计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20170830
56(3):555-565,2019

基于聚类索引的多关键字排序密文检索方案

杜瑞忠　李明月　田俊峰
(河北大学网络空间安全与计算机学院　河北保定　071002)
(河北省高可信信息系统重点实验室(河北大学)　河北保定　071002)
(drzh@hbu.edu.cn)

Multi-keyword Ranked Ciphertext Retrieval Scheme Based on Clustering Index

Du Ruizhong, Li Mingyue, and Tian Junfeng
(*School of Cyber Security and Computer, Hebei University, Baoding, Hebei* 071002)
(*Key Laboratory on High Trusted Information System in Hebei Province (Hebei University), Baoding, Hebei* 071002)

Abstract　Data owners prefer to outsource documents in an encrypted form for the purpose of privacy preserving. But existed encrypting technologies make it difficult to search for encrypted data, which limit the availability of outsourced data. This will make it even more challenging to design ciphertext search schemes that can provide efficient and reliable online information retrieval. In order to improve the efficiency and precision of ciphertext retrieval, we propose a multi-keyword ciphertext retrieval scheme based on clustering index. Firstly, the improved Chameleon algorithm is used to cluster the file vectors during which the file vectors are dimensioned by recording the position of the key words. Secondly, a retrieval algorithm suitable for clustering index is proposed, which makes it possible to eliminate a large number of file vectors irrelevant to the query vector in the query process, and reduce unnecessary consumption. Finally, in the clustering process, Jaccard similarity coefficient is introduced to calculate the similarity between the file vectors and to set the appropriate threshold to improve the quality of the cluster. The theory analysis and experimental results show that the scheme can effectively improve the efficiency and precision of ciphertext retrieval under the premise of guaranteeing the privacy and security of data.

Key words　cloud security; ciphertext search; ranked search; clustering index; Chameleon algorithm

摘　要　为了提高密文检索的效率和精度,提出基于聚类索引的多关键字排序密文检索方案.首先利用改进的Chameleon算法对文件向量聚类,聚类过程中通过记录关键字位置对文件向量进行降维处理.其次,提出适合聚类索引的检索算法,使得在查询过程中可以排除大量与查询向量无关的文件向量,减少了不必要的计算消耗.再次,在聚类过程中引入杰卡德相似系数来计算文件向量之间的相似度以及设定合适的阈值提高聚类质量.在真实数据集上进行了实验,理论分析和实验结果表明:在保障数据隐私安全的前提下,该方案较传统的密文检索方案有效地提高了密文检索的效率与精度.

收稿日期:2017-11-03;**修回日期**:2018-04-08
基金项目:国家自然科学基金项目(61170254,60873203);河北省自然科学基金项目(F2016201244,F2018201153);河北省高等学校科学技术研究基金项目(ZD2016043)
This work was supported by the National Natural Science Foundation of China (61170254, 60873203), the Natural Science Foundation of Hebei Province of China (F2016201244, F2018201153), and the Science and Technology Research Project in Colleges and Universities of Hebei Province (ZD2016043).
通信作者:李明月(15630424277@163.com)

关键词 云安全;密文检索;排序检索;聚类索引;Chameleon 算法

中图法分类号 TP309.2

云计算被认为是企业 IT 基础设施的新模式,该模式可以组织庞大的计算,提供可用的、便捷的、按需的网络访问,进入可配置的计算资源共享池存储和应用资源.尽管云服务具有各种优势,但将敏感信息外包给远程服务器也带来了隐私问题.这些隐私安全威胁严重阻碍了云计算的发展.现有的加密技术可以保障云环境下的数据在可预见时间里的安全性,但这些复杂的加密技术使搜索加密数据变得困难,限制了外包数据的可用性[1].因此,如何在保护用户隐私的同时,使云数据获得有效的检索是亟待解决的问题.

可搜索的加密方案使客户能够将加密的数据存储到云中,并通过密文域执行关键字搜索.文献[2]提出了第 1 个基于密文扫描思想的可搜索加密方案,该方案对单个关键词的查询需要扫描整个文件,因此检索的效率低.文献[3]提出了对称可搜索加密的正式安全定义,并设计了一个基于 Bloom 过滤器的方案.该方案搜索时间是 $O(m)$,m 是文档数量.文献[4]提出了实现最优搜索时间的 2 种方案(SSE-1 和 SSE-2).其中,SSE-1 方案对于选择关键字攻击是安全的,SSE-2 对于自适应选择关键字攻击是安全的.但上述方案只是用于单关键字的查询,即用户在一次查询中仅能提交 1 个查询检索词,在功能方面非常简单.

相对于单关键字查询,多关键字查询允许用户输入多个查询关键字来请求需要的文档,可以更好地满足用户的查询需求.文献[5]提出了多关键字排序密文检索问题,通过计算文件向量和查询向量内积的结果进行排序.但是此方案没有考虑关键字权重的问题,查询向量与结果集文件的相关度不高,即检索精度不高.文献[6]在文件排序时用余弦相似度代替内积来计算文件向量与查询向量的相似度,同时引入 TF/IDF(term frequency/inverse document frequency)算法,提高了检索精度,但缺乏对频率信息和实际搜索性能的安全性分析.文献[7]应用了稀疏矩阵的方法实现了安全的大规模方程.文献[8]提出细粒度的多关键字检索方案,采用分类子词典技术以及偏好因子提高了检索效率.为了方便用户查看检索结果,文献[9]提出了一个支持排名搜索安全的多关键字密文排序检索方案,该方案基于二叉平衡树构建了索引,但在生成索引树时该方案随机选取 2 个结点生成父结点,将文档之间的相关性忽略,降低了检索精度,同时为了得到 top-K 个文件在最坏的情况下要遍历整个索引树.

受到以上工作的激励,本文在多关键字密文检索框架上提出基于聚类索引的多关键字排序密文检索方案(multi-keyword ranked ciphertext retrieval scheme based on clustering index, CTMRSE).在构建索引树前用改进的 Chameleon 算法对文件进行动态聚类,通过制定阈值来限制聚类的过程,最后按照聚类结果构建索引树,直到生成根结点.查询的过程利用检索算法自上而下查询.当访问的结点不是叶子结点时,选择与查询向量相似度最高的结点访问;当访问的结点为叶子结点时,利用排序算法按相似度高低将文件插入列表;如果列表中文件数量少于 K,回溯到其父亲结点访问,否则直接返回列表.该检索方法降低了查询的时间复杂度,提高了检索效率.

本文的主要贡献体现在 3 个方面:

1) 在聚类前通过记录关键字位置对文件向量进行降维,降低了聚类过程中不必要的计算消耗,减少了聚类时间.

2) 在动态聚类 Chameleon 算法中引入了杰卡德相似系数来计算高维文件向量之间的相似度以及设定合适的 k 值与 $minMex$ 值来提高在高维空间中聚类结果的质量,从而提高检索精度.

3) 提出新型密文检索结构 CTMRSE 及相应的算法,通过理论分析与实验验证了该方案提升了多关键字密文检索的效率和精度.

1 背景介绍

1.1 系统模型

本文的系统模型如图 1 所示,将云服务按功能不同可以分为 3 个实体:数据拥有者、数据使用者和云服务器.

