面向 DaaS 保护隐私的模糊关键字查询

李晋国[1)] 田秀霞[1),2)] 周傲英[2),3)]

[1)](上海电力学院计算机科学与技术学院 上海 200090)

[2)](上海市高可信计算重点实验室 上海 200062)

[3)](华东师范大学计算机科学与软件工程学院 上海 200062)

摘 要 由于在数据库服务(Database as a Service,DaaS)模式下,数据库服务提供者是半可信的(Honest-But-Curious),因此,为了保证外包数据的机密性和安全可查询,数据拥有者通常采用特定的加密技术加密外包数据,如采用可搜索加密技术、同态加密技术等实现外包加密数据上的安全查询.然而,且当前提出的大多数方法都基于关键字精确匹配查询,即使存在少量针对加密数据上的模糊查询,也在查询效率、存储开销和安全性方面存在一定的局限性,不适用于 DaaS 数据库服务模式.文中首次提出了融合具有高编码效率的 Huffman 编码和具有数据存储优势的布鲁姆过滤器,并结合现有的安全加密方法,实现了 DaaS 模式下保护隐私的模糊关键字查询处理.一方面,基于 Huffman 编码的树型索引提供了较高的查找效率;另一方面,基于布鲁姆过滤器的模糊关键字集合实现了较小的存储开销.安全分析、性能分析以及真实论文集上的实验结果进一步验证了文中查询算法的安全性、存储开销和查询效率.

关键词 DaaS;模糊查询;数据安全;哈弗曼编码;布鲁姆过滤器;云计算

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2016.00414

Privacy Preserving Fuzzy Keyword Search in Database as a Service Paradigm

LI Jin-Guo[1)] TIAN Xiu-Xia[1),2)] ZHOU Ao-Ying[2),3)]

[1)](*College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai* 200090)

[2)](*Shanghai Key Laboratory of Trustworthy Computing, Shanghai* 200062)

[3)](*School of Computer Science and Software Engineering, East China Normal University, Shanghai* 200062)

Abstract The database service provider is Honest-But-Curious in a Database as a Service (DaaS) paradigm. Thus, to guarantee the confidentiality and searchability of all outsourced data, the owners always encrypt them by using particular encryption techniques, such as the searchable encryption, homomorphic encryption, etc. However, most of these proposed works are designed for searching on outsourced data by accurate keyword matching. Only a few research works are concerning about encrypted-database fuzzy keyword search, while they also have limitations in the query efficiency, storage consumption and security. Therefore they are not suitable to the DaaS paradigm. In this paper, a privacy preserving fuzzy keyword search mechanism is proposed by combining the high coding-efficiency Huffman codes with high storage-efficiency Bloom filters under the support of existing secure encryption algorithms. On one hand, the Huffman coding based index tree provides high search efficiency; on the other hand, the similarity keyword set

收稿日期:2015-05-24;在线出版日期:2015-11-15.本课题得到国家"九七三"重点基础研究发展规划项目基金(2010CB328106)、国家自然科学基金(61202020,61370226)、上海市自然科学基金(12ZR1411900)、上海电力学院引进人才启动基金(K2015-008)资助.**李晋国**,男,1985 年生,博士,讲师,中国计算机学会(CCF)会员,主要研究方向为网络信息安全、应用密码学.E-mail:lijg@shiep.edu.cn.**田秀霞**(通信作者),女,1976 年生,博士,教授,中国计算机学会(CCF)会员,主要研究领域为数据库安全、大数据和云数据中的安全和隐私保护、应用密码学.E-mail:xxtian@fudan.edu.cn.**周傲英**,男,1965 年生,教授,博士生导师,主要研究领域为数据管理与信息系统,包括 Web 数据管理、中文 Web 基础设施、Web 搜索与挖掘、复杂事件处理与实时商务智能、不确定数据管理及应用、数据密集的计算、分布存储与计算、对等计算及其数据管理、Web 服务等.

based Bloom filters achieves low storage consumption. Analysis of security and performance, and real data set based experiments further confirm the security, storage consumption and search efficiency of proposed search techniques.

Keywords　DaaS; fuzzy keyword search; data privacy; Huffman code; Bloom filter; cloud computing

1 引　言

随着网络数据的日益规模化和集成化，数据库服务(Database as a Service，DaaS)逐渐成为大数据时代下数据的主要存储模式[1-2]. 企业或组织越来越倾向于将本地专有数据中心的维护和管理工作外包(Outsource)给 DaaS 服务提供商，从而降低设备升级、更新、维护以及专业人员培训等各个方面的运营成本. 新的服务模式引入了新的安全问题，即相比原有管理模式中的企业数据中心，服务提供商的服务器对用户具有信任风险，数据拥有者需要充分考虑存储数据的安全. 目前最有效的可信机制建立办法是对数据进行加密后再外包至服务器，同时为了保证加密数据的可用性和可搜索性，部分研究人员提出了相应的算法，使用户在对存储数据具备相应知识的情形下根据自身需求查询数据库[3-5].

然而，数据的类型和结构日益复杂，且应用规模庞大，用户对数据的认知总会存在一定的误差，因此需要 DaaS 系统支持密文数据上的模糊查询功能，容忍用户在查询过程中出现的少量的输入或格式误差[6]. 例如，用户输入关键字“Chine”查询含有索引关键字“China”的文件. 尽管查询条件无法精确匹配文件的索引关键字，但模糊查询机制仍然允许系统向用户返回所有相似程度较高的文件，这样用户才能以较大概率找到所需文件. 同时，为了保护数据和相关索引信息的机密性，避免用户隐私的泄露，所有的数据查询操作必须基于密文进行. 现有密文查询工作大部分针对精确数据匹配开展研究[7-11]，而对基于密文的模糊查询机制研究仍然较少，代表性工作主要有文献[6，12]. Li 等人在文献[6]中首次针对云环境下的加密数据提出了基于编辑距离的模糊查询算法，以通配符技术为基础实现了模糊关键字集合的压缩，然而这种以枚举为基础的查询算法仍无法满足用户对实时查询操作的效率需求. Wang 等人在文献[12]中以 Li 等人的工作为基础，提出了基于预定义符号集合(predefined symbol set)的语义树改善构建模糊集合的结构，提高查询效率，然而该算法性能对关键字前缀部分的重复率有较高的依赖性，若各个索引关键字之间极少存在相同前缀，则算法在效率方面的性能提升不明显. 除此之外，模糊集合元素和符号集合之间的对应关系会对算法查询结果的安全性产生威胁.

本文通过分析 DaaS 数据文件关键字索引的特征及其对密文查询效率的影响，针对现有工作在查询效率、数据存储和数据安全方面的不足，提出了一种基于 Huffman 编码和布鲁姆过滤器的安全模糊查询算法(Huffman code and Bloom filter-based Fuzzy Keyword Search，HB-FKS). HB-FKS 以关键字的 TF×IDF 评分(Term Frequency and Inverse Document Frequency)为基础构建 Huffman 树型结构，重新对数据文件索引进行组织，同时利用布鲁姆过滤器在数据存储方面的性能优势减少模糊关键字集合的存储功耗，确保以较高的效率实现 DaaS 服务模式下基于密文的模糊查询，最后基于真实数据集对 HB-FKS 的安全性和执行效率进行了较全面的分析和验证.

本文第 2 节对本研究领域中其他学者的相关工作进行比较和总结；第 3 节对本文用到的相关技术进行简要介绍；第 4 节描述本文采用的系统模型；第 5 节详细介绍本文提出的 HB-FKS 算法；第 6 节对提出的算法进行理论分析；第 7 节针对本文算法和相关工作设计仿真实验，并对实验结果进行对比分析；第 8 节总结全文.

2 相关工作

隐私保护是 DaaS 领域的研究热点之一，然而在确保数据隐私性的同时，也要确保数据的可用性，因此在 DaaS 服务模式下实现密文查询功能具有重要意义. 本小节将从 DaaS 模式现有隐私保护技术、隐私保护精确密文查询、隐私保护模糊密文查询和私有信息检索 4 个方面进行分析和总结.

2.1 DaaS 隐私保护技术

随着大数据存储技术的不断发展，DaaS 服务模式下的隐私保护问题日益突出，该问题也得到了研

究人员的广泛关注.传统的对称和非对称加密算法如 DES[13]、RSA[14]等是 DaaS 模式下数据隐私保护的主要技术手段,根据其加密粒度可划分为表、字段、元组、属性 4 种方式[5].算法通过密钥管理等方式[15-16]控制用户数据访问权限,确保数据的隐私性.然而,仅根据用户权限不加区分的反馈所有符合权限范围的数据,将引入大量不必要的解密计算和带宽功耗.密文数据在确保了用户隐私性的同时,也极大地影响了系统的可用性.

2.2 隐私保护精确密文查询

保护数据安全性最有效的手段是采用强加密算法,但为了确保系统的可用性,这些密文需要同时具备可搜索性.针对密文的精确查询技术最早是由 Song 等人[3]提出的,称为可搜索加密机制(The Searchable Encryption).可搜索加密机制根据其采用的加密技术可以分为基于对称加密的搜索算法和基于公钥系统的搜索算法.

