基于可搜索加密机制的数据库加密方案

孙僖泽 周福才 李宇溪 张宗烨

(东北大学软件学院 沈阳 110169)

摘 要 近年来,数据外包的日益普及引发了数据泄露的问题,云服务器要确保存储的数据具有足够的安全性.为了解决这一问题,亟需设计一套高效可行的数据库加密方案.可搜索加密技术可较好地解决面向非结构文件的查询加密问题,但是仍未较好地应用在数据库中.因此,针对上述问题,提出基于可搜索加密机制的数据库加密方案.本文贡献如下:第一,构造完整的密态数据库查询框架,保证了数据的安全性且支持在加密的数据库上进行高效的查询;第二,提出了满足 IND-CKA1 安全的数据库加密方案,在支持多种查询语句的前提下,保证数据不会被泄露,同时在查询期间不会降低数据库中的密文的安全性;第三,本方案具有可移植性,可以适配目前主流的数据库,如 MySQL、PostgreSQL 等.本文基于可搜索加密方案中安全索引的构建思想,利用非确定性加密方案和保序加密方案构建密态数据库安全索引结构,利用同态加密以及 AES-CBC 密码技术对数据库中的数据进行加密,实现丰富的 SQL 查询,包括等值查询、布尔查询、聚合查询、范围查询以及排序查询等.本方案较 BlindSeer 在功能性方面增加了聚合查询的支持,本方案改善了 CryptDB 方案执行完成 SQL 查询后产生相等性泄露和顺序泄露的安全性问题,既保证了数据库中密文的安全性,又保证了系统的可用性.最后,我们使用一个有 10000 条记录的 Student 表进行实验,验证了方案框架以及算法的有效性.同时,将本方案与同类方案进行功能和安全性比较,结果表明本方案在安全性和功能性之间取得了很好的平衡.

关键词 密态数据库;可搜索加密;同态加密;AES 加密;SQL 查询

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2021.00806

A Database Encryption Scheme Based on Searchable Encryption

SUN Xi-Ze ZHOU Fu-Cai Li Yu-Xi ZHANG Zong-Ye

(*Software College*, *Northeastern University*, *Shenyang* 110169)

Abstract In recent years, the increasing popularity of outsourcing data to cloud server has led to data leakage problems, we need to ensure that the data stored in cloud server is sufficiently secure. It is necessary to design efficient and feasible database encryption schemes to solve this problem. Searchable encryption can make encrypted data searchable while solve data leakage problem for non-structural files, but it is still not well applied in the database. Therefore, in this paper, aiming at the problem that the data in the database server is leaked, we designed a database encryption framework based on searchable encryption. The novelty of this work comes with three contributions. First, we construct a well-defined encrypted database query framework, which not only ensure the security of data, but also make the encrypted query efficient. Second, Our scheme is secure under IND-CKA1(semantic security against adaptive chosen keyword attack), it ensures that the data is not compromised and that the security of the ciphertext in the

收稿日期:2019-10-29;在线发布日期:2020-06-30.本课题得到国家自然科学基金(62072090,61872069)、中央高校基本科研业务费项目-重点科学研究引导项目(N2017012)资助.孙僖泽,硕士,主要研究领域为可搜索加密、密态数据库.E-mail:zecomputer@163.com.**周福才**(通信作者),博士,教授,中国计算机学会(CCF)会员,主要研究领域为密码学、网络安全、可信计算.E-mail:fczhou@mail.neu.edu.cn.**李宇溪**,博士,主要研究领域为可搜索加密、云安全.**张宗烨**,博士,主要研究领域为可搜索加密.

database is not compromised during the query. Third, our framework achieves high portability and is suitable for many mainstream databases such as MySQL, PostgreSQL and so on. Based on the idea of constructing secure index in searchable encryption scheme, We use cryptographic techniques such as homomorphic encryption and AES-CBC to encrypt database. Our scheme implements rich SQL queries, including equivalent query, Boolean query, aggregated query, range queries, sort query and so on. Our scheme adds support for aggregated queries compared to BlindSeer in functionality. Compared to CryptDB, our scheme does not reveal the equality and the order of the ciphertext, which not only ensures the security of the ciphertext in the database, but also ensures the availability of the system. Finally, we use a student table which has 10 000 records to evaluate our scheme, and the results show that the proposed framework and algorithm are effective. At the same time, we compare the functionality and security of our scheme with similar schemes, and our scheme achieves a good balance between security and functionality.

Keywords　encrypted database; searchable symmetric encryption; homomorphic encryption; AES encryption; SQL query

1　引　言

随着数据量的增加，人们越来越倾向于选择将自有数据依托于第三方数据库服务提供商进行存储.目前将数据存储在外包数据库服务器上已成为一种常见的方案，常见的如华为、京东、阿里巴巴、亚马逊以及一些医院等机构将一些主要的信息存储在外包数据库服务器上.随着外包数据库服务器中存储信息量的增多，外包数据库服务器上的信息泄露(需要受保护的信息或隐私被泄露)引发了广泛关注.外包数据库服务器方面常常出现泄露或篡改等安全问题，因为潜在的恶意外包数据库服务器可能会尝试从他们存储的数据及其处理的查询中学习信息，甚至将其泄露给某些未经授权方[1].一种行之有效的方法是在将数据存储到外包数据库服务器中之前对数据进行加密.但是，这种需求是以功能为代价，一旦数据被加密，在查询时需要先解密数据，这样导致搜索变得困难. 因此，安全研究人员已转向研究既保护数据库内容又支持高效操作(如搜索)的方案，而不是仅仅加密数据.为了解决云服务器中的文档信息泄露的问题，2000 年 Song 等人提出了可搜索加密方案，该方案通过输入单个关键字在云服务器上对加密文件集进行搜索[2]；Curtmola 等人在 2006 年提出了更高效更安全的方案，且该方案实现了多用户的 SSE[3]；Kamara 等人在 2012 年提出的方案中扩展了倒排索引方法，在保证安全性的前提下满足了次线性搜索效率并可以有效添加和删除文件[4]；2014 年，Cash 等人提出了大数据集下的动态可搜索加密方案[5].上述这些方案都只局限于文件集的搜索，目前更多的企业、政府会把信息存储在数据库中，因此如何实现密态数据库的密文查询问题，具有重要现实意义和实用价值.

为了解决数据库中数据的安全问题，2004 年，Hore 等人提出了一种基于数据库关系表构建的安全索引实现模糊范围查询[6].美国乔治亚理工学院的 Amanatidis 等人于 2007 年针对外包数据库的安全性研究提出了基于分组密码，对称加密方案和消息认证码等标准密码原语的加密模型，该方案能进行简单的密文搜索[7]. 2011 年 Popa 等人提出了 CryptDB，为了支持更多的查询，设计了洋葱加密模型，即一个数据通过多种加密方案进行嵌套加密. CryptDB 针对字符类型列将其扩展为四列密态属性列，分别是初始向量列、通过 EQ 洋葱模型加密的列、ORD 洋葱模型加密的列、Search 洋葱模型加密的列，针对数值类型的就不会生成由 Search 洋葱模型加密的列，但是为了进行 SUM 操作生成了通过 HOM 洋葱模型加密的列[1,8-11].然而 CryptDB 存在两个问题，其一是将明文数据库转换为密文数据库时有些密文列占用了更多存储的空间，这样无疑在查询时增加了磁盘读写开销，从而影响了系统的性能；其二是是通过洋葱模型对每个列进行加密，导致每剥掉一层洋葱加密层，它的安全性都会降低，尤其做等值查询时，会将加密方案降低到 DET 方案，该

方案是确定性加密方案，极易泄露明文之间的信息，因此 CryptDB 的安全性一直饱受争议. 2014 年，Pappas 等人提出了 BlindSeer 数据库加密方案，该方案通过 Bloom filter、Yao 混淆电路、BF 搜索树技术实现了支持等值查询、布尔查询、范围查询的密态数据库系统，该方案的安全性较高，但是在查询过程中存在一定的误报并且支持的查询类型不够丰富[12]. 2016 年，Poddar 等人提出了 Arx 系统，该方案是通过 Yao 混淆电路和 Arx-RANGE 索引结构、Arx-EQ 索引结构实现的，但是该方案适用于非关系型数据库[13]. 2017 年 Monir 等人提出一种通过索引搜索的数据库加密方案，该方案仅支持等值查询、布尔查询和范围查询，不能支持其他 SQL (Structured Query Language)语句查询，同时其范围查询的效率非常低[14].