1) 数据拥有者.数据拥有者即数据的属主方,要搜集数据、数据分类、建立聚类索引以及加密和上传数据与索引.此外,数据拥有者需要给数据使用者合理授权,分享密钥.

2) 数据使用者.数据使用者为数据拥有者授权的用户,要设定返回的文档数 K,然后将查询的内

容生成陷门 ***TD*** 发送给云服务器,在获得返回的 K 个文档后,使用共享密钥对文档进行解密.

3) 云服务器. 云服务器为密文检索提供了大量的计算和存储资源,接到数据使用者合法的查询请求时,云服务器需要根据查询算法以及利用索引树进行计算,返回和查询最相关的前 K 个文档.

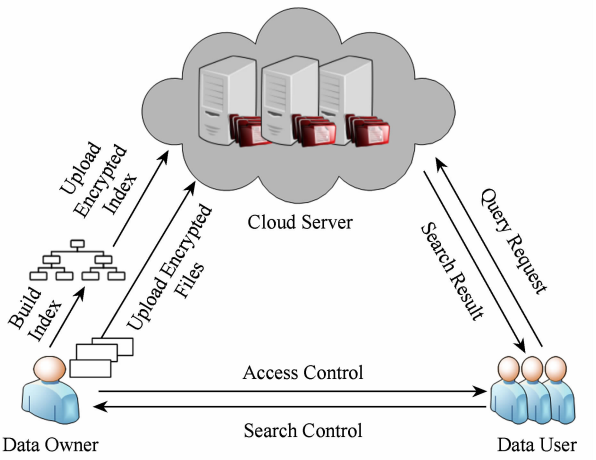


Fig. 1 The architecture of search over encrypted cloud data

图 1 密文检索的系统架构图

1.2 模型威胁分析

为了规范研究范围,本文假设云服务器诚实但是好奇,即云服务器会按着用户的要求执行操作,但它会试图分析用户的数据与索引从而获取用户的隐私信息. 主要考虑 2 种威胁模型[10]:

1) 密文已知

云服务器只知道数据拥有者上传到云端的加密后的文档集 C、索引树 T 和陷门 ***TD***.

2) 背景已知

在背景已知模型中,云服务器不仅知道密文已知模型中的信息,还配备统计信息功能,可以通过分析相应频率分布的直方图,进行 TF 统计攻击,推断甚至识别某些关键词.

1.3 主要符号说明

本文使用的主要符号说明如下.

F: 明文文档集合$\{f_1,f_2,\cdots,f_m\}$;

C: 密文文档集合$\{c_1,c_2,\cdots,c_m\}$;

W: 关键词集合$\{w_1,w_2,\cdots,w_n\}$;

I: 安全索引集合$\{\boldsymbol{I}_1,\boldsymbol{I}_2,\cdots,\boldsymbol{I}_n\}$;

$\boldsymbol{Q}_W$: 查询向量;

TD: 查询关键词陷门;

m: 文档集合文档数量;

E_K: 返回的结果集;

k: 最近邻数量;

M: 一个聚类中最多容纳文件个数;

u: 为了文档的安全性添加的维数;

n: 关键词集合中关键词的个数;

K: 用户指定返回的文档个数.

1.4 Chameleon 算法

Chameleon 算法[11]是一种层次聚类算法,其运作模型如图 2 所示,图 2 中的每个顶点代表 1 个文件对象,每个文件对象用 n 维向量来表示,连接 2 个顶点边的权值为这 2 个文件之间的相似度. 聚类过程分为 3 个步骤:用 k 最近邻算法构建稀疏图,k 值对算法的结果会产生很大影响,要设定适当的取值;在保证割边最小化的情况下,将图划分成多个小簇;使用凝聚层次聚类算法,基于子簇的相似度反复合并子簇,其中,$minMex$(度量函数阈值)取值过小容易发生过拟合,较大则导致近似误差增大,因此也要设定合理值. 其中,度量函数为 $RI(c_i,c_j)\times RC(c_i,c_j)^{\alpha}$. 当 $\alpha>1$ 时,表示更重视相对近似性;当 $\alpha<1$ 时,表示更重视相对互连性;当 $\alpha=1$ 时,表示 2 个量度标准有相等的权重.

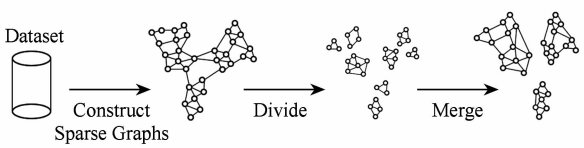


Fig. 2 The operation model of Chameleon algorithm

图 2 Chameleon 算法的运作模型

1.5 结果集排序隐私度

排名隐私度[12]可以量化搜索结果向云服务器的信息泄漏量:

$$R'_K=\sum_{i=1}^{K}|r_i-r'_i|/K^2, \tag{1}$$

其中,r_i 是返回的 top-K 文档的排名,r'_i 是整个排名结果中的真实排名,较大的级别隐私表示该方案的较高安全性.

2 改进的 Chameleon 算法

2.1 向量降维

在建立索引前对文件集向量进行聚类用到的是 Chameleon 聚类算法. 选择该算法来聚类是因其在聚类的过程中不仅考虑每个簇之间的互联性,而且考虑簇的邻近性,动态聚类结果质量高,可提高后期检索效率与精度. 但 Chameleon 算法聚类代价较高,为了降低聚类时间对文档向量进行降维处理.

每个文件对应 1 个 n 维的文档向量,文件包含的关键词在向量中对应的位置为 1,其他位置都为 0.

一个文件一般只会包含极少量的关键词[13],即每个 n 维的文档向量中包含大量的 0,文档向量之间的相似度计算会带来不必要的开销.因此,将高维向量通过记录关键词的位置进行降维处理可减少不必要的计算消耗,从而提高聚类的效率.假设关键词数 $n=1\,000$,则每个文档向量的维度为 1 000,通过记录关键词的位置降维,每个位置只需要用 10 位二进制来表示.如图 3 所示,向量 $\boldsymbol{D}_1$ 和 $\boldsymbol{D}_2$ 分别为文件 1 与文件 2 的文件向量,$\boldsymbol{D}_1$ 向量中的 1 对应的位置为 8,35,255,985,即文件 1 包含关键词集中的 4 个关键词$\{w_8,w_{35},w_{255},w_{985}\}$. $\boldsymbol{D}_2$ 向量中的 1 对应的位置为 8,35,129,255,768,即文件 2 含有关键词集中的 5 个关键词$\{w_8,w_{35},w_{129},w_{255},w_{768}\}$.图 3 为通过记录关键词位置将向量 $\boldsymbol{D}_1$ 和 $\boldsymbol{D}_2$ 转化为向量 $\boldsymbol{A}_1$ 和 $\boldsymbol{A}_2$ 的过程,随着文件向量的维数增加,基于位置降维效果会更明显,聚类效率相应提高.

