doi: 10.12068/j.issn.1005-3026.2019.07.001

一种面向密文大型数据集的可搜索加密方案

贾 强, 张 帅, 周福才
(东北大学 软件学院, 辽宁 沈阳 110169)

摘 要: 为解决可搜索加密方案中由于安全索引过大而导致的关键词搜索时间复杂度过高这一问题,结合云存储应用环境,提出了一种面向密文大型数据集的可搜索加密方案. 针对云存储环境中数据集过大的用户,使用块状存储结构优化安全索引的数据结构,将安全索引按照分块参数分为 Small, Medium, Large 三类. 在关键词搜索过程中采用间接寻址的方式,使得在安全索引过大的情况下,仍然能保持良好的搜索时间复杂度,达到用户可接受的范围. 实验结果表明,随着安全索引的增大,关键词搜索时间达到了亚线性.

关 键 词: 可搜索加密;倒排索引;云存储;文件加密;大型数据集
中图分类号: TP 309 **文献标志码**: A **文章编号**: 1005-3026(2019)07-0913-07

A Searchable Encryption Scheme for Encrypted Large Data Sets

JIA Qiang, *ZHANG Shuai*, *ZHOU Fu-cai*
(School of Software, Northeastern University, Shenyang 110169, China. Corresponding author: ZHOU Fu-cai, E-mail: fczhou@mail.neu.edu.cn)

Abstract: In order to solve the problem of high keywords-search time-complexity caused by excessive security indexes in symmetric searchable encryption scheme, combined with the cloud storage application environment, a new searchable encryption (SE) scheme for encrypted large data sets in distributed environments was presented. The scheme was aimed at data owners who have large sizes of files in the cloud storage application environment. By using block storage structure to optimize the structure of security index, the security indexes were divided into three categories: Small, Medium, and Large, according to the block parameters. And with the method of register indirect addressing, the scheme can maintain good search time complexity in the case of large security index, and be acceptable for users. The experimental result shows that the keyword search time reaches sublinearity as the security index increases.

Key words: searchable encryption; inverted index; cloud storage; file encryption; large data sets

随着云计算的迅速发展,云存储技术被广泛地应用,用户逐渐将数据迁移到云端服务器,来避免本地庞大的存储开销以及繁琐的数据管理,并获得更便捷的服务. 但云端自身的开放性和共享性[1],同样给存储在分布式环境中的数据安全性带来了非常大的挑战. 为了保证数据安全性以及用户隐私,数据一般都是以密文的形式存储在云端服务器中. 但是明文数据经过加密变为密文后,虽然保证了数据的机密性及安全性,却丧失了许多明文数据原有的特性,使得在密文上进行关键字查找成为了难题. 可搜索加密[2](searchable encryption, SE)技术是近几年发展起来的支持在密文上进行关键字搜索的密码学原语,它为用户节省大量的计算和网络开销,并充分利用云环境中的分布式存储和计算能力进行密文上的关键字查找. 随着云计算的发展,在海量用户和海量数据的应用场景下,提供安全灵活高效的 SE 机制将是研究者所极力追求的目标之一[2-3].

根据构造方法的不同,可搜索加密可分为基于对称加密的可搜索加密方案以及基于公钥加密

收稿日期: 2018-06-04
基金项目: 国家自然科学基金资助项目(61772127, 61472184); 国家科技重大专项(2013ZX03002006); 辽宁省科技攻关项目(2013217004); 中央高校基本科研业务费专项资金资助项目(N151704002).
作者简介: 贾 强(1989-),男,江苏徐州人,东北大学博士研究生; 周福才(1964-),男,辽宁沈阳人,东北大学教授,博士生导师.

的可搜索加密方案. 在可搜索加密方案中,用户首先使用加密算法对数据进行加密,并将密文存储到云端服务器中;当用户发起搜索请求时,发送关键字陷门给云端服务器,服务器将收到的陷门对每个文件进行试探匹配,如果匹配成功,说明文件中包含该关键字;最后服务器将匹配到的文件密文发回给用户,用户只需要对返回的文件进行解密. 在安全性上,云端服务器除了得到访问模式、搜索模式以及文件密文、密文大小、文件个数等信息外,不会获得所搜索的关键字内容以及明文的任何信息.

Song 等[4]首先提出了适用于单用户模型的对称可搜索加密(symmetric searchable encryption,SSE)方案——SWP 方案,完成了关键字搜索功能,开创了实用性的可搜索加密机制来实现在密文上进行关键字搜索的先例. 该方案在搜索时需要对每个文件进行扫描,以匹配搜索的关键字,这使其搜索时间与文件大小呈线性相关,搜索效率低下,不适用于大型数据集下的搜索. 针对其缺陷,Goh[5]找到了一种基于 Bloom Filter 建立安全索引的方法,将搜索时间提高到线性时间. 之后,Chang 等[6]利用矢量索引,提出了基于关键字字典的加密方案. 为了提高安全性,Stefanov 等[7]提出了动态可搜索加密方案(dynamic SSE),该方案较之前的方案具有更小的泄露量和更高的效率. 在用户数据较大情况下,Cash 等[8]又提出了一种新的 dynamic SSE 方案,该方案利用 Multi-map 数据结构,完成了对安全索引的优化,实现了安全索引较大情况下的单关键字搜索密文. Kamara 等[9]使用 Multi-map 数据结构实现了多关键字查询,并得到了更优的通信复杂性和更少的泄露.

