加密云数据下基于 Simhash 的模糊排序搜索方案

杨 旸[1),3)] 杨书略[2),3)] 柯 闽[1),3)]

[1)](福州大学数学与计算机科学学院 福州 350108)

[2)](福州大学物理与信息工程学院 福州 350108)

[3)](网络系统信息安全福建省高校重点实验室 福州 350108)

摘 要 为了保护数据隐私,数据拥有者会将敏感数据的密文外包到云服务器,这使得传统明文搜索技术难以使用.因此可搜索加密技术被用于对密文数据进行搜索,实现高效的数据利用.然而目前在加密云数据中,关键词模糊搜索方案主要是通过构造关键词模糊集合来实现,其需要大量的计算和存储开销.本文提出的搜索方案,无需构造关键词模糊集合,而是基于 Simhash 的降维思想,将文档关键词做 n-gram 处理并得到 Simhash 指纹来实现模糊搜索.该文结合汉明距离和关键词相关度分数,设计了双因子排序算法对查询结果进行排序.使用树索引结构和新型遍历方法进一步提高了搜索效率.通过新型遍历方法,即使树的节点值与期望值不相等,也能够对树进行遍历.理论分析和实验结果表明:该方案实现了加密云数据下的关键词模糊搜索,同时极大地节约了时间和空间成本.

关键词 云计算;加密云数据;隐私保护;可搜索加密;模糊排序搜索;Simhash

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2017.00431

Ranked Fuzzy Keyword Search Based on Simhash over Encrypted Cloud Data

YANG Yang[1),3)] YANG Shu-Lue[2),3)] KE Min[1),3)]

[1)](*College of Mathematics and Computer Science, Fuzhou University, Fuzhou* 350108)

[2)](*College of Physics and Information Engineering, Fuzhou University, Fuzhou* 350108)

[3)](*Key Lab of Information Security of Network System in Fujian Province, Fuzhou University, Fuzhou* 350108)

Abstract With the development of cloud computing, data owners are motivated to outsource their data and the corresponding complex management tasks to the public cloud for convenience and economic savings. In order to protect data privacy, data owners prefer to outsource their sensitive data in an encrypted form to the cloud, which makes the traditional search techniques useless. Searchable encryption is a technique to search on encrypted data without decryption to realize efficient data utilization. There have been some studies on secure searching over encrypted cloud data, which pay attention to both privacy and practicability of data. However, most of them are based on accurate keyword matching. The fuzzy keyword search problem remains unsolved. Up to date, the existing construction of fuzzy keyword search schemes has to build fuzzy keyword set. It will lead to tremendous computation and storage overheads. In this paper, we propose a new scheme without constructing fuzzy keyword set. Based on the idea of dimension reduction of Simhash, each keyword is transformed to a Simhash fingerprint by n-gram method to achieve fuzzy matching. Combining the hamming distance and keyword relevance score, we design a double factor ranking algorithm to sort the results accurately. In addition, tree structure and a novel traversal method are utilized to further improve the efficiency of our proposed scheme. The

收稿日期:2016-03-20;在线出版日期:2016-09-18.本课题得到国家自然科学基金(61402112,61472307,61472309,61303198)、福建省教育厅科技项目(JA12028)、福建省重大科技项目(2015H6013)、福州大学科技发展基金项目(2012-XY-17)资助.**杨 旸**,女,1984 年生,博士,副教授,主要研究方向为信息安全和隐私.E-mail:yang.yang.research@gmail.com.**杨书略**(通信作者),男,1990 年生,硕士研究生,主要研究方向为可搜索加密.E-mail:yangshulue@foxmail.com.**柯 闽**,女,1992 年生,硕士研究生,主要研究方向为可搜索加密.

tree can be traversed even if the value of the tree node is not equal to the expected value by the proposed traversal method. Theoretical analysis and experimental results show that the scheme realizes the ranked fuzzy keyword search over encrypted cloud data. Meanwhile, the computation and storage overheads are greatly reduced.

Keywords cloud computing; encrypted cloud data; privacy-preserving; searchable encryption; ranked fuzzy keyword search; Simhash

1 引 言

云计算的发展使越来越多的用户将自己的数据外包给云服务器,享受方便快捷的服务[1].但云服务器不是完全可信的,可能会对用户的数据隐私产生很大的威胁.为了保障敏感数据不被泄露,用户会先对数据进行加密,再将数据存储到云服务器.然而数据加密使得高效的数据利用成为挑战,如何在密文数据中检索用户感兴趣的资料成为亟待解决的问题.可搜索加密技术就是为了解决密文检索的难题而提出的.该技术是指用户将加密后的数据存储到云服务器中,云服务器可以根据用户提交的关键词陷门进行搜索,并返回相关文档给用户,而不泄露相关的明文信息.

Song 等人[2]最早提出了可搜索加密方案,为解决密文检索的问题提供了思路,引起了学术界的普遍关注.Goh[3]和 Chang 等人[4]提出为文档创建索引来提高检索效率,每个索引中包含了一篇文档的关键词陷门信息.Curtmola 等人[5]构造了加密的哈希表索引,表中包含关键词陷门和关键词的文档标识集合.Wang 等人[6-7]提出了关键词排序搜索方案,主要通过对相关度分数进行保序加密,实现对搜索结果的精确排序.Cao 等人[8]引入向量空间模型和安全 KNN(secure K-Nearest Neighbor)方法,提出了多关键词排序可搜索加密方案,通过矩阵对索引向量进行加密,并对索引向量和搜索向量计算内积相似度,实现了对搜索结果的排序.Boneh 等人[9]通过引入双线性对的方式,提出了公钥可搜索加密方案,利用公钥可以加密文档并生成索引,私钥拥有者可以生成关键词陷门并进行搜索.

然而以上方案只支持关键词精确搜索,在实际应用中,用户输入的搜索请求经常会出现拼写错误或格式不匹配的情况.Li 等人[10]提出密文关键词模糊搜索,利用通配符的方法构造关键词模糊集合.随后,Li 等人[11]提出了更加节省存储空间的克方法来构造模糊集合,并引入符号索引树提高搜索效率.Liu 等人[12]通过基于字典的模糊集构造方案,虽然减少了模糊集的存储开销,但是却降低了搜索精确度.Wang 等人[13]利用通配符和索引树实现了高效可用的模糊搜索方案.Wang 等人[14]通过提取索引树结构的路径信息,实现了可验证的模糊搜索方案.然而在这些模糊搜索方案中,需要对每个关键词构造模糊集合,这些模糊集合将占用云服务器大量的存储空间.例如,在基于通配符的模糊集构造方法中,随着编辑距离的增加,模糊集合的大小会呈指数增长,因此构造模糊集合会耗费大量的计算和存储开销(详细分析参看 4.3 节).并且这些方案只能通过编辑距离对搜索结果进行排序,排序效果比较粗糙,无法返回精确的搜索结果.虽然在一些方案中[15-16]提到引入布隆过滤器可以有效减少存储空间,但由于针对模糊集合中的每个关键词,都需要用多个哈希函数来将其插入到布隆过滤器中,因此会增加计算开销.

Simhash 算法最早由 Charikar[17]提出,Manku 等人[18]将这个技术运用于海量网页的去重.随后,Simhash 也被运用到可搜索加密领域[19-20],但主要是针对整篇文档生成指纹索引.由于该指纹索引是由文档的多个关键词映射得到的,因此如果只用一个或者少量关键词进行查询会出现较大误差,所以并不适用于关键词搜索.

针对目前的密文关键词模糊搜索方案中,计算和存储开销大,排序结果不精确等问题,本文提出了一种可以减少索引存储空间,并实现精确排序的模糊关键词搜索方案.本文的主要贡献如下:

(1)高效的模糊关键词索引存储.本文改进了 Simhash 算法,能够针对关键词(而不是文档)生成指纹索引,使得该索引结构适用于关键词搜索.通过对关键词做 n-gram 处理,再利用 Simhash 的降维思想,能够将关键词处理成 Simhash 指纹.由于Simhash 指纹本身的特性,授权用户在拼写错误的情况下也可以匹配到正确的关键词,从而实现了密文模糊检索.不同于传统模糊搜索方案,本方案无需构造庞大的关键词模糊集合,而只需要将一个关键词处理为

一个对应的指纹，再构造成索引存储在云服务器即可，因此极大减少了计算和存储开销.

(2) 精确返回排序结果. 为了获得更加精确的排序结果，本文通过结合汉明距离和相关度分数，设计了高效的双因子排序方法，云服务器能够对搜索结果进行精确的排序并返回给搜索用户.

