doi: 10.12068/j.issn.1005-3026.2020.09.005

基于BSSEVD的可搜索加密方案原型系统设计与实现

厉 鹏[1,2], 周福才[1], 张 帅[1]
(1. 东北大学 软件学院, 辽宁 沈阳 110169; 2. 辽东学院 信息工程学院, 辽宁 丹东 118000)

摘 要: 针对大型数据集条件下,安全索引文件过大而导致可搜索加密方案的关键字搜索时间复杂度过高、效率低的问题,提出了大型数据集下支持布尔搜索的可搜索加密方案(BSSEVD). 方案采用三层间接寻址块状存储安全索引的方法优化安全索引存储结构,通过增加关键字交集安全索引解决多关键字布尔搜索导致的泄露增加问题. 并在该方案基础上,设计与实现可搜索加密方案原型系统. 该系统主要包括文件预处理模块、初始化模块和关键字搜索模块等三大模块. 通过实验测试对系统的性能进行分析,实验结果表明该方案计算效率得到了较大的提升.

关 键 词: 云存储;大型数据集;布尔搜索;可搜索加密;倒排索引
中图分类号: TP 309 **文献标志码**: A **文章编号**: 1005-3026(2020)09-1244-07

Design and Implementation of Searchable Encryption Scheme Prototype System Based on BSSEVD

LI Peng[1,2], *ZHOU Fu-cai*[1], *ZHANG Shuai*[1]
(1. School of Software, Northeastern University, Shenyang 110169, China; 2. School of Information Engineering, Eastern Liaoning University, Dandong 118000, China. Corresponding author: ZHOU Fu-cai, E-mail: fczhou@mail.neu.edu.cn)

Abstract: In order to solve the problem of high time-complexity and low efficiency of keyword search of searchable encryption scheme caused by large security index file in large data set, a scheme of Boolean symmetric searchable encryption in very-large databases (BSSEVD) was proposed. The storage structure of security index was optimized by using three-layer indirect addressing block storage, and the leakage problem caused by multi keyword Boolean search was solved by adding keyword intersection security indexes. Based on this scheme, a prototype system of searchable encryption scheme is designed and implemented, which mainly includes three modules: file preprocessing module, initialization module and keyword search module. The performance of the system was analyzed by experiments, and the experimental result showed that calculation efficiency of the scheme is significantly improved.

Key words: cloud storage; large data sets; Boolean search; searchable encryption; inverted index

随着云存储技术应用的日益普及,云存储服务以其价格低廉、可扩展性高等特点吸引了大量用户. 但是,在云存储服务带来极大的便利的同时,许多安全问题也随之产生,网络攻击技术日新月异的发展以及人们对经济利益的不当追逐,导致云存储服务器可能同时面临外部敌手的攻击或服务器内部管理者的人为泄漏和破坏. 为了防止云服务器及其他非授权用户获取这些数据,加密技术是最安全有效的手段之一. 然而,明文数据在经过加密方法的处理变成密文数据之后,会丧失许多原有特性[1],虽然通过这种方式保证了用户存储在云端的数据的机密性,但同时也使得用户

收稿日期: 2020-04-28
基金项目: 国家自然科学基金资助项目(61772127, 61472184, 61872069); 国家科技重大专项(2013ZX03002006); 中央高校基本科研业务费专项资金资助项目(N151704002).
作者简介: 厉 鹏(1976-),男,河北三河人,东北大学博士研究生; 周福才(1964-),男,吉林长春人,东北大学教授,博士生导师.

的许多应用无法直接在密文数据上进行，例如：通过关键字来搜索密文文件.

最早可搜索加密[2]（searchable encryption，SE）这一概念是由 Goldreich 和 Ostrovsky 提出的，是在不解密密文情况下实现对加密数据的搜索. Song 等[3]在 2000 年首先提出了基于密文扫描思想的 SWP 方案，该方案将明文文件划分为“关键字”，并对关键字进行特殊的双层加密. 该方案基于对称密码学算法实现在密文上进行关键字搜索，然而，该方案需要扫描整个文件，搜索效率极低，不适合大型数据集. 在此基础上，Goh[4]在 2003 年提出了一种利用安全索引进行关键字搜索的 SE 方案. 该方案基于文件和关键字构建安全索引，安全索引不会向服务器泄露关于明文文件和关键字的任何信息. 在可搜索加密领域，已有很多研究者从实现的角度来提出可搜索加密方案，可以大致分为两类：基于对称密钥的可搜索加密方法以及基于公钥加密的可搜索加密方法. 文献[3－7]提出基于对称加密的可搜索加密方法，文献[8－13]提出基于公钥加密的可搜索加密方法. 在用户数据集较大情况下，由 Cash 等[14]提出了一种动态可搜索加密方案，该方案利用 multi-map 的结构特性，对安全索引进行了优化，可实现大型安全索引情况下的单关键字密文搜索.

