计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2020.20190233
57(1):3-16,2020

一种灵活的精度可控的可搜索对称加密方案

李西明[1] 陶汝裕[1] 粟 晨[1] 黄 琼[1] 黄欣沂[2]
[1](华南农业大学数学与信息学院 广州 510642)
[2](福建师范大学数学与信息学院 福州 350117)
(liximing@scau.edu.cn)

A Flexible Accuracy-Controllable Searchable Symmetric Encryption Scheme

Li Ximing[1], Tao Ruyu[1], Su Chen[1], Huang Qiong[1], and Huang Xinyi[2]
[1](*College of Mathematics & Informatics, South China Agricultural University, Guangzhou* 510642)
[2](*College of Mathematics & Informatics, Fujian Normal University, Fuzhou* 350117)

Abstract In the traditional keyword-based searchable symmetric encryption technology, the keyword set is usually generated by the keyword extraction algorithm, so that the content and quantity of the keywords are limited by the keyword extraction algorithm. Therefore, in the keyword-based encryption search system, in addition to the keyword set generated by the system at the time of initial construction, the user cannot search for other related contents, thereby limiting the application of the encryption search technology. In view of the above problems, this paper proposes a flexible accuracy-controllable searchable symmetric encryption (FASSE) that supports flexible and precise control. By flexibly generating keywords and indexes generated by document summary during system operation, the dependence of the keyword collection effectively improves the flexibility of the encryption search technology. FASSE provides three basic searches, namely one-shot search reinforcement search and filter search, which respectively correspond to the user finding the keyword record in the dictionary only once, not finding the keyword record in the dictionary and only using it once. The search finds records or three search cases where keyword records are found in the dictionary and abstract multiple times. At the same time, the system also combines three kinds of search to design a fuzzy reinforcement search to further enhance the practicability of the system. The specific implementation language of the FASSE program is the Java programming development language and the final experiment shows that FASSE averages 114.26 ms in searching each paper in the https://eprint.iacr.org/complete/ paper dataset.

Key words searchable symmetric encryption (SSE); suffix array (SA); LF mapping; SubLF mapping; FM-index; Burrows Wheeler transform (BWT)

收稿日期:2019-04-11;**修回日期**:2019-11-28

基金项目:国家自然科学基金优秀青年科学基金项目(61822202);国家自然科学基金项目(61872152,61872409);广东省自然科学基金杰出青年基金项目(2014A030306021);广东省特支计划科技创新青年拔尖人才项目(2015TQ01X796);广东省基础与应用基础研究重大项目(2019B030302008)

This work was supported by the National Natural Science Foundation of China for Excellent Young Scientists (61822202), the National Natural Science Foundation of China (61872152, 61872409), the Guangdong Natural Science Funds for Distinguished Young Scholar (2014A030306021), the Guangdong Program for Special Support of Top-notch Young Professionals (2015TQ01X796), and the Major Program of Guangdong Basic and Applied Research (2019B030302008).

通信作者:黄琼(qhuang@scau.edu.cn)

摘 要 在传统基于关键词集合的可搜索对称加密技术中,关键词集合通常由关键词提取算法生成,使得其关键词的内容和数量受到关键词提取算法的限制.因此,在基于关键词的加密搜索系统中,除了初始化系统生成的关键词集,用户无法搜索其他相关内容导致限制了加密搜索技术的应用.针对以上问题,提出了支持灵活精度可控的可搜索对称加密方案(flexible accuracy-controllable searchable symmetric encryption, FASSE),通过在系统运行过程中灵活地生成关键词和文档摘要生成的索引,从而减少对关键词集合的依赖,从而有效提高了加密搜索技术灵活性.FASSE 提供 3 种基本搜索,分别是一次命中搜索、增强搜索和过滤搜索,它们分别对应着用户只用一次就在字典中找到关键词记录、没有在字典中找到关键词记录而只用一次就在摘要中找到记录或者多次在字典和摘要中查找到关键词记录的这 3 种搜索情况.同时,系统也结合 3 种搜索设计了一种模糊增强搜索进一步增强系统的实用性.FASSE 方案的具体实现语言是 Java 编程开发语言,并且最终实验得出 FASSE 在 https://eprint.iacr.org/complete/的论文数据集中平均搜索完每一篇论文的时间为 114.26 ms.

关键词 可搜索对称加密;后缀数组;LF 映射;SubLF 映射;FM 索引;BWT 转换

中图法分类号 TP391

随着云计算的快速发展,数据外包在数据存储市场的应用也越来越广泛.通常情况下,用户会选择将数据外包给云服务器存储,同时又不公开数据.为了保护数据和隐私,数据拥有者会先选择加密数据,然后将加密数据上传到云端.然而当用户搜索关键词获得了一系列包含关键词的文档都太大时并且用户只想查询这些文档中的一部分内容时,检索就显得十分麻烦,例如在线 DNA 测序等.最简单的解决方法是将所有加密文档下载到客户端解密并检索.但是,这不仅会造成巨大的网络开销,还会带来不必要的计算成本.如果服务器不可信,那么用户数据的安全性就很难得到保护.因此,可搜索的对称加密方案(searchable symmetric encryption, SSE)迅速成为研究者们研究的热点.

关键词搜索技术通常分为精确关键词搜索和模糊关键词搜索.但是目前精确关键词搜索存在很大的问题.当用户描述不准确或输入错误时,精确关键词搜索可能无法找到用户真正需要的文件.因为模糊搜索技术的引入可以灵活地找到与用户输入的搜索词相关的文件,所以模糊关键词搜索技术迅速发展了起来.但是由于数据是以密文形式存储的,因此明文中使用的模糊搜索方法不能直接应用于密文搜索.从而模糊可搜索的加密机制是研究者重点研究的方向之一.

1 相关工作

近年来,研究人员针对不同的问题提出了许多可搜索的对称加密方案(SSE).Popa 等人[1]在 2011 年 SOSP 会议上首次提出 CryptDB,它可以在加密文本上执行搜索操作,如 MySQL 中的 LIKE 等操作,并且提供了很好的安全性保证.陈萍等人[2]首次提出了可以在密文上执行所有 SQL 语句的 CryptDB 系统.CryptDB 是以第三方中间代理人的形式来对数据库提供安全高效的加密和解密方案.其主要核心分为 3 个模块:加密策略、洋葱加密模型和密钥链管理模块.但 CryptDB 存在着一些明显的缺陷,例如,当前的同态加密只支持整数型运算不支持浮点型运算、不能做到对原始数据库不作任何修改.在文献[3]中汪海伟等人提出一个可搜索数据库加密系统,解决了传统的加密数据库无法在保证数据安全性的同时实现执行复杂的 SQL 语句并解决了文献[1]中的缺陷.Cash 等人提出了一种针对不同数据库规模的安全有效的数据处理方案[4],特别是对于大型数据库,它可以高效并秘密地搜索具有数百亿记录-关键词对的加密的服务器数据库.文献[4]的基本构造方案支持单个关键词搜索,并提供渐近优化的服务器索引大小、完全并行的搜索和最小的泄露,同时其所有方案和优化都被证明是安全的,并且泄露给不可信服务器的信息被精确地量化.

一方面可搜索加密方案[1-4]侧重于解决密文数据库存储以及查询的问题.另一方面在密文文本中搜索实现文本文件的可搜索加密技术也是当下研究的热点.当服务器返回许多文档到本地时,用户无法将每个匹配了关键词的文档都下载到客户端以确定哪些文档是需要的,尤其是查询结果中存在许多大型文档时.这个问题是通常的 SSE 所不能解决的.此时子串可搜索对称加密成为必要手段.在文献[5]中

Faber 为了支持子串查询需要构建重叠 k-gram 的索引,但是这个方案需要大量的存储开销来存储所有的 k-gram,这显然是不现实的.Chase 等人在文献[6]中使用后缀树来构建搜索框架.相比之下,后缀树的数据结构没有固定的大小,最多达 $2n$ 个节点,每个节点存储其边、父节点和子节点的信息,从而增加了存储需求.

传统的精确关键词加密搜索是不能容忍任何的拼写错误,否则将返回错误的结果,这显然不能满足用户的需求.Hu 等人通过使用通配符来构建模糊关键词集的方法来实现加密搜索[7].Zhou 等人使用了编辑距离(edit distance)来设计加密搜索方案[8],通过保存与关键词在一定编辑距离内的所有单词来缓解这种情况.虽然 Zhou 等人的这种方法可以实现对关键词的模糊搜索,但是需要的存储开销较大并且最终得到的搜索结果中存在大量的弱相关性文档从而导致计算开销和网络开销的增加.Zhang 等人提出一种高效的模糊关键词集构建方法[9],它是基于 gram 来构建模糊关键词集,达到了较好的效果.

