

基于STE的减少查询泄露的关系数据库加密方案

苏雨晨, 马昌社

(华南师范大学计算机学院, 广东 广州 510631)

摘要: 近年提出的基于部分预计算的SQL连接索引(PpSj)方案是一种基于结构化加密技术的关系数据库加密方案, 它利用部分预计算索引连接技术和过滤哈希集技术支持高效的连接查询和布尔查询。但是, 该方案也存在一些缺陷, 主要表现在执行布尔查询时会泄露过多的信息, 以及不能支持范围查询。针对这些问题, 提出一种改进的多功能加密数据库(MFEDB)方案, 该方案在PpSj方案基础上, 引入一种混合过滤技术, 结合2种过滤方法, 减少了布尔查询的信息泄露, 扩展了支持的结构化查询语言(SQL)的查询子集, 包括等值查询、连接查询、布尔查询和范围查询, 同时平衡了服务器存储开销与客户端和服务端之间产生的通信开销。

关键词: 等值查询; 连接查询; 范围查询; 布尔查询; 结构化加密

中图分类号: TP309

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2024020

STE-based encrypted relational database scheme with reduced leakages

SU Yuchen, MA Changshe

School of Computer Science, South China Normal University, Guangzhou 510631, China

Abstract: Recently, the concept of partially precomputed indexing for Structured query language (SQL) join (PpSj), which was predicated on structured encryption, has been introduced. This approach employed partial precomputed index join and hash filter set technology to facilitate efficient execution of join queries and Boolean queries. However, the scheme has encountered some limitations, primarily characterized by excessive information leakage during the execution of Boolean queries and an inability to support range queries. To resolve these issues, an enhanced relation database encryption scheme, termed multi-function encrypted database (MFEDB), was proposed. This scheme incorporated a hybrid filter technique derived from the PpSj scheme, integrating two filtering methods. It aimed to minimize the information leakage associated with Boolean queries, expand the subset of supported SQL queries to include equivalent queries, join queries, Boolean queries, and range queries, and balance the trade-off between the server's storage costs and the communication overhead between the client and the server.

Keywords: equivalent query, join query, range query, Boolean query, structured encryption

收稿日期: 2023-11-14; 修回日期: 2024-03-04

通信作者: 马昌社, chsma@sncu.edu.cn

引用格式: 苏雨晨, 马昌社. 基于STE的减少查询泄露的关系数据库加密方案[J]. 网络与信息安全学报, 2024, 10(02): 165-181.

Citation Format: SU Y C, MA C S. STE-based encrypted relational database scheme with reduced leakages[J]. Chinese Journal of Network and Information Security, 2024, 10(02): 165-181.

0 引言

随着云存储技术的快速发展,外包数据库以其海量存储资源以及高效的计算能力吸引了众多企业将其数据存储于云服务器,这大幅降低了企业维护数据的成本。然而,云服务器因为收集和存储了大量重要的用户隐私数据,如健康数据、财务记录和社交互动等,成为不法分子攻击的中心目标,而且服务器内部管理员也可能会利用职务之便窃取和贩卖存储在服务器的用户隐私数据,这使得云数据库处在一个非常严峻的环境中。因此,数据库加密对于数据库安全具有重要的意义。数据库加密指将数据加密后存储在数据库中,并保证数据的机密性、完整性和可用性。通过对数据库中的敏感数据进行加密,可以最大限度地减少未经授权访问和数据泄露的风险,避免敏感数据被不良分子窃取或者篡改。

为了支持丰富的结构化查询语言(structured query language, SQL)查询功能,现有许多数据库加密方案的实现依赖属性保留加密(property-preserving encryption, PPE)或属性揭示加密(property-revealing encryption, PRE)。它们将部分属性列按照有序索引进行存储,允许加密值的顺序关系和明文的顺序关系保持一致。或者将同一属性列相同的明文确定性加密成相同的密文。这样做虽然实用,但也揭示了加密属性列的有序性和相等性,容易受到泄露滥用攻击(leakage abuse attack, LAA)^[1-3]。

一种比基于PPE和PRE的方案泄露更少的是2010年Chase等^[4]引入的结构化加密(structured encryption, STE)方案,它使用辅助加密数据结构(如加密字典和加密多映射)来支持部分SQL查询。这是通过将SQL查询转换为结构化加密查询来实现的。客户端首先将SQL查询语句转换为结构化查询令牌,然后将令牌传递给服务器以查询辅助数据结构,服务器输出密文检索结果,最后由客户端对输出结果进行编译、解密和处理,得到SQL查询的明文结果表。

现有许多基于标准STE技术的数据库加密方案都存在缺陷。Kamara等^[5]提出的选择投影索引(selection-projection index, SPX)方案将SQL查

询转化为选择投影叉乘(selection-projection-cross, SPC)代数的形式,着重解决连接查询,这局限于一部分SQL查询,并且泄露了连接列的相等模式。后续,Cash等^[6]提出的基于部分预计算的SQL连接索引(partially precomputed indexing for SQL join, PpSj)方案对SPX方案进行了改进和拓展,利用部分预连接索引技术解决了相等模式泄露的问题,并且采用盲化交叉标签(oblivious cross-tags, OXT)^[7]中的过滤哈希集技术支持执行布尔查询,但是这种过滤方法会泄露任意2个关键字之间的交集。

本文提出一种关系数据库加密方案,该方案支持更多SQL查询,减少查询泄露并提供范围查询功能。该方案对PpSj方案进行了优化和拓展,丰富了查询功能并保障了数据隐私。PpSj方案虽然能够简单地实现布尔查询,但是会泄露任意2个关键字之间的交集。为了减少这个泄露,利用容斥索引(inclusion-exclusion index, IEX)^[8]和高效交叉标签(efficient cross-tags, EXT)^[9]方案的优势,提出一种混合过滤器,它能够在减少隐私泄露的同时,平衡方案的性能。其次,本文方案拓展了所支持的SQL查询类型,该方案不仅支持等值查询、连接查询和布尔查询,还支持范围查询。最后,实验结果表明,本文提出的方案是高效且实用的。

4种方案理论性能对比如表1所示。其中 I 代表所有关键字的集合, l_q 表示查询 q 中查询条件的个数, $\#DB(I)$ 表示关键字 I 在数据库DB中搜索结果集的个数, $\#MM_I$ 是任意2个关键字的结果的交集大小的总和,可以表示为 $\#MM_I = \sum_i \sum_j \#DB(I_1) \cap \#DB(I_2)$ 。表1中各方案的搜索效率在理论上是一致的,但是在具体实现中,EXT方案和PpSj方案存在更多的索引操作,这导致了搜索效率下降。通过对各方案的研究,可以分析出在执行布尔查询时,PpSj方案会泄露任意2个关键字的交集,而多功能加密数据库(multi-function encrypted database, MFEDB)方案只泄露了第1个关键字与其他关键字的交集。

1 相关工作

2000年,Song等^[10]首次提出可搜索对称加

表1 4种方案理论性能对比
Table 1 Theoretical performance among four schemes

方案	搜索复杂度	通信复杂度	服务器存储	查询泄露
EXT	$O(\#DB(I))$	$O(\#DB(I))$	$O(\sum_i \#DB(I))$	$I_i \wedge I_j$
IEX	$O(\#DB(I))$	$O(I_q)$	$O(\#MM_I + \sum_i \#DB(I))$	$I_i \wedge I_j$
PpSj	$O(\#DB(I))$	$O(I_q)$	$O(\sum_i \#DB(I))$	$I_i \wedge I_j$
MFEDB	$O(\#DB(I))$	$[O(I_q), O(\#DB(I))]$	$[O(\sum_i \#DB(I)), O(\#MM_I + \sum_i \#DB(I))]$	$I_i \wedge I_j$

密 (searchable symmetric encryption, SSE) 的概念, 该方案能够在加密文档上进行关键字搜索。Curtmola 等^[11]在 2006 年提出一种高效安全的方案, 该方案基于倒排索引构造, 并支持多用户的 SSE。随后, SSE 的研究工作在提高查询效率^[12-13]、实现动态查询^[14-16]、保证前后向隐私^[17]上取得了许多进展。然而, 以上方案仅适用于文档集合的搜索, 而在实际中, 大部分数据通常存储在关系数据库中。因此, 利用加密技术构造关系数据库的加密方案, 以实现加密关系数据库的各种 SQL 查询, 具有重要的现实意义和实用价值。

2011 年, Popa 等^[18]提出一种具备实用价值的加密数据库 (crypto database, CryptDB) 方案, 该方案利用可搜索加密技术和属性保序加密 (order-preserving encryption, OPE) 技术来保证数据安全性和支持一系列 SQL 查询。CryptDB 方案的主要加密构造是洋葱层加密, 洋葱层加密对数据库中表的属性列进行不同安全等级的加密。加密等级越高, 安全性越高, 功能性越弱; 加密等级越低, 安全性越弱, 功能性越强。CryptDB 方案的功能实现部分依赖一种具有实用性的工具 OPE, 第 1 个保序加密方案是 Popa 等^[19]在 2004 年的 ACM SIGMOD 会议上提出的。OPE 将部分属性列按照有序索引进行存储, 它允许加密值的顺序关系和明文的顺序关系保持一致, 可以实现范围查询和求最值的查询。OPE 技术虽然实用, 但也揭示了加密属性列的有序性和相等性。过去几年的研究工作^[20-22]表明, OPE 难以抵抗许多 LAAs。攻击者可以根据密文的相等关系和顺序关系, 结合辅助信息, 恢复出许多明文信息。这破坏了数据的隐私性。而基于 STE 构建的加密结构产生的泄露要等于或小于基于 PPE (包括

