云存储中高效密文检索的中文数据加密方案

张蜀男 蔡 英 范艳芳 夏红科
(北京信息科技大学计算机学院 北京 100101)

摘 要 数据加密是确保云存储数据安全的主要技术,高效的密文检索技术对于提高密文检索效率和减小存储开销起到了决定性作用。大多数已有的基于密文检索的中文数据加密方案需要上传密钥,增加了密钥泄漏的风险。在密文检索过程中,对于关键字的匹配,一些方案需要多次对密文解密,严重降低了密文检索的效率;另一些方案构建了大量的索引文件,浪费了云存储空间。文中提出了一种能兼顾检索效率和存储开销的中文数据加密方案,其在数据加密阶段利用了数据分块随机排序和标号向量加密技术,在密文检索阶段配合构建的索引向量文件对密文进行类明文检索。在整个过程中不需要将密钥上传至云服务器,建立索引向量文件所消耗的存储空间也小于其他基于索引的方案。实验表明,本方案在检索效率和存储开销上优于其他两种方案,并且能在耗费较少时间和存储空间的情况下准确地找到用户存储的数据。

关键词 数据加密,密文检索,检索效率,存储开销
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.06.021

Chinese Data Encryption Scheme of Efficient Ciphertext Retrieving in Cloud Storage

ZHANG Shu-nan CAI Ying FAN Yan-fang XIA Hong-ke
(Department of Computer Science, Beijing Information Science & Technology University, Beijing 100101, China)

Abstract Data encryption is the primary technology to ensure the data safety in cloud storage, and efficient ciphertext retrieval technology shows its promising capability to improve the retrieval efficiency and reduce the storage overhead. During ciphertext retrieving, some schemes need to decrypt the ciphertext for many times, reducing the retrieval efficiency of ciphertext severely. Other schemes construct a large set of index files, which cost a great deal of storage space in cloud storage. In this paper, Chinese data encryption scheme was proposed by taking both retrieval efficiency and storage overhead into consideration, which utilizes the random sorting of data blocks and label vectors encryption in data encryption process firstly, and then cooperates with the index vector files to retrieve the ciphertext in ciphertext retrieving process. Experiment shows that this scheme can find user's data accurately under the condition of consuming shorter time and less storage space.

Keywords Data encryption, Ciphertext retrieving, Retrieval efficiency, Storage overhead

1 引言

随着大数据时代的来临,云存储应运而生,其是一种能方便用户存储数据的应用。云存储在发展过程中面临着3个问题:数据的安全性问题[1]、密文的检索效率问题[2-3]和额外的存储开销问题。本文提出了一种基于密文检索的中文数据加密方案,面对云存储中的海量中文数据,它不仅能够保证数据的安全性,还能够在耗费较少时间和存储空间的情况下准确找到用户的数据。

已有的基于密文检索的中文数据加密方案分为两大类。1)基于对称密码学算法的可搜索加密(Searchable Encryption, SE)[4-9],该类方案中加密和解密的密钥都衍生于同一个密钥,它们或者相等或者需要一些简单的转换。其优点是计算开销小;缺点是加密方和解密方都需要事先协商好密钥,且密钥很容易在传输和存储的过程中被泄漏。2)基于公钥密码学算法的SE[10-12],其能适用于一些不安全的网络,但是计算开销较大。

由已有的方案可知,密文检索技术主要分为两种[13]:1)直接对密文进行线性检索,这种检索方法是将关键字与密文进行逐条比对,在每次比对的过程中都需要对密文进行加密和解密,面对存有海量文件的云存储时实现效率很低;2)基于索引的方法,这种方法在对数据文件进行加密的同时还对文件建立索引,以便后期通过索引进行检索比对。后者实现的检索效率很高,但是建立索引会占用大量的存储空间,尤其是使用布隆过滤器的加密方案时更甚。

随着密文检索技术的不断发展,学者们已提出了基于模糊数据检索[14-15]、基于区间关键字检索[16]以及能够返回 k 个最优结果[17]的加密方案,但这些方案的检索效率较低,在云

到稿日期:2017-03-29 返修日期:2017-06-18 本文受国家自然科学基金项目(61672106),北京市教委科技发展项目(KM201611232013)资助。
张蜀男(1991-),男,硕士生,主要研究方向为无线网络安全等,E-mail:291469736@qq.com;蔡 英(1966-),女,博士,教授,主要研究方向为网络安全、无线网络和密码算法等,E-mail:ycai@bistu.edu.cn(通信作者);范艳芳(1979-),女,博士,讲师,主要研究方向为安全模型与访问控制;夏红科(1979-),女,博士,讲师,主要研究方向为机器学习、社会网络、知识图谱。