本文提出了一种基于 Leontiadis 等人[10]的灵活精度可控的可搜索对称加密方案(flexible accuracy-controllable searchable symmetric encryption, FASSE),FASSE 方案支持灵活精度可控的加密搜索并且节省了大量的存储开销.FASSE 考虑使用 Leontiadis 的方案为底层框架主要是因为其基于后缀数组的索引设计,其所允许的最佳存储成本是 $O(n)$,且在大小为 n 的字符串中具有小的隐藏因子.

相比许多基于关键词集查询的 SSE 方案,本文方案在 4 个方面有提高:

1) FASSE 方案允许动态发出可变大小的子字符串查询,而无需事先定义固定的查询大小.

2) FASSE 方案通过搜索加密的文档摘要的 FM 索引(abstract FM index),即 AFM 索引,解决了传统的基于关键词集查询的 SSE 方案中可能会由于关键词集创建的不友好从而导致没有在关键词集中查到任何记录的问题;FM 索引的结构和使用详见第 2 节.

3) FASSE 方案通过一次命中搜索、增强搜索和过滤搜索这 3 种搜索方法解决了在基于关键词集合查询的 SSE 中用户得到许多低相关性文档的问题.

4) 为了提升搜索的实用性,FASSE 方案通过调节在模糊搜索中的一些参数实现了灵活搜索精度可控的模糊增强搜索.

2 预备知识

2.1 BWT 数据转换算法

Burrows Wheeler Transform(BWT)转换[11]将原始文本数据转换为一个相似的文本,转换后使得相同的字符位置连续或者相邻,经过 BWT 操作之后的文本数据可以使用其他编码技术如 Move-to-front transform 和游程编码进行文本压缩.

BWT 有 4 个主要步骤:

1) 在原始字符串 $S=$“banana”的末尾添加一个‘$’字符,令 $S=$“banana$”;

2) 字符串 S 循环向左旋转 n 次得到矩阵 $\boldsymbol{W}'$;

3) 按字典顺序排列矩阵的每一行字符串以获得新的矩阵 $\boldsymbol{W}$;

4) 分别取矩阵 $\boldsymbol{W}$ 的第 1 列和最后 1 列,并将它们定义为 F 和 L 列,把 L 列作为 $BWT(S)$的结果.

算法 1 给出了 BWT 转换的具体描述.值得注意的是 BWT 中有一个十分重要的属性就是第 i 次出现在 F 列中的字符 x 是对应于第 i 次出现在 L 列中的字符 x,Burrows 和 Wheeler 在文献[11]中给出了证明.

算法 1. BWT 转换算法.

输入:字符串 S;

输出:转换结果矩阵 $\boldsymbol{W}$.

$l=S.length()+1$;

$S.append(‘\$’)$;

$i=0$;

char $W'[l][l]$;

S 往左旋转 l 次,每次的结果保存在 $\boldsymbol{W}'$ 的第 i 行中;

while $i<l$ do

  $W'[i++]=rotate(S,i)$;

  将矩阵 $\boldsymbol{W}'$ 的每一行按照字典顺序重新排列最终得到 $\boldsymbol{W}$;

end while

return $\boldsymbol{W}=$Sorted $\boldsymbol{W}'$.

2.2 后缀和后缀数组

字符串 S 的长度为 n,那么 S 的后缀是 $suffix[i]=S[i,\cdots,n-1]$.字符串 $S=$“banana”,那么字符串 S 的后缀 $suffix[i]=\{$“banana”,“anana”,“nana”,“ana”,“na”,“a”$\}$.后缀数组 SA 是字符串 S 所有后缀按照字典顺序排序后,每个后缀对应于 S 中的位置的数组.S 的后缀数组是 $SA[i]=\{5,3,1,$

0,4,2}.通常计算后缀数组 SA 的效率都比较低.例如朴素暴力穷举算法的时间复杂度为 $O(n^2\log n)$,它把字符串所有的后缀全部都计算出来后依次去排序最终得到后缀数组 SA,这种方法的效率是十分低效的.倍增法[12]的时间复杂度为 $O(n\log n)$,它是基于基数排序思想去计算后缀数组 SA 的.虽然倍增法已经极大地缩短了计算后缀数组 SA 的时间,但是相对于在一个线性时间复杂度 $O(n)$下快速计算后缀数组 SA 的算法,倍增法就失去了优势.Sanders 等人提供了一个非常有效的 skew 算法[13].skew 算法是根据原始字符串 S 的所有后缀的位置 pos 递归地将其分成 3 组:$pos \bmod h\{1,2,3\}$,最后合并结果求出后缀数组 SA.

2.3 LF 映射

LF 映射[11]是通过 $BWT(S)$后得到矩阵 $\boldsymbol{W}$ 中的 F 列字符串和 L 列字符串多次反复迭代最终恢复原始字符串 S 的过程.它等同于在原始字符串 S 中搜索子字符串 T,其中 $S=T$.LF 映射如算法 2 所示.

算法 2. LF 映射算法.

输入:矩阵第 1 列 F、矩阵最后 1 列 L;

输出:原始字符串 S.

```
D=0;
i=0;
执行算法 1 得到转换结果矩阵 W;
判断 W 的第 F 列和第 L 行;
在 F 列中查找排名等于 r 的 L[i]的位置;
while L[i]≠'$' do
  D.push(L[i]);
  r=rank(L[i]);/*L[i]在 L 列中的排名*/
  i=find(F[L[i]],r);
end while
while D≠'0' do
  S=S+D.POP;
end while
return S.
```

然而,实际上方案可以通过 SubLF 映射来部分恢复原始字符串 S 从而达到实现子串匹配的目的.它等同于在原始字符串 S 中搜索子字符串 T,其中 $S\neq T$.这也是搜索子字符串的重要手段.SubLF 映射的具体步骤如算法 3 所示.

算法 3. SubLF 映射算法.

输入:矩阵第 1 列 F、矩阵最后 1 列 L、字符串 m;

输出:字符串 m 在串 S 中的位置 pos.

```
l=m.length();
flag=0;
num=rank_max(m[l-1]);/*查找 m[l-1]的最大排名*/
for i=0 to num do
  for j=0 to l do
    if a≠⊥
      a=find(F[m[l-j-1]],i);
    end if
      /*查找 m[l-j-1]在 F 中的位置*/
    if a==⊥ or L[a]≠m[l-j-2]
      /*如果 flag=0 匹配出错,程序结束*/
      flag=0,pos=null;
    end if
    return pos;
  end for
end for
flag=1;
r=rank(L[a]);
a=find(F[L[a]],r);
if flag==1
  pos.append(a);
end if
return pos.
```

2.4 FM 索引

FM 索引是由 3 列数组组成.FM 索引的第 1 列 F 列是矩阵 $\boldsymbol{W}$ 的第 1 列,第 2 列 L 列是矩阵 $\boldsymbol{W}$ 的最后 1 列,最后 1 列是原始字符串 S 的后缀数组 SA.为了支持 LF 映射,F 列和 L 列中每个字符的唯一排名也都需要存储在 r_{F} 和 r_{L} 中,其中 r_{F} 和 r_{L} 均来自于 rank$\langle r_{\mathrm{F}},r_{\mathrm{L}}\rangle$二元组.最终通过 LF 映射利用公式[14] $BWT(S)[i]=L[i]=S[SA[i]]$,可以计算出后缀数组 SA 来完成子串查询.FASSE 方案利用文献[10]中加密的 FM 索引的方法来构造整个方案.其中 FASSE 方案中的加密方法包括对称加密方案 $SKE=\{Gen,Enc,Dec\}$和轻量级加密原语:伪随机函数 $PRFF_{k_{\mathrm{f}}}(\cdot)$,伪随机置换 $PRP\Pi_{k_\pi}(\cdot)$.

