软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(6):1566−1576 [doi: 10.13328/j.cnki.jos.004998]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于相似查询树的快速密文检索方法*

田 雪[1], 朱晓杰[1], 申培松[1], 陈 驰[1], 邹 洪[2]

[1](信息安全国家重点实验室(中国科学院 信息工程研究所),北京 100093)

[2](广东电网有限责任公司 信息中心,广东 广州 510000)

通讯作者: 陈驰, E-mail: chenchi@iie.ac.cn

摘 要: 随着云计算的广泛应用,数据中心的数据量急速增加;同时,用户文档通常包含隐私敏感信息,需要先加密然后上传到云服务器.面对如此大量的密文数据,现有技术在大数据量的密文数据上的检索效率很低.针对这一问题,提出在大数据下的基于相似查询树的密文检索方法(MRSE-SS).该方法通过设置聚类中心和成员之间的最大距离对文档向量进行聚类,并把中心向量看成 n 维超球体的球心,最大距离作为半径,再逐步将小聚类聚合成大聚类.使用该方法构建的密文文档集合,在查询阶段,仅需检索查询向量相邻的聚类即可获得理想的查询结果集合,从而提高了密文检索的效率.以《软件学报》最近 10 年的论文作为样本进行了实验,数据集中选取 2 900 篇文档和 4 800 个关键词.实验结果显示:当文档集个数呈指数增长时,检索时间仅呈线性增长,并且检索结果的关联性比传统检索方法更强.

关键词: 云计算;密文检索;多关键字排序检索;相似查询树;云安全

中图法分类号: TP309

中文引用格式: 田雪,朱晓杰,申培松,陈驰,邹洪.基于相似查询树的快速密文检索方法.软件学报,2016,27(6):1566−1576. http://www.jos.org.cn/1000-9825/4998.htm

英文引用格式: Tian X, Zhu XJ, Shen PS, Chen C, Zou H. Efficient ciphertext search method based on similarity search tree. Ruan Jian Xue Bao/Journal of Software, 2016,27(6):1566−1576 (in Chinese). http://www.jos.org.cn/1000-9825/4998.htm

Efficient Ciphertext Search Method Based on Similarity Search Tree

TIAN Xue[1], ZHU Xiao-Jie[1], SHEN Pei-Song[1], CHEN Chi[1], ZOU Hong[2]

[1](State Key Laboratory of Information Security, (Institute of Information Engineering, The Chinese Academy of Sciences), Beijing 100093, China)

[2](Information Center, Guangdong Power Grid Company Limited, Guangzhou 510000, China)

Abstract: With extensive applications of cloud computing, data capacity of data centers has grown rapidly. Furthermore, document information, which usually contains user's sensitive information, needs to be encrypted before being outsourced to data centers. Faced with such a large amount of ciphertext data, current techniques have low search efficiency in this scenario. Aiming at solving this problem, this paper proposes an efficient ciphertext search method based on similarity search tree (MRSE-SS) that can handle big data volume. The proposed approach clusters the documents based on the max distance between the cluster center and its members, constructs an n-dimensional hyper sphere by using the cluster center as the center of sphere and the max distance as radius, and then gradually clusters small clusters into large clusters. In the search phase of the ciphertext document collection constructed by this method, the ideal retrieval

* 基金项目: 广东电网有限责任公司信息中心大数据环境下的数据安全研究项目(K-GD2014-1019); 中国科学院战略性先导科技专项(XDA06040601); 新疆维吾尔自治区科技专项(201230121)

Foundation item: Information Center of Guangdong Power Grid Corporation's Project of Study on Data Security in Big Data Environments (K-GD2014-1019); Strategic Priority Research Program of Chinese Academy of Sciences (XDA06040601); Xinjiang Uygur Autonomous Region Science and Technology Plan (201230121)

收稿时间: 2015-08-14; 修改时间: 2015-10-09; 采用时间: 2015-12-05; jos 在线出版时间: 2016-01-21

CNKI 网络优先出版: 2016-01-22 10:14:49, http://www.cnki.net/kcms/detail/11.2560.TP.20160122.1014.005.html

results can be obtained only by searching the query vector's neighboring clusters, thus improving the efficiency of ciphertext search. An experiment is conducted using the collection set built from the recent ten years' JC publications, containing about 2900 documents with nearly 4800 keywords. The results show that the presented approach can reach a linear computational complexity against exponential size of document collection. In addition, the retrieved documents have a better relationship with each other than by traditional methods.

Key words: cloud computing; ciphertext search; multi-keyword ranked search; similarity search tree; cloud security

云计算环境中的数据安全问题越来越多地受到人们关注,为了确保个人数据的隐私性,用户通常先将文档加密,然后再上传到云服务器.然而,数据加密使传统的检索机制失效,随着数据量的增加,如何高效地取回加密存储在云中的数据已成为重要的挑战,密文检索问题已成为近年来信息安全领域重要的研究问题之一.