存储中实现较为困难。

本文提出了基于对称加密算法的优化方案，该方案包含了基于索引的密文检索技术，由数据加解密和数据检索两个过程组成。

在数据加解密过程中，所提方案会将明文数据文件划分为多个数据分块，并进行一些处理后产生密文文件和对应的索引向量文件，然后使用 DES 加密算法对索引向量文件进行加密。密文的解密为加密的逆过程。在数据检索的过程中，本文使用字符线性匹配的方法，根据不同类型的关键字，借助索引向量文件进行密文检索。所提方案的优点是在密文检索过程中无需上传密钥，消除了密钥泄漏的风险，且关键字的密文检索过程与在明文字符串上进行线性匹配类似，极大程度地减少了关键字的匹配时间。

2　相关工作

云存储中的可搜索加密方案主要分为两类：基于对称密码学算法的 SE 和基于公钥密码学算法的 SE。

2.1　基于对称密码学算法的 SE

Song 等人[6]首先设计了一种检索方案 SWP 来解决密文检索的问题，但是该方案需要在密文检索之前上传密钥，增加了密钥在传输过程中的泄漏风险；而且，SWP 方案在密文检索时需要进行多次解密操作，导致计算开销随着文件大小的增加呈线性增长趋势。Goh 等人[8]提出了一种基于布隆过滤器和伪随机函数的 Z-IDX 算法，该算法会使用额外的存储空间来存储索引文件，当源文件很小时，对应的索引文件可能会达到源文件大小的数倍。Curtmola[9]基于树结构设计了一种高效的 SSE-1 算法，但是该算法需要构建相关关键字的列表，即每个文件的上传或删除都涉及列表的更新，所以在处理动态更新的数据时非常耗时。

2.2　基于公钥密码学算法的 SE

Boneh 等人[10]第一次将 SE 应用于公钥密码学算法，提出了能够进行关键字搜索的公钥加密概念，并且运用了基于双线性对的算法，但是该算法不符合一致性，每次加密和解密都需要进行一次双线性对的计算，效率很低。Boldyreva 等人[11]利用有损陷门函数（Lossy Trapdoor Function，LTF）给出了标准模型下可证安全的确定性加密方案，但是该方案的安全模型较弱。Crescenzo 等人[12]设计了一种基于二次剩余问题的可检索公钥加密算法 PEKS，但是其检索效率仍然较低。

3　方案设计

本节将详细介绍所提方案的各组成模块。方案中每个模块之间的联系非常紧密，尤其是数据检索技术需依赖于数据加密技术。

所提方案由文件预处理过程和 3 个模块组成，3 个模块分别是明文加密模块、密文检索模块和密文解密模块。在明文加密模块中，用户使用该方案在客户端加密文件，加密密钥由客户保存，然后将加密后的文件上传到云存储服务器。在密文检索模块中，用户使用 RSA 加密算法加密关键字，然后将其上传到云存储服务器，由云存储服务器根据方案对密文进行检索并返回符合条件的密文文件序列。在密文解密模块中，用户在客户端对返回的密文文件序列进行解密并最终得到源文件。本方案的结构如图 1 所示。

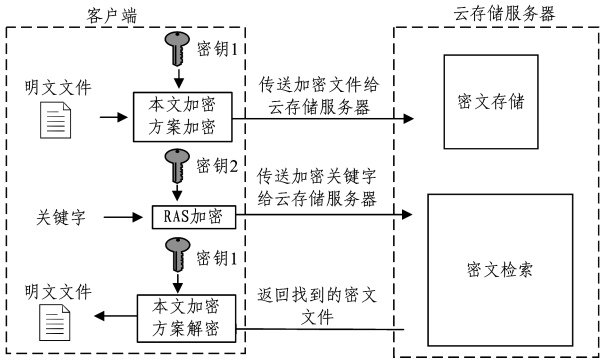

图 1　中文数据加密方案的整体设计

Fig. 1　Overall design of Chinese data encryption scheme

3.1　文件预处理

在进行中文数据文件预处理时，首先将整个文件按照语义顺序以一个汉字或一个英文字母（允许文件中包含少量的英文）为单位进行排列，每 m 个单位划分为一个数据块，将每个数据块记为 $B_i(b_1,b_2,b_3,\cdots,b_m)$，其中 $i\in\{0,1,\cdots,c\}$，c 为生成的数据块块号，b_* 代表一个单位。分组完毕后，为每一个分组中的单位顺序分配一个标号，每个标号代表对应单位在数据块中的位置，记为标号向量 $L_i(1,2,3,\cdots,m)$，使得每一个数据块 B_i 和 L_i 都构成一个线性映射关系 $G_i\langle B_i,L_i\rangle$。