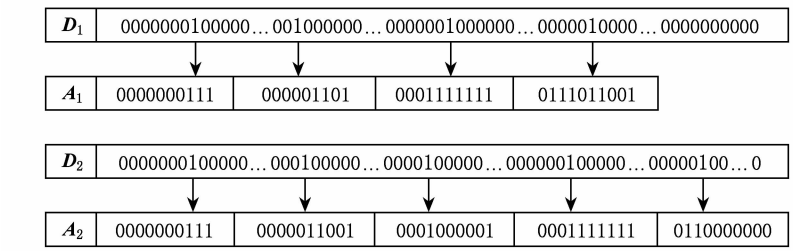


Fig. 3 The dimensionality reduction process of $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$

图 3 文件向量 $\boldsymbol{D}_1$ 与 $\boldsymbol{D}_2$ 的降维过程

2.2 相似度

在 Chameleon 算法中利用 k 邻近算法构建稀疏图时,连接 2 个边的权值是 2 个文件向量的相似度(邻近图中的文件向量是经过降维处理的).为了在高维空间得到有意义的簇,提高聚类结果的质量,引入杰卡德相似系数计算经过降维处理的文件向量之间的相似度.杰卡德相似系数是衡量 2 个集合相似度的一种指标,用杰卡德相似系数计算高维向量之间的相似系数时,用 $J(\boldsymbol{D}_1,\boldsymbol{D}_2)$表示:

$$J(\boldsymbol{D}_1,\boldsymbol{D}_2)=\frac{p}{p+q+r}=\frac{p}{n-s},\tag{2}$$

其中,p 是文件 1 与文件 2 包含相同关键字的个数;q 是文件 1 包含文件 2 不包含关键字的个数;r 是文件 2 包含文件 1 不包含关键字的个数;s 是文件 1 与文件 2 都不包含关键字的个数.

文件向量 $\boldsymbol{D}_1$ 与 $\boldsymbol{D}_2$ 经过降维后得到 $\boldsymbol{A}_1$ 与 $\boldsymbol{A}_2$,如图 3 所示,$J(\boldsymbol{D}_1,\boldsymbol{D}_2)$可以通过向量 $\boldsymbol{A}_1$ 与 $\boldsymbol{A}_2$ 来计算,由于 $\boldsymbol{A}_1$ 与 $\boldsymbol{A}_2$ 记录着文件 1 与文件 2 包含关键字的位置,且相同的关键字在文档向量中对应的位置相同,可通过对比记录的关键词的位置即可迅速的计算出 p,q,r,s 的值,不需要对比向量 $\boldsymbol{A}_1$ 与 $\boldsymbol{A}_2$ 的每个比特位,在提高聚类质量的同时也提升了聚类效率.例如图 3 中文件向量 $\boldsymbol{D}_1$ 和文件向量 $\boldsymbol{D}_2$ 之间的杰卡德相似系数为 1/4.

2.3 改进的 Chameleon 算法

在 Chameleon 算法中引入杰卡德相似系数来计算文件向量之间的相似度以及在聚类过程中通过记录关键字位置对文件向量进行降维处理后的算法如下.

算法 1. 改进的 Chameleon 算法.

输入:文件集合 F;

输出:聚类结果.

① for each file f_i in cluster C_i do

$\boldsymbol{D}_i \leftarrow RedDimPro(f_i)$;

$d \leftarrow JaccardCoef(\boldsymbol{D}_i,\boldsymbol{D}_j)$;

if ($d=MinDistance$ && $Num\leqslant k$) then

$Connection(\boldsymbol{D}_i,\boldsymbol{D}_j)$;

$Num++$;

end if

end for

② Built k-nearest graph;

③ for each cluster C_i in $ClusterList$

$C_i,C_j \leftarrow EC(C_i)$;

$EC(C_i,C_j) \leftarrow AbsolutInter(C_i,C_j)$;

$EC(C_i) \leftarrow Inter(C_i)$;

$RI(C_i,C_j) \leftarrow Relative(C_i,C_j)$;

$SEC\{C_i,C_j\} \leftarrow AbsolutClose(C_i,C_j)$;

$SEC\{C_i\} \leftarrow InterClose(C_i)$;

$RC(C_i,C_j) \leftarrow RelativeClose(C_i,C_j)$;

$tempMetric=\max\{RI(C_i,C_j)\times RC(C_i,C_j)\}$;

$minMex \leftarrow SetThreshold$;

if ($tempMetric>minMex$) then

New $C_i \leftarrow Merge(C_i,C_j)$;

else

Delete C_i from $ClusterList$;

end if

end for

在算法1中,$RedDimPro(f_i)$为文件 f_i 生成文档向量,并对文档向量进行降维的函数;$JaccardCoef(\boldsymbol{D}_i,\boldsymbol{D}_j)$为用杰卡德相似系数来计算文件向量 $\boldsymbol{D}_i$ 和 $\boldsymbol{D}_j$ 之间相似度的算法.$RI(C_i,C_j)$定义为簇 C_i 和 C_j 的互联度:

$$RI(C_i,C_j)=\frac{|EC_{|C_i,C_j|}|}{\frac{1}{2}(|EC_{|C_i|}+EC_{|C_j|}|)},\tag{3}$$

其中,$|EC_{|C_i,C_j|}|$为连接簇 C_i 和 C_j 的所有边的权重和.EC_{C_i}(EC_{C_j})为把簇 $C_i(C_j)$划分成2个大致相等部分的最小等分线切断的所有边的权重和.

$RC(C_i,C_j)$定义为簇 C_i 和 C_j 相对近似度:

$$RC(C_i,C_j)=\frac{\bar{S}_{EC_{|C_i,C_j|}}}{\frac{|C_i|}{|C_i|+|C_j|}\bar{S}_{EC_{C_i}}+\frac{|C_j|}{|C_i|+|C_j|}\bar{S}_{EC_{C_j}}},\tag{4}$$

其中,$|C_i|$表示子簇 C_i 包含数据点的数量,$|C_j|$表示子簇 C_j 包含的数据点的数量.$\bar{S}_{EC_{|C_i,C_j|}}$是连接簇 C_i 和 C_j 边的平均权重,即 $\bar{S}_{EC_{|C_i,C_j|}}=\frac{EC_{|C_i,C_j|}}{|C_i|+|C_j|}$,$\bar{S}_{EC_{C_i}}$($\bar{S}_{EC_{C_j}}$)是小二分簇 $C_i(C_j)$的边的平均权重,即 $\bar{S}_{EC_{C_i}}=\frac{|EC_{C_i}|}{|C_i|^2}$,$\bar{S}_{EC_{C_j}}=\frac{|EC_{C_j}|}{|C_j|^2}$.

利用改进的 Chameleon 算法对文件向量聚类可提高聚类质量,从而提高检索精度.此外先聚类再建立索引可提高检索的效率.

2.4 聚类结果

为了更好地显示聚类过程,从文件集合$\{f_1,f_2,\cdots,f_m\}$中选取了12个文件,根据改进的聚类算法将文件动态聚类得到 D,E,F,H,G 这5个簇.图4为文件向量映射在2维平面上的效果图,其中簇 D,E,F 相似度分数高,将 E,F,D 这3个簇聚为一类(即 B),类似地,也可将簇 H,G 聚为一类(即 C),A 为 B 和 C 的总和.