OPE) 构建的加密结构, 并且可以抵抗大多数已知的 LAAs。

2014 年, Naveed 等^[23]提出一种基于 STE 技术的盲存 (bloom filter index search of encrypted results, Blind Seer) 方案, 该方案利用 Yao 混淆电路^[24]和布隆过滤器^[25], 在此线性搜索时间内实现任意布尔公式的布尔查询、范围查询和等值查询, 但是 Blind Seer 不能实现连接查询。2016 年, Kamara 等^[5]提出基于 STE 技术的 SPX 方案, SPX 方案解决 SQL 查询的一大子类: 连接查询, 并且 SPX 构造不使用任何属性保护加密。SPX 构造实现连接查询的方法是将 SQL 查询转换为由选择、投影和叉乘算子表示的 SPC^[26]代数, 为 SPC 代数的不同操作生成不同的索引结构, 执行不同的搜索。虽然 SPX 是实用的, 但是仍然存在缺陷。一方面是 SPX 不能最优地处理相关链接查询, 另一方面是 SPX 方案的效率还不够高, 不足以产生一个与商业纯文本数据库管理为系统竞争的系统。为了突破 SPX 的限制, 2018 年, Codd 等^[27]提出优化的 SPX 加密方案——优化投影索引 (optimal projection index, OPX) 方案, OPX 可以最佳地处理任何形式的连接查询。

SPX 和 OPX 方案实现连接查询时, 索引表中存储 2 个表的连接行的对应关系元组。在最坏的情况下, 客户端和服务端有二次幂数量的行的通信量, 因为返回一个表某一行的同时, 也要返回另一个表匹配的连接行, 并且还会泄露整个连接列的相等模式, 即表中的两行在连接列的值相等。2021 年, Cash 等^[6]提出 PpSj 方案, 对 SPX 和 OPX 进行了扩展和改进, 该方案使用了部分预连接索引技术, 分别存储 2 个表可连接的行集, 而不是存储行之间的匹配关系, 具体的连接操作由客户端完成。这样的索引结构隐藏了相等模式。

然而 PpSj 方案也存在缺陷, 在实现布尔查询时, PpSj 方案采用过滤哈希集技术以求取 2 个集合的交集, 该过滤方法通过判断密钥和值的伪随机函数 (pseudo random function, PRF) 值是否存在于过滤哈希集中, 来判断值是否属于交集。然而, 在执行布尔查询时, 服务器可以通过记录每个关键字产生的密钥, 计算任意 2 个关键字之间的交集。因此, 这种过滤方法产生的泄露较大, 本文提出一种混合过滤器来减少此类泄露。

2 预备知识

2.1 符号说明

设 S 为一个集合, 在伪代码中用 $S \stackrel{U}{\leftarrow} x$ 表示 $S \leftarrow S \cup \{x\}$ 。用 $x \stackrel{s}{\leftarrow} X$ 表示从一个集合 X 中均匀随机抽取一个元素 x 。用 $x \leftarrow A$ 表示执行算法 A 并输出 x 。函数 $\text{negl}(k): \mathbf{N} \rightarrow \mathbf{R}$, 如果对于任意的正多项式 $p(\cdot)$, 都存在一个正整数 N , 对于所有 $k > N$, 使得 $\text{negl}(k): \frac{1}{p(k)}$ 成立, 就称 $\text{negl}(k)$ 是可忽略函数。

本文方案使用了字典和多映射。字典 D 由 n 个标签/值对 $\{(l_i, v_i)\}_{i \leq n}$ 组成, 其中每个 l_i 来自标签的全集 U , 每个值 v_i 来自值的全集 V 。用 $D[l_i] = v_i$ 表示根据标签获取值的操作, 如果 l_i 不存在, 就令 $D[l_i] = \emptyset$ 。多映射 MM 由 n 个标签/元组对 $\{(l_i, v_i)\}_{i \leq n}$ 组成, 用 $MM[l_i] = v_i$ 表示根据标签获取值元组的操作, 如果 l_i 不存在, 则 $MM[l_i] = \emptyset$ 。

2.2 关系数据库

关系数据库用 $DB = (T_1, \dots, T_n)$ 表示, T_i 代表一张二维表, 表中的每一行代表一条记录, 每一列代表记录的属性, 其中 $1 \leq i \leq n$ 。本文使用 $T_i.\text{Ats}$ 表示表 T_i 的所有属性列。对于行 $r \in T_i$, $\text{tbl}(r) = i$ 表示行 r 所在表的标识符为 i , $\text{rrk}(r)$ 表示行在表 T_i 中的位置。 $\chi(r) = (\text{tbl}(r), \text{rrk}(r))$ 唯一表示行 r 的坐标。令 $\chi(c, v) = \{(\text{tbl}(r), \text{rrk}(r)) | r[c] = v\}$, $\chi(c, v)$ 表示表 T_i 中第 c 列的值为 v 的所有行的集合。对于属性域相同的 2 个属性列 $c_1 \in T_1.\text{Ats}$ 和 $c_2 \in T_2.\text{Ats}$, 用 $\text{dom}(c_1) = \text{dom}(c_2)$

表示。由于数据库中的属性是独一无二的, 因此可以通过 $\text{getID}(\text{at}, \text{DB}) = \text{id}$ 来获取, 属性所在表的标识符 id , 其中 $\text{at} \in T_{\text{id}}.\text{Ats}$ 。

2.3 伪随机函数

定义 1^[28] 令 $F: \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ 为一个带密钥的函数集合, 用 $F: K \times X \rightarrow Y$ 表示, 其中 K 表示密钥空间, X 表示输入空间, Y 表示输出空间。如果对于任意概率多项式时间 (probabilistic polynomial time, PPT) 区分器 $\mathcal{D}$, 存在一个可忽略函数 negl , 使以下等式成立。

$$|\Pr[\mathcal{D}^{F(\cdot)}(1^n) = 1] - \Pr[\mathcal{D}^{f(\cdot)}(1^n) = 1]| \leq \text{negl}(n) \quad (1)$$

则称 F 是一个伪随机函数。其中密钥 $k \leftarrow \{0, 1\}^n$, f 是从 $\{0, 1\}^n \rightarrow \{0, 1\}^n$ 函数集合中均匀随机选择出来的。

2.4 结构化加密

本节给出结构化加密的语义和安全性。结构化加密根据响应结果是显示还是隐藏分为 2 种类型, 本文使用响应隐藏结构化加密构建索引。以下是响应隐藏结构化加密的形式化定义。

定义 2^[4] 响应隐藏结构化加密方案, $\Sigma = (\text{Setup}, \text{Token}, \text{Query}, \text{Dec})$ 由初始化算法 Setup 、令牌生成算法 Token 、搜索算法 Query 和解密算法 Dec 组成。

(1) $(K, \text{EDS}) \leftarrow \text{Setup}(1^k, \text{DS})$: 客户端输入一个安全参数 k 和一个明文数据结构 DS , 服务器无输入。客户端执行算法后输出密钥 K 和加密结构 EDS 分别由客户端和服务端保存。

(2) $\text{token} \leftarrow \text{Token}(K, q)$: 客户端以密钥 K 和查询 q 作为输入, 输出查询令牌 token 。

(3) $\{\emptyset, \text{ct}\} \leftarrow \text{Query}(\text{EDS}, \text{token})$: 查询协议, 服务器和客户端分别以加密结构 EDS 和令牌 token 为输入, 协议执行后输出 $\emptyset$ 或密文 ct 。

(4) $r \leftarrow \text{Dec}(K, \text{ct})$: 客户端输入解密密钥 K , 解密密文 ct , 得到最终的查询结果 r 。

对于结构化加密, 使用泄露函数 $\mathcal{L} = (\mathcal{L}_s, \mathcal{L}_q)$ 定义敌手获取信息的上限, 即结构化加密的安全性要求保证除设置泄露 $\mathcal{L}_s$ 和查询泄露 $\mathcal{L}_q$ 之外不会透露任何有关查询和底层数据结构的信息。为了形式化结构化加密的安全概念, 使

用了基于模拟的方法。

定义3^[4] 给定响应隐藏结构化加密方案 $\Sigma = (\text{Setup}, \text{Token}, \text{Query}, \text{Dec})$ ，进行以下的实验，其中 $\mathcal{A}$ 是自适应的任意 PPT 对手， S 是以泄露函数 $\mathcal{L}$ 为输入的模拟器， $\text{Real}_{\mathcal{A}}^{\Sigma}(k)$ 代表在真实世界执行的实验， $\text{Ideal}_{\mathcal{A}, S}^{\Sigma}(k)$ 是模拟器以泄露为输入模拟输出的实验，具体实验流程如下。

Real _{$\mathcal{A}$} ^{Σ} (k):

(1) 敌手 $\mathcal{A}$ 选择一个数据结构 DS 发送给挑战者 C 。

(2) 挑战者 C 执行数据结构 DS 初始化算法 $(K, \text{EDS}) \leftarrow \text{Setup}(1^k, \text{DS})$ ，并返回加密数据结构 EDS 给敌手 $\mathcal{A}$ 。

(3) 敌手 $\mathcal{A}$ 自适应地选择多项式数量的查询 $q_1, \dots, q_n$ 发送给挑战者 C ，其中 n 是关于 k 的多项式。对于 $1 \leq i \leq n$ ，挑战者 C 计算 $\text{token}_i \leftarrow \text{Token}(K, q_i)$ ，并将令牌 token_i 发送给敌手 $\mathcal{A}$ 。

(4) 敌手 $\mathcal{A}$ 输出一个比特 $b \in \{0, 1\}$ 。

Ideal _{$\mathcal{A}, S$} ^{Σ} (k):

(1) 敌手 $\mathcal{A}$ 选择一个数据结构 DS 发送给挑战者 C 。

(2) 模拟器 S 以挑战者 C 的设置泄露 $\mathcal{L}_S(\text{DS})$ 为输入，输出加密数据结构 EDS 给敌手 $\mathcal{A}$ 。

(3) 敌手 $\mathcal{A}$ 自适应选择多项式数量的查询 $q_1, \dots, q_n$ 发送给挑战者 C 。对于 $1 \leq i \leq n$ ，模拟器 S 接收来自挑战者 C 发送的一个元组 $(\emptyset, \mathcal{L}_Q(\text{DS}, q_i))$ ，并返回令牌 token 给敌手 $\mathcal{A}$ 。其中 $\text{DS}(q_i)$ 表示 q_i 的查询结果。

(4) 敌手 $\mathcal{A}$ 输出一个比特 $b \in \{0, 1\}$ 。

如果存在一个 PPT 模拟器 S ，对于所有 PPT 敌手 $\mathcal{A}$ 以下表达式成立：

$$|\Pr[\text{Real}_{\mathcal{A}}^{\Sigma}(k) = 1] - \Pr[\text{Ideal}_{\mathcal{A}, S}^{\Sigma}(k) = 1]| \leq \text{negl}(k) \quad (2)$$