(3) 相关度分数的隐私保护. 引入保序加密方法对相关度分数进行加密，既保护了相关度分数的隐私安全，又能够将排序操作交给云服务器完成，减少了用户查询有效文档的时间开销，节约了带宽资源.

(4) 新型查询方式. 为了进一步提高检索效率，本文结合关键词指纹的特征，构建了指纹索引树，并针对汉明距离的特性设计了新型的指纹索引树遍历方式.

(5) 真实数据集验证方案效率. 理论分析了本方案实现的具体功能及原理，并通过真实数据集上的实验结果，证实了本方案实现了加密云数据下的关键词模糊搜索，并且能够极大地节省计算和存储开销.

2　预备知识

2.1　编辑距离和汉明距离

编辑距离[21]是指将一个字符串转换成另一个字符串需要的最少操作次数，是可以用来定量衡量字符串间相似度的方法. 3 种基本的编辑操作包括：

(1) 插入. 在字符串的任意一个位置插入一个字符.

(2) 删除. 删除字符串中的一个字符.

(3) 替换. 将字符串中的一个字符替换为另一个字符.

汉明距离指的是两个等长字符串之间对应位置的不同字符的个数，即将一个字符串转化成另外一个字符串所需要改变的对应字符个数. 因此，汉明距离可以被用来衡量两个字符串的相似度. 显然，汉明距离大的两个字符串相似度小，汉明距离小的两个字符串相似度大.

2.2　Simhash

Simhash[17]主要思想是将文档特征映射为一个固定长度的指纹，由于相似的文档能够产生相似的指纹，因此文档之间的相似度可以通过指纹之间的汉明距离来测量. 计算某篇文档的 Simhash 指纹的方法如下. 首先，抽取该篇文档的特征关键词. 接着为该篇文档初始化一个 τ 维的向量 $\boldsymbol{S}$，其中 τ 就是 Simhash 指纹的位数. 使用标准哈希算法，如 SHA-1 等，计算每个关键词的哈希值. 如果哈希值的第 i 位为 1，则向量 $\boldsymbol{S}$ 的第 i 位加上此关键词的权重，如果哈希值的第 i 位为 0，则减去此权重. 处理完该篇所有关键词后，如果向量 $\boldsymbol{S}$ 第 i 位的值大于 0，则将第 i 位的值置为 1，反之则置为 0. 最后向量 $\boldsymbol{S}$ 就是该篇文档的指纹.

2.3　相关度分数

在信息检索领域中，为了取得更精确的排序结果，广泛采用 $tf-idf$ 权值计算方法[22]计算关键词相关度分数. $tf-idf$ 主要包括 $tf_{t,f}$ (词项频率) 和 idf_t (逆文档频率) 两项. $tf_{t,f}$ 指的是关键词 t 在某篇文档中出现的频率；idf_t 的定义如下：

$$idf_t = \log \frac{N}{df_t} \tag{1}$$

其中：N 为所有文档的数量；df_t 表示包含关键词 t 的文档数量. 一个低频词的 idf_t 往往比较高，而高频词的 idf_t 较低. 相关度分数计算公式如下：

$$tf-idf = tf_{t,f} \times idf_t \tag{2}$$

2.4　保序加密

保序加密(Order Preserving Encryption，OPE)[23]是一种能够允许数值型数据保持顺序的加密方案. 因此通过对数据进行保序加密，能实现对密文状态下的数据进行数值大小的比较，例如 $a<b$，则 $OPE(ek,a)<OPE(ek,b)$.

3　问题描述

3.1　系统模型

系统模型如图 1 所示，在本文中，一个云环境包含三个实体：数据拥有者、授权用户和云服务器. 数据拥有者想要将文件集 $F=(f_1,f_2,\cdots,f_m)$ 外包给云服务器，为了保障这些数据的安全，在上传之前需要使用标准对称加密算法，将文件集 F 加密为密文形式 $C=(c_1,c_2,\cdots,c_m)$. 为了使授权用户能够实现密文检索，需要对文件集 F 提取关键词集合 $W=(w_1,w_2,\cdots,w_n)$，然后通过基于 Simhash 的关键词指纹生成算法，生成每个关键词 $w_i \in W$ 的关键词指纹 S_i，再通过 S_i 构造出索引 I，最后将密文集合 C 以及索引 I 共同上传至云服务器.

授权用户在得到数据拥有者提供的陷门密钥后，生成查询关键词 w 的指纹，构造出关键词陷门 T_w 并上传到云服务器. 在接收到陷门后，云服务器通过索引 I 搜索到相应的密文集并返回给授权用户. 为了提高搜索的准确性，云服务器需要对搜索结

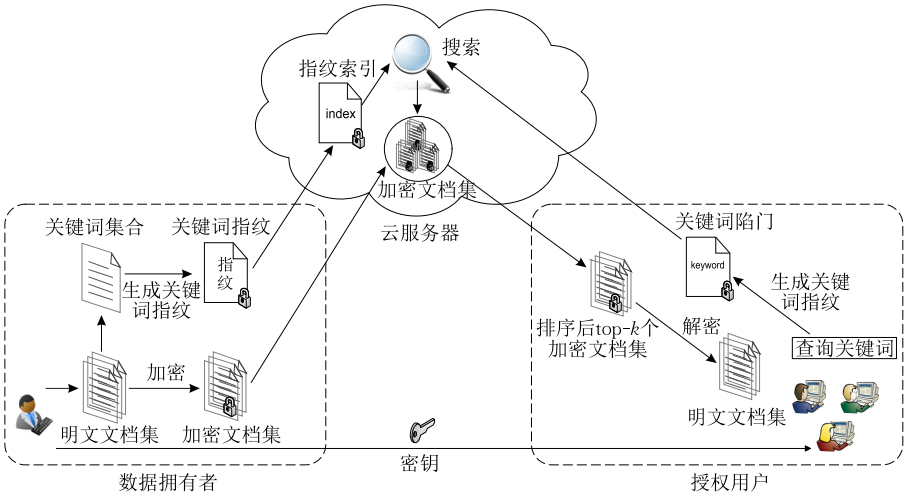

图 1 系统模型

果进行排序. 为了节省带宽开销,授权用户可以将一个可选择的整数 k 连同 T_w 共同上传至云服务器,则云服务器只会返回最相关的 top-k 篇密文文档. 最后,授权用户能够使用密钥解密返回的密文文档.

3.2 威胁模型

在本文中,云服务器被认为是"诚实但是好奇"的. 云服务器会诚实地执行各项存储和搜索操作,并且不会添加或删除相关的安全索引和密文文档,在完成搜索之后会如实返回查询结果. 但云服务器又是好奇的,它希望获得关键词和索引的明文信息,以及其他一些通过统计分析获得的额外信息. 因此需要对文档集合和索引进行加密处理,再将它们上传到云服务器,并且防止云服务器的统计分析. 本文考虑的威胁模型是已知密文模型,在该模型中,云服务器只能获取到密文集和安全索引,以及授权用户上传的陷门.

3.3 主要符号说明

本文使用的主要符号和其说明如表 1 所示.

表 1 主要符号说明

符号	说明
$F=(f_1,f_2,\cdots,f_m)$	明文文档集合
$C=(c_1,c_2,\cdots,c_m)$	F 对应的密文文档集合
$FID=(FID_1,FID_2,\cdots,FID_m)$	文档的标识符集合
$W=(w_1,w_2,\cdots,w_n)$	F 中提取的关键词集合
S_i	关键词 w_i 的指纹
$Score$	关键词的相关度分数
d_{w_i}	S_i 与 T_w 的汉明距离
I	倒排索引
G_W	指纹索引树
$h(hk,\cdot)$	密钥 hk 的单向哈希函数
$\varphi(sk,\cdot)$	密钥 sk 的对称加密算法
$OPE(ek,\cdot)$	密钥 ek 的保序加密函数
T_w	查询关键词陷门

4 基于 Simhash 模糊关键词搜索的基础方案

4.1 基础搜索方案

首先基于 Simhash 算法构造了一个基础的搜索方案,其主要过程如下:

(1) KeyGen(λ). 输入一个安全参数 λ,生成文档加密密钥 sk、单向哈希函数 h 的密钥 hk. 将密钥 sk、hk 发送给授权用户.