早期的可搜索加密算法主要基于对称加密算法进行研究,代表工作主要有,2000 年 Song 等人[3]率先以伪随机函数和对称加密机制为基础实现了对密文数据精确快速的查询,同时严格证明了算法的安全性.2003 年 Goh 等人[4]基于布鲁姆过滤器结合伪随机函数进一步提出了一种高效且满足语义安全的索引构造算法 Z-IDX,有效地缩减了文件索引的存储开销,提高了搜索计算效率.2005 年 Chang 等人[17]引入随机比特位增强了索引抵御字典攻击的能力.

随着可搜索加密机制的发展,以公钥加密机制为基础的搜索算法得到了发展.2004 年 Boneh 等人[18]首次提出了基于双线性映射和 IBE 加密机制(Identity Based Encryption)的公钥密文搜索算法 PEKS.在该算法中,公钥加密的数据能被网关认证并发送至对应的用户,但内容不会被网关获取.Abdalla 等人[19]对 Boneh 等人提出的 PEKS 算法进行了改进,解决了原有算法中的一致性问题,设计了 3 种改良型算法:匿名身份加密算法 HIBE(Anonymous Hierarchical IBE),临时可搜索加密算法 PETKS 和基于身份的可搜索加密算法 IBEKS.

然而上述所有搜索算法中,系统均会根据用户的请求不加区分地返回所有满足查询需求的结果,这样的处理方式会导致不必要的带宽和传输能量的浪费,且增大了用户端的后处理负担.相比之下,基于密文的排名搜索算法(ranked search)提供了更加便捷有效的办法.排名搜索算法只会返回和用户需求最相关的 k 个文件.Wang 等人[10]针对云存储文件首次提出了安全关键字排名搜索算法.算法采用对称保序加密(Order-Preserving Symmetric Encryption,OPSE)实现了密文的安全排序,同时基于 TF×IDF 构造相应的排序函数.Cao 等人[11]则基于协调匹配原则(Coordinate Matching)和内积相似度量化规则(Inner Product Similarity)实现了针对多关键字的安全排名查询机制,通过引入冗余关键字提升了算法的安全性.然而由于 Cao 等人的算法没有将关键字频率作为计算参数引入相关度评分的计算公式,因此部分包含高频率关键字的搜索结果可能被系统忽略,从而导致最终的查询结果出现错误.

上述排名搜索算法以较高的查询效率满足了用户的需求,然而算法并不能容忍用户在输入查询条件时出现的少量的字符或格式误差.

2.3 隐私保护模糊密文查询

目前针对 DaaS 下基于密文的模糊查询研究工作不多,相对精确的密文查询,模糊密文查询难度更高,这是由于在加密强度较高的算法中,明文之间 1 比特的微小误差在对应密文中可能产生巨大的差异,用户输入过程中的细节错误会直接导致异常的查询结果.Park 等人[20]在 2007 年针对这一问题提出了基于汉明距离(Hamming distance)的模糊查询算法,采用了伪随机函数和椭圆曲线实现对索引的安全性保护,获得了很高的安全性,但算法计算复杂度较高,且以汉明距离为基础的相似度度量体系对用户提出了较高的专业背景要求,因此系统可用性难以保证.

2010 年 Li 等人[6]引入了编辑距离衡量关键字之间的相似度,简化了数据相似度的计算,同时采用通配符"*"替代关键字不同位置的字母元素,使特定位置字符取值的可能性由 26 种变为 1 种,减少了关键字模糊集合的存储空间.然而,模糊查询的实现基础仍然是以关键字模糊集合为基础的枚举模式,查询效率受到了极大影响.

2012 年 Wang 等人[12]采用了 Li 等人的算法,结合通配符和编辑距离计算索引,并在此基础上进一步提出采用预定义符号集合对应这些索引关键字以构建基于密文符号的语义树,文件索引通过索引树重新组织后使得查询效率得以提高.然而,语义树的查询效率极大地依赖于索引关键字的重复率,因此,算法对模糊查询效率的性能提升很有限.此外,关键字和符号集合一一对应的模式存在安全隐患,难以有效地抵御字典攻击.

2.4 私有信息检索技术

私有信息检索技术(Private Information Retrieval,PIR),其应用目的和可搜索加密技术是一致的,即

在数据库服务器查询数据过程中，不暴露用户的任何隐私.

Chor 等人[21]于 1998 年首次提出 PIR 的概念，并提出了一种 k 服务器的方案. 用户通过向不同服务器发送根据同一查询条件生成的随机副本，随后利用不同服务器反馈的结果进行异或计算获取最终结果. 然而其通信复杂度达到了 $O(kn^{1/\log k})$，n 为查询条件长度. Kushilevitz 等人[22]于同年提出了一种基于二次剩余定理的 c-PIR 协议，有效地降低了通信复杂度，并从理论上证明了在数据查询过程中的计算性隐私可得到有效保障，然而每次数据查询过程中，服务器均需要对所有数据进行取模计算，其计算复杂度太高，系统难以承受. Wang 等人[23]针对这一问题进行研究，采用含有目标数据子矩阵对原有数据矩阵进行了替换，提出了一种改进的 bbPIR 技术，在损失部分隐私性的同时提高了查找效率. 为了解决 PIR 协议计算复杂度过高的问题，Papadopoulos 等人[24]提出了一种分布式的方案 pCloud，通过节点协作处理不同的数据分块. 然而该算法的实现仍是基于二次剩余定理的，因此，用户仍需通过所有数据分块来计算最终的结果. 一旦出现单点故障就无法顺利获取部分数据分块，用户需要重新进行查询，这将导致大量重复计算，消耗额外的网络资源.

现有的 PIR 技术一方面计算复杂度过高，不能适用于存储大规模数据的 DaaS 服务模型；另一方面，现有的 PIR 技术侧重用户隐私，但在服务器端存储的数据并非密文数据，考虑到本文中的数据库服务器并不是完全可信的，因此，PIR 技术不满足本文的安全需求.

3　预备知识

3.1　TF×IDF 规则

TF×IDF 规则是一种基于加权的字词统计技术，用于衡量单个关键字和文件集合中指定文件之间的关联程度[25]. 关联强度随着该关键字在该文件中的出现次数增加而增强，但和关键字在整个文件集合中的出现频率呈反比关系[26-27]. TF(Term Frequency)，即词频，衡量的是关键字在指定文件中的出现频率；IDF(Inverse Document Frequency)，即逆向文件频率，衡量的是关键字在整个文件集合中的出现次数. 关键字 w_i 和文件 f_j 的 TF×IDF 评分计算规则为

$$sc(w_i,f_j)=\frac{1}{|f_j|}(1+\ln ft_{w_i})\ln\left(1+\frac{N}{fn_{w_i}}\right) \quad (1)$$

其中，$|f_j|$ 表示文件长度；ft_{w_i} 是关键字 w_i 在文件中出现的频率，即 TF；fn_{w_i} 是整个文件集合中包含关键字 w_i 的文件个数；N 是文件集合中文件的总数量.

3.2　布鲁姆过滤器

布鲁姆过滤器是一种存储有效的向量型数据结构，通常用于判别单个元素和指定集合之间的隶属关系[28-29]. 标准的布鲁姆过滤器是由多个比特位组成的特殊向量阵列，各个比特位的初始值设置为 0. 各个元素通过该布鲁姆过滤器对应的 m 个独立同分布的哈希函数映射至向量阵列相应的比特位. 被映射的比特位的值设置为 1. 例如，长度为 BL 的布鲁姆过滤器 B 对应的哈希函数为 $h_1,h_2,\cdots,h_m$，若向 B 中添加元素 x，则 $h_1(x),h_2(x),\cdots,h_m(x)$ 对应的比特位的值设为 1. 当判别元素 y 是否隶属布鲁姆过滤器 B 时，需判别 $h_1(y),h_2(y),\cdots,h_m(y)$ 对应的 m 个比特位的值是否全部为 1. 这 m 个值中任意一个为 0，则 y 一定不在 B 中，反之 y 以较大的概率存在于 B 中.

图 1 给出了布鲁姆过滤器成员隶属关系判别过程. (1) 布鲁姆过滤器各个比特位的初始值设置为 0；(2) x_1,x_2,x_3 分别通过哈希函数 h_1,h_2,h_3 被映射至布鲁姆过滤器，作为其对应集合的元素成员；(3) 通过对 y_1,y_2,y_3 进行相应的哈希映射，判别其是否是该布鲁姆过滤器对应集合的成员. 由于 $h_1(y_3)$ 对应的比特位为 0，因此 y_3 不在该布鲁姆过滤器中. y_1 和 y_2 对应的哈希映射比特位的值均为 1，因此二者以较大概率存在于该布鲁姆过滤器中. 但是，根据原有集合成员 x_1,x_2 和 x_3 的实际映射位置判断，y_2 实际上并没有存储在布鲁姆过滤器中，因此 y_2 是假阳性数据.

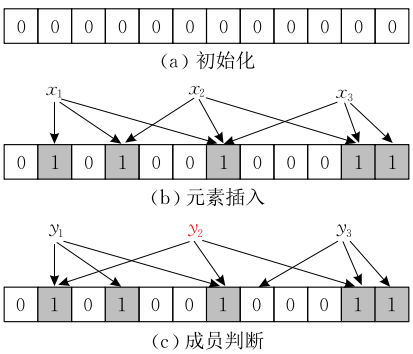


图 1　布鲁姆过滤器

3.3　Huffman 树

Huffman 树是带权路径长度最小的最优二叉树，权值越大的节点越接近根节点，即深度越浅，反之则远离根节点[30-31]. 以此为基础构建的可变字长 Huffman 编码可基于字符或关键字的出现频率来实现最短平均长度的最佳编码. Huffman 树的构造

方法是自底向上的:首先通过关键字频率设置叶子节点的权值,随后重复合并权值最小的叶子节点或中间节点以构造新树,并以新树的根节点作为新的合并操作对象,最终形成 Huffman 树[32]. 对 Huffman 树的各个左右路径赋予 0-1 编码生成相应的 Huffman 编码. 图 2 给出了叶子节点的频率权值分别为 4,5,7,8,11,17,19,29 的 Huffman 树构造过程.