通过以上分析，目前的可搜索加密方法虽然可以实现在密文数据上进行关键字搜索，但依然存在以下两个问题：1）在大型数据集条件下，关键字对应文件信息的键值对数量过多，使得安全索引过大，从而导致传统可搜索加密方案的关键字搜索时间复杂度过高、效率低；2）方案多为单关键字搜索方案，不能满足实际应用中对多关键字布尔搜索的需求.

针对以上两个问题，论文提出了大型数据集下支持布尔搜索的可搜索加密（BSSEVD，Boolean symmetric searchable encryption in very-large databases）方案，包括初始化算法、搜索令牌生成算法、搜索算法三个主要算法，以及文件加密算法、文件解密算法. 该方案在保证访问模式不泄露的前提下，提供一种在大型数据集条件下支持布尔搜索的可搜索加密方案. 在该方案基础上，设计与实现原型系统.

1　BSSEVD 方案

1.1　BSSEVD 架构

大型数据集下支持布尔搜索的可搜索加密（BSSEVD）方案的架构如图 1 所示.

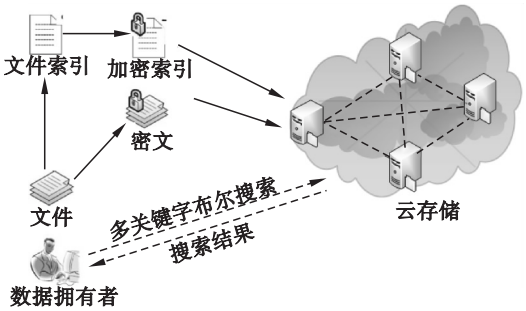

图 1　方案模型

Fig. 1　Model of scheme

在图 1 中，BSSEVD 方案中包括两个实体：数据拥有者和云存储服务器.

数据拥有者（用户）：负责在本地进行原始文件集预处理和初始化处理，生成安全索引结构以及密文数据文件，将其上传至云服务器进行存储；搜索阶段负责布尔搜索表达式解析、搜索令牌生成，并将搜索请求发送给云存储服务器.

云存储服务器：云存储服务器存储加密文件和加密索引，并利用用户的搜索令牌实现高效安全的多关键字布尔搜索计算，并将搜索结果返回给数据拥有者（客户）.

1.2　方案模型

本方案主要由 5 个多项式时间算法：初始化算法、文件加密算法、令牌生成算法、搜索算法、文件解密算法组成，大型数据集下支持布尔搜索的可搜索加密方案可形式化描述为 BSSEVD = (Init, Enc, TokenGen, Search, Dec). 下面对方案中的 5 个算法进行形式化定义.

1）初始化算法：$(K, I_E) \leftarrow \mathrm{Init}(1^k, I_P)$ 为概率性算法. 输入安全参数 k 和明文倒排索引 I_P，输出令牌生成密钥 K 和安全索引 I_E.

2）文件加密算法：$c \leftarrow \mathrm{Enc}(K_c, f)$ 为确定性算法. 输入对称加密密钥 K_c 和文件集合 f，输出密文集合 c.

3）令牌生成算法：$\mathrm{tk} \leftarrow \mathrm{TokenGen}(K, \mathrm{Bw})$ 为概率性算法. 输入令牌生成密钥 K 和搜索布尔表达式 Bw，输出搜索令牌 tk.

4）搜索算法：$R \leftarrow \mathrm{Search}(\mathrm{tk}, I_E)$ 为确定性算法. 输入搜索令牌 tk 和安全索引 I_E，输出搜索令牌所对应的含有密文标签信息的搜索结果集合 R.

5）文件解密算法：$f \leftarrow \mathrm{Dec}(K_c, c)$ 为确定性算法. 输入对称加密密钥 K_c 和密文文件 c，输出为明文文件 f.

1.3 关键技术

1) 倒排索引. 倒排索引(inverted index)是一种实现关键字到文件映射关系的有效索引结构,是实现"关键字 - 文件矩阵"的一种具体存储形式. 通过倒排索引,可以根据关键字快速检索出包含这个关键字的文件列表. 倒排索引主要由两部分组成:关键字字典和倒排文件.

