一种面向隐私保护的密文检索算法

陈超群[1] 李志华[1,2]
(江南大学物联网工程学院 无锡 214122)[1]
(江南大学物联网应用技术教育部工程研究中心 无锡 214122)[2]

摘 要 针对移动云计算环境下数据外包所带来的安全问题,为了保证数据的安全性和密文检索的效率,通过改进传统的密文检索结构,增加私有云索引服务器以实现索引文件与密文文件的存储分离,并在此基础上提出了一种面向隐私保护的密文检索算法。考虑到移动设备的弱计算能力,算法采用对称可搜索加密的方式以减少计算开销,并以 Trie 树作为索引结构以提高检索效率,同时支持对检索结果排序。理论分析与实验结果表明,该算法能够实现对用户的隐私保护,并具有较好的存储空间和检索时间的性能。

关键词 云计算,密文检索,隐私保护,可搜索加密,Trie 树
中图法分类号 TP309.2 **文献标识码** A

Privacy-preserving Oriented Ciphertext Retrieval Algorithm

CHEN Chao-qun[1] LI Zhi-hua[1,2]
(School of IOT Engineering, Jiangnan University, Wuxi 214122, China)[1]
(Engineering Research Center of IOT Technology Application Ministry of Education, Wuxi 214122, China)[2]

Abstract Aiming at the safety problem of outsourcing data to cloud servers for mobile cloud environment, in order to ensure the safety of data and the efficiency of ciphertext retrieval, by improving the traditional ciphertext retrieval system structure, a private cloud index file server is introduced for separating the index file and ciphertext file. Based on this, a privacy-preserving oriented ciphertext retrieval algorithm was proposed. Considering the weak computing ability of mobile devices, to reduce the computational overhead, a symmetrical searchable encryption scheme is adopted. Furthermore, an index structure based on trie tree was proposed to improve the efficiency of retrieval, and support retrieve for result. The theoretical analysis and experimental results show that the proposed scheme can achieve the privacy guarantee, and has a good performance on the stored space and retrieval time.

Keywords Cloud computing, Ciphertext retrieval, Privacy-preserving, Searchable encryption, Trie

1 引言

随着云计算技术的飞速发展,越来越多的用户享受到了由云计算服务带来的便利。利用云端存储数据可以不受空间、时间的约束,并且不需要用户花费大量的资金去购买和维护物理存储设备,其使用便捷、低成本等优势吸引了越来越多的普通用户和企业用户。为了节省本地存储开销,用户往往会将一些私人文件,例如邮件信息、会议文档和个人笔记等个人敏感信息存储到云端。但对于用户来说,由于云服务供应商(Cloud Service Provider, CSP)的可信度是未知的,一旦将数据外包给 CSP,就将失去对数据的控制,面临网络攻击和 CSP 中不可信(诚实而好奇)管理员的双重威胁[1],个人隐私信息存在着被泄露的风险。为了实现对用户信息的隐私保护,一种最简单有效的方法就是在客户端对敏感数据进行加密处理,然后再外包给 CSP,以保证数据在传输和存储过程中的安全性,达到对用户信息隐私保护的目的。

虽然对数据进行加密存储解决了用户的隐私安全问题,但是当用户需要检索这些信息时,由于存储在云端的都是密文数据,用户无法直接获得明文信息,因此不能利用传统的明文信息检索算法进行检索。对于密文检索,如果将所有数据文件下载到本地解密后再进行检索是不现实的,因为这一方面增加了网络负载,另一方面也增加了本地存储和计算资源的开销。同时,用户在进行检索时,应当保证检索信息不被泄露,防止恶意攻击者通过检索信息获得用户的明文信息。因此,密文检索给数据检索带来了很大的困难,设计一种安全、高效的面向隐私保护的密文检索算法具有重要的意义和价值。

为了实现对密文数据的检索,文献[2,3]采用了公钥加密搜索算法,用户可以利用公钥进行数据加密上传,但只有数据拥有者才可以使用私钥进行检索。算法计算复杂度高,加解密和检索效率相对较低。文献[4]以布隆过滤器作为索引结构,对关键词进行检索时需要对每个文件进行计算与判断,检索时间性能较低。SONG 等人[5]第一次提出了基于对称密钥的单关键词可检索加密方法,采用流密码方法对字符型数据

本文受江苏省科技厅产学研前瞻项目(BY2013015-23)资助。
陈超群(1990-),男,硕士生,主要研究领域为云计算与云安全,E-mail:mr.ccq@foxmail.com;李志华(1969-),男,博士,教授,主要研究领域为网络技术、信息安全技术等。