针对外包数据库中的数据泄露问题，以及现有的数据库加密方案不能将方案的安全性与功能性兼顾的问题，本文提出基于可搜索加密机制的数据库加密方案(DataBase encryption scheme based on Searchable Encryption，DB_SE)，相比于其他方案，本方案既实现了数据的安全性，又能满足现实场景中功能性的需求，主要贡献如下：

(1) 提出基于可搜索加密机制的密态数据库查询框架，该框架有效地保证了在外包数据库服务器中存储的数据的安全性，并可以在加密的数据库上进行高效地查询.

(2) 基于提出的框架，设计了一个满足 IND-CKA1[3,15](选择关键词攻击下的索引不可区分性)安全的数据库加密方案，通过语义安全的加密方案对数据进行加密，不会造成信息泄露问题.

(3) 方案支持丰富的 SQL 查询，包括等值查询、布尔查询、比较查询、聚合查询、范围查询、排序查询等多种复杂查询. 其中，本方案通过构建保序安全索引进行范围查询，其范围查询的效率优于同类主流方案.

(4) 本方案的框架具有可移植性，可适配于 MySQL、PostgreSQL 等主流数据库，支持透明的 SQL 查询，即在不需要更改 SQL 语义的情况下进行正常地查询，保证了本方案的可移植性.

本文第 2 节介绍关于可搜索加密技术、哈希函数以及 OPES 加密算法；在第 3 节介绍本方案的框架、形式化定义、关键算法等内容；第 4 节对本方案进行安全性分析；第 5 节从功能性和安全性方面与两种主流的数据库加密方案进行对比，并通过实验进行性能测试；第 6 节总结本文所述的工作，并对未来研究作出展望.

2 预备知识

2.1 可搜索加密技术

可搜索加密最早是由 Song 等人[2]提出的，该技术是一种加密搜索模式，使得在对加密数据进行搜索的过程中，不会对恶意服务器泄露敏感信息. 可搜索的对称加密(SSE)允许用户以私密方式将其数据存储在外包云(服务器)上，同时保持有搜索它的能力. 可搜索加密技术包含用户和云服务器两个实体：用户对文件进行加密处理生成密文，同时生成文件索引，再对文件索引处理生成安全索引. 将密文与安全索引上传到云服务器存储. 在进行更新时，用户使用私钥与更新内容生成更新索引，并发送至服务器，服务器根据更新索引进行计算，完成更新操作. 在进行搜索时，用户使用私钥与搜索关键字生成搜索令牌，将其发送至云服务器，云服务器根据构建的搜索令牌在加密的文件中进行搜索，将查询出的密文结果进行解密，最后将明文结果返回给客户端.

对称可搜索加密方案由 Curtmola 等人[3]于 2006 年提出，该方案基于索引构建，方案包括了五个函数：Gen、Enc、$Trapdoor$、$Search$、Dec.

$sk \leftarrow Gen(1^k)$为概率性算法，用于生成密钥，由用户执行. 输入安全参数 k，输出私钥 sk.

$(I,c) \leftarrow Enc(sk,D)$为概率性算法，用于构建安全索引和加密文档集合，由用户执行. 输入私钥 K 和文档集合 $Doc=(Doc_1,\cdots,Doc_n)$，输出安全索引 I 和加密文档集合 $EncDoc=(EncDoc_1,\cdots,EncDoc_n)$.

$t \leftarrow Trapdoor(sk,w)$为确定性算法，用于生成关键字的陷门，由用户执行. 输入私钥 sk 和关键字 w，输出陷门 t.

$X \leftarrow Search(I,t)$为确定性算法，用于搜索包含关键字 w 的文档，运行于服务端. 输入安全索引 I 和陷门 t，输出加密文档集合 X.

$Doc_i \leftarrow Dec(sk,EncDoc_i)$为确定性算法，用于解密密文文档，由用户执行. 输入私钥 sk 和加密文档 $EncDoc_i$，输出文档 Doc_i.

2.2 哈希函数

哈希函数也被称为散列函数，可以将任意长度

的数据输出为固定长度哈希值.理想的哈希函数具有单向性、抗碰撞性和映射分布均匀性等特性.可应用于消息认证、数字签名、文件校验等方面.

哈希函数族可表示为多项式时间计算映射 $H:\mathcal{K}\times\mathcal{D}\to\mathcal{R}$，$H_K(\cdot)$表示哈希函数关于 K 的运算.对于一个好的哈希函数来说，攻击者很难在 $\mathcal{D}$ 中找到两个不同元素，使它们能够计算出 $\mathcal{R}$ 中的相同元素，这一属性被称为抵抗碰撞攻击.

使用 $H:\mathcal{K}\times\mathcal{D}\to\mathcal{R}$ 表示哈希函数族，敌手 A 找到两个不同元素使计算出相同结果的优势表示为

$$Adv_{H,A}^{\mathrm{col}}(\lambda)=\mathbb{P}\left[K\xleftarrow{\$}\mathcal{K},(M,M')\leftarrow A(K):M\neq M'\wedge H_K(M)=H_K(M')\right].$$

如果对于任何多项式时间的敌手 A 来说，$Adv_{H,A}^{\mathrm{col}}(\lambda)$是可忽略的，则 H 是能够抵抗碰撞攻击的哈希函数族.

随机预言机(RO)一般由哈希函数来实例化.但是许多哈希函数具有的属性使其不适合直接用作随机预言机，比如：当消息和密钥长度已知时，采取 H(密钥‖消息)方式构造的哈希函数易被进行长度扩展攻击.密钥散列消息认证码(HMAC)未采用该构造方式，故可以抵抗该攻击.因此，本文使用带有公钥的 HMAC 构造来实例化 RO，对于哈希函数来说，HMAC 的定义为 $\mathrm{HMAC}(K,x)=H((K\oplus opad)\|H(K\oplus ipad)\|x)$，其中，$opad$ 和 $ipad$ 是两个常数，$\oplus$是异或操作.

2.3　OPES 加密

OPES(Order Preserving Encryption Scheme)加密算法是一种保序加密算法，该算法可以使密文仍然保留对应的明文的顺序信息.传统的加密方案为了保护数据的机密性需要精确匹配才能查询，在实际需求中针对数值类型的数据经常会使用比较查询操作，但使用传统的加密算法不能容易地执行比较查询操作，OPES 加密算法是为了解决这一问题被设计的，OPES 算法在保证数据机密性的同时具有较好的查询功能[16].

OPES 加密算法允许在不对密文进行解密的情况下直接对加密数据进行比较操作，因此对于密文的比较查询、排序操作以及求最大最小值操作使用 OPES 加密算法非常适合.同时 OPES 还具有以下特性：

(1)对使用 OPES 加密的数据进行查询处理的结果是准确的.既不存在任何误报，也不遗漏任何答案元组.不会产生的查询结果的超集，因此省去了复杂的过滤步骤.

(2)OPES 可以很好地处理更新操作.在修改列中的值或添加列中的值的时候不需要更改其他已存在的密文值.

(3)OPES 可以很容易地与现有的数据库系统集成，因为它已经被设计为与现有的索引结构(如 B 树)一起工作.数据库被加密的事实可以对应用程序透明.

(4)OPES 的时间和空间开销对于在实际系统中部署 OPES 是合理的.