3.2　明文加密模块

数据文件的加密过程分为两个阶段：数据块随机排序阶段和数据块标号加密阶段。完整的数据加密过程如图 2 所示。本文定义明文文件为 F，密文数据块文件为 D，索引向量文件为 L。

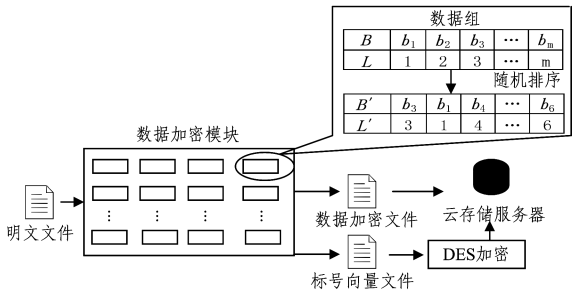

图 2　数据加密流程

Fig. 2　Process of data encryption

3.2.1　数据块随机排序

首先，根据标号随机排序算法（见算法 1）对 F 中每个数据块对应的标号向量 L_i 进行随机排序，排序后的标号记为 L_i'。随后将数据块 B_i 与 L_i' 内的乱序标号进行逐一对应，得到 $G_i'\langle B_i',L_i'\rangle$。

算法 1　标号随机排序算法

```
input：标号向量数组 L(1,2,3,…,m)
output：标号向量数组 L(3,1,4,…,m)
1. Initialize data=L[0] and point=0;
2. for all rand()∈{0,1,2,…,m} do
3.    while point <=m then
4.       data=L[rand()];
5.       L[rand()]=L[point];
6.       L[point]=data;
7.       point++;
8.    end while
```

9. end for

10. return L;

数据块随机排序前后的对比图如图 3 所示。

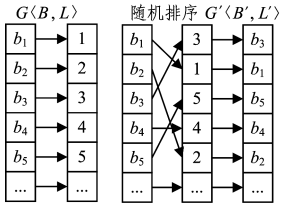

图 3 数据块随机排序前后的对比

Fig. 3 Comparison before and after random sorting of data block

待所有数据块 B_i' 完成上述随机排序后，再将所有数据块进行块间随机排序，并按行将结果存入 D 中。

3.2.2 数据块标号加密阶段

标号向量的加密过程如图 4 所示。考虑到用户存入云端的数据量非常巨大，存储的标号向量在存储体积上必须尽可能地小。本文采用将 L_i' 转化为 m 位整数的方法，这样可以大幅减小标号向量所占用的存储空间。

将 L_i' 转化为整数 d_i' 的具体方法如式(1)所示：

$$d_i' = \sum_{i=1}^{n} 10^{len-i} * L_i', i = 1,2,3,\cdots,n \tag{1}$$

其中，len 为 L_i' 的长度。

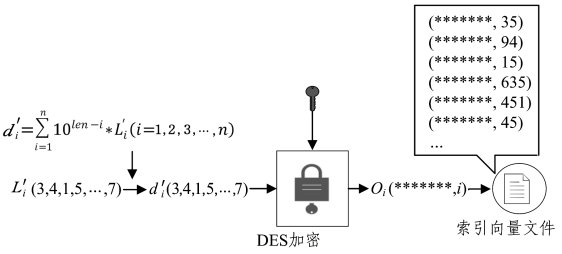

图 4 标号向量的加密过程

Fig. 4 Encryption process of label vector

采用 DES 对称加密算法对 d_i' 进行加密，并将加密后的 $d'_{i\text{-encrypt}}$ 存储于索引向量 $O_i(d'_{i\text{-encrypt}}, i)$，其中 i 代表 L_i' 对应的数据块 B_i' 在明文文件 F 中的块号。当全部索引向量 O_i 处理完毕后，将所有的 O_i 按照 D 中数据块存入的顺序存入 L 中。注意，D 与 L 之间存在一个映射关系，即 D 中的第 i 行与 L 中的第 i 行构成一个 G_i'。

用户在客户端对数据文件完成加密过程后，将生成的 D 和 L 上传到云存储服务器，而用于对文件加密的密钥则由用户保管。

3.3 密文检索模块

所提方案对用户需要检索的关键字有一个要求，即检索关键字必须与原文中存在的某词或某句在格式和顺序上保持一致。

由于用户将要检索的关键字 keyword 可能是一个词组，也可能是一条完整的语句，还有可能是更多语句的组合，用户将关键字上传到云存储服务器的过程中可能会泄漏 keyword 的信息，因此需要对 keyword 进行加密。对 keyword 进行加密时，可以选用传统的 RSA 公钥加密方式。

