

多关键词动态可搜索加密方案

袁承昊¹, 李勇¹, 任爽²

(1. 北京交通大学电子信息工程学院, 北京 100044; 2. 北京交通大学计算机与信息技术学院, 北京 100044)

摘要: 云存储用户将数据外包存储至云服务器以节省本地存储资源。然而, 云存储数据脱离了用户的物理控制范围, 可导致云端隐私数据被恶意窃取或泄露。目前动态可搜索加密方案多以对称可搜索加密为主, 需要预先建立安全的密钥共享信道, 难以直接应用于云存储数据共享场景。针对云存储多方数据的安全共享场景, 提出了一种多关键词动态可搜索加密方案。该方案通过引入布谷鸟过滤器构建正向索引, 实现数据拥有者对文档及索引的动态添加和删除。同时结合双线性对与拉格朗日插值多项式, 可支持多关键词的联合搜索。为降低密文检索阶段的计算开销, 采用倒排索引与正向索引结合的构造, 以提高云服务器的检索效率。基于判定线性 Diffie-Hellman 问题, 在适应性选择关键词攻击下可证明方案的安全性。通过实验分析所提方案在不同数据集中进行关键词搜索及索引更新的执行效率, 结果表明所提方案可有效避免检索时间与密文数量的线性相关性, 降低了数据量较大情况下更新操作中的计算开销。

关键词: 可搜索加密; 云存储; 布谷鸟过滤器; 倒排索引

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2023028

Dynamic multi-keyword searchable encryption scheme

YUAN Chenghao¹, LI Yong¹, REN Shuang²

1. School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China

2. School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China

Abstract: Users of cloud storage often outsource their data to cloud servers to save local storage resources. However, cloud storage data is out of the user's physical control, which may lead to malicious theft or disclosure of private data on cloud. At present, dynamic searchable encryption schemes are mostly based on symmetric searchable encryption, which need to establish a secure key sharing channel in advance, and cannot be directly applied to data sharing in cloud storage scenarios. For the secure sharing scenario of multi-party data in cloud storage, a dynamic multi-keyword searchable encryption scheme was proposed. The forward index was constructed by a cuckoo filter to enable data owners to perform documents and index dynamic updates. The conjunctive multi-keywords search

收稿日期: 2022-07-21; 修回日期: 2023-02-19

通信作者: 李勇, liyong@bjtu.edu.cn

基金项目: 国家自然科学基金 (62072025); 2020 年工业互联网创新发展工程项目 (TC200H01V)

Foundation Items: The National Natural Science Foundation of China (62072025), 2020 Innovative Development Project of the Industrial Internet (TC200H01V)

引用格式: 袁承昊, 李勇, 任爽. 多关键词动态可搜索加密方案[J]. 网络与信息安全学报, 2023, 9(2): 143-153.

Citation Format: YUAN C H, LI Y, REN S. Dynamic multi-keyword searchable encryption scheme[J]. Chinese Journal of Network and Information Security, 2023, 9(2): 143-153.

was supported by combining bilinear pairing with Lagrangian interpolation polynomials. To reduce the computational overhead in the ciphertext retrieval phase, a combination of inverted index and forward index was constructed to improve the retrieval efficiency of cloud server. This scheme is provably secure with the indistinguishability in adaptively chosen keyword attack under decision linear Diffie-Hellman problem. Simulation experiments were conducted to analyze the execution efficiency of the scheme for keyword search and index update in different datasets. The results show that the scheme effectively avoids the linear correlation between the retrieval time and the number of ciphertexts, and reduces the computational overhead in the update operation with a large amount of data.

Keywords: searchable encryption, cloud storage, cuckoo filter, inverted index

0 引言

云存储技术可为各类互联网应用提供灵活便捷的智能化服务,帮助用户低成本、高效率地按需外包存储个人数据^[1]。虽然云存储技术具有灵活性、可靠性、高可扩展性等优势,但仍然面临着较为突出的安全问题,其中数据安全性是云存储技术面临的最为严峻的问题之一^[2]。攻击者可在未经授权的情况下查看、窃取甚至恶意篡改云平台中存储的敏感数据,导致用户或企业的隐私信息泄露,造成无法挽回的损失。