虽然目前多数可搜索加密方案中的索引在理论上具有最佳的搜索时间,但是在大型数据集上执行的性能却并不理想. 并且,I/O 延迟、存储利用率、以及数据集分布式存储都会降低对称可搜索加密方案的实际性能[10]. 当面向大型数据集时,构建安全索引过大,并且通过安全索引顺序地匹配关键字进行搜索,是实践中搜索效率低下的一个重要原因.

为解决在可搜索加密方案安全索引生成过程中由于关键字对应文件标识信息的键值对数量过多造成安全索引过大,导致传统方案搜索时间复杂度过高这一问题,本文提出分布式环境下针对大型数据集的可搜索加密方案,该方案通过在安全索引生成算法中将索引分层进行间接寻址的思想,优化了安全索引的存储结构,使得在安全索引过大的情况下,仍然保持一个良好的时间复杂度. 同时,用户在上传数据的过程中,对数据进行加密操作,使得服务器无法获取数据信息,从而保证了数据的机密性.

1 预备知识

1.1 基于安全索引的可搜索加密方案

基于安全索引的可搜索加密方案中,明文数据在本地进行加密处理得到密文数据,之后将密文数据及生成的安全索引上传至云端服务器. 在进行搜索的过程中,云端服务器根据安全索引与用户输入的搜索关键字集合进行计算,得到最终的搜索结果. 其形式化定义为

Scheme = (KeyGen, IndexGen, TrapdoorGen, Search).

$K \leftarrow \text{KeyGen}(1^k)$:密钥生成算法,输入为安全参数 k,输出为主密钥 K. 该算法为概率性算法.

$I_F \leftarrow \text{IndexGen}_K(F)$:安全索引生成算法,输入为主密钥 K 及原始文件集 F,输出 F 的安全索引 I_F. 该算法为确定性算法.

$\tau_w \leftarrow \text{TrapdoorGen}_K(w)$:陷门生成算法,输入为主密钥 K 以及关键字 w,输出为该 w 对应的陷门 τ_w. 该算法为确定性算法.

$b_{\{0,1\}} \leftarrow \text{Search}(\tau_w, I_F)$:匹配算法,输入为 τ_w 及 I_F,输出为 F 中关键字 w 匹配结果 $b_{\{0,1\}}$. 当 $b=0$ 时,表示 I_F 中不包含 w,即认为 F 中未匹配到 w;当 $b=1$ 时,表示 I_F 中包含 w,即认为 F 中匹配到 w. 该算法为确定性算法.

1.2 亚线性曲线

亚线性(sublinear)是用于描述变量之间变化关系的概念. 例如对于函数 $y = a + bx^n$,当 $n=1$ 时,x 与 y 之间的变化关系为线性关系;当 $0<n<1$ 时,x 与 y 之间的变化关系即为亚线性关系,其图像为亚线性曲线.

亚线性曲线典型的特性是 y 的变化速率随着 x 的增大而减小,即其一阶导数随着 x 的增大而减少.

2 密文大型数据集可搜索加密方案

2.1 参与的实体

1) 数据拥有者. 拥有的数据:原始文件集、安全索引、关键词陷门、密钥;参与的活动:文本分词、文件上传和下载、文件加解密、安全索引生成、

关键字陷门生成.

2）索引服务器. 拥有的数据：安全索引；参与的活动：索引信息检索.

3）数据服务器. 拥有的数据：加密后的数据集；参与的活动：文件检索.

2.2　系统模型

为了保证数据的机密性，本文的系统模型中数据的处理过程均在本地完成，分布式服务器作为云存储服务的提供者，除了存储密文数据及索引，还会根据数据拥有者的请求执行密文数据的搜索任务. 此外，假设云端服务器是诚实且好奇的，服务器会执行正确的搜索操作并且返回所有的搜索结果，但也会尝试获取用户的数据隐私. 本文的系统模型主要包括 3 个阶段：文件上传、关键字搜索以及文件下载.

文件上传阶段：数据拥有者 Owner_F 首先对原始文件集进行预处理，包括使用对称加密生成密文数据、对原始文件集 F 进行语义分析并提取关键词、为关键词构造倒排索引并生成加密索引 I；之后，数据拥有者将密文数据按照某种规则等分为 N 份上传至数据服务器 $S_{F_1}, S_{F_2}, \cdots, S_{F_N}$，并将加密索引上传至索引服务器 S_I.

关键词搜索阶段：Owner_F 向 S_I 发出对关键词 w 搜索请求，并向 S_I 提供陷门 τ_w；S_I 根据 τ_w 及 I 计算出 w 所在的数据服务器 S_F；S_F 向 Owner_F 返回 τ_w 对应的密文数据 $E(f_1, f_2, \cdots)$.

文件下载阶段：Owner_F 下载 $E(f_1, f_2, \cdots)$，使用 K 解密 $E(f_1, f_2, \cdots)$，得到包含原始文件集 f_1，$f_2, \cdots$.