入侵者可以访问通过 OPES 算法加密的数据库，但没有数据库中值的分布之类的先验信息，并且不能对他选择的任意值进行加密或解密.在这样的环境中，OPES 对于能够获得加密值的严格估计的对手是稳健的.最终在 DB2 上的实现的测量结果表明，OPES 在查询处理上的性能开销很小，对于在生产环境中部署 OPES 是合理的.

3　DB_SE 方案

3.1　方案框架

本文采用满足 IND-CPA(自适应选择明文攻击)安全的加密方案加密数据表，结合可搜索加密思想构建安全索引，在保证数据表机密性的同时，实现功能性查询.本文将加密的数据表和基于数据表构造的安全索引结构独立存放在数据库服务器中；引入一个代理服务器，负责构建加密索引、SQL 解析与重写、加解密操作；数据库服务器主要负责将代理服务器生成的加密的 SQL 语句在安全索引中查询、更新操作.本方案框架如图 1 所示.

本方案的系统框架由三种实体组成，分别是客户端、代理服务器端、托管敏感数据的不受信任的数据库服务器端.本框架分为两个区域，分别是可信区域(Trusted Zone)和不可信区域(Untrusted Zone).

可信区域包括用户的客户端和代理服务器端 Proxy 两个实体.客户端负责进行正常的 SQL 查询；代理服务器端在密态数据库生成阶段负责生成密钥、根据策略表对原始数据表进行表结构重构、按照策略表的加密方案对原数据表进行加密并生成每张表对应的安全索引结构；Proxy 在查询阶段负责截取由客户端发送的 SQL 请求，结合可搜索加密思想提取 SQL 中的关键词并生成搜索陷门，将 SQL

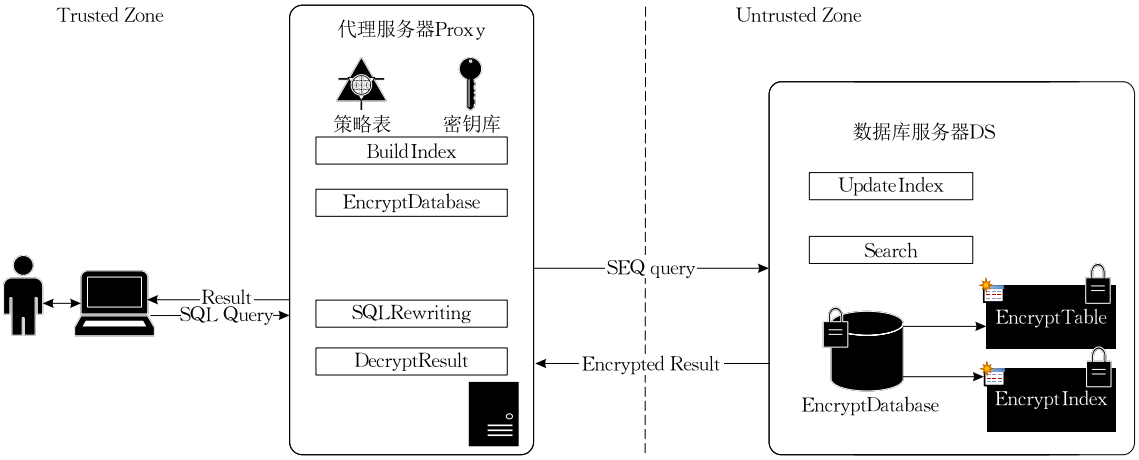

图 1 系统框架图

转换为 SEQ(Search Encrypt Query),发送给数据库服务器,当数据库服务器返回查询结果时,将返回的密文数据解密并返回给客户端.

不可信区域为不可信的数据库服务器 DS,DS 中存储着加密的数据库和用于搜索的安全索引表,索引表中每条记录对应着由数据表构建的安全索引.由于在数据库服务器中,数据库存储的信息是加密的,用户和数据库管理员即使获得了内部数据也无法进行解密获得明文数据.在查询阶段,DS 主要负责执行由代理服务器生成的 SEQ 以及更新安全索引操作.

3.2 用 例

本方案解决了基于可搜索加密机制在密态数据库中进行透明的 SQL 查询,即在不改变 SQL 语义的情况下进行 SQL 查询.本方案主要包括密钥生成、索引的构建、索引的更新、结构化数据加密以及透明的 SQL 查询五个部分.为了更好地解释这些算法,通过一个实例来说明.假设数据库系统中有一个记录学校信息的数据库,里面包含多张数据表,其中一张是学生数据表 Student,Student 表的部分信息如表 1 所示.

表 1 Student 表信息

sid	sname	sage	sgender	Math	English	portrait
1	Peter	23	male	145	101	http://school/stu/peter.jpg
2	Alice	22	female	78	112	http://school/stu/alice.jpg
3	Bob	23	male	106	98	http://school/stu/bob.jpg
4	Susan	20	female	106	107	http://school/stu/Susan.jpg

3.3 形式化定义

本方案包含 5 个算法,分别是:密钥生成算法、安全索引生成算法、数据表加密算法、安全索引更新算法以及安全索引搜索算法.具体如下:

(1)密钥生成算法

KeyGen(1^k)$\rightarrow GK$:客户端运行该算法,给定一个安全参数 k 作为输入,系统输出密钥组 GK.密钥组 GK 由三个伪随机生成的密钥 K_Ω,K_ξ 和 K_γ 组成,分别用于两个伪随机排列函数(Ω 和 ξ)和一个散列函数 γ.

(2)安全索引生成算法

$Index \leftarrow$BuildIndex($Table$,GK):Proxy 运行该算法,将数据表 $Table$ 和密钥组 GK 作为输入,输出安全索引,其中 GK 为由 KeyGen 算法输出的密钥组.

(3)数据表加密算法

$EncTable \leftarrow$EncryptTable($Table$,Key):Proxy 运行该算法,将明文数据表 $Table$ 和密钥 Key 作为输入,输出密文数据表 $EncTable$.

(4)安全索引更新算法

$Index' \leftarrow$UpdateIndex($TrapdoorSet$,Id,$Index$):DS 运行该算法,将从 $Proxy$ 发送的更新关键词令牌集合 $TrapdoorSet$ 和更新的 Id 值以及 DS 中的原安全索引 $Index$ 作为输入,输出新的安全索引 $Index'$.

(5)安全索引搜索算法

$IdSet(w) \leftarrow$Search(t_w,$Index$):DS 运行该算法,将关键词生成的搜索令牌和安全索引作为算法的输入,算法输出符合该查询条件的记录 Id 值集合.

3.4 方案的详细描述

3.4.1 安全索引生成算法

$Index \leftarrow$BuildIndex($Table$,GK):Proxy 运行

该算法，将数据表 $Table$ 和密钥 GK 作为输入，输出安全索引. 该算法生成的安全索引包含两种索引，分别是用于等值查询的等值安全索引和范围查询的保序安全索引，构建安全索引的具体步骤如算法 1 所示. 本节将分别介绍等值安全索引和保序安全索引的结构与构建方式.

算法 1.　安全索引生成算法.

