计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(10):287−294 [doi: 10.15888/j.cnki.csa.008120]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于语义分组的动态可搜索加密方案①

王泽贤, 汪学明

(贵州大学 计算机科学与技术学院, 贵阳 550025)
通讯作者: 汪学明, E-mail: xmwang1965@163.com

摘　要: 为满足用户对云端文档动态更新的需求, 支持动态更新的可搜索加密方案成为了研究热点. 但目前已知方案对于索引结构的更新多采用尾部直接插入的方法, 造成了新添加关键字和文档之间关联性的泄露. 为此本文提出一种基于语义分组的动态可搜索加密方案. 首先构建分组平衡二叉树作为索引结构, 通过语义分组减少搜索时访问的节点数, 提高搜索效率. 然后结合分区矩阵的思想, 在矩阵中添加虚拟关键字保证更新时的安全性. 最后通过形式化的证明分析了本文方案的安全性.

关键词: 可搜索加密; 多关键词排序搜索; 动态更新; 分组平衡二叉树

引用格式: 王泽贤,汪学明.基于语义分组的动态可搜索加密方案.计算机系统应用,2021,30(10):287–294. http://www.c-s-a.org.cn/1003-3254/8120.html

Dynamic Searchable Encryption Scheme Based on Semantic Grouping

WANG Ze-Xian, WANG Xue-Ming

(College of Computer Science and Technology, Guizhou University, Guiyang 550025, China)

Abstract: Searchable encryption scheme supporting dynamic update has become a research hotspot to meet the needs of users for dynamic update of cloud documents. However, most of the known schemes update the index structure by the direct insertion at the tail, which leads to the leakage of the relationship between the newly added keywords and the document. Therefore, this study proposes a dynamic searchable encryption scheme based on semantic grouping. Firstly, the balanced binary tree is constructed as the index structure, and the number of nodes is reduced by semantic grouping for search efficiency improvement. Then, in light of the idea of partition matrix, virtual keywords are added to the matrix to ensure the security of update. Finally, the security of the scheme is analyzed by formal proof.

Key words: searchable encryption; multi-keyword sorting retrieval; dynamic update; balanced binary tree

为了降低本地资源的计算开销, 越来越多的人选择将个人数据上传到不可信的云存储服务器中, 然而云服务器存在泄露用户隐私的诸多威胁. 权衡云环境既存在安全威胁又具备便利性的特点, 很多人将本地信息加密后在上传到云服务器中从而起到保护数据安全的作用. 随着用户和企业数据存储量逐年增长, 如果将全部数据不加区分地放在一起进行加密和搜索, 不仅增加加密复杂度, 而且影响搜索效率.

可搜索加密 (Searchable Encryption, SE) 技术[1] 是实现隐私数据加密上传到云端后搜索的重要技术. 对此国内外学者展开研究取得了许多成果. Song 等[2] 首先提出了一个数据共享者和一个查询用户模型下的对称可搜索加密方案, 该方案通过对所有密文进行线性扫描的方式完成搜索. 为了提升 SE 方案的可用性, 2011 年 Cao 等人[3] 首次提出了 MRSE 方案, 该方案成功解决了多关键字的排序问题, 但该方案的缺点是既不支

① 收稿时间: 2021-01-05; 修改时间: 2021-02-03; 采用时间: 2021-02-08

持相似性搜索查询且效率也较低. 随后 Xia 等[4] 设计了一种特殊的树状索引结构, 采用贪婪深度搜索算法提高效率, 同时用 KNN 技术加密文件和索引保证方案的安全性.

上述方案基本解决了 SSE 方案搜索需求的问题. 但是本地数据的大量产生, 用户急需对云端的文档进行更新从而保证文档的完整性. 这使得支持动态更新的 SE 方案成为新的研究热点. 然而 Zhang 等人[5] 在 2016 年发现文档更新时通过文件注入式攻击会泄露用户的隐私. 为解决这一问题 Bost 等人[6] 在 2017 年提出了支持前向安全和后向安全的 SSE 方案. 但是该方案只支持单关键字的搜索, 为了进一步提升动态可搜索加密的性能和效率国内外学者展开广泛研究并取得丰硕成果.

为应对恶意服务器环境, Wang 等人[7] 提出一种可验证的动态可搜索加密方案. 该方案构造了基于时间戳链表和 Merkle 树结果验证机制, 实现恶意服务器环境下的正确搜索.

为提高搜索效率, 2019 年 Xu 等人[8] 提出了基于 R-树动态范围搜索 SE 方案. 该方案采用 R-树索引结构显著提高搜索效率, 并提出一种新的访问控制策略 EGRQ 实现用户的细化访问.