为提高云存储数据的安全性及可用性,动态可搜索加密技术允许用户对云存储数据进行动态管理,同时保证加密数据的搜索功能,在云存储环境中具有重要的应用价值。

1 相关工作

Song 等^[3]提出了第一个对称可搜索加密方案。早期的可搜索加密方案主要围绕静态可搜索加密,默认对一个不可更改的数据库进行加密搜索。若存储数据需要更新时,静态可搜索加密方案需要对整个数据库重新构建加密索引,庞大的计算开销无法适应云存储用户对数据添加或删除的需求。为解决该问题,Kamara 等^[4]提出动态可搜索加密方案,该方案以对称可搜索加密为基础,在索引构建过程设置一个删除矩阵,删除矩阵中存储各文档包含的关键词列表。倒排索引中的“关键词-文档”对均存在一个指针指向删除矩阵中的对应条目。当用户对某文档提交删除操作时,删除矩阵中该文档的关键词列表将会被删除。然而,删除操作通常仅针对小范围的文档,而该方案需要对全部文档重新建立一个用于判断删除操作的

矩阵,这将带来较高的存储开销,但并未分析文档的动态更新带来的安全风险。

对于动态更新可搜索加密方案的安全性,Stefanov 等^[5]定义了对称可搜索加密前向安全性,并给出满足该安全性的具体构造。前向安全性是指云服务器无法获知添加的文档是否包含已被检索的关键词。但该方案要求用户使用若干个密钥计算文件的动态添加后的索引,因此其计算开销较大,且不支持文档的删除操作。Bost 等^[6]定义文档删除场景的后向安全性,即云服务器仅可获知关键词匹配的文档以及该关键词和文档的更新时间。在二者的研究基础上,Chamani 等^[7]提出一种同时满足前向及后向安全性的对称可搜索加密方案,并且该方案通过不经意映射将文档更新的复杂度降低至亚线性,有效提升了检索效率。Demertzis 等^[8]同样使用不经意映射,从减轻更新过程用户端存储开销的角度提出了具体的方案,但该方案需要进行多次信息交互,带来较高的计算开销。

近年来,加密云场景下支持动态更新的多关键词搜索问题,得到了广泛关注。Zhong 等^[9]提出了一种支持文件动态更新的多关键词模糊搜索方案,通过局部敏感哈希和布隆过滤器构建索引及查询向量,并利用索引树提升了方案的检索效率。Zhang 等^[10]结合索引树及连接多关键词的搜索模式,提出了一种支持动态添加及删除的可搜索加密方案 T-PECK。该方案将算法复杂度降低至亚线性,一定程度上提升了检索效率。但随着索引数量增加,在更新过程中,数据拥有者需要重新生成一个新的索引树,再由云服务器进行合并,因此对于更新阶段的效率仍有进一步研究及优化的空间。

2 方案概述

2.1 系统模型

多关键词动态可搜索加密系统模型包含以下3种实体：数据拥有者、数据使用者和云服务器，如图1所示。数据拥有者可以是个人、企业或者

组织，其希望将部分文档外包存储于云服务器，以节约本地的存储资源开销。数据使用者由数据拥有者指定，可通过关键词查询方式获取数据拥有者上传的文档。云服务器具有海量的存储资源、高效的计算能力，可完成加密文件的检索任务。