则称是 Σ -自适应语义安全的。

2.5 结构化索引

2.5.1 等值查询的索引

给定一个数据库 $\text{DB} = (T_1, \dots, T_n)$ ，客户端首先创建一个多映射 MM，它将数据库中每个属性值映射到包含属性值的行坐标。即，对于 $1 \leq i \leq n$ ，列 $c \in T_i.\text{Ats}$ 的所有属性值 $v \in \text{dom}(c)$ ，

MM 存储标签/元组对 $(\langle c, v \rangle, \chi(c, v))$ 。

2.5.2 连接查询的索引

客户端创建多映射 MM 将连接属性列映射到连接行的行坐标。对于表 $T_1, T_2 \in \text{DB}$ ， $c_1 \in T_1.\text{Ats}$ ， $c_2 \in T_2.\text{Ats}$ ， $r_1 \in T_1$ ， $r_2 \in T_2$ ，有 $\text{dom}(c_1) = \text{dom}(c_2)$ 且 $r_1(c_1) = r_2(c_2)$ ，多映射 MM 存储两组标签/元组对， $(c_1, \chi(r_1))$ 和 $(c_2, \chi(r_2))$ 。这种多映射存储在最坏的情况下，响应结果的行数是线性的，因为响应结果是两组行集而不是它们的叉乘。

3 混合过滤器的构造

PpSj 方案利用 OXT 方案的过滤哈希集技术来实现布尔查询。过滤哈希集技术的基本流程如下：首先，客户端构造一个过滤哈希集 (HashSet, HS)，用于存储标签/值元组的对应关系，并将其发送给服务器；在进行过滤操作时，客户端输入一个由标签 label 生成的密钥，服务器输入过滤哈希集 HS 和一个待过滤的值元组，并判断密钥和值的 PRF 值是否在 HS 中，如果在则保留，如果不在则丢弃。

通过仔细思考发现，这种算法容易受到以下攻击：当服务器执行 $I_1 \wedge I_2$ 查询时，它会接收客户端发送的标签 I_1 的 PRF 密钥 $K = F(K_f, I_1)$ ，并将其保存。然后，服务器输入 I_2 标签对应的 value 值元组，通过测试 $F(K, \text{value})$ 是否属于过滤哈希集来学习 $I_1 \wedge I_2$ 的响应量。若服务器进行更多查询，它还可以使用标签 I_1 的密钥 K 跟任何标签对应的值元组执行过滤操作。当服务器获取到所有标签的键时，就可以获得任意 2 个标签求交集的响应量，其泄露的基本查询集为 $\{I_i \wedge I_j\}_{i \neq j}$ 。这种攻击是可能的，因为服务器可以根据客户端发送的密钥计算 $F(K, \cdot)$ 。这意味着服务器有能力使用保存的键执行多个查询，从而推断出标签之间的交集关系，并推断出敏感信息。

为了减少此种泄露，本文研究了几种过滤器方案。Kamara 等^[8]提出的 IEX 过滤器结合了 OXT 和 Blind Seer 几个良好的特性，是第 1 个非交互式加密布尔多映射，它能够处理任意布尔查询，能实现最坏情况下的次线性搜索时间和最优通信复杂度。IEX 泄露的基本查询集是 $\{I_i \wedge I_j\}_{i \neq j, 1}$ ，该泄露是 PpSj 方案泄露的子集。但是，IEX 在服

服务器存储任意2个关键字的交集, 占用大量的服务器存储空间。IEX实现布尔查询流程如图1所示。

由古宜平等^[9]提出的EXT过滤器是一个支持合取形式布尔查询的高效可搜索对称加密方案。该方案采用2轮客户端与服务器通信来实现布尔查询, 通信开销与第1个关键字对应的文档数和关键字个数相关, 服务器只需要维护一张索引表(Index)和一张过滤表(HashSet)。EXT实现布尔查询流程如图2所示。通过IEX和EXT实现布尔查询的流程对比显示, IEX方案的令牌大小只与查询 q 的关键字个数相关, 与关键字对应的值的数量无关, 因此, IEX方案的通信开销小于EXT方案的通信开销。

通过对多种过滤方案的研究, 我们努力寻找

权衡隐私保护和性能需求的解决方案。受到IEX和EXT方案的启发, 提出一种混合过滤方案, 该方案在执行布尔查询时启发式地选择适合的过滤器类型。为了控制通信开销, 客户端预设了一个带宽限制 L , 并根据关键字对应的值的数量进行带宽预估。由于值的数量是影响带宽的主要因素, 并且每个值会被填充到相同长度, 因此, 在构建索引时, 可以对每个关键字进行带宽预估。具体而言, 按递增顺序对每个关键字对应值的数量进行排序, 如果值的数量超过带宽限制 L , 就采用IEX构建索引; 否则, 采用EXT构建索引。同时, 客户端会记录使用IEX构建索引的关键字集合IEXkeys, 当用户发起布尔查询时, 客户端会检查关键字集合IEXkeys, 如果布尔查询表达式中存在一个关键字属于IEXkeys, 说明此时是

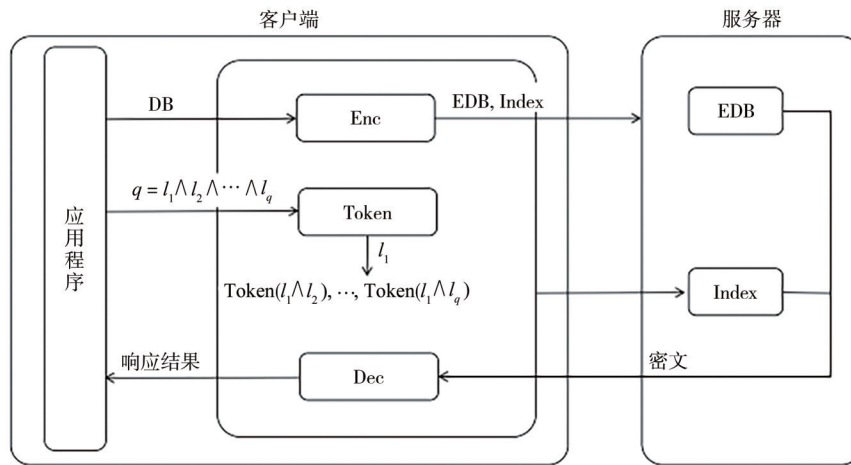


图1 IEX实现布尔查询流程
Figure 1 IEX implements Boolean query flow

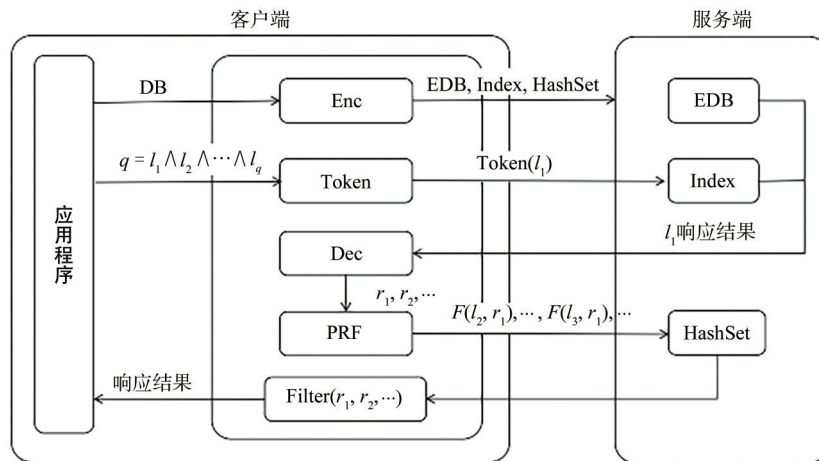


图2 EXT实现布尔查询流程
Figure 2 EXT implements Boolean query flow

最优带宽，可以选择生成 IEX 查询令牌；否则，选择生成 EXT 查询令牌。混合过滤器流程如图3所示。这一方案的关键在于根据带宽限制和关键字对应值的数量来动态选择使用 IEX 索引 (IEX-Index) 或 EXT 索引 (EXTIndex)。通过这种灵活性和优化的索引选择策略高效地处理布尔查询，能在保护数据隐私的同时控制通信开销和减少服务器存储负担。

为了更好地理解混合过滤器的构造，下面给出混合过滤器算法的描述。

(1) 客户端输入数据库 DB，并预设一个阈值 L ，根据阈值将关键字分成 $labels_1$ 和 $labels_2$ 两个部分。

(2) 对 $labels_1$ 的关键字采用 EXT 方案构建索引。算法首先创建本地多映射 MM_g ，它将关键字映射到关键字所在行，并为关键字和行生成 PRF 值存储在哈希表 SET 中。

(3) 对 $labels_2$ 的关键字采用 IEX 方案构建索引。算法首先创建本地多映射 MM_l ，它存储 $labels_2$ 中关键字与任意关键字的交集。

(4) 在查询阶段，客户端首先输入关键字个数为 q 的查询 $l_1 \wedge l_2 \wedge \dots \wedge l_q$ ，然后判断是否存在关键字属于关键字集合 IEXkeys，若存在，则生成 q 个查询令牌；若不存在，则需要生成 $q * \min(|l_i|)$ 个查询令牌。其中 $\min(|l_i|)$ 表示关键字集合中对应的行集最少的关键字 l_i 。

(5) 服务端获取令牌后，对相应的索引结构

进行搜索得到密文，并返回给客户端进行解密。

根据算法描述，在存储方面，仅本文存储了部分关键字与其他关键字的交集，这显著降低了服务器的存储空间。在通信开销方面，本文仅对阈值 L 以下的关键字使用了 EXT 构建索引，这也有效减少了通信开销。

4 关系数据库加密方案

本节将详细描述关系数据库加密方案 MFEDB，该方案基于 STE 和混合过滤器设计，用于支持在加密关系数据库上的 SQL 查询。简单地，它采用自适应安全的响应隐藏多映射加密方案加密索引表和响应隐藏的字典加密方案加密数据表。该方案的基本流程是：首先，客户端输入一个明文数据库，并根据该数据库构建一个存储标签/值元组的多映射 MM；然后，客户端使用 MM 作为输入，输出一个 IEX 和 EXT 的混合索引结构；最后，使用结构化加密方案对索引结构进行加密，由客户端保存加密密钥的同时，将加密索引发送给服务器。在查询期间，客户端为 SQL 查询生成查询令牌并发送给服务器，服务器接收到令牌后，在加密索引中执行检索操作，并将检索结果响应给客户端。