为提高更新效率, 2020 年 Huang 等人[9] 提出了基于完美二叉树的前向安全 SE 方案. 作者通过预估更新大小, 构建完美索引树结构, 避免更新时重构节点造成的本地开销. Kermanshahi 等人[10] 首次提出了内容隐私的定义. 该方案在更新时, 本地客户端首先构建更新令牌 (包含更新的位置和更新内容) 并加密上传到云端. 具体的更新过程由服务器完成, 从而降低本地开销. 孙晓玲等人[11] 提出了基于三链表索引结构的高效动态可搜索加密方案. 该方案通过每次搜索操作更新查询链表和删除链表的策略有效的提高了更新的效率.

在多用户场景下, Wang 等人[12] 通过引入半可信的第三方服务器用于记录关键词的状态信息, 提出了一种支持多用户访问的前向安全动态可搜索加密方案. 但该方案存在无法抵抗重放攻击和窃听攻击等安全缺陷. 随后卢冰洁等人[13] 对此做出了改进, 通过引入完全可信第三方服务和构建新的索引结构, 提高了方案的安全性和删除效率.

上述方案虽然在性能和效率方面取得巨大提升, 但对索引结构的更新大多采用在已有索引的尾部直接添加的方法, 泄露了新添加关键字和已有文档的信息, 以及新添加关键字在字典中排列的位置. 这造成更新时极大的安全隐患.

为此本文提出了基于语义分组的动态更新可搜索加密方案 (Scheme of Dynamic Searchable Encryption based on semantic grouping, SDSE). 首先通过 TF-IDF 和空间向量模型构建节点向量, 根据计算节点的相关性进行语义分组构建索引树. 然后向节点向量中添加虚拟关键字完成向量扩充, 保证字典排布的安全性. 最后结合分区矩阵思想提出一种更新方法. 实现了在原有关键字字典的基础上能够动态更新云端文件.

1 系统模型

(1) 数据拥有者 DO 创建关键词字典、加密文档、构建安全索引, 将它们上传到云服务器.

(2) 用户 DU 生成查询陷门, 并将陷门和想要返回的结果数量 K 发送到云服务器, 收到云端计算完成的最相关的 K 个结果文档后对其进行解密.

(3) 云服务器收到 DU 的搜索请求后, CS 对加密索引进行搜索, 并计算返回用户最希望的前 K 个结果文档.系统模型图如图 1 所示.

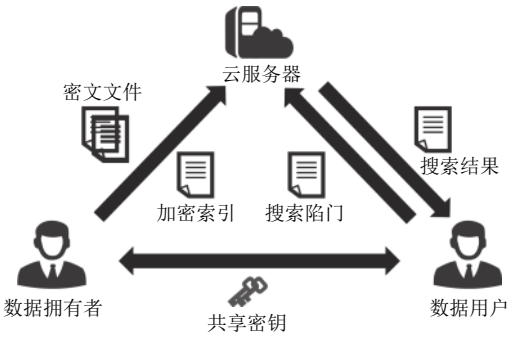

图 1 系统模型图

2 相关工作

2.1 符号定义

F: 表示明文文件集合 $F=\{f_1, f_2, \cdots, f_n\}$.

C: 表示对应明文 F 的密文集合 $C=\{c_1, c_2, \cdots, c_n\}$.

W: 表示关键词字典 $W=\{w_1, w_2, \cdots, w_m\}$.

Q: 查询向量.

T: 未加密的索引向量.

I: 加密的索引向量.

T_w: 搜索陷门.

K: 返回结果数.

sk: 对称密钥.

F_{up}: 更新文档集.

W_{up}: 更新关键词集.

SK: 安全密钥对$\{S, M_1, M_2\}$.

S: $m+d$ 阶随机二进制向量.

M_1, M_2: $(m+d)\times(m+d)$ 阶可逆矩阵.

$u.ID$: 树节点 u 的唯一标识.

$u.FID$: 叶子结点指向的文件的标识. 如果 u 为非叶子结点, 则 $u.FID$=none.

$u.V$: 表示结点 u 所包含的 m 长的向量, 如果 u 是叶节点, 则 $u.V$ 为文件 $u.FID$ 的加权索引. 如果 u 是非叶子结点, 则 $u.V$ 表示为以 u 为根的子树中所有结点向量对应维度的最大值 $u.V[t]$= max $(u_1.V[t], u_2.V[t],\cdots)$ $t\in\{1, 2, \cdots, m\}$.

Left, right: 当节点 u 为非叶子节点时, left 为其左孩子节点, right 为其右孩子节点.

Tu: 表示树节点 u 的加权索引向量.

ResultNodeSet: 表示一个分组集合.

2.2 安全性定义

本文方案的安全性主要有如下两个概率游戏验证 $Real_A(k)$, $Ideal_{A,S}(k)$, 其中, A 表示敌手, S 表示模拟器. 其中, $Real_A(k)$由 SDSE 执行, $Ideal_{A,S}(k)$是通过 SDSE 方案的泄露信息进行模拟. 我们把 SDSE 方案的泄露信息定义为如下 3 个泄露函数 L_1, L_2, L_3.