加密的 FM 索引结构如图 1 所示.其中 L 列和 F 列的值为 $F_{k_{\mathrm{f}}}(c_{L_j})\oplus F_{k_{\mathrm{f}}}(r_{F_j}\parallel c_{F_j})\parallel F_{k_{\mathrm{f}}}(r_{L_j}\parallel c_{L_j})$和 $F_{k_{\mathrm{f}}}(c_{F_j})\oplus F_{k_{\mathrm{f}}}(r_{F_j}\parallel c_{F_j})$.后缀数组 SA 为 $SKE.Enc(k_{\mathrm{e}},SA[j])$,最后执行 $PRP\Pi_{k_\pi}(FM)$打乱整个 FM 索引的顺序.其中 $k_{\mathrm{f}},k_{\mathrm{l}}$ 是伪随机函数密钥;k_π 是伪随机置换密钥;k_{e} 是对称加密密钥.加密的 FM 索引执行 LF 映射算法会比明文 FM 索引执行 LF 映射算法要复杂.具体做法是:先生成字符 c 的搜索令牌 $F_{k_{\mathrm{f}}}(c)$,计算 $F_{k_{\mathrm{f}}}(c)\oplus F_{k_{\mathrm{l}}}(c)$,通过查找对应位置的 LLset 异或解密相应的 LL 得到字

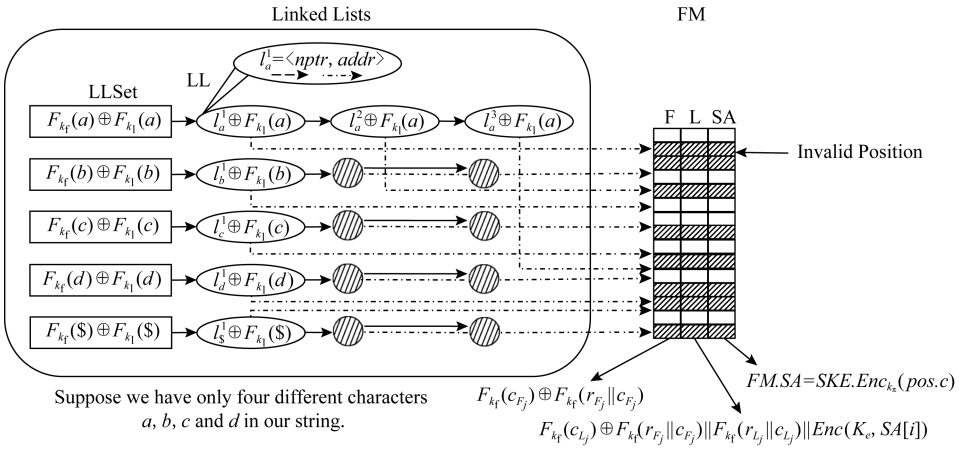


Fig. 1 Construction of an encrypted AFM

图 1 加密的 AFM 的构建

符 c 对应的二元组 $l_c=\langle nptr, addr\rangle$,其中 $nptr$ 是下一个 l_c 的位置,$addr$ 是字符 c 在 F 列中的位置.在 $F[addr]$上执行 $F_{k_f}(c)\oplus F_{k_f}(c_{F_j})\oplus F_{k_f}(r_{F_j}\|c_{F_j})$异或运算解密得到 $F_{k_f}(r_{F_j}\|c_{F_j})$并且得到的 $F_{k_f}(r_{F_j}\|c_{F_j})$可以在 L 列中执行 $F_{k_f}(r_{F_j}\|c_{F_j})\oplus F_{k_f}(c_{L_j})\oplus F_{k_f}(r_{F_j}\|c_{F_j})$异或解密得到 $F_{k_f}(c_{L_j})$,计算 $F_{k_f}(c_{L_j})\oplus F_{k_f}(r_{L_j}\|c_{L_j})$,继续比对 F 列中的 $F_{k_f}(c_{F_j})\oplus F_{k_f}(r_{F_j}\|c_{F_j})$,经过多轮循环后得到搜索结果,即加密的 $SA[j]$.最终使用 k_e 解密得到真正的 $SA[j]$.

3 定义 FASSE 方案

3.1 方案系统模型

如图 2 所示,FASSE 方案系统模型包含 3 个实体:云服务器、授权用户和数据拥有者.数据拥有者

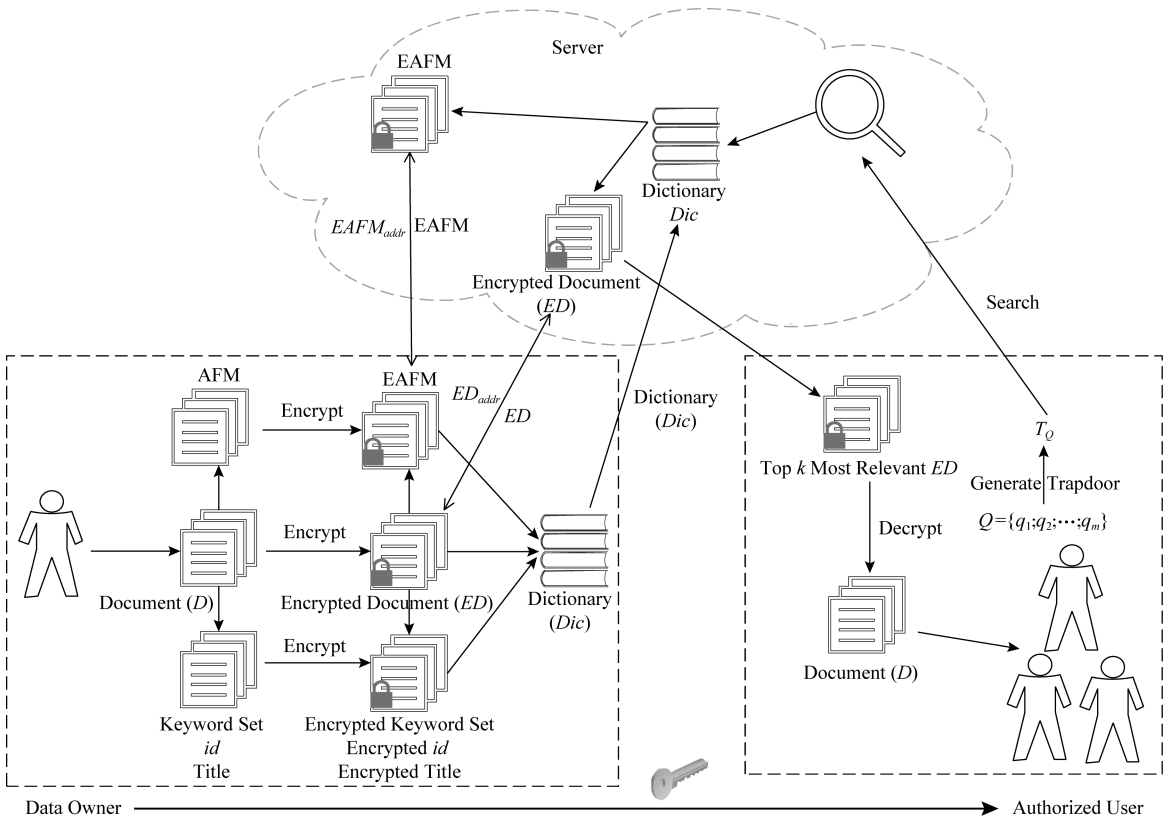


Fig. 2 FASSE system model

图 2 FASSE 方案系统模型

使用任意关键词提取算法对文档集合 D 提取的关键词 KW.数据拥有者为每一个文档生成唯一文档标示符 id.加密的唯一文档 id 得到唯一加密标示符 d.数据拥有者加密文档标题 TIT 得到 ETIT 且为每一个文档构造出安全加密的 EAFM 索引.最后将加密文档集 ED 及 EAFM 索引上传至云服务器.云服务器保存 ED 和 EAFM 索引返回保存地址给用户.数据拥有者构建并发送字典 Dic 到服务器保存.在搜索阶段,授权用户先对搜索的关键词 $Q=\{q_1;q_2;\cdots;q_u\}$ 构造陷门 T_Q 并上传.当授权用户接收到陷门 T_Q 后,云服务器通过字典 Dic 进行搜索,如果云服务器执行一次命中搜索协议则将结果返回.如果云服务器执行增强搜索协议则通过搜索 EAFM 索引将结果返回.最后,授权用户解密密文文档 ED.

3.2 FASSE 方案威胁模型

在 FASSE 方案构建的威胁模型中,服务器被定义为是"半诚实并且好奇"的,敌手和云服务器被视为潜在威胁对象.FASSE 方案中云服务器确保会完整正确地执行 FASSE 方案设计的所有搜索协议,不会恶意删除用户文档和恶意泄露用户隐私.但是云服务器可能会分析文档和索引,例如统计分析字符频率、文档与关键词匹配的数目、文档的历史信息等.在 FASSE 方案中敌手可能会和云服务器合谋,那么上述的这些隐私信息很可能会被敌手利用去分析攻击系统,导致系统存在安全隐患.所以在此威胁模型中 FASSE 方案要确保所有历史信息、陷门、字典、文档信息和索引的安全.

3.3 FASSE 方案算法描述

FASSE 方案是 7 个多项式时间算法(KeyGen; Encrypt; PreProcess; DicGen; Trapdoor; Search; Decrypt)的集合,定义为:

1) $K=KeyGen(1^\lambda)$.其中,λ 是安全参数.该算法用于生成加密密钥 $K=\{k,k_f,k_1,k_\pi,k_e,k_t,k_{id},k_{addr}\}$,其中 k_f,k_1 是伪随机函数密钥,k_π 是伪随机置换密钥,k_e 是对称加密密钥,k_{addr} 由服务器生成.这些密钥都是随机数,均由系统随机生成.

2) $ED=Encrypt(k,D)$.D 为明文文档集合,$D=\{D_1,D_2,\cdots,D_n\}$.ED 为密文文档集合,$ED=\{ED_1,ED_2,\cdots,ED_n\}$.使用密钥 k 以及对称加密算法(比如 AES)加密生成 ED.