服务器在收到 keyword 后，首先对其进行解密，然后根据解密后的 keyword 进行密文检索。密文检索算法如算法 2 所示。首先，将 keyword 定义为 $K(k_1, k_2, k_3, \cdots, k_v)$，其中 v 为关键字的长度；然后，将 K 中的第一个单位 k_1 定位到数据块文件 D 中进行逐行检索。这里会出现两种可能的检索情况，只有一种情况会用到索引向量文件 L。不需要使用 L 的情况是 keyword 的长度小于一个数据块长度 m，且其在明文文件中属于同一个数据块。需要用到 L 的情况还可细分为两种：1)keyword 的长度小于 m，并且其在明文文件中属于相邻的两个数据块；2)keyword 的长度大于 m。

算法 2 密文检索算法

Input：关键字 keyword

Output：数据分块文件 D 和索引向量文件 L

1. keyword→K(k_1, k_2, k_3, …, k_v)，其中 v∈Z^+；
2. Initial k_{in}=0，$Data_{in}$=0，g=keyword. length/m+1，其中 m 为数据分块长度；
3. for all 数据分块行 Data_line∈数据分块文件 D do
4. while K[k_{in}]∈Data_line[$Data_{in}$] then
5. k_{in}++；
6. end while；
7. if k_{in}!=0 then
8. 在索引向量文件 L 中找到第 $Data_{in}$ 行，取出原始数据行号 i 的值；
9. 在 L 中找到 i 的取值为〈i,i+g〉的向量行，并将其行号对应到 D 中；
10. 在 D 中对应行进行 K 中剩余关键字的查找；
11. if k_{in}==keyword. length then
12. return 找到的数据分块文件 D 和索引向量文件 L；
13. else break；
14. end if；
15. end if；
16. end for；

由于使用索引向量文件进行检索的过程比较复杂，并且不需要使用 L 的情况也包含在第二种情况中，因此本文以 keyword 的长度大于 m 的情况为例进行分析。

在分析之前，需要对关键字检索的范围进行说明。如果在文件预处理时将每 m 个单位划分为一个数据块，且查询关键字的长度大于 m，则在一个数据块 B_i' 中成功匹配到查询关键字中的第一个单位后，剩余单位需要检索数据块的范围为 $i \sim i+g$，其中 g 为 $keyword.length/m+1$。如果剩余单位中的任何一个在该范围内未被检索到，则匹配失败，继续对关键字的第一个单位进行逐行匹配。

假设 $m < keyword.length < 2m$，在数据块文件 D 中的第 h 行检索到了 k_1，但是 k_2 不在该行中，这时需要使用索引向量文件 L。根据行号 h 找到 L 中的第 h 行，即为相对应的索引向量。在索引向量中提取数据块在明文文件 F 中的块号 s，并在 L 中找到块号为 $s+1$ 的索引向量。该索引向量的行号为 y，继续对应到数据块文件 D 中，找到第 y 行并检索剩余的单位。如果有一个单位匹配失败，则继续对关键字的第一个单位在 h 行后的数据行进行匹配。完整的检索过程如图 5 所示。

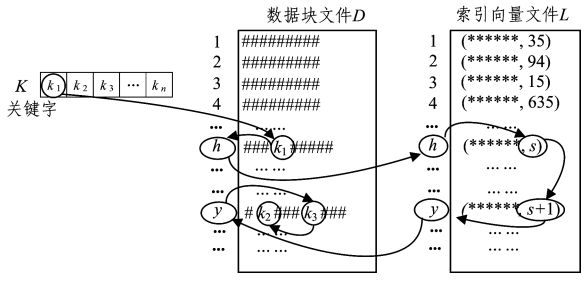

图 5 密文检索流程

Fig. 5 Process of ciphertext retrieval

3.4　密文解密模块

密文的解密过程是加密过程的逆过程，解密工作在客户端进行。密文解密算法如算法 3 所示。用户收到密文文件 D 和索引向量文件 L 后，排序函数 $sorting_1(B_i', i, L)$ 能够将乱序数据块 B_i' 按行号 i 顺序还原，这里可以借助中转文件来实现。当 $sorting_1$ 完成以后，解密函数 $decrypt(d'_{i\text{-encrypt}}, key, \text{DES})$ 使用 DES 算法的解密部分对所有向量行标号 $d'_{i\text{-encrypt}}$ 进行解密，最终得到明文标号向量整数 d_i'。$formula(1)(d_i')$ 负责将 d_i' 还原为标号向量 L_i。排序函数 $sorting_2(Data\_line_*, D)$ 根据 L_i 的顺序对 $Data\_line_*$ 排序并得到明文文件 F，最后删除索引向量文件 L。