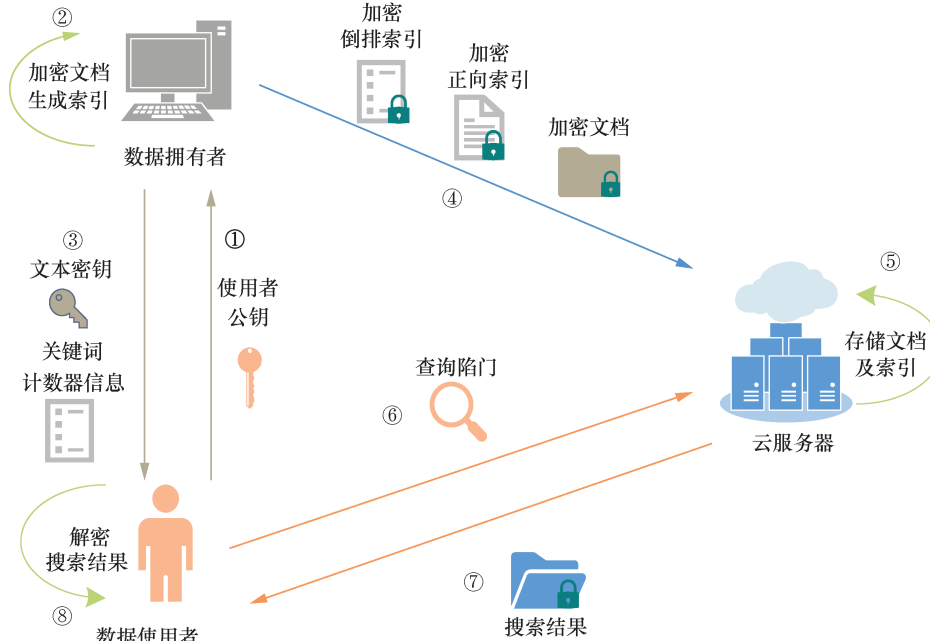


图1 多关键词动态可搜索加密系统模型
Figure 1 Dynamic multi-keyword searchable encryption system model

云存储数据的动态更新包括文档及索引的添加和删除两个过程，更新过程示意如图2所示。在添加过程中，数据拥有者发送给云服务器请求更新的消息，包括待添加的加密文档标签及其加密正向索引，以及倒排索引条目。云服务器将正向索引添加至布谷鸟过滤器^[11]中，更新加密的倒排索引和文档。同时，数据拥有者将更新的关键词计数器信息发送给数据使用者。

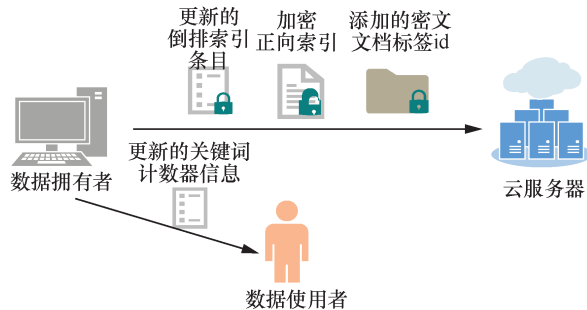


图2 云存储数据更新过程示意
Figure 2 Diagram of cloud storage data update process

对于删除过程，数据拥有者向云服务器发送请求删除的消息，提交待删除文档的标签、该文档对应的正向索引和更新的倒排索引条目。删除过程与添加过程类似，不同之处在于云服务器需要在布谷鸟过滤器和文档存储路径中删除正向索引和密文。

2.2 方案形式化定义

本文方案由以下多项式时间算法组成。

(1) 密钥生成算法

$\text{KeyGen}(1^\lambda) \rightarrow (\text{params}, \text{pk}, \text{sk})$ ：输入安全参数 1^λ ，输出系统参数 params 和数据使用者的公钥对 (sk, pk) 。

(2) 文档加密算法

$\text{EncFile}(\text{mk}, F) \rightarrow \text{CT}$ ：输入文本密钥 mk 和待加密文档集合 F ，输出密文 CT 。

(3) 索引建立算法

$\text{BuildIndex}(W, \text{params}, \text{pk}) \rightarrow (S_n, \text{Inv}, \text{cnt}^W)$ ：

输入文档包含的关键词集合 W 、系统参数 params ，输出对应的加密正向索引 S_n 、加密倒排索引 Inv 和 W 中各关键词的计数器 cnt^W 。关键词计数器 cnt^W 表示倒排索引中各关键词对应的文档数量。

(4) 陷门构造算法

$\text{Trapdoor}(\text{params}, \text{sk}, Q, \text{cnt}^Q) \rightarrow T$ ：输入关键词集合 Q 、私钥 sk 、 Q 中各关键词计数器 cnt^Q 和系统参数 params ，计算并输出查询陷门 T 。

(5) 匹配搜索算法

$\text{Search}(S_n, \text{Inv}, T) \rightarrow \text{CT}_m$ ：输入查询陷门 T 后，首先匹配倒排索引 Inv ，获得计数器值最小的关键词对应的文档列表；随后将陷门 T 与列表中各文档的正向索引 S_n 匹配，返回匹配成功的加密文档 CT_m 。

(6) 文档解密算法

$\text{DecFile}(\text{mk}, \text{CT}_m) \rightarrow F_m$ ：输入文本密钥 mk 对返回的结果 CT_m 进行解密，得到搜索结果的明文文档 F_m 。