进行加密处理，对关键词进行检索时，需要对文件内容进行遍历查询，逐词匹配密文信息，存储开销大，检索时间性能较低，不适应于海量数据检索。文献[6]提出了一种支持排序的密文检索算法，采用了基于树的检索结构，将检索结果按关键词的相关度分数从大到小返回给用户，但是存储开销较大。文献[7]提出了一种支持多关键查找的安全高效查找算法，采用基于二叉排序树结构，虽然具有较高的存储空间性能，但是检索时间性能低。文献[8]给出了一种基于 Trie 树的索引结构，虽然能提升检索时间性能，但是树中的每个节点都拥有一个固定长度的子节点数组，会增加额外的存储开销。

因此为了提高密文检索的时间性能和降低存储开销，并针对移动云环境下移动终端设备计算和存储能力较弱的特点，以及支持对检索结果排序，提出了一种面向隐私保护的密文检索算法(Privacy-Preserving Oriented Cihpertext Retrieval Algorithm, PPCR)，从以下 4 个方面实现密文检索：1)增加私有云索引服务器，实现索引文件与密文文件的存储分离，提高系统的安全性；2)采用对称可搜索加密的方式，具有计算开销小、算法简单、速度快的特点，适用于移动终端设备；3)通过优化 Trie 树索引结构提高密文检索的效率；4)对检索结果进行排序，按关键词的相关度分数大小返回给用户，减少网络负载。

2 问题描述

2.1 密文检索框架

传统的密文检索结构如图 1 所示，主要由 3 部分组成：数据拥有者、用户和云服务器。数据拥有者负责对明文数据文件集合 F 进行加密，然后将密文文件集合 C 上传到云服务器。为了保证密文文件可被检索到，需要为 F 构建可搜索的加密索引 I，同样将其上传到云服务器。用户只有从数据拥有者处获得授权后，才能对密文进行检索。云服务器上则存储了数据拥有者上传的索引文件和加密数据文件，并负责根据用户提交的陷门返回相关的密文文件。对于攻击者来说，只要统计用户每次检索的关键词信息，就可获得该关键词和文件之间的映射关系，分析出某些文件的部分信息，从而破坏了密文检索的安全性。

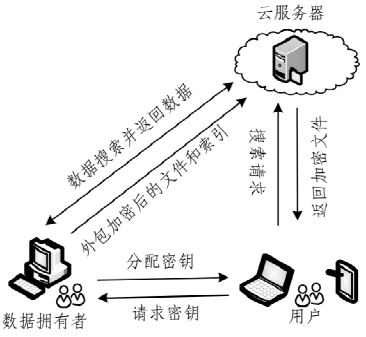

图 1 传统的密文检索框架

2.2 主要符号说明

F：待加密外包的文件集合，表示为 n 个数据文件的集合，即 $F=\{f_1,f_2,\cdots,f_n\}$。

FID：文件集合 F 中所有文档的文件标识符，$FID=\{fid_1,fid_2,\cdots,fid_n\}$。

W：文件集合 F 中所有不同关键词的集合，$W=\{w_1,w_2,\cdots,w_n\}$。

FID_w：文件集合 F 中所包含关键词 w 的文件集合。

I：为可隐私保护的所有关键词检索而构造的索引，$I=\{I_1,I_2,\cdots,I_n\}$。

I_i：文件集合 F 中文件 f_i 中关键词的索引集合，$I_i=\{I_{w_{i1}},I_{w_{i2}},\cdots,I_{w_{im}}\}$。

$List_i$：文件集合 F 中文件 f_i 中关键词的集合，$List_i=\{w_{i1},w_{i2},\cdots,w_{im}\}$。

T_{w_i}：陷门，即用户输入搜索关键词 w_i 后由单向函数所生成的检索请求。

C：文件集合 F 中经加密后的密文文件集合，$C=\{c_1,c_2,\cdots,c_n\}$。

2.3 预备知识

相关度分数评价：在信息检索中，为了评价一个字词在一个文件集合中的其中一份文档中的重要程度，最广泛使用的就是 TF-IDF 统计方法。其中，TF 指的是关键词词频，即某个给定的关键词在一份文档中出现的频率，定义如下：

$$tf_w=\frac{n_w}{\sum w_i} \tag{1}$$

其中，n_w 表示关键词 w 在该文档中出现的次数，$\sum w_i$ 表示该文档包含的关键词总数。IDF 指的是逆文档频率，如果包含某个关键词的文档较少，那么说明该关键词具有良好的文档区分能力，定义如下：

$$idf_w=\log\left(\frac{N}{nf_w}\right) \tag{2}$$

其中，N 表示文档集合的文件总数，nf_w 表示包含关键词 w 的文件数目。通过 TF 和 IDF 的乘积可以得到某个关键词的相关度分数，如式(3)所示：

$$tf\text{-}idf_w=tf_w\times idf_w \tag{3}$$

通过计算关键词的相关度分数，可以实现对检索结果进行排序。

Trie 树：又称字典树，是哈希树的一种变种，利用这种多叉树结构可以实现快速检索。因为其利用字符串的公共前缀减少了必要的字符串比较，减少了查询时间，因此它的查询效率要比哈希树高。Trie 树具有以下 3 个基本性质：1)根节点不包含任何字符，除根节点以外的任一节点只包含单个字符；2)从根节点到树中的某一节点，将该路径上的所有字符连接起来，可以得到该节点所对应的字符串；3)每个节点的所有子节点不能包含相同的字符。

