

隐藏共现模式的可搜索对称加密方案

何丽梅

(华南师范大学人工智能学院, 广东 佛山 528225)

摘要: 可搜索对称加密(SSE)是一种在大数据和云计算环境下解决数据安全与隐私保护问题的关键技术,其允许用户在不解密的情况下对加密数据进行安全查询。大多数现有的SSE方案为保证方案的高效性会泄露访问模式,访问模式泄露了查询令牌与被匹配文档之间的关系,进而会揭露共现模式。共现模式为每一对关键字在一个文档集中出现概率所组成的概率矩阵,其是IKK攻击这一类推理攻击的基础。因此,共现模式的泄露已经严重威胁到加密数据安全。针对这一问题,提出一种隐藏共现模式的文档填充算法,并基于该算法设计了可有效隐藏共现模式的可搜索对称加密方案。安全性分析表明,该方案是安全且有效的,同时基于真实数据集进行的实验结果也表明该方案能够有效防止共现模式泄露。

关键词: 可搜索对称加密;访问模式;共现模式;IKK攻击;推理攻击

DOI: 10.11907/rjdk.241168

开放科学(资源服务)标识码(OSID):



中图分类号: TP309

文献标识码: A

文章编号: 1672-7800(2025)004-0123-07

The Searchable Symmetric Encryption Scheme with Hiding Co-occurrence Pattern

HE Limei

(School of Artificial Intelligence, South China Normal University, Foshan 528225, China)

Abstract: Searchable symmetric encryption is a crucial technology addressing data security and privacy protection problems in big data and cloud computing environments. It enables users to securely query encrypted data without the need for decryption. Most existing SSE schemes leak access pattern to ensure the efficiency of the scheme. Access pattern reveals the relationship between query tokens and matched documents, which in turn exposes co-occurrence pattern. The co-occurrence pattern is the number of common documents of any pair of keywords. It is the basis of inference attacks among which IKK attack is the typical one. As a result, the leakage of co-occurrence pattern has a practical impact on the privacy of searchable symmetric encryption schemes. In response to this issue, a padding algorithm to hide the co-occurrence pattern based on which a searchable symmetric encryption scheme that hides co-occurrence pattern is proposed. Security analysis shows that this scheme is secure and effective, and experimental results using real datasets also indicate that this scheme can effectively prevent the leakage of co-occurrence pattern.

Key Words: searchable symmetric encryption; access pattern; co-occurrence pattern; IKK attack; inference attacks

0 引言

可搜索对称加密(Searchable Symmetric Encryption, SSE)允许客户端在服务器中存储加密的数据库,并对其安全地执行搜索^[1]。客户端一般首先在本地生成一个加密的数据库和查询索引,并将其同时外包给服务器,之后发起关键字查询生成搜索令牌,用搜索令牌从加密的查询索引表中查询对应加密文档数据的索引,服务器使用索引检

索加密数据表以获取相匹配的文档并返回客户端,最终客户端在本地对返回的一系列加密文档进行解密,得到明文文档。

近年来,许多敌手利用共现模式的泄露进行攻击,如IKK攻击、计数攻击等推理攻击。考虑一个文档集,其中单词“New”、“York”和“Yankees”在任何文档中出现的概率比其他单词高。用户发送一系列查询请求,敌手在查询结束后获得一个令牌列表和文档集,通过计算每一对令牌一起出现的文档数量来计算这些被查询关键字的共现率。

收稿日期: 2024-02-26

扫描二维码阅读全文:



基金项目: 国家自然科学基金项目(61672243)

作者简介: 何丽梅(1997-),女,华南师范大学人工智能学院硕士研究生,研究方向为可搜索对称加密。

由于“New”、“York”和“Yankees”几个单词在文档中出现的概率较高,因此敌手可以分离出代表关键字集“New”、“York”和“Yankees”的令牌。然而,“New”和“York”两个词通常结合在一起指代一个著名城市,因而其同时出现在文件中的概率会比“New”和“Yankees”或“York”和“Yankees”更高。这一重要的观察结果使得敌手能够唯一地识别出“Yankees”这个词的令牌。此外,如果敌手还了解到关键字“New”的令牌标识,即可推断出关键字“York”的令牌。因此,即使只有少量的先验知识,也可能导致加密查询信息泄漏。该例子只是考虑了由少数查询显示的信息,但具有强大计算能力的敌手可以截取多个查询和响应对,将访问模式与其所拥有的关于文档内容的每一条信息结合起来,并发起强有力的攻击。

针对上述问题,本文首先给出共现模式、查询共现模式和共现模式隐藏的形式化定义,然后提出一种隐藏共现模式的文档填充算法,并基于该算法设计了可有效隐藏共现模式的可搜索对称加密(Searchable Symmetric Encryption, SSE)方案,最后分析并证明了该方案的安全性,并将其与未进行共现模式隐藏的SSE方案进行性能对比分析。实验结果表明,该方案能有效防止共现模式泄漏,并且填充后的性能牺牲不大,具有实用性。

1 相关工作

1996年,Goldreich等^[2]提出ORAM(Oblivious Random Access Machine)的概念,并用ORAM构造了一个基础方案。该方案能保护访问模式,但需要巨大的计算和通信开销。2000年,Song等^[1]首次提出SSE方案,该方案每次查询需遍历整个密文文件,查询效率低下,同时容易受到统计攻击的威胁。2003年,Goh^[3]给出一种SSE所需的安全定义——IND-CKA (Indistinguishability Against Chosen-Keyword Attacks),并提出基于布隆过滤器的SSE方案。2005年,Chang等^[4]给出一个基于模拟的SSE定义,但该定义未考虑敌手如何自适应生成查询的情况。为解决自适应生成查询的问题,2006年,Curtmola等^[5]提出基于倒排索引构造的SSE方案,其安全定义允许敌手自适应地生成查询,但并不能有效保证隐藏任何特定用户的访问模式。后续工作对SSE方案的功能性、安全性和运行效率作了进一步研究^[6-9]。