(2) BuildIndex(hk,W). 数据拥有者从文档集合 $F=(f_1,f_2,\cdots,f_m)$ 中抽取关键词集合 $W=(w_1,w_2,\cdots,w_n)$,通过基于 Simhash 的关键词指纹生成算法 $sim(hk,w_i)$,生成每个关键词 $w_i\in W$ 对应的指纹 S_i,并为每篇文档创建唯一的文档标识符 $FID_j(1\leq j\leq m)$. 然后对所有关键词创建倒排索引 $I=(I_{w_1},I_{w_2},\cdots,I_{w_n})$,其中 $I_{w_i}=\{S_i,FID_{w_i}\}(1\leq i\leq n)$,其中 FID_{w_i} 表示包含关键词 w_i 的所有文档的标识符集合. 最后将倒排索引 I、密文文档集 $C=(c_1,c_2,\cdots,c_m)$ 上传至云服务器端.

(3) Trapdoor(hk,w). 每一个授权用户都能够获得数据拥有者分发的密钥 hk. 当授权用户需要搜索感兴趣的关键词 w 时,首先通过关键词指纹生成算法 $sim(hk,w)$ 计算关键词 w 的指纹值,该指纹值即陷门 T_w. 然后将产生的陷门 T_w 提交至云存储服务器进行查询.

(4) Search(I,T_w). 云存储服务器在接收到授权用户的搜索请求后,计算关键词陷门 T_w 和索引 $I_{w_i}=\{S_i,FID_{w_i}\}$ 中的指纹 S_i 的汉明距离 d_{w_i}. 根据汉明距离 d_{w_i} 从小到大的顺序,将 top-k 篇密文文档

的集合 $C'=(c_1,c_2,\cdots,c_k)$返回给用户.

(5) Decrypt(C',sk). 授权用户使用密钥 sk,将返回的 top-k 篇密文文档 C'进行解密,获得所需的明文文件集.

4.2　具体实现

4.2.1　基于 Simhash 的关键词指纹生成算法

由于传统的 Simhash 算法是针对每篇文档生成指纹,因此只适用于文档查询文档的场景. 然而在信息检索领域中,利用关键词查询文档的情况则更为普遍. 因此需要对 Simhash 算法进行改进,使之能够针对关键词生成指纹. 为了能够提取每个关键词的多个特征,本方案中引入了 n-gram 方法处理关键词. 另外,传统 Simhash 算法主要运用于明文数据中,因此只使用了普通的哈希函数. 而本方案是运用于加密数据中,为了提高方案的安全性,本文使用了带密钥的单向哈希函数 $h(hk,\cdot)$. 基于 Simhash 生成关键词指纹的具体步骤如下:

(1) 输入需要处理的关键词 w,并将一个 τ 维的向量 $\boldsymbol{V}$ 初始化为 0,一个 τ 维的向量 $\boldsymbol{S}$ 初始化为 0,τ 的值与哈希函数 h 产生哈希值的位数相同.

(2) 将关键词 w 做 n-gram 处理,获得关键词 w 的多个特征. 本方案中,为了得到更精确的搜索结果,需要尽可能细分关键词,因此令 $n=2$. 例如 w 为 encrypt 时,经过 2-gram 处理得到 gramset={en,nc,cr,ry,yp,pt},gramset 中的各个元素就是关键词 w 的特征.

(3) 为了保护关键词和索引的隐私,需要使用带密钥的单向哈希函数 h,对 gramset 中的每个元素计算哈希值. 哈希函数 h 可以选择使用 Hmac-SHA1 或 Hmac-MD5 等,不同的 h 会产生不同位数的哈希值,这会对最后搜索的精确度造成影响,位数越多精确度越高.

(4) 将这些元素的哈希值逐个映射到向量 $\boldsymbol{V}$,如果哈希值的第 i 位为 1,则向量 $\boldsymbol{V}$ 的第 i 位加 1,如果哈希值的第 i 位为 0,则向量 $\boldsymbol{V}$ 的第 i 位减 1.

(5) 最后再将向量 $\boldsymbol{V}$ 映射到向量 $\boldsymbol{S}$,如果向量 $\boldsymbol{V}$ 的第 i 位大于或等于 0,则向量 $\boldsymbol{S}$ 第 i 位的值为 1,如果向量 $\boldsymbol{V}$ 的第 i 位小于 0,则向量 $\boldsymbol{S}$ 第 i 位的值为 0.

(6) 输出 S 作为这个关键词的指纹.

以上的步骤可参考图 2 所示.

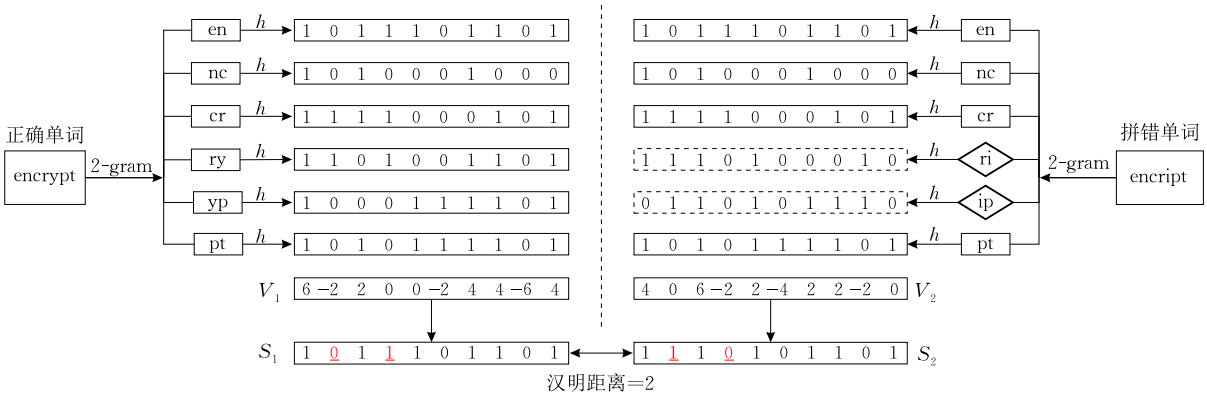

图 2　关键词指纹生成算法流程与搜索原理简例

4.2.2　搜索算法

图 2 展示了基于 Simhash 的关键词指纹生成算法的实现模糊搜索的原理. 假设用户需要检索关键词为 w=encrypt,但是由于用户的粗心或者遗忘等原因,他实际输入的关键词为 w'=encript,即把 y 拼成 i. 在传统的哈希算法中,如果拼写错了关键词的一个字母,那么关键词哈希出的结果会与原来完全不同(图中的哈希值只截取了带密钥的哈希函数 Hmac-SHA1 的前 10 位). 但在关键词指纹生成算法中,由于关键词经过了 n-gram 处理,关键词指纹的值实际上是由 gramset 中的多个元素共同决定的. 当用户出现拼写错误,改变的是 gramset 中 6 个元素的其中两个(即 ry 变为 ri,yp 变为 ip),虽然这两个元素本身的哈希值相比之前会有很大改变(图中虚线框内),但由于指纹是由 6 个元素的哈希值共同映射得到的,因此其它 4 个未改变的元素对指纹的值还是起到主导作用的. 所以相对于其他不相关关键词的指纹,拼错关键词的指纹和正确关键词的指纹之间的汉明距离会更小.

利用这一原理,用户即使在拼写错误的情况下,仍然可以通过比较指纹之间的汉明距离来搜索到正确的关键词. 如图 2,通过计算得到指纹 S_1 和指纹 S_2 的汉明距离为 2,可以判断指纹 S_1 对应的关键词 encrypt 很可能是用户希望搜索的关键词. 在本方案

的搜索阶段，云存储服务器同样会根据这一原理，计算关键词陷门 T_w 和索引 $I_{w_i}=\{S_i, FID_{w_i}\}$ 中的指纹 S_i 的汉明距离 d_{w_i}. 再根据汉明距离 d_{w_i} 从小到大的顺序，将 top-k 篇密文文档的集合 $C'=(c_1, c_2, \cdots, c_k)$ 返回给用户.

4.3 方案优势分析

由上面的分析可以看出，两个关键词指纹之间的汉明距离和它们的 gramset 中相同元素的个数有着非常密切的关系，相同元素多的关键词之间往往汉明距离更小. 正因如此，运用关键词指纹生成算法，用户不仅能够在拼写错误的情况下匹配到正确的关键词，并且不论用户的拼写正确与否，还可以以很小的代价匹配到与输入的关键词相近的词. 而如果在使用通配符或克构造模糊集合的传统方案中实现这些，将要付出非常大的存储和计算开销. 即使将模糊集合插入到布隆过滤器中来减少存储空间，也会以牺牲计算开销为代价.