3) $EAFM=PreProcess(\{k_f,k_1,k_\pi,k_e\},F\|L\|SA)$.F 是 $BWT(S)$ 生成矩阵 $\boldsymbol{W}$ 的第 1 列,L 是 $\boldsymbol{W}$ 的最后 1 列.SA 是字符串文本 S 的后缀数组.使用密钥$\{k_f,k_1,k_\pi,k_e\}$以及对称加密算法加密 F,L,SA 生成 EAFM 索引.

4) $Dic=DicGen(Enc(\{k_f,k_{id},k_t,k_{addr}\},KW,id,tit,EAFM_{addr},ED_{addr}))$.其中,$KW$ 是关键词,id 是文档标示符,tit 是文档标题,$EAFM_{addr}$ 是 $EAFM$ 的地址,ED_{addr} 是 ED 的地址.使用密钥$\{k_f,k_{id},k_t,k_{addr}\}$以及对称加密算法加密 $KW,id,tit,EAFM_{addr}$ 和 ED_{addr} 生成字典 Dic.

5) $T_{KW}=SearchToken(k_f,KW)$.其中 KW 是用户查询的关键词.利用密钥 k_f 加密关键词 KW 生成对应的搜索令牌 T_{KW}.

6) $ED(KW)=Search(Dic,EAFM,T_{KW})$.利用搜索令牌 T_{KW}、字典 Dic 和 $EAFM$ 索引进行查找,输出与 KW 匹配的加密的文档集 $ED(KW)$.具体的实现细节需要依赖具体的实现算法,将在第 4 节的 4 个搜索过程中进行详细描述.

7) $D_i(KW)=Decrypt(K,ED_i(KW))$.使用对称加密算法以及密钥 k 解密包含关键词 KW 的密文文档 $ED_i(KW)$,生成包含关键词的明文文档 $D_i(KW)$.

如果灵活精度可控的对称加密方案 FASSE 是正确的,那么对于 $\forall\lambda\in\mathbb{N},n\in\mathbb{Z},D=\{D_1,D_2,\cdots,D_n\}$,$KeyGen(1^\lambda)$ 和 $Encrypt(k,D)$ 输出的 k 和 ED 都有:

$Search(Dic,EAFM,SearchToken(k_f,KW))=ED_i(KW)$ 和 $Decrypt(k,ED_i(KW))=D_i(KW)$ 成立.

3.4 FASSE 方案系统安全性分析

本文设计的 FASSE 方案假设服务器是"半诚实且好奇"的,所有的数据查询操作对服务器都是透明的,并且所有的明文数据处理都是由可信客户端处理后以密文的形式上传到服务器.服务器仅用于存储和计算,而并不知道存储和计算的具体内容.

当存在一个敌手 A 在不知道密钥的情况下是无法得知存储在服务器上加密文档的任何信息.当敌手 A 在线攻击分析时,首先敌手 A 并不知道搜索令牌 $token_m$ 对应的是哪个关键词的令牌,那么敌手 A 就不可能知道搜索的是哪个关键词.即便敌手得知搜索令牌对应的是哪个关键词,最后也因为无法解密 SA 所以无法得知本次搜索结果是否有效,这也保护了用户的隐私信息安全.

敌手 A 不能通过 LLset 离线分析字符的频率去猜测攻击,因为 FASSE 方案使用了 $F_{k_1}(c)$ 加密,

$F_{k_f}(c)$使敌手 A 无法在不观察任何搜索令牌的情况下去执行离线频率攻击.即便敌手 A 获得了搜索关键词令牌也无法执行在线频率攻击,因为 FASSE 方案在 LL 中进行了填充使得每一个字符的频率都一样.

在存储安全性方面如果存在恶意敌手 A 想要恶意删除存在某个关键词 k 的文档时,在没有破解字典之前就无法得知文档的具体存储地址、破坏文件,这一定程度上提升了文档存储的安全性.当存在一个敌手 A 试图去离线分析字典 Dic 时,他无法得知服务器中存储了多少文档和每个关键词对应多少文档,因为客户端已经将随机记录添加到字典 Dic 中了.为了提高安全性,字典 Dic 按字典顺序排序.那么字典 Dic 在历史上是独立的,敌手 A 不能得知任何历史相关信息.但是,当用户在线搜索字符串时,服务器将向 A 暴露有多少个文档包含字符串 m.如果敌手 A 想要分析 EAFM 索引,他也不能得知任何与明文有关的内容,因为 Leontiadis[10] 针对离线频率攻击(offline frequency attack)、在线差别攻击(online difference attack)、离线差别攻击(offline difference attack)、在线频率攻击(online frequency attack)已经做出了相应的对抗手段.

3.5 数据结构图

1) AFM. AFM 是文档摘要的 FM 索引.FM 索引在文献[10]中已经有了详细的描述,它是子字符串搜索的核心部分.客户端加密 AFM 后会得到 EAFM,EAFM 的结构如图 1 所示.

2) 字典 Dic. FASSE 用一个字典 Dic 代替创建关键字集合.字典 Dic 中每一条记录的第 1 个属性是 E_{KW},它等于 $F(k_f,KW)$,表示加密的关键词 $F_{k_f}(KW)$.第 2 个属性是 d,它等于 $Enc(k_{id},id)$,表示加密文档的标识符,其中 k_{id} 是加密文档标示符 id 的对称加密密钥.这里将加密文档表示为 ED.第 3 个属性是 E_{tit},它等于 $Enc(k_f,tit)$,表示文档的加密标题.第 4 个属性是 EED_{addr},它等于 $Enc(k_{addr},ED_{addr})$,表示 ED_{addr} 的加密地址,其中 k_{addr} 是加密 ED_{addr} 的对称加密密钥,ED_{addr} 是加密文档的地址.最后一个属性是 $EEAFM_{addr}$,等于 $Enc(k_{addr},EAFM_{addr})$ 表示 $EAFM_{addr}$ 的加密地址,其中 $EEAFM_{addr}$ 是加密的 AFM 的加密地址.通过对 ED_{addr} 和 $EAFM_{addr}$ 加密尽可能地阻止敌手 A 在离线分析这些地址的时候可以获取到任何与明文文档相关的任何内容或者破坏文档.同时为了提高安全性,FASSE 还将记录(E_{KW},d,E_{tit},EED_{addr},$EEAFM_{addr}$)按字典顺序添加到字典 Dic 中.另外,方案还会随机添加 q 个记录到词典 Dic 中(q 的大小可以手动设置也可以随机生成)以防敌手 A 离线分析服务器端的文档总数以及每个关键词对应的文档总数.字典 Dic 的结构如表 1 所示:

Table 1 Dictionary *Dic*

表 1 字典表 *Dic*

E_{KW}	d	E_{tit}	EED_{addr}	$EEAFM_{addr}$
0c7ba	b9d69	U2FsdGVkX1	U2FsdGVkX1+I/Lu0	U2FsdGVk
b99	b9	8Sqveb	NKivZtBBe	X19+f/t
⋮	⋮	⋮	⋮	⋮

3.6 初始化

首先,数据拥有者会将原始的明文文档 D 上传到客户端同时会生成唯一的加密文档标识符 d.客户端加密文档 D,生成加密文档 ED、文档摘要的 FM 索引 AFM、加密的文档摘要的 FM 索引 EAFM 和加密的文档标示符 d,并通过有效地提取关键词算法提取有效关键字 k_{wi} 或者也可以使用预先设定好的关键词 k_{wi}.同时使用 k_f 作为密钥的伪随机函数 PRF $F(\cdot)$加密有效关键字 k_{wi} 生成 $F_{k_f}(k_{wi})$.客户端通过计算 $Enc(k_f,tit)$来加密文档的标题得到加密的文档标题 E_{tit}.最后客户端将这些加密的数据全部都上传到服务器.服务器保存这些加密数据并且将这些加密文档的地址 ED_{addr} 和加密的文档摘要的 FM 索引即 $EAFM$ 的地址 $EAFM_{addr}$ 用 k_{addr} 进行加密得到加密文档的加密地址 EED_{addr} 和 $EAFM$ 的加密地址 $EEAFM_{addr}$.最终得到(E_{wi},d,E_{tit},EED_{addr},$EEAFM_{addr}$)并生成记录上传到字典表 Dic 中.

4 FASSE 加密搜索过程

一次命中搜索、增强搜索和过滤搜索,它们分别对应着用户只用一次就在字典中找到关键词记录、没有在字典中找到关键词记录而只用一次就在摘要中找到记录或者多次在字典和摘要中查找到关键词记录的这 3 种搜索情况.

一次命中搜索主要适用于关键词提取精度比较高的搜索场景.增强搜索主要适用于关键词提取的比较粗糙的搜索场景.过滤搜索适用于存在关键词重要程度概念的搜索场景.