定义 1. 泄露函数 L_1, L_2, L_3

1) $L_1(F)$以文档作为输入, L_1 输出关键词的个数 m, 文档数 n, 每个文档的标识以及文档f_j包含关键词的个数, 关键字字典 W. 具体形式如下:

$$L_1(F,I) = (n, m, id_1 \cdots id_m, |f_1| \cdots |f_m|, W) \tag{1}$$

2) $L_2(F,I,w,t)$以文档, 索引, 在时间节点 t 的搜索关键词 w 作为输入, L_2输出搜索模式和访问模式$s(I,q,t)$和$a(F,I,w,t)$, 具体形式如下:

$$L_2(F,I,w,t) = (s(I,q,t), a(F,I,w,t)) \tag{2}$$

3) $L_3(f_{up})$以更新文档作为输入, L_3输出更新文档标识, 文档f_j包含的关键词个数, 新的关键字字典. 具体形式如下:

$$L_3(f) = (id_i \cdots id_j, |f_i| \cdots |f_j|, W_{up}) \tag{3}$$

定义 2. $Real_A(k), Ideal_{A,S}(k)$

$Real_A(k)$: 挑战者执行$srtup(1^l)$产生对称密钥 sk 和安全密钥 SK. 然后敌手 A 选择文档集, 挑战者执行 $Index(F, sk, SK)$生成密文文档和索引并发送给敌手 A. 敌手在多项式时间内进行普通搜索和更新搜索. 假设查询关键字为 w, 挑战者执行$Trapdoor(q_w, SK)$产生搜索陷门. 通过执行$update(F_{up})$得到最新关键字字典. 最后, A 返回一个比特 b 作为实验的输出结果.

$Ideal_{A,S}(k)$: 敌手 A 进行多次多项式时间内的搜索 $q\in(w_i, f_j, f_z)$, 每一次的搜索过程中, 模拟器 S 都会得到如下的泄露$L_1L_2L_3$. 模拟器 S 根据泄露信息执行 SDSE, 并返回相应的密文文档C^*, 索引I^*, 陷门T^*, 更新字典W^*_{up}. 最后敌手 A 返回一个比特 b 作为实验的输出结果.

如果方案 SDSE 对于自适应关键字搜索是安全的, 那么对于任意多项式时间内的敌手, 一定存在模拟器 S 使得式 (4) 成立:

$$|pr[Real_A(k) = 1] - pr[Ideal_{A,s}(k) = 1]| \le neg(k) \tag{4}$$

2.3 TF-IDF 和空间向量模型

空间向量模型和 TF-IDF 规则是解决传统搜索领域的有效手段, 通过建立关键词字典集合与查询文档之间的关系, 实现多关键词的有效排序搜索. $TF_{fj,wi}$ 表示在文档f_j 中关键词 w_i 出现的频率. 令 IDF_{wi} 表示含有关键词 w_i 的文件在文档总集中出现的频率. $TF_{fj,wi}$ 和 IDF_{wi} 分别采用式 (5) 和式 (6) 计算得到:

$$IDF_{wi} = \ln(1 + n/n_{wi}) \tag{5}$$

$$TF_{fj,wi} = 1 + \ln N_{fj,wi} \tag{6}$$

其中, $TF_{fj,wi}$ 为f_j 中关键字 w_i 的个数; n 为 F 中的文件总数; n_{wi} 为包含关键词 w_i 的文件数. 一般采用归一化的词频 $TF'_{fj,wi}$ 和逆文档频率 IDF'_{wi}, 即:

$$TF'_{fj,wi} = \frac{TF_{fj,wi}}{\sqrt{\sum_{wi\in W}(TF_{fj,wi})^2}} \tag{7}$$

$$IDF'_{wi} = \frac{IDF_{wi}}{\sqrt{\sum_{wi\in W_s}(IDF_{wi})^2}} \tag{8}$$

向量空间模型将文档和查询都看成是关键词的集合, 是关键词的有序排列. 我们可以根据倒排序列构建关键词的字典, 用二进制向量来表示文档和查询. 把查询抽象化为数学形式表示, 用 0 和 1 表示查询文档中是否存在词典对应位置的关键词.

2.4 ASPE 加密算法

ASPS 算法由 Wang 等人[14] 在 2009 年 Sigmoid 上提出, ASPE 算法核心功能其实是实现了两个加密向量之间的安全内积计算.