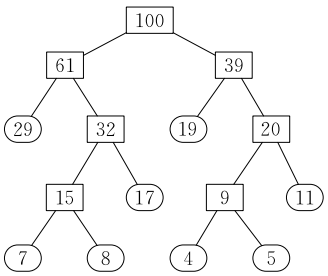


图 2 Huffman 树

4 模 型

4.1 系统模型

DaaS 服务系统模型如图 3 所示,通常主要包含 3 种角色[5,33]:数据拥有者(Data Owner,DO)、DaaS 数据库服务提供方(DaaS Provider,DSP)和用户(Data User,DU). (1) DO 通常是拥有原始数据的企业或组织,将本地数据以密文形式委托 DSP 进行存储,为了在保护数据安全性的同时实现 DSP 数据库的可查询性,DO 需要结合特定的加密技术针对这些密文建立相应的隐私保护索引;(2) DSP 是专业维护、更新和升级 DaaS 服务数据库和硬件服务器的提供商,负责存储备份和管理 DO 数据,并向 DU 提供数据查询服务. 由于 DSP 管理人员的个人行为无法完全掌控,数据的机密性得不到有效保障. 因此,DSP 存储的数据均是事先由 DO 加密的. 在提供查询服务时,为了保护数据和索引隐私,DSP 正确执行查询操作的同时不能获知查询条件和查询数据的真实内容;(3) DU 是指经过系统授权的用户,可以合法查询 DSP 中存储的 DO 数据. 出于安全性考虑,DU 采用的查询条件也需要经过加密技术处理;(4) TTP(Trusted Third Party). 本文采用的模型引入了完全可信第三方 TTP[5],负责查询服务中的复杂计算任务,主要包含对 DO 数据进行加密预处理、索引计算,并对 DU 查询条件进行相应编码转换. TTP 降低了查询服务模型中 DU 和 DO 的计算负担,但为了保护 DU 查询条件和 DO 数据的机密性,TTP 必须独立于外包服务提供商 DSP,由第三方权威机构提供. 例如,DaaS 服务模式下的线上医疗信息系统中,病人需上传病历等敏感信息为医生看诊提供辅助材料,可通过 TTP 提供的可信服务将这些信息加密并建立索引,存储到医疗信息中心;医生根据看诊需求查询病历信息时,通过 TTP 提供的可信服务将查询信息转换成相应加密编码,并发送至医疗信息中心;信息中心在不对查询编码和病历信息解密的情形下,通过计算将相应的文件反馈给被授权的医生(TTP 授权). 该系统的 TTP 可以由政府相应监管部门承担;(5) DaaS 服务模式下的数据库通常规模庞大,在现有的基于密文查询的相关研究工作中[6,11-12],查询关键字通常是系统利用数据库中存储文件预先计算的,可降低关键字更新频率,避免文档样本不足导致的相关参数计算误差. 因此,在本文提出的模糊关键字查询机制中,合法 DU 和 DO 之间约定的常用查询关键字 $w_1,w_2,\cdots,w_k$,是由 TTP 通过对数据库服务器 DSP 中存储的文档数据库 $F=\{f_1,f_2,\cdots,f_n\}$ 进行计算后得到的. 数据库拥有数量充分的文档样本,对于查询关键字的更新需求较低.

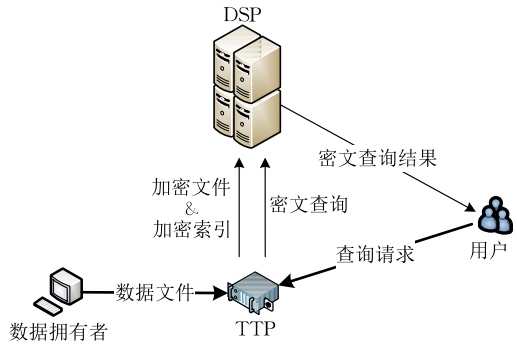


图 3 DaaS 服务系统模型

4.2 安全模型

本文采用的安全模型设计如下:DO 和 DU 通过安全信道向可信方 TTP 传输数据(如图 3 所示). TTP 和 DSP 之间的传输信道可能被窃听,且 DSP 是 Honest-but-Curious 的. 因其 Curious 特性,DSP 作为系统内部成员可能发起两种形式的攻击:一种是查看数据拥有者 DO 委托的数据,从而泄露与 DO 相关的隐私信息,本文主要采用现有对称加密技术加强对数据的保护;另一种是对用户 DU 的查询条件进行分析,获取 DU 查询行为等隐私信息,本文主要通过 Huffman 编码和布鲁姆过滤器技术隐藏查询条件和数据之间的关联信息. 在本文的安全模型中,DO,DU 和 TTP 均是可信的.

4.3 符 号

本文安全模糊查询机制中所采用的符号如表 1 所示.

表 1　符号说明

符号	说明
F	数据文件集合
f_x	$f_x \in F$
cf_x	f_x对应的密文
w_x	索引和查询条件中的关键字
c_{w_x}	w_x对应的密文
Sc	关键字的 TF×IDF 评分
$S_{w_x\_d}$	编辑距离为 d 的 w_x模糊集合
$B_{w_x\_d}$	$S_{w_x_d}$对应的布鲁姆过滤器
haf_{w_x}	w_x的 Huffman 编码

5　保护隐私的模糊查询

针对现有算法在查询效率和存储功耗方面的不足，本节基于 Huffman 编码和布鲁姆过滤器提出了一种安全模糊查询算法，目的是在保护数据私密性的同时以较低的功耗实现较高效率的查询处理. 算法主要分为预处理、Huffman 编码索引建立和保护隐私的查询转换 3 个部分，算法的具体细节如下.

5.1　预处理

系统对数据文件的预处理主要包含 3 个部分，即对文件集合关键字的 TF×IDF 评分计算，相关加密算法和密钥的约定以及对 DO 数据文件的加密存储，具体细节如下：

(1) 由可信第三方 A 对准备存储至数据库服务器中的文件 $F=\{f_1,f_2,\cdots,f_n\}$，通过执行算法 1 进行关键字关联度计算，获取 TF×IDF 评分最高的 k 个关键字 $w_1,w_2,\cdots,w_k$，并以此为基础建立各个存储文件的索引树 I. 根据式(1)，关键字 w_i在文件集合 F 下的 TF×IDF 评分计算公式如下：

$$sc(w_i,F)=\sum_{f_j\in F}\frac{1}{|f_j|}(1+\ln ft_{ij})\ln\left(1+\frac{N}{fn_{w_i}}\right) \quad (2)$$

其中，$|f_j|$是文件 f_j的长度，ft_{ij}是关键字 w_i在文件 f_j中的出现频率，fn_{w_i}是文件集合中包含关键字 w_i的文件数量，N 是文件集合中的文件总数.

算法 1.　文件集合 TF×IDF 评分算法.

输入：文件集合 $F=\{f_1,f_2,\cdots,f_n\}$

输出：评分 sc 最高的 k 个关键字 $w_1,w_2,\cdots,w_k$

1. 对各个文件f_j进行扫描，结合现有分词算法[34]划分文件集合F包含的所有关键字$W=\{w_1,w_2,\cdots,w_{\max}\}$；
2. 统计关键字在文件集合中的词频 $ft_1,ft_2,\cdots,ft_{\max}$以及包含同一关键字的文件数量 $fn_1,fn_2,\cdots,fn_{\max}$；
3. 根据式(2)以及上述计算结果计算各个关键字的相应 TF×IDF 评分 $sc_1,sc_2,\cdots,sc_{\max}$；
4. FOR $1\le i\le$max
　　FOR $i\le j\le$max
　　IF $sc_i>sc_j$
　　$tempvalue=sc_i$；
　　$sc_i=sc_j$；
　　$sc_j=tempvalue$；
　　END IF
　　END FOR
　　END FOR
5. 返回 $sc_1,sc_2,\cdots,sc_k$对应的关键字 $w_1,w_2,\cdots,w_k$.

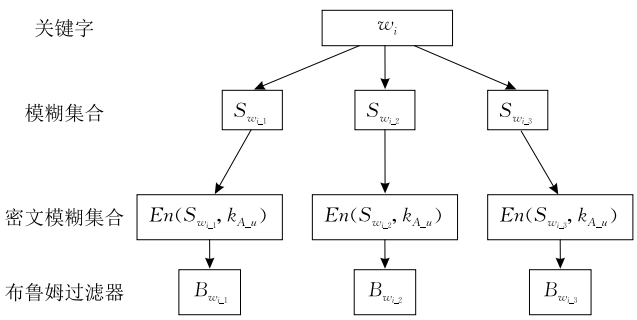
(2) A 和授权用户共享加密函数En和密钥k_{A_u}，用以实现安全通信；

(3) 文件$f_1,f_2,\cdots,f_n$被加密后得到$cf_1,cf_2,\cdots,cf_n$. 密文结果和相应索引树 I 被传输至 DSP 服务器进行存储，索引树生成过程详见 5.2 节.