算法 3　密文解密算法

Input：数据分组文件 D，索引向量文件 L，密钥 key

Output：明文数据文件 F

1. L←sorting$_1$(B_i'，i，L)；
2. d_i'←decrypt($d'_{i\text{-encrypt}}$，key，DES)；
3. L_i←fomula(1)(d_i')；
4. F←sorting$_2$($Data\_line_*$，D)$\cup L_i$；
5. delete L；
6. return F；

4　方案分析

本节对加密方案的安全性、计算开销以及存储效率进行了分析。所提方案只适用于中文数据的加密，对于具有大篇幅英文或其他种类文字的文本并不适用。

4.1　安全性分析

1）密钥安全性。攻击者获取密钥的形式有 3 种：①侵入用户客户端进行窃取；②在密钥传输过程中进行窃取；③侵入云存储服务器进行窃取。所提方案并没有传输密钥给云存储服务器，密钥全程保存在客户端中，这就使得密钥被窃取的风险降低了 2/3。相对于其他可检索加密算法需要传输密钥这一特点，所提方案对密钥的保护程度较高。

2）密文文件安全性。对于索引向量文件，其内容的形式为 $(d'_{i\text{-encrypt}}, i)$，而 $d'_{i\text{-encrypt}}$ 是使用 DES 对称加密算法加密后的密文，在没有密钥的情况下只能进行暴力破解，暴力破解的时间 t 随着密钥长度的增长而变长，当 $key.length>100$ 时，破解时间可以达到几十亿年。

对于数据块文件，它是由明文文件经过分块处理并根据随机排序函数对其进行块内与块间的随机排序而生成的，数据块文件生成后即可销毁明文文件。随机排序函数的不可逆性，使得数据块文件无法单独通过随机排序函数逆向还原为明文文件。由于明文文件是建立在语义基础之上的，因此数据块文件中的乱序明文已经没有了语义。假设攻击者仅获取到密文文件，攻击者对于单个数据块重新排列成功的概率为 $1/m!$，而对所有分组全部排列成功的概率为 $(1/m!)^c$。对于 m，存在一个小整数 u，当 $m>u$ 时，所提方案即可保证密文文件的安全性。从攻击者的角度而言，一个分组 B_i 的形式为 $(b_3, b_2, b_7, \cdots, b_6)$，正确的明文结果为 $B_i(b_1, b_2, b_3, \cdots, b_m)$，其中 b_1 与 b_2 的组合在语义上可能是词组 $word_1$，但是 b_1 与 b_4 的组合在语义上可能是词组 $word_2$，以此类推，每个字节都可能与其他字节组成一个新的词组 $word_*$，这解密出来的明文结果可能为 $B_i(b_4, b_6, b_3, \cdots, b_9)$，但是这样可能与正确的明文语义出现偏差，因此攻击者暴力破解存在不确定性。

因此，攻击者只要不同时获得密钥、数据块文件和索引向量文件，就无法准确解析出明文文件。

4.2　计算开销分析

本节从现有密文检索方案中选出两个具有代表性的方案（SWP 方案和 Z-IDX 方案）与本文方案进行计算开销的对比分析，分析结果如表 1 所列。

表 1　各方案计算复杂度的对比

Table 1　Comparison of computation overhead of schemes

	数据加密开销	密文检索开销	数据解密开销
所提方案	$(Q\cdot \text{DES})$	$O(Q/!+nkv)$	$O(Q/m\cdot\log_2(Q/m)+\text{DES})$
SWP 方案	$O(Q\cdot(S+F+\text{XOR}+\text{DES}))$	$O(Q\cdot \text{XOR})$	$O(Q\cdot F+\text{XOR}+\text{DES})$
Z-IDX 方案	$O(Q\cdot(F_1+F_2+\text{DES}))$	$O(Q\cdot(H+\text{find}))$	$O(Q\cdot \text{DES})$

由于每种方案对数据文件的处理方式不同，因此通过上传文件的总长度 Q 来进行变量的统一。假设用户向云存储服务器上传了 n 个文件，在本方案中进行文件预处理时数据块的单位长度为 m，每个数据块文件的总行数为 l_q，其中 $q\in\{0,1,\cdots,n\}$，文件总长度为 $Q=(l_1+l_2+\cdots+l_n)m$。在其他两种方案中，对文件处理后生成关键字的个数为 w_q，其中 $q\in\{0,1,\cdots,n\}$，关键字的平均长度为 $\lfloor m/3\rfloor$，文件总长度为 $Q=(w_1+w_2+\cdots+w_n)\lfloor m/3\rfloor$。下面详细分析 3 种方案的计算开销。