文献[1]提出了密文顺序检索架构和一种可证明安全的加密方法,即证明了:在只知道密文的情况下,云服务器无法得到任何明文的信息.然而,加密算法和查询算法都需要 $O(n)$的时间复杂度,其中,n 表示文档长度.文献[2]形式化地定义了安全索引结构——Z 索引,该索引模型通过伪随机函数和布隆过滤器(Bloom filter)实现,可以抵抗适应性选择关键字攻击.然而,Z 索引并不提供查询排序机制,若查询词出现在大量文档中,用户需要从大量的结果集中筛选所需文档.通过在倒排表中加入相关度分数,文献[3]实现了支持结果集排序的密文检索方法.在查询阶段,云服务器仅需返回与查询条件匹配的前 k 个相关文档,而不是所有满足条件的文档,这不但减少了带宽的消耗,还改善了用户体验.然而,上述工作仅能解决单关键词密文检索的问题,即用户在一次查询中仅能提交一个查询检索词.

为了更全面地表达用户的查询意图,多关键字检索技术应运而生.文献[4]提出了一种密文检索框架 MRSE,以解决多关键字密文检索问题.在索引建立阶段,每个文档被表示成一个二进制向量,其中,每一位的值代表当前文档是否包含该关键字.查询向量以同样的方式被表示成一个二进制向量.云服务器通过执行矩阵运算和安全 k 近邻算法获取排序的结果集并返回给用户.然而,MRSE 框架的查询响应时间随着文档集的增长而增长,难以适应大数据时代数据迅速增长的需求.

为了加快查询的速度,树形结构普遍应用于索引的构建.比如在数据库领域,文献[5]使用 B 树加快查询速度,文献[6]通过构造 M 树加快了对度量空间的索引过程.

本文在 MRSE 框架上进行改进,提出了一种基于相似查询(similarity search)树的新型高效索引结构(MRSE-SS).该方法在索引建立阶段将聚类算法和相似查询树进行结合,按照领域相关度将文档集合组织起来;采用动态 DK-MEDOIDS 算法,通过限制中心点到最远成员之间的距离控制聚类的大小,实现聚类的生成过程;使用相似查询树将不同的领域的聚类组织起来,通过控制上级超球体中子节点超球体的数量,动态调整结构以达到新增体积最小的目标,直至生成根节点.在查询阶段,将用户提交的查询向量表示为一个超球体,云服务器通过判断查询向量所代表的超球体与相似查询树中节点所代表的超球体之间的位置关系进行判定,仅当查询向量与某领域构成的超球体有交集时,才将该领域纳入结果集评价范围.递归执行这一步骤直至叶子节点,即可返回叶子节点中 k 个最相关的文档列表.由于在树形结构中进行了剪枝,减少了相关度计算量,从而获取了更高的效率.实验结果表明:当文档集合呈现指数增长时,使用 MRSE-SS 索引结构进行检索的查询响应时间仅呈线性增长.

本文的贡献主要体现在以下 3 个方面:

(1) 提出了一种新型的密文索引结构 MRSE-SS,将相似查询树结构引入密文索引框架,用于提升多关键字排序检索的效率,并取得了良好的效果;

(2) 提出了一种动态聚类算法 DK-MEDOIDS,聚类过程随文档量增加而动态变化,适用于云计算环境下的密文检索场景;

(3) 通过构建实验,验证了 MRSE-SS 方法的实际运行效果.

1 定义和背景

1.1 系统模型

在本文中系统涉及 3 个实体,分别为数据拥有者、数据使用者和云服务器(如图 1 所示):

- 数据拥有者负责搜集数据,建立索引,并把数据和索引加密外包到云服务器上.除了关于数据的操作外,数据拥有者还需要给需要查询数据的使用者合理授权;
- 数据使用者需要在得到数据使用者授权的情况下才能访问数据;
- 云服务器提供了密文检索所需的大量存储空间和计算资源,一旦接到数据使用者的合法请求,云服务器需要返回和查询最相关的前 k 个文档,其中,k 值是由用户设定.

系统的设计目标就是:在不向服务器泄露文档信息的前提下,提升密文检索速度并得到更符合用户意图的查询结果.

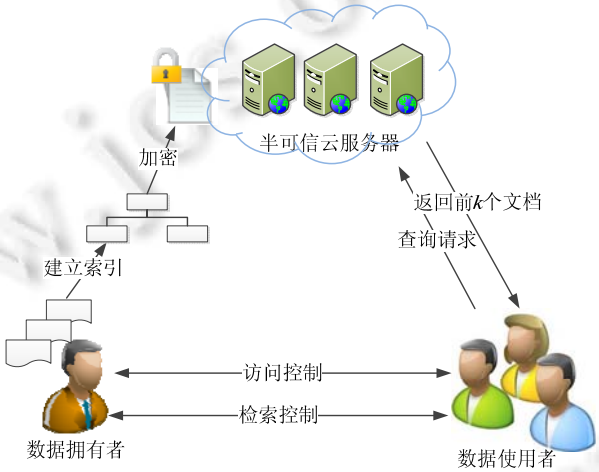


Fig.1 Architecture of search over outsourced encrypted cloud

图 1 密文检索的基本架构

1.2 安全模型

本文假定数据使用者是可信的,但是云服务器是半可信的.即:云服务器会严格按照指令执行用户的操作请求,但会尽量观察和分析服务器上的数据和索引结构,以期获知用户文档的内容.根据云服务器可以获得的信息,主要有以下的两种安全模型[7,8]:

- 已知密文模型:云服务器只知道用户外包到云上的数据,即加密后的文档集、相似查询树和加密后的查询超球体;
- 已知背景信息模型:在这个模型中,云服务器不仅知道第 1 个模型中的所有信息,还可以统计和分析云服务器上的数据.例如:知道文档集的统计信息,进而推测出明文关键字.

1.3 K-MEDOIDS聚类算法

K-MEDOIDS 算法[9]的目标主要是将 n 个数据对象划分为 k 个聚类,选用聚类中最接近所有对象平均值的对象作为聚类中心.K-MEDOIDS 算法采用绝对误差标准作为标准测量函数,其定义如下:

$$E=\sum_{i=1}^{k}\sum_{p\in C_i}|p-O_i| \tag{1}$$

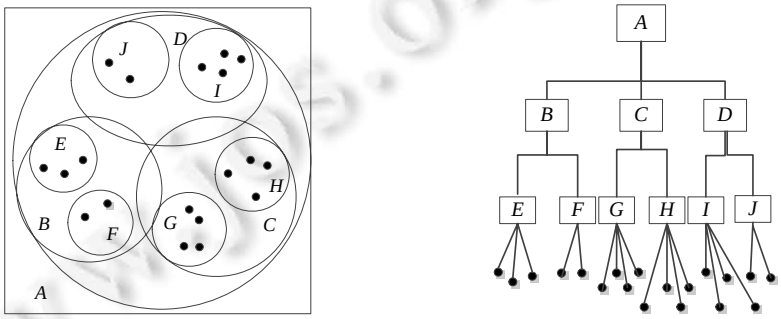
其中,p 为聚类 C_i 中的点;O_i 为聚类 C_i 的代表对象,即中心.

K-MEDOIDS 算法主要由以下 4 步组成:

(1) 在数据集中随机选择 k 个对象,每个对象代表一个聚类的初始均值或中心;
(2) 对剩下的每个对象,根据其与各个聚类中心的绝对误差大小,将它分配到最小绝对误差的聚类中;
(3) 在每个聚类中,顺序选取一个元素,用该元素代替原来的中心,计算得到各个元素和当前中心的绝对误差总和,选取绝对误差和最小的元素作为中心;
(4) 迭代步骤 2 和步骤 3,直至所有的中心点不改变.

1.4 SS-树

相似查询树[10]是 R 树的变形,它采用超球体进行空间的分割,在二维平面上如图 2(a)所示,其用树形结构如图 2(b)所示.相似查询树从下到上构建而成,上层节点为恰好覆盖下层节点的所有元素的超球体,每个节点由一个中心点和半径表示.若该节点为叶子节点,则中心点即为文档向量值;若为中间节点,则表示超球体的球心,其数据结构如图 3 所示.



(a) 高维特征空间在 2 维平面上的显示 (b) 内存或者磁盘上使用的树形结构

Fig.2
图 2

```
Struct SSElem{
SSElePtr child_array_ptr;  //孩子节点指针
int immed_children;        //当前孩子节点的大小
int total_children;        //子树中孩子节点总数
int height;                //叶子节点之上的高度
int update_count;          //更新值
float radius;              //半径
float centroid[DIM];       //中心点向量
Char data[dataSize];       //存储的数据
};
```

Fig.3 Node structure of SS tree
图 3 相似查询树中节点的数据结构

1.5 评价标准

1.5.1 相似度

在相似度计算的时候,本文采用了欧氏距离的方式进行度量.归一化后的向量内积值等价于两个向量之间的余弦值.计算方法见公式(2),其中,x 和 y 分别表示归一化后的两个不同的文档的向量.

$$\begin{aligned} D(x,y) &= \sqrt{|x|^2 - 2x\cdot y + |y|^2} \\ &= \sqrt{|x|^2 - 2|x|\cdot|y|\cdot\cos(x,y) + |y|^2} \\ &= \sqrt{2 - 2\cos(x,y)} \\ &= \sqrt{2(1-\cos(x,y))} \end{aligned} \tag{2}$$

1.5.2 检索结果集评价

较好的查询结果应该保持文档与文档之间的相似度.相似度高的文件容易同时被查询,所以文档集中文档与文档之间的相似度可以由下式度量:

$$P_l=\sum_{j'=1}^{k'}\sum_{i'>=j}^{k'}D(d_{i'},d_{j'})\Big/\sum_{j=1}^{k}\sum_{i>=j}^{k}D(d_i,d_j) \tag{3}$$

其中,k'表示最终密文查询返回的 k 个文件,k 表示明文查询中的 top k 个文件.