SSE通常会泄漏访问模式,拥有关于底层文档集关键字分布知识背景的敌手可利用访问模式推断出查询的有效信息。2012年,Islam等^[10]首次利用共现模式发起了IKK攻击,通过模拟退火方法尝试根据返回文档的模式将查询与关键字匹配。2015年,David等^[11]提出依赖于共现模式和响应长度模式的计数攻击。Zhang等^[12]利用响应标识(即访问模式)进行注入攻击,该标识显示了响应长度、共现模式以及每个查询相关联的文档标识符。因

此,IKK和计数攻击可应用于任何泄漏响应标识的构造。2016年,Pouliot等^[13]提出一种改进的IKK攻击方案,该方案利用了共现模式的泄漏,并依赖于图匹配算法。然而,随着泄漏扰动参数的增加,IKK攻击效率显著降低。2021年,Gui等^[14]针对现有泄漏扰动技术的不足,使用统计建模和模拟退火相结合实现推理式泄漏滥用攻击。在该模型中,敌手可用的辅助信息独立于采样,但在统计上接近目标数据库。以上攻击方法都是基于共现模式泄漏进行的。

2018年,Chen等^[15]提出一种在数据外包之前混淆数据库索引的方案来保护SSE中的访问模式。在该方案中,服务器只能学到模糊的访问模式,难以攻击客户端的隐私。但当使用相同关键字进行查询时,该方案总是产生相同的混淆访问模式,从而泄漏搜索模式。2021年,Shang等^[16]提出一种基于内积谓词加密的OSSE(Oblivious SSE)方案,该方案使关键字与文件相关联,产生假阳性或假阴性的结果。

综上所述,目前大多数能够完全隐藏访问模式的方法都是基于ORAM进行设计的方案。这些方案通过连续洗牌和重新加密数据实现访问模式的完全隐藏,但开销大。访问模式的泄漏会导致IKK类推理攻击和计数攻击,这些攻击都利用了共现模式的泄漏。因此,对于保护共现模式的研究是十分必要的。本文提出一种隐藏共现模式的文档填充算法,通过该算法对原始文档进行填充,使任意两个不同关键字对应的文档交集数量相同,再运用可搜索对称加密技术便可有效防止共现模式的泄漏。

2 预备知识

2.1 伪随机函数

定义 1 令 $F: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ 表示一个函数族,其中 $\{0, 1\}^k$ 代表密钥空间,定义域和值域都为 $\{0, 1\}^n$ 。如果对于任意的概率多项式时间(Probabilistic Polynomial Time, PPT)区分器 D ,存在一个可忽略的函数 $\text{negl}(\lambda)$,且满足式(1):

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

则称 F 是一个伪随机函数(Pseudorandom Function, PRF)。其中,密钥 K 从 $\{0, 1\}^k$ 中均匀随机选择,函数 f 从将 n 比特字符串映射到 n 比特字符串的函数集合中随机均匀选择^[17]。

2.2 IND-CPA 安全的对称加密方案

定义 2 对称加密方案(Symmetric Key Encryption, SKE) $\Sigma = (\text{Gen}, \text{Enc}, \text{Dec})$ 包含 3 个算法,给定安全参数 λ 和任意的 PPT 敌手 A ,定义方案 Σ 的 IND-CPA 安全性实验 $\text{Exp}_{A, \Sigma}^{\text{IND-CPA}}(\lambda)$ 。实验流程如下: ①挑战者调用密钥生成算法 $\text{Gen}(1^\lambda)$ 产生一个随机密钥 K ; ②敌手 A 选取两个相同大小的明文消息 m_0 和 m_1 发送给挑战者; ③挑战者随机选取

一个比特位 $b \leftarrow \{0, 1\}$, 计算密文 $c \leftarrow \text{Enc}_k(m_b)$, 并将密文 c 发送给敌手 A ; ④ A 根据返回的密文 c , 输出判断值的比特位 b' ; ⑤ 如果 $b' = b$, 则实验输出 1, 否则输出 0。

如果对任意 PPT 敌手 A , 存在一个可忽略的函数 $\text{negl}(\lambda)$, 使得:

$$\Pr [\text{Exp}_{A, \Sigma}^{\text{IND-CPA}}(\lambda) = 1] \leq 1/2 + \text{negl}(\lambda) \quad (2)$$

则称对称加密方案 Σ 具有 IND-CPA 安全性^[19]。

2.3 可搜索对称加密

定义 3 一个可搜索对称加密方案 $\text{SSE} = (\text{Setup}, \text{Token}, \text{Search}, \text{Resolve})$ 包含 4 个算法, 其中有: ① $(K, I, C) = \text{Setup}(\lambda, D)$ 。输入安全参数 λ 和明文文件集 $D = (D_1, D_2, \dots, D_n)$, $D_i \in 2^W$, 输出密钥 K 、文件索引 I 和密文文件集 $C = (C_1, C_2, \dots, C_n)$; ② $T_w = \text{Token}(K, w)$ 。输入密钥 K 和关键词 w , 输出关键词 w 对应的搜索令牌 T_w ; ③ $ED(w) = \text{Search}(I, T_w)$ 。输入文件索引 I 和搜索令牌 T_w 进行查找, 输出密文文件集 $ED(w)$; ④ $D(w) = \text{Resolve}(K, ED(w))$ 。用密钥 K 解密返回的密文文件集 $ED(w)$, 生成相应的明文文件集 $D(w)$ 。

对于 $\forall \lambda \in N, n \in Z, w \in W, D = (D_1, D_2, \dots, D_n)$ 以及 $(K, I, C) = \text{Setup}(\lambda, D)$, 都有 $\text{Search}(I, \text{Token}(K, w)) = ED(W)$ 和 $\text{Resolve}(K, ED(w)) = D(w)$ 成立, 则称 SSE 方案是正确的^[20]。