Trie 树的查找过程为：1)从根节点出发，比较要查找的关键词的首字符，根据该字符选择对应节点的子树，并转到该节点的子树继续进行检索；2)在相应节点的子树上，比较要查找的关键词的第二个字母，再选择该字符对应节点的子树进行检索；3)重复执行步骤 2)，直到某个节点要查找的关键词的所有字符都被取出，则读取该节点上的信息，完成关键词检索。

2.4 隐私保护要求

为了保护用户数据的隐私安全，密文检索算法必须满足以下隐私要求：

(1)数据隐私：应当保证数据文件的保密性和私密性，除了合法用户，任何人无法从云服务器上的密文中得到明文信息。

(2)索引隐私：如果云服务器端可以推理出关键词和文件之间的联系，则有可能获得文件的部分内容，因此查询索引不能泄漏其相应关键词的任何信息。

(3)陷门隐私:对于陷门函数,产生的关键词陷门应当是随机的,云服务器端无法推导出任意陷门之间的关系,从而产生其他合法的陷门,保证了陷门的隐私。

3 面向隐私保护的密文检索算法

3.1 新的密文检索框架

为了提高密文检索的安全性,提出了一种新的密文检索结构,将索引文件与加密数据文件进行存储分离,新的密文检索框架如图 2 所示,主要包含 3 部分:数据拥有者、用户和云服务器,其中云服务器包括私有云索引服务器和公有云文件服务器。数据拥有者可以是个人或者组织机构,为了防止敏感信息被未经授权的用户使用,需要对数据文件集合 F 进行加密处理,然后再上传至公有云文件服务器。为了方便以后对密文进行检索,在上传之前需要对文件集合 F 进行预先处理,提取关键词集合 W 并构造索引文件。同时,为了保证关键词索引集合信息的保密性,应当对关键词索引进行加密处理,避免索引服务器获取用户的明文信息。用户对云服务器中的密文文件进行检索时,首先需要从数据拥有者那获得访问授权,即得到密钥 K_1,K_f。当密钥分配完成后,对于任意一个被授权的用户,都可以利用密钥生成检索关键词的陷门,并将请求提交给索引服务器。索引服务器负责将检索到的文件标识符集合按照关键词的相关度分数大小进行排序,然后将排序后的结果提交给文件服务器,文件服务器在未解密数据的情况下将对应的密文文件按顺序返回给用户。用户得到公有云文件服务器返回的密文文件集合 C 后,利用数据拥有者分发的密钥对密文进行解密 $f_i=dec(k_i,c_i)$,得到明文检索结果。

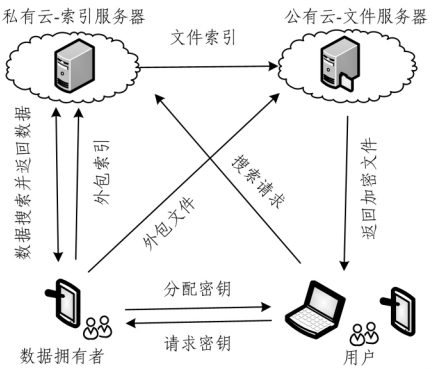

图 2 新的密文检索框架

实现索引服务器与文件服务器的分离后,需要建立相互之间的映射关系,实现索引到密文文件之间的对应,映射结构如图 3 所示。

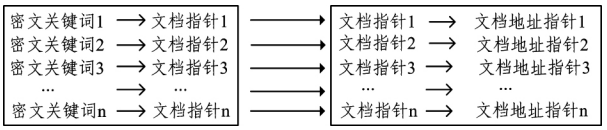

图 3 密文检索索引结构

3.2 索引树构造

为了提高关键词的检索效率,构造了基于 Trie 树的索引树结构,如图 4 所示。树中节点主要包括以下两部分信息:关键词陷门对应的当前字符以及关键字陷门对应的文件标识符与相关度分数的集合。节点的信息如表 1 所列。

表 1 节点信息

Word	w_i			
File ID	fid_1	fid_2	…	fid_k
Relevance score	$score(w_i,fid_1)$	$score(w_i,fid_2)$	…	$score(w_i,fid_k)$

索引树的构造过程为:从根节点出发,将陷门的当前字符插入到当前节点的适当位置,然后沿着该节点当前字符对应的子树往下遍历,直到陷门遍历完成,在最后的字符处标记为陷门,表示该陷门已插入索引树,同时将该陷门对应的信息保存到该节点。