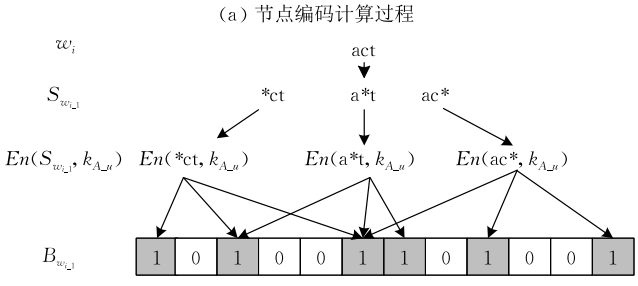
5.2　Huffman 编码索引树建立

索引生成的基本思想：首先，根据文件索引关键字的 TF×IDF 评分构建 Huffman 索引树并生成相应编码，评分越高的关键字在索引树的深度越小，随后，将各个关键字的模糊集合存储到相应的布鲁姆过滤器，节省存储空间：

(1) 叶子节点的生成. 为了支持模糊查询机制，本文采用 Levenshtein[35] 提出的基于编辑距离的相似度量化方法来衡量关键字之间的相似程度. 为了降低存储开销，本文基于 Li 等人[6] 提出的关键字模糊元素集合计算方案，结合具有存储优势的布鲁姆过滤器构造叶子节点. 具体如下：可信第三方 A 首先针对关键字 w_i计算其不同编辑距离下相应的关键字模糊元素集合 $S_{w_{i\_1}},S_{w_{i\_2}},S_{w_{i\_3}}$. 如图 4 所示，$w_i=$



(a) 节点编码计算过程



(b) 节点计算示例

图 4　索引树叶子节点的构建

“act”，则其编辑距离 $d=1$ 时，相应的模糊元素集合 $S_{w_{i\_1}}=\{$“*ct”，“a*t”，“ac*”$\}$. 随后根据不同的编辑距离，将相应的模糊元素集合中的各个关键字，先加密再分别映射至不同的布鲁姆过滤器 $B_{w_{i\_1}}$，$B_{w_{i\_2}}$，$B_{w_{i\_3}}$，以构造对应各个关键字的叶子节点. 加密函数为 En，密钥为 k_{A_u}.

(2) 基于叶子节点及其父节点的子树生成. 在本文算法中，叶子节点根据编辑距离划分共有 3 类，对应的编辑距离分别为 1，2，3. 其父节点为相应关键字的 TF×IDF 评分. 例如叶子节点 $B_{w_{i\_1}}$，$B_{w_{i\_2}}$，$B_{w_{i\_3}}$ 对应父节点的存储信息为 sc_{w_i}. 叶子节点和其父节点构成相应 Huffman 子树 t_{w_1}，t_{w_2}，…，t_{w_k} 对应的各个关键字. 图 5 给出了 Huffman 子树的基本结构.

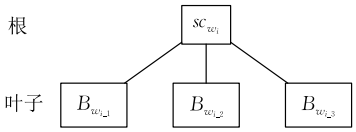
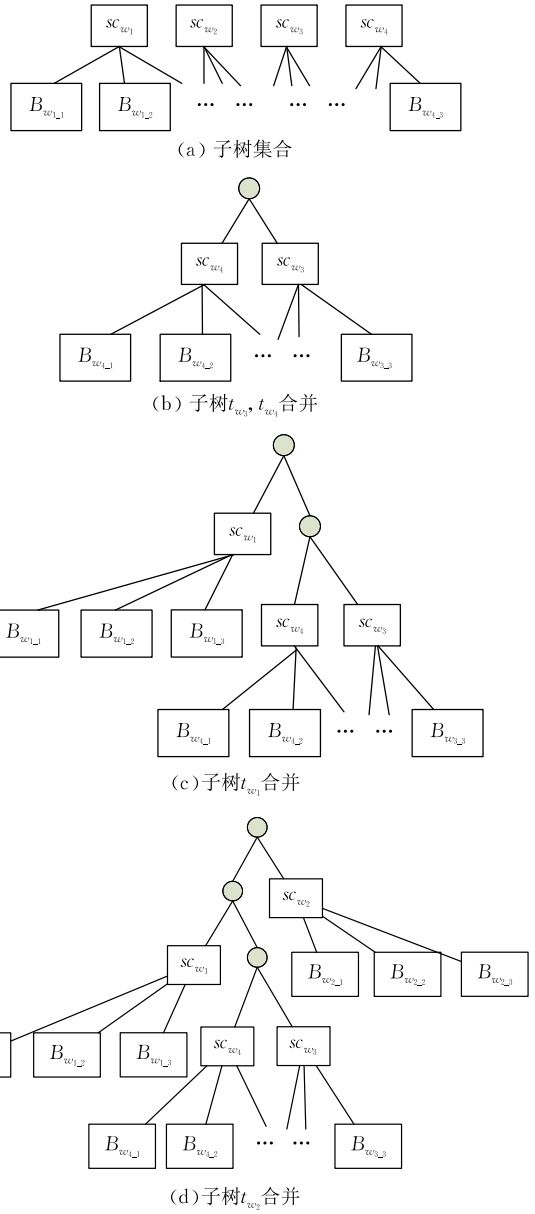


图 5 Huffman 子树构建

(3) Huffman 树的生成和节点编码计算. 上述子树 t_{w_1}，t_{w_2}，…，t_{w_k} 经过进一步计算合并处理构建 Huffman 树. 具体如下：可信方 A 首先在上述子树组成的森林 T 中选择根节点 TF×IDF 评分信息最小的两棵子树 t_i 和 t_j 并进行合并，组成一棵新树 t_{i_j}. 新树根节点的 TF×IDF 评分信息值为子树 t_i 和 t_j 根节点评分信息之和，且这两棵子树分别作为新树 t_{i_j} 的左右子树. 随后 A 将新树 t_{i_j} 加入森林 T，同时删除子树 t_i 和 t_j. 重复上述过程直到森林 T 只包含一棵树为止，此时，Huffman 树建立. 例如图 6(a)～(d)所示，共存在 4 棵子树 t_{w_1}，t_{w_2}，t_{w_3}，t_{w_4}，对应根节点关键字的 TF×IDF 评分分别为 $sc_{w_1}=0.5$，$sc_{w_2}=1.4$，$sc_{w_3}=0.4$，$sc_{w_4}=0.3$，经上述计算过程建立的 Huffman 树如图 6 所示.

节点 Huffman 编码的计算方式如下：可信方 A 从 TF×IDF 评分信息最低的两个节点开始，左节点设置为 0，右节点设置为 1，对各个节点进行相应编码 haf_i. 由上述树型结构和 Huffman 编码构成相应的索引树 I，经可信方 A 发送至数据库服务器进行存储. 索引树 I 的各个节点不包含 TF×IDF 评分信息，叶子节点为布鲁姆过滤器编码，其余部分则为 Huffman 编码，同时各个关键字 w_i 的 Huffman 编码即为各个叶子节点父节点的编码. 以上图中数据为例，相应编码如图 7 所示.



(a) 子树集合

(b) 子树t_{w_3}，t_{w_4}合并

(c)子树t_{w_1}合并

(d)子树t_{w_2}合并

图 6 基于关键字 TF×IDF 评分的 Huffman 树建立

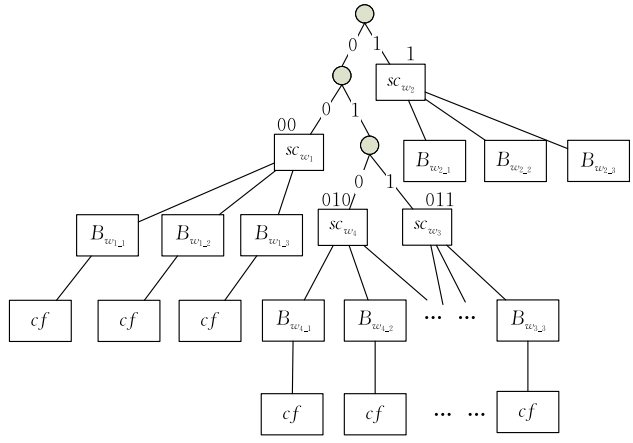


图 7 基于 Huffman 编码的索引树

5.3 保护隐私的查询转换

用户 DU 的查询条件将被 TTS 转换成相应密文和编码，再由 DSP 执行查询：

(1) 用户查询条件转换. 用户首先将查询条件 $Q=(c_{w_q}\,|\,d_q)$ 传输至可信方 A 进行转换 ($d_q\leq 3$, $c_{w_q}=En(w_q,k_{A\_u})$). A 对查询条件解密后，首先计算和 w_q 编辑距离小于或等于 d_q 的相似关键字 w_i, w_{i+1}, …. 随后将关键字 w_i, w_{i+1}, … 转换成为对应的密文和 Huffman 编码，形成新的查询条件 $Q'=(c_{w_i}\,|\,haf_{w_i},c_{w_{i+1}}\,|\,haf_{w_{i+1}},\cdots)$，再发送至数据库存储服务器进行查询.

(2) 执行查询. 如算法 2 所示，数据库服务器对比 Huffman 编码 haf_{w_i}, $haf_{w_{i+1}}$, … 和索引树 I，查找相应的节点 N_{w_i}, $N_{w_{i+1}}$, …. 随后使用查询结果 N_{w_i}, $N_{w_{i+1}}$, … 各个子节点的布鲁姆过滤器对密文 c_{w_i}, $c_{w_{i+1}}$ 进行映射，判断 $B_{w_{i\_1}}$, $B_{w_{i\_2}}$, …, 是否包含该元素.