2.4 安全定义

假设 $L = (L_{\text{Setup}}, L_{\text{Search}})$ 表示 SSE 方案在初始化阶段和查询阶段的具体信息泄露, 考虑如下真实游戏和理想游戏, 其中 A 代表任意的 PPT 敌手, S 为模拟器, $k \in N$ 为安全参数。

(1) 真实游戏 $\text{Real}_A^{\Pi}(k)$ 。首先, 敌手 A 选择数据库 DB , 该游戏运行 $\text{Setup}(DB)$ 算法生成查询索引 I 并发送给 A 。其次, A 自适应地选择一系列查询 q , 该游戏运行 $\text{Token}(q)$ 算法生成搜索令牌 T_q 并发送给 A 。再次, 该游戏运行 $\text{Search}(I, T_q)$, 并将运行的全部结果都发送给 A 。最后, 敌手 A 输出一个比特位 b 。

(2) 理想游戏 $\text{Ideal}_{A, S}^{\Pi}$ 。首先, 敌手 A 选择数据库 DB , 该游戏运行算法 $S(L_{\text{Setup}}(DB))$ 生成查询索引 I 后发送给 A 。其次, A 自适应地选择一系列查询 q , 运行 $S(L_{\text{Search}}(q, DB))$ 并将结果发送给 A 。最后, 敌手 A 输出一个比特位 b 。

如果对于任意的 PPT 敌手 A , 存在一个高效的模拟器 S 和一个可忽略的函数 $\text{negl}(\lambda)$, 使得:

$$|\Pr [\text{Real}_A^{\Pi}(k) = 1] - \Pr [\text{Ideal}_{A, S}^{\Pi}(k) = 1]| \leq \text{negl}(k) \quad (3)$$

则称 SSE 是 L -自适应安全的。

2.5 共现模式

定义 4 关键字集 W 上文档集 D 的共现模式定义为:

$$\text{COR}_D = \begin{bmatrix} 1 & n_{12} & n_{13} & \dots & n_{1m} \\ n_{21} & 1 & n_{23} & \dots & n_{2m} \\ \dots & \dots & \dots & \dots & \dots \\ n_{m1} & n_{m2} & n_{m3} & \dots & 1 \end{bmatrix} \quad (4)$$

其中, n_{ij} 表示包含关键字 w_i 和包含关键字 w_j 的文档交集个数, 即 $n_{ij} = |D(w_i) \cap D(w_j)|$, $i, j = 1, 2, \dots, m$ 且 $i \neq j$ 。

定义 5 给定一个查询序列 $Q = (q_1, q_2, q_3, \dots, q_t)$, 假设这些查询所访问的关键字按字典顺序排列为 $w_1, w_2, \dots, w_s$, 在文档集 D 上定义查询序列 Q 的共现模式为:

$$\text{COR}_{Q, D} = \begin{bmatrix} 1 & n_{12} & n_{13} & \dots & n_{1s} \\ n_{21} & 1 & n_{23} & \dots & n_{2s} \\ \dots & \dots & \dots & \dots & \dots \\ n_{s1} & n_{s2} & n_{s3} & \dots & 1 \end{bmatrix} \quad (5)$$

其中, n_{ij} 表示包含关键字 w_i 和包含关键字 w_j 的文档交集个数, 即 $n_{ij} = |D(w_i) \cap D(w_j)|$, $i, j = 1, 2, \dots, s$ 且 $i \neq j$ 。

比如一个文档集合 $DB = \{(d_1, \{kw_1, kw_2, kw_3\}), (d_2, \{kw_1, kw_2\}), (d_3, \{kw_3\})\}$, 将在文档集合 DB 上查询 q 的泄漏表示为 $L_{\text{Search}}(q, DB) = \{d_i | kw(q) \in kw(d_i), (d_i, kw(d_i)) \in DB\}$ 。其中, d_i 表示第 i 个文档的标识符, $kw(q)$ 表示与查询 q 相关联的关键字。若原始的访问模式被泄露, 敌手则知道 $L_{\text{Search}}(q_1, DB) = L_{\text{Search}}(q_2, DB) = \{d_1, d_2\}$, $L_{\text{Search}}(q_3, DB) = \{d_1, d_3\}$, 其中 q_i 是对关键字 kw_i 的查询。之后对任意两个查询泄漏之间求交集, 可以得到 $L_{\text{Search}}(q_1, DB) \cap L_{\text{Search}}(q_2, DB) = \{d_1, d_2\}$, $L_{\text{Search}}(q_1, DB) \cap L_{\text{Search}}(q_3, DB) = \{d_1\}$, 其中 $x \in \{1, 2\}$ 。最终文档集合的共现模式可以表示为一个共现矩阵 $\bar{M}$:

$$\bar{M}(q_1, q_2, q_3, DB) = \begin{bmatrix} 2 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \quad (6)$$

其中, 矩阵第 i 行第 j 列的元素表示为在 DB 上查询 q_i 与查询 q_j 的文档交集个数, 即 $|L_{\text{Search}}(q_i, DB) \cap L_{\text{Search}}(q_j, DB)|$ 。此时, 如果敌手还了解底层文档集合的分布情况, 即可重新识别 q_3 为 kw_3 上的查询, 因为它是唯一一个只与其他关键字共享一个文档的关键字。