输入：数据表 $Table$ 和密钥组 GK

输出：安全索引 $Index$

1. $D\leftarrow\varnothing$，$IdSet\leftarrow\varnothing$
2. shuffle $Table$，Add Id $column$
3. FOR $record\in Table$
4. FOR $column$
5. IF $\langle columnName=value\rangle\notin D$
6. $num\leftarrow getKeywordNum(Table,\langle columnName=value\rangle)$
7. $D\leftarrow\langle columnName=value\rangle=num$
   END IF
   ENF FOR
   END FOR
8. FOR $w_i\in W$
9. $IdSet[w_i]\leftarrow[Id]$
   END FOR
10. Create S
11. FOR $w_i\in W$
12. Generate random key $k_{i,1}$，$j\leftarrow1$
13. FOR $Id_j\in IdSet[w_i]$
14. Generate random key k_{j+1}
15. $Node_{i,j}\leftarrow\langle Id_j\|k_{i,j+1}\|\Omega_{k_\Omega}(S)\rangle$
16. $EncNode\leftarrow\sigma(Node_{i,j},k_{i,j})$
    END FOR
    END FOR
17. Create $HTable$
18. FOR $w_i\in W$
19. $Val\leftarrow\gamma_{k_\gamma}(w_i)\ \mathrm{xor}\langle address(S[Node_{i,0}])\|k_{i,1}\rangle$
20. $\zeta(Val,k_\zeta)$
    END FOR
21. Output $EQIndex$
22. $OrderIndex\leftarrow BuildOrderIndex(Table,GK)$
23. Random padding random text into S
24. $I\leftarrow EQIndex\|OrderIndex$

等值安全索引是为了支持等值查询而设计的，本文基于 Curtmola 等人[3]和 Monir 等人[14]的索引构造思想设计等值安全索引，其索引结构中的哈希表的构建方式与 Monir 等人[14]的不同，如算法 1 中 1～21 行所示，等值安全索引构建算法包含四个主要步骤：字典的构建、关键词标识符集合的构建、数组 S 的构建和哈希表 $HTable$ 的构建. 其中字典构建阶段是构造 $columnName=value$ 形式的关键词作为字典 D 的键值 key，如关键词 $sage=23$，并将该关键词在数据表中出现的次数作为 $value$ 存储在字典中记为 num，如在 Student 表中关键词 $sage=23$ 对应的值为 2；关键词的标识符集存储每个关键词对应的标识符集合；大数组 S 存储的是关键词的标识符集中的每一个关键词对应的加密形式的链表，链表中的每个节点的结构如图 2 所示，节点由包含该关键词的标识符 Id、用于加密下一个节点的密钥 $k_{i,j+1}$、存放下一个节点的地址 $\Omega_{k_\Omega}(S)$ 构成，其中链表中每个节点均是通过 σ 表示的语义安全的非确定性加密算法进行加密，在本文中使用的是 AES-CBC 加密；哈希表存储着由某个关键词 w_i 的哈希值与 w_i 在 S 中存储的链表的首节点 $Node_{i,1}$ 的信息进行异或生成的密文值，并通过伪随机排列函数 ς 将生成的密文值存储在哈希表 $HTable$ 中.

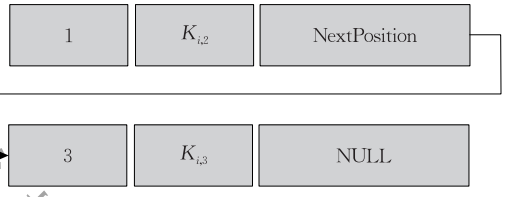

图 2　等值索引中节点的结构图

为了支持更多查询语句，本文基于属性值构造保序安全索引，从而支持比较查询、排序查询以及部分聚合查询如 MAX 和 MIN 操作. 保序安全索引构建的方法不同于等值安全索引，保序安全索引是基于属性构建的，保序安全索引中每个节点是由 4 部分组成. 保序安全索引的生成步骤如算法 2 所示.

算法 2.　保序安全索引生成算法.

输入：数据表 $Table$ 和密钥 GK

输出：安全索引 $Index$

1. Create $IdListSet$
2. FOR $col_i\in IdListSet.keyset$
3. Generate random key $k_{i,1}$，$j\leftarrow1$
4. FOR $IdList_j\in IdListSet[col_i][v]$
5. Generate random key k_{j+1}
6. $Node_{i,j}\leftarrow\langle Id_j\|OPE(v)\|k_{i,j+1}\|\Omega_{k_\Omega}(S)\rangle$
7. $EncNode\leftarrow\sigma(Node_{i,j},k_{i,j})$
   END FOR
   END FOR
8. FOR $col_i\in column$

9. $Val \leftarrow \gamma_{k_\gamma}(col_i)\ \text{xor}\ \langle address(S[Node_{i,0}]) \| k_{i,1} \rangle$
10. $\zeta(Val, k_\zeta)$

END FOR

保序安全索引的构建具体过程如下：

保序安全索引的构建方法包含三个步骤：属性标识符集合 *IdListSet* 的生成、数组 S 的构建、哈希表 *HTable* 的构建，其中保序安全索引与等值安全索引共用一个 S 和 *HTable*. 保序安全索引将表中的属性名作为关键词建立链表，*HTable* 的键值是每个属性名的哈希值，具体步骤如下：

(1) 某个属性值包含的 *Id* 集合 *IdListSet*. *IdListSet* 的键值对应表中的属性名，键值对应的值是由一个哈希表 $H1$ 组成，$H1$ 的键值是该列的某一个值 *columnValue*，$H1$ 的键值对应的值是该列包含该值 *columnValue* 的 *Id* 集合，这里的 *Id* 集合记作 *IdList*.

(2) 数组 S. 与等值索引不同的是保序索引根据属性名建立链表，每一条 L_i 中的节点 $Node_{i,j}$ 由四部分组成，分别是标识符集 *IdList*、加密的属性值，用于加密下一个节点的密钥 $k_{i,j+1}$、存放下一个节点的地址 $\Omega_{k_\Omega}(S)$，图 3 所示的是保序索引中的节点结构图，其中 $Node_{i,j}$ 表示为 $\langle IdList \| \text{OPES}(v) \| k_{i,j+1} \| \Omega_{k_\Omega}(S) \rangle$，其中加密值 OPES($v$) 是通过保序加密(Order Preserving Encryption Scheme，OPES)算法加密的[16]，由于 OPES 算法在保证数据机密性的情况下具有密文保序性，即通过比较 OPES 算法加密生成的密文来获取其明文的大小关系，本方案采用 OPES 算法进行加密. 在为每一个属性生成链表时要满足链表中每个节点之间按照 v 的升序或者降序连接的，即保证链表是有序的. 每个节点是通过与等值索引中相同的加密算法 σ 进行加密的.

(3) 哈希表 *HTable*. *HTable* 存储的值由属性名的哈希值与该属性存储在数组 S 中首节点的信息进行异或操作得到的，即首节点的位置和密钥，最后，通过伪随机排列函数 ς 存储加密信息.

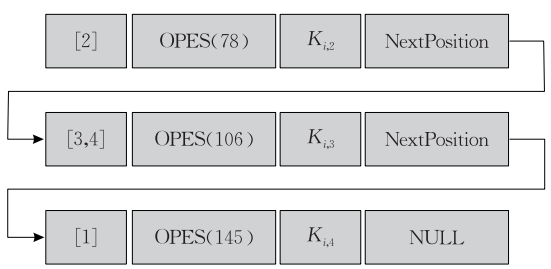

图 3 保序索引中节点的结构图

如算法 1 中 23～24 行所示，在生成两种加密索引后，将数组 S 的空余位置填入随机数，将两种加密索引合成一条加密索引并输出最终的加密索引. 由于将等值安全索引和保序安全索引合成一条安全索引，因此本文中等值安全索引和保序安全索引都是由同一个大数组 S 和一个哈希表组成.

3.4.2 数据表加密算法

$EncTable \leftarrow$ EncryptTable($Table$, Key)：Proxy 运行该算法，将明文数据表 *Table* 和密钥 *Key* 作为输入，输出一个密文数据表. 具体过程如下：

本方案采用语义安全的加密算法 $En=(scheme, Table)$ 来加密数据表，其中 *scheme* 是策略表，策略表根据不同类型的列采取不同的加密算法进行加密，同时根据原数据表的属性大小计算密文列的大小. 此外，数据表的表名和属性名也是加密的，这里不会将新增的 *Id* 列加密，但这并不会影响本方案的安全性，因为 *Id* 列不会泄露任何信息. 数据表加密算法如算法 3 所示.

算法 3. 数据表加密算法.