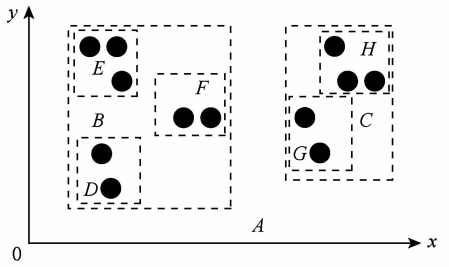


Fig. 4 The process of clustering

图4 聚类过程

3 方案构建

3.1 基本方案

1) $SK\leftarrow setup(1^n,u)$

数据拥有者从$\{f_1,f_2,\cdots,f_m\}$中提取 n 个关键词集$\{w_1,w_2,\cdots,w_n\}$.然后,随机生成一个$(n+u+1)$维的向量 $\boldsymbol{S}$ 作为分割指示向量,同时生成2个$(n+u+1)\times(n+u+1)$可逆矩阵 $\boldsymbol{M}_1$ 和 $\boldsymbol{M}_2$,组成密钥 $SK=\{\boldsymbol{S},\boldsymbol{M}_1,\boldsymbol{M}_2\}$.

2) $I\leftarrow Genindex(F,SK)$

在生成文档向量前,数据拥有者计算关键词权重[10]:

$$p_j=\frac{TF(w_j)\times IDF(w_j)}{\sqrt{\sum_{i=1}^{n}(TF(w_i)\times IDF(w_i))^2}},\tag{5}$$

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,与出现该关键词的文档个数成反比,$IDF=\lg\frac{N}{idf}$,w_j 表示关键词,N 表示词典中词的个数,idf 表示包含该词的文档个数.然后基于向量空间模型[14],数据拥有者为每篇文档 f_i 生成一个文档向量 $\boldsymbol{D}_i$,如图5所示,若文档 f_i 中包含关键词 w_j,则 $D_i[j]=1$,否则 $D_i[j]=0$.接着将文档 $D_i[j]$中为1的位置置为该关键字的权重 p_j.然后根据聚类结果建立索引,通过向量 $\boldsymbol{S}$ 将索引中所有结点的中心向量分割为2个子向量 $\boldsymbol{D}_i'$ 和 $\boldsymbol{D}_i''$.若 $S[i]=1$,$\boldsymbol{D}_i'$ 和 $\boldsymbol{D}_i''$ 都为随机值,且它们之和为 $\boldsymbol{D}_i$;若 $S[i]=0$,$\boldsymbol{D}_i'=\boldsymbol{D}_i''=\boldsymbol{D}_i$.最后,通过2个矩阵 $\boldsymbol{M}_1$ 和 $\boldsymbol{M}_2$ 对索引树加密.对索引树中每个结点的2个向量进行加密得到 $\boldsymbol{I}_i=(\boldsymbol{M}_1^{\mathrm{T}}\boldsymbol{D}_i',\boldsymbol{M}_2^{\mathrm{T}}\boldsymbol{D}_2'')$,然后加密后的索引 I 被上传到云端.

3) $\boldsymbol{TD}\leftarrow GenTrapdoor(\boldsymbol{Q}_W,SK)$

如图6所示,用户根据文件的关键字在词典中是否出现设置 $\boldsymbol{Q}_W$,如果在词典中出现,则对应位置 $Q[i]=1$;否则为0.然后,通过向量 $\boldsymbol{S}$ 对 $\boldsymbol{Q}_W$ 进行分割:当 $S[i]=1$ 时,$Q[i]'=Q[i]''=Q[i]$;当 $S[i]=0$ 时,$Q[i]'$和$Q[i]''$取随机值,但它们的和为 $Q[i]$.通过矩阵 $\boldsymbol{M}_1$ 和 $\boldsymbol{M}_2$ 对 $\boldsymbol{Q}'$和$\boldsymbol{Q}''$进行加密,陷门可以表示为 $\boldsymbol{TD}=(\boldsymbol{M}_1^{-1}\boldsymbol{Q}',\boldsymbol{M}_2^{-1}\boldsymbol{Q}'')$.

4) $E_K\leftarrow Query(I,\boldsymbol{TD},K)$

通过陷门 $\boldsymbol{TD}$,云服务器利用检索算法自上而下进行检索,先计算陷门与根结点所有孩子结点的相关度得到相关度最高的子结点,然后访问该子结

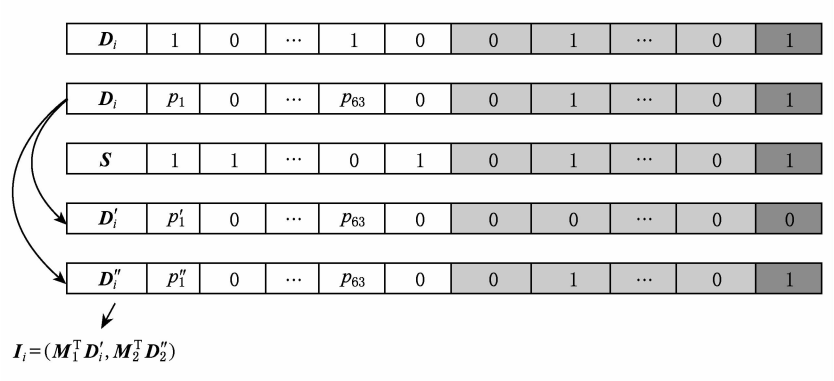


Fig. 5 The process of index tree encryption

图 5 索引树加密过程

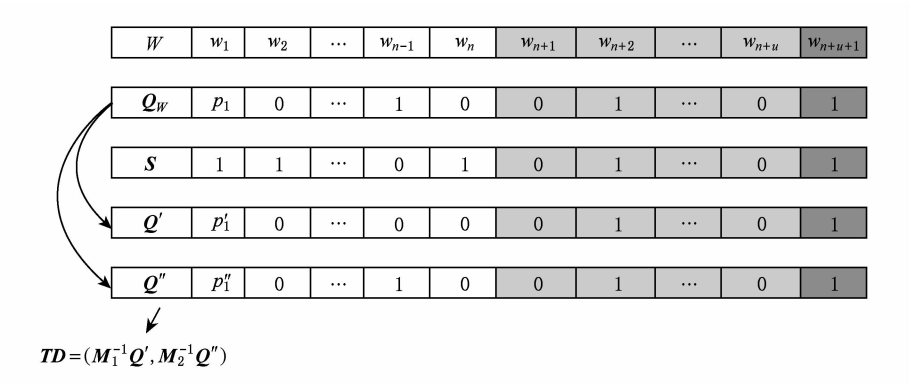


Fig. 6 The process of trap generation

图 6 陷门生成过程

点的所有孩子结点,以此类推直到找到目标簇,如果该簇的文件量小于返回文件数 K,回溯到其父亲结点,遍历其兄弟结点的孩子结点,直到找到相似度分数最高的前 K 个文档向量,返回给数据使用者.

其中,相关度分数的计算:

$$\begin{aligned} Rscore=\boldsymbol{TDI}_i&=\boldsymbol{M}_1^{\mathrm{T}}\boldsymbol{D}_i'\,\boldsymbol{M}_1^{-1}\boldsymbol{Q}'+\boldsymbol{M}_2^{\mathrm{T}}\boldsymbol{D}_i''\,\boldsymbol{M}_2^{-1}\boldsymbol{Q}''=\\ &(\boldsymbol{M}_1^{\mathrm{T}}\boldsymbol{D}_i')^{\mathrm{T}}\,\boldsymbol{M}_1^{-1}\boldsymbol{Q}'+(\boldsymbol{M}_2^{\mathrm{T}}\boldsymbol{D}_i'')^{\mathrm{T}}\,\boldsymbol{M}_2^{-1}\boldsymbol{Q}''=\\ &(\boldsymbol{D}_i')^{\mathrm{T}}\boldsymbol{M}_1\,\boldsymbol{M}_1^{-1}\boldsymbol{Q}'+(\boldsymbol{D}_i'')^{\mathrm{T}}\boldsymbol{M}_2\,\boldsymbol{M}_2^{-1}\boldsymbol{Q}''=\\ &\boldsymbol{D}_i'\,\boldsymbol{Q}'+\boldsymbol{D}_i''\,\boldsymbol{Q}''=\boldsymbol{D}_i\,\boldsymbol{Q}_W. \end{aligned} \tag{6}$$

5) $F_K\leftarrow Dec(E_K, SK_f)$

数据使用者接收到云服务器返回的密文 E_K 后利用密钥 SK_f 解密,得到明文 F_K.