2.6 隐藏共现模式

定义 6 给定关键字集 W 上的两个数据集 D_0 和 D_1 , 其共现矩阵满足 $\max \{ \text{COR}_{D_0}(ij) \} = \max \{ \text{COR}_{D_1}(ij) \}$, $i \neq j$ 。其中, $\max \{ \text{COR}_{D_0}(ij) \}$ 表示在数据集 D_0 上的共现矩阵中除对角线外所有矩阵元素的最大值。如果存在一个泄漏函数 L 满足对任意的一个查询序列 $Q = (q_1, q_2, q_3, \dots, q_t)$, 其共现矩阵都有 $\text{COR}_{Q, D_0} = \text{COR}_{Q, D_1}$, 即在文档集 D_0 与文档集 D_1 上的查询共现模式相同, 则称可搜索加密方案是隐藏共现模式。

3 隐藏共现模式的 SSE 方案

3.1 方案主要思想

在访问模式泄露的情况下只对共现模式进行保护, 共

现模式泄漏了同时包含关键字 w_1 和关键字 w_2 的文档数量。隐藏共现模式即防止真实的共现矩阵泄漏出去。若想完美地隐藏共现模式,则应让任意两个不同关键字对应的文档交集数量达到一致,即共现矩阵中除对角线外的其他所有元素都为同一数字。本文通过文档填充算法对原始文档进行填充,使得共现矩阵中除对角线外的其他所有元素都为同一数字,再运用 SSE 技术便可有效隐藏共现模式。

3.2 隐藏共现模式的文档填充算法

本文提出一种可有效隐藏共现模式的文档填充算法。假设文档集 $D = (d_1, d_2, \dots, d_n)$, 关键字集 $W = (w_1, w_2, \dots, w_m)$, 将任意两个关键字对应的文档集合求交集可得到共现矩阵 CM 。矩阵中元素的最大值为 $l = \max_{1 \leq i, j \leq m} \{u_{ij}\}$, u_{ij} 表示同时包含关键字 w_i 和关键字 w_j 的文档个数。

$$CM = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1m} \\ u_{21} & u_{22} & u_{23} & \dots & u_{2m} \\ u_{31} & u_{32} & u_{33} & \dots & u_{3m} \\ \dots & \dots & \dots & \dots & \dots \\ u_{m1} & u_{m2} & u_{m3} & \dots & u_{mm} \end{bmatrix} \quad (7)$$

3.2.1 算法操作流程

该文档填充算法具体操作流程如下:

(1) 在共现矩阵 CM 中找到含 l 数量最多的行, 按含 l 数量由大到小进行排序(移动矩阵的行和列, 并交换位置), 得到共现矩阵 M_1 。假设 M_1 如式(8)所示, 按每行从左到右逐个进行填充。如在矩阵 M_1 中, 首先对元素 u_{13} 进行填充, 使其达到最大值 l 。

$$M_1 = \begin{bmatrix} l & l & u_{13} & \dots & u_{1m} \\ l & u_{22} & u_{23} & \dots & u_{2m} \\ u_{31} & u_{32} & u_{33} & \dots & u_{3m} \\ \dots & \dots & \dots & \dots & \dots \\ u_{m1} & u_{m2} & u_{m3} & \dots & u_{mm} \end{bmatrix} \quad (8)$$

(2) 计算文档集 $D' = D(w_1) - D(w_1 \cap w_3) - \{D(w_j; u_{3j} = l)\}$, 该公式表示包含关键字 w_1 的所有文档减去同时包含关键字 w_1 和 w_3 的所有文档, 再减去关键字 w_3 与其他关键字 w_j 文档交集个数为 l 的所有文档。在文档集 D' 的关键字中, 找出与 w_3 交集文档个数最大的关键字 w_β 。

(3) 计算 $\alpha = \min\{l - u_{13}, l - |D(w_\beta \cap w_3)|\}$, 在文档集 D' 中取出 α 个含关键字 w_β 的文档, 使其包含关键字 w_3 , 并更新矩阵 M_1 的第 3 行和第 3 列。

(4) 若 $u_{13} = l$, 则对下一个元素 u_{14} 进行填充, 使其达到最大值 l ; 若 $u_{32}, u_{34}, \dots, u_{3m} = l$, 则用 dummy 文档进行填充, 使其仅含关键字 w_1 和 w_3 , 并更新矩阵。若 D' 中的文档为空, 则优先找只包含 w_3 一个关键字的文档填充; 若不足再用 dummy 文档填充, 使其仅含关键字 w_1 和 w_3 , 并更新矩阵。否则, 返回步骤(2)。

(5) 若矩阵中除对角线外的所有元素均为同一数字, 则运行结束。最终填充好的共现矩阵如 M_2 所示。

$$M_2 = \begin{bmatrix} a_1 & l & l & \dots & l \\ l & a_2 & l & \dots & l \\ l & l & a_3 & \dots & l \\ \dots & \dots & \dots & \dots & \dots \\ l & l & l & \dots & a_m \end{bmatrix} \quad (9)$$

3.2.2 算法操作案例

下文用一个具体案例来描述该文档填充算法是如何进行操作的。假设一个数据库 $DB = \{(d_1, \{a, b, c, d\}), (d_2, \{a, b\}), (d_3, \{c\}), (d_4, \{b, d\})\}$, 其中 d_1, d_2, d_3, d_4 代表不同文档, a, b, c, d 代表不同关键字。

(1) 从数据库 DB 中得到各关键字对应的文档集合为 $D(a) = \{d_1, d_2\}$, $D(b) = \{d_1, d_2, d_4\}$, $D(c) = \{d_1, d_3\}$, $D(d) = \{d_1, d_4\}$, 对任意两个不同关键字对应的文档集合求交集, 可得到 $D(a \cap b) = \{d_1, d_2\}$, $D(a \cap c) = \{d_1\}$, $D(a \cap d) = \{d_1\}$, $D(b \cap c) = \{d_1\}$, $D(b \cap d) = \{d_1, d_4\}$, $D(c \cap d) = \{d_1\}$, 从而得到共现矩阵 CM 。其中, 矩阵中行与列的关键字序列都为 (a, b, c, d) 。

$$CM = \begin{bmatrix} 2 & 2 & 1 & 1 \\ 2 & 3 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 1 & 2 & 1 & 2 \end{bmatrix} \quad (10)$$