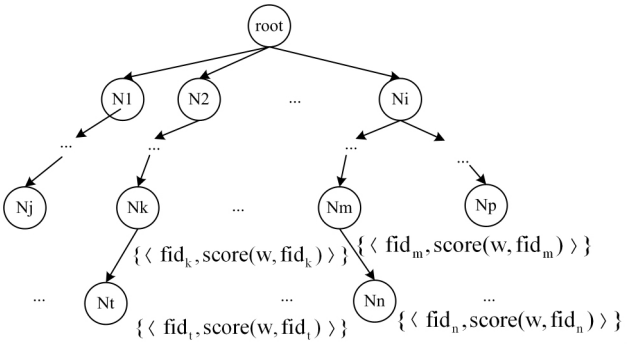

图 4 索引树

3.3 索引树优化

由于 Trie 树的每个节点都保存了一个长度固定的节点数组,假设节点数组的长度为 L,节点大小为 M,共有 N 个节点,则需 $L*M*N$ 的存储空间,但数组的每个位置不一定保存有节点记录,如果 N 相当大,则存在着严重的存储空间浪费问题。为了节省存储空间,同时保持原有的查找效率,需要对 Trie 树进行优化。对 Trie 树进行如下改进:根节点不包含字符,树中的节点的子节点不再用数组表示,每个节点只包含 LeftChild,CenterChild 和 RightChild 3 个子节点。在索引树进行关键词插入时,首先判断插入的关键词当前字符与当前节点字符的关系,然后根据其大小关系将节点插入到相应位置。经优化后,索引树中的节点定义如下:

$$N=\langle D,U,N_r,N_c,N_l\rangle$$

其中,D 表示关键词的当前字符,N_r,N_c,N_l 分别表示当前节点的右子节点、中间子节点和左子节点,U 表示保存的文件相关信息,定义如下:

$$U=\{\langle fid_i,score(w,fid_i)\rangle\}$$

索引树插入算法如算法 1 所示。

算法 1 InsertIndexTree

```
Require:a set of words{w1,w2,…,wi}
for all words wi do
  for each char ch∈wi
  if(value(ch,N.D)==0)
      Nc.D==ch;
  else if(value(ch,Nc.data)<0)
      Nl.D=ch;
  else
      Nr.D=ch;
  end for
end for
```

优化后的索引树节点查找过程为:从根节点出发,比较要查找关键词 w_i 陷门的当前字符与节点当前字符的大小,若 w_i 陷门的当前字符的值大于节点当前字符的值,则转到该节

点的 RightChild，与下一个节点的字符继续进行比较；若相等，则转到节点的 CenterChild 与下一个节点的字符进行比较；若小于，则转到节点的 LeftChild，与下一个节点的字符继续进行比较。重复执行上述过程，直到某个节点要查找的关键词陷门的所有字符都已被取出，则读取该节点上的信息，完成关键词检索。

索引树查找算法如算法 2 所示。

算法 2　SearchIndexTree

```
Require: a set of search words{w1, w2, ···, wi};
for all words wi do
  for ch=wi.charAt(j), j<wi.length()
  if(value(ch, N.D)==0)
      if(j=wi.length())
      return N;
      N=Nc;
  else if(value(ch, N.D)<0)
      N=Nl;
  else
      N=Nr;
  end for
end for
```

3.4　关键词检索排序

获得授权的用户进行关键词检索时，在计算得到关键词的陷门后，将陷门发送至索引服务器，索引服务器将进行如下检索：

(1)利用构建好的索引树对关键词陷门执行索引树查找过程。

(2)通过索引树查找算法检索到该陷门 T 后，根据用户提交的参数 k，以及关键词在文档中的相关度分数大小，按以下式子得到前 top-k 个符合用户要求的文件标识符：

$$Q=K-Max\{\langle fid_i, score(w, fid_i)\rangle\}=\{fid_i \mid fid_i \in FID_w\}, 1\leqslant i\leqslant n$$

然后将 Q 发送至公有云文件服务器。

因为用户往往只需要得到最相关的 k 个文档，而不是所有的文档，所以要避免返回所有相关的文档，增加不必要的网络负载。基于上述原因，需要对检索结果按相关度分数进行排序。为了评价关键词 w_i 在文件 f_i 中的相关度分数，给出一种一般性的 TF-IDF 统计方法，定义如下[9]：

$$score(W, F_{d_j})=\sum w_i\in W \frac{1}{|F_{d_j}|} * (1+\ln f_{d_j, w_i}) * \ln(1+\frac{N}{f_{w_i}}) \tag{4}$$

其中，f_{d_j, w_i} 表示关键词 w_i 在文件 f_{d_j} 中出现的次数；f_{w_i} 表示包含关键词 w_i 的文件数目；$|F_{d_j}|$ 表示文件 f_{d_j} 中的关键词数目；N 表示所有文件的数目。