输入：数据表 *Table* 和密钥 *Key*

输出：加密数据表 *EncTable*

1. $EncTableName \leftarrow \gamma(TableName, k_\gamma)$
2. $scheme \leftarrow AnalysisTable(Table)$
3. $NewTable \leftarrow AlterTable(Table, scheme)$
4. FOR *column* in *NewTable*
5. $EncColumnName \leftarrow \gamma(column, k_\gamma)$

END FOR

6. FOR $record \in NewTable$
7. FOR *column* DO
8. IF $getcolumnType(column, scheme) = num$
9. $val1 \leftarrow paillier(value, Key)$
10. $val2 \leftarrow \sigma(value, Key)$

END IF

11. $val \leftarrow \sigma(value, Key)$

END FOR

END FOR

本方案将数据库中的数据类型分为两类，数值型和字符型，由于数值型需要进行 SUM、AVG、加法操作，因此针对数值型的列额外扩展一列通过同态加密算法加密生成的密文列，由于全同态加密的效率比较低，尽管近些年提出了一些改善全同态加密效率的方案[17-18]，由于上述查询不需要同态乘的性质，因此本方案采用加法同态，使用 Paillier 算法实现[1,8,11]. 本方案涉及了两种加密算法，同态加密

算法和非确定性加密算法，具体的加密方案的说明如表 2 所示.

表 2　加密方案

类型	加密算法	说明
同态加密	paillier(*value*,*key*)	通过 paillier 算法对数据进行同态加密
非确定性加密	AES-CBC(*value*,*key*)	通过非确定性加密函数对数据进行加密

AES-CBC 算法是语义安全的非确定性加密算法，该算法满足 IND-CPA(自适应选择明文攻击)安全，该加密算法是概率性的，可以保证两个相同的明文值被加密为不同的密文值，由该加密算法加密生成的密文不能执行有效地计算[1].

Paillier 算法是一种安全的概率加密算法，允许服务器在密文数据上进行计算. Paillier 满足加法同态性质，将两个加密的值相乘，结果是两个明文求和的加密值，即 $HOM_k(a)\cdot HOM_k(b)=HOM_k(a+b)$，同态加密也可用于通过 DBMS 返回的个数和密文的乘积来计算平均值操作.

本文将根据策略表中的加密算法对不同类型的数据进行加密. 针对数值型的属性采取同态加密算法和非确定性加密算法进行加密，字符型通过非确定性加密算法进行加密.

密文数据表空间优化策略：由于很多 SQL 语句会对整个数据表进行扫描，密文的长度不仅增大空间的占用同时也影响查询的性能. 出于安全性考虑，同态加密会将明文值加密为 1024 位或 2048 位，本文采用密文打包技术，即将多个数值型数据打包进一个 1024 位的 Paillier 密文中. 本文将一行中的多个数值型的属性列和多个行的值打包进一个密文，这种打包方案可以有效地减少密文的空间占用，对于一个 64 位的明文属性列，这种打包方案能使 Paillier 密文的每行空间开销降低 90%.

如算法 3 所示，通过 *AnalysisTable* 方法返回一个字典类型的策略表，里面包含了将原属性转换为密态属性的数据类型和针对某个属性列的加密算法. 将策略表和数据表作为输入，运行 *AlterTable* 方法来扩展原数据表的属性列，并根据策略表来重构密态数据库表结构. 通过 *getcolumnType* 方法来获取策略表中的某个属性的所属类型，根据所属类型采用相应的加密算法加密.

本方案在查询过程中不会降低密文数据的安全性，不会像 cryptDB 将不安全的加密层暴露在数据库中，本方案不仅保证了数据的安全性，同时也保证了密文数据较小的空间占用.

3.4.3　安全索引更新算法

$Index'\leftarrow$UpdateIndex(*TrapdoorSet*, *Id*, *Index*)：DS 运行该算法，将从 Proxy 发送的更新关键词令牌集合和更新的 *Id* 值以及 DS 中的原加密索引作为输入，输出新的安全索引.

当用户执行 INSERT、UPDATE、DELETE 操作时，DS 会执行索引的更新操作. 由于索引的更新操作在 DS 上执行，因此为了保证安全索引的安全性，需要在不解密安全索引的情况下对安全索引进行更新，这导致每次添加或者删除数据项时都需要重新构造索引结构，会严重影响方案的性能，而且会泄露较多信息给数据库服务器. 本方案结合可搜索加密思想，在数据库服务器中运行索引更新算法，通过代理服务器端构建的搜索令牌集合更新索引，实现安全索引的高效更新. 具体更新方法如下：

针对不同类型的 DML，获取更新的表名、该表的索引，将索引解析成 *S* 和 *HTable* 两部分，在这里 *S* 和 *HTable* 是加密形式的，因此不会泄露任何敏感信息，通过分别对等值安全索引和保序安全索引内部的 *S* 和 *HTable* 进行更新实现对安全索引进行更新.

(1) INSERT

Proxy 截取用户发送的 INSERT 请求，获取表名和关键词集合 *keywordSet*，服务器生成插入的新记录对应的 *Id* 值，为 *keywordSet* 生成搜索令牌集合 *TrapdoorSet*，本方案生成搜索令牌的方法采用哈希函数 HMAC 来实现，代理重写 INSERT 查询，将其转换为加密的 SQL(SEQ)；代理将 *TrapdoorSet* 和 *SEQ* 作为输入同时更新等值安全索引和保序安全索引，生成 *SEQ* 和 *TrapdoorSet* 详细步骤如算法 4 所示.

算法 4.　生成 *SEQ* 和 *TrapdoorSet* 算法.

输入：SQL 语句

输出：令牌集合 *TrapdoorSet*, *SEQ*

1. *tableName*$\leftarrow$*getTableName*(*query*)
2. *Generate Id*, *TrapdoorSet*$\leftarrow\varnothing$
3. *keywordSet*$\leftarrow$*getkeywordSet*(*query*)
4. FOR *keyword* in *keywordSet*
5. $t_w\leftarrow$HMAC(*keyword*, k_γ)
6. *TrapdoorSet*$\leftarrow$*push*(t_w)

END FOR

7. *UpdateIdSet*(*keywordSet*, *Id*)

8. $SEQ \leftarrow RewriteQuery(query, Id)$

DS 使用函数 $getIndex$ 解析索引，将其解析为 S 和 $HTable$ 两个数据结构，分别通过 $HTable$ 定位 S 中的等值安全索引和保序安全索引的链表的首节点. 首先更新等值安全索引，获取该节点对应的链表的尾节点信息(尾节点在 S 中的地址和解密尾节点所需的密钥)，根据搜索令牌集合构造新节点，将新的节点加密后作为该链表的尾节点插入到数组 S 中，如果 $HTable$ 中不存在该关键词，则在数组 S 中创建新的链表，将该节点作为新链表的首节点，并将该节点的信息存储到 $HTable$ 中；在更新保序安全索引时，利用属性名生成的令牌和对应值生成的令牌更新 S 中该属性对应的链表节点. 最后将新的 S 和 $HTable$ 合并更新到索引表中，针对 INSERT 操作的更新如算法 5 所示.

算法 5. 安全索引更新算法.

输入：令牌集合 $TrapdoorSet$，新插入的记录 Id 值，安全索引 $Index$

输出：新的安全索引 $Index'$

1. $S, HTable \leftarrow getIndex(Index)$
2. FOR t_w in $TrapdoorSet$
3. IF t_w xor $HTable$ is not NULL
4. Generate $newkey, newpos$
5. $Node \leftarrow getLastNode(pos, S)$
6. $Node \leftarrow \langle oldID \| newkey \| newpos \rangle$
7. $newNode \leftarrow \langle Id \| NULL \| NULL \rangle$
8. $EncryptNode \leftarrow \sigma(Node, curkey)$
9. $EncryptnewNode \leftarrow \sigma(newNode, newkey)$
10. $S[lastpos] \leftarrow EncryptNode$
11. $S[newpos] \leftarrow EncryptnewNode$
12. ELSE
13. Create new nodes in S and $HTable$

END IF

END FOR

当从客户端发送诸如 INSERT INTO Student (sid, $sname$, $sage$, $sgender$) VALUES (4, '$Lucy$', 25, '$female$') 的更新请求时，Proxy 获取表名 Student 和关键词集合 $keywordSet = \{sid = 4, sname = 'Lucy', sage = 25, sgender = 'female', sid, sage, sgender\}$，代理服务器为关键词集合中的每个关键词生成搜索令牌，并将更新查询重写为 SEQ 查询 INSERT INTO $EncTable$ ($EncColSet$) VALUES ($EncValSet$)，其中 $EncColSet$ 为密态属性列集合，$EncValSet$ 为通过对应的加密规则加密的加密值集合. 最后数据库根据相应的加密规则对数据库进行更新.