1）所提方案的计算开销分析。文件加密过程的计算开销为：$(2m+1+encrypt_{\text{DES}})\cdot(l_1+l_2+\cdots+l_n)$，其中 $encrypt_{\text{DES}}$ 为对内容的加密开销；关键字处理的计算开销为：$RSA_{\text{enc}}\cdot(keyword)$，其中 RSA_{enc} 为公钥加密算法；文件检索过程的计算开销为：$RSA_{\text{dec}}\cdot(keyword)+2(l_1+l_2+\cdots+l_n)+n(m-1)\,k_v/m$，其中 RSA_{dec} 为公钥解密算法；文件解密过程的计算开销为：$2\sum_{i=1}^{n} l_i\log_2 l_i+(l_1+l_2+\cdots+l_n)\cdot decrypt_{\text{DES}}$，其中 $decrypt_{\text{DES}}$ 为对内容的解密开销。

2）SWP 方案的计算开销。文件加密过程的计算开销为：$(S+encrypt_{\text{DES}}+F(f,S)+\text{XOR}(E,Str))\cdot(w_1+w_2+\cdots+w_n)\lfloor m/3\rfloor$，其中 F 和 f 代表伪随机函数，S 代表根据文件长度生成的伪随机流，$\text{XOR}(E,Str)$ 代表对密文和基于密文单词左半部分生成的二进制字符串进行异或。在密文检索阶段，生成陷门的计算开销为：$T_w(keyword)$，其中 T_w 为陷门产生的函数，根据关键字检索的计算开销为：$\text{XOR}(E,T_w)\cdot(w_1+w_2+\cdots+w_n)\lfloor m/3\rfloor$。文件解密过程的计算开销为：$(decrypt_{\text{DES}}+F(f,S)+\text{XOR}(E,Str))\cdot(w_1+w_2+\cdots+w_n)\lfloor m/3\rfloor$。

3）Z-IDX 方案的计算开销。文件的加密阶段包括用布隆过滤器制作文件索引的过程和对数据文件本身加密的过程，该过程的计算开销为：$(F_1(w_i)+F_2(x_1,x_2,\cdots,x_r)+encrypt_{\text{DES}})\cdot(w_1+w_2+\cdots+w_n)\lfloor m/3\rfloor$，其中 F_1 为对关键字 w_i 进行散列的伪随机函数，F_2 为对经由 F_1 作用后得到的码

字 $x_1,x_2,\cdots,x_r$ 进行再散列的伪随机函数，r 为码字的长度。在密文检索阶段，生成陷门的计算开销为：$T_w(keyword)$，对关键字进行匹配检索的计算开销为：$(H(T_w)+find(y_1,y_2,\cdots,y_r))\cdot(w_1+w_2+\cdots+w_n)\lfloor m/3\rfloor$，其中 H 为基于陷门生成码字的函数，$find$ 为进行码字搜索匹配的函数。文件解密过程的计算开销为：$decrypt_{\mathrm{DES}}\cdot(w_1+w_2+\cdots+w_n)\lfloor m/3\rfloor$。

上述对比分析表明，在数据文件加密的过程中，所提方案的加密方法的计算功耗最小，而 SWP 方案和 Z-IDX 方案都需要进行额外的索引计算操作以及预处理操作。密文检索的过程中，具体的开销差异体现在不同的处理函数上；而在文件解密阶段，则是 Z-IDX 方案最优。

4.3 存储效率分析

在存储效率方面，仍将所提方案与 SWP 方案和 Z-IDX 方案进行对比分析。以单一数据文件为例，对比结果如表 2 所列。

表 2 各方案存储开销的对比

Table 2 Comparison of storage overhead of schemes

	所提方案	SWP 方案	Z-IDX 方案
加密文件	大小与明文文件基本相同	平均大小约为明文文件的 3～4 倍	平均大小约为明文文件的 2～3 倍
文件索引	平均大小约为明文文件的 2 倍	/	平均大小为明文文件的数倍甚至十几倍

1)所提方案的存储效率。所提方案在云存储服务器中分别存储了密文文件和索引向量文件，增加了服务器的存储开销。根据实验验证，一个数据文件对应的索引向量文件大约为数据文件本身的两倍，相当于在云存储服务器中存储了 3 个数据明文文件。

2)SWP 方案的存储效率。该方案直接对明文文件进行关键字划分并加密。实验证实，每个关键字所形成的密文大小是明文关键字的 3～4 倍，这就导致了密文文件整体大小是明文文件的 3～4 倍。

3)Z-IDX 方案的存储效率。该方案使用布隆过滤器来进行明文文件的索引处理，当数据文件较小时，通过布隆过滤器产生的索引长度是明文文件的数倍，存储效率很低。

从上述分析中可以看出，所提方案的存储效率比其他两种方案更优，虽然其在一定程度上增加了服务器的存储开销，但是综合考虑，尤其是在存储效率和计算开销不能兼得的情况下，只能尽量平衡二者，使方案的性能达到最优。同时，云存储的存储空间极大，可以在一定程度上抵消所提方案的额外存储开销。