(7) 文档添加算法

$\text{AddFile}(\text{op}, \text{CT}_n, \text{ID}_n, S_n, \text{Inv}_n) \rightarrow (\text{Inv}, \text{CF})$ ：数据拥有者输入待添加的密文文档 CT_n 、文档标签 ID_n 、正向索引 S_n 、更新的倒排索引条目 Inv_n 和操作 op ($\text{op}=\text{add}$)。云服务器存储密文 CT_n ，更新倒排索引 Inv ，根据文档标签 ID_n ，计算出布谷鸟过滤器中相应桶（一个桶中一般包含若干个条目，每个条目存储一个指纹，指纹即哈希值）的位置并添加 S_n 。

(8) 文档删除算法

$\text{DeleteFile}(\text{op}, \text{ID}_n, S_n, \text{Inv}_n) \rightarrow (\text{Inv}, \text{CF})$ ：数据拥有者输入待删除的密文文档标签 ID_n 、正向索引 S_n 、更新的倒排索引条目 Inv_n ，并备注操作 $\text{op}=\text{del}$ 。云服务器更新倒排索引 Inv ，根据文档标签 ID_n 计算其在布谷鸟过滤器中桶的位置，删除 S_n 并在文件存储路径中删除 ID_n 对应的文档 CT_n 。

2.3 安全模型

本文方案假设云服务器为“诚实但好奇”的半可信实体。在每次密文搜索执行过程中，云服务器尽可能获取并记录与加密文档和索引关键词相关的所有信息来进行计算来试图猜测明文信息。

因此，方案应保证云服务器无法获知各文档包含的关键词和数据使用者查询的关键词。

一旦数据使用者提交了一个查询关键词陷门，由于搜索匹配的统计信息和用户提交的关键词搜索请求会被泄露给云服务器，“诚实但好奇”的云服务器可以检验任何一个新上传的文档中是否包含该组关键词。而关键词空间信息熵通常较小，云服务器可以在搜索历史中不断选择关键词陷门对文档索引进行还原，该过程被称为适应性选择关键词攻击（IND-CKA）。定义本文方案 IND-CKA 安全模型为一个多项式时间内的敌手 $\mathcal{A}$ 和挑战者 $\mathcal{C}$ 参与的安全游戏，描述如下。

1) 初始化阶段 **Setup**：挑战者 $\mathcal{C}$ 输入一个安全参数 1^λ 并运行 $\text{KeyGen}(1^\lambda)$ 算法，将生成的系统参数集合 params 以及公钥 pk 发送给敌手 $\mathcal{A}$ 。挑战者 $\mathcal{C}$ 保留私钥 sk 。

2) 挑战前陷门查询 **QueryPhase1**：敌手 $\mathcal{A}$ 适应性地随机选择一系列关键词集 $\{Q_1, \dots, Q_h\}$ ，逐一发送给挑战者 $\mathcal{C}$ 。挑战者 $\mathcal{C}$ 执行 $\text{Trapdoor}(\text{params}, \text{sk}, Q, \text{cnt}^Q)$ 算法，生成各个关键词集所对应的陷门 T_h ，将其反馈给敌手 $\mathcal{A}$ 。

3) 挑战阶段 **Challenge**：敌手 $\mathcal{A}$ 选择一个关键词集合 Q^* 并将其发送给挑战者 $\mathcal{C}$ ，挑战者选择一个随机关键词集合 R^* 。对关键词集合 R^* 与 Q^* 唯一的限制是，不能在敌手 $\mathcal{A}$ 挑战前查询的关键词集合中出现。 $\mathcal{C}$ 设定 $W_0 = Q^*$ 和 $W_1 = R^*$ ，同时选择随机比特 $\beta \in \{0, 1\}$ ，运行 $\text{BuildIndex}(W_\beta, \text{params}, \text{pk})$ 算法生成关键词集 W_β 的对应索引 S_β 。之后挑战者 $\mathcal{C}$ 将三元组 (W_0, W_1, S_β) 发送给敌手 $\mathcal{A}$ 。

4) 挑战陷门后查询 **QueryPhase2**：敌手 $\mathcal{A}$ 再另外自适应地选择一系列关键词集 $Q_{h+1}, \dots, Q_\lambda$ ，这些关键词集不能包括挑战阶段返回的 W_0 和 W_1 。 $\mathcal{A}$ 将这些关键词集依次发送给挑战者 $\mathcal{C}$ ， $\mathcal{C}$ 运行 $\text{Trapdoor}(\text{params}, \text{sk}, Q, \text{cnt}^Q)$ 算法生成各关键词集相应的陷门 T_λ 并将其反馈给 $\mathcal{A}$ 。敌手 $\mathcal{A}$ 在概率多项式时间内的查询次数为 t 。