定义 3. 非对称保内积加密. 对于加密算法 $E()$, 查询点记为 q, 数据 D 中任意点记为 P_i, 如果 $E()$ 满足下列两个条件, 则称之为非对称的保内积加密算法:

$$\begin{cases} d(E(p_i),E(q)) = d(p_i,q) \\ d(E(p_i),E(q_j)) \neq d(p_i,q_,) \end{cases}, \quad p_i,p_j \in D \tag{9}$$

3 基于语义分组的动态可搜索加密方案定义

SDSE 方案由如下 5 个算法组成: (初始化 (*Setup*)、构建索引 ($Index$)、构建陷门 ($Trapdoor$)、查询 (*Search*) , 更新 (*Update*) 具体定义如下所示:

(1) $Setup(1^l) \rightarrow (sk,SK)$: 初始化阶段, 数据拥有者根据安全参数$l$得到密钥组 ($sk$, SK), 其中, sk 是加密文档的对称密钥. SK 是用于加密索引和查询向量的密钥.

(2) $Index(F,sk,SK) \rightarrow I$: 索引构建阶段, 数据所用者根据明文文档 F 构建关键词字典 W. 并以每个文档为叶子节点构建索引向量, 通过计算节点相似性组成明文索引结构. 最后密钥 SK 加密后上传到云端.

(3) $Trapdoor(SK,W) \rightarrow T_w$: 陷门生成阶段, 数据用户根据想要查询的关键词 w 构建搜索向量 q_w, 并用密钥 SK 加密后生成陷门 T_w 上传到云端.

(4) $Search(T_w,I,K) \rightarrow R$: 搜索阶段, 云服务器收到用户的搜索陷门 T_w 后与云端存储的索引结构 I 计算, 返回得分最高的 K 个结果.

(5) $Update(F_{up},w_{up})$更新阶段, 数据拥有者从更新文档 f_{up} 中得出新的关键词 w_{up}, 并为其重构索引结构, 并把新的索引结构和文档加密后上传到原始云端存储中. 得到新的密文文档集和索引结构.

4 基于语义分组的动态可搜索加密方案

4.1 方案构造

(1) $Setup(1^l) \rightarrow (sk,SK)$

初始化阶段 DO 根据关键词字典产生用于拆分索引和查询向量的 $m+d$ 维二进制向量 S, 以及两个 $(m+d)\times(m+d)$ 阶可逆矩阵$\{M_1, M_2\}$用于加密拆分后的向量. 对称密钥 sk 用于加密明文文档, 安全密钥 $SK = \{S, M_1, M_2\}$用于加密安全索引和查询向量.

(2) $Index(F,sk,SK) \rightarrow I$

构建索引分为如下 5 个步骤.

1) 抽取关键词 : 数据拥有者对文档集 F 抽取关键词得到关键字字典 $W=\{w_1, w_2,\cdots, w_n\}$.

2) 构建明文形式的二进制索引: 根据空间向量模型数据拥有者为文档 f_j, 生成二进制向量 T_j. T_j 的每个比特表示 f_j 中相应关键词存在与否.

3) 生成加权索引: 本文中关键词的权重由关键词出现的概率以及它出现的位置决定. TF-IDF 的方法来计算关键词和文档之间的相关度. 在根据关键词在文中所处的位置分别赋予不同的权重, 本文把关键词的位置分为如下 3 类: 标题, 摘要, 正文. 分别使得标题占 0.6, 摘要占 0.3, 正文占 0.1 的权重. 假设某文档 f 中关键词 w 出现在标题和正文中, 我们用如下公式来表示它的权重: X=0.6×1+0.3×0+0.1×1=0.7, 采用 TF-IDF 权值计算方法来计算关键词相关度分数:

$$Y = \frac{1}{|f|} \times (1 + \ln tf_{i,j}) \times \ln\left(1 + \frac{N}{df_i}\right) \tag{10}$$

其中, $|f|$表示文档的长度, $tf_{i,j}$ 表示关键词 i 在文档 f_j 中出现的概率, df_i 表示包含关键词 i 的文档数量.

4) 构建索引树: 本方案设计了一种新的索引结构-分组平衡二叉树, 构建这种树型索引时, 以自下而上的策略对每一层的树节点进行分组, 使得主题越接近的文档尽可能地分布于索引树的同一分支.

所有叶子结点生成完毕后, 采用自下而上的策略, 基于分组构造索引树. 每一层的非叶子结点都是由分组算法生成, 把加权索引的余弦值较小的两个结点组成一对.

为完成节点分组, 设计了算法 1.