算法 2. 索引树查询算法 Search(haf_{w_i}, $root$).

输入：Huffman 编码 haf_{w_i}，其长度为 $colen$，索引树 I 根节点 $root$ 和节点集合 $N=\{N_1,N_2,\cdots\}$

输出：查询结果 N_{w_i}

1. 设置 N_t 初始值为索引树 I 的起始查询节点，即 $N_t=root$；
2. 使用计数器 $count$ 记录 Huffman 编码 haf_{w_i} 各个比特位的相应下标，其初始值为 $count=0$；
3. IF $count\leq colen$
   IF $haf_{w_i}[count]==1\&\&N_t.leftchild!=$NULL
   $N_t=N_t.leftchild$;
   $count++$;
   Search(haf_{w_i}, N_t);
   ELSE IF $haf_{w_i}[count]==0\&\&N_t.rightchild!=$NULL
   $N_t=N_t.rightchild$;
   $count++$;
   Search(haf_{w_i}, N_t);
   END IF
   END IF
4. IF $count==colen$
   IF $N_t.leftchild==$NULL $\|$ $N_t.rightchild==$NULL
   $N_{w_i}=N_t$;
   END IF
   END IF
5. RETURN N_{w_i}.

根据布鲁姆过滤器成员隶属关系判断规则，当 $h_1(c_{w_i})$, $h_2(c_{w_i})$, …, $h_m(c_{w_i})$ 均为 1 时，c_{w_i} 以概率 $1-p$ 存在于 $B_{w_{i\_j}}$ 中，数据库服务器将向用户返回 $B_{w_{i\_j}}$ 指向的加密数据文件 cf. 用户通过解密 cf 得到相应文件 f. 其中，h_1, h_2, …, h_m 是布鲁姆过滤器 $B_{w_{i\_j}}$ 对应的独立同分布哈希映射函数. 参照布鲁姆过滤器的假阳性计算公式[36]，概率 p 值为

$$p=\left(1-\left(1-\frac{1}{BL}\right)^{mn}\right)^m\approx(1-\mathrm{e}^{-mn/BL})^m \quad (3)$$

其中，BL 为布鲁姆过滤器的比特数，m 为布鲁姆过滤器对应的哈希函数个数，n 则为布鲁姆过滤器中插入的元素个数. 查询执行过程如图 8 所示.

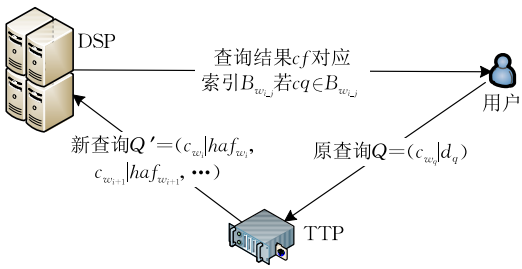


图 8　查询处理

6 分　析

本节对 HB-FKS 算法的计算复杂度、安全性和查询性能分别进行了分析，并与主要文献[6,12]中提出的算法进行了对比.

6.1 算法复杂度分析

假定 DSP 服务器存储了 n 个文件，DO 和 DU 约定的关键字个数为 k 个且关键字频率各不相同，关键字前缀重复率为零，单个布鲁姆过滤器包含 m 个相互独立的哈希函数，查询的最大编辑距离为 d. 为了方便表述，不妨令所有关键字长度为 l，从 l 个元素中任选 d 个元素的组合种类数量为 C_l^d，则 HB-FKS 和文献[6,12]中查询算法各个步骤的计算功耗分析如下(本文将文献[6]的算法称为 WFKS(Wildcard-based Fuzzy Keyword Search)，将文献[12]的算法称为 TTSS(Trie-tree-based Similarity Search))：

(1) HB-FKS 计算功耗分析. HB-FKS 中索引的生成计算主要包括各个关键字的模糊集合计算、计算结果的布鲁姆过滤器映射操作和以关键字 TF×IDF 评分为基础的 Huffman 编码生成，其计算开销为 $(C_l^1+C_l^2+\cdots+C_l^d)km+\log_2 k$；HB-FKS 中查询条件的生成主要包括查询关键字模糊集合的生成和对应 Huffman 编码的计算，其计算开销为 $(C_l^1+C_l^2+\cdots+C_l^d)\times\log_2 k$；HB-FKS 中模糊查询操作主要是使用上述步骤中生成的查询关键字

Huffman 编码集合对 Huffman 索引树进行遍历，并在叶子节点处进行布鲁姆过滤器映射操作，其计算开销为$(C_l^1+C_l^2+\cdots+C_l^d)\times(\log_2 k+m)$.

(2) WFKS 算法计算功耗分析. WFKS 中索引的生成计算主要是构建各个关键字的相应模糊集合，因此其计算开销为$(C_l^1+C_l^2+\cdots+C_l^d)k$；WFKS 的查询条件生成计算也较为简单，是对查询关键字进行相应的模糊集合计算，其计算开销为$(C_l^1+C_l^2+\cdots+C_l^d)$；WFKS 中的模糊查询操作是使用查询关键字的模糊集合对各个文件的索引关键字模糊集合进行遍历，其计算开销为$(C_l^1+C_l^2+\cdots+C_l^d)^2 n$.

(3) TTSS 算法计算功耗分析. TTSS 中索引的生成计算主要包括各个关键字的模糊集合计算以及计算结果中模糊集合各个元素经预定义符号集合元素替换后构建的符号语义树，在关键字之间不存在相同前缀的最坏情形下，其计算开销为$(C_l^1+C_l^2+\cdots+C_l^d)k+(C_l^1+C_l^2+\cdots+C_l^d)kl$；TTSS 中查询条件的计算主要包括查询关键字模糊集合的计算和语义树符号编码的生成，其计算开销为$(C_l^1+C_l^2+\cdots+C_l^d)+(C_l^1+C_l^2+\cdots+C_l^d)l$；TTSS 中模糊查询操作是使用上述步骤中生成的查询关键字符号集合对符号语义树进行遍历再获取最终结果，其计算开销为$(C_l^1+C_l^2+\cdots+C_l^d)kl$.

HB-FKS、WFKS 和 TTSS 的计算复杂度如表 2 所示.

表 2 算法计算复杂度比较

	索引	查询条件	模糊查询
HB-FKS	$O(ldkm+\log_2 k)$	$O(ld\log_2 k)$	$O(ld(\log_2 k+m))$
WFKS[6]	$O(ldk)$	$O(ld)$	$O(l^2d^2n)$
TTSS[12]	$O(ldk(l+1))$	$O(ld(l+1))$	$O(l^2dk)$

从上述分析结果和表 2 可以看出，在索引生成和查询条件生成方面，HB-FKS 并不是计算功耗最低的算法，为了加强数据的安全性和节省存储开销，查询条件和索引关键字需要经过一定程度的预处理再存入布鲁姆过滤器，WFKS 和 TTSS 主要采用枚举集合，因此存储开销更大. 在执行密文模糊查询时，HB-FKS 性能优势得到了体现，Huffman 树的叶子节点深度会根据关键字 TF×IDF 评分的不同进行相应的变化，评分高的关键字叶子深度较浅，反之则较深.

6.2 算法安全性分析

(1) 索引安全. WFKS、TTSS 和 HB-FKS 这 3 种算法在索引构建过程中均采用了基于通配符的模糊集合构造算法以节省存储开销，然而 3 种算法在索引的安全性保护方面采用了不同的方法. WFKS 算法中，索引关键字 w_i 通过常规的对称加密算法进行加密保护，数据拥有者持有的对称密钥 sk_i 对服务器保密，然而以常规的对称加密算法生成的安全索引难以抵御基于关键字频率信息的字典攻击；TTSS 算法在安全性方面没有进行加强，索引关键字的保护机制同样采用了基于共享密钥的对称加密算法. 该算法致力于减少索引关键字重复率较高情形下的存储开销，引入了特殊符号集合 $\Delta=\{\alpha_1,\alpha_2,\cdots\}$ 用于替换索引关键字，以构建基于符号的前缀语义树(The Symbol-based Trie-Tree). 基于特殊符号集合元素替换生成的安全索引并未对关键字频率做相应处理，因此对于字典攻击同样缺乏有效抵御能力；HB-FKS 算法中，文件索引中的关键字如 w_i 的频率信息 sc_{w_i} 会通过基于 TF×IDF 评分的 Huffman 编码 haf_{w_i} 进行转换和隐藏，随后各个索引关键字的模糊集合 $S_{w_{i\_1}},S_{w_{i\_2}},\cdots,S_{w_{i\_d}}$ $(1\leq i\leq k)$ 对应的密文被进一步映射至相应的布鲁姆过滤器 $B_{w_{i\_1}},B_{w_{i\_2}},\cdots,B_{w_{i\_d}}$，以确保索引信息的安全性. 各个布鲁姆过滤器的映射函数是一系列独立同分布哈希函数 $h_1,h_2,\cdots,h_m$，具有理论上的不可逆性，攻击者无法通过哈希值对原始信息进行逆推.