5) 猜测阶段 **Guess**：敌手 $\mathcal{A}$ 需要输出 $\beta' = 0$ 或 $\beta' = 1$ 作为对 $\mathcal{C}$ 选择随机值的判断。如果 $\beta = \beta'$ ，则 $\mathcal{A}$ 获得游戏胜利，反之 $\mathcal{A}$ 游戏失败。

对于任意一个多项式时间敌手 $\mathcal{A}$ ，其在此安

全游戏中获得胜利的优势为

$$\text{Adv}_{\mathcal{A}}(1^\lambda) = |\Pr(\beta = \beta') - 1/2|$$

在安全参数 1^λ 的条件下, 若敌手 $\mathcal{A}$ 的优势为一个可忽略函数, 则称该方案满足 IND-CKA 安全。

3 方案具体构造

本文主要符号及其定义如表 1 所示。

表 1 符号及定义
Table 1 Symbol and definitions

符号	含义
CT	上传的加密文档集合
label = {ID ₁ , ..., ID _n }	各文档标签
W = {w ₁ , ..., w _k }	某文档包含的关键词集合
CF	布谷鸟过滤器
G	有阶循环群
g	循环群 G 的生成元
Z _p [*]	整数加法群
H ₁ , H ₂ , H ₃	3 个不同的抗碰撞哈希函数
H _F , H _C	构造布谷鸟过滤器哈希函数
H _λ , P _{key}	构造倒排索引的伪随机函数
(sk, pk)	数据使用者的公私钥对
S _n	加密正向索引
T	查询陷门
Inv	加密倒排索引
L _W	构造正向索引的映射函数

多关键词动态可搜索加密方案的构造过程如下。

(1) 密钥生成算法 KeyGen(1^λ) → (params, pk, sk)

首先输入安全参数 1^λ , 输出系统参数 params = {G, G_T, e, g, H₁, H₂, H₃, H_λ, P_{key}}。其中, G 和 G_T 为两个不同的有阶循环群, e 为双线性对映射, g 为群 G 的生成元, H₁, H₂, H₃ 为 3 个不同的抗碰撞哈希函数。H_λ 为一个由密钥控制的伪随机函数, P_{key} 为一个由密钥 key 控制的伪随机置换。

$$H_1 : \{0,1\}^* \rightarrow G, H_2 : \{0,1\}^* \rightarrow G, H_3 : \{0,1\}^* \rightarrow Z_p^*$$

$$H_\lambda : \{0,1\}^\lambda \times \{0,1\}^p \rightarrow \{0,1\}^u$$

$$P_{\text{key}} : \{0,1\}^u \times \{0,1\}^v \rightarrow \{0,1\}^v$$

数据使用者选择一个随机数 $x \in Z_p^*$ 作为其私钥 sk, 并计算 $y = g^x$ 作为其公钥 pk。

(2) 文档加密算法 EncFile(mk, F) → CT

数据拥有者首先生成一个文本密钥 mk, 并使用 mk 对外包存储文档集合 F 进行高级加密标准 (AES) 加密生成密文集合 CT。此后, 数据拥有者利用数据使用者提供的公钥 pk 加密 mk, 将其发送给数据使用者。

(3) 索引建立算法 BuildIndex(W, params, pk) → (S_n, Inv, cnt^W)

数据拥有者首先赋予待上传文档一个独有的标签, 并提取各文档所包含的关键词集合。定义关键词集合为 $W = \{w_1, \dots, w_k\}$, 数据拥有者使用系统参数 params, 计算得到关键词 $w_i (i = 1, 2, \dots, k)$ 的 3 个不同的哈希值。

$$h_i = H_1(w_i), f_i = H_2(w_i), p_i = H_3(w_i)$$

数据拥有者统计各关键词对应的文档标签, 生成以“关键词 w_i : 文档列表 ID_{list}; 文档数 cnt”为格式的倒排索引并存储。随后, 数据拥有者统计各关键词对应的文档数量 cnt, 计算得到伪随机函数 P_{key} 的密钥 key^{w_i}, 构建加密倒排索引 Inv。

$$\text{key}^{w_i} = H_\lambda(f_i, \text{cnt}^{w_i})$$

$$\text{Inv} : \{h_i : P_{\text{key}^{w_i}}(\text{ID}_{\text{list}})\}, \text{ID}_{\text{list}} = \text{ID}_1, \dots, \text{ID}_n$$