1.5.3 信息泄露评价

本文采用了文献[11]中定义的信息泄露评价方法,即:使云服务器返回前 k 个文档,同时,尽量少地向云服务器泄漏排序结果的信息.评估排序结果的信息泄露量按以下公式计算:

$$P_k=\sum P_i/k \tag{4}$$

$$P_i=|C_i-C_i'| \tag{5}$$

其中,C_i 表示文档 i 在排序结果 Top k 中的位置,C_i' 表示文档 i 在未添加随机值情况下的真实排序位置.从公式中可以看出:当 P_k 的值越大,说明查询结果隐藏得越好,信息泄露的越少.

1.6 符号定义

为使表达简洁和准确,我们定义了以下符号:

- DC:明文文档向量集,其中有 m 篇文档,表示为 $DC=\{d|d_1,d_2,...,d_m\}$;
- C:密文文档向量集,对应于明文文档集,表示为 $C=\{c|c_1,c_2,...,c_m\}$;
- WC:词典,其中有 n 个关键字,表示为 $WC=\{w|w_1,w_2,...,w_n\}$;
- m:SS 树中非叶子节点的最少子元素个数,根节点除外;
- M:SS 树中非叶子节点的最多子元素个数,根节点除外;
- $D(O_i,O_j)$:表示两超球体球心 O_i,O_j 之间的距离;
- Qw:表示查询向量;
- T:表示一个聚类中能够容纳的最大文档向量个数;
- r:最小超球面体的半径;
- n:词典中关键字个数;
- u:为了维护文档向量的安全性,添加的维度数目;
- v:从 u 维度中随机选取的维度数目;
- k:用户指定的返回文档数目.

2 算法和方法框架

2.1 基于相似查询树的索引框架

MRSE-SS 索引的框架主要由以下 4 个算法组成:生成密钥的 $Keygen(1^n,u)$算法、建立密文索引的 $Build\text{-}index(DC,SK)$算法、生成陷门算法 $Trapdoor(w,SK)$、查询算法 $Query(Tw,k,I)$.

- $Keygen(1^n,u)$

数据拥有者随机的生成了一个$(n+u+1)$位的向量作为分割指示向量 S,同时生成两个$(n+u+1)\times(n+u+1)$的可逆矩阵(M_1,M_2),组成密钥 $SK=\{S,M_1,M_2\}$.

- $Build\text{-}index(DC,SK)$

数据拥有者为每个文档生成一个 n 维的文档向量 $DC[i]$,其中的每一位 $DC[i][j](0<j<n)$记录关键词 w_j 对应当前文档的权重[12].调用相似查询树构建算法(将在第 2.4 节介绍),对文档向量集 DC 建立相似查询树索引 I.其次,将索引中所有节点的中心向量(包括每个超球体的球心向量和所有文档向量)从 n 维扩展到$(n+u+1)$维,其中,$(n+j)(0<j<u+1)$为随机值 R_j,最后一维$(n+u+1)$为 1.

然后,通过向量 S 将索引所有节点的中心向量分割为两个子向量 $DC[i][j]'$ 和 $DC[i][j]''$,若 S_i 为 1,则 $DC[i][j]'$ 和 $DC[i][j]''$ 都为随机值,且它们之和为 $DC[i][j]$;若 S_i 为 0,则 $DC[i][j]'=DC[i][j]''=DC[i][j]$.此时,索引向量 $I=\{I',I''\}$.最后,通过两个矩阵 M_1 和 M_2 对索引 I 加密,即 $I=\{M_1^T I', M_2^T I''\}$,加密后的索引被上传到云端.

- *Trapdoor*(w,SK)

用户根据关键字在词典中是否出现设置 Qw:如果在词典中出现,则对应位置的 $Qw[i]$ 值为 1;否则为 0.随机地从 u 个关键字中选择 v 个,在 Qw 对应位置中设为 1,其余位置设为 0,最后一维取随机值 t.再生成一个随机值 q 对 Qw 的前(n+u)维进行整体变化,即 $Qw=(q\times Qw(n+u),t)$.然后,通过向量 S 对 Qw 进行分割:当 S_i 值为 1,$Qw[i]'=Qw[i]''=Qw[i]$;当 S_i 值为 0 的时候,$Qw[i]'$ 和 $Qw[i]''$ 取随机值,但它们的和为 $Qw[i]$.通过矩阵 M_1 和 M_2 对 Qw' 和 Qw'' 进行加密,陷门可以表示为三元组:$Tw=\{M_1^{-1}Qw', M_2^{-1}Qw'', r\}$,其中,$r$ 表示最小超球体的半径.

- *Query*(Tw,k,I)

通过陷门 Tw,服务器首先计算查询超球体和根节点各个超球体之间的关系,得到交集最多的某个超球体,然后根据得到的超球体,继续向下一层节点寻找,直到叶子节点.计算叶子节点中的文档向量和查询向量超球体的中心向量之间的距离,得到距离最近的前 k 个文档向量,并返回这 k 个文档向量的列表.