关系数据库加密方案 MFEDB = (Setup、Token、Search、Resolve)，由 Setup 算法、Token 算法、Search 算法和 Resolve 算法组成，以下是算法的具体描述。

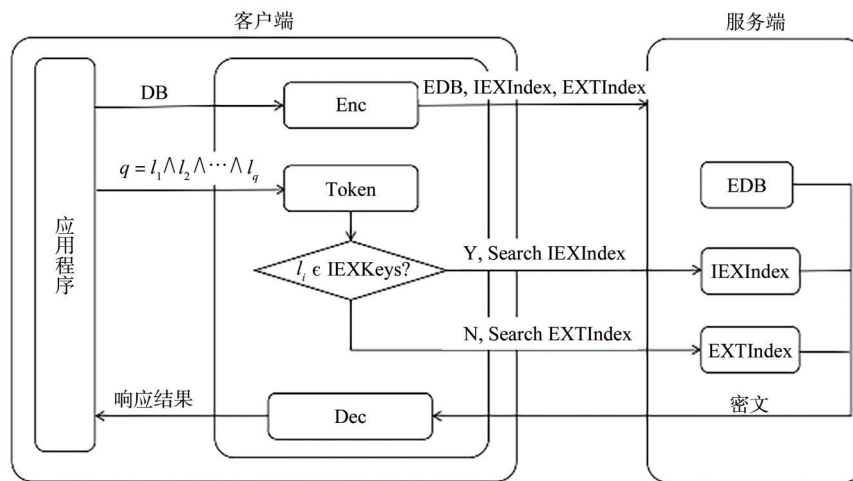


图3 混合过滤器流程
Figure 3 Flow of hybrid filter

4.1 Setup算法

在执行Setup算法之前, 先将SQL查询转化为对关键字的布尔查询。将SQL查询分为多表查询和单表查询2种类型。

针对单表查询进行布尔转换的方法如下: 例如, 查询 q 的查询语句是 $\text{select } * \text{ from } T_1 \text{ where } at_1 = x_1 \text{ and } at_2 = x_2$, 其中 $T_1 \in \text{DB}$, $at_1, at_2 \in T_1$ 。由于查询只涉及一个表, 所以直接提取条件转换成布尔查询。查询 q 转换为 $(\text{select}, at_1, x_1) \wedge (\text{select}, at_2, x_2)$ 。针对多表查询进行布尔转换的方法如下: 例如, 查询 q 的查询语句是 $\text{select } * \text{ from } T_1, T_2 \text{ where } T_1.at_1 = T_2.at_3 \text{ and } T_1.at_2 = x_1 \text{ and } T_2.at_4 = x_2$ 。其中 $T_1, T_2 \in \text{DB}$, $at_1, at_2 \in T_1$, $at_3, at_4 \in T_2$ 。由于涉及多个表, 所以需要将多表查询转换成对2个表的单表查询。查询 q 可以转换成 $(\text{join}, at_1, at_3) \wedge (\text{select}, at_2, x_1)$ 和 $(\text{join}, at_3, at_1) \wedge (\text{select}, at_4, x_2)$ 。通过将多表查询转为2个单表查询, 可以简化查询过程并更好地处理多个表之间的关联关系。这种转换使得每个表的查询可以独立执行, 并生成每个表的结果集。

如果单表查询中存在范围查询, 情况有所不同。在执行范围查询时, 需要将范围查询分解成最佳范围集合, 这些集合的并集将构成查询的范围。例如, 查询 q 的查询语句是 $\text{select } * \text{ from } T_1 \text{ where } at_1 = x_1 \text{ and } a \leq at_2 \leq d$, 其中 $T_1 \in \text{DB}$, $at_1, at_2 \in T_1$ 。假设范围 $[a, d]$ 的最佳范围集合是 $[a, b]$ 和 $[c, d]$, 其中, $a \leq b < c \leq d$, 查询 q 转化成 $(\text{select}, at_1, x_1) \wedge ((\text{range}, at_2, [a, b]) \vee (\text{range}, at_2, [c, d]))$ 。令 $S_1 = (\text{select}, at_1, x_1) \wedge (\text{range}, at_2, [a, b])$, $S_2 = (\text{select}, at_1, x_1) \wedge (\text{range}, at_2, [c, d])$, 此时, 查询 q 变成了对2个布尔查询的结果求并集, 即 $S_1 \cup S_2$ 。在计算并集时, 去重是一个非常重要的步骤。考虑2个范围查询的结果集之间的互斥性, 可以推断 S_1 中不可能出现 S_2 中的元素, 即 $S_1 \cap S_2 = \emptyset$ 。因此, 在计算并集时, 服务器可以直接将2个结果集合并, 不必对合并后的结果集进行重复元素的检查和消除, 从而节省了计算资源和时间。这种特性的存在对于处理大规模数据和复杂查询比较有益。

当SQL查询完成布尔转换后, 通过计算2个

或多个标签对应行集的交集, 就可以得到SQL查询结果集。因此, 首先执行SetIndex算法构建多映射MM, 它将标签映射到行令牌集合, 行令牌是加密行的唯一标识, 数据库DB所有行的行令牌由EncRows算法生成。多映射MM中存在等值查询标签、连接查询标签和范围查询标签的映射关系, 等值查询标签和连接查询标签的映射已经在第3节中说明, 将范围查询标签映射到范围对应的行令牌集合的过程需要结合Range算法。

4.1.1 EncRows算法

$(K_R, \text{ED}, \text{TK}) \leftarrow \text{EncRows}(\text{DB})$: 该算法由客户端执行, 客户端输入安全参数 λ 和数据库 $\text{DB} = (T_1, \dots, T_n)$, 输出加密字典ED和令牌字典TK, TK的主要作用是使用行令牌唯一标识加密行。EncRows算法如算法1所示, 具体流程如下。

(1) 客户端创建一个字典 D , 将行坐标 $\chi(r)$ 映射到行 r 。

(2) 客户端使用响应隐藏结构化加密方案 Σ_{DX} 对字典 D 进行加密, 并保存加密密钥 K_R 。加密完成后, 将加密字典ED发送给服务器。同时, 客户端创建一个令牌字典TK, 将行坐标 $\chi(r)$ 映射到相应的行令牌 rt 。行令牌 rt 在后续的多映射构建过程中将作为关键字对应的值。

算法1 EncRows算法

输入 λ, DB

输出 $K_R, \text{ED}, \text{TK}$

初始化一个空字典 D

$K_e \xleftarrow{\$} \{0,1\}^\lambda$

for $T \in \text{DB}$ do

for $r \in T$ do

$D[\chi(r)] \leftarrow \text{Enc}(K_e, r)$

end for

end for

$(K_R, \text{ED}) \leftarrow \Sigma_{\text{DX}}.\text{Setup}(1^\lambda, D)$

for $T \in \text{DB}$ do

for $r \in T$ do

$l \leftarrow \chi(r)$

$\text{TK}[l] \leftarrow \Sigma_{\text{DX}}.\text{Token}(K_R, l)$

end for

end for

4.1.2 Range 算法

Range 算法用来实现范围查询，范围查询的索引构建如图4所示。根据数值型属性列的属性域大小，自下向上构建一棵二叉树，对树中的每个节点进行二进制编码，并将数值型属性值表示为前缀编码存储在叶子节点上。例如，层高为 m 的二叉树，数值型属性值 x 的前缀编码表示为 $b_1b_2\cdots b_m$ 。完成二叉树的构建后，客户端根据二叉树列出包含属性值 x 的所有前缀编码，这些前缀编码被称为属性值 x 的前缀族，表示为 $\{b_1b_2\cdots b_m, b_1b_2\cdots b_{m-1}*, \cdots, b_1\cdots*, *\cdots*\}$ ，前缀族中每个前缀编码都表示一个范围。例如，4位的数字3的前缀编码是0011，它的前缀族是 $\{0011, 001*, 00**, 0***, ****\}$ ，其中，001*表示范围 $[0010, 0011]$ 。客户端得到每个属性值对应的前缀族后，创建一个多映射MM，将前缀编码映射到包含属性值的行坐标。这种方法所需的最大存储空间是二叉树节点的总数，较于存储包含该属性值的所有范围，存储包含关键字的前缀集合能够很大地减少服务器存储。索引构建完成后，客户端接收用户发来的查询请求，并为用户给定的查询范围寻找最佳范围覆盖^[29]，即覆盖该范围的最佳前缀集合，这些集合的并集就是用户给定的查询范围。例如，用户给定范围 $[1, 5]$ ，它的最佳的前缀集合是 $\{0001, 001*, 010*\}$ 。客户端对每个前缀范围执行单关键字查询，并计算所有查询结果的并集得到最终结果。通过使用最佳前缀集合，能够快速而准确地在多映射MM中找到符合查询范围的行坐标，提高查询效率和响应速度。

$MR \leftarrow \text{Range}(\text{DB})$ ：该算法由客户端执行，客户端输入一个数据库DB，输出一个多映射

MR，存储每个数值型关键字对应的前缀族。Range 算法如算法2所示，具体流程如下。

(1) 客户端构建一棵逻辑二叉树，并将所有数值型属性值转化为二进制编码。

(2) 客户端根据关键字的二进制编码生成前缀族。

(3) 创建一个多映射MR，将关键字坐标映射到关键字对应的前缀族。

算法2 Range 算法

输入 DB

输出 MR

for $T \in \text{DB}$ do

for $r \in T$ do

for $\text{at} \in T.\text{Ats}$ do

if 属性值是整数 then

$r[\text{at}] = (b_1b_2\cdots b_m)_2$

$\text{MR}[(\text{at}, \chi(r))] \leftarrow \{b_1\cdots b_m, \cdots, *\cdots*\}$

end if

end for

end for

end for

4.1.3 SetIndex 算法

$\text{MM} \leftarrow \text{SetIndex}(\text{DB}, \alpha, \text{TK}, \text{MR})$ ：该算法由客户端执行，客户端输入数据库DB、多映射MR、令牌字典TK和连接属性元组 α ，输出一个多映射MM。SetIndex 算法如算法3所示，具体流程如下。