关键字字典(lexicon):关键字字典是由文件集合中出现过的所有关键字构成的字符串集合,关键字字典内的索引项中记录着关键字本身的信息以及指向"倒排列表"的指针. 倒排列表中记录着出现过某个关键字的所有文件的相关信息,每条记录被称之为一个倒排项.

倒排文件(inverted file):所有关键字的倒排列表一般顺序地存储在磁盘上的某个文件中,这个文件被称之为倒排文件.

2) 分层块状安全索引结构. 传统方案基于安全索引的可搜索加密方案,在大型数据集下构建的庞大索引中,通过顺序匹配关键字实现关键字搜索的方法效率低. 受到数据存储在外部存储器(磁盘)与存储在内部存储器中读取效率的差异巨大的启发,通过将安全索引三层分块进行间接搜索的方法,优化安全索引结构,使本方案在安全索引过大情况下,仍然具有良好的搜索性能.

分层块状安全索引构造过程:

①设定分块参数 B 和 b,关键字 ω 倒排索引长度不大于 B^2b. 如果一级分块长度不足 b,二级、三级分块长度不足 B,填充随机数据.

②当关键字 ω 倒排索引长度 $|I_P(\omega)| \leqslant b$,形成 1 块一级索引块.

③当 $b < |I_P(\omega)| \leqslant Bb$,将 $I_P(\omega)$ 分成 $\lceil I_P(\omega)/B \rceil$ 块,作为二级索引块. 计算得到 $\lceil I_P(\omega)/B \rceil$ 个块的标签和块指针键值对,形成 1 块一级索引块.

④当 $Bb < |I_P(\omega)| \leqslant B^2b$,将 $I_P(\omega)$ 分成 $\lceil I_P(\omega)/B \rceil$ 块作为三级索引块,计算得到 $\lceil I_P(\omega)/B \rceil$ 个三级索引块的标签和块指针键值对,再将其分成 $\lceil I_P(\omega)/B/B \rceil$ 块作为二级索引块,计算得到 $\lceil I_P(\omega)/B/B \rceil$ 个二级索引块的标签和块指针键值对,形成 1 块一级索引块.

2 关键算法描述

本方案涉及的 5 个算法中文件加密算法、文件解密算法与传统可搜索加密方案类似,使用成熟的对称加密算法,故在本文中主要介绍大型数据集下支持布尔搜索的可搜索加密方案的核心内容——初始化算法、令牌生成算法、搜索算法. 算法中应用到传统 SSE 方案中的伪随机数函数 F、对称加密算法 Enc、字典加密算法 DX(Init, TokenGen, Search)和 multi-map 加密算法 MM(Init, TokenGen, Search)不在此赘述. 算法中涉及的符号定义如表 1 所示. 下面分别对这 3 个主要算法进行具体的描述.

表 1 符号说明

Table 1 Instructions of symbol

符号	表示
I_P, I_E	倒排索引、安全索引
GMM, IMM	multi-map 存储结构
W	关键字集合
tag_{id}	对应每个关键字生成的标签
Btag	块标签
$\#tl(\omega)$	关键字的数量
B, b	分块参数
$\mathrm{GMM}_E, \mathrm{IMM}_E$	加密之后的倒排索引
D_P, D_E	存储两个关键字交集加密倒排索引的字典、加密字典
K	令牌生成密钥
K_g, K_d, K_ω	用于倒排索引、字典、交集倒排索引加密的密钥
tk	布尔表达式搜索令牌
gtk, dtk, Itk	单关键字令牌、字典令牌、交集令牌
Bw	布尔搜索表达式
Δ	关键字析取表达式
R	查询返回的结果集

2.1 初始化算法

初始化算法 $\mathrm{Init}(1^k, I_P)$:数据拥有者执行该算法. 算法的主要工作是对文件预处理构建完成的明文倒排索引 I_P 进行加密、关键字交集索引计算、分块处理构建安全索引 I_E,并生成搜索令牌生成密钥 K. 具体算法执行步骤如下:

1) 利用安全参数 k,使用伪随机数生成器生成随机数 K_1, K_2 作为随机函数的密钥.

2) 初始化字典结构 D_P 和 multi-map 结构 GMM.

3) 生成单关键字的加密倒排索引和关键字交集加密倒排索引. 对关键字集合 W 中的每一个关键字 ω,执行如下操作:

①使用伪随机函数 F 为包含关键字 ω 的每一个文件计算标识符加密标签 tag_{id}.

②根据用户定义的分块参数 B, b,将包含关键字 ω 的文件标识符加密标签 $(\mathrm{tag}_{id})_{id \in I_P(\omega)}$ 分块

存入 GMM[ω].