算法 1. 树节点分组算法

输入: 当前需要处理的节点集 PendingNodeSet
输出: 节点分组完的集合 ResultNodeSet

1. while PendingNodeSet 中的节点大于 1do
2. 在 PendingNodeSet 里边随机选择节点 u
3. 寻找一个非 u 的节点 t, 使得 cos(Tu, Tt) 的值最大
4. 将节点对 (u, t) 添加到 ResultNodeSet 集合
5. 将 u 和 t 从集合 PendingNodeSet 中移除
6. end while
7. if PendingNodeSet 非空 then
8. 将 PendingNodeSet 中唯一的节点插入到 ResultNodeSet 集合的末尾
9. end if
10. return ResultNodeSet

树节点分组完成后, 如果节点总数为 $2n$ 个, 那么 ResultNodeSet 中就有 n 个节点对; 如果节点总数为 $2n+1$ 个, 那么就有 n 个节点对和 1 个单节点. 然后在分组对的基础上构造一个新的节点作为他们的父节点, 这样反复迭代直到最终生成唯一的根节点.

由算法 1 构成的索引树如图 2 所示.

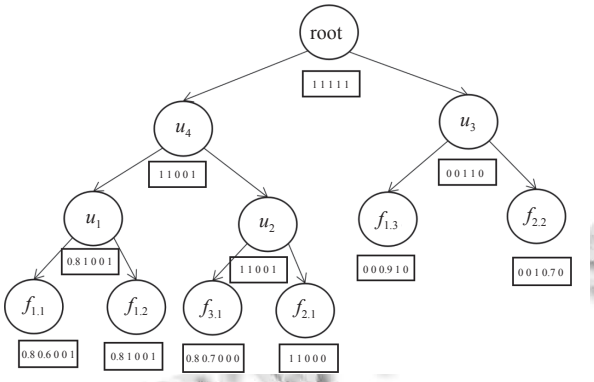

图 2 由 6 个文件、5 个关键词构成的索引树

5) 加密明文索引树: 索引树构建完成后把节点向量动态扩充为 $m+d$ 阶, 其中, d 为虚拟关键字数, $(m+1,\cdots, m+d)$ 的值选择 z 个置为 ε, $\varepsilon<(TF)_{\min}$, 其余为 0 $(z\in Z_q^*)$. 然后使用安全 ASPE 加密算法, 给加权索引树中的每个结点加密. 具体的操作是把 $u.V$ 分割成两个随机向量$\{u.V', u.V''\}$拆分思想如下:当 $S[i]=0$ 时, 置$u.V'=u.V''=u.V$; 当 $S[i]=1$ 时, 使得$u.V'+u.V''=u.V$. 再通过$\{M_1, M_2\}$对$\{u.V', u.V''\}$进行加密, 加密成如下形式$\{M_1^{\mathrm{T}}\cdot u.V', M_2^{\mathrm{T}}\cdot u.V''\}$待加密完所有的节点把加密索引树上传到云服务器.

(3) $Trapdoor(SK, W)\rightarrow T_w$

1) 首先根据查询的关键词 W_i 和关键词字典 W 生成 m 阶明文查询 Q, 然后对查询向量扩充为 $m+d$ 阶. 在 d 个虚拟的关键字中随机选择 z 个, 使其值置为 ε 其余的为 0. $(z<1/2d, z\in Z_q^*, \varepsilon<TF_{\min})$.

2) DO 通过 ASPE 加密算法对向量 Q 加密. 根据向量 S, 将扩展后的查询向量Q拆分为$\{Q', Q''\}$,与对 $u.V$ 的拆分方法不同: 当$S[i]=0$时, 置$Q'[i]+Q''[i]=Q[i]$. 当$S[i]=1$时, 置$Q'[i]=Q''[i]=Q[i]$. 再通过矩阵$\{M_1, M_2\}$对$\{Q', Q''\}$进行加密, 加密结果.

(4) $Search(T_w, I, K)\rightarrow R$

收到 DU 的搜索请求后, CS 使用深度贪婪优先算法在上传的加密安全索引树上进行搜索. 具体形式如算法 2.

算法 2. 深度贪婪优先搜索算法

输入: 索引树 t 的根节点和查询向量 Q
输出: top$K$ 的查询结果的文档

1. if u 不是叶子结点 then
2. if score(Iu, Q)>d (d 为前 k 个临界值) then
3. search (u.rchild)
4. search (u.lchild)
5. end if
6. else
7. if score(Iu, Q)>d then
8. 更新结果列表 result 和 d 的值
9. end if
10. end if

在分组平衡二叉树中是如何执行贪婪深度优先搜索的过程如图 3 所示.

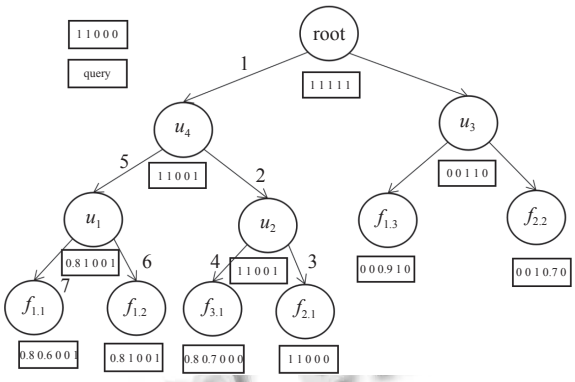

图 3 分组平衡二叉树查询

当计算加密索引和安全陷门相关后, 返回得分前 K 的文档, 用户使用对应密钥解密文档. 搜索完毕.