(1) 客户端初始化一个空的多映射MM，它存储3种不同类型的标签，每种标签对应不同的SQL操作，并将标签映射到行令牌。

(2) select 标签表示选择查询，MM将属性值 $r[\text{at}]$ 映射到包含属性值的行令牌rt。range

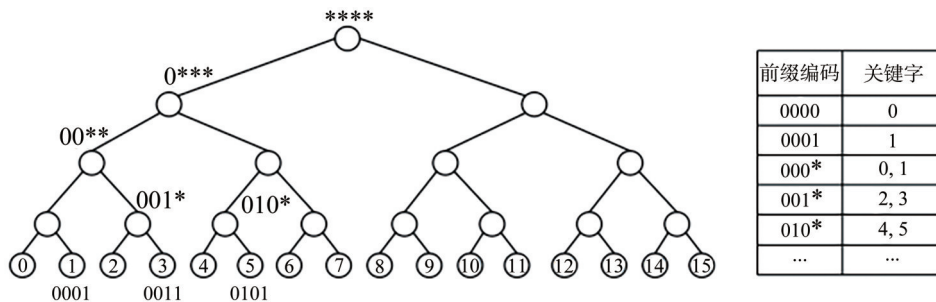


图4 范围查询的索引构建
Figure 4 Index construction for range query

标签表示范围查询, MM 将前缀编码 $\text{prefixes}[i]$ 映射到该前缀编码包含的数值型关键字所在行令牌 rt 。join 标签表示连接查询, MM 将 2 个表的连接属性 $(\text{at}_1, \text{at}_2)$ 映射到连接行令牌 rt 。值得注意的是, 客户端在对连接属性做映射时, 要分别存储 2 个表的行令牌, 以标签中的连接属性的顺序区别 2 个表的行令牌。

算法 3 SetIndex 算法

输入 DB, α , TK, MR

输出 MM

初始化一个空的多映射 MM

```

for  $T \in \text{DB}$  do
  for  $r \in T$  do
     $\text{rt} \leftarrow \text{TK}[\chi(r)]$ 
    for  $\text{at} \in T.\text{A}ts$  do
       $\text{MM}[(\text{select}, \text{at}, r[\text{at}])] \xleftarrow{\cup} \text{rt}$ 
      if  $\text{MR}[(\text{at}, \chi(r))] \neq \emptyset$  then
         $\text{prefixes} = \text{MR}[(\text{at}, \chi(r))]$ 
        for  $i = 0, \dots, \text{prefixes.length} - 1$  do
           $\text{MM}[(\text{range}, \text{at}, \text{prefixes}[i])] \xleftarrow{\cup} \text{rt}$ 
        end for
      end for
    end if
     $T_1 \leftarrow \text{getID}(\text{at}_1); T_2 \leftarrow \text{getID}(\text{at}_2)$ 
    end for
    for  $r_1, r_2 \in T_1 \bowtie_{\text{at}_1, \text{at}_2} T_2$  do
      end for
      for  $i = 1, 2$  do
        end for
      end for
      if  $i = 1$  then
         $\text{rt}_1 \leftarrow \text{TK}[\chi(r_1)]$ 
         $\text{MM}[(\text{join}, \text{at}_1, \text{at}_2)] \xleftarrow{\cup} \text{rt}_1$ 
      end if
      if  $i = 2$  then
         $\text{rt}_2 \leftarrow \text{TK}[\chi(r_2)]$ 
         $\text{MM}[(\text{join}, \text{at}_2, \text{at}_1)] \xleftarrow{\cup} \text{rt}_2$ 
      end if
    end for
  end for
end for

```

4.1.4 Setup 算法

$(K, \text{EMM}, \text{EDX}, \text{SET}) \leftarrow \text{Setup}(\lambda, \text{DB})$:

该算法由客户端执行, 客户端输入安全参数 λ 、数据库 DB, 输出密钥 K 保存在客户端, 输出加密多映射 EMM、加密字典 EDX 和一个哈希集合 SET 存储在服务端。Setup 算法流程如算法 4 所示。

(1) 客户端从密钥空间 $\{0, 1\}^\lambda$ 随机均匀抽样生成 3 个密钥并初始化多映射、字典和哈希表。

(2) 客户端对数据库 DB 中的所有标签 labels 对应的值元组长度排序, 并给定一个阈值 L , 将值元组长度小于 L 的标签归入 labels_1 , 将值元组长度大于 L 的标签归入 labels_2 , 并将 labels_2 包含的标签存储在客户端。标签 labels 包括 select 标签、range 标签和 join 标签, 但在后续操作中, 将各种标签统一抽象为标签 l , 不区分每个标签的查询类型。

(3) 对于 labels_1 类的标签 l , 采用 EXT 方案构建索引。算法首先创建本地多映射 MM_g , MM_g 将标签 l 映射到标签对应的加密行令牌 ert ; 然后为每对标签/行令牌生成 PRF 值, 并存储在 SET 哈希表中。

(4) 对于 labels_2 类的标签 l , 采用 IEX 方案构建索引。对于每个标签 l , 算法创建一个局部多映射 MM_l , MM_l 将标签 $l' \in \text{labels}$ 映射到行令牌 rt 的 tag_{rt} 值, 其中, $\text{rt} \in \text{MM}[l] \cap \text{MM}[l']$ 。本地多映射 MM_l 的目的是快速找出同时满足标签 l 和 l' 的行令牌 rt 。多映射 MM_l 构建完成后, 使用 Σ_{MM} 进行加密得到 EMM_l 。然后, 算法创建一个字典 DX, 将标签 l 映射到标签对应的加密多映射 EMM_l 。值得注意的是, DX 中存储 EMM_l 的指针, 而非存储整个 EMM_l 结构。

(5) 客户端用 Σ_{MM} 、 Σ_{DX} 分别对 MM_g 和 DX 加密, 得到加密多映射 EMM_g 和加密字典 EDX。

算法 4 Setup 算法

输入 λ, DB

输出 $K, \text{EMM}, \text{EDX}, \text{SET}$

$K_1, K_2, K_3 \xleftarrow{s} \{0, 1\}^\lambda$

初始化一个字典 DX

初始化多映射 MM_g, MM_l

```

初始化一个集合 SET
MM ← SetIndex (DB, α, TK, MR)
for  $l \in \text{labels}_1$  do
    for  $rt \in \text{MM}[l]$  do
         $ert \leftarrow \text{Enc}(F(K_1, l), rt)$ 
         $\text{MM}_g[l] \leftarrow \text{ert}$ 
         $\text{SET} \leftarrow F(K_2, l \parallel rt)$ 
    end for
end for
for  $l \in \text{labels}_2$  do
    for  $l' \in \text{labels}$  do
        if  $\text{MM}[l] \cap \text{MM}[l'] \neq \emptyset$  then
            for  $rt \in \text{MM}[l] \cap \text{MM}[l']$  do
                 $\text{tag}_{rt} \leftarrow \text{Enc}(F(K_3, l), rt)$ 
                 $\text{MM}_l[l'] \leftarrow \text{tag}_{rt}$ 
            end for
             $(K_l, \text{EMM}_l) \leftarrow \Sigma_{\text{MM}}.\text{Setup}(l^1, \text{MM}_l)$ 
        end if
         $\text{DX}[l] = \text{EMM}_l$ 
    end for
     $(K_g, \text{EMM}_g) \leftarrow \Sigma_{\text{MM}}.\text{Setup}(l^1, \text{MM}_g)$ 
end for
 $(K_d, \text{EDX}) \leftarrow \Sigma_{\text{DX}}.\text{Setup}(l^1, \text{DX})$ 
 $K = (K_1, K_2, K_3, K_g, K_d, \{K_l\}_{l \in \text{labels}_2})$ 

```

4.2 Token 算法

$(\text{tk}) \leftarrow \text{Token}(K, q)$: 该算法由客户端执行, 客户端输入密钥 K 和查询 q , 输出令牌元组 tk 。Token 算法如算法 5 所示, 具体流程如下。

(1) 客户端判断查询 q 中是否存在一个标签 l 属于 labels_2 。

(2) 若存在标签 l 属于 labels_2 , 客户端生成字典令牌 tk_l , 允许服务器查询加密字典 EDX 以获取标签 l 的本地多映射 EMM_l 。然后, 客户端生成本地多映射 EMM_l 的令牌 tk_l , 允许服务器查询 l 的加密本地多映射 EMM_l , 以获取同时满足 l 和 l_i 的行令牌 rt 的 tag_{rt} 值。

(3) 如果不存在属于 labels_2 的标签 l , 客户端将根据标签 l_1 生成本地多映射 EMM_g 的令牌 tk_1 , 允许服务器查询加密本地多映射 EMM_g , 获取 $\text{MM}_g[l]$ 的加密行令牌 ert 。

值得注意的是, 若查询 q 是多表查询, 需要将多表查询转化为多个单表查询, 因此, 客户端会产生多个布尔查询公式, 对应多个令牌元组, 服务器需要执行多次搜索协议。

算法 5 Token 算法

输入 $K, q = (l_1 \wedge l_2 \wedge \dots \wedge l_q)$

输出 tk

```

if  $l \in \text{labels}_2$  then
     $\text{tk}_l \leftarrow \Sigma_{\text{DX}}.\text{Token}(K_d, l)$ 
    for  $1 \leq i \leq q$  do
    end for
     $\text{tk}_i \leftarrow \Sigma_{\text{MM}}.\text{Token}(K_l, l_i)$ 
    set  $\text{tk} = (\text{tk}_l, \text{tk}_1, \dots, \text{tk}_q)$ 
else
     $\text{tk}_1 \leftarrow \Sigma_{\text{MM}}.\text{Token}(K_l, l_1)$ 
    set  $\text{tk} = (\text{tk}_1)$ 
end if

```

4.3 Search 算法

$R \leftarrow \text{Search}(\text{tk}, \text{EMM}_g, \text{EDX}, \text{SET})$: 搜索协议, 该协议在客户端和服务端之间交互进行, 客户端输入令牌元组 tk , 服务器输入加密多映射 EMM_g 、加密字典 EDX 和过滤哈希表 SET , 服务器输出加密行集合 SET , 输出响应结果集 R 。Search 算法如算法 6 所示, 具体流程如下。