从式(4)可以发现，对于检索的关键词，其相关度分数的评价主要来自 TF 和 IDF 两部分，因此在本算法中，同样以这两部分来评价相关度分数。同时，为了提高关键词的检索效率，只考虑单关键词检索，因此对式(4)进行如下改进：

$$score(w_i, F_{d_j})=\frac{1}{|F_{d_j}|} * (1+\ln f_{d_j, w_i}) * \ln(1+\frac{N}{f_{w_i}}) \tag{5}$$

数据拥有者在建立索引文件时，通过式(5)计算得到关键词的相关度分数，并写入到索引文件，然后将其上传至索引服务器。

3.5　算法实现

为了实现安全高效的密文检索及对用户的隐私保护，算法实现具体包括以下 4 个步骤：

Step1　$KeyGen(mk)\rightarrow\{1^k, 1^l\}$：用户输入两个安全参数 x_E 和 x_I，算法为用户生成文件加密密钥 $K_f=\{k_1, k_2, \cdots, k_n\}$，$k_i=h_1(x_E \parallel fid_i)$，$K_f\in\{0,1\}^k$，索引加密密钥 $K_I=h_2(x_I)$，$K_I\in\{0,1\}^l$。$h_1(\cdot)$ 和 $h_2(\cdot)$ 为散列函数。

Step2　$IndexBuild(F)\rightarrow I$：按以下步骤为文件集合 F 生成安全的索引：

(1)$WordExtractor(F)\rightarrow W$：对于输入的文件集合 F，构造停用词表，以便去除无意义的字词，通过分词算法提取所有文件的关键词，得到关键词集合 $W=\{\langle fid_1, List_1\rangle, \langle fid_2, List_2\rangle, \cdots, \langle fid_n, List_n\rangle\}=\{w_1, w_2, w_3, \cdots, w_n\}$，同时提取所有文件的文件标识符，得到文件标识符集合 $FID=\{fid_1, fid_2, \cdots, fid_n\}$。

(2)$WordIndex(w_i)\rightarrow x_{w_i}$：为关键词集合中的所有关键词 w_i 生成索引，即构造关键词和文件之间的映射关系 $Index_{w_i}=\{\langle w_i, \{fid_1, fid_2, \cdots, fid_i\}\rangle\}$，$w_i\in List_1\cup List_2\cup\cdots\cup List_n$。

(3)$Encryption(F, x_{w_i})\rightarrow(C, I)$：使用 $Keygen$ 阶段生成的索引加密密钥，构造关键词的陷门 $T_{w_i}=H(w_i, K_I)$，以陷门作为索引文件的安全索引，$I_{w_i}=\{\langle T_{w_i}, \{\langle fid_1, score(w_i)\rangle, \langle fid_2, score(w_i)\rangle, \cdots, \langle fid_i, score(w_i)\rangle\}\rangle\}$，$w_i\in List_1\cup List_2\cup\cdots\cup List_n$。同时，使用文件加密密钥对所有的明文文件进行加密 $c_i=\{enc(k_i, f_i)\}$，$k_i\in k_f$，$f_i\in F$，$1\leqslant i\leqslant n$，enc 是分组加密算法，如 DES 和 AES，H 是一个散列函数，如 MD5、SHA1。经过加密处理后得到加密文件集合 $C=\{c_1, c_2, \cdots, c_n\}$ 和加密索引 $I=\{I_1, I_2, \cdots, I_n\}$。

索引构造的算法实现如算法 3 所示。

算法 3　IndexBuild

```
Require: F: the document collection, KI: the secret key for word
for all documents fi∈F do
  Listi←WordExtractor(fi)
  end for List={List1, List2, ···, Listn}
  for all keyword Listi∈List do
    for all wi∈Listi
      Iwi←⟨Twi(KI), ⟨fidj, score(wi)⟩⟩
    end for
    Ii←{Iwi1, Iwi2, ···, Iwij}
  end for
return I={I1, I2, ···, In}
```

Step3　$Search(w_i)\rightarrow c_i$：对任意的关键词 w_i，用户首先计算它的陷门值 $T_{w_i}=H(w_i, K_I)$，然后将陷门 T_{w_i} 发送到索引服务器，索引服务器通过索引树找到该陷门对应的文件标识符集合，按相关度分数的大小将请求 $Q=(fid_1, fid_2, \cdots, fid_k)$ 发送至文件服务器。

关键词检索算法实现如算法 4 所示。

算法 4　WordSearch

```
Require: a set of search terms{w1, w2, ···, wi}; the secret key KI for
word
for all search terms wi do
  id←Twi(KI)
  if id not in IndexTree
  return
  end if
  Q←(fid1, fid2, ···, fidi), 1≤i≤n
```

return Q

end for

Step4 $Decrypt(c_i)\rightarrow f_i$：文件服务器接收到索引服务器发过来的文件请求后，将密文文件按顺序返回给用户，然后用户利用解密密钥 k_i 解密文件 $f_i=\{dec(k_i,c_i)\}$，得到所需要的明文信息。