此后, 数据拥有者根据 $p_i (i = 1, 2, \dots, k)$ 构建映射函数 L_W, 选择两个随机数 $r, s \in Z_p^*$, 计算得到 A, B, c_i。

$$A = g^r, B = y^s, c_i = h_i^r f_i^s$$

使用映射函数 L_W 构建 (C₁, C₂, ..., C_k)。

$$(C_1, C_2, \dots, C_k) = L_W(c_i), i = 1, \dots, k$$

该集合中任意元素 C_i 都由集合 (c₁, c₂, ..., c_k) 中的所有元素参与计算得到。最后, 得到该文档对应的加密正向索引 S_n = (A, B, C_i)。数据拥有者将正向索引 S_n 与倒排索引 Inv 以及密文集合 CT 上传至云服务器, 将各关键词的计数器列表 cnt^W 使用公钥 pk 加密后发送给数据使用者。

(4) 陷门构造算法 Trapdoor(params, sk, Q, cnt^Q) → T

数据使用者想要获取包含关键字集合

$Q = \{w_u, u=1, \dots, m\}$ 的文档, 其首先使用系统参数 params 完成计算查询关键词的哈希值。

$$\begin{aligned} h_u &= H_1(w_u), f_u = H_2(w_u), \\ q_u &= H_3(w_u), u=1, \dots, m \end{aligned}$$

随后读取各关键词的计数器 cnt^Q , 选择计数器数值最小的关键词 (假设为 w_1), 计算 w_1 对应的倒排索引密钥。

$$\text{key}^{w_1} = H_\lambda(f_1, \text{cnt}^{w_1})$$

此后, 数据使用者选择一个随机数 $t \in Z_p^*$, 使用私钥 $\text{sk} = x$, 对除 w_1 以外的其余关键词 $\{w_u, u=2, \dots, m\}$ 计算得到 D, E, F 。

$$D = g^t, E = (h_2 \cdots h_m)^t, F = (f_2 \cdots f_m)^{t/x}$$

最终输出查询陷门 T 并发送至云服务器。

$$T = \{D, E, F, q_u, (h_1, \text{key}^{w_1}), u=2, \dots, m\}$$

(5) 匹配搜索算法 $\text{Search}(S_n, \text{Inv}, T) \rightarrow (CT_m)$

本文方案的搜索模式为多关键词联合搜索, 即搜索结果中的文档应包括数据使用者输入的全部关键词。云服务器得到查询陷门 T 后, 首先在加密倒排索引 Inv 中查找 h_1 , 使用 key^{w_1} 还原出 w_1 的文档列表 ID_{list} 。

假设 ID_{list} 中的某一文档标签为 ID_n , 则云服务器根据 ID_n 查找该文档的正向索引在布谷鸟过滤器中存放的位置。由于布谷鸟过滤器的特性, 存在两个不同的哈希函数对文档标签 ID_n 进行运算, 而最终只有一个桶存放 ID_n 的正向索引。云服务器首先查找这两个桶的位置。若两个桶中只有一个桶不为空, 则此桶存放的条目即正向索引 S_n ; 若两桶均不为空, 则随机选择一个桶, 从中提取出正向索引 $S_n = (A, B, C_i)$, 同时提取陷门 T 中的 q_u , 计算 qc_u 。

$$qc_u = \prod_{i=1}^k C_i^{q_u^{i-1}}, u=1, \dots, m$$

提取查询陷门 T 中的 (D, E, F) , 进行双线性对运算并检验如下等式是否成立。

$$e(A, E)e(B, F) = e(D, \prod_{u=1}^m qc_u)$$

若上式成立, 则将 ID_n 对应的加密文档添加到检索结果集合 CT_m 中; 若不成立, 在两桶均不

为空的情况下, 选择 ID_n 对应的另一个桶并对其进行再次检验。直至 ID_{list} 中所有的文档标签计算完成后, 云服务器输出检索结果集合 CT_m 并发送给数据使用者。