更具体的情况是，在使用通配符的方法中，假设关键词 $w_i(1\leq i\leq n)$ 的长度为 L，则关键词 w_i 的编辑距离 d 为 1 的模糊集 $S_{w_i,1}$ 大小为 $2\times L+1+1$. 编辑距离为 2 和 3 的模糊集 $S_{w_i,2}$ 和 $S_{w_i,3}$ 大小分别为 $C_{L+1}^1+C_L^1\cdot C_L^1+2C_{L+2}^2$ 和 $C_L^1+C_L^3+2C_L^2+2C_L^2\cdot C_L^1$. 虽然运用通配符的方法能够使得索引文件的拓展尽可能最小化，但 w_i 模糊集存储开销仍为 $O(L^d)$[10]. 而克方法构造的模糊集合 $S'_{w_i,d}$ 的大小为 $C_L^L+C_L^{L-1}+\cdots+C_L^{L-d}$，虽然其存储开销小于通配符方法[11]，但是依然要占用较大的存储空间. 在构造索引和陷门时，以上两种方法的主要计算开销是针对模糊集计算相应的哈希值. 显然，随着关键词的 L 和 d 增大，模糊集会快速增加，存储和计算开销也会随之快速增加. 如果将模糊集合插入布隆过滤器，虽然能够有效降低存储空间，但是需要对模糊集中的每个词项进行 r 次哈希运算（r 为布隆过滤器使用的哈希函数个数），这使得计算量大大增加.

而本方案无需构造模糊集合，云服务器只需要存储每个关键词 w_i 对应的一个指纹值，存储空间不会受到 L 和 d 的影响，每个关键词 w_i 的存储空间即 $O(1)$. 但由于 w_i 在生成指纹的过程中需要针对 gramset 中的元素计算哈希值，因此其主要计算开销为 $L-1$（当 2-gram 时）次哈希运算，这显然优于以上方案的计算开销. 例如在本方案实验中，用户输入 encript 进行检索，可以利用指纹匹配到 encrypt、encrypted 和 encryption 等相似的词，进而找到它们对应的文档集合. 然而，在使用通配符或克的方案中，要达到同样的搜索效果，则需要至少构造编辑距离为 3 的模糊集合，针对这个模糊集合的存储和计算开销都是十分庞大的.

5 改进方案

5.1 改进目的

在基础方案中，基于汉明距离能够实现对搜索结果的排序，但包含同一个关键词的文档可能有很多篇，仅通过汉明距离还无法将包含同一个关键词的文档区分开，因此排序结果较为粗糙. 例如：用户输入 encript 进行查询，通过汉明距离的排序，找到了比较相关的关键词 encrypt，但是包含 encrypt 的文档非常多，仅靠汉明距离还无法对这些文档进行排序. 为了能够使用细粒度的排序方法，需要得到关键词在每篇文档中的权重信息，为此本文引入了关键词的相关度分数计算方法. 接着对相关度分数进行保序加密操作，实现对相关度分数的隐私保护. 为了获得更精确的排序结果，本文结合汉明距离和相关度分数，实现更为精确和高效的双因子排序. 最后，本文基于关键词的指纹构造了指纹索引树，旨在进一步提高搜索效率. 然而传统树结构采用了等值比较的遍历方法，这并不适用于在指纹索引树中比较汉明距离. 为了适应汉明距离的特性，本文设计了一种新型的索引树遍历方法.

5.2 细粒度的排序方法

5.2.1 相关度分数和保序加密

为了实现更加精确的排序，本文基于 $tf-idf$ 权值计算方法，并参考 tf 的亚线性尺度变换方法[15] 计算词频权重：

$$wf_{t,f}\begin{cases}1+\log tf_{t,f}, & tf_{t,f}>0\\ 0, & tf_{t,f}=0\end{cases} \tag{3}$$

最终的相关度分数计算公式如下：

$$wf-idf=wf_{t,f}\times idf_t \tag{4}$$

这样就可以利用式(4)计算关键词在文档中的相关度分数 $Score_j(1\leq j\leq m)$.

为了保护相关度分数 $Score_j$ 的隐私性，需要对其进行加密操作. 假设使用传统的对称加密方法 $\varphi(sk, Score_j)$，由于云服务器没有密钥 sk，无法对 $Score_j$ 进行排序，因此只能将 $\{FID_j, \varphi(sk, Score_j)\}$ 返回给用户，让用户在本地进行解密和排序操作，显然这样会增加用户的计算和带宽资源的开销.

因此，为了减轻用户在本地的计算压力，进而提高搜索效率，让排序操作全部由云服务器完成会更加符合用户需求. 本文采用非线性保序加密[24]，即 $f(x)=\log_c(d+e\times|x|)$，对 $Score_j$ 进行加密操作 $OPE(ek,Score_j)$. 即使 $Score_j$ 处于密文状态，云服务器仍然可以对其进行高效排序. 这样充分利用了云服务器强大的计算能力，在保护相关度分数隐私的同时，减少了用户的计算开销和带宽资源浪费，这在按需付费的云计算环境下显得更为必要.

5.2.2　双因子排序算法

当云服务器完成搜索操作后，如果将找到的所有文档返回给搜索用户，可能会浪费大量的带宽资源，特别是在海量文档集的环境下，更加重了服务器、授权用户和网络传输的负担. 因此需要对搜索的结果进行排序，只返回最相关的 top-k 篇文档.

如果云服务器对所有文档进行逐一比较并且排序，将会浪费大量的计算资源. 因此本文对所有关键词创建倒排索引 $I=(I_{w_1},I_{w_2},\cdots,I_{w_n})$. 其中，每个 $I_{w_i}\in I$ 包含两方面内容：一是关键词 w_i 的指纹值 S_i；二是含有关键词 w_i 的文档的信息集合，这些信息包括文档的标识符与相关度分数，可以表示为 $(FID_{i_j},Score_{i_j})(1\le i\le n,1\le j\le m)$. 则 $I_{w_i}=\{S_i,(FID_{i_1},Score_{i_1}),\cdots,(FID_{i_j},Score_{i_j})\}$，即 I_{w_i} 包含指纹 S_i 和 j 篇文档的相关信息. 为了简化说明，此处暂时不提及保序加密操作.

为了减少排序时间和提高搜索效率，本文借助汉明距离 d_{w_i} 和相关度分数 $Score_{i_j}$，设计了一种双因子排序方法. 首先，计算搜索陷门 T_w 和关键词索引 I_{w_i} 中的指纹值 S_i 之间的汉明距离 d_{w_i}. 然后，利用 d_{w_i} 对搜索结果进行初步排序，得到较为粗糙的排序结果集合 $I'=(I_{w_1},I_{w_2},\cdots,I_{w_n})$，且集合 I' 中的 I_{w_i} 是按照汉明距离从小到大排序的. 由于同一个关键词可能会被多篇文档所包含，此时的排序结果还不能够将包含相同关键词的文档区分开来，需要利用相关度分数进行更精确的排序，因此对每个 I_{w_i} 中各自的 $Score_{i_j}$ 进行排序. 最后按照 d_{w_i} 优先，$Score_{i_j}$ 次之的规则，从排序结果中选出前 k 篇文档返回给用户. 详细的步骤可参考算法 1.

算法 1.　双因子排序算法.

输入：倒排索引 I，搜索陷门 T_w

输出：精确排序的 top-k 篇密文文档 $C'=(c_1,c_2,\cdots,c_k)$

1. FOR each $I_{w_i}\in I$
　　计算 T_w 和 $S_i\in I_{w_i}$ 的汉明距离 d_{w_i}
　　END FOR
2. 将 I_{w_i} 按照 d_{w_i} 从小到大进行排序，得到 $I'=(I_{w_1},I_{w_2},\cdots,I_{w_n})$
3. 初始化 $Num=0$
4. FOR each $I_{w_i}\in I'$
　　将 FID_{i_j} 按照 $Score_{i_j}$ 从大到小排序，得到 $FID_{w_i}=(FID_{i_1},FID_{i_2},\cdots,FID_{i_j})$
　　FOR each $FID_{i_j}\in FID_{w_i}$
　　IF $Num<k$ 且 $FID_{i_j}\notin FID'$
　　将 FID_{i_j} 添加到集合 FID' 中
　　$Num++$
　　ELSE RETURN FID'
　　END IF
　　END FOR
　　END FOR
5. 找到 $FID'=(FID_1,FID_2,\cdots,FID_k)$ 对应的密文文档 $C'=(c_1,c_2,\cdots,c_k)$
6. RETURN $C'=(c_1,c_2,\cdots,c_k)$

5.3　指纹索引树

为了进一步拓展本方案，使之更适用于海量数据集，本文构建指纹索引树 G_W 来提高搜索效率. 如图 3，在构建索引树时，首先产生根节点 $\varnothing$，索引树的根节点为空集，然后计算 $w_i\in W$ 的指纹 S_i. 若 S_i 为 τ 个比特长，可以将 S_i 分成 τ/λ 段，每段都用 α_ρ 进行表示，α_ρ 为 λ 位的二进制比特流，则 S_i 可以表示为 $\alpha_1\alpha_2\cdots\alpha_{\tau/\lambda}$. 接着，用 α_ρ 代表一个节点，相同的 α_ρ 则为同一个节点. 当 α_ρ 为叶子节点时，在叶子节点中插入 $\{FID_{i_j},OPE(ek,Score_{i_j})\}$. 每一条从根节点到叶子节点的路径表示了一个关键词的指纹 S_i. 对每个关键词都进行上述操作，直至一棵完整的指纹索引树构造完成.