3.2 索引构建

根据聚类结果构建索引树的算法如下.

算法 2. 构建聚类索引的算法.

输入: 文件集 F;

输出: 根结点 *rootNode*.

MakeIndexTree(F)

① *InitQueue*($*Q$);

② for each document $f_i\in F$ do

$\boldsymbol{v}\leftarrow\boldsymbol{D}_i$;

EnQueue(SqQueue Q,QElem $\boldsymbol{v}$);

end for

③ while ((*QueueLength*(SqQueue Q)>1)do

for each *data* in Q do

$C_i\leftarrow$*Chameleon*(Q->*data*);

end for

end while

④ for each cluster C_i do

Comput Cluster Center M_i;

ParentNode $PN\leftarrow M_i$;

if ((Q->*rear*+1)%*MaxSize*=Q->*front*)then

return 0;

else

EnQueue(SqQueue Q,QElem PN);

$N\leftarrow$*QueueLength*(SqQueue Q);

DeQueue(LinkQueue $*Q$,N,*NodeSet*);

end if

end for

rootNode$\leftarrow$$Q$->*data*;

return *rootNode*.

在算法 2 中,*QueueLength*(SqQueue *Q*)为求队列 *Q* 长度的算法;*EnQueue*(SqQueue *Q*, QElem *PN*)是将元素 *PN* 插入队列 *Q* 的队尾;*DeQueue*(LinkQueue *Q*,*N*,*NodeSet*)将队列 *Q* 前 *N* 个元素全部删除,然后把删除的元素插入到 *NodeSet* 中.

3.3 检索过程

利用索引树查询的算法如下.

算法 3. 查询算法.

输入:索引树 *T*;

输出:搜索结果文件集 *Rlist*.

GDFABTS(*IndexTreeNode*)

① if *u.child* is not a leaf node then

  max←0;

  while(*u.child*)do

    Compute $RScore(u.child, \boldsymbol{Q}_W)$;

    if ($u.child > max$) then

      $max \leftarrow RScore(u.child, \boldsymbol{Q}_W)$;

    else

      return

    end if

  end while

  GDFABTS(*u.child*);

② else

  while (*u.child*) do

    SORT(*Rlist*, *u.child*, $RScore(u.child, \boldsymbol{Q}_W)$);

  end while

  if $Rlist.num = K$ then

    return *Rlist*;

  else

    GDFABTS(*u.rightsib*);

  end if

end if

在算法 3 中,$RScore(u.child, \boldsymbol{Q}_W)$为计算结点 *u.child* 中存储的文件向量与查询向量相关分数的算法;*GDFABTS*(*u.rightsib*)为回溯算法;*SORT*(*Rlist*, *u.child*, $RScore(u.child, \boldsymbol{Q}_W)$)为按照相关分数排序的算法.

图 7 是由图 4 的聚类结果根据构建索引算法生成的索引树,以图 7 中的索引树为例,设 $K=4$, $n=5$,查询向量 $\boldsymbol{Q}_W=(0,1,0,1,1)$,图 7 中的向量都尚未加密.从根结点 *A* 开始搜索,先计算查询向量和根结点 *A* 的 2 个子结点 *B* 与 *C* 的相似度分数.由于查询向量与结点 *C* 的相似度分数高,接下来搜索结点 *C* 的子结点,经过计算得到查询向量与结点 *H*,*G* 中 *G* 相似度最高,接着查询结点 *G* 的子结点,根据检索算法,由于 *G* 的孩子结点是叶子结点,则用直接插入算法按照相关度分数的大小依次将其插入 *Rlist* 中.要求返回 4 个文件,目前 *Rlist* 中只有 2 个文件,则回溯到结点 *C* 查询另 1 个孩子结点 *H*.同样利用直接插入算法将其孩子结点按着相似度分数插入 *Rlist*.最终返回文件的是图 7 中的①②③④.

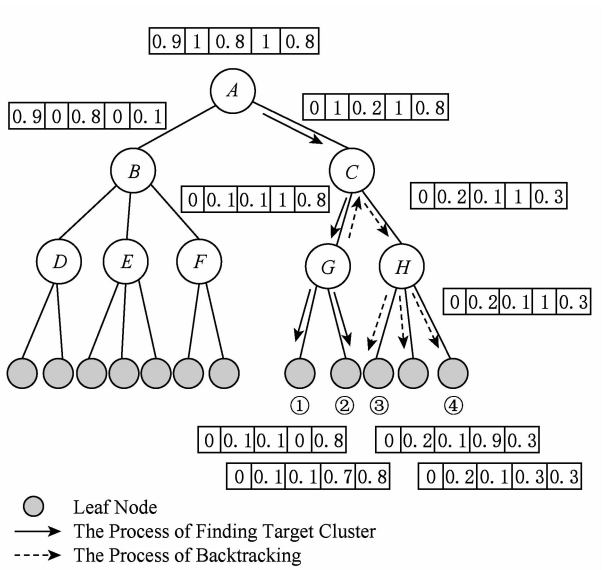


Fig. 7 Index tree

图 7 索引树

4 检索效率、精度与安全分析

4.1 查询效率分析

查询过程可以分为陷门生成与查询 2 部分.如表 1 所示,传统的 MRSE,EDMRS,CTMRSE 方案陷门生成的时间复杂度只与关键词个数有关,不随着文件集的增长而增长,时间复杂度都为 $O(1)$.其次,在查询阶段传统的 MRSE 方案未建立索引,服务器需要计算查询向量与每个文件向量之间的相似度并排序返回 top-*K* 个文档,时间复杂度为 $O(m)$;EDMRS 方案在搜索过程中利用深度优先搜索算法,搜索的时间复杂度是 $O(\theta n \operatorname{lb}(m))$.其中 θ 为查询中包含 1 个或多个关键字的叶结点数量,θ 一般大于所需文件数量 *K*,但远远小于文件数量 *m*. $\operatorname{lb}(m)$是平衡二叉树的高度,$O(n)$为计算相关度分数的复杂度.CTMRSE 方案先聚类再构建索引树,查询时使用本文检索算法可以快速的定位到最匹配的簇,从根结点到目标簇的路径上涉及的结点总数

不超过 $\log_{\lceil h/2 \rceil}\frac{(m+1)}{2}+1$，因此在最坏情况下检索的时间复杂度为 $O\left(\eth n\log_{\lceil h/2 \rceil}\frac{(m+1)}{2}\right)$. 其中，$h$ 为每个簇含有文件数量的最小值，$\eth$ 为用 CTMRSE 方案检索需要求相似度的叶子结点数，$K<\eth<\theta$.