③初始化一个#tl(ω)大小的 multi-map 结构 IMM,计算关键字 ω 与 tl(ω)中每一个关键字 v 的文件标识符交集,为交集中每一个文件计算标识符加密标签 tag_{id},分块存入 IMM[v]. 应用 multi-map 加密算法加密生成关键字 ω 的交集安全所引 IMM_{E} 和密钥 K_ω,将安全索引 IMM_{E} 存入字典 $D_{\text{P}}[\omega]$.

4）应用字典加密算法加密字典 D_{P},生成关键字交集安全索引字典 D_{E} 和密钥 K_{d}.

5）应用 multi-map 加密算法加密 GMM 生成单关键字安全索引 GMM_{E} 和密钥 K_{g}.

6）生成完整的安全索引 I_{E} 和令牌生成密钥 K,并输出.

初始化算法伪代码：

01 sample $K_1, K_2 \xleftarrow{\$} \{0,1\}$;
02 create a dictionary D_{P} and a multi-map GMM;
03 for all $\omega \in \text{W}$:
04 for all $\text{id} \in I_{\text{P}}(\omega)$: $\text{tag}_{\text{id}} := \text{Enc}_{K_1}(\text{id}; F_{K_2}(\text{id} \| \omega))$;
05 $\text{Btag}_\omega := \text{Blockput}((\text{tag}_{\text{id}})_{\text{id} \in I_{\text{P}}(\omega)}, B, b)$;
06 $\text{GMM}[\omega] := \text{Btag}_\omega$;
07 create a multi-map IMM of size #tl(ω);
08 for all $v \in \text{tl}(\omega)$:
09 $\text{SI} = \text{Intersection}(I_{\text{P}}(v), I_{\text{P}}(\omega))$;
10 for all $\text{id} \in \text{SI}$: $\text{tag}_{\text{id}} := \text{Enc}_{K_1}(\text{id}; F_{K_2}(\text{id} \| \omega))$;
11 $\text{Btag}_v := \text{Blockput}((\text{tag}_{\text{id}})_{\text{id} \in I_{\text{P}}(v) \cap I_{\text{P}}(\omega)}, B, b)$;
12 $\text{IMM}[v] := \text{Btag}_v$;
13 $(K_\omega, \text{IMM}_{\text{E}}) \leftarrow \sum_{\text{MM}} \text{Init}(1^k, \text{IMM})$;
14 $D_{\text{P}}[\omega] := \text{IMM}_{\text{E}}$;
15 $(K_{\text{d}}, D_{\text{E}}) \leftarrow \sum_{\text{DX}} \text{Init}(1^k, D_{\text{P}})$;
16 $(K_{\text{g}}, \text{GMM}_{\text{E}}) \leftarrow \sum_{\text{MM}} \text{Init}(1^k, \text{GMM})$;
17 $K = (K_{\text{g}}, K_{\text{d}}, \{K_\omega\}_{\omega \in W})$;
18 $I_{\text{E}} = (\text{GMM}_{\text{E}}, D_{\text{E}})$;
19 ouput(K, I_{E}).

2.2　令牌生成算法

令牌生成算法 TokenGen(K,Bw)：数据拥有者执行该算法. 解析用户输入的布尔搜索表达式 Bw,利用初始化生成的令牌生成密钥 K,生成对应的搜索令牌 tk. 具体算法执行步骤如下：

1）解析密钥 K,得到 GMM_{E} 加密密钥 K_{g},D_{E} 密钥加密 $K_{\text{d}}, D_{\text{E}}$ 中 IMM_{E} 加密密钥 K_ω.

2）解析布尔搜索表达式 Bw,转换成合取范式(conjunctive normal form ,CNF)$\Delta_1 \wedge \cdots \wedge \Delta_l$. Δ_l 为搜索关键字的析取范式(disjunctive normal form ,DNF)$\Delta_i = \omega_{i,1} \vee \cdots \vee \omega_{i,q}$.

3）生成搜索 Δ_1 的令牌 tk_1,具体过程如下：

①从关键字 $\omega_{1,1}$ 至 $\omega_{1,q-1}$：生成 $\omega_{1,i}$ 对应的 GMM_{E} 搜索令牌 $\text{gtk}_{1,i}$,生成 $\omega_{1,i}$ 对应的 D_{E} 搜索令牌 $\text{dtk}_{1,i}$,生成 $\omega_{1,i+1}$ 到 $\omega_{1,\#\Delta_1}$ 中 $\omega_{1,j}$ 与 $\omega_{1,i}$ 的交集对应的 IMM_{E} 搜索令牌 $\text{Itk}_{1,i,j}$. 合并生成 $\omega_{1,i}$ 对应的交集搜索令牌集合 $\text{tk}_{1,i} = (\text{dtk}_{1,i}, \text{gtk}_{1,i}, \text{Itk}_{1,i,i+1}, \cdots, \text{Itk}_{1,i,\#\Delta_1})$.