4.1 一次命中搜索

用户在输入搜索字符串 m 之后服务器会在词典

Dic 中依次去查找 $E_{KW}=F_{k_f}(m)$ 的记录.如果存在相应的记录服务器会立即执行一次命中搜索,解密这些 EED_{addr} 得到相应的加密文档地址 ED_{addr},并将这些加密文档标识符 d、加密文档标题 E_{tit} 和加密文档地址 ED_{addr} 全都发送给客户端.客户端会通过计算 $Dec(k_f,E_{tit})$ 来解密这些文件加密的标题得到明文的标题 tit.用户通过 tit 选择对应需要的加密文档标识符 d.客户端通过这些加密文档标识符 d 找到对应的加密文档地址 ED_{addr} 并将这些加密文档地址 ED_{addr} 发送到服务器并请求服务器下载这些加密文档 ED.如图 3 所示,一次命中搜索协议的特点是对于搜索关键词已经在字典 Dic 中的这次搜索服务器会直接检索字典 Dic 中的 E_{KW} 属性从而最快得到检索结果,它是整个 FASSE 搜索过程中耗时最短、精度最高的.但是和传统的基于关键词集合的 SSE 一样,对关键词的提取和关键词集的建立的要求都十分高.综上可得一次命中搜索主要适用于关键词提取精度比较高的搜索场景.

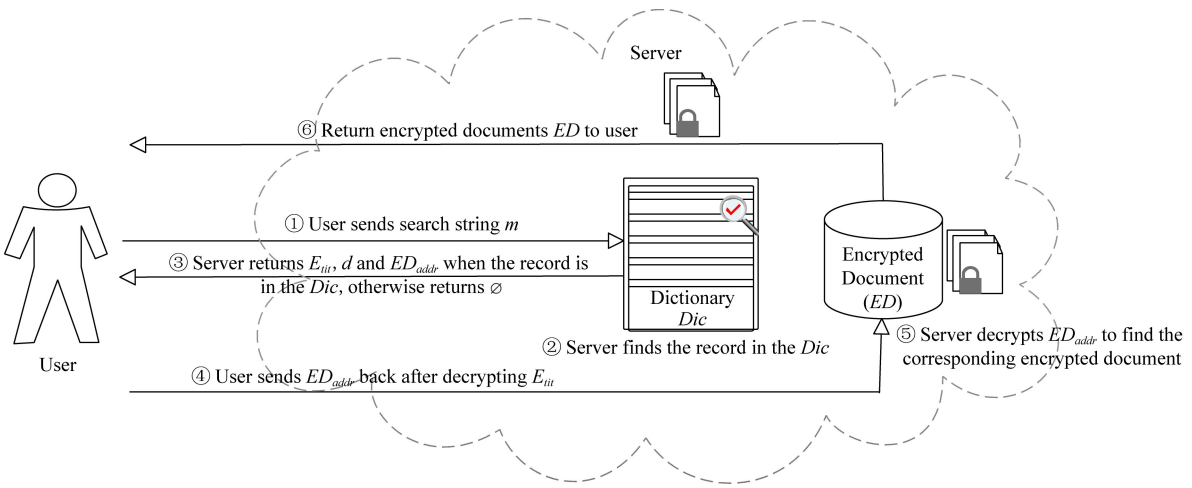


Fig. 3 One-shot search

图 3 一次命中搜索

一次命中搜索协议如下:

客户端:生成搜索字符串 m 的搜索令牌 $token_m$ 并发送到服务器.

服务器:if $Find(token_m,Dic)==\perp$

/* 字典 Dic 中未匹配到 $token_m$ */

返回⊥(空)到客户端;

else

返回 d,E_{tit},EED_{addr} 和 $relativity$ 到客户端;

其中 $relativity$ 的每一个元素值都为"Max".

end if

客户端:if 从服务器端接收到⊥

请求服务器执行增强搜索 or 结束退出;

else /* 从服务器收到 d,E_{tit},EED_{addr} 和 $relativity$ */

if 搜索结果相关性低 /* 由于关键词提取算法粗糙 */

设置 $relativity$ 的每一个元素值都为 1.0;

生成新的搜索字符串 m_i 的搜索令牌 $token_{m_i}$;

发送 $token_{m_i}$,d,E_{tit},EED_{addr} 和 $relativity$ 到服务器请求执行过滤搜索;

or 发送 EED_{addr} 后等待接收从服务器发送的 ED;

else

解密 E_{tit} 得到 tit;

根据 tit 选择需要的 d 并得到相应的 EED_{addr};

发送 EED_{addr} 到服务器请求服务器进行下载.

end if

end if

服务器:解密 EED_{addr} 下载 ED 发送到客户端.

4.2 增强搜索

如果服务器在词典 Dic 中未找到任何相应的记录时,这种情况是经常遇到的,那么增强搜索将起到重要的作用.服务器将通过字典 Dic 去搜索每个文档的 EAFM,并将与字符串搜索令牌即 $F_{k_f}(m)$ 相匹配的记录中的 d,E_{tit} 和服务器解密后得到的 ED_{addr} 发送到客户端.如果担心服务器返回大量无

效填充位置,那方案也可以使用文献[10]中的 *bucketization* 技术并选取适当大小的窗口使填充节点的数量大大减少.通过计算 $Dec(k_f, E_{tit})$ 来解密得到这些文件的标题 tit.用户通过 tit 选择需要的 d.最后,客户端将 ED_{addr} 发送到服务器请求下载这些 ED.增强搜索结构如图 4 所示.如果服务器有足够的空间资源 FASSE 也可以采用建立内容的 FM 索引即 CFM(FM index of the content)的方法来进一步提高搜索结果的精确性.增强搜索协议的特点是针对字典 Dic 中不存在搜索关键词的情况下从而对文档摘要进行搜索,那么增强搜索相对一次命中搜索速度就会慢上很多,但它也打破了传统基于关键词集合的 *SSE* 因为在关键词集合中未检索到搜索关键词而无结果的尴尬局面.综上可见增强搜索主要适用于关键词提取的比较粗糙的搜索场景.

增强搜索协议如下:

服务器:解密整个字典 Dic 中的 $EEAFM_{addr}$ 得到 EAFM;

对 EAFM 执行 SubLF 映射算法;

/* 当从过滤搜索执行增强搜索时字典 Dic 只保留加密的文档标识符为 d 的那部分 */

计算 m 在所有摘要中不算填充的有效次数 $Occur$;

if $Occur$ 中的每一元素的值都为 0

返回⊥并结束退出;

/* 当从过滤搜索执行增强搜索时则返回上一步的搜索结果 */

else

从 $Occur$ 中选最大的 n_k 项 $Occur_k$;

取出 $Occur_k$ 对应字典 Dic 中 n_k 项的记录 $Temp$;

设置 $relativity$ 的每一个元素值与 $Temp[Occur_k]$ 相对应; /* 当从过滤搜索执行增强搜索时 $relativity$ 等于上一次的 $relativity$ 加上 $0.2 * Temp[Occur_k] * /$

发送 $Temp[d, E_{tit}, EED_{addr}, Occur_k]$ 到客户端.

end if

客户端:取出 $Temp[Occur_k]$;

if 搜索结果相关性低/* $Occur_k$ 都很小 */

生成新的搜索字符串 m_i 的搜索令牌 $token_{m_i}$;

发送 $token_{m_i}, d, E_{tit}, EED_{addr}$ 和 $relativity$ 到服务器请求执行过滤搜索;

or 发送 EED_{addr} 后等待接收从服务器发送的 ED;

else

解密 $Temp[E_{tit}]$ 得到 tit;

根据 tit 从 $Temp[d]$ 选择需要的 d 并得到相应的 EED_{addr};

发送 EED_{addr} 到服务器请求服务器进行下载.

end if

服务器:解密 EED_{addr} 下载 ED 发送到客户端.