2.3　数据结构

本文中分布式环境下大型数据集的可搜索加密方案所用到的数据结构如下：

1）关键词 – 文档倒排索引结构. 本文中使用 Multi-map(K, V) 集合建立关键词 – 文档矩阵模型.

2）安全索引块状存储结构. 传统基于安全索引的 SSE 方案中，执行搜索会遍历整个加密索引，使得在安全索引过大情况下会浪费很多时间. 本文通过调整安全索引的存储结构，减少安全索引的搜索时间.

安全索引块状结构构造过程如下：

1）给定索引分块参数 B 和 b，根据关键词 w 的倒排索引长度 $|\text{DB}(w)|$ 将 $\text{DB}(w)$ 分为 Small，Medium，Large 三类：

$$\text{DB}(w) \subseteq \begin{cases} \text{Small}, & 0 < |\text{DB}(w)| \leqslant b; \\ \text{Medium}, & b < |\text{DB}(w)| \leqslant Bb; \\ \text{Large}, & Bb < |\text{DB}(w)| \leqslant B^2 b. \end{cases}$$

2）$\text{DB}(w) \subseteq$ Small 时，取分块个数 $\text{Num}_{\text{BS}} = 1$，即无需分块操作，当 $|\text{DB}(w)| < b$ 时，进行随机数据填充，补齐到 b 大小，记该块为 Block_{S}.

3）$\text{DB}(w) \subseteq$ Medium 时，取分块个数 $\text{Num}_{\text{BM}} = \lceil \text{BD}(w)/B \rceil$，$\text{Num}_{\text{BM}} \leqslant b$，最后一块不足 B 大小的，进行随机数据填充，补齐到 B 大小. 对于每个分块 BM_i，$I \leqslant i \leqslant \text{Num}_{\text{BM}}$，使用对称加密计算其标签 tag_{BM_i}，将 tag_{BM_i} 随机存储到 S_I 中，其指针记为 $\text{address}_{\text{BM}_i}$，得到二元组 $< \text{address}_{\text{BM}_i}, \text{tag}_{\text{BM}_i} >$，其中，$1 \leqslant i \leqslant \text{Num}_{\text{BM}}$. 将 $\text{address}_{\text{BM}_i}$ 写入数组 A 中，对 A 按照 b 大小进行分块，取分块个数 $\text{Num}_{\text{BM}} = \lceil \text{Len}(\text{address}_{\text{BM}})/b \rceil$，由于 $\text{Num}_{\text{BM}} \leqslant b$，此时仅分为一块，即 $\text{Num}_{\text{bL}} = 1$，分块中若不足 b 大小，进行随机数据填充，补齐到 b 大小，记该块为 Block_{M}，该过程实际是进行一次间接寻址操作.

4）$\text{DB}(w) \subseteq$ Large 时，取分块个数 $\text{Num}_{\text{BL}} = \lceil \text{DB}(w)/B \rceil$，$b < \text{Num}_{\text{BL}} \leqslant Bb$，在计算得到数组 A 后，对 A 中的 Num_{BL} 条数据继续按照 B 大小进行分块操作，进行第二次间接寻址. 取分块个数 $\text{Num}_{\text{BL}} = \lceil \text{Len}(\text{address}_{\text{BL}}/B \rceil$，$\text{Num}_{\text{BL}} \leqslant b$，最后一块不足 B 大小的，进行随机数据填充，补齐到 B 大小. 对于每个分块 BL'_j，$1 \leqslant j \leqslant \text{Num}'_{\text{BL}}$，使用对称加密算法计算其标签 $\text{tag}_{\text{BL}'_j}$，将 BL_j' 随机存储到 S_I 中，其指针记为 $\text{address}_{\text{BL}'_j}$，得到二元组 $< \text{address}_{\text{BL}'_j}, \text{tag}_{\text{BL}'_j} >$，其中，$1 \leqslant j \leqslant \text{Num}'_{\text{BL}}$. 将 $\text{address}_{\text{BL}'_j}$ 写入数组 A. 最后，将 A 中由 $\text{address}_{\text{BL}'_j}$ 组成的数据按照 b 大小进行分块，取分块个数 $\text{Num}_{\text{bL}} = \lceil \text{Len}(\text{address}_{\text{BL}'})/b \rceil$，由于 $\text{Num}_{\text{BL}'} \leqslant b$，此时仅分为一块，即 $\text{Num}_{\text{bL}} = 1$，分块中若不足 b 大小，进行随机数据填充，补齐到 b 大小，记该分块为 Block_{L}.

按照上述方法分类后，得到用来存储块信息的 A 以及 Block，之后将 Block 和使用 K 生成的加密标签存储在列表 L，记录间接寻址时用的指针列表. 搜索过程中，$\text{DB}(w) \subseteq$ Small 时，进行直接寻址，返回搜索结果；$\text{DB}(w) \subseteq$ Medium 时，进行一次间接寻址，返回搜索结果；$\text{DB}(w) \subseteq$ Large 时，进行两次间接寻址，返回搜索结果. 在进行安全索引块状结构构建过程中，分块大小 B 和 b 由数据所有者按照原始文件集大小进行指定. 一般地，单一数据拥有者的 $\text{DB}(w)$ 大小不会超过 $B^2 b$，因此分为三类即可.