(1) 服务器根据令牌元组 tk 执行相应的搜索协议。

(2) 如果令牌元组 $\text{tk} = (\text{tk}_l, \text{tk}_1, \dots, \text{tk}_q)$, 服务器执行 IEX 搜索协议。首先, 使用 tk_l 查询加密字典 EDX 获取 l 的本地多映射 EMM_l ; 然后, 使用 tk_i 查询 EMM_l 获取同时满足 l 和 l_i 的行令牌 rt 的 tag_{rt} 值, 并将其保存在集合 R_l 中。处理完所有令牌 tk_i 后, 服务器将持有集合 R_l 到 R_q ; 最后, 服务器对所有集合求交集得到最终结果集 R 并返回给客户端。

(3) 如果令牌元组 $\text{tk} = (\text{tk}_1)$, 服务器执行 EXT 搜索协议。

1) 服务器使用 tk_1 查询多映射 EMM_g 获取加密行令牌元组, 并将其返回给客户端。

2) 客户端对加密行令牌元组 ert 进行解密, 得到行令牌元组 rt 并缓存在本地。行令牌 rt 结合每个标签 l_i 生成 PRF 值, 并存储在多映射 X_{tag}

中。Xtag 将下标映射到行令牌的 PRF 值。当客户端处理完所有行令牌 tk，将 Xtag 返回给服务器。

3) 服务器判断 Xtag_i 的所有 PRF 值是否存在于 SET 集合中，如果对于任意一个 PRF 值都属于 SET 集合，则将该下标存储在响应结果集 R 中，否则丢弃该下标。当服务器处理完所有 PRF 值后，将 R 响应给客户端。

算法 6 Search 算法

输入 tk, EMM_g, EDX, SET

输出 R

if tk = (tk₁, tk₂, ..., tk_q) then

EMM_i ← Σ_{DX}.Get(EDX, tk_i)

for 1 ≤ i ≤ q do

R_i ← Σ_{MM}.Get(EMM_i, tk_i)

end for

R ← R₁ ∩ ... ∩ R_q

end if

if tk = (tk₁) then

服务器:

ert ← Σ_{MM}.Get(EMM_g, tk₁)

将 ert 向量发送给客户端

客户端:

rt ← Dec(F(K₁, l₁), ert)

for rt ∈ rt do

for 2 ≤ i ≤ q do

Xtag[i - 2] ← ^UF(K₂, l_i || rt)

end for

将 Xtag 发送给服务器

end for

服务器:

for 0 ≤ i ≤ |ert| - 1 do

if Xtag[i] ∈ SET then

R ← ^Ui

4.4 Resolve 算法

4.4.1 DecRows 算法

(r₁, ..., r_n) ← DecRows(K_e, ED, (rt₁, ..., rt_n)): 服务器输入行令牌集合 (rt₁, ..., rt_n) 和加密字典 ED，客户端输入解密密钥 K_e。DecRows 算法如算法 7 所示，具体流程如下。

(1) 服务器根据每个行令牌 rt_i 在 ED 中进行

检索，并将检索结果反馈给客户端。

(2) 客户端根据密钥 K_e 对加密行进行解密，还原加密行为原始的明文行。

算法 7 DecRows 算法

输入 K_e, ED, (rt₁, ..., rt_n)

输出 (r₁, ..., r_n)

服务器:

for 1 ≤ i ≤ n do

C_i ← Σ_{DX}.Get(ED, rt_i)

end for

将 (C₁, ..., C_n) 发送给客户端

客户端:

for 1 ≤ i ≤ n do

r_i ← Dec(K_e, C_i)

end for

4.4.2 Resolve 算法

(r₁, ..., r_n) ← Resolve(R, K, K_e, ED): 客户端输入响应结果集 R、密钥 K 和加密行的密钥 K_e，服务器输入加密字典 ED。输出明文结果集 (r₁, ..., r_n)。Resolve 算法如算法 8 所示，具体流程如下。

(1) 客户端初始化集合 res 为空集合。

(2) 客户端对结果集 R 进行遍历。若 R 中存储的是 tag_{rt} 值，则对每个 tag_{rt} 值进行解密得到行令牌 rt，并将其存储在 res 集合中。

(3) 若 R 中存储的是下标值，客户端从缓存中取出 rt 数组，获取数组中对应下标的 rt 值，并将其存储在 res 集合中。

(4) 最后，客户端将 res 集合返回给服务器，服务器和客户端执行 DecRows 算法获取最终明文结果集。如果查询 q 是多表查询，客户端需要对最终的明文结果集进行连接操作。

算法 8 Resolve 算法

输入 R, K, K_e, ED

输出 (r₁, ..., r_n)

初始化一个集合 res

if tag_{rt} ∈ R then

for tag_{rt} ∈ R do

rt ← Dec(F(K₃, l), tag_{rt})

res ← ^Urt

```

    end for
  else
    for  $i \in R$ ,  $rt_i \in \mathbf{rt}$  do
       $\text{res} \stackrel{\cup}{\leftarrow} rt_i$ 
    end for
  end if
   $(r_1, \dots, r_n) \leftarrow \text{DecRows}(K_R, \text{ED}, \text{res})$ 

```

5 安全性分析

本节将证明 MFEDB 方案是 $(\mathcal{L}_s, \mathcal{L}_Q)$ 自适应安全的, 并给出泄露函数的形式化描述, 其中 $\mathcal{L}_s$ 和 $\mathcal{L}_Q$ 的定义将在后面给出。客户端输入一个多映射 MM 和一个连接查询序列 $Q = (q^1, \dots, q^q)$ 。其中, 第 i 个查询的查询形式是 $q^i = (l_i^1 \wedge \dots \wedge l_i^{q_i})$, q_i 是查询 q^i 的标签数量。定义泄露函数为 $\mathcal{L} = (\mathcal{L}_s, \mathcal{L}_Q)$ 。

在描述泄露之前, 先介绍重复模式的概念。给定一个长度为 N 的对象实例的序列, 重复模式将由 N 个整数序列组成, 每出现一个对象实例, 就在重复模式中记录遇到该实例的第 1 个索引。例如, 如果对象实例的序列是 a, b, c, a, d, b , 那么重复模式就是 $1, 2, 3, 1, 5, 2$ 。

(1) 设置泄露: 设置泄露是对手在执行任何操作之前就了解到的信息, 包括加密全局多映射和加密字典的设置泄露, 定义为 $\mathcal{L}_s = (\mathcal{L}_s(\text{DB}), \mathcal{L}_s(\text{MM}_g), \mathcal{L}_s(\text{DX}), \mathcal{L}_s(\text{SET}))$, 假设使用标准的最优时间多映射和字典加密方案, setup 算法泄露了数据库 DB 的大小、本地多映射 MM_g 的大小、 SET 集合的大小以及存储在字典中的本地多映射 MM_l 的总大小。 SET 的每个元素都是 PRF 值, SET 大小与 MM_g 的大小相同, 攻击者没有学习到超过 SET 大小的信息。

(2) 查询泄露: 查询泄露是对手通过令牌获取的信息和对手使用令牌在加密多映射中搜索泄露的信息, 定义为 $\mathcal{L}_Q = (\#\text{labels}(Q), \mathcal{L}_Q^{\text{lex}}, \mathcal{L}_Q^{\text{ext}})$ 。首先, 客户端搜索第 i 个查询, 产生的查询令牌的个数泄露了第 i 个查询的标签数, 用 $\#\text{labels}(q_1, \dots, q_{|Q|})$ 来表示。然后, 服务器使用查询令牌执行搜索, 若搜索 IEX 索引, 会产生 $\mathcal{L}_Q^{\text{lex}} = (\mathcal{L}_Q(\text{DX}, l), \mathcal{L}_s(\text{MM}_l), \mathcal{L}_Q(\text{MM}_l), \text{tagRP})$ 搜

索泄露。其中, $\mathcal{L}_Q(\text{DX}, l)$ 指搜索加密字典产生的搜索泄露, $\mathcal{L}_s(\text{MM}_l)$ 指泄露本地多映射 MM_l 的大小。 $\mathcal{L}_Q(\text{MM}_l)$ 由搜索模式和访问模式组成, 指前 i 个查询是否搜索了相同的查询和标记, 还包括 $\text{MM}_l[l_j]$ 的响应大小, $j \in [1, q_i]$ 。 tagRP 指 tag 的重复模式。假设 2 个查询共享第 1 个标签, 则对应的查询令牌中包含相同的 EMM_l 令牌。查询后的 2 个结果集有相同的 tag , 因此会泄露 tag 重复模式。 tagRP 是一个数组, 其长度等于 tag 数量, 每个条目对应于 tag 第 1 次被看到的索引。若搜索 EXT 索引, 会产生 $\mathcal{L}_Q^{\text{ext}} = (\mathcal{L}_Q(\text{MM}_g, l), \text{XtagRP})$ 搜索泄露。 $\mathcal{L}_Q(\text{MM}_g, l)$ 会泄露 $\text{MM}_g[l] \wedge \text{MM}_g[l_j]$ 的大小, $j \in [1, q_i]$, 并且会泄露 Xtag 的重复模式, XtagRP 与 tagRP 定义类似。下面给出方案的安全性定理。

定理 1 如果 Σ_{DX} 是 $(\mathcal{L}_s^{\text{dx}}, \mathcal{L}_Q^{\text{dx}})$ 自适应安全的, Σ_{MM} 是 $(\mathcal{L}_s^{\text{mm}}, \mathcal{L}_Q^{\text{mm}})$ 自适应安全的, 对称加密算法 SKE 满足选择明文攻击不可区分性 (indistinguishability against chosen-plaintext attacks, IND-CPA) 安全, F 是安全的伪随机函数, 则方案是 $(\mathcal{L}_s, \mathcal{L}_Q)$ 自适应安全的加密多映射方案。

证明 通过构造一系列游戏来证明对于所有概率多项式时间对手 $\mathcal{A}$, $\text{Real}_{\mathcal{A}, s}(\lambda)$ 输出 1 的概率与 $\text{Ideal}_{\mathcal{A}, s}(\lambda)$ 输出 1 的概率差异可以忽略不计。这一系列游戏中, 相邻的游戏之间都存在微小的差异, 但是相邻的游戏之间都不可区分。