然而在获知部分明文的情形下，攻击者可通过截取查询结果索引的方式累计足够的索引编码样本，随后结合已知明文信息对密文数据进行破解. 通过明文信息破解密文的攻击方式主要分为两类，一类是以密文信息长度的可区分性为基础，另一类则是利用密文信息的频率可区分性. 后者又称为字典攻击. 本文采用的 Huffman 编码具有长度可区分性，但结合布鲁姆过滤器 $B_{w_{i\_1}},B_{w_{i\_2}},\cdots,B_{w_{i\_d}}$ 建立相应索引树后，所有文件索引具有了相同的字节长度，避免了因长度信息导致的安全问题，即能有效抵御第一类攻击；对于第二类攻击，本文的 Huffman 编码树并不是基于关键字内容进行构建的，其计算基础是 TF×IDF 评分信息，对关键字频率信息进行了保护，并通过布鲁姆过滤器各个独立同分布的哈希函数进行映射编码存储. 布鲁姆过滤器引入的随机性和假阳性对关键字频率信息进行了扰动，使其可区分度进一步降低. 为了方便表达，不妨令各个哈希函数的冲突率均为 P_h，布鲁姆过滤器的假阳性为 fp，假定攻击者能在 $O(1)$ 时间内对任意哈希函数进行破解，则特定的索引关键字 w_i 被攻击者获知的概率为$(1-fp)\times P_h^m(1/2^{BL})\times(1/2^{HL})$，其中 HL

为关键字的 Huffman 编码长度.

(2)查询条件安全. WFKS 算法中,用户的查询条件集合元素 $T_{w_i'}$ 生成过程和索引关键字模糊集合的构建方式是一致的,即采用同样的对称加密函数以及共享的对称密钥 sk_i 对查询关键字 w_i' 的模糊集合元素进行相应处理,因此查询条件面临着索引构建过程中同样的安全问题,即字典攻击;TTSS 算法中的查询条件 $T_{w_i'}$ 的生成也同样采用了索引生成中应用的符号替换算法,生成的查询条件可用于对前缀语义索引树进行高效遍历,但相比 WFKS,该算法在安全性方面并未进行提升,因此也同样难以有效地抵御字典攻击;在 HB-FKS 算法中,为了节省计算开销,查询条件的生成和索引关键字模糊集合的计算存在一定的差异. 查询条件包含的关键字 w_q 对应的模糊关键字集合会生成相应的 Huffman 编码 $haf_{w_i}, haf_{w_{i+1}}, \cdots$,并被加密成相应的密文 $c_{w_i}, c_{w_{i+1}}, \cdots$,以确保其内容和频率信息的安全性. 和上述算法加密方式不同在于:为了避免已知明文情形下的攻击,HB-FKS 采用不可逆的加密算法,关键字 w_q 的模糊集合成员被加密成为固定字节长度的密文. 假定攻击者能在 $O(1)$ 时间内对加密算法进行破解,则关键字 w_q 某特定模糊集合成员真实值被攻击者获知的概率为 $(1/2^{HL})\times(1/2^{CL})$,其中 CL 为密文长度(本文采用 MD5 加密函数,其密文长度为 64 位,可扩展替换为密钥更长的强加密算法).

(3)查询计算安全. WFKS 算法中,服务器通过对比查询条件集合 $\{T_{w_i'}\}$ 和索引关键字模糊集合 $\{S_{w_{i\_d}}\}$ 中的相同元素,以查找用户所需的文件数据. 然而只有在攻击者未获取相应频率信息的情形下,算法才能保障查询计算的安全性,此外这种枚举方式构建的索引集合的查找效率较低;TTSS 算法对索引的构建方式有所改进,因此服务器在查找计算过程中可通过基于特殊符号的查询条件 $\{T_{w_i'}\}$ 对树型结构的前缀语义索引 G_w 进行遍历查找. 然而由于安全机制方面仍然沿袭了 WFKS 中对称加密的方法,因此,查询计算的核心仍是密文匹配,对于字典攻击缺乏有效的抵御能力;在查询处理过程中,攻击者可以不断截获查询结果和查询条件的编码以构建样本,并根据密文样本长度和出现频率,结合已知的明文推测密文内容. HB-FKS 算法中,DSP 服务器系统的查询计算包含两个方面,一方面是基于查询条件对 Huffman 编码树进行高效遍历,另一方面是遍历至叶子节点处时,DSP 通过将查询条件映射至布鲁姆过滤器 $B_{w_{i\_1}}, B_{w_{i\_2}}, \cdots, B_{w_{i\_d}}$,以判别查询关键字 w_q 和各个模糊集合 $S_{w_{i\_1}}, S_{w_{i\_2}}, \cdots, S_{w_{i\_d}}$ 之间的隶属关系,最终返回查询结果文件 $cf_1, cf_2, \cdots, cf_n$. 布鲁姆过滤器的采用使得文件索引长度固定,不再具有长度可区分性,而查询过程中传输的最终结果 $cf_1, cf_2, \cdots, cf_n$ 包含了一定数量的假阳性数据,对频率信息进行了扰动. 假定攻击者能在 $O(1)$ 时间内对布鲁姆过滤器的各个哈希函数以及 MD5 加密算法进行破解,则查询条件和索引的明文被攻击者推算出来的概率为 $(1-fp)\times P_h^m(1/2^{BL})\times(1/2^{HL})\times(1/2^{CL})$.

6.3　算法假阳性分析

HB-FKS 采用了布鲁姆过滤器作为 Huffman 索引树的叶子节点,用以节省模糊关键字集合的存储开销,因此,查询结果中难免引入部分假阳性数据. 对于布鲁姆过滤器处理的关键字,需确定其具体长度 l 和布鲁姆过滤器涉及的编辑距离 d,以确定布鲁姆过滤器中存储的元素个数 $n=C_l^1+C_l^2+\cdots+C_l^d$. n 个元素的区分至少需要 $\log_2 n$ 个比特位,即最低的布鲁姆过滤器长度. 然而在实际情况中,为了确保较低的假阳性,布鲁姆过滤器的长度 BL 远远大于该值. 假定布鲁姆过滤器对应的独立哈希函数个数为 m,根据假阳性计算式(2),假阳性 $fp\approx(1-e^{-mn/BL})^m\approx 0.6185^{BL/n}$. 系统通过控制相应参数实现查询结果中较低的假阳性.

6.4　关键参数分析

本小节主要分析布鲁姆过滤器中独立同分布函数数量 m 对 HB-FKS 算法索引生成、存储开销和查询效率 3 个方面的影响.

在索引计算方面,由于关键字模糊集合 $S_{w_{i\_1}}, S_{w_{i\_2}}, \cdots, S_{w_{i\_d}}$ 中的各个元素均需通过布鲁姆过滤器中 m 个独立同分布函数的映射处理,因此,HB-FKS 算法中系统在预处理阶段对索引进行计算时,索引的计算时间和函数数量呈线性关系(参见 6.1 节);系统的存储开销不会受到独立同分布函数数量的影响,这是由于布鲁姆过滤器的长度 BL 是固定值,不会受到存储元素数量的影响,我们在第 7 节实验部分对该结论进行了进一步验证;在查询方面,由于 HB-FKS 中模糊查询操作需要首先对 Huffman 索引树进行遍历,随后在叶子节点处进行布鲁姆过滤器映射操作,因此,独立同分布函数数量在系统对叶子节点进行映射操作时产生影响,叶子节点处的搜索时间和函数数量呈线性关系,但由于 HB-FKS 采用了 Huffman 索引树,结合树型结构在搜索效率方面的优势,从而能够降低多次映射带来的影响.

事实上,为了使保证的假阳性达到最小,独立同分布哈希映射函数的个数 m 需根据以下计算公式进行计算[37],即 $m=(BL/n)\times\ln 2$,其中 BL 为布鲁姆过滤器长度,n 为插入布鲁姆过滤器的元素个数.

7 实 验

为了验证 HB-FKS 查询算法的查询效率,本文将采用 IEEE 标准数据库提供的 2005～2015 年 INFOCOM,MOBICOM 等 8 个会议论文集作为测试数据(包含 50000 个 PDF 格式文件和 50 个高频关键字,关键字最大长度为 16 个字符,最小长度为 3 个字符,平均长度 7.22 字符,编辑距离最大为 3),以内存 2G,CPU 主频 2.10GHz 的 PC 机作为测试平台,对 HB-FKS 算法在索引生成、查询条件生成和查询操作的计算时间和存储开销进行仿真和分析,并和 WFKS、TTSS 算法进行比较,具体测试结果如下.

7.1 索 引

(1)生成时间. 图 9 给出了 HB-FKS,WFKS 和 TTSS 这 3 种算法在 DSP 存储文件数量变化情形下的索引生成时间. 从仿真结果可以观察到,WFKS 索引生成时间略高于 TTSS;HB-FKS 的索引生成时间最长,约为 TTSS 生成时间的两倍. 这是由于在生成索引关键字的模糊集合后,TTSS 不再改变索引结构,而 WFKS 和 HB-FKS 均需要额外的计算时间对模糊关键字集合进行重新组织:WFKS 将模糊关键字集合向预定义字符集合进行映射后重新构建基于共同前缀的语义树,而 HB-FKS 在构建基于关键字 TF×IDF 评分的 Huffman 树的同时,需要在 Huffman 树叶子节点处对关键字集合进行布鲁姆过滤器映射处理,存储不同编辑距离下的关键字,额外增加了哈希计算时间.