(2) 在共现矩阵 CM 中最大值 $l = 3$, 找到含 l 数量最多的行, 按含 l 数量由大到小进行排序(移动矩阵的行和列, 并交换位置), 得到共现矩阵 M_1 。其中, 矩阵中行和列的关键字序列都为 (b, a, c, d) 。

$$M_1 = \begin{bmatrix} 3 & 2 & 1 & 2 \\ 2 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 2 & 1 & 1 & 2 \end{bmatrix} \quad (11)$$

(3) 对矩阵中第 1 行第 2 列的元素 2 进行处理, 即对关键字 b 与关键字 a 对应的文档交集进行填充。

首先, 计算文档集 $D' = D(b) - D(b \cap a) - \{D(\{j \cap a \mid l = l\})\} = \{d_1, d_2, d_4\} - \{d_1, d_2\} - \{\} = \{d_4\}$, 其中 j 为除关键字 b 和 a 外的其它关键字。在文档集 D' 的关键字中, 找出与关键字 a 对应文档交集个数最大的关键字 $w_\beta = b$ 。

其次, 计算 $\alpha = \min(l - u_{12}, l - |D(w_\beta \cap a)|) = \min(3 - 2, 3 - 2) = 1$ 。在文档集 D' 中取出一个含关键字 b 的文档, 使其包含关键字 a , 即取出文档 d_4 , 并在 d_4 中添加关键字 a 。之后, 更新矩阵 M_1 的第 2 行和第 2 列, 得到矩阵 M_2 。此时, 关键字对应的文档集合为 $D(a) = \{d_1, d_2, d_4\}$, $D(b) = \{d_1, d_2, d_4\}$, $D(c) = \{d_1, d_3\}$, $D(d) = \{d_1, d_4\}$ 。

$$M_2 = \begin{bmatrix} 3 & 3 & 1 & 2 \\ 3 & 3 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 2 & 2 & 1 & 2 \end{bmatrix} \quad (12)$$

(4) 对矩阵中第 1 行第 3 列的元素 1 进行处理, 即对关键字 b 与关键字 c 对应的文档交集进行填充。

首先, 计算文档集 $D' = D(b) - D(b \cap c) - \{D(\{j \cap c \mid l = l\})\} = \{d_1, d_2, d_4\} - \{d_1\} - \{\} = \{d_2, d_4\}$, 其中 j 为除关键字 b 和 c 外的其它关键字。在文档集 D' 的关键字中, 找

出与 c 对应文档交集个数最大的关键字 $w_\beta = b$ 。

其次,计算 $\alpha = \min(l - u_{13}, l - |D(w_\beta \cap c)|) = \min(3 - 1, 3 - 1) = 2$ 。在文档集 D' 中取出两个含关键字 b 的文档,使其包含关键字 c ,即取出文档 d_2 和 d_4 ,并在 d_2 和 d_4 中分别添加关键字 c 。之后,更新矩阵 M_2 的第3行和第3列,得到矩阵 M_3 。此时,关键字对应的文档集合为 $D(a) = \{d_1, d_2, d_4\}$, $D(b) = \{d_1, d_2, d_4\}$, $D(c) = \{d_1, d_2, d_3, d_4\}$, $D(d) = \{d_1, d_4\}$ 。

$$M_3 = \begin{bmatrix} 3 & 3 & 3 & 2 \\ 3 & 3 & 3 & 2 \\ 3 & 3 & 4 & 2 \\ 2 & 2 & 2 & 2 \end{bmatrix} \quad (13)$$

(5)对矩阵中第1行第4列的元素2进行处理,即对关键字 b 与关键字 d 对应的文档交集进行填充。

首先,计算文档集 $D' = D(b) - D(b \cap d) - \{D(\{j \cap d\} = l)\} = \{d_1, d_2, d_4\} - \{d_1, d_4\} - \{\} = \{d_2\}$,其中 j 为除关键字 b 和 d 外的其它关键字。在文档集 D' 的关键字中,找出与 d 对应文档交集个数最大的关键字 $w_\beta = b$ 。

其次,计算 $\alpha = \min(l - u_{14}, l - |D(w_\beta \cap d)|) = \min(3 - 2, 3 - 2) = 1$ 。在文档集 D' 中取出一个含关键字 b 的文档,使其包含关键字 d ,即取出文档 d_2 ,并在 d_2 中添加关键字 d 。之后,更新矩阵 M_3 的第4行和第4列,得到矩阵 M_4 。此时,关键字对应的文档集合为 $D(a) = \{d_1, d_2, d_4\}$, $D(b) = \{d_1, d_2, d_4\}$, $D(c) = \{d_1, d_2, d_3, d_4\}$, $D(d) = \{d_1, d_2, d_4\}$ 。

$$M_4 = \begin{bmatrix} 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 4 & 3 \\ 3 & 3 & 3 & 3 \end{bmatrix} \quad (14)$$

矩阵 M_4 中除对角线外的其余元素都为同一数字,因此填充结束。

3.2.3 算法优化

为了保持原有文档内容不更改,并进一步减少填充的文档数量,本文对上述算法进行了优化。优化的主要思想为尽量少地填充文档,把找矩阵元素的最大值 $l = \max_{1 \leq i, j \leq m} \{u_{ij}\}$ 调整为找矩阵中除对角线外的元素最大值,即 $l = \max_{1 \leq i, j \leq m, i \neq j} \{u_{ij}\}$,以及将每行按除对角线外的矩阵元素之和从小到大进行排序,让新生成的文档尽可能包含多个关键字,从而减少生成只包含两个关键字的新文档。然后逐行进行处理,使共现矩阵中除对角线外的所有元素均为同一数字。