Game0: Game0 为真实游戏, 输入数据库 DB 和查询序列 $Q = (q^1, \dots, q^q)$ 。

Game1: 使用模拟器 S_{mm} 接收设置泄露 $\mathcal{L}_s(\text{MM}_g)$ 并输出 EMM_g , 模拟器 S_{dx} 接收设置泄露 $\mathcal{L}_s(\text{DX})$ 并输出 EDX , 模拟器 S_{set} 接收设置泄露 $\mathcal{L}_s(\text{SET})$ 并输出 SET 。可以推断 Game2 与 Game1 是无法区分的。

Game2: 判断查询 q^i 中是否存在一个标签 l 属于 labels_2 。若存在, 则黑盒调用模拟器 S_{lex} 。因为不同查询所产生的 tok 是否相同, 与被选中的标签 l 相关, 因此需要以查询 q^i 的泄露 $\mathcal{L}_Q^{\text{lex}}(\text{EDX}, q^i)$ 作为输入, 模拟器 S_{lex} 输出令牌 tok 。可以推断 Game2 与 Game1 是无法区分的。

Game3: 若不存在任何一个标签属于 la-

bels₂, 则黑盒调用模拟器 S_{ext} , 以查询 q^i 的泄露 $\mathcal{L}_Q^{\text{ext}}(\text{EMM}_g, \text{SET}, q^i)$ 作为输入, 模拟器 S_{ext} 输出令牌 tok。可以推断 Game3 与 Game2 是无法区分的。

设定一个模拟器 S , 这个模拟器 S 只输入本文方案定义的泄露函数而不会输入任何真实的数据和查询, 然后它将输出一个模拟的视图。模拟器 S 的算法如算法 9 所示。

算法 9 模拟器 S

输入 $\mathcal{L}_S, \mathcal{L}_Q, Q = (q^1, \dots, q^q)$

输出 $\text{EMM}_g, \text{EDX}, \text{SET}, \text{tok}$

$\text{EMM}_g \leftarrow S_{\text{mm}}(\mathcal{L}_S(\text{MM}_g))$

$\text{EDX} \leftarrow S_{\text{dx}}(\mathcal{L}_S(\text{DX}))$

$\text{SET} \leftarrow S_{\text{set}}(\mathcal{L}_S(\text{SET}))$

for $1 \leq i \leq q$ do

 if $l \in \text{labels}_2$ then

$\text{tok} \leftarrow S_{\text{icx}}(\mathcal{L}_Q^{\text{icx}}(\text{EDX}, q^i))$

 else

$\text{tok} \leftarrow S_{\text{ext}}(\mathcal{L}_Q^{\text{ext}}(\text{EMM}_g, \text{SET}, q^i))$

 end if

end for

在算法 9 中, 由模拟器 S_{icx} 和模拟器 S_{ext} 模拟 tok 的生成。当执行 IEX 搜索协议时, 对于查询 q^i , 模拟器 S_{icx} 输入查询泄露:

$$\begin{aligned} \mathcal{L}_Q^{\text{icx}}(\text{EDX}, q^i) = & (\mathcal{L}_Q(\text{DX}, l), \mathcal{L}_S(\text{MM}_l), \\ & \mathcal{L}_Q(\text{MM}_l, l_1), \mathcal{L}_Q(\text{MM}_l, l_2), \dots, \mathcal{L}_Q(\text{MM}_l, l_{q_i}), \\ & \text{tagRP}_i(\text{EDX}, q^i)) \end{aligned} \quad (3)$$

IEX 搜索协议首先搜索加密字典得到加密多映射 EMM_l , 因此会产生加密字典搜索泄露 $\mathcal{L}_Q(\text{DX}, l)$ 和加密多映射的设置泄露 $\mathcal{L}_S(\text{MM}_l)$ 。接着, 多次搜索加密多映射 EMM_l 会产生搜索泄露 $\mathcal{L}_Q(\text{MM}_l)$ 以及 tag 重复模式的泄露。模拟器 S_{icx} 会依据 $\mathcal{L}_Q^{\text{icx}}$ 模拟出 tok, 在真实游戏中, $\text{tok} = (\text{tk}_l, \text{tk}_1, \dots, \text{tk}_{q_i})$, 在模拟过程中, tk_l 由 $\text{tk}_l \leftarrow S_{\text{dx}}(\mathcal{L}_Q(\text{DX}, l), \text{EMM}_l)$ 模拟生成, 其中 EMM_l 由 $\text{EMM}_l \leftarrow S_{\text{mm}}(\mathcal{L}_S(\text{MM}_l))$ 模拟生成。 tk_j 由 $\text{tk}_j \leftarrow S_{\text{mm}}(\mathcal{L}_Q(\text{MM}_l, l_j), (\text{tag}_{\text{rt}})_{\text{rt} \in \text{MM}[l] \cap \text{MM}[l_j]})$ 模拟生成, 其中 $j \in [1, q_i]$ 。

当执行 EXT 搜索协议时, 对于查询 q^i , 模拟

器 S_{ext} 输入查询泄露, 如式 (4) 所示。

$$\begin{aligned} \mathcal{L}_Q^{\text{ext}}(\text{EMM}_g, \text{SET}, q^i) = & (\mathcal{L}_Q(\text{MM}_g, l), \text{XtagRP}_i(\text{SET}, q^i)) \end{aligned} \quad (4)$$

EXT 搜索协议首先搜索加密多映射 EMM_g , 得到标签 l 的响应结果, 因此会产生搜索泄露 $\mathcal{L}_Q(\text{MM}_g, l)$ 。接着, 在客户端生成 Xtag, 不会产生泄露。但是, 服务端利用 Xtag 执行过滤时, 会泄露 Xtag 的重复模式 $\text{XtagRP}_i(\text{SET}, q^i)$ 。模拟器 S_{ext} 会依据 $\mathcal{L}_Q^{\text{ext}}$ 模拟出 tok, 在真实游戏中, $\text{tok} = (\text{tk}_l, \text{Xtag})$, 在模拟过程中, tk_l 由 $\text{tk}_l \leftarrow S_{\text{mm}}(\mathcal{L}_Q(\text{MM}_g, l))$ 模拟生成, Xtag 由 $\text{Xtag} \leftarrow S_{\text{set}}((\text{Xtag}_{\text{rt}})_{\text{rt} \in \text{MM}[l] \cap \text{MM}[l_j]})$ 模拟生成, 其中 $j \in [1, q_i]$ 。

由 Game3 生成的视图和模拟器 S 生成的视图是完全无法区分的, 关联上述一系列游戏, 有:

$$|\Pr[\text{Real}_{\mathcal{A}}(\lambda) = 1] - \Pr[\text{Ideal}_{\mathcal{A}, S}(\lambda) = 1]| \text{negl}(\lambda) \quad (5)$$

值得注意的是, 在 Resolve 算法的最后一步需要从加密的数据库 ED 中查询每个行令牌对应的密文。不同的查询可能返回相同的密文, 这就导致了不同查询的密文集合之间存在交集。这种泄露没有在安全性分析中进行讨论, 也没有在 Cash 的 PpSj 方案中进行讨论分析, 因此, 本文暂不对这种泄露进行分析。

6 实验分析

本节将从搜索效率、服务器存储开销和令牌大小方面与 PpSj 方案、IEX 方案和 EXT 方案进行比较分析。本文实验采用的 SSE 框架是使用 Java 实现的 Clusion 框架, 该框架的加密使用了 Bouncy Castle 库, 本文实验在 CTR 模式中使用具有 16 byte 密钥的 AES 作为对称加密方案, 并使用 HMAC-SHA256 作为具有 16 byte 的 PRF。实验在具有 6 核的 CPU (Intel (R) Xeon (R) CPU E5-2643 v4 @ 3.40 GHz)、128 GB 内存、500 GB 存储空间机器上进行。

对于输入数据集, 本文实验使用的数据集是 Chicago's Open Data Portal 和 Mysql Sakila 的所有数据表, 总共包括 30 个关系表、175 个属性列和 219 992 行记录。在根据所有关系表生成多映射的过程中, 过滤掉多映射中一些无意义的连接标签或者选择标签。例如, 对 film 表属性 descrip-

tion的等值查询。对于连接查询的生成,只考虑Chicago关系表中的连接和Sakila关系表中的连接,不考虑2个数据集之间的连接。在实验过程中,为了模拟用户发起的SQL查询,客户端随机生成5组SQL查询,每一组查询的查询条件数量不同,并且每组有10个SQL查询,丢弃不返回行的查询以及返回结果集过大的查询。

6.1 泄露比较

4种方案的泄露差异表现在查询泄露方面,因为采用了不同的过滤方法,所以产生不同的泄露。PpSj方案的过滤方案通过给定服务器任意关键字的键来过滤,服务器通过保存的键执行多次过滤,导致泄露任意2个关键字的交集。IEX方案和MFEDB方案只给出第1个关键字与其他关键字的令牌,因此只会泄露第1个关键字与其他关键字的交集。EXT方案在客户端生成Xtag,只让服务器做过滤,因此只泄露了第1个关键字的响应量和第1个关键字与其他关键字的交集。

6.2 搜索效率

服务器端的搜索操作需查询加密索引表和执行过滤操作,因此搜索效率关键取决于处理索引的效率和执行过滤的效率。方案搜索性能对比如图5所示,搜索效率对比如表2所示,展示了4种方案处理不同组查询的搜索效率,每组实验测量服务器执行具有2~10个查询条件的SQL语句的平均时间,不计算客户端的网络延迟和令牌生成时间。