3.6 算法分析

3.6.1 空间复杂度

假设在最坏的情况下，索引树是一棵满的 L 层 K 阶树，则这棵树含有的节点数为：

$$\sum_{i=0}^{L}K^i=\frac{K^L-1}{K-1} \tag{6}$$

每个节点占用的存储空间为 M，则索引树所占用的存储空间为：

$$Mem=\frac{K^L-1}{K-1}\times M \tag{7}$$

当 L 和 K 的取值都很大时，即最坏的情况下，索引树的存储开销将非常高。但在本算法中，索引树中间节点存储的为单个字符，M 取值为 1，K 取值为 3，关键词陷门的长度为 32，即 L 的取值大约为 32。同时，由于不同关键词的数量是有限的，因此陷门数量也是有限的，索引树的实际存储开销并不大。

3.6.2 时间复杂度

假设用户查找某一关键词，其生成的陷门为 $t=c_0c_1c_2\cdots c_{i-1}c_i$，索引树的层数为 L，节点所含最大子节点数为 K，即使在最坏的情况下，执行一次陷门查找也只需要进行 $L*K$ 次比较。由之前的分析可知，L 和 K 都是一个较小的常数，因此检索的时间复杂度为 O(1)。由此可见，算法具有较低的时间复杂度。

3.6.3 安全性

云服务供应商被定义为“诚实而好奇”的，因此有可能对用户存储的数据造成威胁，主要存在以下两种攻击方式[10,11]：1)已知密文攻击：攻击者能够获取到用户的密文数据和查询索引；2)已知背景攻击：攻击者除了掌握用户的密文数据，还获取到了部分关键词及其词频等静态信息、部分关键词与密文数据之间的关联信息等。攻击者可以通过统计分析出文档和关键词之间的对应关系，从而获取用户的隐私信息。

针对以上攻击及隐私保护的要求，从以下两方面阐述该算法具有隐私保护的能力。

(1)数据存储安全性

用户数据在发送给 CSP 之前，用户通过私钥对文件和关键词进行了加密处理，加密算法的安全强度保证了密文的安全性。只有授权用户才能获取私钥，未经授权的用户和 CSP 无法对密文数据进行解密，从而得不到文件和关键词的明文信息。同时，为了进一步增强对数据的保护，实现了密文文件与索引文件的存储分离，公有云服务器上存储的只是密文文件，而索引文件存储在私有云服务器上，攻击者无法得到关键词与密文文件之间的关联关系，提高了数据存储的安全性。

(2)检索安全性

参照文献[12]给出的相关安全性定义，设用户与云服务器交互的密文集合为 C，设 fid_i 表示 $enc(k_i,f_i)$ 在 C 中的唯一标识，$Id(u)$ 表示用户 u 在云服务器中的唯一标识，k 为正整数，设 $q_j=(u_j,query_{u_j}(w_1,\cdots,w_j))(1\leqslant j\leqslant k)$ 是用户 $U=(u_1,u_2,\cdots,u_k)$ 发出的 k 个查询请求，设 r_k 是用户 U 请求的相应回复。则云服务器对应于这 k 个查询的视图为 $V_k=(C,\{q_j\}_{1\leqslant j\leqslant k},r_k)$，并可得到这 k 个查询的迹为 $Tr_k=(\{|C_u|_{u\in U}\},|U|,Id(U_k),r_{q_1},\cdots,r_{q_k})$，其中 $|C_u|$ 为用户 u 从云服务器中得到返回的加密文件的个数，$Id(U_k)=\{Id(u_j)_{1\leqslant j\leqslant k}\}$，$r_{q_j}$ 为其内所有信息集合。对于任何信息，如果云服务器可以从 V_k 中获得，那么其均可仅由 Tr_k 得到，则称算法满足查询隐私性。

定理 当 $H(\cdot)$ 是散列函数，$enc(\cdot)$ 是伪随机排列时，算法满足查询隐私性。

证明：要证明 V_k 中所有信息均包含在 Tr_k 中，可以用 Tr_k 尝试构造出一个与 V_k 在计算上不可区分的视图 V_k^*，步骤如下：

首先，随机选取一个 $x_i^*\in\{0,1\}^l$ 作为索引私钥，计算 $q_i^*=(u_i^*,H(w_i^*,x_i^*))$。

生成 $\{q_j\}_{1\leqslant j\leqslant k}$：对 $1\leqslant j\leqslant k$，若 $\neg\exists j(j<i\wedge(r_{q_j}==r_{q_i}))$，选 $u_j^*\in rU$，取 x_j^* 作为其索引私钥，选 $w_j^*\in rW$，设 $q_j^*=(u_j^*,H(w_j^*,x_j^*))$；否则，设 $q_j^*=q_i^*$。易知，若 $H(\cdot)$ 是散列函数，则 q_j^* 与实际的 q_j 在计算上不可区分。