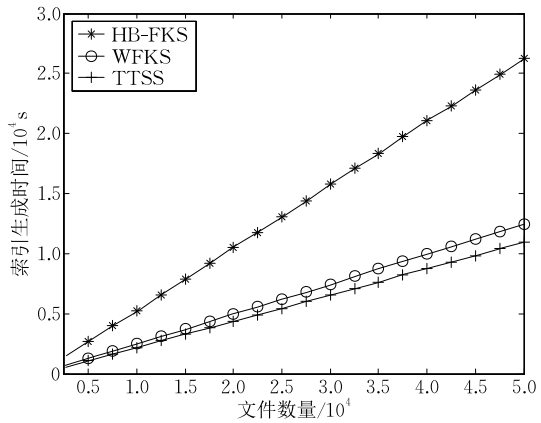


图 9 索引生成时间

尽管 HB-FKS 索引的生成计算时间更长,但索引和相关关键字的计算可通过线下预处理生成,并不会对查询系统性能产生影响;另外,相比 TTSS 而言,HB-FKS 和 WFKS 索引的存储开销更小,安全性更高.

(2)存储空间. 图 10 是 HB-FKS,WFKS 和 TTSS 这 3 种算法在 DSP 存储文件数量从 2500 变化至 50000 情形下的索引存储空间. 从实验仿真结果可以观察到,相比 TTSS 的索引存储开销,WFKS 有了一定程度的提升;HB-FKS 则在索引存储开销方面占有较大优势,比 TTSS 低 96.46%,比 WFKS 低 94.66%. 这是由于一方面 Huffman 编码(又称为最优编码)的采用降低了树型结构的高度和关键字长度,另一方面布鲁姆过滤器的使用实现了采用固定字长比特位表示不同编辑距离下的模糊集合,在很大程度上降低了模糊集合引入的大量存储开销.

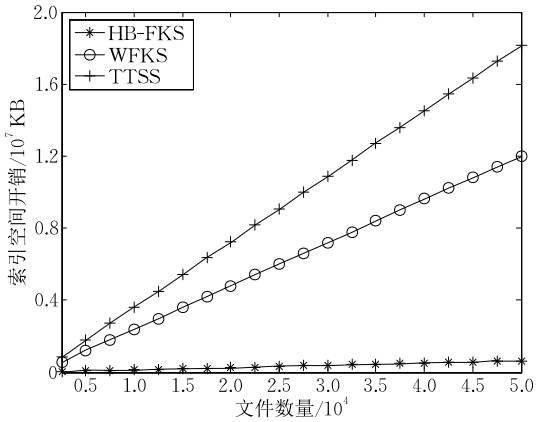


图 10 索引存储空间

7.2 查询条件

(1)生成时间. 图 11 给出了 HB-FKS、WFKS 和 TTSS 这 3 种算法在 DU 查询请求数量从 2 变化至 40 情形下的查询条件生成时间(查询条件包含关键字长度最长为 16 字符,最短为 3 字符). 实验仿真结果表明,HB-FKS 的查询条件生成时间最长,是 TTSS 的两倍左右;WFKS 的查询条件生成时间略高于 TTSS. 事实上,在大部分基于对称加密的查询算法中,查询条件的计算处理过程和索引的计算处理过程通常是类似的. 上述 3 种算法均是基于对称加密算法实现安全查询的,因此查询条件的生成时间也会受到索引生成计算中相同计算因素的影响. 在 WFKS 中,影响查询条件生成时间的主要因素仍然是前缀语义树的生成,而在 HB-FKS 中,查询条件计算时间则主要受到 Huffman 树构建和布鲁姆

过滤器映射的影响，布鲁姆过滤器哈希映射带来的计算开销影响相对 Huffman 树更大.

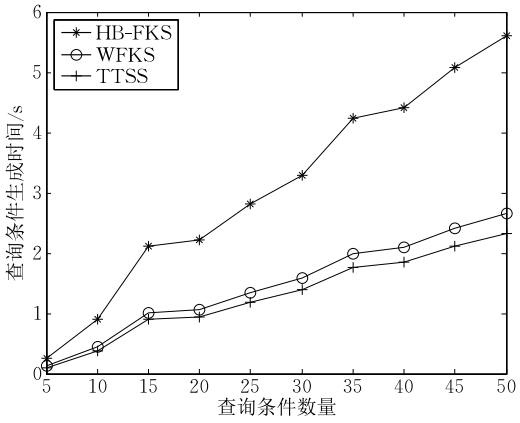


图 11　查询条件生成时间

(2)存储空间. 图 12 是 HB-FKS、WFKS 和 TTSS 这 3 种算法在不同数量 DU 查询请求情形下的查询条件存储空间大小. 在这 3 种基于对称加密的查询算法中，查询条件的存储方式和索引存储方式是类似的. 即 TTSS 不对查询关键字的模糊集合做进一步处理；WFKS 通过预定义符号集合构建语义树，前缀相同的模糊集合元素可共用部分存储空间，在一定程度上降低了存储开销；HB-FKS 则从降低树型结构高度和简化模糊集合表达方式两个方面节省存储开销，采用 Huffman 编码和布鲁姆过滤器实现这一目的. 从仿真结果中可以看到，查询请求数量越少，3 种算法的存储开销差异就越小，树型结构和布鲁姆过滤器带来的存储优势越不明显.

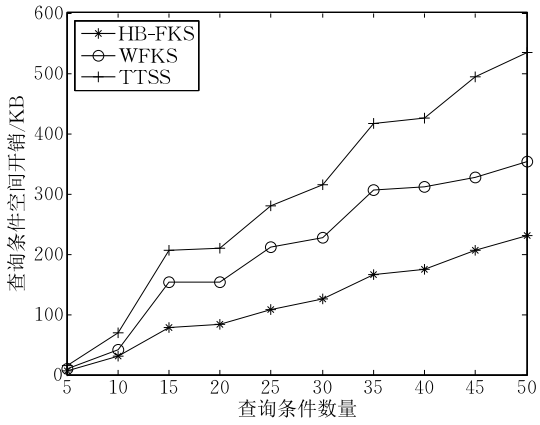
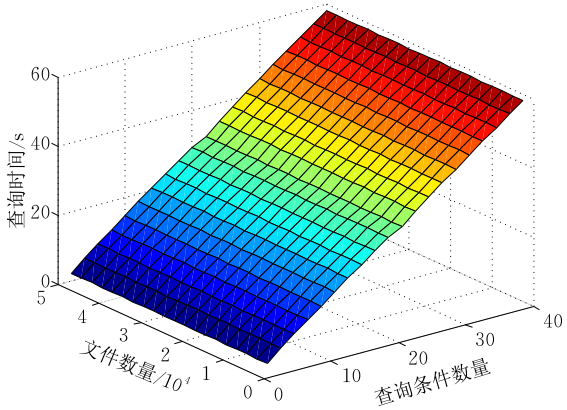


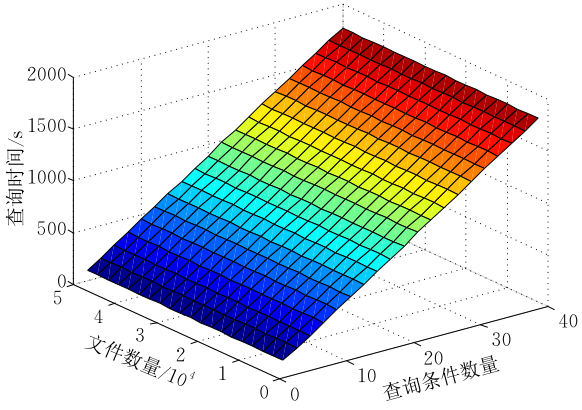
图 12　查询条件存储空间

7.3　模糊查询

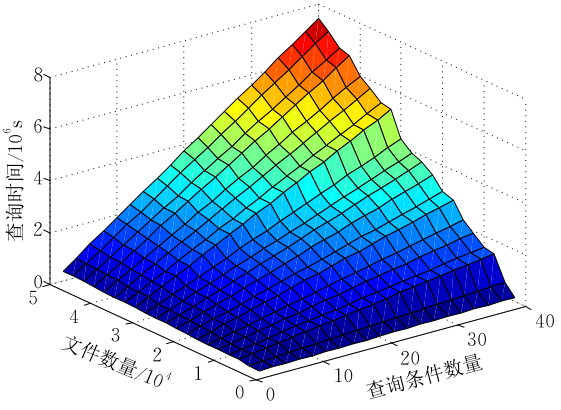
图 13(a)～(c)分别给出了 HB-FKS、WFKS 和 TTSS 这 3 种算法在 DSP 存储文件数量和 DU 查询



(a) HB-FKS



(b) WFKS



(c) TTSS

图 13　模糊查询时间

请求数量变化的情形下所需的查询时间.采用了树型结构的 HB-FKS 和 WFKS 具有较高的查询效率,分别比 TTSS 低 99.96%和 98.82%,其中 HB-FKS 的查询效率更高,比 WFKS 低 96.74%.这是由于相对 TTSS 中基于模糊集合枚举方式的查询算法,树型结构具有显著的查询效率优势,在 DU 和 DSP 预约定关键字数量固定的情形下,树型结构可将查询效率的线性增长变为对数级增长.相比 WFKS 基于共同前缀的符号语义树方法,HB-FKS 的 Huffman 树的高度更低,查询速度更快,且在数据频率保护方面的安全性更高.仿真结果中,TTSS 的查询效率对文件数量的影响较为敏感,而在 WFKS 和 HB-FKS 中,查询效率受文件数量的影响较小.这是由于上述两种算法中的树型结构相对固定,高度只和事先约定的关键字数量相关,文件索引均分类放置在叶子节点,相比查询操作,索引和查询条件的生成受文件数量的影响较大.