Table 1 Time Complexity of Each Scheme

表 1 各方案时间复杂度表

Scheme	GenTrapdoor	Query
MRSE	$O(1)$	$O(m)$
EDMRS	$O(1)$	$O(\theta n\,\mathrm{lb}(m))$
CTMRSE	$O(1)$	$O\left(\eth n\log_{\lceil h/2 \rceil}\frac{(m+1)}{2}\right)$

4.2 检索精度分析

文档之间的关系通常被隐藏在加密过程中[15]，这将导致显著的搜索精度性能下降. 较好的查询结果应该保持查询文档与文档之间的相似度，相似度越高，检索精度越高. 本方案检索精度量化为

$$S_K=\sum_{i=1}^{K}RScore(\boldsymbol{Q}_W,\boldsymbol{D}_i)\Big/\sum_{i=1}^{K'}RScore(\boldsymbol{Q}_W,\boldsymbol{D}_i),\quad(7)$$

其中，K 为最终密文检索返回的前 K 个文件，K' 是明文查询中返回的前 K' 个文件，$RScore(\boldsymbol{Q}_W,\boldsymbol{D}_i)$ 为查询向量 $\boldsymbol{Q}_W$ 与返回结果集中文档的相似度.

传统的 MRSE 方案在加密 $\boldsymbol{D}_i$ 过程中以及 EDMRS 方案在索引构建过程的阶段将文档之间的相关性忽略，导致检索精度性能下降. 经 3.1 节式(6)推导，CTMRSE 方案中相关度分数 $Rscore=\boldsymbol{TD}\cdot\boldsymbol{I}_i=\boldsymbol{D}_i\cdot\boldsymbol{Q}_W$，即密文检索计算的相关度分数与明文检索的相同. 另外，该方案在建立索引之前对文件聚类，经过检索算法搜索返回结果集的文件在同一个簇或者相邻簇，而文件的相关度与聚类质量有关，聚类质量越高搜索结果集的文件相关度越高. 为了提高聚类质量，在聚类过程中引入杰卡德相似系数来计算文件向量之间的相似度以及设定合适的阈值降低误差率. 因此查询向量与结果集的文件相关度高.

4.3 安全分析

1) 索引和查询保密性

CTMRSE 方案用随机的 $(n+u+1)$ 维向量 $\boldsymbol{S}$ 对文档向量 $\boldsymbol{D}_i$ 与查询向量 $\boldsymbol{Q}_W$ 进行分割，可以保障文档在已知背景攻击模型中的索引确定性和查询保密性. 同时通过矩阵 $\boldsymbol{M}_1$ 和 $\boldsymbol{M}_2$ 对向量进行变换的难以确定性，使保密性进一步增强.

2) 查询无关联性

向量添加了 u 维同时从中随机选择 v 维置为 1，且最后 1 维的值被置为随机值，相同的搜索请求将产生不同的查询向量并接收不同的相关性分数分布，从而更好地保护查询无关联性.

3) 关键字隐私

云服务器能够通过分析关键词的 TF 分布直方图来推断识别关键词[16]. CTMRSE 方案采取添加维度且随机赋值的加密方式能抵抗已知背景模型中的统计攻击.

5 性能分析

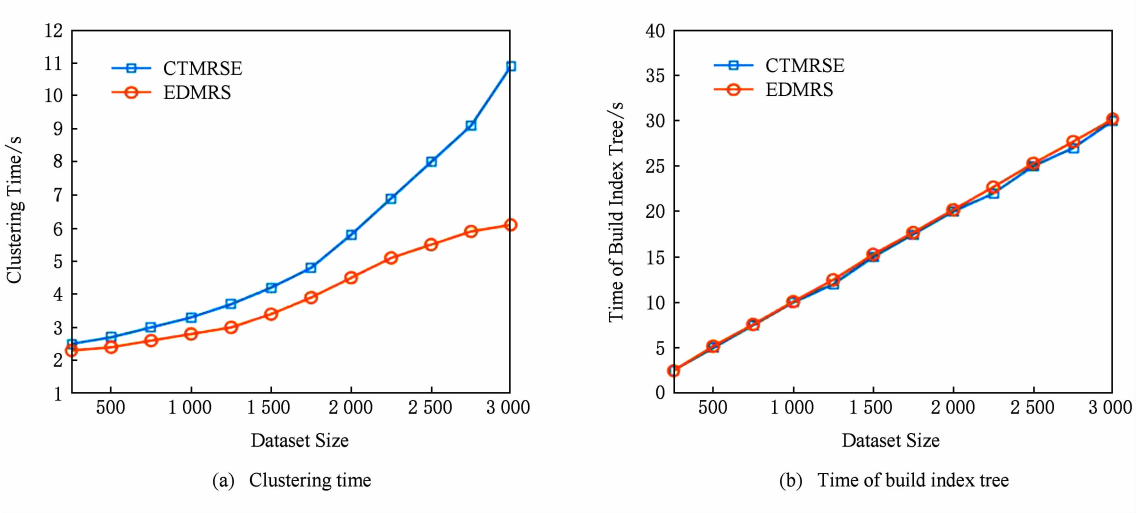
本文实验使用国内云存储提供商阿里云的云存储平台(搭载 centos 7.3 64 位系统、主频为 2.5 GHz 的 4 核 CPU、16 GB 内存、内网宽带为 0.8 Gbps、公网宽带为 100 Mbps、系统盘为 40 GB 高效云盘)搭建存储系统. 原型系统的开发和测试环境是基于 centos 7 的 Linux[17] 平台，具体硬件配置是 intel® Core™i7-6700(3.40 GHz)处理器，配备 8 GB 内存和网速为 1 Gbps 的校园网环境. 实验数据使用 20 431 篇英文新闻作为测试数据集[18]，共有 20 个类别，类别是非均匀的. 然后通过全文检索工具 Lucene[19] 用分词器对纯文本字节流进行分词，滤掉 26.1%的停用词(如 as，but)，对这些文档进行关键词提取形成关键词数为 7 200 的关键词集合.

5.1 聚类效率与精度

首先为了验证改进的方案提高了聚类效率，对改进方案与未改进方案的聚类时间进行实验，实验使用 $m=3\,000$ 的文件集合，字典大小 $n=1\,000$. 实验结果如图 8(a)所示，未改进方案聚类时间比改进方案所需时间明显长. 因此，用改进的方案使得聚类效率有了很大提升. 图 8(b)是本文方案与文献[9]中的 EDMRS 方案对于构建索引所需时间进行对比的实验，实验表明 EDMRS 方案构建二叉平衡树与本方案根据聚类结果动态构建索引树所用的时间基本保持一致.