同样,令数据库 $DB = \{(d_1, \{a, b, c, d\}), (d_2, \{a, b\}), (d_3, \{c\}), (d_4, \{b, d\})\}$,其中 d_1, d_2, d_3, d_4 代表不同文档, a, b, c, d 代表不同关键字。

文档填充算法优化的具体操作流程如下:

(1)从数据库 DB 中得到各关键字对应的文档集合为 $D(a) = \{d_1, d_2\}$, $D(b) = \{d_1, d_2, d_4\}$, $D(c) = \{d_1, d_3\}$, $D(d) = \{d_1, d_4\}$,对任意两个不同关键字对应的文档集合求交集,可得到 $D(a \cap b) = \{d_1, d_2\}$, $D(a \cap c) = \{d_1\}$, $D(a \cap$

$d) = \{d_1\}$, $D(b \cap c) = \{d_1\}$, $D(b \cap d) = \{d_1, d_4\}$, $D(c \cap d) = \{d_1\}$,从而得到共现矩阵 CM 。其中,矩阵中行与列的关键字序列都为 (a, b, c, d) 。

$$CM = \begin{bmatrix} 2 & 2 & 1 & 1 \\ 2 & 3 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 1 & 2 & 1 & 2 \end{bmatrix} \quad (15)$$

(2)在共现矩阵 CM 中,找除对角线元素外矩阵元素的最大值 $l = \max_{1 \leq i, j \leq m, i \neq j} \{u_{ij}\} = 2$,然后对每行除对角线元素外的矩阵元素分别求和。第一行求和为 $2+1+1=4$,第二行求和为 $2+1+2=5$,第三行求和为 $1+1+1=3$,第四行求和为 $1+2+1=4$,并按该元素之和从小到大进行排序,从而得到矩阵 CM_1 。其中,矩阵中行与列的关键字序列都为 (c, a, d, b) 。

$$CM_1 = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 3 \end{bmatrix} \quad (16)$$

(3)对矩阵 CM_1 中第1行第2列的元素1进行处理,即对关键字 c 与关键字 a 对应的文档交集进行填充,使其文档交集个数达到最大值2。因为要保证原文档内容不更改,所以只能生成新文档。生成的新文档中至少要包含关键字 c 和 a ,再寻找是否还可以把其它关键字加入进来,能加入的关键字必须保证不会使矩阵中除对角线外的任意矩阵元素超过最大值 $l=2$ 。这里已加入的关键字 c 和 a 分别与关键字 d 对应的文档交集数都小于 $l=2$ (关键字 c 与关键字 d 的文档交集数为1,关键字 a 与关键字 d 的文档交集数为1),所以可以把关键字 d 加入进去。而关键字 c, a 和 d 中存在与关键字 b 的文档交集数大于等于 $l=2$ 的情况(如关键字 a 与关键字 b 的文档交集数为2,关键字 d 与关键字 b 的文档交集数为2),所以不能加入关键字 b 。因此,新生成的文档 $d_5 = \{c, a, d\}$ 。之后,更新矩阵 CM_1 得到新矩阵 CM_2 。

$$CM_2 = \begin{bmatrix} 3 & 2 & 2 & 1 \\ 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 1 & 2 & 2 & 3 \end{bmatrix} \quad (17)$$

(4)对矩阵 CM_2 中第1行第4列的元素1进行处理,即对关键字 c 与关键字 b 对应的文档交集进行填充,使其文档交集数达到最大值2。生成的新文档中至少要包含关键字 c 和 b ,再寻找是否还可以把其它关键字加入进来,能加入的关键字必须保证不会使矩阵中除对角线外的任意矩阵元素超过最大值 $l=2$ 。这里没有其它关键字可加入,因此新生成的文档 $d_6 = \{c, b\}$ 。之后,更新矩阵 CM_2 得到新矩阵 CM_3 。

$$CM_3 = \begin{bmatrix} 4 & 2 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 2 & 2 & 2 & 4 \end{bmatrix} \quad (18)$$

矩阵 CM_3 中除对角线外的其余元素都为同一数字,因此文档填充结束。此时,数据库 $DB = \{(d_1, \{a, b, c, d\}), (d_2, \{a, b\}), (d_3, \{c\}), (d_4, \{b, d\}), (d_5, \{c, a, d\}), (d_6, \{c, b\})\}$ 。使用该算法填充后,最终的文档数取决于关键字个数和任意两个不同关键字文档交集的最大数量。

3.3 方案设计

本节利用上文提出的文档填充算法和一个 SSE 方案 $\Sigma = (Setup, Token, Search, Resolve)$ 设计一个隐藏共现模式的 SSE 方案 HCOP-SSE = $(setup, token, search, resolve)$, 具体描述如下:

(1) *setup* 算法。输入数据集 DB 先调用文档填充算法得到一个填充好的数据集 PDB , 然后以 PDB 作为输入调用 SSE 方案 Σ 的 *Setup* 算法得到加密的数据集 EDB 和密钥 K , 并把 EDB 上传到服务器。

(2) *token* 算法。输入待查询的关键字 w , 调用 SSE 方案 Σ 的 *Token* 算法得到搜索令牌 tk , 即 $tk \leftarrow \Sigma.Token(K, w)$ 。

(3) *search* 算法。执行 SSE 方案 Σ 的 *Search* 算法, 即 $C \leftarrow \Sigma.Search(tk, EDB)$ 。

(4) *resolve* 算法。首先执行 SSE 方案 Σ 的 *Resolve* 算法, 即 $R \leftarrow \Sigma.Resolve(K, C)$, 然后以明文结果集 R 和原文档文件名的长度 l_1 作为输入, 再调用过滤算法得到未填充的明文文件集 R_1 , 即 $R_1 \leftarrow filter(R, l_1)$ 。通过删除明文结果集 R 中文件名长度小于 l_1 的文档, 即能得到真实的明文文件集 R_1 。