生成 C：对于 $\{q_i^*\}_{1\leqslant i\leqslant t}$ 中涉及到的 u_i^*，由于用户选择的 x_i^* 的随机性，易知 $h_1(x_i^*)$ 与 $h_1(x_i)$ 实际是不可区分的。设用户的加密文件 $enc(k_i,f_i)$ 为 $E_m^*\in\{0,1\}^p$，易知，当 $enc(\cdot)$ 为伪随机排列时，实际密文数据与 E_m^* 是不可区分的。当 $1\leqslant i\leqslant k$ 时，对于 r_{q_i} 所涉及到的文件，选 $c_i^*\in\{0,1\}^p$，对于 q_i^* 所涉及到的 $w_j(1\leqslant j\leqslant n_i)$，设 $T_{w_j}^*=H(w_j,h_2(x_i^*))$，其余的 $T_{w_j}^*$ 则设为随机值。对于 r_{q_i} 未涉及到的文件，其索引值均设为随机值。

生成 r_k：当 $1\leqslant i\leqslant k$ 时，对 r_{q_i} 涉及到的所有文件标识集合，输出其相应的 E_m^* 集合。

由此可知，算法满足查询隐私性。同时，对于攻击者来说，K_I 是未知的，H 函数又是一个单向哈希函数，具有单向性和抗碰撞性，要进行逆向计算是非常困难的，因此索引是不可伪造的，从而保证了算法检索的安全性。

4 实验分析

本文使用 50 篇 IEEE 数据库中的英文论文作为测试数据集。运行环境为 Windows7 64 位操作系统，内存为 4GB，处理器主频为 2.60GHz，开发工具为 Myeclipse 10，开发环境为 JDK 1.7，使用开源的分词工具包 IKAnalyzer 3.2.8 对文件的关键词进行提取，使用 PDFbox 工具包将 pdf 文档转化为文本文档，采用的加密算法为国密 SMS4 算法[13]。

实验主要对 PPCR 索引结构的构造时间、存储空间开销以及关键词检索时间进行测试，并通过与文献[8]所采用的检索算法在存储空间和时间上进行比较，来验证 PPCR 检索算法的有效性。

图 5 为不同检索算法的索引结构在随着文件数量增多时，存储空间开销的变化图。两者虽然都用于对字符串进行检索，但是文献[8]所采用的基于 Trie 树的检索算法，树中的每个节点都拥有一个节点数组，长度为 26。因此，Trie 树中每插入一个字符时，相当于还需要申请额外的 26 个字符的存储空间，如果节点数组中的某些位置没有字符被插入，将造成存储空间浪费。所以当文件数量不断增加时，插入的字符不断增加，额外申请的存储空间就越来越多，索引结构的存储空间开销快速增加。PPCR 检索算法在 Trie 树结构中进行了改

进，当每插入一个字符到树中的节点时，不用再申请额外 26 字符的存储空间，只需要为插入的字符申请存储空间，节约了存储空间，极大地减少了索引结构的存储空间开销。因此，从图 4、图 5 可以看出，当文件数量不断增加时，PPCR 检索算法的存储空间开销远小于文献[8]所采用的检索算法。当文件数目为 50 时，PPCR 检索算法的存储空间开销比文献[26]的检索算法减少了近 1/2，有效地节省了存储空间。

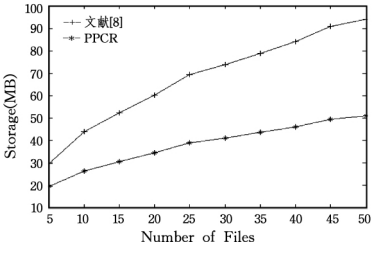

图 5　索引树存储空间开销

图 6 为不同检索算法的索引结构在随着关键词陷门数量增多时，构造时间的变化图。由前面的分析可知，文献[8]采用的检索算法的索引树的每个节点都拥有一个子节点数组，长度为 26，需申请额外 26 个字符的存储空间，而 PPCR 检索算法就不需要申请额外的存储空间。因此从图中可以看出，PPCR 检索算法的索引结构构造时间要明显小于文献[8]采用的检索算法。由此可见，本算法有效地节省了索引结构的构造时间。