②生成 Δ_1 中最后一个关键字 $\omega_{1,q}$ 对应的 GMM_{E} 搜索令牌 $\text{gtk}_{1,q}$.

③生成 Δ_1 的搜索令牌 $\text{tk}_1 = (\text{tk}_{1,1}, \cdots, \text{tk}_{1,q-1}, \text{gtk}_{1,q})$.

4）生成从 Δ_2 到 Δ_l 对应的搜索令牌 $\text{tk}_2, \cdots, \text{tk}_l$,具体过程执行如下：

①生成 Δ_i 中关键字 $\omega_{i,j}$ 和 Δ_1 中关键字 $\omega_{i,s}$ 的交集搜索令牌 $\text{Itk}_{s,i,j}$,组合生成 Δ_i 中关键字 $\omega_{i,j}$ 与 Δ_l 的交集搜索令牌 $\text{Itk}_{i,j} = (\text{Itk}_{1,i,j}, \cdots, \text{Itk}_{q,i,j})$.

②将 Δ_i 中关键字 $\omega_{i,j}$ 与 Δ_1 的交集搜索令牌组合生成 Δ_i 与 Δ_1 的交集搜索令牌 $\text{tk}_i = (\text{Itk}_{i,1}, \cdots, \text{Itk}_{i,l})$,也可称之为 Δ_i 的搜索令牌.

5）将 Δ_1 到 Δ_l 的令牌组合生成总搜索令牌 $\text{tk} = (\text{tk}_1, \cdots, \text{tk}_l)$,并输出.

令牌生成算法伪代码：

01 convert K to$(K_{\text{g}}, K_{\text{d}}, \{K_\omega\}_{\omega \in W})$;
02 convert Bw to$(\Delta_1 \wedge \cdots \wedge \Delta_l)$ where for $i \in [l]$, $\Delta_i = (\omega_{i,1} \vee \cdots \vee \omega_{i,q})$;
03 for $1 \leqslant i \leqslant [q-1]$:
04 $\text{gtk}_{1,i} \leftarrow \sum_{\text{MM}} \text{TokenGen}(K_{\text{g}}, \omega_{1,i})$;
05 $\text{dtk}_{1,i} \leftarrow \sum_{\text{DX}} \text{TokenGen}(K_{\text{d}}, \omega_{1,i})$;
06 for $i+1 \leqslant j \leqslant \#\Delta_1$: $\text{Itk}_{1,i,j} \leftarrow \sum_{\text{MM}} \text{TokenGen}(K_{\omega_{1,i}}, \omega_{1,j})$;
07 $\text{tk}_{1,i} = (\text{dtk}_{1,i}, \text{gtk}_{1,i}, \text{Itk}_{1,i,i+1}, \cdots, \text{Itk}_{1,i,\#\Delta_1})$;
08 $\text{gtk}_{1,q} \leftarrow \sum_{\text{MM}} \text{TokenGen}(K_{1,q}, \omega_{1,q})$;
09 $\text{tk}_1 = (\text{tk}_{1,1}, \cdots, \text{tk}_{1,q-1}, \text{gtk}_{1,q})$;
10 for $2 \leqslant i \leqslant l$ and $j \in [q]$:
11 for $1 \leqslant s \leqslant q$: $\text{Itk}_{s,i,j} \leftarrow \sum_{\text{MM}} \text{TokenGen}(K_{\omega_{1,s}}, \omega_{i,j})$;

$\mathrm{Itk}_{i,j} = (\mathrm{Itk}_{1,i,j}, \cdots, \mathrm{Itk}_{q,i,j})$;
$\mathrm{tk}_i = (\mathrm{Itk}_{i,1}, \cdots, \mathrm{Itk}_{i,l})$;
output $\mathrm{tk} = (\mathrm{tk}_1, \cdots, \mathrm{tk}_l)$.

2.3 搜索算法

搜索算法 Search($\mathrm{tk}, I_{\mathrm{E}}$):服务器执行该算法.解析数据拥有者端发送的搜索令牌 tk,在安全索引 I_{E} 中搜索存储于服务器端的加密文件的索引信息.具体算法执行步骤如下:

1)解析 I_{E} 得到单关键字安全索引 $\mathrm{GMM}_{\mathrm{E}}$ 和关键字交集安全索引字典 D_{E}.