3.4 安全性分析

在初始化阶段, 该方案由于对文档进行了填充, 便不会泄漏真实的数据库, 故初始化泄漏函数表示为 $L_{Setup}(DB) = \emptyset$ 。在搜索阶段, 该方案会泄漏访问模式和搜索模式, 但该访问模式中不泄漏真实的共现矩阵, 只泄露两两不同关键字对应文档交集的最大数量。也就是说, 共现模式得到了保护。搜索阶段的泄漏函数表示为 $L_{Search}(w) = (ap(w), sp(w))$, 搜索模式为 $sp(w) = \{tl(t, w) \in Q\}$, 其中 w 为关键词, t 为时间戳, Q 为搜索请求记录表。访问模式为 $ap(w) = \{tl(t, id, l)\}$, 其中 l 为任意两个不同关键字对应文档交集的最大数量, id 为文档标识符。

定理 1 如果所使用的对称加密方案 SKE 具有 IND-CPA 安全性, 则 HCOP-SSE 方案是 $L = (L_{Setup}, L_{Search})$ 自适应安全的。

证明: 证明该定理的关键是针对于自适应的 PPT 敌手 A 构造一个高效的模拟器 S , 使得公式 $|\Pr[\text{Real}_{\Sigma, A}(1^\lambda) = 1] - \Pr[\text{Ideal}_{\Sigma, A, S}(1^\lambda) = 1]| \leq \text{negl}(\lambda)$ 成立, 其中 $\text{negl}(\lambda)$ 是可以忽略的。下面通过构建一个模拟器 S 来证明对于任意 PPT 敌手 A , 真实游戏 $\text{Real}_{\Sigma, A}(1^\lambda)$ 和理想游戏 $\text{Ideal}_{\Sigma, A, S}(1^\lambda)$ 的输出在计算上是无法区分的, 其中 Σ 表示 HCOP-SSE 方案。最后, 本文构造了如下游戏序列, 并认

为其在计算上彼此之间无法区分。

(1) 构造游戏 G_0 。 G_0 定义为完全按照 Σ 方案执行的真实游戏 $\text{Real}_{\Sigma, A}(1^\lambda)$, 则有:

$$\Pr[\text{Real}_{\Sigma, A}(1^\lambda) = 1] = \Pr[G_0 = 1] \quad (19)$$

(2) 构造游戏 G_1 。在 G_0 的基础上, G_1 中使用随机数代替伪随机函数生成查询令牌。由于真随机函数与伪随机函数不可区分, 故这两个游戏也是不可区分的。具体表示为:

$$\Pr[G_1 = 1] - \Pr[G_0 = 1] \leq \text{negl}(\lambda) \quad (20)$$

(3) 构造游戏 G_2 。 G_2 中使用同等长度的随机字符串替换原来使用的语义安全对称密钥加密方案生成的密文, 因加密方案具有语义安全性, 故无法区分游戏 G_1 和 G_2 。具体表示为:

$$\Pr[G_2 = 1] - \Pr[G_1 = 1] \leq \text{negl}(\lambda) \quad (21)$$

(4) 构造游戏 G_3 。 G_3 定义为理想世界中的游戏 $\text{Ideal}_{\Sigma, A, S}(1^\lambda)$, 模拟器 S 通过泄漏函数生成敌手需要的视图。在初始化中, 模拟器 S 中使用的数据并非实际的数据, 而是来源于泄漏函数 L 。前面游戏中的检索算法直接使用关键字 w 进行查询, 而模拟器 S 使用关键字 w 第一次查询索引, 将 $sp(w)$ 的最小值作为关键字 w 的唯一映射。具体表示为:

$$\Pr[G_3 = 1] - \Pr[G_2 = 1] \leq \text{negl}(\lambda) \quad (22)$$

将上述一系列游戏相关联, 可得到:

$$|\Pr[\text{Real}_{\Sigma, A}(1^\lambda) = 1] - \Pr[\text{Ideal}_{\Sigma, A, S}(1^\lambda) = 1]| \leq \text{negl}(\lambda) \quad (8)$$

定理 2 HCOP-SSE 方案是隐藏共现模式的。

证明: 对于任意给定关键字集 W 的两个数据集 D_0 和 D_1 , 满足当 $i \neq j$ 时, $\max\{COR_{D_0}(ij)\} = \max\{COR_{D_1}(ij)\}$; 当 $i = j$ 时, $COR_{D_0}(ij) = COR_{D_1}(ij)$ 。使用上文中的文档填充算法对两个数据集进行填充, 得到填充数据集 PD_0 和 PD_1 , 填充数据集中任意两个不同关键字对应的文档交集个数都相同。对任意一个填充数据集进行一系列查询, 得到查询序列为 $Q = (q_1, q_2, q_3, \dots, q_s)$, 其查询共现模式都有 $COR_{Q, D_0} = COR_{Q, D_1}$ 。因此, HCOP-SSE 方案是隐藏共现模式的。

4 性能分析

4.1 复杂度分析

在该方案中客户端只需存储密钥, 因此客户端的存储开销为 $O(1)$ 。服务器需要存储查询索引表和密文文档, 故存储开销为 $O(1 + |PDB|)$ 。其中, $|PDB|$ 表示加密填充后的所有文档数量。在查询过程中, 计算复杂度为 $O(1)$, 通信复杂度为 $O(n_w)$, n_w 表示填充后匹配关键字 w 的查询结果集大小。