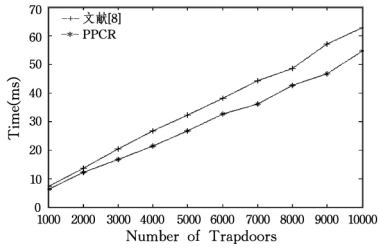

图 6　索引树构造时间开销

用户进行关键词检索时，其检索时间主要由陷门构造时间 $T(TrapdoorGen)$ 和陷门查找时间 $T(Query)$ 组成，定义如下：

$$T(SearchTime)=T(TrapdoorGen)+T(Query) \quad (8)$$

图 7 比较了不同检索算法随着检索关键词数量增加时检索时间的变化。文献[8]采用基于 Trie 树的检索算法，树中的每个节点都拥有一个节点数组，长度为 26。在进行关键词检索时，首先要生成该关键词的陷门，然后按照 Trie 树查找算法，找到陷门当前字符在数组中的位置。如果数组中该位置不存在字符，即该关键词不在索引中；否则，检索成功。因此，对于陷门中的每个字符而言，文献[8]采用的检索算法只需要通过计算一次字符在数组中的位置，然后查看数组当前位置是否存在元素即可。由于只需要进行陷门长度次数的计算，因此时间复杂度为 O(1)，查找效率很高。当关键词数量越来越多时，其检索时间增长较为缓慢。而 PPCR 检索算法每次对陷门的当前字符进行一次大小比较之后，如果大小不相等，还要进行下一次比较。因此其检索速度要比文献[8]采用的检索算法慢。但关键词陷门的长度固定，只需进行常数次比较即可，其时间复杂度也为 O(1)。由图 7 可以看出，文献[8]采用的检索算法与 PPCR 检索算法两者之间的检索时间非常接近，检索效率基本相同。因此 PPCR 检索算法具有较高的检索性能。

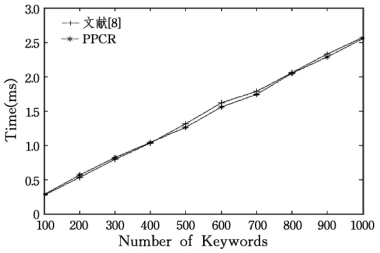

图 7　关键词检索时间开销

结束语　本文提出了移动云计算下一种面向隐私保护的密文检索算法。通过采用对称可搜索加密的方式减少了计算复杂度，并实现了索引文件与密文文件的存储分离，优化了基于 Trie 树的索引树结构，以及通过对文档的相关度分数进行排序，返回满足用户检索的 top-k 个文档，减少了网络负载，很好地解决了移动云计算下的密文检索问题。通过实验表明，本算法在确保安全性的前提下，有效地节省了密文索引的存储空间和构造时间，具有高效的检索性能。未来将对密文检索中密钥管理和访问控制等问题进行深入研究。

参 考 文 献

[1] 程芳权，彭志勇，宋伟，等. 云环境下一种隐私保护的高效密文排序查询方法[J]. 计算机学报，2012，35(11)：2215-2227

[2] Boneh D，Di Crescenzo G，Ostrovsky R. et al. Public Key Encryption with Keyword Search[J]. Eurocrypt，2004，49(16)：506-522

[3] Rhee H S，Park J H，Susilo W，et al. Trapdoor Security in a Searchable Public-key Encryption Scheme with a designated Tester[J]. Journal of and Systems Software，2010，83(5)：763-771

[4] Goh E J. Secure Indexes [EB/OL]. [2012-12-13]. http://eprint. iacr. org

[5] Song D X，Wanger D，Perrig A. Pratical Techniques for Searches on Encrypted Data[J]. IEEE Symposium on Security & Privacy，2012：44-55

[6] Wang C，Cao N，Li J，et al. Secure Ranked Keyword Search over Encrypted Cloud Data[C]// Proceedings of ICDCS. Genova，Italy，2010：253-262

[7] 李倩，岳风顺，王国军. 安全云存储中高效的多关键词查找方案[J]. 计算机科学，2012，39(12)：158-161

[8] Lu Wen-jun，Swaminathan A，Avinash L，et al. Enabling Search over Encrypted Multimedia Databases[C] // Proceedings of SPIE-The International Society for Optical Engineering. 2009

[9] Wang C，Cao N，Ren K，et al. Enabling secure and efficient ranked keyword search over outsourced cloud data[J]. IEEE Transaction on Parallel and Distributed Systems，2012，23(8)：1467-1479

[10] Cao Ning，Wang Cong，Li Ming，et al. Privacy-preserving multi-keyword ranked search over encrypted cloud data[J]. IEEE Transactions on Parallel and Distributed Systems，2011，25(1)：829-837

[11] Chen Chi，Zhu Xiao-jie，Shen Pei-song，et al. an efficient privacy-preserving ranked keyword search method[J]. IEEE Transactions on Parallel and Distributed Systems，2015：1-1

[12] Yang Yan-jiang，Lu Hai-bing，Weng Jian. Muti-user private keyword search for cloud computing[C] // Proceedings of Third IEEE International Conference on Cloud Computing Technology and Science(CloudCom). Athens，Greece，2011：264-271

[13] 李浪，李仁发，李静，等. 一种 SMS4 加密算法差分功耗攻击[J]. 计算机科学，2010，37(7)：39-41