8 结 论

DaaS 模式是大数据时代主要的数据存储模式,密文查询是其提供的重要服务之一.精确密文数据查询难以容忍用户查询过程中的输入或格式误差,模糊密文查询作为功能补充实现了对非精确查询条件的兼容.针对现有模糊查询算法在安全性、查询效率和存储功耗方面的不足,本文提出了一种基于 Huffman 编码和布鲁姆过滤器的模糊查询机制 HB-FKS.该机制基于关键字的 TF×IDF 评分规则构建 Huffman 树并进行相应编码,实现对数据高效查找的同时避免其频率信息的泄露.此外,本文采用了布鲁姆过滤器对不同编辑距离下的模糊关键字集合进行集成存储,降低了模糊集合带来的额外存储开销.算法分析和实验结果表明,HB-FKS 算法在索引生成和查询条件生成时,由于需要执行布鲁姆过滤器映射计算,算法执行时间比现有的 WFKS 和 TTSS 算法更长,但在安全性能和存储开销方面更具优势.在执行查询算法时,布鲁姆过滤器和 Huffman 编码带来的优势得到了充分体现,相比现有工作,HB-FKS 实现了高效率模糊查询.

参 考 文 献

[1] Xian He-Qun, Feng Deng-Guo. An integrity checking scheme in outsourced database model. Journal of Computer Research and Development, 2010, 47(6): 1107-1115 (in Chinese)
(咸鹤群,冯登国. 外包数据库模型中的完整性检测方案. 计算机研究与发展, 2010, 47(6): 1107-1115)

[2] Tian X, Sha C, Wang X, et al. Privacy preserving query processing on secret share based data storage//Proceedings of the Database Systems for Advanced Applications. Hong Kong, China, 2011: 108-122

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

[4] Goh E. Secure indexes. Cryptology ePrint Archive, 2003 (2003): 216-235

[5] Tian Xiu-Xia, Wang Xiao-Ling, Gao Ming, et al. Database as a service — security and privacy preserving. Journal of Software, 2010, 21(5): 991-1006(in Chinese)
(田秀霞,王晓玲,高明等. 数据库服务——安全与隐私保护. 软件学报, 2010, 21(5): 991-1006)

[6] 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

[7] Katz J, Sahai A, Waters B. Predicate encryption supporting disjunctions, polynomial equations, and inner products//Proceedings of the EUROCRYPT. Istanbul, Turkey, 2008: 146-162

[8] Lewko A, Okamoto T, Sahai A, et al. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption//Proceedings of the EUROCRYPT. Riviera, French, 2010: 62-91

[9] Song Wei, Peng Zhi-Yong, Cheng Fang-Quan, et al. A service-oriented adaptive search method over encrypted data in trusted database. Chinese Journal of Computers, 2010, 33(8): 1324-1338(in Chinese)
(宋伟,彭智勇,程芳权等. 可信数据库环境下面向服务的自适应密文数据查询方法. 计算机学报, 2010, 33(8): 1324-1338)

[10] 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

[11] Cao Ning, Wang Cong, Li Ming, et al. Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(1): 222-233

[12] 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

[13] Thakur J, Kumar N. DES, AES and Blowfish: Symmetric key cryptography algorithms simulation based performance analysis. International Journal of Emerging Technology and Advanced Engineering, 2011, 1(2): 6-12

[14] Jonsson J, Kaliski B. Public-key cryptography standards (PKCS) #1: RSA cryptography specifications version 2.1, 2003

[15] Shamir A. How to share a secret. Communications of the ACM, 1979, 22(11): 612-613

[16] Rafaeli S, Hutchison D. A survey of key management for secure group communication. ACM Computing Surveys, 2003, 35(3): 309-329

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

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

[19] Abdalla M, Bellare M, Catalano D, et al. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions//Proceedings of the Annual International Cryptology Conference (CRYPTO). Santa Barbara, USA, 2005: 205-222

[20] Park H, Kim B H, Lee D H, et al. Secure similarity search//Proceedings of the IEEE International Conference on Granular Computing. San Jose, USA, 2007: 598-598

[21] Chor B, Kushilevitz E, Goldreich O, et al. Private information retrieval. Journal of the ACM, 1998, 45(6): 965-981

[22] Kushilevitz E, Ostrovsky R. Replication is not needed: Single database, computationally-private information retrieval //Proceedings of the IEEE Symposium on Foundations of Computer Science. Miami Beach, USA, 1997: 364

[23] Wang S, Agrawal D, El Abbadi A. Generalizing pir for practical private retrieval of public data//Proceedings of the Data and Applications Security and Privacy. Rome, Italy, 2010: 1-16

[24] Papadopoulos S, Bakiras S, Papadias D. pCloud: A distributed system for practical PIR. IEEE Transactions on Dependable and Secure Computing, 2012, 9(1): 115-127

[25] Aizawa A. An information-theoretic perspective of TF-IDF measures. Information Processing & Management, 2003, 39(1): 45-65

[26] Ramos J. Using TF-IDF to determine Word relevance in document queries//Proceedings of the 1st Instructional Conference on Machine Learning. 2003: 1-4

[27] Zhang W, Yoshida T, Tang X. A comparative study of TF*IDF, LSI and multi-words for text classification. Expert Systems with Applications, 2011, 38(3): 2758-2765

[28] Song H, Dharmapurikar S, Turner J, et al. Fast hash table lookup using extended Bloom filter: An aid to network processing. ACM SIGCOMM Computer Communication Review, 2005, 35(4): 181-192

[29] Broder A, Mitzenmacher M. Network applications of Bloom filters: A survey. Internet Mathematics, 2004, 1(4): 485-509

[30] Hashemian R. Memory efficient and high-speed search Huffman coding. IEEE Transactions on Communications, 1995, 43(10): 2576-2581

[31] Aggarwal M, Narayan A. Efficient huffman decoding//Proceedings of the IEEE International Conference on Image Processing. Vancouver, Canada, 2000: 936-939

[32] Kavousianos X, Kalligeros E, Nikolos D. Optimal selective Huffman coding for test-data compression. IEEE Transactions on Computers, 2007, 56(8): 1146-1152

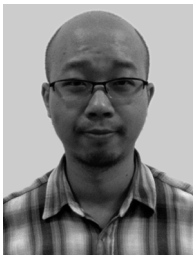
[33] Alzain M A, Pardede E. Using multi shares for ensuring privacy in Database-as-a-Service//Proceedings of the International Conference on System Sciences (HICSS). Koloa, USA, 2011: 1-9

[34] Frank M C, Goldwater S, Griffiths T L, et al. Modeling human performance in statistical word segmentation. Cognition, 2010, 117(2): 107-125

[35] Levenshtein V. Binary codes capable of correcting spurious insertions and deletions of ones. Problems of Information Transmission, 1965, 1(1): 8-17

[36] Tarkoma S, Rothenberg C E, Lagerspetz E. Theory and practice of Bloom filters for distributed systems. IEEE Communications Surveys & Tutorials, 2012, 14(1): 131-155

[37] Bonomi F, Mitzenmacher M, Panigrah R, et al. Beyond Bloom filters: From approximate membership checks to approximate state machines//Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM). Pisa, Italy, 2006: 315-326



LI Jin-Guo, born in 1985, Ph. D., lecturer. His current research interests include network security, applied cryptography.

TIAN Xiu-Xia, born in 1976, Ph. D., professor. Her current research interests include database security, security and privacy preserving for big data and cloud computing, applied cryptography.

ZHOU Ao-Ying, born in 1965, professor, Ph.D. supervisor. His current research interests focus on data management and information system, including Web data management, Chinese Web infrastructure, Web searching and mining, complex event processing and real-time business intelligence, uncertain data management and applications, data-intensive computing, distributed storage and computing, peer to peer computing and management, Web service.

Background

Privacy preserving fuzzy keyword search is an important problem in Database as a Service paradigm (DaaS), which has not gained much attentions of researchers. There are some research works concerning about secure searching over encrypted database outsourced by data owners, but most of them are search techniques based on accurate keyword matching. A few of these works support fuzzy keyword search, while they also suffers from the limitations in efficiency, storage and security during the query process. To resolve these problems efficiently, a privacy preserving fuzzy keyword search mechanism called HB-FKS for DaaS is proposed in this paper. The HB-FKS combines Huffman codes with Bloom filters seamlessly to construct the index tree for encrypted files stored in Database Service Provider (DSP). The Huffman codes can provide high search efficiency by generating a tree-based data structure, while the Bloom filters can save the storage space by hashing each element in the keyword similarity set to a few binary bits. Finally, we analyze the algorithm complexity, security and false positive of HB-FKS, and verify the efficiency of HB-FKS by performing experiments on real data.

This paper is supported by the National Basic Research Program (973 Program) of China (No. 2010CB328106), the National Natural Science Foundation of China (Nos. 61202020, 61370226), the Natural Science Foundation of Shanghai (No. 12ZR1411900), the Startup Fund for Talent Introduction of Shanghai University of Electric Power (No. K2015-008). The research aims of these projects include the privacy preserving issues of data users and owners, access control mechanisms, efficient search techniques on encrypted outsourced database.