4.2 效率分析

本文方案实验测试使用的操作系统是 Windows10, 机

器配备了 Intel(R) Core(TM) i7-10700 CPU @ 2.90 GHz 四核处理器,具有 16 GB 内存。该实验在本地局域网(Local Area Network, LAN)环境下使用 JAVA 编程语言和 JDK_17 开发环境模拟了一个客户端与一台服务器,实验采用的真实数据集为维基百科中的数据集。其中的对称加密算法是采用 AES 密码算法实现的,且实现过程中均调用了 java.security 和 javax.crypto 下的工具包。忽略数据库读写时间,实验数据如表 1 所示。

Table 1 Efficiency analysis table
表 1 效率分析表

关键字数量 (文档数量)	填充增加的文档 数量	填充前查询时间 /ms	填充后查询时间 /ms
43(14)	232	7.6	128.4
301(300)	596	47.7	203.5
510(500)	5 949	77.7	1 353.4
1 009(1 000)	13 889	152.2	3 237.2

在表 1 中,通过执行查询 1 000 次取平均值得到查询时间,该时间包含了查询关键字生成搜索令牌的时间、服务器用搜索令牌检索的时间,以及下载并解密文档到客户端的时间。填充后查询时间长主要是因为填充后关键字所对应的文件数增多。由以上结果可知,填充文档实现共现模式隐藏后,其牺牲的存储和查询开销是可接受的。对于文档多而关键字少的数据集,其性能更优。实验结果表明,该方案具有较高的查询效率,并且能有效防止共现模式的泄漏,能够满足实际工程的需要。

5 结论与展望

本文针对可搜索对称加密重要泄漏模式中的共现模式泄漏进行了讨论,给出了共现模式、查询共现模式和共现模式隐藏的形式化定义,并提出一种隐藏共现模式的文档填充算法。将该文档填充算法与可搜索对称加密方案结合使用,可有效防止共现模式的泄漏。本文还分析证明了 HCOP-SSE 方案的安全性,并对其进行了性能评估。实验结果表明,该方案在有效保护共现模式的情况下,其存储和效率是可接受的。满足隐藏共现模式的方案具有实用性,但该隐藏共现模式的方法还不适用于动态 SSE 方案。如何进一步减少填充的文档数,以及在动态 SSE 方案中防止共现模式泄漏,可作为下一步的研究方向。

参考文献:

[1] SONG D X, WAGNER D, PERRIG A. Practical techniques for searches on encrypted data [C]//Proceeding of 2000 IEEE Symposium on Security and Privacy, 2000: 44-55.
[2] GOLDREICH O, OSTROVSKY R. Software protection and simulation on oblivious RAMs[J]. Journal of the ACM, 1996, 43(3):431-473.
[3] GOH E. Secure indexes[EB/OL]. <https://eprint.iacr.org/2003/216.pdf>.
[4] CHANG Y, MITZENMACHER M. Privacy preserving keyword searches on remote encrypted data[C]//New York:International Conference on Ap-

plied Cryptography and Network Security, 2005.
[5] 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.
[6] CASH D, JARECKI S, JUTLA C, et al. Highly-scalable searchable symmetric encryption with support for boolean queries[C]//Advances in Cryptology—CRYPTO 2013, 2013: 353-373.
[7] ZHU B R, SUN J M, QIN J, et al. A secure data sharing scheme with designated server [EB/OL]. <https://onlinelibrary.wiley.com/doi/10.1155/2019/4268731>.
[8] ZHANG X, ZHAO B, QIN J, et al. Practical wildcard searchable encryption with tree-based index [J]. International Journal of Intelligent Systems, 2021, 36(12): 7475-7499.
[9] LIU J L, ZHAO B, QIN J, et al. Multi-keyword ranked searchable encryption with the wildcard keyword for data sharing in cloud computing [J]. The Computer Journal, 2023, 66(1): 184-196.
[10] ISLAM M S, KUZU M, KANTARCIOGLU M. Access pattern disclosure on searchable encryption: ramification, attack and mitigation [C]//San Diego:The 19th Annual Network and Distributed System Security Symposium, 2012.
[11] DAVID C, PAUL G, JASON P, et al. Leakage-abuse attacks against searchable encryption[C]//22nd Conference on Computer and Communications Security,2015:668-679.
[12] ZHANG Y P, KATZ J J., PAPAMANTHOU C. All your queries are belong to us: the power of file-injection attacks on searchable encryption [C]//25th USENIX Security Symposium, 2016:707-720.
[13] POULIOT D, WRIGHT C V. The shadow nemesis: inference attacks on efficiently deployable, efficiently searchable encryption[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security,2016:1341- 1352.
[14] GUI Z C, PATERSON K G, PATRANABIS S. Leakage perturbation is not enough: breaking structured encryption using simulated annealing [EB/OL]. <https://www.xueshufan.com/publication/3183361258>.
[15] CHEN G X, LAI T H, REITER M K, et al. Differentially private access patterns for searchable symmetric encryption[C]//IEEE INFOCOM 2018-IEEE Conference on Computer Communications, 2018: 810-818.
[16] SHANG Z, OYA S, PETER A, et al. Obfuscated access and search patterns in searchable encryption [DB/OL]. <https://arxiv.org/abs/2102.09651?context=cs>.
[17] 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, 1996:514-523.
[18] XIE J B. A medical data sharing scheme with attribute-based searchable encryption[J]. Software Guide, 2024, 23(5):146-155.
谢杰彪. 一种属性基可搜索加密的医疗数据共享方案[J]. 软件导刊, 2024, 23(5):146-155.
[19] BELLARE M, DESAI A, JOKIPII E, et al. A concrete security treatment of symmetric encryption [C]//Proceedings of the 38th Annual Symposium on Foundations of Computer Science, 1997: 394-403.
[20] LI J W, JIA C F, LIU Z L, et al. Survey on the searchable encryption [J]. Journal of Software, 2015, 26(1):109-128.
李经纬, 贾春福, 刘哲理, 等. 可搜索加密技术研究综述[J]. 软件学报, 2015, 26(1): 109-128.

(责任编辑:黄健)