2.4　形式化定义

本方案由密钥生成算法、文件加密算法、安全索引生成算法、陷门生成算法、搜索算法、更新算

法和文件解密算法等 7 个多项式时间算法构成,其形式化定义为

LDSSE = (KeyGen, Enc, IndexGen, TrapdoorGen, Search, Update, Dec).

1) 密钥生成算法:$K \leftarrow \text{KeyGen}(k)$ 为概率性算法,输入安全参数 k,输出密钥 K;

2) 文件加密算法:$c \leftarrow \text{Enc}(f)$ 为确定性算法,输入原始文件 f,输出加密文件 c;

3) 安全索引生成算法:

EDB $\leftarrow$IndexGen(K, PRF, W, DB),输入密钥 K、长度可变的伪随机函数 PRF、关键字集 W 和倒排索引 DB,输出安全索引 EDB;

4) 陷门生成算法:$\tau \leftarrow \text{TrapdoorGen}(K, w)$ 为确定性算法,输入密钥 K 和关键字 w,输出其陷门 τ;

5) 搜索算法:$c_{\text{res}} \leftarrow \text{Search}(\tau, \text{EDB})$,输入陷门 τ 和安全索引 EDB,输出 τ 对应的密文集 c_{res};

6) 更新算法:$\text{EDB}' \leftarrow \text{Update}(K, \text{PRF}, F_{\text{op}})$,输入密钥 K、长度可变的伪随机函数 PRF、更新操作 op $\in \{\text{add}, \text{del}\}$ 的原始文件 F,输出更新安全索引 EDB′;

7) 文件解密算法:$f \leftarrow \text{Dec}(c)$ 为确定性算法,输入密文文件 c,输出 c 对应的原始文件 f.

2.5 安全性定义

对于一个 SSE 方案,其安全性保证:给定安全索引 DB 及密文集合 c,没有敌手可以知道原始文件集 F 的任何信息,同时对于关键词搜索询问 $Q = (w_1, w_2, \cdots, w_q)$,没有敌手可以知道 Q 的任何信息. 但是要实现这样的安全性是很困难的,已知最好的 SSE 方案也泄露了访问模式和搜索模式,因此适当地弱化安全性,允许泄露给敌手一些有限的信息.

定义泄露函数 $L_1(F)$ 和 $L_2(F, W)$ 来表示本方案中允许泄露的信息.

$(m, n, u, s) \leftarrow L_1(F)$:输入原始文件集 F,输出关键字个数 m、文件个数 n、文件标识符 u 以及每个文档大小 s.

$(\xi, \zeta) \leftarrow L_2(F, w)$:输入原始文件集 F 和关键词 w,输出搜索模式 ξ 和访问模式 ζ.

系统安全性满足以下要求:

1) 安全索引不向服务器泄露除 L_1 和 L_2 以外任何对获取文件内容有价值的信息;

2) 客户端和云端服务器进行交互的过程中,云端服务器不会获得文件上传或更新过程中任何明文信息.

定义 1 可忽略函数:对于任意一个多项式函数 $\mu(x): N \rightarrow \mathbf{R}$,总存在一个自然数 C,当 $x > C$ 时,$p(x) < 1/\mu(x)$ 总成立,称 $p(x)$ 为可忽略函数. 反之,称 $p(x)$ 为不可忽略函数.

定义敌手 A,挑战者 C,有状态的模拟器 S,G_{real} 为 A 与 C 之间的交互游戏,G_{ideal} 为 A 与 S 之间的交互游戏.

定义 2 动态自适应安全性:存在一个多项式时间算法 PPT 内的模拟器 S,对于任何 PPT 内的敌手 A,其优势:

$\text{Adv}_{\text{A,S}}(k) = |P[G_{\text{real}} = 1] - P[G_{\text{ideal}} = 1]| \leqslant \text{negl}(k)$ 成立,则称该 SSE 方案满足动态自适应安全,其中 negl(k) 为可忽略函数.

2.6 方案的具体构造

面向密文大型数据集的对称可搜索加密方案由密钥生成算法 KeyGen(k)、文件加密算法 $\text{Enc}_K(F)$、安全索引生成算法 $\text{IndexGen}_K(W, \text{DB}(W))$、陷门生成算法 $\text{TrapdoorGen}_K(w)$、搜索算法 $\text{Search}((\tau_1, \tau_2), I)$、更新算法 $\text{Update}_K(\text{add}, \text{del})$ 和文件解密算法 $\text{Dec}_K(c)$ 构成,具体描述如下:

1) KeyGen(k):数据拥有者执行该算法,输入安全参数 k,使用伪随机数生成器 $K \xleftarrow{\$} \{0,1\}^k$ 生成 3 个随机数 K_1, K_2, K_3 作为伪随机函数 PRF 的密钥. 然后通过 $K_3 \leftarrow \text{SKE.Gen}(1^k)$ 来加密文档集 F,输出 $K = (K_1, K_2, K_3)$ 作为密钥. 其中 SKE = (Gen, Enc, Dec) 为一个对称加密方案.

2) $\text{Enc}_K(F)$:数据拥有者执行该算法,输入原始文件集 F,使用密钥 K 生成密文 c. 对于文件 f_i,执行 $c_i \leftarrow \text{SKE.Enc}_{K_3}(f_i)$,$0 < i \leqslant |F|$,产生密文 c_i,并按照某种规则发送给云端服务器 S_F.