(5) $Update(F_{\mathrm{up}}, w_{\mathrm{up}})$

DO 根据本地文档更新需求, 对云端的文档进行更新. 为避免动态更新时泄露关键字和文档相关性信息, 本文采用分区矩阵的思想. 在更新关键字字典时同时添加 $l-1$ 个虚拟关键字 $(l\leftarrow Z_p^*)$, 增加了关键字字典添加关键字时的随机性. 通过构建两个 $l\times l$ 的可逆矩阵完成关键字字典的更新具体形式如下:

$$M_1'=\begin{pmatrix} M_1 & 0 \\ 0 & M_l \end{pmatrix}, M_2'=\begin{pmatrix} M_2 & 0 \\ 0 & M_l' \end{pmatrix} \quad (11)$$

为完成加密生成 $m+d+l$ 阶分割向量 s_1, 其在向量 s 的基础上得到 $s_1=(s, l)$. 然后对新矩阵$M_1'M_2'$进行加密. 具体形式如式 (12):

$$\begin{cases} M_1'^{\mathrm{T}} = \begin{pmatrix} M_1^{\mathrm{T}} & 0 \\ 0 & M_l^{\mathrm{T}} \end{pmatrix}, M_2'^{\mathrm{T}} = \begin{pmatrix} M_2^{\mathrm{T}} & 0 \\ 0 & M_2'^{\mathrm{T}} \end{pmatrix} \\ M_1'^{-1} = \begin{pmatrix} M_1^{-1} & 0 \\ 0 & M_l^{-1} \end{pmatrix}, M_1'^{-2} = \begin{pmatrix} M_1^{-2} & 0 \\ 0 & M_l'^{-1} \end{pmatrix} \end{cases} \tag{12}$$

分区矩阵保证了已有的节点向量不变的基础上, 仅对更新文档进行构建加密. 既不会泄露新添加关键字在字典中的分布, 也减少了更新时产生的计算开销.

4.2 方案正确性分析

本节进行证明即使矩阵扩展到 $m+d+l$ 阶, 仍然可以正常匹配. 并不会影响搜索结果的排序. 其中 ε 为添加虚拟关键字时产生的平衡参数. 即使搜索相同的关键字得分也存在差异从而保证了在搜索过程中索引和陷门的安全. 但由于 ε 的取值远小于 TF 的最小值, 故不影响相关性结果的正常排序. 以下为索引和陷门匹配的过程.

$$\begin{aligned} I \cdot T_w &= I' \cdot T_w' + I'' \cdot T_w'' \\ &= M_1^{\mathrm{T}} D' \cdot M_1^{-1} Q' + M_2^{\mathrm{T}} D'' \cdot M_2^{-1} Q'' \\ &= D'Q' + D''Q'' \\ &= DQ + \sum \varepsilon \end{aligned} \tag{13}$$

4.3 方案安全性分析

根据定义 1 可知, 本文方案执行搜索和更新时会泄露相关的信息. 在分析 SDSE 方案的安全性之前, 我们假设加密明文文档的对称加密算法是满足选择明文攻击 (CPA) 的.

根据定义 2 可得, 如果本文方案满足对于任意多项式时间内的敌手都能使得式 (14) 成立:

$$|pr[Real_A(k)=1] - pr[Ideal_{A,S}(k)=1]| \le neg(k) \tag{14}$$

那么我们认为本文方案在自适应不可区分的意义上是安全的. 敌手可以通过分析密钥、索引、密文、陷门的不相关、关键字字典来获得模拟游戏的胜利.

$$\begin{aligned} pr[Ideal_{A,S}(k)=1] &= 1/2 + Adv(A(key)) + Adv(A(I)) \\ &+ Adv(A(C)) + Adv(A(T_w)) + Adv(A(W))) \end{aligned} \tag{15}$$

具体的证明过程如下:

$Adv(A(SK, sk))$表示敌手 A 判断密钥和随机矩阵的优势. $Adv(A(I))$表示敌手 A 判断云端存储索引的优势. $Adv(A(C))$表示敌手 A 区分真实密文中加密文档的优势. $Adv(A(T_w))$表示敌手判断不同陷门之间相关性的优势. $Adv(A(W))$表示敌手 A 判断关键字字典中关键字分布的优势.