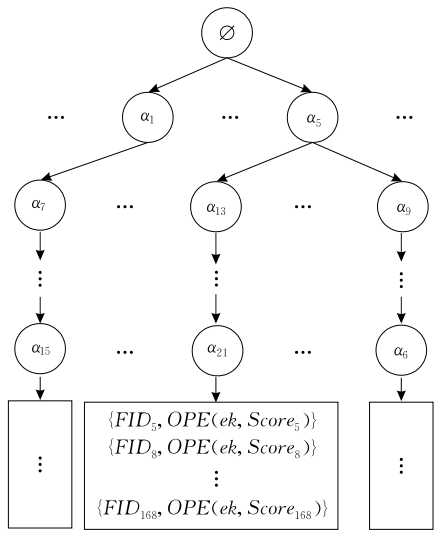

图 3　指纹索引树

5.4 新型遍历方法

在引入指纹索引树后会面临两个问题.第一,传统的索引树遍历方法是简单的等值比较,即如果比较的节点值与期望的不相等,就无法继续遍历下去.这样显然无法用于汉明距离的比较,因为汉明距离就是用于评估两个字符串的相近程度,它允许字符串间的差异性;第二,如果要计算某个关键词的指纹与陷门的汉明距离,就必须遍历到叶子节点,如果要得到所有汉明距离,就不得不遍历整棵索引树,但是这样做又违背了构建索引树来提高效率的初衷.为了达到无需遍历整棵索引树,同时又能进行汉明距离比较的目的,本文根据汉明距离的特性,设计了一种新型的遍历方法.

阶段 1.假设搜索用户希望获得排序最靠前的 k 篇文档,在遍历索引树之前,云服务器先构建一个容量为 k 的集合 U.在最开始的搜索阶段,无需任何判定条件,利用深度优先的方式直接检索最先能够找到的叶子节点,计算出根节点∅到此叶子节点路径上的字符串(即 S_i)和陷门 T_w 之间的汉明距离 d_{w_i},然后将该叶子节点关联的信息$[d_{w_i},\{FID_{i_j},OPE(ek,Score_{i_j})\}]$存入 U 中.继续检索叶子节点,同样将其关联的信息相继按照 d_{w_i} 从小到大的顺序存入 U 中.当集合 U 被 k 篇文档填满时,则不再继续寻找叶子节点,但此时 U 中包含的这些文档还不是搜索的最终结果.最后将集合 U 中 d_{w_i} 的最大值记为 $d_{\max}$,并进入下个阶段.

阶段 2.(1)若遍历到某个中间节点:可以算出根节点∅到此节点路径上的字符串和 T_w 之间的汉明距离 $\tilde{d}$,如果 $\tilde{d}>d_{\max}$,则说明根节点∅到此节点之后的节点路径上的汉明距离都大于 $d_{\max}$.这意味着即使沿着此节点继续往下寻找,得到的文档也不会比集合 U 中的文档更相关,所以此节点之后的节点都无需再遍历;如果 $\tilde{d}\leq d_{\max}$,则会继续遍历下去.

(2)若遍历到某个叶子节点:由于之前只有在满足 $\tilde{d}\leq d_{\max}$ 的情况下才能往下遍历,因此如果能遍历到叶子节点,则此时满足 $d_{w_i}\leq d_{\max}$,其中 d_{w_i} 表示根节点∅到此叶子节点路径上的字符串(即 S_i)和 T_w 之间的汉明距离.这表明了通过此叶子节点计算出的汉明距离 d_{w_i} 小于或等于之前计算出的汉明距离 $d_{\max}$.其中,若 $d_{w_i}=d_{\max}$,则不改变 U 中的内容;若 $d_{w_i}<d_{\max}$,则此叶子节点关联的信息$[d_{w_i},\{FID_{i_j},OPE(ek,Score_{i_j})\}]$,比存储在 U 中的 $d_{\max}$ 关联的信息$[d_{\max},\{FID_{i_j},OPE(ek,Score_{i_j})\}]$相关度更高.所以将此叶子节点关联的信息$[d_{w_i},\{FID_{i_j},OPE(ek,Score_{i_j})\}]$按照 d_{w_i} 从小到大的顺序插入集合 U.因为 U 的容量有限,因此会有部分最不相关的文档被淘汰出 U 集合,始终留下 k 篇文档.根据此时的 U 集合,将 $d_{\max}$ 重新设置,然后继续进行遍历直到找到最相关的 k 篇文档.最后再将这 k 篇文档根据相关度分数进行排序,就能返回精确排序的搜索结果.以上方法不需要遍历整棵索引树,节省了大量的遍历操作,又能够得到用于排序的汉明距离,搜索效率得到了进一步提升,更适用于海量数据集环境.详细的步骤可参考算法 2.

算法 2. 新型遍历方法.

输入:指纹索引树 G_W,搜索陷门 T_w

输出:精确排序的 top-k 篇密文文档 $C'=(c_1,c_2,\cdots,c_k)$

1. 初始化容量为 k 的空集合 U
2. WHILE (U 中包含的文档数量小于 k 时)
   计算 S_i 和陷门 T_w 之间的汉明距离 d_{w_i}
   令 $\Omega=[d_{w_i},\{FID_{i_j},OPE(ek,Score_{i_j})\}]$
   将 Ω 按照 d_{w_i} 从小到大顺序插入 U 中
   END WHILE
3. 将集合 U 中 d_{w_i} 的最大值赋给 $d_{\max}$
4. FUNCTION *SearchTree* (a)
   访问节点 a
   b=节点 a 的第 1 个子节点;
   WHILE(b 存在)
   IF b 未被访问 &$\tilde{d}\leq d_{\max}$
   IF b 是叶子节点 &$d_{w_i}<d_{\max}$
   将 Ω 按照 d_{w_i} 从小到大顺序插入 U
   将集合 U 中 d_{w_i} 的最大值赋给 $d_{\max}$
   ELSE *SearchTree* (b) //递归执行
   END IF
   END IF
   b=节点 a 的下一个子节点
   END WHILE
   END FUNCTION
5. FOR each $FID_{i_j}\in U$
   按 d_{w_i} 从小到大 $OPE(ek,Score_{i_j})$ 从大到小排序
   得 $FID'=(FID_1,FID_2,\cdots,FID_k)$
   END FOR
6. 找到 $FID'=(FID_1,FID_2,\cdots,FID_k)$ 对应的密文文档 $C'=(c_1,c_2,\cdots,c_k)$
7. RETURN $C'=(c_1,c_2,\cdots,c_k)$

6 性能测试

仿真实验使用 Java 采用程语言对本方案的性能进行测试,运行的环境为采用 Windows 7 64 位操作系统的台式机,处理器主频为 2.80GHz,内存为4GB.

使用 2139 篇 IEEE 等英文论文作为测试数据集，大小为 4.75GB. 使用 PDFBox 的 Java 类库对 PDF 文档进行内容抽取，通过提取文档标题的关键词，并过滤掉一些停用词(例如 a、the、about 等)后，形成关键词集合. 总共提取的关键词数量为 12811 个，不相同的关键词数量为 5242. 在实验中使用了 Hmac-SHA1 作为单向哈希函数，输出的长度为 160 bits. 通过将本方案和 Li 等人[10-11] 介绍的通配符和克方案进行比较，证明了本方案在空间、时间上的优势.