(6) 文档解密算法 $\text{DecFile}(\text{mk}, \text{CT}_m) \rightarrow F_m$

数据使用者通过私钥 sk 解密获得文本密钥 mk , 使用 mk 解密检索结果集合 CT_m 中的密文, 恢复出所需文档的明文 F_m 。

本文方案支持数据拥有者动态添加和删除外包存储文档及其索引。下面将介绍该方案的文档添加算法和文档删除算法。

(1) 文档添加算法 $\text{AddFile}(\text{op}, \text{CT}_n, \text{ID}_n, S_n, \text{Inv}_n) \rightarrow (\text{Inv}, \text{CF})$

数据拥有者发送带有 $\text{op}=\text{add}$ 的消息要求云服务器对特定文档进行添加, 该消息包含待添加的密文 CT_n 、 CT_n 的文档标签 ID_n 、更新的加密倒排索引条目 Inv_n 及加密正向索引 S_n 。加密正向索引 S_n 的构造方法在前文索引建立算法中已详细介绍, 而对于更新的加密倒排索引条目 Inv_n 的构建, 数据拥有者执行如下计算。

$$\begin{aligned} \text{cnt}^{w_n} &= \text{cnt}^{w_n} + 1 \\ \text{key}^{w_n} &= H_\lambda(f_n, \text{cnt}^{w_n}) \\ \text{Inv}_n &: \{h_n : P_{\text{key}^{w_n}}(\text{ID}_{\text{list}}^*)\} \end{aligned}$$

云服务器接收到该文档添加消息后, 首先添加密文 CT_n , 并在完整的加密倒排索引 Inv 中将 Inv_n 替换为更新后的条目。同时, 使用两个哈希函数 H_F, H_C 确定该文档的正向索引 S_n 在布谷鸟过滤器中对应桶的位置, 其中 H_F 用于生成文档标签 ID_n 对应的指纹 fp_n , H_C 用于计算 ID_n 对应的两个桶的位置, 具体计算如下。

$$\text{fp}_n = H_F(\text{ID}_n)$$

$$i_1 = H_C(\text{ID}_n), i_2 = i_1 \oplus H_C(\text{fp}_n)$$

i_1 和 i_2 为标签 ID_n 对应的两个桶, 而正向索引 S_n 最终只可能存放在桶 i_1 或桶 i_2 中。云服务器分别读取桶 i_1 和 i_2 中的内容确定最终存放位置。若两个桶均为空, 则随机选择一个空桶存放指纹 fp_n 及索引 S_n ; 若两个桶中只有一个桶不为空, 则在空桶处存放 fp_n 和 S_n ; 若两者均不为空, 则随机选择其中一个空桶, 替换原本存放在此的标签指纹

和索引。而对于原本存放在该桶的内容，云服务器提取其中的指纹重新执行上述计算，得出其对应的另一个桶的位置并存放。最终得到更新后的布谷鸟过滤器 CF。

(2) 文档删除算法 DeleteFile(op, ID_n, S_n, Inv_n) → (Inv, CF)

当数据拥有者想对部分存储文档进行删除时，需向云服务器提交 op=del 的文档删除请求，同时提交待删除文档的标签 ID_n、更新的倒排索引条目 Inv_n 和正向索引 S_n。

$$\begin{aligned} \text{cnt}^{w_n} &= \text{cnt}^{w_n} - 1 \\ \text{key}^{w_n} &= H_\lambda(f_n, \text{cnt}^{w_n}) \\ \text{Inv}_n &: \{h_n : P_{\text{key}^{w_n}}(\text{ID}_{\text{list}}^*)\} \end{aligned}$$

云服务器在收到文档删除请求后，首先在存储的倒排索引 Inv 中将待更新的条目替换为 Inv_n，计算得到 ID_n 对应的两个桶，选择其中一个桶，比较该桶中存储的索引与 S_n 是否一致。若一致，则删除该桶所有条目；若不一致，则删除 ID_n 对应的另一个桶中的所有条目。

4 方案分析

4.1 安全分析

本文方案的安全性基于判定线性 Diffie-Hellman 问题 (DLDH)，在随机预言机模型下，该方案是 IND-CKA 安全的。

定理 1 设 (t, ε) -DLDH 假设在群 G 中成立。对于任意的 q_t, q_h, k 以及 $t' < t$ ，本文方案在随机预言机模型下是 $(t', q_t, q_h, 4kq_t\varepsilon)$ IND-CKA 安全的。其中 q_t 为敌手 $\mathcal{A}$ 向敌手 $\mathcal{B}$ 进行陷门查询的最大次数， q_h 为敌手 $\mathcal{A}$ 向随机预言机进行陷门查询的最大次数， k 为一次查询请求中最多的关键词数量， ε 为敌手 $\mathcal{A}$ 求解 DLDH 问题的优势， t' 为算法的执行时间。