(2) UPDATE

客户端向服务器发送诸如 UPDATE Student SET $sage = 25$ WHERE $sage = 24$ 的 SQL 请求时，服务器会获取 3 个关键词，"$sage = 25$"、"$sage = 24$"和"$sage$"，为这些关键词生成搜索令牌集合 $TrapdoorSet$，查询包含关键词"$sage = 24$"的 Id 值，Proxy 将 $TrapdoorSet$、Id 和 SEQ 发送给 DS. DS 根据 $HTable$ 获取"$sage = 24$"在 S 中的首节点地址，进行遍历并删除包含该 Id 值的节点，如果需要被删除的是首节点，同时需要删除 $HTable$ 中该关键词对应的首节点信息；通过 $HTable$ 查找保序安全索引的链表并进行更新 24 这个节点的 $IdList$. 查询 $HTable$ 中关键词"$sage = 25$"的信息，将需要更改的记录的 Id 值分别更新到等值安全索引和保序安全索引中.

3.4.4 安全索引搜索算法

$IdSet(w) \leftarrow \text{Search}(t_w, Index)$：DS 运行该算法，将关键词生成的搜索令牌和安全索引作为算法的输入，算法输出符合该查询条件的记录 Id 值集合.

本文为了解决密文查询难度较大以及查询泄露问题，结合可搜索加密思想，通过在代理服务器端解析 SQL 查询并根据查询构建关键词的搜索令牌，即 $t_w = \gamma(keyword, k_\gamma)$，通过搜索令牌将 SQL 查询转换为 SEQ 查询，并发送到数据库服务器. 服务器运行搜索算法，将满足条件的 Id 集合返回，搜索算法模型图如图 4 所示.

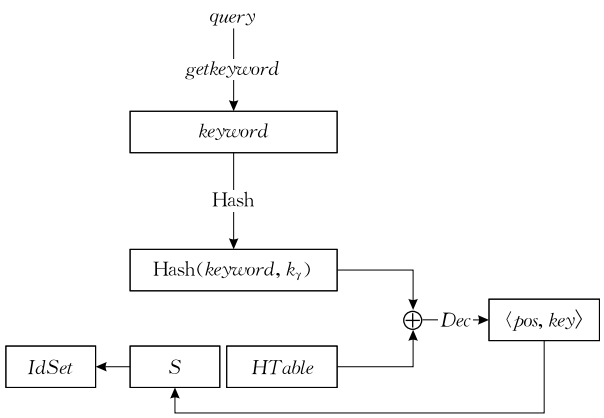

图 4 搜索算法模型图

为了支持丰富的 SQL 查询，本方案给出了等值查询、范围查询、布尔查询、聚合查询方案. 具体如下：

(1) 等值查询

针对等值查询，本文通过为关键词生成搜索令牌并通过令牌在索引中进行搜索. 用户发送诸如 SELECT * FROM Student WHERE *sgender*='*male*' 的 SQL 请求，系统获取关键词"*sgender*='*male*'"，将关键词和密钥作为输入运行 *CreateTrapdoor* 生成搜索令牌，并在该表对应的安全索引 I 上执行搜索算法，通过 *HTable* 和 S 搜索包含该关键词的 *Id* 集合，最后将包含这些 *Id* 值的记录返回，具体的搜索索引算法如算法 6 所示.

算法 6.　安全索引搜索算法.

输入：令牌 t_w，安全索引 *Index*

输出：符合条件的标识符 *Id* 集合 *IdSet*

1. S, *HTable* ← *getIndex*(*Index*)
2. $pos \| k \leftarrow t_w$ xor *HTable*
3. *Node* ← *Decrypt*($S[pos]$, k)
4. Continue to decrypt the linked list corresponding to the keyword until the last node

(2) 范围查询

客户端发送带有比较运算符的查询，例如 *sage*>20. 本方案分别为 *sage* 和 20 生成搜索令牌，获取该表的索引，通过搜索令牌查找安全索引对应的链表，由于链表是有序的，假设链表是升序的，系统从首节点遍历，依次将每个节点的加密属性值与 20 生成的令牌作比较，由于索引中每个节点的 OPES(v)部分是可以比较大小的，从链表的首节点开始比较，找到符合条件的第一个节点后，将链表的该节点及后面节点的标识符集合求并集，其结果则为满足 *sage*>20 条件的记录的标识符集合. Monir 等人提出的范围查询解决方案是将范围查询转换为多个关键词查询，将每个关键词查询结果求并集来实现范围查询，这会导致如果查询范围是 25>*sage*>20 时，会将该范围查询转换为 4 个关键词查询，因此该范围查询效率非常低，相较于 Monir 等人的方案，本方案只需查询一次，并且得益于链表是有序的，因此本方案的范围查询在保证数据安全的同时查询效率非常高.

针对求最大值最小值的聚合查询，如 MIN(*sage*)，系统同样通过保序安全索引查找，利用 *sage* 的搜索令牌查找 S 中该属性对应的链表，由于链表是升序的，所以链表的首节点即为要查找的.

(3) 布尔查询

针对布尔查询本文将等值查询或范围查询的结果做布尔操作. 在查询诸如SELECT * FROM Student WHERE *sage*=23 OR *sgender*='*male*' 的 SQL 语句时，Proxy 将其改写为 SEQ 查询，即更改为 SELECT * FROM *EncTable* WHERE *Id* in Search (t_{w1}, *Index*) OR Search (t_{w2}, *Index*)，为两个关键词分别生成对应的令牌，并分别执行 Search 的操作，最后将返回的 *Id* 集合做并集操作，将两个 *Id* 集合的并集中的记录去重后返回，布尔查询的算法如算法 7 所示.

算法 7.　布尔查询算法.

输入：SQL，安全索引 *Index*

输出：符合条件的标识符 *Id* 集合 *IdSet*

1. *keywordSet* ← *getkeywordSet*(SQL)
2. FOR *keyword* in *keywordSet*
3. 　t_w ← *CreateTrapdoor*(*keyword*, k_y)
4. 　*IdSet*(*keyword*) ← Search(t_w, *Index*)
   END FOR
5. Make the result a corresponding Boolean operation

(4) SUM 操作

SUM 操作是针对数值类型求和的操作，在执行 SUM 操作时，调用采用 Paillier 算法加密的列. DS 在接收到 SEQ 查询后，调用 UDF 将求和的密文列进行密文值间的相乘操作，最终得到的就是结果的加密值，将密文发送到 Proxy 中并进行解密. 在 SUM 求和的操作基础上，通过 DBMS 返回的合计和记录数计算平均值来实现 AVG 操作.

通过以上方案可以很好地支持单列的求和操作和求平均数操作，针对 SUM(Math * English)的操作，即通过同态加密执行两个加密值的乘积的求和的聚合查询操作，本方案使用行预计算技术，本方案只支持单表聚合查询，不支持跨表查询. 在生成加密数据库结构时，为每张加密数据表增加一列，这个列包含表中一个表达式的加密值，这个表达式一般为该行的其他数值类型的列的计算表达式. 这个值可以在查询需要运行该表达式时迅速响应并返回结果，避免了在加密数据上做表达式计算的问题. 例如执行诸如 SUM(Math * English)的查询，系统需要在附加列中存储 Math * English 的同态加密值，这样在运行查询时，就能够直接在数据库服务器上计算 SUM(Math * English)了，客户端仅需要下载加密的聚合值，而不需要分别下载列 Math 和列 English 的所有值.