本文方案中的安全密钥是由两个 $(m+d)\times(m+d)$ 阶可逆矩阵和分割向量组成 $SK=\{S, M_1, M_2\}$. 可逆矩阵和分割向量 S 组成的密钥同随机数矩阵和一串随机二进制数组成的密钥在计算上是不可区分的, 概率可以忽略不计. 故式 (16) 成立.

$$\begin{aligned} Adv(A(key)) &= |pr[setup(1^l) \to key] - pr[random \to key^*]| \\ &\le neg1_1(k) \end{aligned} \tag{16}$$

本文方案中的索引是由关键字字典加虚拟关键字在 ASPE 加密算法加密后得到. 通过添加虚拟关键字模糊真实关键字的数量起到隐藏与关键字有关的标识符. 由于 ASPE 算法加密结果的不确定性使得即使得相同明文加密得出的密文也不同. 故在计算上也是不可区分的. 故式 (17) 成立:

$$\begin{aligned} Adv(A(I)) &= |pr[buildIndex(F, SK, sk) \to I] - pr[random \\ &\to I^*]| \le neg1_2(k) \end{aligned} \tag{17}$$

根据前提假设可知本文加密算法是满足 CPA 安全的, 故式 (18) 成立:

$$\begin{aligned} Adv(A(C)) &= |pr[Encrypt(F, sk) \to C] - pr[random \to C^*]| \\ &\le neg1_3(k) \end{aligned} \tag{18}$$

本文方案中陷门是由关键字请求向量加虚拟关键字构成. 即使搜索相同的关键字在不同的时间段产生的请求向量也是不同的. 然后通过 ASPE 加密算法进行加密更加无法判断. 故式 (19) 成立:

$$\begin{aligned} Adv(A(T_w)) &= |pr[Trapdoor(w, sk) \to T_w] \\ &- pr[random \to T_w^*]| \le neg1_4(k) \end{aligned} \tag{19}$$

本文方案中初始的关键字字典的安全性是由虚拟关键字和 ASPE 加密保证的. 着重介绍更新时关键字字典的安全性. 对于新产生的关键字通过添加虚拟关键字来模糊其在字典中的位置, 然后通过分区矩阵的思想使其添加到原始字典中. 这使得在判断新添加关键字在字典中的位置以及相关文档的概率可以忽略不计. 故式 (20) 成立:

$$\begin{aligned} Adv(A(W)) &= |pr[bulid(F) \to W] - pr[random \to W^*]| \\ &\le neg1_5(k) \end{aligned} \tag{20}$$

综上所述可得:

$$\begin{aligned} &pr[Ideal_{A,S}(k)=1] = 1/2 + Adv(A(key)) + Adv(A(I)) \\ &+ Adv(A(C)) + Adv(A(T_w)) + Adv(A(W))) \\ &\le 1/2 + neg1_1(k) + neg1_2(k) + neg1_3(k) \\ &+ neg1_4(k) + neg1_5(k) \end{aligned} \tag{21}$$

使得式 (22) 成立:

$$neg1(k)=neg1_1(k)+neg1_2(k)+neg1_3(k)+neg1_4(k)+neg1_5(k)|pr[Real_A(k)=1]-pr[Ideal_{A,s}(k)=1]|\le neg(k) \quad (22)$$

因此本文方案在泄露信息 L_1, L_2, L_3 的情况下是安全的.

4.4 方案效率分析

本节从构建加密索引、搜索、更新 3 个方面进行效率分析, 将本文方案与文献 [8–11,13] 方案进行对比.

在构建索引上, 本文方案和文献 [8–10] 所提方案是基于二叉树索引结构. 分析加密索引树效率时分 3 步; 构建未加密索引树需要经过生成叶子结点、结点分组和构建索引树 3 个步骤. 生成结点需要 O(z),因为关键词字典的大小为 m, 所以每个结点生成索引向量耗时 O(m). 计算关键词之间相关度时间复杂度为 O(m). 由于树节点分组算法的时间复杂度为 O(zm^2). 所以构建索引树时间复杂度为 O(zm^2). 二叉树中共有 z 个节点, 需要加密 O(z) 个节点, 向量分割时间复杂度为 O(m), 语义相似度矩阵的两次乘法时间复杂度为 O(m^2). 所以加密索引树需要耗时 O(zm^2).

在关键词搜索方面, 其搜索的时间取决于访问的叶子节点数, 假设实际访问的叶子节点数为 L 个, 显然 $L>k$ 与 k 呈正相关, 文献 [9] 中方案所构造的搜索的时间复杂度 O($L\log_2 n$). 但本文方案与文献 [9] 中方案的区别在于, 平衡分组二叉树把相关度高的结点都分到了同一子树下, 所以需要遍历搜索的叶子节点数会大大少于未分组的平衡二叉树, 从而得到更高的查询效率.

在文档更新方面. 本文采用分区矩阵思想进行更新, 保持原有字典不变的基础上通过构建 $l\times l$ 阶矩阵完成更新. L 为真实关键字和虚拟关键字的总数. 故时间复杂度为 O(l^2).