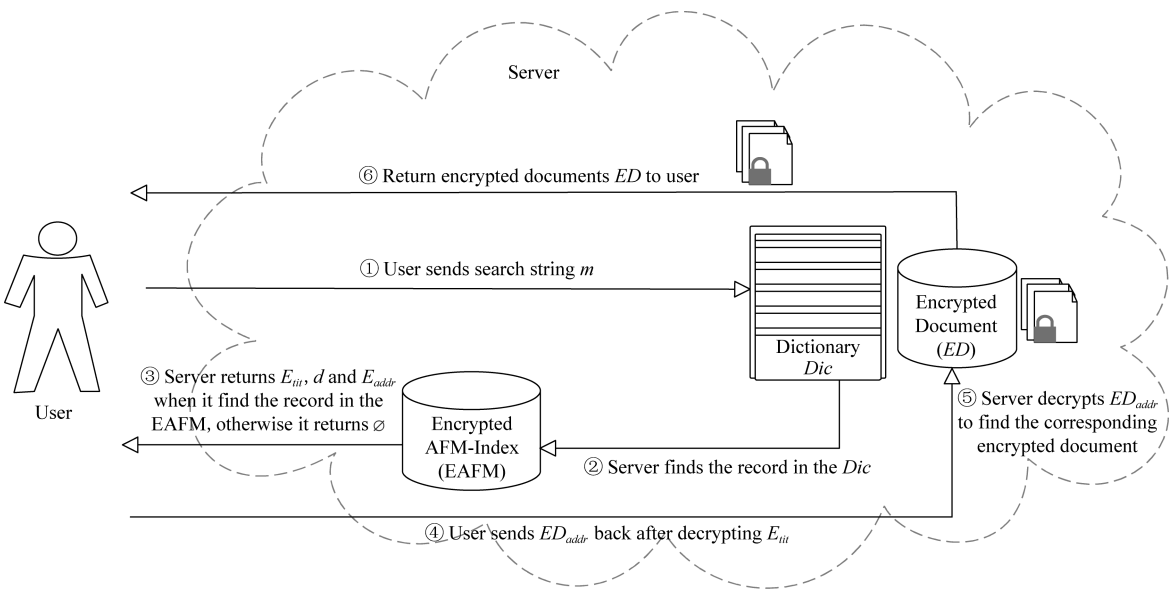


Fig. 4 Reinforcement search

图 4 增强搜索

4.3 过滤搜索

用户在经过了一次命中搜索或者增强搜索之后仍然得到了许多难以选择的 tit 和 d，这是最常见的但也同时是最麻烦的情况.这种情况如果服务器的查询结果中只有少量的小文件时，那么用户就可以直接发送请求给服务器请求服务器下载这些文档到本地来筛选并确定哪些文档是自己需要的.但是如果服务器的查询结果中有许多大文档时，那么巨大的网络开销是服务器和客户端双方都难以承受的.事实上 FASSE 中的过滤搜索其实是相当于一个筛子，可以连续地过滤用户不满意的文档.FASSE 将成功匹配 m_1 搜索令牌的数量定义为相关性，字符串 m_1 出现在摘要中的频率越高，那么文档的相关性越大，反之亦然.服务器会选择相关度最高的前 $n\%$ 个文件进行下一步处理.n 可以通过计算成功匹配了字符串 m_1 摘要的总数的百分比或手动设置合适的数字来确定.服务器将 nd，nE_{tit} 和解密后的 nED_{addr} 返回给客户端，客户端通过计算 $Dec(k_f, E_{tit})$ 来解密这些 tit.最后，用户通过 tit 选择需要的 d 和 ED_{addr} 最终获得需要的 ED.过滤搜索是通过使用不同的字符串 m_i 在这些难以选择的 d 上执行重复搜索.用户将得到 n_1，n_2…直到有确定可以下载的 n_k 个 d 时，下载这 n_k 个文件，因为重复过滤后文件的相关性达到了最大.过滤搜索结构如图 5 所示.因此可得，用户可以控制搜索的精准性，以便搜索需要的文档.搜索的准确度越高，用户获得预期文档的可能性就越大，FASSE 通过控制用户获得预期文档的可能性就越大.FASSE 通过控制过滤搜索的次数来达到精度的控制，从而实现精度可控的搜索.过滤搜索协议的特点是针对在一次关键词搜索字典和文档摘要后得到太多的结果的局面，对文档进行反复多次地一次命中搜索或增强搜索，它消耗的时间最多、速度最慢，并且需要用户手动设置新的关键词.综上所述过滤搜索适用于存在关键词重要程度概念的搜索场景.

过滤搜索协议如下：

服务器：接收从客户端发送的 d，E_{tit}，ED_{addr}，$relativity$；

if $Find(token_{m_i}, Dic, d) == \perp$

/* 在字典 Dic 属性 d 的值等于 d 的记录中未匹配到 $token_{m_i}$ */

返回 $\perp$（空）到客户端；

else

返回 d，E_{tit}，EED_{addr} 和 $relativity$ 到客户端；

其中 $relativity$ 的每一个元素值都为“Max”.

end if

客户端：if 从服务器端接收到 $\perp$

请求服务器在 d 上执行增强搜索；

or 发送 EED_{addr} 后等待接收从服务器发送的 ED.

else /* 从服务器收到 d，E_{tit}，EED_{addr} 和 $relativity$ */

end if

if 搜索结果相关性低

/* 由于关键词提取算法粗糙 */

设置 $relativity$ 的每一个元素值都为 1；

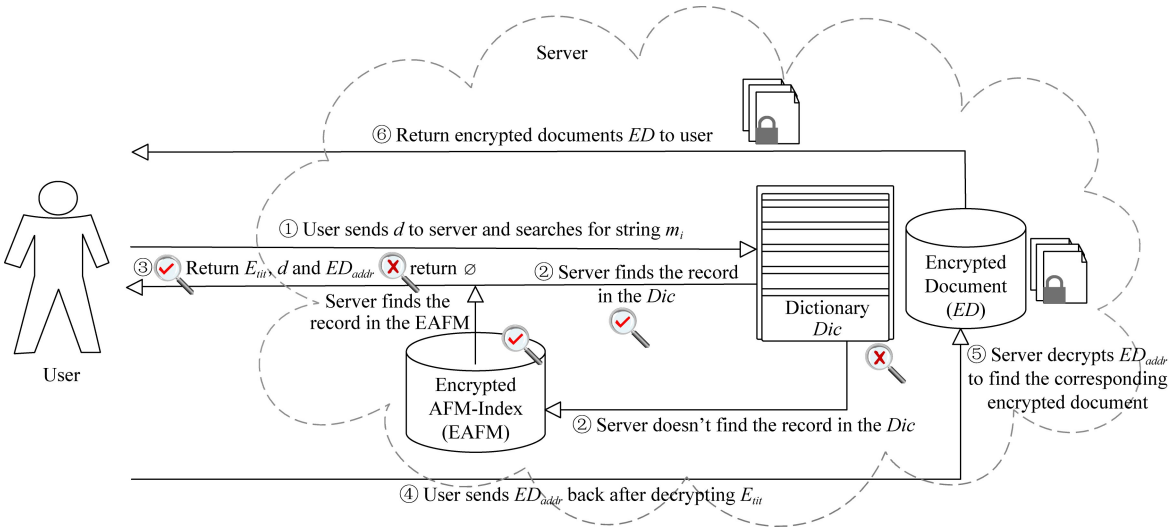


Fig. 5 Filter search

图 5 过滤搜索

生成新的搜索字符串 m_i 的搜索令牌 $token_{m_i}$;

发送 $token_{m_i}$,d,E_{tit},EED_{addr} 和 $relativity$ 到服务器请求执行过滤搜索;

or 发送 EED_{addr} 后等待接收从服务器发送的 ED;

else

解密 E_{tit} 得到 tit;

根据 tit 选择需要的 d 并得到相应的 ED_{addr};

发送 ED_{addr} 到服务器请求服务器进行下载.

end if

服务器:解密 ED_{addr} 下载 ED 发送到客户端.

4.4 基于通配符的模糊增强搜索

模糊增强搜索操作是利用通配符'*'和'?'在普通增强搜索的基础上进行简单的修改.当服务器在成功匹配到搜索令牌 $F_{k_f}(T[i])$ 时,如果发现 $F_{k_f}(T[i])=F_{k_f}('?')$,那么直接跳过这次与 L 列中 $F_{k_f}(c_{L_j})$ 的比对,利用从 F 列中得到的 $F_{k_f}(r_{F_j}\|c_{F_j})$ 异或 L 列中的 $F_{k_f}(c_{L_j})\oplus F_{k_f}(r_{F_j}\|c_{F_j})\|F_{k_f}(r_{L_j}\|c_{L_j})$ 获得 L 列中的 $F_{k_f}(c_{L_j})\|F_{k_f}(r_{L_j}\|c_{L_j})$,继续通过异或运算得到 $F_{k_f}(c_{L_j})\oplus F_{k_f}(r_{L_j}\|c_{L_j})F_{k_f}(c_{L_j})\|F_{k_f}(r_{L_j}\|c_{L_j})$ 并通过 LLset 依次在 F 列中查找与 $F_{k_f}(c_{L_j})\oplus F_{k_f}(r_{L_j}\|c_{L_j})$ 相等的 $F_{k_f}(c_{F_k})\oplus F_{k_f}(r_{F_k}\|c_{F_k})$,经过多次迭代后就可以找到需要搜索的关键词.为了进一步提高方案的灵活性,方案提出了最大容忍错拼字符数 em,并且默认 $em=0$;也可以手动设置 em 的值,只要 em 的值合理即可.关于错拼的搜索本质基本和'?'一样,它需要引入一个错误程度变量 e,当发现 $F_{k_f}(c_{L_j})$ 不等于 $F_{k_f}(T[i-1])$ 时直接跳过这次在 L 列中的比对,同时执行 $e++$ 一次,只要 e 不超过设定的最大容忍错拼字符数 em 就属于错误可容忍范围内.关于通配符'*',FASSE 设置'*'最大能够表示 r 个字符.本文在实现 FASSE 方案时将 r 的值设置为 2.例如"$he*lo$"={"$helo$","$he?lo$","$he??lo$"}.通过观察发现增强模糊搜索本质上还是对包含'?'的字符串搜索.关于 e,它和"?"一样相当于在不匹配的位置上加上了"?".本方案通过对 em,r,"?"和"*"的设置来实现灵活的精度可控的模糊搜索.最后可能还会得到一些不相关的结果,可以通过返回结果的相关性来剔除相关性低的结果,从而得到一个相对不错的结果.