2.2 关键词权重计算方法

本文使用了文献[13]定义的关键词的权重计算方法:

$$w_i=\frac{TF(t_i)\times IDF(t_i)}{\sqrt{\sum_{i=1}^{n}(TF(t_i)\times IDF(t_i))^2}}=\frac{TF(t_i)\times\log\left(\frac{N}{n_i}+1\right)}{\sqrt{\sum_{i=1}^{n}\left(TF(t_i)\times\log\left(\frac{N}{n_i}+1\right)\right)^2}} \tag{6}$$

其中,TF 是特征项频率,即关键词在文档中出现的次数;IDF 是逆文档频率,即与出现该关键词的文档个数成反比;t_i 表示关键词;N 表示词典中词的个数;n_i 表示出现该词的文档个数.

2.3 DK-MEDOIDS算法

传统 K-MEDOIDS 算法[9]的 k 值需要提前确定,不能在聚类的过程中随着文档数量的变化而动态变化,因此不适用于密文检索场景.本文设计了 DK-MEDOIDS 算法,其标准测量函数如下:

$$E=\sum_{i=1}^{k}\sum_{p\in C_i}D(p,O_i) \tag{7}$$

其中,p 为聚类 C_i 中的点;O_i 为聚类 C_i 的代表对象,表示聚类中心;$D(p,O_i)$函数用于计算点 p 和点 O_i 之间的距离.

算法分为以下几个步骤:

(1) 设置聚类中心和它的成员之间的距离门限值 r 以及单个聚类中元素的门限值 T;

(2) 在 DC 中随机的选择 k 个对象,每个对象代表一个簇的初始均值或中心;

(3) 在每个簇中,顺序选取一个元素,用该元素代替原来的聚类中心,计算得到各个元素和当前聚类中心的距离总和,选取距离和最小的元素作为中心;

(4) 检测各个聚类是否存在元素和聚类中心距离大于 r 的情况:若存在,则 k=k+1,即:加入一个新的聚类中心,并将异常点作为此新的聚类中心,重新执行第 2 步;

(5) 直到所有的聚类中心不再变化;

(6) 检测所有聚类,成员元素是否超过门限值 T:若存在,则对该聚类调用 DK-MEDOIDS 算法,生成子聚类;

(7) 检测所有元素和各个聚类中心之间的距离:若元素和当前聚类中心之间的距离小于 r,并且该元素未在当前聚类中出现,则将该元素加入到当前聚类中.

2.4 相似查询树构建算法

本文结合动态 DK-MEDOIDS 算法和相似查询树构建算法构建新型 MRSE-SS 树,主要分为以下几个步骤:

(1) 设定每个节点的最少子节点个数 m 和最大子节点个数 M;
(2) 调用动态 DK-MEDOIDS 算法,建立 N_0 个最小超球体;
(3) 将距离相近的 $x(m\leqslant x\leqslant M)$个超球体重新组成一个大集合,调用 K-MEDOIDS 算法,此时 k=1,只需求出中心节点,并以中心节点和最远成员之间的距离作为半径;
(4) 重复步骤(3),选取增加“体积”最小的一种组合方案,输出 N_i 个超球体;
(5) 将 N_i 个超球体组织成相似查询树中的一层;
(6) 迭代步骤(3)~步骤(5),直到超球体个数 N_i 小于 m 个;
(7) 将最后的 N_i 个超球体构成根节点.

2.5 查询算法

本文所提出查询算法的核心思想是,寻找和查询超球体有最大交集的超球体.查询超球体和相似查询树中的超球体的位置情况可分为包含、相交和不相交.用 R_{Qw} 表示查询超球体的半径,R_n 表示相似查询树中的超球体半径.

- 相交的判定方法,如图 4(a)和公式(8)所示.

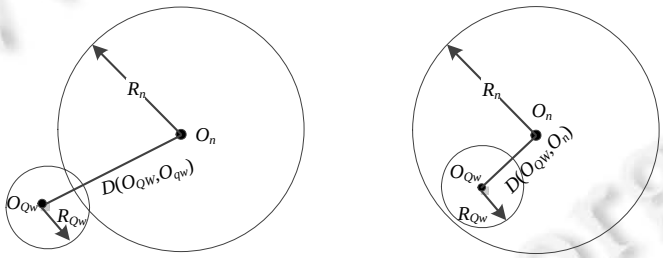
$$(R_{Qw}+R_n)>D(O_{Qw},O_n)>|R_{Qw}-R_n| \tag{8}$$

- 包含的判定方法,如图 4(b)和公式(9)所示.

$$D(O_{Qw},O_n)<|R_{Qw}-R_n| \tag{9}$$

- 不相交的判定方法,如图 5 和公式(10)所示.

$$D(O_{Qw},O_n)>(R_{Qw}+R_n) \tag{10}$$



(a) 两个超球体之间的相交关系 (b) 两个超球体之间的包含关系

Fig.4

图 4

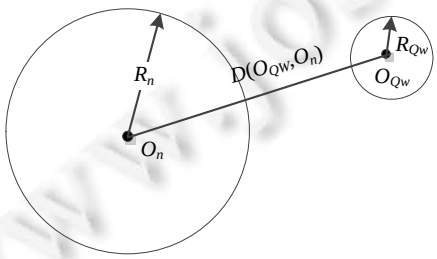


Fig.5 Disjoint relation between two hyper spheres

图 5 两个超球体之间不相交关系

查询的过程中,针对以上位置关系,通过设置状态标记值来记录相似查询树中节点和查询超球体之间的关系,状态标记值取值范围为:{未标记,不相交,相交但不包含,包含},初始值为未标记.