由于 k 值与度量函数取值过小容易发生过拟合，较大则导致近似误差增大，对聚类结果有巨大影响，实验通过聚类结果的误差率来确定 k 值与 $minMex$ 取值. 文件数量相同时，如图 9(a)当文件数量 $m=1\,000$ 时，$k=6$ 时误差最小. 文件数量变化时，k 值随着文件数量变化而改变. 对应的 k 值也通过聚类结果的误差率来确定；$minMex$ 的取值不同得到聚类结果的误差率不同，如图 9(b)所示，文件数量 $m=1\,000$，$minMex=0.18$ 时误差率最小. 但

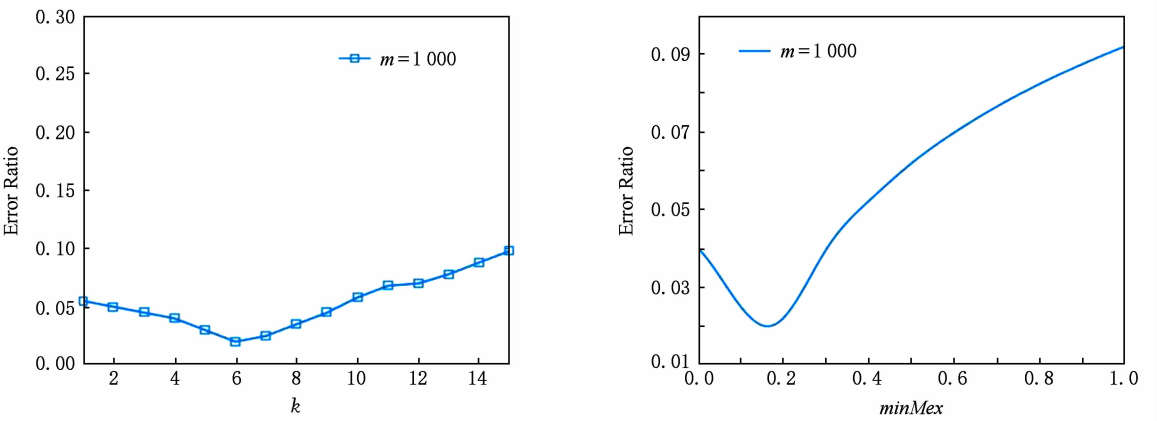
$minMex$ 取值随着文件数量的变化保持不变.因此,通过对 k 值与 $minMex$ 设定合适的取值可以使聚类结果误差率降低,提高了聚类结果质量,从而提高检索的精度.



(a) Clustering time (b) Time of build index tree

Fig. 8 Clustering effect

图 8 聚类效果



(a) Error ratio with the increasing k (b) Error ratio with the increasing $minMex$

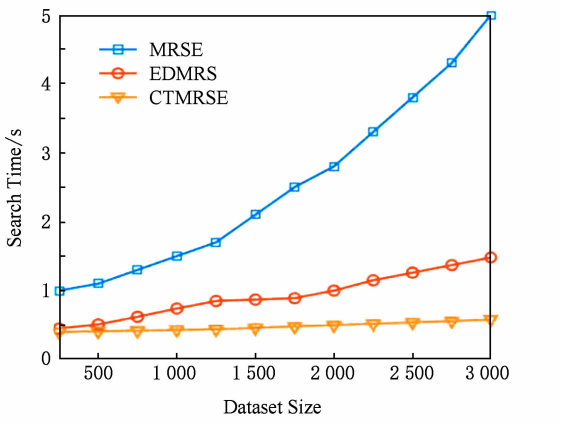
Fig. 9 Error ratio with the increasing thresholds

图 9 取不同阈值对应的误差率

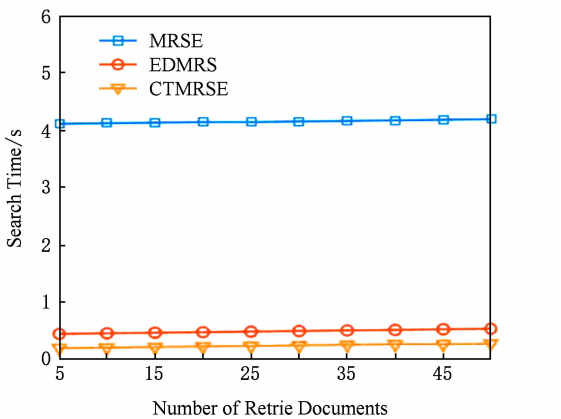
5.2 检索效率

授权用户在进行检索时,总希望快速地得到检索结果,为了检测 3 个方案的检索效率,分别在文件数量、关键词个数以及授权用户要求返回文档数量不同时进行实验,实验结果如图 10 所示.由于方案 MRSE 查询时需要计算所有的密文文档向量和密文查询向量之间的相似度,时间复杂度为 $O(m)$,在图 10(a)中,当文档集合的文档数量按着指数级增加时,MRSE 的方案的查询响应时间也呈指数级增长.而 EDMRS 和 CTMRSE 的方案的时间复杂度与索引树的层次相关,都是近似线性增长,但是相同的文件数量构建的索引树的层次 CTMRSE 方案比 EDMRS 少,因此 CTMRSE 方案查询时间增长更缓慢,检索效率最高;无论用户要求返回多少文件,MRSE 方案没有用到索引树,需要与所有的文件向量计算相似度,再根据相似度大小进行排序,在图 10(b)中检索时间几乎不变,但是该方案的检索时间最长.随着用户要求返回文档数量的增加,查询时要计算相似度的叶结点数量增多,即 θ 与 δ 的值增大,在图 10(b)中 EDMRS 和 CTMRSE 方案的检索时间都增长,但是比较稳定.由于 $\theta>\delta$,CTMRSE 方案的查询时间最少;随着关键词集合中关键字数量的增多,在计算查询向量与文档向量的相似度时,计算时间增长,尤其是 MRSE 方案,要计算与所有文件向量的相似度,在图 10(c)中 MRSE 方案增长幅度最高,查询时间最长.EDMRS 和 CTMRSE 方案的查询时间也会随着关键词集合中关键字数量的增多而增长,图 10(c)中,EDMRS 和 CTMRSE 方

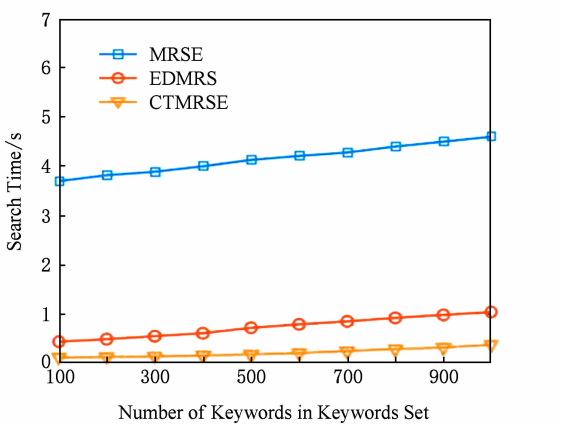
案的查询时间增长的都比较稳定,CTMRSE 方案的查询效率最高.