5 实验

为了验证所提方案的计算开销和存储效率，本文将以 900 篇平均大小为 300k 的科技论文作为测试数据，以 8GB 内存和具有 4 核 CPU 的 PC 机作为测试的硬件设备，以 $m=9$ 作为数据分块的大小。实验的软件测试平台利用 Hadoop 框架，通过其中的 HDFS(分布式文件系统)[18]和 MapReduce[19]对密文检索进行并行计算，这样可以显著减少数据检索的时间。实验中每个测试数据均为实验运行 10 次所取的平均值，测试结果如下。

5.1 数据加密文件所需

图 6 给出了所提方案、SWP 方案和 Z-IDX 方案在数据文件数量增多的情况下加密文件所需的时间。从仿真结果可以看出，所提方案的数据文件加密时间最短，Z-IDX 方案的加密时间略多于 SWP 方案，且这两个方案的加密时间都约为所提方案的两倍。

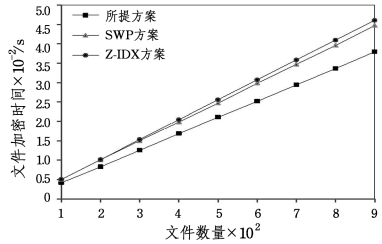

图 6 文件加密时间

Fig. 6 Time overhead of data encryption

5.2 密文检索

图 7 给出了 3 种方案在数据文件数量增长且关键字长度为 5 的情况下检索密文所需的时间。所提方案对数据块进行逐行检索，每检索一个数据行只需匹配关键字的第一个单位，只有匹配成功时才继续进行关键字中其他单位的匹配。SWP 方案每次都需要使用异或的方法来进行关键字的匹配，且密文没有明确的划分，匹配时需要按密文字符推进，检索效率很低。Z-IDX 方案需要对陷门进行解析，通过哈希函数把关键字的每个字符散列成相对应的码字并与索引进行位匹配，在检索过程中只与索引有关，与密文无关，因此检索效率很高。从仿真结果可以看出，所提方案的检索时间略多于 Z-IDX 方案，但远少于 SWP 方案。

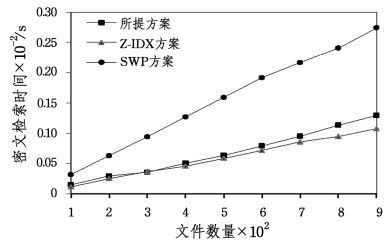

图 7 文件检索时间

Fig. 7 Time overhead of ciphertext retrieval

5.3 文件解密

图 8 给出了 3 种方案在数据文件数量增多的情况下解密文件所需的时间。

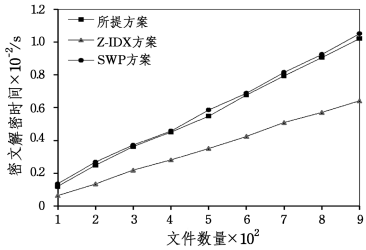

图 8 文件解密时间

Fig. 8 Time overhead of ciphertext decryption

从仿真结果可以看出，Z-IDX 方案的文件解密时间最短，所提方案的文件解密时间少于 SWP 方案的解密时间。

5.4　存储开销

图 9 给出了 3 种方案在数据文件数量增多的情况下存储文件的开销。实验验证发现，所提方案索引向量文件的大小平均为明文数据文件大小的两倍。SWP 方案在进行密文存储时只存储一个密文文件，通过这种方式进行加密的密文文件的大小是明文文件大小的 3～4 倍。Z-IDX 方案首先利用布隆过滤器进行关键字的散列，形成索引之后还需要对明文文件进行传统形式的加密，存储开销是明文文件大小的数倍，当明文文件较小时，可以达到十几倍，存储效率非常低。从仿真结果可以看出，所提方案的存储开销是 3 种方案中最低的，SWP 方案的存储开销比所提方案的略高，而 Z-IDX 方案的存储开销远高于其他两种方案。