3) $\text{IndexGen}_K(W, \text{DB}(W))$:数据拥有者执行该算法,输入关键字集 W 及倒排索引集 DB(W). 对于关键词 $w \in W$,执行 $K_1, K_2 \leftarrow F(w)$,生成与 w 对应的密钥 K_1 和 K_2;对于其倒排索引 DB(w),按照 $|\text{DB}(w)|$ 分别执行:

当 $\text{DB}(w) \subseteq$ Small 时,执行

$\alpha \leftarrow F_{K_1}(0)$, $\beta \leftarrow \text{Enc}_{K_2}(\text{Block}_{\text{S}})$,

$L \leftarrow (\alpha, \beta)$,

产生列表 L 并上传至 S_I;

当 $\text{DB}(w) \subseteq$ Medium 时,执行

$\text{tag}_{\text{BM}_i} \leftarrow \text{Enc}_{K_2}(\text{BM}_i)$,

$\text{address}_{\text{BM}_i} \xrightarrow{\text{AddTo}} A[i]$, $A \xrightarrow{\text{Pad}} \text{Block}_{\text{M}}$,

$\alpha \leftarrow F_{K_1}(0)$, $\beta \leftarrow \text{Enc}_{K_2}(\text{Block}_{\text{M}})$,

$L \leftarrow (\alpha, \beta)$,

产生列表 L 并上传至 S_I;

当 $\mathrm{DB}(w)\subseteq\mathrm{Large}$ 时，执行

$\mathrm{tag}_{\mathrm{BL}_i}\leftarrow\mathrm{Enc}_{K_2}(\mathrm{BL}_i)(1\leqslant i\leqslant\mathrm{Num}_{\mathrm{BL}})$，

$\mathrm{address}_{\mathrm{BL}_i}\xrightarrow{\mathrm{AddTo}}A[i]$，$A\xrightarrow[\mathrm{CutTo}]{\mathrm{Pad}}\mathrm{Block}_{1,2,\cdots,\mathrm{Num}'_{\mathrm{BL}}}$

$\mathrm{tag}'_{\mathrm{BL}_j}\leftarrow\mathrm{Enc}_{K_2}(\mathrm{Block}_j)$，

$\mathrm{address}_{\mathrm{BL}'_j}\xrightarrow{\mathrm{AddTo}}A[j]$，$A\xrightarrow{\mathrm{Pad}}\mathrm{Block}_{\mathrm{L}}$，

$\alpha\leftarrow F_{K_1}(0)$，$\beta\leftarrow\mathrm{Enc}_{K_2}(\mathrm{Block}_{\mathrm{L}})$，

$L\leftarrow(\alpha,\beta)$，

产生列表 L 并上传至 S_I.

L 产生后，执行 $D\leftarrow\mathrm{Create}(L)$，产生字典 D. 最后输出 K,D,A.

4）$\mathrm{TrapdoorGen}_K(w)$：数据拥有者执行该算法，输入搜索关键词 w，执行$(\tau_1,\tau_2)\leftarrow F_{K_1,K_2}(w)$，生成与 w 对应的陷门 τ_1 和 τ_2，将(τ_1,τ_2)作为 w 的搜索令牌上传给 S_I.

5）$\mathrm{Search}((\tau_1,\tau_2),I)$：索引服务器 S_I 执行该算法，输入(τ_1,τ_2)和 I，输出(τ_1,τ_2)对应的密文集 c_{res}. 其过程为

执行 $\alpha\leftarrow F_{\tau_1}(I)$，获得 $\mathrm{DB}(w)$所属分类；

当 $\mathrm{DB}(w)\subseteq\mathrm{Small}$ 时，执行

$\mathrm{Block}_{\mathrm{S}}\leftarrow\mathrm{Dec}_{\tau_2}(D)$，$c_{\mathrm{res}}\leftarrow\mathrm{Get}(S_F;\mathrm{Block}_{\mathrm{S}})$；

当 $\mathrm{DB}(w)\subseteq\mathrm{Medium}$ 时，执行

$\mathrm{Block}_{\mathrm{M}}\leftarrow\mathrm{Dec}_{\tau_2}(D)$，$(\mathrm{address}_{\mathrm{BM}_1},\cdots,\mathrm{address}_{\mathrm{BM}_b})\leftarrow\mathrm{Block}_{\mathrm{M}}$，

$(\mathrm{address}_{\mathrm{BM}_1},\cdots,\mathrm{address}_{\mathrm{BM}_b})\xrightarrow{\mathrm{Find}}(\mathrm{tag}_{\mathrm{BM}_1},\cdots,\mathrm{tag}_{\mathrm{BM}_b})$，

$c_{\mathrm{res}}\leftarrow\mathrm{Get}(S_F,(\mathrm{tag}_{\mathrm{BM}_1},\cdots,\mathrm{tag}_{\mathrm{BM}_b}))$；

当 $\mathrm{DB}(w)\subseteq\mathrm{Large}$ 时，执行

$\mathrm{Block}_{\mathrm{L}}\leftarrow\mathrm{Dec}_{\tau_2}(D)$，

$(\mathrm{address}_{\mathrm{BL}'_1},\cdots,\mathrm{address}_{\mathrm{BL}'_b})\leftarrow\mathrm{Block}_{\mathrm{L}}$，