2)解析令牌 tk,得到每一个析取范式 Δ_i 对应的索搜令牌 tk_i.

3)根据令牌 tk_1 搜索获得关键字析取范式 Δ_1 对应的文件标签集合 R_1 作为第一轮搜索结果,执行如下操作:

①从 $\omega_{1,1}$ 到 $\omega_{1,q-1}$ 在 $\mathrm{GMM}_{\mathrm{E}}$ 中搜索并分层按块提取 $\omega_{1,i}$ 的文件标识符集合 $T_{1,i}$,依次减去从 $D_{\mathrm{E}}[\omega_{1,i}]$ 对应的 $\mathrm{IMM}_{\mathrm{E}}$ 搜索得到的 $\omega_{1,i}$ 和 $\omega_{1,i+1}$ 到 $\omega_{1,q}$ 的交集 T'.

②从 $\mathrm{GMM}_{\mathrm{E}}$ 搜索并分层按块提取 $\omega_{1,q}$ 的文件标识符集合 $T_{1,q}$.

③计算 $T_{1,1}$ 到 $T_{1,q}$ 的并集,得到 Δ_1 的搜索结果 $R_1 = \cup_{i \in |q|} T_{1,i}$.

4)从 Δ_2 到 Δ_l,对每一个 Δ_i 计算并保留上一轮搜索结果集合 R_{i-1} 与 Δ_i 中关键字 $\omega_{i,j}$ 和 Δ_1 中关键字 $\omega_{i,s}$ 的关键字标识符交集再计算交集.执行如下操作:

①搜索并分层按块提取 Δ_i 中关键字 $\omega_{i,j}$ 和 Δ_l 中关键字 $\omega_{i,s}$ 的关键字标识符交集.

②计算并保留 R_{i-1} 与 Δ_i 中关键字 $\omega_{i,j}$ 和 Δ_l 中关键字 $\omega_{i,s}$ 的关键字标识符交集的交集.

5)最后一轮搜索结果集合 R_l 即为最终的搜索结果.输出最终搜索结果 R.

搜索算法伪代码:

convert I_{E} to $(\mathrm{GMM}_{\mathrm{E}}, D_{\mathrm{E}})$;
convert tk to $(\mathrm{tk}_1, \cdots, \mathrm{tk}_l)$;
Search from I_{E} with Δ_1's token tk_1:
  convert tk_1 as $\mathrm{tk} = (\mathrm{tk}_{1,1}, \cdots, \mathrm{tk}_{1,q-1}, \mathrm{gtk}_{1,q})$;
  for $1 \leqslant i \leqslant [q-1]$:
    convert $\mathrm{tk}_{1,i}$ as $(\mathrm{dtk}_{1,i}, \mathrm{gtk}_{1,i}, \mathrm{Itk}_{1,i+1}, \cdots, \mathrm{Itk}_{1,q})$;
    Btag$\leftarrow \sum_{\mathrm{MM}} \mathrm{Search}(\mathrm{GMM}_{\mathrm{E}}, \mathrm{gtk}_{1,i})$;
    $T_{1,i} \leftarrow$ Blockget(Btag);
    $\mathrm{IMM}_{\mathrm{E}} \leftarrow \sum_{\mathrm{DX}} \mathrm{Search}(D_{\mathrm{E}}, \mathrm{dtk}_{1,i})$;
    for $i+1 \leqslant j \leqslant q$:
      Btag$\leftarrow \sum_{\mathrm{MM}} \mathrm{Search}(\mathrm{IMM}_{\mathrm{E}}, \mathrm{Itk}_{1,i})$;
    $T' \leftarrow$ Blockget(Btag);
    $T_{1,i} = T_{1,i} \backslash T'$;
  Btag$\leftarrow (\sum_{\mathrm{MM}} \mathrm{Search}(\mathrm{GMM}_{\mathrm{E}}, \mathrm{gtk}_{1,q}))$;
  $T_{1,q} \leftarrow$ Blockget(Btag);
  $R_1 = \cup_{i \in |q|} T_{1,i}$;
for $2 \leqslant i \leqslant l$:
  create an empty set R_i;
  convert tk_i as $(\mathrm{Itk}_{i,1}, \cdots, \mathrm{Itk}_{i,q})$;
  for all $j \in [q]$:
    $\mathrm{IMM}_{\mathrm{E}} \leftarrow \sum_{\mathrm{DX}} \mathrm{Search}(D_{\mathrm{E}}, \mathrm{dtk}_{1,j})$;
    convert $\mathrm{Itk}_{i,j}$ to $(\mathrm{Itk}_{1,i,j}, \cdots, \mathrm{Itk}_{q,i,j})$;
    for $1 \leqslant s \leqslant q$:
      Btag$\leftarrow \sum_{\mathrm{MM}} \mathrm{Search}(\mathrm{IMM}_{\mathrm{E}}, \mathrm{Itk}_{s,i,j})$;
      $R' \leftarrow$ Blockget(Btag);
      $R_i = R_i \cup (R_{i-1} \cap R')$;
output R_l.