4　安全性分析

本节对密文数据库存储的安全性、查询与安全

索引更新的安全性进行分析.

4.1 存储安全性分析

首先,分析数据存储安全.本文在数据存储方面包括密文数据表、加密索引.密文数据表中的密文是通过非确定性加密算法和同态加密算法对明文数据进行加密的.非确定性加密算法采用的是 AES-CBC 算法,该加密算法可以保证相同的明文对应的密文不同,解决了如性别这种易区分的列产生密文等值泄露的问题,该算法符合 IND-CPA(自适应选择明文攻击的不可区分性)安全.同态加密算法采用的是 Paillier 同态加密算法,该算法是一种安全的概率加密算法,满足 IND-CPA 安全.密文数据表中只有标识符 Id 列是没有被加密的,但由于 Id 列与明文数据表的数据没有特殊的关联,敌手通过 Id 列不会获取任何明文信息,密文数据库中存储的密文数据均是通过满足 IND-CPA 安全的加密算法生成的,因此密文数据库中存储的数据不会泄露任何明文信息,存储的密文不会产生等值泄露和顺序泄露.本文中的加密索引的节点均采用 AES-CBC 算法加密,该加密算法满足 IND-CPA 安全,对于保序安全索引的节点的 OPES(v)部分采用了保序对称加密算法(Order Preserving Encryption Scheme,OPES),该算法在保证了功能性的情况下也保证了安全性,OPES 算法不会泄露具体的明文值,它只会泄露明文间的顺序,OPES 加密算法具有可证明的安全性保证,加密相当于一个保留顺序的随机映射,并且 OPES 算法的安全性在 Agrawal 等人的论文[16]和 Popa 等人提出的 CryptDB 方案[1,8]中已经得到了证明.当敌手通过某种渠道获取加密索引时,由于索引的每个节点是通过满足 IND-CPA 安全的非确定性加密算法加密,因此敌手不会获取节点中的信息包括保序索引的节点的 OPES(v)部分.因此在不可信的数据库服务端存储的密文数据表和加密索引不会泄露任何与明文信息相关的信息包括具体的明文值和明文间的顺序.

4.2 查询与索引更新安全性分析

其次,分析数据库系统查询和更新的安全性.查询的过程是基于可搜索加密机制在密文索引中进行操作,在查询加密索引的过程中,系统只会将 Id 暴露出来,但是 Id 不会泄露任何敏感信息.在查询和更新的过程中不会对密文数据表中的密文做任何降低安全性的操作,密文数据表中的密文会始终显示为由 AES-CBC 和 Paillier 加密算法加密生成的密文.在进行 SQL 查询时,Proxy 会将 SQL 中的关键词进行加密处理,这里采用的均是带盐哈希加密算法,由于该算法的不可逆性和抗碰撞性可以保证重写后的加密 SQL 的安全性,敌手从重写的 SQL 和返回的结果中无法学习到任何的东西.在数据库服务器端进行查询时,首先会将 Proxy 生成的搜索令牌作为输入执行搜索索引算法,在搜索的过程中,等值安全索引不会泄露任何明文信息,保序安全索引只会泄露节点间的顺序但不会泄露具体的明文值,每个节点都是通过满足 IND-CPA 安全的非确定性加密方案加密的,敌手即使知道了节点的顺序关系,也不会得到任何明文值.因此,基于安全索引和密文数据表采用的加密算法的安全性,查询过程中不会泄露任何明文值信息.

综上所述,由于本文在构建安全索引时使用满足 IND-CPA 安全的加密方案加密索引中的每个节点,故加密后的节点具有不可区分性,同时存储在数据库中的密文均通过满足 IND-CPA 安全的加密算法加密生成的,其密文具有概率性;在搜索过程中,本文为关键词通过加盐哈希算法生成搜索令牌,由于采用的哈希算法具有抗碰撞性和不可逆性,因此敌手无法从搜索令牌中获取关键词的明文信息,在查询过程不会降低数据库中存储的密文的安全性;本方案的安全索引构建过程是在 Curtmola 等人的方案[3]的基础上设计的,因此安全性满足 IND-CKA1 安全.因此从密态数据库建立到查询结束本文的安全性始终满足 IND-CKA1(选择关键词攻击下的索引不可区分性)安全.

5 实验分析

首先从功能性与安全性角度对比 DB_SE、CryptDB 和 BlindSeer,具体结果如表 3 所示.

表 3 方案对比

系统	支持的操作类型						安全性	
	等值	布尔	范围	聚合	连接	更新	等值泄露	顺序泄露
CryptDB	√	√	√	√	√	√	√	√
BlindSeer	√	√	√	×	×	√	×	×
DB_SE	√	√	√	√	×	√	×	×

从功能性角度,CryptDB 支持的查询类型最丰富,本方案除了连接查询以外的所有查询类型都支持,BlindSeer 支持的查询类型较少,不支持聚合查询和连接查询.

从安全性角度分析，由于 CryptDB 方案在查询过程中进行剥层操作，如在进行等值查询和分组查询时，需要解密最外层的 RND 层，这会导致 DET 密文曾暴露在最外层，而 DET 密文层是通过确定性加密方案加密的，DET 层的密文会产生密文等值泄露，即密文数据库中相同的明文对应的密文也相同，并且查询结束后不会将该层的密文加密到最安全的状态，因此在执行查询后会将安全性低的密文暴露在外面，最终存储在数据库的密文会产生密文等值泄露和顺序泄露；相较于 CryptDB 方案，本方案和 BlindSeer 方案隐藏了密文等值泄露和顺序泄露.

接下来进行性能分析，根据本方案的架构进行部署实现与测试，Proxy 与 DS 上的算法通过 Java 实现，数据库采用的是 MySQL5.5 进行测试，运行环境的系统为 Ubuntu14.04LTS. Proxy 和 DS 配置为 Intel(R) Core(TM) i7-8550U CPU、8 GB 内存，每个实验结果均为实验重复 50 次并取平均值后得到的.

测试用例为 3.1 节所示的 Student 表，该表包含 10000 条记录，每条记录包含 7 个属性，最终结果显示我们的方案总执行时间为 34.1 s，原始数据库上传数据库时间为 4.09 s，其中原始数据库上传数据库总时间不包含加密时间和索引建立时间. 单表密态数据库转化测试结果如图 5 所示.

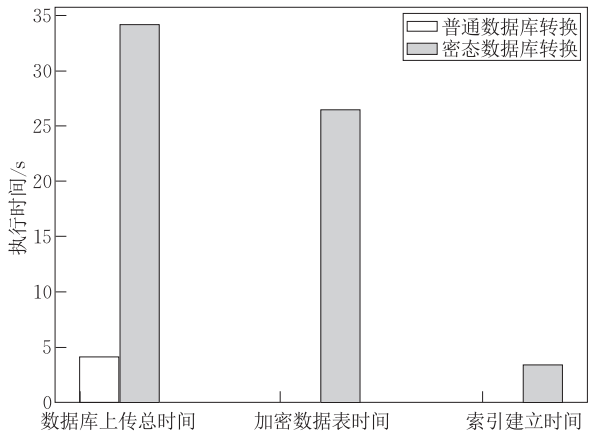

图 5　单表多表密态数据库转化结果

我们将本方案与 CryptDB 方案分别进行查询效率测试，测试用例同为 Student 表，分别测试了等值查询、布尔查询、SUM 求和、MAX、MIN 和范围查询操作. 测试结果如图 6 所示，实验结果表明本方案相较于 CryptDB 方案在查询效率上有明显的提升. 由于本方案采用保序安全索引进行范围查询，相较于 CryptDB 方案省去了脱层处理，并且本方案中保序安全索引中的链表是有序的，因此本方案进行排序查询和范围查询的效率相比 CryptDB 方案更高；由于本方案采用行预计算技术，因此部分聚合查询(SUM)的速度要优于 CryptDB 方案.