查询算法的具体步骤如下:

(1) 服务器首先计算查询超球体和根节点各个超球体之间的关系,得到交集最多的某个超球体.该步骤使

用的查找算法伪代码见算法 1-FindClosestSphere;

(2) 根据得到的超球体,继续向下一层节点寻找交集最多的超球体,在下一层中调用算法 1;

(3) 重复步骤 2,直到叶子节点,计算叶子节点和查询超球体球心 O_{Qw} 之间的距离,选取其中距离最近的前 k 个文档,并返回给用户.

算法 1. FindClosestSphere/在本层中查找索引树中和查询超球体有最大交集的超球体.

简明起见,定义操作集合 A 包含以下两个语句:

- *candidate_node=current_node*; //当前节点作为候选节点;
- *min_dis*=$D(O_{Qw},O_n)$; //查询超球体和当前节点的距离作为最小距离.

```
state=unmarked;
interate all nodes in this layer {
  if (D(O_Qw,O_n)>=R_Qw+R_n) {                    //如果不相交
    if (state==intersected||state==contained) {
            cut this branch;
    } else if (state==disjoint && D(O_Qw,O_n)<min_dis) {
            A;
    } else if (state==unmarked) {
            A; state=disjoint;
    }
  } else if (D(O_Qw,O_n)>abs(R_Qw−R_n)) {          //如果相交
    if (state==contained) {
          cut this branch;
    } else if (state==intersected && D(O_Qw,O_n)<min_dis) {
          A;
    }else if (state==unmarked||state==disjoint) {
          A; state=intersected;
    }
  } else {                                         //如果包含
    if (state==unmarked||state==intersected||state==disjoint) {
            A; state=contained;
    } else if (D(O_Qw,O_n)<min_dis) {
          A;
    }
  }
}
```

3 效率和安全分析

3.1 查询效率分析

查询过程可以分为陷门生成 *Trapdoor*(w,*SK*)和查询 *Query*(*Tw*,k,*I*)两部分.陷门生成阶段需要的操作见公式(11),其中,n 表示词典中的关键字个数,w 表示查询的关键字个数:

$$O(\text{MRSE-SS})=5n+u-v-w+5 \tag{11}$$

当文档向量集 *DC* 以指数级增长的时候,陷门的生成时间独立于 *DC* 的增长,只与词典有关,其时间复杂度可以表示为 $O(1)$.MRSE 和 MRSE-SS 的陷门生成时间复杂度一样.

在 MRSE 的查询阶段,云服务器需要计算所有的密文文档向量和密文查询向量之间的相关度分数,并排序返回前 k 个最相关的文档列表.MRSE 在查询阶段需要的操作见公式(12),其中,m 表示文档向量的个数,n 表示词典中关键字个数:

$$O(\text{MRSE})=2m\times(2n+2u+1)+m-1 \tag{12}$$

相比于 MRSE,在 MRSE-SS 的查询阶段,云服务器使用相似查询树索引查询文档,其所需要的操作数见公式(13),其中,B_i 表示相似查询树中第 i 层需要计算的节点的个数;l 表示相似查询树的最大层数;m 表示文档向量个

数;n 表示词典中关键字个数;t 表示最小超球体中成员的个数,且 $t<=T$.

$$O(\text{MRSE-SS})=\left(\sum_{i=1}^{l}B_i\right)\times(2\times B_i\times(2n+2u+1)+2\times B_i)+t\times(2\times(2n+2u+1))+t+1 \tag{13}$$

当文档向量集 DC 呈指数增长时,m 可以表示为 2^l,那么 MRSE 的查询时间复杂度就是 $O(2^l)$,而 MRSE-SS 的时间复杂度则为 $O(l)$.

总的查询时间复杂度可以通过公式(14)表示:当文档数量以指数级增长的时候,MRSE-SS 的查询时间只是线性增长;而传统方法如 MRSE,则是以指数级的形式快速增长.

$$O(SearchTime)=O(Trapdoor)+O(Query) \tag{14}$$

3.2 安全分析

3.2.1 已知密文模型

在已知密文模型下,敌手可获得相应的密文信息包括加密的文档向量和查询向量等,但是加密密钥是保密的.本方案是基于方案[14]做出了适应性的改进和安全增强,本方案的加密密钥由两部分组成,分别为$(n+u+1)$位的分割指示向量 S 和$(n+u+1)\times(n+u+1)$的可逆矩阵(M_1,M_2).为简单起见,不失一般性,我们假设 $u=0$,即,不添加随机值(添加随机值是为了增加方案的安全性).由于分割指示向量 S 是未知的,所以文档向量 $DC[i]'$和 $DC[i]''$可被看作是两个随机的$(n+1)$维的向量.