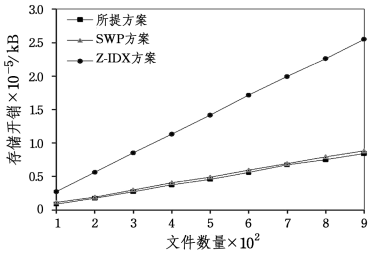

图 9　存储开销

Fig. 9　Overhead of storage

结束语　本文提出了一种适合云存储环境的高效密文检索的中文数据加密方案，通过对明文数据文件进行数据块的划分和一系列处理来产生密文文件和相应的索引向量文件，并将它们上传到云存储服务器。加解密密钥永远存储在客户端，因此数据安全性能够得到保证。对于密文检索，本文直接对密文进行关键字的首字符匹配，且在匹配过程中不需要对密文进行解密操作，提升了密文检索的效率。与此同时，所提方案还拥有很低的存储开销，这在实验中已经得到了验证。但是所提方案还不够完善，因为目前只能实现关键字的精确匹配，不能实现关键字的模糊查找、范围查找以及返回前 k 个最优文件的功能，这些问题将在所提方案的基础上进行进一步的研究。

参 考 文 献

[1] KAMALRAJ D, BALAMURUGAN B, JEGADEESWARI S, et al. Shamir's key based confidentiality on cloud data storage[C]// International Conference on Advances in Computing, Communications and Informatics. 2015: 418-423.

[2] YEH T, LEE H. Enhancing Availability and Reliability of Cloud Data through Syncopy[C]// IEEE International Conference on Internet of Things. 2014: 125-131.

[3] ZHANG Q, LI S, LI Z, et al. CHARM: A Cost-efficient Multi-cloud Data Hosting Scheme with High Availability[J]. IEEE Transactions on Cloud Computing, 2015, 3(3): 372-386.

[4] AWAD A, MATTHEWS A, QIAO Y, et al. Chaotic Searchable Encryption for Mobile Cloud Storage[J]. IEEE Transactions on Cloud Computing, 2015, PP(99): 1.

[5] CUI B, LIU Z, WANG L. Key-Aggregate Searchable Encryption (KASE) for Group Data Sharing via Cloud Storage[J]. IEEE Transactions on Computers, 2015, 65(8): 1.

[6] SONG D X, WAGNER D, PERRIG A. Practical Techniques for Searches on Encrypted Data[C]// IEEE Symposium on Security and Privacy. IEEE Computer Society, 2000: 44-55.

[7] DENG Z, LI K, LI K, et al. A multi-user searchable encryption scheme with keyword authorization in a cloud storage[J]. Future Generation Computer Systems, 2017, 72: 208-218.

[8] GOH E J. Secure Indexes[J]. IACR Cryptology ePrint Archive, 2003, 2003: 216.

[9] 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.

[10] DAN B, CRESCENZO G D, OSTROVSKY R, et al. Public Key Encryption with Keyword Search[M]// Advances in Cryptology-EUROCRYPT 2004. Springer Berlin Heidelberg, 2003: 506-522.

[11] BOLDYREVA A, FEHR S, O'NEILL A. On Notions of Security for Deterministic Encryption, and Efficient Constructions without Random Oracles[M]// Advanced in Cryptology—CRYPTO 2008. Springer Berlin Heidelberg, 2008: 335-359.

[12] CRESCENZO G D, SARASWAT V. Public Key Encryption with Searchable Keywords Based on Jacobi Symbols[J]. International Transactions on Computers, 2016, 65(8): 2374-2385.

[13] XU L, XU C. Efficient and Secure Data Retrieval Scheme Using Searchable Encryption in Cloud Storage[C]// International Symposium on Security and Privacy in Social Networks and Big Data. IEEE Computer Society, 2015: 15-21.

[14] GAO G, LI R, GU X, et al. Mimir: Term-distributed indexing and search for secret documents[C]// International Conference on Collaborative Computing: Networking, Applications and Worksharing. 2010: 1-9.

[15] KHAN N S, KRISHNA C R, KHURANA A. Secure ranked fuzzy multi-keyword search over outsourced encrypted cloud data[C]// International Conference on Computer and Communication Technology. IEEE, 2014: 241-249.

[16] WANG Z, GONG K, JIN S, et al. An efficient interval query algorithm based on inverted list in cloud environment[C]// 2012 International Conference on Information and Automation (ICIA). IEEE, 2012: 221-225.

[17] YU J, LU P, ZHU Y, et al. Toward Secure Multikeyword Top-k Retrieval over Encrypted Cloud Data[J]. IEEE Transactions on Dependable & Secure Computing, 2013, 10(4): 239-250.

[18] ISLAM N S, RAHMAN M W, JOSE J, et al. High performance RDMA-based design of HDFS over InfiniBand[C]// Proceedings of the International Conference on High Performance Computing, Storage and Analysis. IEEE Computer Society Press, 2012: 35.

[19] PALANISAMY B, SINGH A, LIU L. Cost-Effective Resource Provisioning for MapReduce in a Cloud[J]. IEEE Transactions on Parallel & Distributed Systems, 2015, 26(5): 1265-1279.