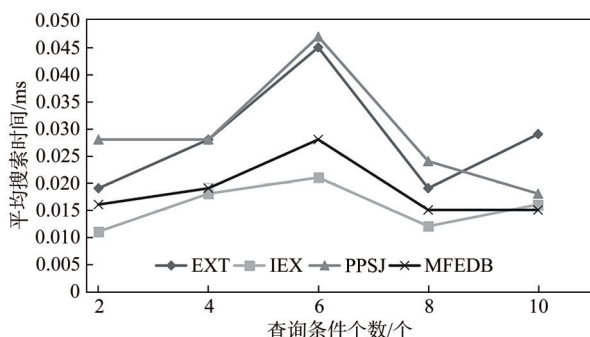


图5 方案搜索性能对比
Figure 5 Comparison of search performance

根据表2、图5和图6可以观察到,搜索性能的趋势与查询响应量的趋势相似,这是因为响应量较大,则说明需要执行过滤的行数较多,而搜

索效率关键在于执行过滤的效率,过滤需要的时间越多,搜索需要时间越多。EXT方案和PpSj方案的搜索时间较大,这是因为它们需要生成标签和值的PRF值,并判断每个PRF值是否存在于过滤集合SET中,在PRF值较多的情况下,对搜索时间消耗较大。不仅如此,EXT方案还存在客户端和服务器的两轮交互,交互过程中有解密操作,该操作所花费的时间与选取的标签对应的行令牌数量成正比。IEX方案服务端只需要根据令牌从多映射中取出tag值,不需要在哈希表中搜索每个PRF值,因此搜索效率较高。MFEDB方案根据标签对应的值数量动态选择过滤方式,因此搜索效率处于IEX方案和其他方案中间,并与IEX方案差距不大。总体来说,MFEDB方案的搜索效率比EXT方案提升了16%~38%,比PpSj方案提升了17%~40%。

表2 搜索性能对比
Table 2 Comparison of search performance

条件数量	搜索时间/ms			
	EXT	IEX	PpSj	MFEDB
2	0.019	0.011	0.028	0.016
4	0.028	0.018	0.028	0.019
6	0.045	0.021	0.047	0.028
8	0.019	0.012	0.024	0.015
10	0.029	0.016	0.018	0.015

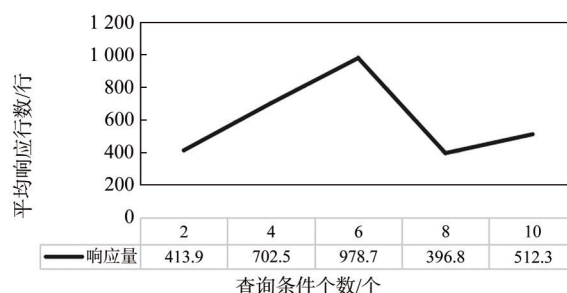


图6 查询响应量
Figure 6 Query response quantity

6.3 存储开销

不同方案加密存储2个数据库的服务器存储开销如表3所示。对于具有5万行的Sakila数据库, IEX方案所需服务器存储空间比EXT方案所需服务器存储空间大18倍多,这是因为IEX方案需要存储任意2个标签的交集。相比IEX方案, MFEDB方案减少了66%的服务器存储空间。

表3 服务器存储开销
Table 3 Server storage overhead

方案	存储开销
EXT	51.6
IEX	966.5
PpSj	51.6
MFEDB	328.9

6.4 令牌大小

令牌大小如表4所示, 在SQL查询条件个数为6时, EXT方案的令牌大小较大, 这是因为该组查询所选择的标签对应的行令牌数量较大, EXT方案的令牌大小与标签对应的行令牌数量成正比, 受行令牌数量影响较大, 而IEX方案和PpSj方案的令牌大小只与查询条件个数相关, 不受行令牌数量影响, 因此令牌大小较小, 并随着条件数量增多, 令牌大小逐渐增大。MFEDB方案会根据标签对应的行令牌数量动态选择方案生成令牌, 受行令牌数量影响较小, 令牌大小相比EXT方案减少了78%。

表4 令牌大小
Table 4 Token size

条件数量	令牌大小/KB			
	EXT	IEX	PpSj	MFEDB
2	0.07	0.20	0.16	0.06
4	6.90	0.30	0.21	1.22
6	14.60	0.44	0.31	3.55
8	7.30	0.56	0.38	3.52
10	11.20	0.71	0.46	0.27

7 结束语

本文提出一种混合过滤技术, 并基于该技术提出一种改进的关系数据库加密方案MFEDB。该方案与IEX方案相比, 减少了约66%的服务器存储空间。与EXT方案相比, 减少了约78%的令牌大小, 同时提升了约16%~38%的搜索效率。与PpSj相比, 减少了查询泄露, 同时提升了约17%~40%的搜索效率。

面对更加复杂的查询, MFEDB方案使用PpSj的存储连接行的方式, 其响应量可能会比SPX、OPE方案更大, 如何进一步降低MFEDB的响应量可作为进一步研究方向。

参考文献:

- [1] BINDSCHAEDLER V, GRUBBS P, CASH D, et al. The Tao of inference in privacy-protected databases[J]. Proceedings of the VLDB Endowment, 2018, 11(11): 1715-1728.
- [2] BLACKSTONE L, KAMARA S, MOATAZ T. Revisiting leakage abuse attacks[C]//Proceedings 2020 Network and Distributed. System Security Symposium. Reston: Internet Society, 2020: 1175.
- [3] DEMERTZIS I, PAPADOPOULOS D, PAPAMANTHOU C, et al. SEAL: attack mitigation for encrypted databases via adjustable leakage[C]//Proceedings of the 29th USENIX Conference on Security Symposium. New York: ACM Press, 2020: 2433-2450.
- [4] CHASE M, KAMARA S. Structured encryption and controlled disclosure[C]//Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2010: 577-594.
- [5] KAMARA S, MOATAZ T. SQL on structurally-encrypted databases[C]//Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security. Cham: Springer, 2018: 149-180.
- [6] CASH D, NG R, RIVKIN A. Improved structured encryption for SQL databases via hybrid indexing[C]//Proceedings of the International Conference on Applied Cryptography and Network Security. Cham: Springer, 2021: 480-510.
- [7] CASH D, JARECKI S, JUTLA C, et al. Highly-scalable searchable symmetric encryption with support for Boolean queries[C]//Proceedings of the Annual Cryptology Conference. Berlin: Springer, 2013: 353-373.
- [8] KAMARA S, MOATAZ T. Boolean searchable symmetric encryption with worst-case sub-linear complexity[C]//Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham: Springer, 2017: 94-124.
- [9] 古宜平, 马昌社. 支持联合查询的高效可搜索对称加密方案[J]. 华南师范大学学报(自然科学版), 2020, 52(3): 120-128.
GU Y P, MA C S. Efficient searchable symmetric encryption for conjunctive queries[J]. Journal of South China Normal University (Natural Science Edition), 2020, 52(3): 120-128.
- [10] SONG D X, WAGNER D, PERRIG A. Practical techniques for searches on encrypted data[C]//Proceedings of IEEE Symposium on Security and Privacy. S&P. Piscataway: IEEE Press, 2000: 44-55.
- [11] CURTMOLA R, GARAY J, KAMARA S, et al. Searchable symmetric encryption: improved definitions and efficient constructions[J]. Journal of Computer Security, 2011, 19(5): 895-934.
- [12] CASH D, JAEGER J, JARECKI S, et al. Dynamic searchable encryption in very-large databases: data structures and implementation[C]//Proceedings of the 2014 Network and Distributed System Security Symposium. Reston: Internet Society, 2014: 853.
- [13] DEMERTZIS I, TALAPATRA R, PAPAMANTHOU C. Efficient searchable encryption through compression[J]. Proceed-

- ings of the VLDB Endowment, 2018, 11(11): 1729-1741.
- [14] KAMARA S, PAPAMANTHOU C, ROEDER T. Dynamic searchable symmetric encryption[C]//Proceedings of the 2012 ACM conference on Computer and communications security. New York: ACM press, 2012: 965-976.
- [15] MIERS I, MOHASSEL P. IO-DSSE: scaling dynamic searchable encryption to millions of indexes by improving locality[J]. Cryptology ePrint Archive. 2017: 830.
- [16] STEFANOV E, PAPAMANTHOU C, SHI E. Practical dynamic searchable encryption with small leakage[C]//Proceedings of the 2014 Network and Distributed System Security Symposium. Reston: Internet Society, 2014: 832.
- [17] BOST R, MINAUD B, OHRIMENKO O. Forward and backward private searchable encryption from constrained cryptographic primitives[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM press, 2017: 1465-1482.
- [18] POPA R A, REDFIELD C M S, ZELDOVICH N, et al. CryptDB: protecting confidentiality with encrypted query processing[C]//Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. New York: ACM Press, 2011: 85-100.
- [19] AGRAWAL R, KIERNAN J, SRIKANT R, et al. Order preserving encryption for numeric data[C]//Proceedings of the 2004 ACM SIGMOD international conference on Management of data. New York: ACM press, 2004: 563-574.
- [20] GRUBBS P, SEKNIQI K, BINDSCHAEDLER V, et al. Leakage-abuse attacks against order-revealing encryption[C]//Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2017: 655-672.
- [21] DURAK F B, DUBUISSON T M, CASH D. What else is revealed by order-revealing encryption? [C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM press, 2016: 1155-1166.
- [22] NAVEED M, KAMARA S, WRIGHT C V. Inference attacks on property-preserving encrypted databases[C]//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. New York: ACM press, 2015: 644-655.
- [23] PAPPAS V, KRELL F, VO B, et al. Blind seer: a scalable private DBMS[C]//Proceedings of the 2014 IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2014: 359-374.
- [24] BELLARE M, HOANG V T, ROGAWAY P. Foundations of garbled circuits[C]//Proceedings of the 2012 ACM conference on Computer and communications security. New York: ACM Press, 2012: 784-796.
- [25] BLOOM B H. Space/time trade-offs in hash coding with allowable errors[J]. Communications of the ACM, 1970, 13(7): 422-426.
- [26] CODD E F. A relational model of data for large shared data banks[J]. Communications of the ACM, 1970, 13(6): 377-387.
- [27] KAMARA S, MOATAZ T, ZDONIK S, et al. An optimal relational database encryption scheme[J]. IACR Cryptol EPrint Arch, 2020: 274.
- [28] BELLARE M, CANETTI R, KRAWCZYK H. Pseudorandom functions revisited: the cascade construction and its concrete security[C]//Proceedings of the 37th Conference on Foundations of Computer Science. Piscataway: IEEE Press, 2002: 514-523.
- [29] DEMERTZIS I, PAPADOPOULOS S, PAPAPETROU O, et al. Practical private range search in depth[J]. ACM Transactions on Database Systems, 2018, 43(1): 1-52.

[作者简介]



苏雨晨（2000—），女，湖南益阳人，华南师范大学硕士生，主要研究方向为可搜索加密、加密数据库。



马昌社（1974—），男，湖南邵阳人，博士，华南师范大学教授，主要研究方向为加密数据隐私计算、应用密码技术。