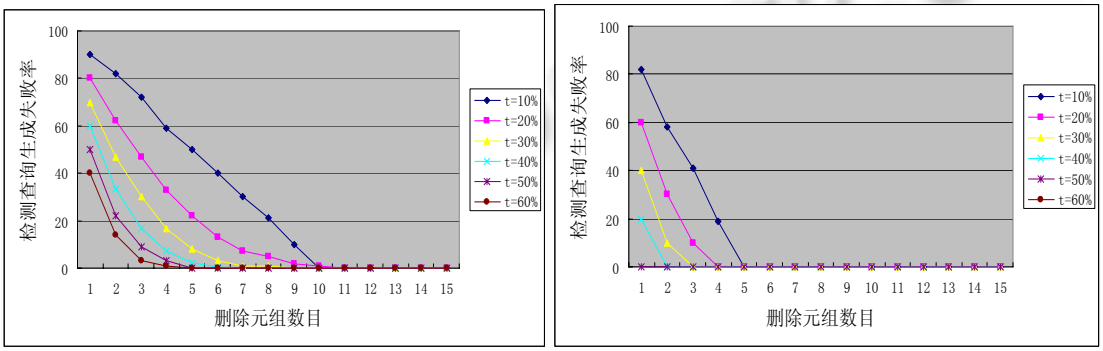
综上所述,可以得到如下结论:如果 DSP 对一组查询进行了攻击,且用户生成了有效检测查询,则通过我们的验证方法便可做出正确的完整性判断.

4 实 验

我们模拟实现了检测查询的生成算法.首先,随机生成 10 000 条均匀分布的数据,设置 4 组等宽查询,分别对查询个数为 10 和 5 的情况下,查询条件重叠和无重叠的情况进行了实验.因为生成有效查询后,通过我们的验证方法可以正确判断是否存在攻击(第 3 节安全性说明),因此检测查询生成失败率一定程度上代表了逃逸率.

实验过程:对每组查询结果集删除不同数目的元组,对每个删除数目,执行 100 次随机删除,对每次删除进行以下过程:重复生成检测查询 100 次,计算每次被抽取的查询中不含被攻击查询的次数 n_i .对每个删除数目,求 $\frac{1}{100}\sum_{i=1}^{100} n_i$.

实验结果:如图 6 所示.当 u=10 时,对于查询条件重叠和不重叠的两组查询,实验结果基本相同,都满足图 6(a)中的分布.同样,当 u=5 时,查询条件是否重叠对分布无影响,满足图 6(b)的分布.



(a) u=10 (b) u=5

Fig.6 Comparisons on the proportion of invalid validating query under different number of queries

图 6 不同查询数下无效检测查询比例

实验分析:对实验结果分析可知:

(1) 随着删除元组的增多,生成无效检测查询的次数逐渐减少,删除元组数多于查询个数时无效查询生成次数趋于 0.

(2) 删除元组较少时,t 越大,无效的查询验证越少.因此用户通过调整 t 值,可有效减少无效检测查询的生成.从而降低逃逸率,进一步验证了第 2.2.3 节中对生成算法的有效性分析.

(3) 用户发出的一组查询中的查询个数对检测查询的生成是有影响的,查询个数越少,删除同样多的少量元组时,生成无效的检测查询次数越少.因此,用户在选择验证对象时,验证对象所含查询个数也是生成有效的检测查询的考虑因素.

(4) 查询条件是否有重叠与生成有效检测查询并无影响,因为 DSP 是在结果集上进行的随机删除,DSP 并不会进行区分不同查询的结果集上元组是否相同,,因此不会影响 DSP 的攻击对象分布.

5 结束语

对于我们来说,提出一种新的完整性验证方法是一种尝试,还存在很多可以改进的空间,今后我们要进一步研究的问题包括:数据分布不均匀时,如满足正态分布时,且 DSP 非随机的对数据进行恶意删除时,如何保证检测查询的有效性;如何更好的降低检测查询的带宽消耗,减少客户端计算开销的问题.