表 1 给出了本案与已有方案的效率分析对比. 其中, z 表示二叉树中节点总数, m 表示关键词字典大小, n 表示文档总数, L 表示二叉树索引中实际访问节点数, R 表示范围查询的半径, t 表示范围查询坐标位长, p 表示更新文档数, l 表示更新关键词数.

通过与最新成果对比可得, 本文在搜索和更新方面具备较高效率.

5 结束语

本文提出了一种基于语义分组的动态关键词可搜索加密方案. 提出平衡分组二叉树作为索引结构使得返回的文档更符合用户的请求. 结合分区矩阵思想进行文档更新. 在保证关键字字典安全性的同时, 减少了更新时的计算开销. 与现有的方案相比, 本文方案具备更高的搜索效率和更新效率. 但本文方案只考虑了单用户模型, 为更贴近云服务工作的实际情况. 接下来的工作把本文方案拓展到多用户场景下.

表 1 效率分析对比

方案	Index	Search	Update
文献[9]	O(zm)	O($L\log_2 n$)	O($pmL\log_2 n$)
文献[11]	O($3nm$)	O(nm)	O(pm)
文献[8]	O(zm^2)	O(m^4t^2)	O(pt)
文献[10]	O(zm^2)	O($n\log_2 R$)	O($2^t pm$)
文献[13]	O(nm)	O(nm)	O(pm)
本文	O(zm^2)	<O($L\log_2 n$)	O(l^2)

参考文献

1 吴志强, 李肯立, 郑蕙. 高效可扩展的对称密文检索架构. 通信学报, 2017, 38(8): 79–93. [doi: 10.11959/j.issn.1000-436x.2017166]

2 Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. Proceedings of 2000 IEEE Symposium on Security and Privacy. Berkeley: IEEE, 2000. 44–55.

3 Cao N, Wang C, Li M, *et al*. Privacy-preserving multi-keyword ranked search over encrypted cloud data. Proceedings of the 30th IEEE International Conference on Computer Communications. Shanghai: IEEE, 2011. 829–837.

4 Xia ZH, Chen L, Sun XM, *et al*. A multi-keyword ranked search over encrypted cloud data supporting semantic extension. International Journal of Multimedia and Ubiquitous Engineering, 2016, 11(8): 107–120. [doi: 10.14257/ijmue.2016.11.8.12]

5 Zhang YP, Katz J, Papamanthou C. All your queries are belong to us: the power of file-injection attacks on searchable encryption. Proceedings of the 25th USENIX Conference on Security Symposium. Austin: USENIX, 2016. 707–720.

6 Bost R, Minaud B, Ohrimenko O. Forward and backward private searchable encryption from constrained cryptographic primitives. Proceedings of 2017 ACM SIGSAC Conference on Computer and Communications Security. Dallas: ACM, 2017. 1465–1482.

7 Wang HY, Fan K, Li H, *et al*. A dynamic and verifiable

multi-keyword ranked search scheme in the P2P networking environment. Peer-to-Peer Networking and Applications, 2020, 13(6): 2342–2355. [doi: 10.1007/s12083-020-00912-7]

8 Xu GW, Li HW, Dai YS, *et al*. Enabling efficient and geometric range query with access control over encrypted spatial data. IEEE Transactions on Information Forensics and Security, 2019, 14(4): 870–885. [doi: 10.1109/TIFS.2018.2868162]

9 Huang K, Dong XL, Cao ZF, *et al*. Dynamic searchable symmetric encryption schemes with forward and backward security. IOP Conference Series: Materials Science and Engineering, 2020, 715(1): 012062.

10 Kermanshahi SK, Sun SF, Liu JK, *et al*. Geometric range search on encrypted data with Forward/Backward security. IEEE Transactions on Dependable and Secure Computing, 2020. [doi: 10.1109/TDSC.2020.2982389]

11 孙晓玲, 杨秋格, 沈焱萍, 等. 改进的高效动态可搜索加密方案. 计算机应用研究, 2020, 37(8): 2472–2476.

12 Wang Q, Guo Y, Huang HJ, *et al*. Multi-user forward secure dynamic searchable symmetric encryption. Proceedings of the 12th International Conference on Network and System Security. Cham: Springer, 2018. 125–140.

13 卢冰洁, 周俊, 曹珍富. 一种增强的多用户前向安全动态对称可搜索加密方案. 计算机研究与发展, 2020, 57(10): 2104–2116. [doi: 10.7544/issn1000-1239.2020.20200439]

14 Wong WK, Cheung DWL, Kao B, *et al*. Secure kNN computation on encrypted databases. Proceedings of 2009 ACM SIGMOD International Conference on Management of Data. Providence: ACM, 2009. 139–152.