6.1　指纹与密文模糊集

(1) 空间开销. 图 4 表示 Simhash 指纹和传统方法构造的关键词密文模糊集的空间开销比较. 分别使用通配符和克两种传统方法，在编辑距离 d 分别为 1、2、3 的情况下构造密文模糊集. 由于需要构造模糊集，因此其所需的存储空间会随关键词个数增加而快速增大. 而本方案中每个关键词只产生一个对应的 Simhash 指纹，因此在关键词个数相同时，指纹的空间开销远小于上述两种方法构造的密文模糊集. 并且与传统方法相比，本方案的空间优势会随着数据集的增大而愈发明显.

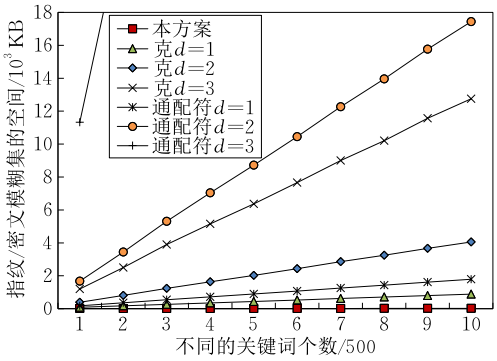

图 4　指纹/密文模糊集的空间开销

表 2 列出了当关键词个数为 5000 时，Simhash 指纹和不同编辑距离下的密文模糊集的空间开销的具体数值. 可以看出，密文模糊集的空间开销随编辑距离的增大而快速增加. 当编辑距离相同时，克方法的空间开销小于通配符方法. 在关键词相同的情况下，Simhash 指纹的空间开销最小，并且不会受到编辑距离的影响. 与传统方法相比，本方案的空间优势会随着编辑距离的增大而愈发明显. 当 $d=3$ 时，克方法的空间开销约是本方案的 700 倍，通配符方法的空间开销约是本方案的 6700 倍.

表 2　指纹/密文模糊集的空间开销

使用方法	空间开销/KB
本方案	18
克 $d=1$	883
克 $d=2$	1791
克 $d=3$	12752
通配符 $d=1$	4057
通配符 $d=2$	17433
通配符 $d=3$	120760

(2) 时间开销. 图 5 表示 Simhash 指纹和传统方法构造的关键词密文模糊集的时间开销比较. 在生成指纹的阶段，本方案对关键词进行了 n-gram 处理、哈希计算以及一系列映射操作. 如图，本方案生成指纹的时间开销，与 $d=2$ 的克方法相近，时间开销处于较低水平，由于还能极大的降低存储开销，因此本方案是十分高效的.

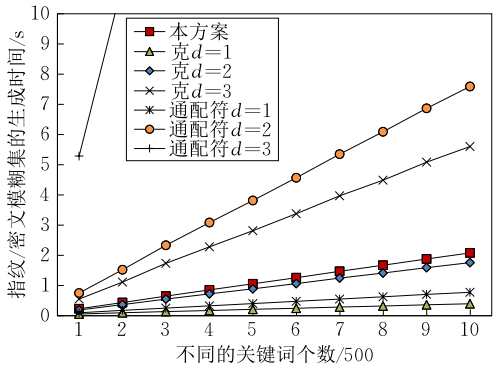

图 5　指纹/密文模糊集的时间开销

6.2　索　引

(1) 时间开销. 如图 6，本文使用 Li 等人[11] 介绍的索引树结构为传统方法构造索引，并将构造时间与本方案进行比较. 由图可以看出，传统方法的编辑距离增加时，因为模糊集合的大小急剧增加，所以索引构造时间也急剧增加. 而当编辑距离相同时，由于通配符产生的模糊集合相对较大，因此索引构造时间也较大. 可见传统方法的索引构造时间受模糊集合的影响非常大. 而本方案使用关键词指纹构造指纹索引树，不会受到模糊集合大小的影响. 在索引构造阶段，本方案计算了关键词指纹、相关度分数，并使用了非线性保序加密函数，在降低存储空间和优

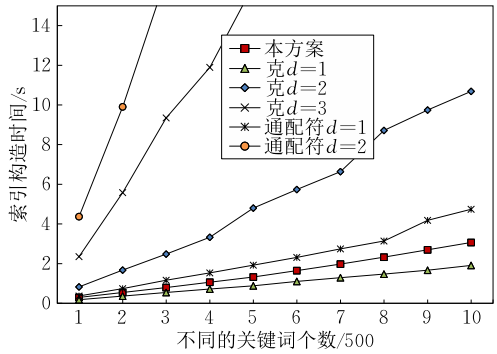

图 6　索引构造时间

化排序结果的同时，仍然能保持很少的时间开销.

(2) 保序加密. 图 7 表示保序加密前与保序加密后的索引构造时间开销比较. 两条曲线纵向的差值即保序加密的时间开销. 由图 7 可以看出，相比于保序加密前，保序加密后的索引构造时间开销只有少量增加，在关键词数量为 5000 个时，仅增加了 400 ms 左右. 由于在实际的搜索系统中，索引结构生成后并不会频繁地更新，因此增加的时间开销对系统影响很小. 从另一方面看，虽然在使用保序加密后，牺牲了一定的索引构造时间，却能让计算功能强大的云服务器完成排序操作，极大地减少了本地的计算压力，并且实现了对相关度分数的隐私保障.

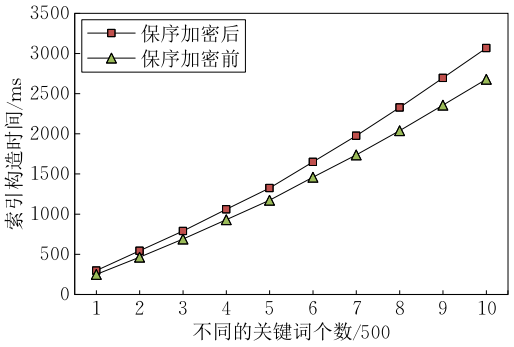

图 7 索引构造时间

6.3 搜 索

图 8 表示本方案与传统方法的搜索时间开销比较. 同样的，传统方法的编辑距离增加时，模糊集合的大小急剧增加，其搜索时间也相应增加. 本方案搜索阶段大致分为计算汉明距离和排序两个步骤，由于不会受到模糊集合大小的影响，因此搜索时间开销非常小. 另外，为了不对所有汉明距离进行排序，还可以通过设置一个阈值，将汉明距离大于阈值的结果过滤等方法提高搜索效率. 由于当编辑距离 $d=3$ 时，通配符的索引构造时间和搜索时间都非常大，因此不在图 6、图 8 中绘制出来.

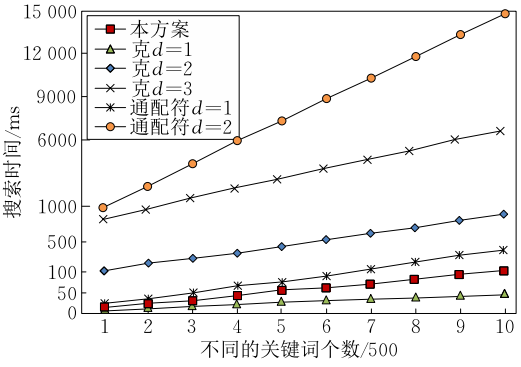

图 8 搜索时间

6.4 精确性

(1) 正确率与召回率. 为了对本方案的搜索结果进行评价，实验采用了信息检索领域最常用的两个基本评价指标——正确率(Precision)与召回率(Recall). 正确率是返回的结果中相关文档所占的比例，定义为

$$\text{正确率}=\frac{\text{返回结果中相关文档的数目}}{\text{返回结果数目}} \tag{5}$$

召回率是返回的相关文档占所有相关文档的比例，定义为

$$\text{召回率}=\frac{\text{返回结果中相关文档的数目}}{\text{所有相关文档的数目}} \tag{6}$$

图 9 为多次搜索结果的平均正确率. 正如 4.2.1 节(3)中提到的，不同的哈希函数 h 会产生不同位数的哈希值，从而对搜索结果的正确率造成影响，位数越多正确率越高. 由图 9 可以看出，在返回的文档数量小于 60 时，使用 Hmac-SHA1 的作为哈希函数的方案，其结果正确率接近 100%，即返回的基本是相关的文档. 这是由于双因子排序算法综合汉明距离和相关度分数，能够对搜索结果进行排序优化，优先返回最相关的文档集合. 随着返回的文档数量进一步增大，排序靠后的部分文档将被逐渐返回，其中可能包含了一些无关文档，因此搜索结果的正确率将呈现下降趋势. 而另一个使用了 Hmac-MD5 作为哈希函数的方案，由于其产生消息摘要的比特数(128 位)小于 Hmac-SHA1(160 位)，使得生成的指纹不能够将关键词的特征较好的区分开，从而影响了双因子排序算法的排序结果. 因此其搜索结果的正确率低于另一个方案. 同样的，在返回的文档较少时，其正确率较高，能够达到 80%左右的水平. 但随着返回的无关文档数量的增加，其正确率也将呈现下降趋势. 该实验证明了本方案使用的哈希函数生成消息摘要的比特数，会对搜索结果的正确率产生影响，也说明了双因子排序算法能够对搜索结果进行优化排