$(\mathrm{address}_{\mathrm{BL}'_1},\cdots,\mathrm{address}_{\mathrm{BL}'_b})\xrightarrow{\mathrm{Find}}(\mathrm{tag}_{\mathrm{BL}'_1},\cdots,\mathrm{tag}_{\mathrm{BL}'_b})$，

$(\mathrm{Block}_1,\cdots,\mathrm{Block}_{\mathrm{Num}_{\mathrm{BL}'}})\leftarrow\mathrm{Dec}_{\tau_2}(\mathrm{tag}_{\mathrm{BL}'_1},\cdots,\mathrm{tag}_{\mathrm{BL}'_b})$，

$(\mathrm{tag}_{\mathrm{BL}_1},\cdots,\mathrm{tag}_{\mathrm{BL}_d})\leftarrow(\mathrm{Block}_1,\cdots,\mathrm{Block}_d)//d=\mathrm{Num}_{\mathrm{BL}}$，

$c_{\mathrm{res}}\leftarrow\mathrm{Get}(S_F,(\mathrm{tag}_{\mathrm{BL}_1},\cdots,\mathrm{tag}_{\mathrm{BL}_d}))$.

6）$\mathrm{Update}_K(\mathrm{add},\mathrm{del})$：算法包括文件添加(add)和文件删除(del)两个子过程，数据拥有者执行.

①文件添加过程：

为了降低在现有安全索引的基础上做更新操作的计算代价，新增字典 D_{add}，其构造过程和 D 类似，来存储新添加文件的索引信息.

输入待添加的原始文件集 F_{add}，执行 $\mathrm{Enc}_K(F_{\mathrm{add}})$产生密文 c_{add}并上传至 S_F；

输入待添加的关键字集合 W_{add}及倒排索引集 $\mathrm{DB}(W_{\mathrm{add}})$，执行 $\mathrm{Enc}_K(W_{\mathrm{add}},\mathrm{DB}(W_{\mathrm{add}}))$产生 L_{add} 并上传至 S_I，输出 $K,D_{\mathrm{add}},A_{\mathrm{add}}$.

②文件删除过程：

新增字典 D_{del}，其构造过程和 D 相同，来存储删除文件的索引信息.

输入待删除的原始文件集 F_{del}，提取 F_{del}的关键字集 W_{del}并生成倒排索引集 $\mathrm{DB}(W_{\mathrm{del}})$，执行 $\mathrm{Enc}_K(W_{\mathrm{del}},\mathrm{DB}(W_{\mathrm{del}}))$产生 L_{del}并上传至 S_I，输出 $K,D_{\mathrm{del}},A_{\mathrm{del}}$.

文件添加或删除结束后，对于搜索关键字 w，其搜索过程转化为对于 $D+D_{\mathrm{add}}-D_{\mathrm{del}}$的搜索，方法同 $\mathrm{Search}_K(w)$，最后将三部分的搜索结果进行合并返回给 c_{res}，并在客户端进行解密即可.

7）$\mathrm{Dec}_K(c)$：数据拥有者执行该算法，输入搜索后返回的密文 c，使用对称加密算法将 c 还原为明文文件. 对于密文 c_i，执行

$f_i\leftarrow\mathrm{SKE}.\mathrm{Dec}_{K_3}(c_i),0<i\leqslant|c_{\mathrm{res}}|$，

还原对应明文 f_i，其中 c_{res} 为搜索后返回的密文集.

3　安全性及效率分析

3.1　安全性证明

本文在 L_1 和 L_2 的前提下给出了选择关键字安全性定义，其安全性证明的主要目标是在证明 L_1 和 L_2 之外，所有 PPT 敌手都无法从密文和查询列表中获得任何有用的信息.

定义游戏 G_{real0}，G_{real1}，G_{ideal0} 和 G_{ideal1}，其中 G_{real0}和 G_{real1} 代表 real 过程，G_{ideal0} 和 G_{ideal1} 代表 ideal 过程. 定义 $Q=(w_1,w_2,\cdots,w_q)$ 表示 q 次关键词询问，每次询问仅包含一个关键词.

在 G_{real0}和 G_{real1}中，选择密钥 K，使用 PRF 计算 w 对应的陷门 τ，并使用 K 为所选 DB 中的每个关键字计算其 <标签，密文> 对，然后创建字典. 其形式化定义如下：

$\mathrm{Init}(\mathrm{DB},Q)$

$K\xleftarrow{\$}\{0,1\}^k;L\leftarrow\varepsilon$

for $i\in\{1,\cdots,m\}$

do

$G_{\mathrm{real0}}:\tau_i\leftarrow F_K(w_i)$

$G_{\mathrm{real1}}:\tau_i\leftarrow T[w_i]$

for $j\in\{1,\cdots,m\}$

do

$G_{\text{real0}}: K_1 \| K_2 \leftarrow F(w_j)$

$G_{\text{real1}}: K_1 \| K_2 \leftarrow T[w_j]$

for $c \in \{1, \cdots, u\}$

do

$\ell \leftarrow F_{K_1}(c)$

$C \xleftarrow{\$} \text{Enc}_{K_2}(\text{DB})$

$L \leftarrow L \cup (\ell, C)$

$D \leftarrow \text{Create}(L)$

return$(D, \tau_1, \cdots, \tau_q)$

在 G_{real1} 中,τ_i 的返回过程和 K 的生成过程与 G_{real0} 不同. 由形式化定义可知,在 G_{real0} 中 $P[G_{\text{real0}}] = P[\text{Real}(k) = 1]$;在 G_{real1} 中,对于一个有效的敌手 A_1,其优势

$$\text{Adv}^{\text{PRF}}_{\text{A}_1, F}(k) = |P[G_{\text{real0}}] - P[G_{\text{real1}}]|.$$