(a) Search time with the increasing document



(b) Search time with the increasing retrie documents



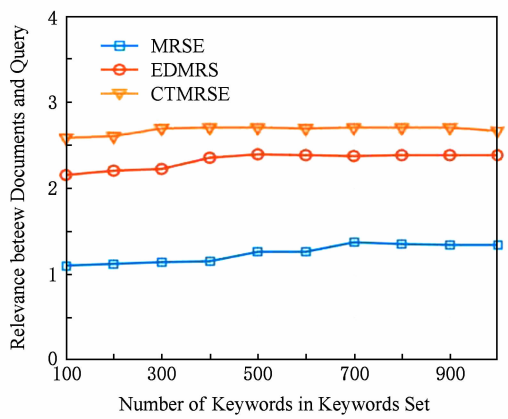
(c) Search time with the increasing keywords

Fig. 10 Query efficiency

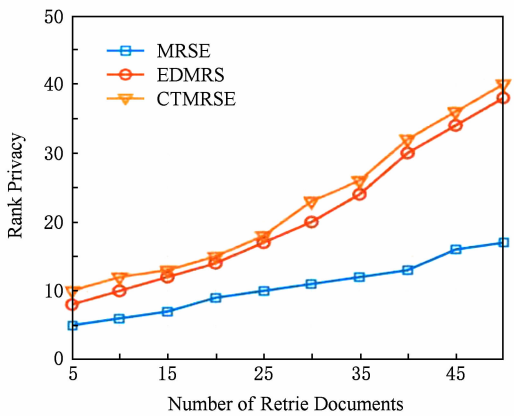
图 10 查询效率

5.3 结果集相似度与排序隐私度对比

用户不仅看重检索效率,还在乎检索的精度和隐私安全.实验通过查询向量与结果集的相似度来度量检索的精度,如图 11(a)所示,当关键词数量变化时,CTMRSE 方案相似度最高,具有明显优势.实验通过结果集排序隐私度检测 MRSE,EDMRS,CTMRSE 方案的隐私安全性,如图 11(b)所示,CTMRSE 方案的排序隐私更好,安全性更高.



(a) Search precision



(b) Rank privacy

Fig. 11 Search precision and rank privacy

图 11 结果集相似度与排序隐私度

6 总 结

本文提出了基于聚类索引的多关键字排序密文检索方案(CTMRSE).该方案先聚类再建立索引,聚类时通过记录关键词位置对向量进行降维,减少聚类过程中不必要的计算消耗.同时,引入了杰卡德相似系数来计算高维向量之间的相似度,并给 Chameleon 算法设定合适的 k 值与 $minMex$ 值减少聚类结果的误差率,提高聚类质量与检索精度.该方案还提出了相应的索引结构和相应的算法,并通过建立密文检索实验平台验证了本方案在保障数据隐私安全的同时,有效的提高了密文检索的效率与精度.但 CTMRSE 方案是在文件集不变的情况下进行的实验,当添加、删除和修改文件时动态更新索引树,以及数据改变后验证检索结果的真实性、完整性是未来需要进行的工作.

参 考 文 献

[1] Dong Xiaolei, Zhou Jun, Cao Zhenfu. Research progress of searchable encryption [J]. Journal of Computer Research and Development, 2017, 54(10): 2107-2120 (in Chinese)
(董晓蕾, 周俊, 曹珍富. 可搜索加密研究进展[J]. 计算机研究与发展, 2017, 54(10): 2107-2120)

[2] Song Xiaodong, Wagner D, Perrig A. Practical techniques for searches on encrypted data [C] //Proc of the 21st IEEE Symp on Security and Privacy 2000. Piscataway, NJ: IEEE, 2000: 44-55

[3] Liesdonk P V, Sedghi S, Doumen J, et al. Computationally efficient searchable symmetric encryption [C] //Proc of VLDB Conf on Secure Data Management. Berlin: Springer, 2010: 87-10

[4] Curtmola R, Garay J, Kamara S, Ostrovsky R. Searchable symmetric encryption: Improved definitions and efficient constructions [C] //Proc of the 13th ACM Conf Computer and Communications Security. New York: ACM, 2006: 79-88

[5] Cao Ning, Wang Cong, Li Ming, et al. Privacy-preserving multi-keyword ranked search over encrypted cloud data [J]. IEEE Transactions on Parallel & Distributed Systems, 2013, 25(1): 222-233

[6] Sun Wenhai, Wang Bing, Cao Ning, et al. Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking [J]. IEEE Transactions on Parallel & Distributed Systems, 2014, 25(11): 3025-3035

[7] Chen Xiaofeng, Huang Xinyi, Li Jin, et al. New algorithms for secure outsourcing of large-scale systems of linear equations [J]. IEEE Transactions on Information Forensics & Security, 2014, 10(1): 69-78

[8] Li Hongwei, Yang Yi, Luan T H, et al. Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data [J]. IEEE Transactions on Dependable & Secure Computing, 2016, 13(3): 312-325

[9] Xia Zhihua, Wang Xinhui, Sun Xingming, et al. A sucure and dynamic multi-keyword ranked search scheme over encrypt cloud data [J]. IEEE Transactions on Parallel and Distributed System, 2016, 27(2): 951-963

[10] Tian Xue, Zhu Xiaojie, Shen Peisong, et al. Fast cryptographic retrieval based on similar query tree [J]. Journal of Software, 2016, 27(6): 1566-1576 (in Chinese)
(田雪, 朱晓杰, 申培松, 等. 基于相似查询树的快速密文检索方法[J]. 软件学报, 2016, 27(6): 1566-1576)

[11] Wang Chong. R-tree index method based on Chameleon clustering algorithm [D]. Harbin: Harbin Institute of Technology, 2017 (in Chinese)
(王冲. 基于 Chameleon 聚类算法的 R 树索引方法研究[D]. 哈尔滨: 哈尔滨工大学, 2017)

[12] Chen Chi, Zhu Xiaojie, Shen Peisong, et al. An efficient privacy-preserving ranked keywords search method [J]. IEEE Transactions on Parallel & Distributed Systems, 2016, 27(4): 951-963

[13] Yang Yang, Yang Shulüe, Ke Min. Ranked fuzzy keyword search based on simhash over encrypted cloud data [J]. Chinese Journal of Computers, 2017, 40(2): 431-444 (in Chinese)
(杨旸, 杨书略, 柯闽. 加密云数据下基于 Simhash 的模糊排序搜索方案[J]. 计算机学报, 2017, 40(2): 431-444)

[14] Witten I H, Moffat A, Bell T C. Managing gigabytes: Compressing and indexing documents and images [J]. IEEE Transactions on Information Theory, 1995, 41(6): 79-80

[15] Yang Jun, Liu Zheli, Li Jin, et al. Multi-key searchable encryption without random oracle [J]. Intelligent Networking and Collaborative Systems, 2014, 30(1): 179-190

[16] Hui Zhen, Feng Dengguo, Zhang Min, et al. A secure index against statistical analysis attacks [J]. Computer Research and Development, 2017, 54(2): 295-304 (in Chinese)
(惠榛, 冯登国, 张敏, 等. 一种可抵抗统计攻击的安全索引[J]. 计算机研究与发展, 2017, 54(2): 295-304)

[17] Red Hat. CentOS-7-x86_64-DVD-1708. iso [OL]. [2017-01-28]. http://is oredirect. centos. org/centos/7/isos/x86_64/CentOS-7-x86_64-DVD-1708. iso

[18] Jason R. 20_newsgroups. tar. gz [DB/OL]. [2017-01-29]. http://download. cs dn. net/index. php/mobile/source/download/bukaohuaxue/851012

[19] Apache. Lucene3 [OL]. [2017-09-28]. http://Jakarta. apache. org/lucene

Du Ruizhong, born in 1975. PhD, professor. His main research interests include network security, trusted computing and network technology.

Li Mingyue, born in 1993. Master candidate. Her main research interests include network security, trusted computing and network technology.

Tian Junfeng, born in 1964. PhD, professor, PhD supervisor. His main research interests include distributed computing, network security, network technology.