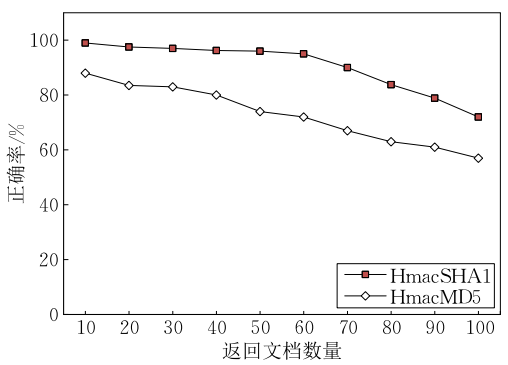

图 9 正确率

序，但针对不同的哈希函数排序效果有优劣之分.

图 10 为多次搜索结果的平均召回率. 由图 10 可以看出，在使用 Hmac-SHA1 作为哈希函数的方案中，随着返回文档数量的增加，召回率呈上升趋势. 由于召回率是返回的相关文档占所有相关文档的比例，因此在返回文档数量较少时，召回率的值较低. 由于本方案的双因子排序算法能够优化排序结果，因此当返回文档数量达到 90 时，召回率就已经趋近 100%. 这说明此时数据集中的相关文档已经基本上被返回. 而使用了 Hmac-MD5 的方案召回率低于上述方案，其原因和图 9 相似，即由于其产生消息摘要的比特数(128 位)小于 Hmac-SHA1(160 位)，使得生成的指纹不能够将关键词的特征较好的区分开，从而影响了双因子排序算法的排序结果，造成了最先返回的文档中包含的相关文档数量较少，因此召回率也较低. 随着返回文档数量的增加，其余相关文档被逐渐返回后，该方案的召回率也会逐渐上升，但在返回文档数量达到 100 时，召回率仍然低于使用 Hmac-SHA1 的方案，达到 80%左右的水平. 该实验证明了本方案使用的哈希函数生成消息摘要的比特数，也会对搜索结果的召回率产生影响.

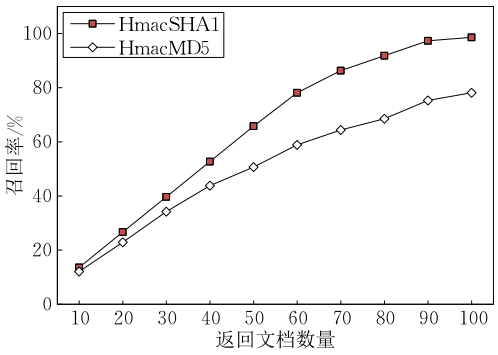

图 10　召回率

(2) 匹配效果. 为了更直观地体现本方案的模糊匹配和排序效果，表 3 列举了几个查询范例. 其中分别以拼错词 comput(正确词为 computer)、拼错词 keywold(正确词为 keyword)、正确词 secure 作为用户输入的查询词，并在下面列出了它们分别对应的模糊匹配词，而且这些匹配词是按照查询词指纹和匹配词指纹之间的汉明距离由小到大进行排序的. 由表 3 可以看出，每个关键词对应的匹配词大多数与查询词在结构上有相似性，这证明了本方案较好地实现了模糊搜索的功能. 不仅如此，表 3 也显示了排名越靠前的匹配词与查询词就越相似，例如，comput 依次匹配到 computer、computing、computation、computations 等等，显然，computer 比 computations 更接近查询词的结构. 这进一步证明了本方案的双因子排序算法能够对搜索结果进行排序优化，使得优先返回的文档更符合用户的搜索意图.

表 3　匹配效果示例

查询词(拼错)	查询词(拼错)	查询词
comput	keywold	secure
匹配词	匹配词	匹配词
computer	keyword	secure
computing	keywords	secured
computation	key	securely
computations	f5p5_keyword	securing
complex	keyed	security
computational	monkeys	cca-secure
complexes	older	secrecy
computationally	keyword-based	secret
compact	folding	secular
compatible	space	recurrence

(3) 误匹配. 另外，由表 3 可以看出，每个查询词都可能误匹配到一些形式上近似，但实际不是真正希望查询的关键词. 例如 comput 匹配到 complex 或 compact 等关键词，但显然这两个词与 comput 这个查询词的查询意图不符. 由于本方案是关键词模糊搜索方案，因此匹配到这些形式相似的词正体现了本方案的模糊搜索功能. 即使是传统的通配符模糊搜索方法，在根据编辑距离生成模糊集合时，也将产生大量误匹配的关键词. 关键点在于如何将这些误匹配的关键词尽量排到其他更精确的匹配词(如 computer、computing 等)之后，使得搜索能够优先返回符合用户需求的结果. 而本方案对每个关键词生成了指纹，通过双因子排序算法，能够对搜索结果进行排序，使得较精确的匹配词排序靠前，较不精确的匹配词排序靠后，大大减少了误匹配问题对模糊搜索系统带来的影响.

7　安全分析

(1) 文档隐私保障. 在本方案中，数据拥有者外包文档之前，使用传统对称加密方法(AES 或 3DES)对明文文档进行加密，保障文档的隐私性，并且会通过安全信道将密钥发送给授权用户. 在没有密钥的情况下，云服务器或攻击者难以对密文文档进行解密.

(2) 关键词隐私保障. 本方案将关键词进行 n-gram 处理，并使用带密钥的单向哈希函数对 gramset 中的元素进行哈希计算，最后将关键词转化为 Simhash 指纹. 假设关键词指纹生成算法 $sim(hk,\cdot)$ 在选择

明文攻击中，不能够取得不可区分性，则意味着存在一个算法 A，该算法能从指纹值中获得关键词的潜在信息.那么我们创建一个算法 A'，在 A' 中运用算法 A 来判断一些函数 $sim'(\cdot)$ 是一个伪随机函数还是一个随机函数.A' 能够访问预言机 $O_{sim'(\cdot)}$，该预言机可以输入一个参数 x 并返回 $sim'(x)$.在收到计算指纹的请求后，A' 会将 $O_{sim'(\cdot)}$ 的计算结果作为应答.之后，敌手输出两个长度相同的挑战关键词 w_0 和 w_1.A' 选取一个随机值 $b\in\{0,1\}$ 并将 w_b 发送给挑战者.接着，A' 被给定一个挑战值 y，y 是由伪随机函数或是随机函数生成的.A' 将 y 发送给 A，A 将应答一个值 $b'\in\{0,1\}$.假设 A 能以不可以忽视的概率猜测正确，即 $b'=b$，就表示 y 并不是随机生成的.于是，A' 可以判断 $sim'(\cdot)$ 是一个伪随机函数.然而，A 至多只能以约 1/2 的概率猜对，因此 A 不能从指纹值中获得关键词的潜在信息，说明关键词的隐私能够得到保障.

(3)相关度分数隐私保障.本方案引入一对多保序加密对相关度分数进行加密.它让文档标识符作为一个额外的种子参与到加密过程中，使得相同的明文不会固定映射为同一密文，而是会以一定的概率映射为区间 R 中的一个随机值.由于映射后的值有多种可能，因此增大了攻击难度.然而，当区间 R 的范围 $|R|$ 较小时，如果有大量相同的相关度分数，那么就不能有效平滑密文的分布状态.因此应该适当增大 $|R|$ 的值，以降低密文的重复率，从而使云服务器或攻击者难以进行分析.但如果 $|R|$ 太大则造成保序加密效率较低，因此要选取合适的 $|R|$.