根据使用 PRF 不同,在 G_{real0} 和 G_{real1} 中,得到

$P[\text{PRFReal}_{A_1, F}(k) = 1] = P[G_{\text{real0}}]$,

$P[\text{PRFRand}_{A_1, F}(k) = 1] = P[G_{\text{real1}}]$.

在 G_{ideal0} 和 G_{ideal1} 中,该游戏实际是对 G_{real1},的演变,对于查询 $Q = (w_1, w_2, \cdots, w_q)$,若 w_j 不在 Q 中,则随机选取标签 l,从可能未初始化的表 P 中读取. 其形式化定义如下:

Init(DB, Q)

$K \xleftarrow{\$} \{0,1\}^k$; $L \leftarrow \varepsilon$

for $i \in \{1, \cdots, q\}$

do $\tau_i \leftarrow T[w_i]$

for $j \in \{1, \cdots, m\}$

do

$G_{\text{real1}}: K_1 \| K_2 \leftarrow T[w_i]$

for $c \in \{1, \cdots, u\}$

do

$l \leftarrow F_{K_1}(c)$

$C \xleftarrow{\$} \text{Enc}_{K_2}(\text{DB})$

if $w_j \notin Q$

$G_{\text{ideal0}}: l \leftarrow P[j, c]$

$G_{\text{ideal1}}: C \xleftarrow{\$} \{0,1\}^{l(k)}$

$L \leftarrow L \cup (l, C)$

$D \leftarrow \text{Create}(L)$

return$(D, \tau_1, \cdots, \tau_q)$

对比 G_{real0} 和 G_{real1},G_{ideal0} 和 G_{ideal1} 更加精简,但是对于每个 w,G_{ideal0} 和 G_{ideal1} 又进行了一次遍历,对于一个有效敌手 A_2,其优势

$$\text{Adv}^{\text{PRF}}_{\text{A}_2, F}(k) = |P[G_{\text{real1}}] - P[G_{\text{ideal0}}]|.$$

结合 G_{real0},之前的概率计算公式可以转化为

$P[\text{PRFReal}_{\text{A}_2, F}(k) = 1] = P[G_{\text{real1}}]$,

$P[\text{PRFRand}_{\text{A}_2, F}(k) = 1] = P[G_{\text{ideal0}}]$.

在 G_{ideal1} 中,当 $w_j \notin Q$ 时,用一个新的字符串 $\{0,1\}^{l(k)}$ 来代替 C,对于一个有效敌手 A_3,其优势

$$\text{Adv}^{\text{IND-RCPA}}_{\text{A}_3}(k) = |P[G_{\text{ideal0}}] - P[G_{\text{ideal1}}]|.$$

G_{ideal0} 和 G_{real1} 的概率关系为

$P[\text{RCPAReal}_{\text{A}_3, F}(k) = 1] = P[G_{\text{ideal0}}]$,

$P[\text{RCPARand}_{\text{A}_3, F}(k) = 1] = P[G_{\text{ideal1}}]$.

对于 G_{ideal1},有 $P[G_{\text{ideal1}}] = P[\text{Sim}_{\text{A,S}}(k) = 1]$.

因此,对于本方案,其优势

$$\begin{aligned}\text{Adv}_{\text{A,S}}(k) &= |P[\text{Real}_{\text{A,S}}(k) = 1] - P[\text{Sim}_{\text{A,S}}(k) = 1]| = |P[G_{\text{real0}}] - P[G_{\text{ideal1}}]| = \\ &|(P[G_{\text{real0}}] - P[G_{\text{real1}}]) + (P[G_{\text{real1}}] - P[G_{\text{ideal0}}]) + (P[G_{\text{ideal0}}] - P[G_{\text{ideal1}}])| = \\ &|\text{Adv}^{\text{PRF}}_{\text{A}_1, F}(k) + \text{Adv}^{\text{PRF}}_{\text{A}_2, F}(k) + \text{Adv}^{\text{IND-RCPA}}_{\text{A}_3}(k)|.\end{aligned}$$

综上,对于所有有效敌手 A,除了可忽略概率 $\text{negl}(k)$,$\text{Real}_{\text{A,S}}(k)$ 和 $\text{Sim}_{\text{A,S}}(k)$ 的输出是一致的. 即 $|P[\text{Real}_{\text{A,S}}(k) = 1] - P[\text{Sim}_{\text{A,S}}(k) = 1]| = \text{negl}(k)$.

因此,若敌手能以可忽略的概率区分真实环境和理想环境,本方案基于云存储环境下的对称可搜索加密满足动态自适应安全.