3 原型系统设计与实现

基于 BSSEVD 方案,大型数据集下支持布尔搜索的可搜索加密原型系统由文件预处理模块、初始化模块和搜索模块所组成.按照最终实现的系统功能,具体可分为 8 个子模块.系统功能模块结构如图 2 所示.

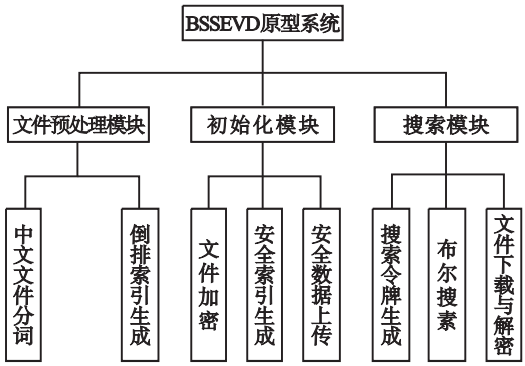

图 2 系统功能模块结构图

Fig. 2 System function module structure diagram

1)文件预处理模块包括中文文件分词模块、倒排索引生成模块.

中文文件分词模块:运行于客户端,实现对中文文件关键字的提取.

倒排索引生成模块:运行于客户端,根据关键字集合、文件集合,生成倒排索引.

2）初始化模块包括文件加密模块、安全索引生成模块、安全数据上传模块.

文件加密模块：运行于客户端，使用对称加密算法对上传云服务器的文件加密.

安全索引生成模块：运行于客户端，处理倒排索引，生成上传云服务器的安全索引和用于令牌生成的密钥.

安全数据上传模块：同时运行于客户和服务器两端，上传加密文件和安全索引，并存储于云服务器.

3）关键字搜索模块包括搜索令牌生成模块、搜索模块和文件下载与解密模块.

搜索令牌生成模块：运行于客户端，解析用户输入的布尔搜索表达式，并生成对应的搜索令牌，将令牌上传至服务器端，进行搜索操作.

搜索模块：运行于服务器端，接收客户端上传的搜索令牌，利用令牌来进行多关键字的布尔搜索.

文件下载与解密模块：同时运行于客户端和服务器两端，将搜索结果返回客户端，根据用户选择下载加密文件，并解密.

本原型采用的系统开发语言为 Java，版本号 1.7.0_75. 在本系统的实现过程中，使用开源 Java 函数库 Bouncy Castle 提供的算法实现加解密；使用开源的跨平台的 Apache POI 提供 API 实现对 Microsoft Office 格式档案读写；使用 Apache 软件基金会开放源代码的全文检索引擎工具包 Lucene 实现文件分词功能；使用 Google Guava Collections 设计实现安全索引数据结构.

4　系统测试与性能分析

本原型系统在由 2 台实体机构建的实验环境中进行性能测试，实体机硬件配置：Intel(R) Core (TM) i5 CPU 9400 @ 2.9 Hz 处理器，16 GB 内存；实体机软件环境：Windows 10 64 位操作系统. 测试数据包含中文数据和英文数据. 中文数据采用搜狗实验室(SougouLabs)提供的新闻数据，英文数据采用 Enron 公开邮件数据.

通过实验，测试原型系统对不同大小的安全索引(反映不同规模数据集)进行同一布尔搜索表达式的搜索时间，找出搜索时间与安全索引规模之间的关系，分析本方案中新的安全索引存储结构带来的搜索性能上的影响. 实验中，分别在 100 KB 到 600 KB 六个不同大小安全索引上对同一搜索布尔表达式进行查询.