此处借助最小熵来寻找区间 R 的大小.在信息论中，最小熵定义为

$$H(\varepsilon)=-\log(\max_{\alpha}\Pr[\varepsilon=\alpha]) \tag{7}$$

其中：ε 表示一个离散随机变量；α 表示 ε 最大概率呈现的状态.通常，$H(\varepsilon)$ 越大，越难预计出 ε.如果 $H(\varepsilon)\in w(\log k)$，则 ε 的最小熵会比较大，其中 k 表示 ε 的所有状态所需的比特长度[25].令 $H(\varepsilon)$ 为 $(\log k)^c$，$c>1$.根据文献[6]的推导过程，可以得到 $|R|$ 的合适取值：

$$\frac{\max\cdot 2^{5\log M+12}}{2^k\cdot\lambda}=\frac{\max\cdot M^5}{2^{k-12}\cdot\lambda}\leq 2^{-(\log k)^c} \tag{8}$$

其中：max 表示相关度分数的最大重复数量；λ 表示每个关键词对应的相关度分数的平均数量.令 $D=\{1,\cdots,M\}$，$M=|D|$.在保序加密的作用下，云服务器或攻击者只能对密文形式的相关度分数进行排序，而无法知道其具体值和匹配到的关键词，因此有效保障了相关度分数的隐私安全.

(4)索引隐私保障.指纹索引树结构中含有指纹、文档标识符以及保序加密后的相关度分数.带密钥的单向哈希函数保障了指纹的隐私，保序加密函数保障了相关度分数的隐私，文档标识符只是用于匹配密文文档的，并不会泄漏相关信息，因此云服务器或攻击者也无法从索引中知道文档、关键词或相关度分数的具体信息.

(5)查询隐私保障.授权用户通过数据拥有者发送来的密钥生成陷门，陷门结构也是一个安全的指纹，由于在方案(2)中已经分析过指纹的安全性，因此陷门的安全性也得到了保障.根据牛津字典，英语中大概包含 25000 个不同的单词，由于关键词数量较少，即使在没有密钥的情况下，云服务器或攻击者也可以通过暴力破解攻击单向函数，而在使用了带密钥的单向哈希函数并生成指纹后，暴力破解就变得十分困难.在不知道密钥的情况下，云服务器或攻击者无法生成有效陷门.

8 结束语

针对目前的密文关键词模糊搜索方案中，索引结构计算和存储开销大，排序结果不够精确等问题，本文提出了一种新型的模糊关键词搜索方案.该方案通过基于 Simhash 的关键词指纹生成算法，实现了模糊搜索的功能，并极大减少了索引的计算和存储开销；通过引入相关度分数，提高了排序结果的准确性，并使用保序加密保障相关度分数的隐私；双因子排序方法能够对搜索结果进行排序优化，提高了搜索结果的正确率与召回率；指纹索引树和新型遍历方法的使用，进一步提高了搜索效率.通过将本方案和传统的模糊搜索方案进行比较，证明了本方案在空间、时间上的优势.下一步工作是将关键词模糊搜索拓展到语义模糊搜索方面，实现更多样的搜索功能.

参 考 文 献

[1] Mell P, Grance T. The NIST definition of cloud computing. Communications of the ACM, 2010, 53(6): 50

[2] Song D X, Wagner D, Perrig A. Practical techniques for searches on encrypted data//Proceedings of the IEEE Symposium on Security and Privacy. Oakland, USA, 2000: 44-55

[3] Goh E J. Secure indexes. Cryptology ePrint Archive, 2003: 216-235

[4] Chang Y C, Mitzenmacher M. Privacy preserving keyword searches on remote encrypted data//Proceedings of the Applied Cryptography and Network Security. New York, USA, 2005: 442-455

[5] Curtmola R, Garay J, Kamara S, et al. Searchable symmetric encryption: Improved definitions and efficient constructions. Journal of Computer Security, 2011, 19(5): 895-934

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

[7] Wang C, Cao N, Li J, et al. Secure ranked keyword search over encrypted cloud data//Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS). Genova, Italy, 2010: 253-262

[8] Cao N, Wang C, Li M, et al. Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(1): 829-837

[9] Boneh D, Crescenzo G D, et al. Public key encryption with keyword search//Proceedings of the Eurocrypt. Interlaken, Switzerland, 2004: 506-522

[10] Li J, Wang Q, Wang C, et al. Fuzzy keyword search over encrypted data in cloud computing//Proceedings of the IEEE INFOCOM. San Diego, USA, 2010: 1-5

[11] Li J, Wang Q, Wang C, et al. Enabling efficient fuzzy keyword search over encrypted data in cloud computing. Cryptology ePrint Archive, 2009: 593

[12] Liu C, Zhu L, Li L, et al. Fuzzy keyword search on encrypted cloud storage data with small index//Proceedings of the 2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS). Beijing, China, 2011: 269-273

[13] Wang C, Ren K, Yu S, et al. Achieving usable and privacy-assured similarity search over outsourced cloud data// Proceedings of the IEEE INFOCOM. Orlando, USA, 2012: 451-459

[14] Wang J, Ma H, Tang Q, et al. Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. Computer Science & Information Systems, 2013, 10(2): 667-684

[15] Jin L I, Chen X. Efficient multi-user keyword search over encrypted data in cloud computing. Computing & Informatics, 2013, 32(4): 723-738

[16] Yu C M, Chen C Y, Chao H C. Privacy-preserving multikeyword similarity search over outsourced cloud data. IEEE Systems Journal, 2015, (99): 1-10

[17] Charikar M S. Similarity estimation techniques from rounding algorithms//Proceedings of the 34th Annual ACM Symposium on Theory of Computing. New York, USA, 2002: 380-388

[18] Manku G S, Jain A, Das Sarma A. Detecting near-duplicates for web crawling//Proceedings of the 16th International Conference on World Wide Web. New York, USA, 2007: 141-150

[19] Buyrukbilen S, Bakiras S. Secure similar document detection with Simhash//Proceedings of the 10th Very Large Data Bases Workshop. Trento, Italy, 2014: 61-75

[20] Fu Z, Shu J, Wang J, et al. Privacy-preserving smart similarity search based on Simhash over encrypted data in cloud computing. Journal of Internet Technology, 2015, 16(3): 458

[21] Ristad E S, Yianilos P N. Learning string-edit distance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998, 20(5): 522-532

[22] Manning C D, Raghavan P, Schütze H. Introduction to Information Retrieval. Cambridge: Cambridge University Press, 2008

[23] Agrawal R, Kiernan J, Srikant R, et al. Order-preserving encryption for numeric data//Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. New York, USA, 2004: 563-574

[24] Liu D, Wang S. Nonlinear order preserving index for encrypted database query in service cloud environments. Concurrency and Computation: Practice and Experience, 2013, 25(13): 1967-1984

[25] Bellare M, Boldyreva A, O'Neill A. Deterministic and efficiently searchable encryption//Proceedings of the 27th Annual International Cryptology Conference on Advances in Cryptology. Santa Barbara, USA, 2007:535-552

YANG Yang, born in 1984, Ph. D., associate professor. Her research interests are information security and privacy.

YANG Shu-Lue, born in 1990, M. S. candidate. His current research interest is searchable encryption.

KE Min, born in 1992, M. S. candidate. Her current research interest is searchable encryption.

Background

With the development of cloud computing, data owners are motivated to outsource their data to the public cloud for great flexibility and economic savings. Outsourced data are usually encrypted before uploading to the cloud for data confidentiality. Searchable encryption techniques are used to search on encrypted data without decryption. At present, there have been some studies on secure searching over encrypted cloud data, however, most of them are based on accurate keyword matching. The existing construction of fuzzy keyword search schemes mainly rely on the fuzzy keyword set, which lead to significantly large computation and storage overhead. To resolve these problems efficiently, a privacy preserving scheme without constructing fuzzy keyword set is proposed in this paper. Based on the idea of dimension reduction of Simhash, each keyword is transformed to a Simhash fingerprint by n-gram method to achieve fuzzy matching. This scheme combines hamming distance with keyword relevance score to sort the results efficiently and accurately. The use of the tree structure further improves the efficiency of proposed scheme. Both the security analysis and experiments results demonstrate that the proposed scheme achieves the fuzzy keyword ranked search over encrypted cloud data. Meanwhile, the computation and storage overhead is greatly reduced.

This paper is supported by the National Natural Science Foundation of China (Nos. 61402112, 61472307, 61472309, 61303198), the Science and Technology Project of Fujian Education Department (No. JA12028), the Major Science and Technology Project of Fujian Province (No. 2015H6013) and the Science and Technology Development Foundation of Fuzhou University (No. 2012-XY-17). The research aims of these projects include the privacy preserving issues of data users and owners, access control mechanisms, efficient searchable encryption techniques in cloud computing.