3.2 效率分析

本方案在 1 台实体机及 4 台虚拟机上完成仿真,其中实体机作为客户端及索引服务器,虚拟机作为数据服务器. 数据来源为搜狗实验室[11]提供的 2012 年 6 月至 7 月产生的部分新闻数据以及 *Enron* 公开邮件数据[12].

针对大型数据集,本方案通过大小不同的安全索引的搜索时间来研究关键词搜索性能. 安全索引的大小在一定程度上反映了关键词 – 文件信息映射对数的大小,随着安全索引的增大,其映射对数也不断增大. 实验中,在相同的搜索关键字集合下,使用 5 种大小的安全索引进行搜索查询,其大小分别为 100,200,300,400,500 *KB*.

在传统方案中,不对上述安全索引作任何存储结构处理,搜索时直接对安全索引进行遍历操作;在本文提出的方案中,对上述安全索引存储前先进行一次块状结构处理,搜索时通过安全索引块状结构完成. 通过对比以上 5 种安全索引大小的差异性引发的关键词搜索时间的差异性结果,得到如图 1 所示的搜索时间与安全索引大小关系图.

图 1 表明，安全索引的大小发生变化时，会对搜索时间产生影响，安全索引越大，搜索时间也会越长；在搜索过程中，与传统的遍历整个安全索引的方法相比，当安全索引较小时，本方案的优势并不明显，甚至搜索时间要略高于传统遍历安全索引方案，但随着安全索引的增大，本方案的优势逐渐显现，搜索时间与传统方案相比越来越短.

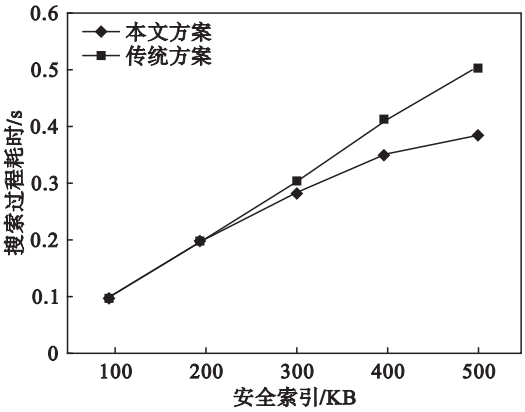

图 1　不同安全索引大小对搜索耗时的影响

Fig. 1　Effect of different security index sizes on search time

本方案使用安全索引分块的思想优化了安全索引的数据结构，根据安全索引的大小在关键字搜索过程中进行直接或间接寻址，从而摆脱了传统 SSE 中需遍历整个安全索引的缺陷. 随着安全索引的增大，当安全索引超过 200 KB 后，搜索时间随着安全索引的增大不再呈线性增长，而降至亚线性增长，从而提高了搜索效率.

4　结　　语

本文提出了一种云环境下针对密文大型数据集的可搜索加密方案，通过对可搜索加密方案中安全索引的优化，解决了传统可搜索加密方案在安全索引过大情况下搜索时间复杂度过高的问题，并且该方案满足动态自适应安全.

参考文献：

[1] Yu Y, Man H A A, Ateniese G, et al. Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage [J]. *IEEE Transactions on Information Forensics & Security*, 2017, 12(4): 767 – 778.

[2] Wang Q, He M, Du M, et al. Searchable encryption over feature-rich data [J]. *IEEE Transactions on Dependable & Secure Computing*, 2018, 12(3): 496 – 510.

[3] Hahn F, Kerschbaum F. Searchable encryption with secure and efficient updates: US9740879 [P]. 2014 – 10 – 29.

[4] Song D X, Wagner D, Perrig A. Practical techniques for searches on encrypted data [C]// IEEE Symposium on Security & Privacy. Berkeley, 2000: 44 – 55.

[5] Goh E J. Secure indexes [EB/OL]. (2003 – 10 – 07). http://eprint. iacr. org/2003/216.

[6] Chang Y C, Mitzenmacher M. Privacy preserving keyword searches on remote encrypted data [C]// International Conference on Applied Cryptography and Network Security. Berlin: Springer, 2005: 442 – 455.

[7] Stefanov E, Papamanthou C, Shi E. Practical dynamic searchable encryption with small leakage [EB/OL]. (2013 – 12 – 08). https://eprint. iacr. org/2013/832.

[8] Cash D, Jarecki S, Jutla C, et al. Highly-scalable searchable symmetric encryption with support for Boolean queries [C]// Advances in Cryptology—CRYPTO 2013. Santa Barbara: Springer, 2013: 353 – 373.

[9] Kamara S, Moataz T. Boolean searchable symmetric encryption with worst-case sub-linear complexity [C]// International Conference on the Theory and Applications of Cryptographic Techniques. Paris: Springer, 2017: 94 – 124.

[10] Ali F S, Lu S. Searchable encryption with conjunctive field free keyword search scheme [C]// International Conference on Network & Information Systems for Computers. Wuhan, 2017: 260 – 264.

[11] Sougou Labs. SougouCS [EB/OL]. (2012 – 10 – 11). http://www. sogou. com/labs/resource/cs. php.

[12] CALO Project. Enron email dataset [EB/OL]. (2011 – 08 – 21). http://www. cs. cmu. edu/ ~ ./enron.