5 实验验证

5.1 测试环境搭建

本方案实验测试使用的计算机操作系统是 Windows7,机器配备了 Intel® Core™ i5-4590 CPU @ 3.30 GHz 四核处理器,具有 8 GB RAM 内存.实验是在本地同时模拟了一台服务器和一个客户端,开发环境为 Eclipse_4.5.0,使用语言为 Java 编程语言.对称加密算法是使用 AES 密码算法且实现均调用的是 java.security 和 javax.crypto 下工具包.本系统的实现 Java 代码已经全部上传至 https://github.com/taoruyu/FA-SSE.git 上.

5.2 实验设计

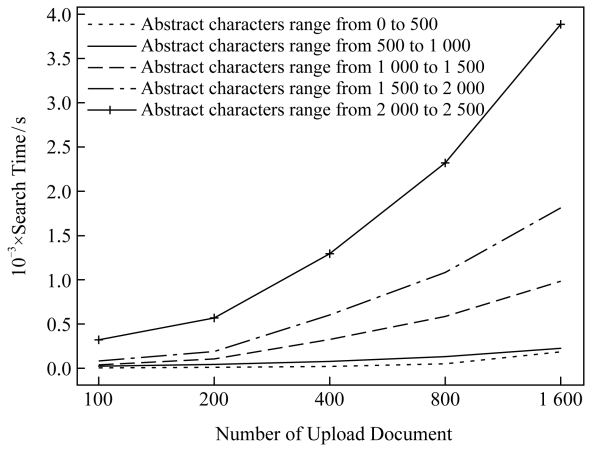
本实验主要从 2 个方面测试方案的性能效率.首先测试上传生成索引的时间效率,研究生成索引的时间与文档的数量和摘要包含的字符数的相关性;其次对搜索的效率进行进一步的实验研究.对于搜索的效率研究,本文从增强搜索和模糊增强搜索 2 个方面进行对比试验,探索文档的数量与搜索关键词长度和文档包含的字符总数对搜索效率的影响.关于本次实验的所有实验数据均来自于真实数据集 eprint 中所有文章的英文摘要.FASSE 方案搜索的时间效率主要取决于是否是执行了一次命中搜索.如果搜索过程中的所有操作都是一次命中搜索,这意味着它在搜索过程中所需的时间几乎等于在字典 Dic 中搜索全部记录的时间.这个时间相对于其他搜索过程是可以被忽略不计的.然而,搜索操作不可能都是一次命中搜索,大多数情况下都是增强搜索或过滤搜索,并且搜索过程中的大部分时间也都会消耗在这 2 种搜索过程中.经过多轮搜索后,增强搜索和过滤搜索的次数将继续不断增加,搜索所需的总时间会越来越多,但是每一轮新的搜索所需要的时间也会越来越短同时准确性会越来越高.本实验中未对过滤搜索的性能进行实验研究,因为过滤搜索的时间是可以从增强搜索的时间效率图中计算出来的.过滤搜索本质也是一次命中搜索和增强搜索的不断循环交替,不同的是过滤搜索的文档范围会越来越小.

5.3 实验结果

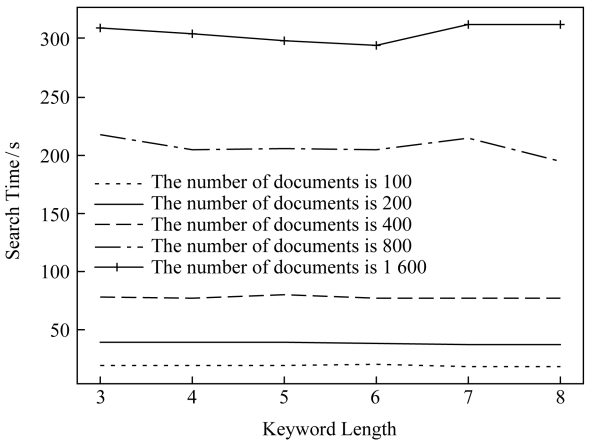
FASSE 方案的上传和搜索的结果具体参考图 6.其中图 6(a)展示了 FASSE 方案上传文件所需的时间效率图.从中可以计算出当摘要包含 500 字符以内

时,上传效率平均每个为 0.09 s;当摘要包含 500～1 000 字符时,上传效率平均每个为 0.32 s,当摘要包含 1 000～1 500 字符时,上传效率平均每个为 0.79 s;当摘要包含 1 500～2 000 字符时,上传效率平均每个为 1.42 s;当摘要包含 2 000～2 500 字符时,上传效率平均每个为 2.34 s.图 6(b)表示增强搜索的时间性能图(固定摘要包含 2 000～2 500 字符),可以发现当文档数逐渐增加时,搜索所需时间越来越不稳定,这是由于不同的搜索关键词在文档中的频率不一样导致的.图 6(c)表示关键词搜索与文档摘要包含的字符总数的关系图(固定搜索关键词长度为 3).图 6(c)可以发现摘要包含字符总数与搜索时间呈现正相关,且呈现出当文档数越多这种正相关性越强的趋势.在图 6(c)中,当摘要包含的字符总数小于 500 时,对于各个不同文档数量的平均搜索时间依次为 36.425 ms,67.375 ms,109.125 ms,149.125 ms,209.25 ms,故平均搜索时间为 114.26 ms.文献[15]讨论了面向多关键字的模糊密文搜索,与本文设计的方案相近,同本文具有一定的可比性.该文通过将文件数分为 6 类进行可搜索对称加密实验,该文的平均搜索时间约为 120 ms,本文的搜索效率略有提高.

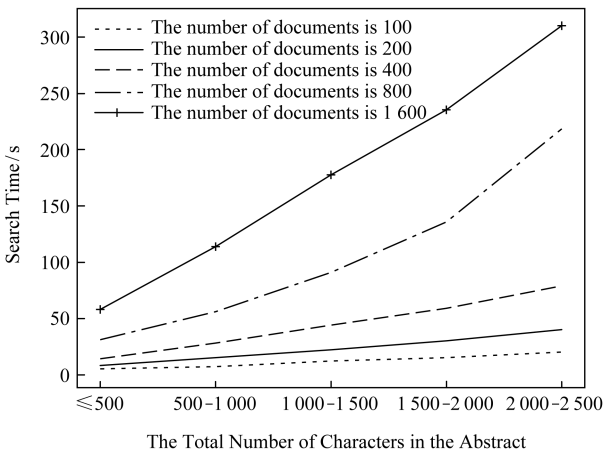
通过图 6 的实验看到了搜索关键词对实验结果



(a) Document upload efficiency



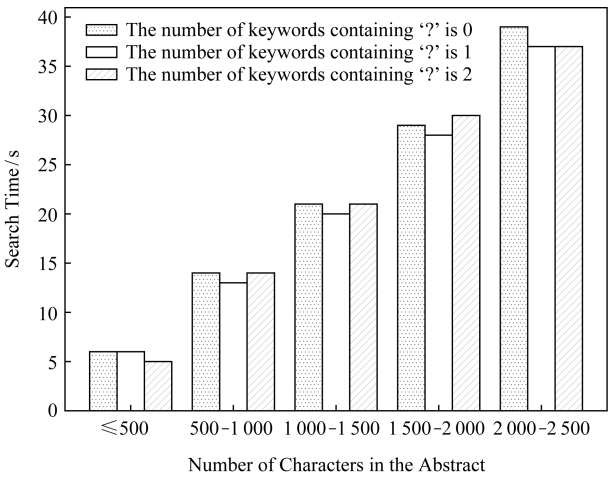
(b) Effect of keyword length on search time



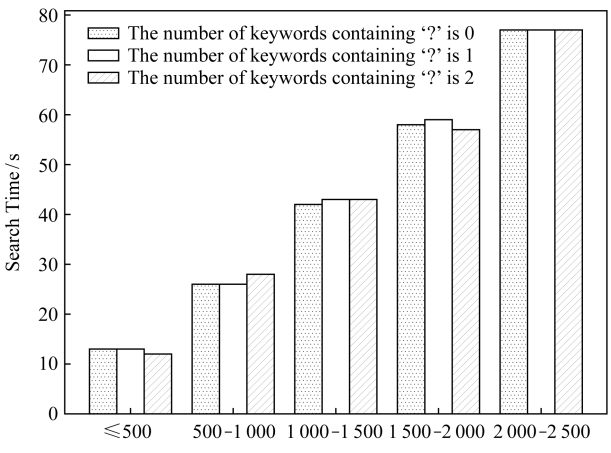
(c) The effect of the total number of characters contained in the abstract on search time