另外,我们的验证方法允许有更新权限的用户进行任何更新操作,但如何验证 DSP 是否正确执行了更新操作,数据是否为最新更新的问题将是我们的后续研究工作.

致谢 在此,我们向对本文的工作给予支持和建议的老师和同学表示感谢.

References:

[1] Hacigumus H, Mehrotra S, Iyer BR. Providing database as a service. In: Agrawal R, Dittrich K, Ngu AHH, eds. Proc. of the 18th Int'l Conf. on Data Engineering (ICDE). Washington: IEEE Computer Society, 2002.

[2] Hacigumus H, Iyer BR, Li C, Mehrotra S. Executing SQL over encrypted data in the database-service-provider model. In: Michael JF, Bongki M, Anastassia A, eds. Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD). New York: ACM, 2002. 216−227.

[3] Agrawal R, Kiernan J, Srikant R, Xu Y. Order-Preserving encryption for numeric data. In: Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD). New York: ACM, 2004. 563−574.

[4] Agrawal D, Aggarwal CC. On the design and quantification of privacy preserving data mining algorithms. In: Proc. of the 20th ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems (PODS). New York: ACM, 2001. 247−255.

[5] Agrawal R, Srikant R. Privacy-Preserving data mining. In: Proc. of the 2000 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD). New York: ACM, 2000. 439−450.

[6] Devanbu PT, Gertz M, Martel C U, Stubblebine SG. Authentic third-party data publication. In: Thuraisingham BM, van de Riet RP, Dittrich KR, Tari Z, eds. Proc. of the 14th Annual IFIP WG 11.3 Working Conf. on Database Security. Deventer: Kluwer Academic Publisher, 2000. 101−112.

[7] Merkle RC. A certified digital signature. In: Brassard G, ed. Proc. of the Crypto'89. LNCS 435, London: Springer-Verlag, 1990. 218−238.

[8] Mykletun E, Narasimha M, Tsudik G. Authentication and integrity in outsourced databases. In: Network and Distributed System Security Symp. (NDSS 2004). Internet Society Press, 2004.

[9] Pang H, Jain A, Ramamritham K, Tan KL. Verifying completeness of relational query results in data publishing. In: Proc. of the 2005 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD). New York: ACM, 2000. 407−418.

[10] Li FF, Hadjieleftheriou M, Kollios G, Reyzin L. Dynamic authenticated index structures for outsourced databases. In: Fatma Ö, ed. Proc. of the 2006 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD). New York: ACM, 2006. 121−132.

[11] Sion R. Query execution assurance for outsourced databases. In: Klemens B, Christian SJ, Laura MH, Martin LK, Per-Ake L, Beng CO, eds. Proc. of the 31st Int'l Conf. on Very Large Data Bases (VLDB). New York: ACM, 2005. 601−612.

[12] Xie M, Wang HX, Yin J, Meng XF. Integrity auditing of outsourced data. In: Proc. of the 33rd Int'l Conf. on Very Large Data Bases (VLDB). Vienna: VLDB Endowment, 2007. 782−793.

[13] Xie M, Wang HX, Yin J, Meng XF. Providing freshness guarantees for outsourced database. In: Proc. of the 11th Int'l Conf. on Extending Database Technology: Advances in Database Technology (EDBT). New York: ACM, 2008. 323−332.

[14] Wang HX, Yin J, Perng CS, Yu PS. Dual encryption for query integrity assurance. In: Proc. of the 17th ACM Conf. on Information and Knowledge Management (CIKM). New York: ACM, 2008. 863−872.

[15] Bakhtiari S, Safavi-Naini R, Pieprzyk J. Cryptographic hash functions: A survey. Technical Report, 95-02, Department of Computer Science, University of Wollongong, 1995.

闫巧芝(1983－),女,河北石家庄人,硕士生,主要研究领域为高性能数据库,数据库安全.

王洁萍(1976－),女,博士生,主要研究领域为高性能数据库,数据库安全.

杜小勇(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为高性能数据库,智能信息检索,知识工程.