实验结果如图 3 所示，当安全索引增大时，搜索时间也相应变长. 但应当注意到，当安全索引大小超过 200 KB 之后，本方案的搜索时间开始呈亚线性增长. 这正是使用分层按块存储数据结构优化传统 SSE 方案的安全索引结构带来的积极影响. 本方案在安全索引尺寸较小时没有优势甚至搜索性能会略差于传统 SSE 方案，但当安全索引规模超过某个阈值之后，本方案表现出其优势，安全索引尺寸越大优势越明显. 显然，本方案更适用于大规模数据集的可搜索加密.

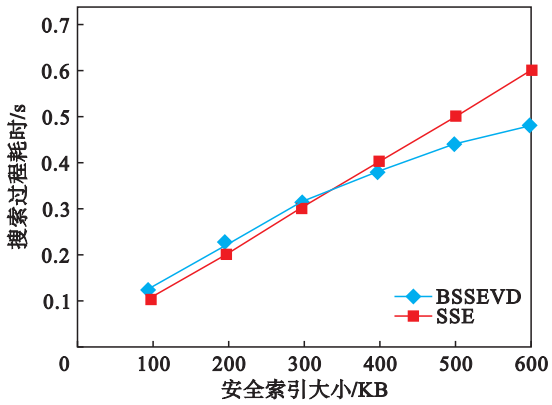

图 3　安全索引大小对搜索耗时的影响

Fig. 3　The effect of different security index sizes on search time

5　结　　语

本文在传统可搜索加密方案基础上，通过应用三层间接寻址块状存储安全索引结构优化加密倒排索引；通过增加关键字交集加密倒排索引的方法解决多关键字布尔搜索带来的泄露问题，提出一种满足动态自适应安全的大型数据集下支持布尔搜索的可搜索加密方案 BSSEVD，并设计实现其原型系统. 在满足动态自适应安全的前提下，实现多关键字布尔搜索，并实现在大型数据集下亚线性搜索效率.

参考文献：

[1] Wang Q, He M, Du M, et al. Searchable encryption over feature-rich data [J]. *IEEE Transactions on Dependable & Secure Computing*, 2018, 12(3): 496 – 510.

[2] Yu Y, Au M H A, Ateniese G, et al. Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage [J]. *IEEE Transactions on Information Forensics & Security*, 2017, 12(4): 767 – 778.

[3] Song D X, Wagner D, Perrig A. Practical techniques for searches on encrypted data [C]//IEEE Symposium on Security & Privacy. Berkeley, 2000: 44 – 55.

[4] Goh E J. Secure indexes [EB/OL]. (2004 – 03 – 16)

[2019-12-18]. https://eprint.iacr.org/2003/216.pdf.

[5] Chang Y C, Mitzenmacher M. Privacy preserving keyword searches on remote encrypted data [C]//Applied Cryptography and Network Security. Berlin: Springer-Verlag, 2005: 442-455.

[6] Stefanov E, Papamanthou C, Shi E. Practical dynamic searchable encryption with small leakage[C]//Network and Distributed System Security Symposium. San Diego, 2014: 1-15.

[7] Kamara S, Moataz T. Boolean searchable symmetric encryption with worst-case sub-linear complexity [C]// Advances in Cryptology-EUROCRYPT 2017. Paris, 2017: 99-124.

[8] Boneh D, Crescenzo G D, Ostrovsky R, et al. Public key encryption with keyword search [J]. *Lecture Notes in Computer Science*, 2003, 49(16): 506-522.

[9] Bellare M, Boldyreva A, O'Neill A. Deterministic and efficiently searchable encryption [C]//Advances in Cryptology-CRYPTO 2007. Berlin: Springer-Verlag, 2007: 535-552.

[10] Lin X, Lu R, Foxton K, et al. An efficient searchable encryption scheme and its application in network forensics [C]//Forensics in Telecommunications, Information and Multimedia. Berlin: Springer-Verlag, 2011: 66-78.

[11] Hwang Y H, Lee P J. Public key encryption with conjunctive keyword search and its extension to a multi-user system [C]//Pairing-Based Cryptography - Pairing 2007. Berlin: Springer-Verlag, 2007: 2-22.

[12] Li M, Yu S, Cao N, et al. Authorized private keyword search over encrypted data in cloud computing [C]// 2011 31st International Conference on Distributed Computing Systems. Minneapolis: IEEE, 2011: 383-392.

[13] Sun W, Yu S, Lou W, et al. Protecting your right: attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud [C]//IEEE INFOCOM 2014. Toronto: IEEE, 2014: 226-234.

[14] Cash D, Jaeger J, Jarecki S, et al. Dynamic searchable encryption in very-large databases: data structures and implementation[C]//2014 Network and Distributed System Security Symposium. San Diego: Internet Seciety, 2014: 853.