Fig. 6 Efficiency of uploading and searching for the FASSE

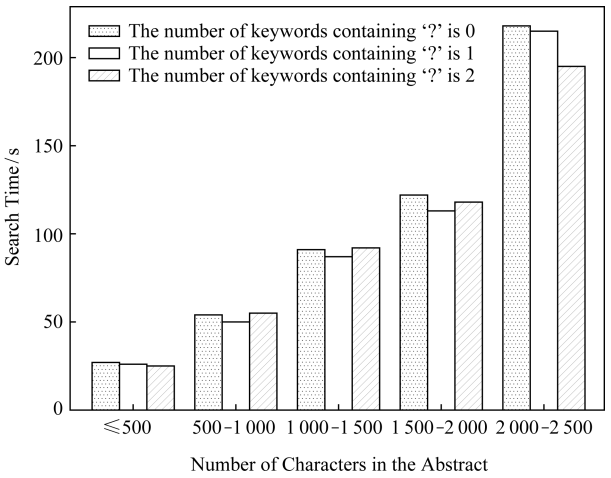
图 6 FASSE 方案的上传和搜索的效率



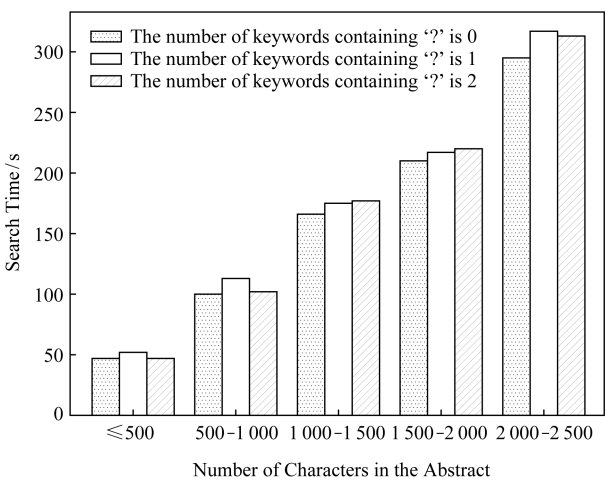
(a) Fuzzy enhanced search efficiency when n=200



(b) Fuzzy enhanced search efficiency when n=400



(c) Fuzzy enhanced search efficiency when $n=800$



(d) Fuzzy enhanced search efficiency when $n=1\,600$

Fig. 7 The efficiency of fuzzy enhanced search with different n

图 7 文档数 n 不同时模糊增强搜索效率

的影响,图 7 将给出模糊增强搜索的搜索效率图.(由图 6(b)得到结果,设置固定关键词长度为 6).

由图 7 可以发现通配符'?'对搜索结果的影响其实并不大,所以方案会在提高实用性的同时也会兼顾搜索的效率.

6 总 结

本文基于 Leontiadis[10] 可搜索对称加密方案提出了一种能够支持灵活的精度可以控制的可搜索对称加密方案——FASSE 方案,它支持灵活精度可控的加密搜索并且节省了大量的存储开销.FASSE 方案提供了 4 种基本搜索方式:一次命中搜索、增强搜索、过滤搜索和模糊增强搜索.同时,本文还结合这 4 种搜索方式设计了一种基于通配符技术的模糊增强搜索,并且通过设置部分参数从而达到灵活可控精度的搜索效果从而进一步增强系统的实用性.最后实验得出 FASSE 方案在实验论文数据集中平均搜索完每一篇论文的时间为 114.26 ms.通过实验可以发现 FASSE 方案可以轻松应对各种关键词的搜索.但是目前 FASSE 方案不支持动态更新 EAFM 索引,所有的 EAFM 索引都是静态的,对用户操作(增、删、改)都无法做到及时更新,必须重新生成 EAFM 索引,这会导致时间和空间开销增加,所以一个可以动态更新 EAFM 索引的解决方案是我们今后的目标.FASSE 将会参考文献[16],并且在未来希望它能帮助改进 FASSE.此外,FASSE 方案的模糊搜索方案中暂时不提供一次命中模糊搜索,所以在今后的工作中希望可以弥补这一点.

参 考 文 献

[1] Popa R A, Redfield C M S, Zeldovich N, et al. CryptDB: Protecting confidentiality with encrypted query processing [C] //Proc of the 23rd ACM Symp on Operating Systems Principles(SOSP). New York: ACM, 2011: 85-100

[2] Chen Ping, Zhang Tao, Zhao Min, et al. Database as service system for business database application hosting and its privacy preservation mechanism [J]. Computer Science, 2013, 40(11): 140-142, 146 (in Chinese)
(陈萍, 张涛, 赵敏, 等. 面向托管的数据库即服务系统及其隐私保护技术[J]. 计算机科学, 2013, 40(11): 140-142, 146)

[3] Wang Haiwei, Yang Geng, Liu Guoxiu, et al. Design and implementation of searchable database encryption system [J]. Computer Technology and Development, 2017, 27(8): 130-134 (in Chinese)
(汪海伟, 杨庚, 刘国秀, 等. 可搜索数据库加密系统的设计与实现[J]. 计算机技术与发展, 2017, 27(8): 130-134)

[4] Cash D, Jaeger J, Jarecki S, et al. Dynamic searchable encryption in very-large databases: Data structures and implementation [C] //Proc of 2014 Network and Distributed System Security Symp. San Diego: Internet Society, 2014: 853

[5] Faber S, Jarecki S, Krawczyk H, et al. Rich queries on encrypted data: Beyond exact matches [C] //Proc of the 20th European Symp on Research in Computer Security. Berlin: Springer, 2015: 123-145

[6] Chase M, Shen E. Substring-searchable symmetric encryption [C] //Proc of the 15th Privacy Enhancing Technologies Symp. Piscataway, NJ: IEEE, 2015: 263-281

[7] Hu Changhui, Han Lidong. Efficient wildcard search over encrypted data [J]. International Journal of Information Security, 2016, 15(5): 539-547

[8] Zhou Wei, Liu Lixi, Jing He, et al. K-Gram based fuzzy keyword search over encrypted loud computing [J]. Journal of Software Engineering and Applications, 2013, 6(1): 29-32

[9] Zhang Zhenjie, Marios H, Beng C O, et al. Bed-Tree: An all-purpose index structure for string similarity search based on edit distance [C] //Proc of the ACM Special Interest Group on Management of Data. Indianapolis, Indiana: Association for Computing Machinery, 2010: 915-926

[10] Leontiadis I, Li Ming. Storage efficient substring searchable symmetric encryption [C] //Proc of the 39th Annual IACR Scientific Conf. Utrecht, Netherlands: Cryptology ePrint Archive, 2017: 153

[11] Burrows M, Wheeler D J. A block sorting lossless data compression algorithm [R]. Palo Alto, CA: Digital System Rearch Center, 1995

[12] Luo Suiqian. Suffix array-A powerful tool for dealing with strings: 2009 China Nation Olympiad in Informatics [CP]. Singapore: School of Computing, NUS, 2009

[13] Karkkainen J, Sanders P. Simple linear work suffix array construction [C] //Proc of Int Colloquium on Automata, Languages and Programming. Berlin: Springer, 2003: 943-955

[14] Ferragina P, Manzini G. Opportunistic data structures with applications [C] //Proc of the 41st Annual Symp on Foundations of Computer Science. Piscataway, NJ: IEEE, 2000: 390-398

[15] Wang Kaixuan, Li Yuxi, Zhou Fucai, et al. Fuzzy ciphertext search for multiple keywords [J]. Journal of Computer Research and Development, 2017, 54(2): 348-360
(王恺璇, 李宇溪, 周福才, 等. 面向多关键字的模糊密文搜索方法[J]. 计算机研究与发展, 2017, 54(2): 348-360)

[16] Salson M, Lecroq T, Léonard M, et al. A four-stage algorithm for updating a burrows-wheeler transform [J]. Theoretical Computer Science, 2009, 410(43): 4350-4359

Li Ximing, born in 1974. PhD, associate professor. His main research interests include secret-sharing based secure database and searchable symmetric encryption technology.

Tao Ruyu, born in 1995. MSc. His main research interests include information security, machine learning, deep learning.

Su Chen, born in 1997. BD. Her main research interests include searchable symmetric cryptography, applied cryptography.

Huang Qiong, born in 1982. Received his BS and MS degrees from Fudan University in 2003 and 2006, respectively, and received his PhD degree from the City University of Hong Kong in 2010. Professor with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. His main research interests include cryptography and information security, in particular, cryptographic protocols design and analysis.

Huang Xinyi, born in 1981. Received his PhD degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. Professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, China. His main research interests include cryptography and information security.