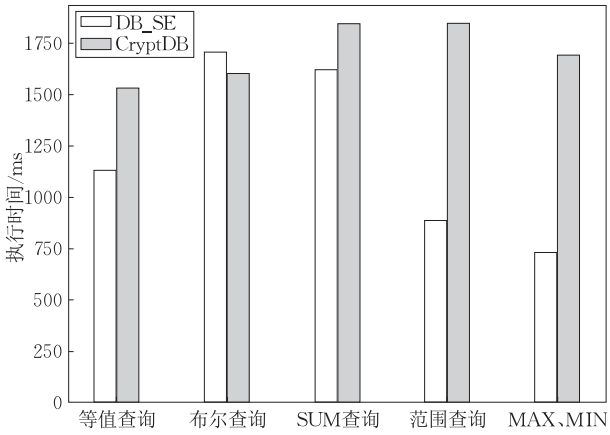

图 6　查询时间测试结果

6　总　结

针对云服务器上数据库系统的信息泄露问题，本文提出了 DB_SE. 方案基于 IND-CPA 安全的非确定性加密方案和同态加密方案对数据进行加密，该方案满足 IND-CKA1 安全，在表结构重构时，针对字符型数据列扩展为两列，针对于数值型列扩展为三列，相较于 CryptDB 密文列共减少了 4 列. 融合可搜索加密索引构建思想，设计安全索引，将明文 SQL 查询更改为密态 SEQ 查询，实现在不泄露数据信息的情况下进行检索. 除此之外，方案基于同态加密的加法同态性，实现了密态数据库上的 SUM 求和操作和 AVG 操作；本方案支持复杂的 SQL 查询语句，包括等值查询、布尔查询、聚合查询、范围查询等，具有较强可用性. 未来将在此研究基础上，优化其在大数据集上的查询效率，并设计支持更多更丰富的 SQL 操作语句，最终发展成一个完整实用的密态数据库系统，并在实际场景进行部署应用.

参　考　文　献

[1] Popa R A, Redfield C, Zeldovich N, et al. CryptDB: Protecting confidentiality with encrypted query processing//Proceedings of the 23rd ACM Symposium on Operating Systems Principles. Cascais, Portugal, 2011: 85-100

[2] Song D, Wagner D, Perrig A. Practical techniques for searches on encrypted data//Proceedings of the 2000 IEEE Symposium on Security and Privacy. Berkeley, USA, 2000: 44-55

[3] Curtmola R, Garay J, Kamara S, Ostrovsky R. Searchable symmetric encryption: Improved definitions and efficient constructions//Proceedings of the ACM Conference on Computer and Communications Security 2006. Alexandria Virginia, USA, 2006: 79-88

[4] Kamara S, Papamanthou C, Roeder T. Dynamic searchable symmetric encryption//Proceedings of the 2012 ACM Conference on Computer and Communications Security. Raleigh North Carolina, USA, 2012: 965-976

[5] Cash D, et al. Dynamic searchable encryption in very-large databases: Data structures and implementation//Proceedings of the Network and Distributed System Security Symposium. San Diego, USA, 2014: 853-885

[6] Hore B, Mehrotra S, Tsudik G. A privacy-preserving index for range queries//Proceedings of the 30th International Conference on Very Large Data Bases. Toronto, Canada, 2004: 720-731

[7] Amanatidis G, Boldyreva A, O'Neill A. New security models and provably-secure schemes for basic query support in outsourced databases//Proceedings of the Working Conference on Data and Applications Security. CA, USA, 2007: 14-30

[8] Popa R A, Redfield C M, Zeldovich N, et al. CryptDB: Processing queries on an encrypted database. Communications of the ACM, 2012, 55(9): 103-111

[9] Refaie R, El-Aziz A A A, Hamza N, et al. A new efficient algorithm for executing queries over encrypted data//Proceedings of the 2015 International Conference on Computing, Communication and Security (ICCCS). Nanjing, China, 2015: 1-4

[10] Akin I H, Sunar B. On the difficulty of securing Web applications using CryptDB//Proceedings of the IEEE 4th International Conference on Big Data & Cloud Computing. Sydney, Australia, 2014: 745-752

[11] Shahzad F, Iqbal W, Bokhari F S. On the use of CryptDB for securing electronic health data in the cloud: A performance study//Proceedings of the International Conference on E-health Networking. Boston, USA, 2016: 120-125

[12] Pappas V, Krell F, Vo B, et al. Blind Seer: A Scalable Private DBMS//Proceedings of the 2014 IEEE Symposium on Security and Privacy. CA, USA, 2014: 359-374

[13] Poddar R, Boelter T, Popa R A. Arx: A strongly encrypted database system. IACR Cryptology ePrint Archive, 2016, 2016: 591

[14] Azraoui M, Melek Önen, Molva R. Framework for Searchable Encryption with SQL Databases//Proceedings of the 8th International Conference on Cloud Computing and Services Science. Madeira, Portugal, 2018: 57-67

[15] Goh E J. Secure Indexes. IACR Cryptology ePrint Archive, 2003: 216-235

[16] Agrawal R, Kiernan J, Srikant R, et al. Order preserving encryption for numeric data//Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. Paris, France, 2004: 563-574

[17] Cao X, Moore C, O'Neill M, et al. High-speed fully homomorphic encryption over the integers//Proceedings of the International Conference on Financial Cryptography and Data Security. Accra Beach Hotel & Spa, Barbados, 2014: 169-180

[18] Gentry C, Halevi S, Smart N P. Better bootstrapping in fully homomorphic encryption//Proceedings of the International Workshop on Public Key Cryptography. Darmstadt, Germany, 2012: 1-16

SUN Xi-Ze, M. S. His main research interests include searchable encryption and encrypted database.

ZHOU Fu-Cai, Ph. D., professor. His main research interests include cryptography and network security, trusted computing.

LI Yu-Xi, Ph. D. Her research interests include searchable encryption and cloud security.

ZHANG Zong-Ye, Ph. D. candidate. Her main research interest is searchable encryption.

Background

Outsourcing data storage and processing to third-party servers, such as cloud servers, has become a common procedure. With the increase in the amount of data stored in cloud servers, information disclosure on cloud servers (requires protected information or privacy being compromised) has caused widespread concern. In order to solve the problem of information leak, encryption is a standard approach to ensure the confidentiality of data outsourced at cloud servers. But some security-enhanced encryption schemes are not functional.

CryptDB Proposed by (Popa et al., 2012) which uses onion model to encrypt data. CryptDB supports a variety of query types, but the security is not high. The major drawback of CryptDB lies in the fact that whenever one layer is removed, the encryption scheme becomes weak. Pappas et al Proposed

BlindSeer which supports a rich query set. BlindSeer designed the index server to implement equivalence query, Boolean query and range query through BF search tree and Yao's garbled circuit. BlendSeer does not support aggregation. Monir et al. proposed a database encryption scheme in 2017. The scheme supports fewer types of queries, and the range query efficiency is very low.

We propose a database encryption scheme based on searchable encryption. Our scheme supports equivalent queries, Boolean queries, range queries, aggregate queries, sort queries etc. Our scheme can satisfy most of the daily query needs. We provide methods for converting an effective encryption database, methods for updating an index, and methods for querying data. Our scheme encrypts data by secure encryption scheme under IND-CPA. Our scheme allows leakage of some search pattern information, but protect the query and data, provide a high level of privacy for individual terms in the executed SEQ. Our scheme is secure under IND-CKA1.

In recent years, we have devoted to the researches of searchable encryption and database encryption, and have gained achievements on searchable encryption which propose a Boolean searchable encryption scheme. The scheme supports integrity-verifiable conjunctive keyword.

This work was supported by the National Natural Science Foundation of China (Nos. 62072090 and 61872069), and the Fundamental Research Funds for the Central Universities (No. N2017012).