为解出加密文档数据构建的线性方程组,即 $C_i'=M_1^T\times DC[j]'$ 和 $C_i''=M_2^T\times DC[j]''$,我们在 m 篇文档向量中有 $2m\times(n+1)$个未知量,在可逆矩阵(M_1,M_2)中有 $2(n+1)\times(n+1)$个未知量.因为我们只有 $2m\times(n+1)$个等式,小于未知量的个数,所以没有足够的信息来推断出文档向量或者矩阵(M_1,M_2).

同理,查询向量 Qw'和 Qw''可被看作 2 个$(n+1)$维的随机向量,为解出加密查询向量构建的方程组,我们在两个查询向量中有 $2(n+1)$个未知量,在矩阵(M_1,M_2)中有 $2(n+1)\times(n+1)$个未知量.因为我们仅有 $2(n+1)$个等式,小于未知量的数量,因此没有足够的信息计算出查询向量或者是矩阵(M_1,M_2).因此,该方案在已知密文场景下是安全的.

3.2.2 已知背景信息模型

在这个模型中,云服务器不仅知道密文索引信息,而且还具备记录查询结果、分析查询过程、推测不同陷门(trapdoor)之间关系、数据库的统计分析知识等能力.下面从查询无联系性和关键字安全两方面进行分析.

- 查询的无联系性:在查询向量的生成过程中,由于加入了随机值,从而使得每次陷门的生成结果都不同,从而使得服务器无法建立查询向量和文档之间的对应关系.在陷门的生成过程中,通过以下 3 个过程实现了查询向量的随机性:
  1. 从添加的 u 位中随机的选择了 v 位,并对这些位赋予随机值;
  2. 通过生成一个随机值 q,对查询向量的前 $n+u$ 位的值做了一个规约化;
  3. 查询向量最后一维的值被置为随机值;
- 关键字安全:用户在进行搜索服务时,倾向于向云服务器隐藏自己的搜索内容.通常使用陷门的方式来保护查询关键字.在已知背景信息模型中,云服务器可利用词频信息(包含关键词的文档个数)来推断关键词.本文的关键字加密方案采用了安全增强的内向量积计算方式[4],由于该加密方式是满足关键字安全的,所以本文提出的加密方式也是满足关键字安全的.

通过合适设置 u 的值,用户可以混淆查询和查询结果的关系,增加云服务器运用统计分析猜测关键字的难度.但是,u 值的增大也会降低查询的准确性,因此,这是关键字安全和查询准确性的平衡.

4 性能分析

为了测试 MRSE-SS 模型在真实数据集上的性能,我们建立了实验平台验证检索的准确性和效率.测试平台建立在 Intel Core i7 3.40GZ 的 Linux 服务器上,数据集采用《软件学报》最近 10 年所刊载的论文,共约 2 900

篇文档,约 4 800 个关键词.图 6 展示了 MRSE-SS 方案与 MRSE 方案的效率对比情况.

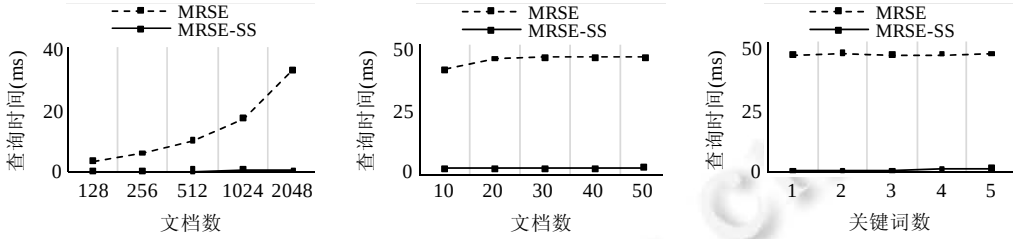


Fig.6 Search efficiency

图 6 查询效率

在图 6(a)中,当文档集合的个数呈指数增加时,MRSE 方案的查询响应时间也是呈指数级增加,而 MRSE-SS 方案的查询响应时间近似线性增加;在图 6(b)中,当查询返回的结果文档数目变化时,MRSE 和 MRSE-SS 方案的查询时间都比较稳定;在图 6(c)中,当查询关键词数目变化时,MRSE-SS 方案仍然有极大的优势.

图 7 展示了 MRSE-SS 和 MRSE 方案在查询结果集内的文档间相似度上的对比情况.从图中可以看出:MRSE-SS 方案返回的文档集中文档间的相似度更高,文档集中的文档更相关.

图 8 展示了 MRSE-SS 和 MRSE 方案在排序结果信息泄漏上的对比,可以看出:MRSE-SS 方案的排序隐私都数值更高,也就是隐私度更好,安全性更高.

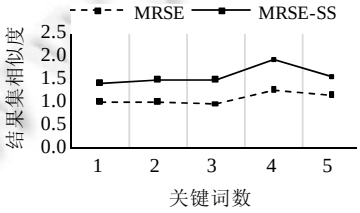


Fig.7 Document similarity

图 7 结果集相似度

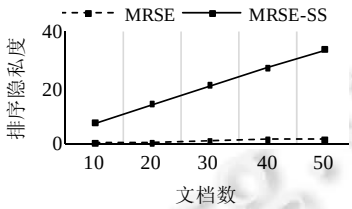


Fig.8 Rank privacy

图 8 结果集排序隐私度

5 总 结

本文提出了 MRSE-SS 密文检索框架、索引结构和相应的算法,并建立了密文检索实验平台验证了本文所提方案的有效性.实验结果表明:MRSE-SS 方案支持密文多关键字排序检索,并且在查询效率、排序隐私和查询结果集的相关度上都较传统的 MRSE 方法有了较大的提升.

References:

[1] Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Titsworth FM, ed. Proc. of the 2000 IEEE Symp. on Security and Privacy. Los Alamitos: IEEE Computer Society, 2000. 44–55. [doi: 10.1109/SECPRI.2000.848445]

[2] Goh EJ. Secure Indexes. Vol.216: IACR Cryptology ePrint Archive, 2004. 1–19.

[3] Wang C, Cao N, Li J, Ren K, Lou W. Secure ranked keyword search over encrypted cloud data. In: Guerrero JE, ed. Proc. of the 2010 Int'l Conf. on Distributed Computing Systems. Los Alamitos: IEEE Computer Society, 2010. 253–262. [doi: 10.1109/ICDCS.2010.34]

[4] Sun W, Wang B, Cao N, Li M, Lou W, Hou YT, Li H. Privacy-Preserving multi-keyword text search in the cloud supporting similarity-based ranking. In: Proc. of the ASIA 8th ACM Symp. on Information, Computer and Communications Security (CCS 2013). New York: ACM Press, 2013. 71–82. [doi: 10.1145/2484313.2484322]

[5] Leslie H, Jain R, Birdsall D, Yaghmai H. Efficient search of multi-dimensional B-trees. In: Dayal U, ed. Proc. of the 21th Int'l Conf. on Very Large Data Bases (VLDB'95). San Francisco: Morgan Kaufmann Publishers Inc., 1995. 710–719.

[6] Ciaccia P, Patella M, Rabitti F, Zezula P. Indexing metric spaces with M-tree. In: Cristani M, ed. Proc. of the SEBD. 1997. 67−86.

[7] Wang C, Cao N, Ren K, Lou WJ. Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans. on Parallel and Distributed Systems, 2012,23(8):1467−1479. [doi: 10.1109/TPDS.2011.282]

[8] Yu S, Wang C, Ren K, Lou W. Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Proc. of the 2010 IEEE INFOCOM. New York: IEEE Press, 2010. 1−9. [doi: 10.1109/INFCOM.2010.5462174]

[9] Kaufman L, Rousseeuw PJ. Finding Groups in Data: An Introduction to Cluster Analysis. New Jersey: John Wiley & Sons, 2005. 108−110. [doi: 10.1002/9780470316801]

[10] White D, Jain R. Similarity indexing with the SS-tree. In: Proc. of the 20th Int'l Conf. on Data Engineering. New York: IEEE Press, 1996. 516−523. [doi: 10.1109/ICDE.1996.492202]

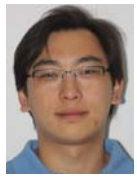
[11] Cao N, Wang C, Li M, Ren K, Lou W. Privacy-Preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans. on Parallel and Distributed Systems, 2014,25(1):222−233. [doi: 10.1109/TPDS.2013.45]

[12] Witten IH, Moffat A, Bell TC. Managing gigabytes: Compressing and Indexing Documents and Images. 2nd ed., San Francisco: Morgan Kaufmann Publishers, 1999. 181−184.

[13] Salton G, Buckley C. Term-Weighting approaches in automatic text retrieval. Information Processing & Management, 1988,24(5): 513−523. [doi: 10.1016/0306-4573(88)90021-0]

[14] Wong WK, Cheung DW, Kao B, Mamoulis N. Secure kNN computation on encrypted databases. In: Binnig C, ed. Proc. of the 2009 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2009). New York: ACM Press, 2009. 139−152. [doi: 10.1145/1559845.1559862]

田雪(1986－),女,山东烟台人,助理研究员,主要研究领域为密文检索.



陈驰(1978－),男,博士,副研究员,主要研究领域为云计算安全,数据安全.

朱晓杰(1989－),男,博士生,主要研究领域为社交网络隐私保护,数据安全存储.

邹洪(1986－),男,工程师,主要研究领域为信息技术应用,信息安全.

申培松(1993－),男,博士生,主要研究领域为密文检索.

www.jos.org.cn