证明 设 $\mathcal{A}$ 为一个概率多项式时间内的敌手，对本文方案进行 IND-CKA 攻击。系统参数 params 中的哈希函数 H_1, H_2, H_3 均由随机预言机建模。另设定一个新的多项式时间内敌手 $\mathcal{B}$ 利用敌手 $\mathcal{A}$ 求解 DLDH 问题，过程如下。

(1) 初始化阶段 Setup

挑战者 $\mathcal{C}$ 首先给出群 G 上 DLDH 问题的相

关参数 $(g_1, g_2, g_3, v_1, v_2, v_3)$ ，并发送给敌手 $\mathcal{B}$ 。

$$\begin{aligned} v_1 &= g_1^a, v_2 = g_2^b, \\ v_3 &= g_3^{a+b} \text{ 或 } v_3 = z, a, b, z \in Z_p^* \end{aligned}$$

敌手 $\mathcal{B}$ 设定 $g = g_1, y = g_2$ ，并设定私钥 $\text{sk} = x$ 满足 $g_1^x = g_2$ 。此外， $\mathcal{B}$ 还将秘密地选择一个随机数 $\mu \in Z_p^*$ ，该随机数将用于随机预言机模型和挑战阶段。

同时， $\mathcal{B}$ 将选取一个安全参数 1^λ 并执行 KeyGen(1^λ) 算法，生成系统参数 $\text{params} = \{g, G, H_1, H_2, H_3\}$ ，将 params 和公钥 $\text{pk} = y$ 发送给 $\mathcal{A}$ 。敌手 $\mathcal{A}$ 向随机预言机最多提交 q_h 次关键词查询，随机预言机响应这些查询，向 $\mathcal{A}$ 返回关键词对应的陷门，敌手 $\mathcal{B}$ 存储一个哈希列表 H_{list} 。

$$H_{\text{list}} : \{w_i, \text{coin}_i; h_i, d_i, f_i, e_i; p_i\}$$

如果某关键词 w_i 已被查询过，则 $\mathcal{B}$ 将该关键词的哈希值返回给 $\mathcal{A}$ 。

$$h_i = H_1(w_i), f_i = H_2(w_i), p_i = H_3(w_i)$$

若关键词 w_i 未被查询，则 $\mathcal{B}$ 为 p_i 选择一个随机值 $p_i \in Z_p^*$ ，并抛掷一个随机硬币 $\text{coin}_i \in \{0, 1\}$ 。若该硬币置 1，其概率为 $1/kq_t$ ；硬币置 0，其概率为 $1 - 1/kq_t$ 。若硬币置 0，则 $\mathcal{B}$ 选择两个随机数 $d_i, e_i \in Z_p^*$ ，令 $h_i = g_1^{d_i}, f_i = g_2^{e_i}$ ；若硬币置 1，则 $\mathcal{B}$ 选择一个随机数 $d_i \in Z_p^*$ ，执行如下计算。

$$e_i = d_i / \mu$$

随后，生成 $h_i = g_3^{d_i}, f_i = g_3^{e_i}$ 。最后， $\mathcal{B}$ 将得到的 h_i, f_i, p_i 返回给敌手 $\mathcal{A}$ ，并将 $\{w_i, \text{coin}_i; h_i, d_i, f_i, e_i; p_i\}$ 添加到列表 H_{list} 中。

(2) 挑战前陷门查询 QueryPhase1

敌手 $\mathcal{A}$ 自适应地查询一系列关键词集合的陷门。设某次查询的关键词集合为 Q ，返回的陷门为 T 。 $\mathcal{B}$ 获取查询关键词在列表 H_{list} 中的对应元组。若至少有一个 $\text{coin}_u = 1$ ，则 $\mathcal{B}$ 选择弃权；若 coin_u 全部为 0，则 $\mathcal{B}$ 生成一个随机数 $t \in Z_p^*$ ， t 会在下次成功的陷门查询中更新。随后， $\mathcal{B}$ 输出陷门 T 。

$$T = \{T_a, T_b, T_c, q_1, \dots, q_u\}$$

其中， T_a, T_b, T_c 如下。

$$T_a = g_1^t, T_b = g_1^{t(\sum_{u=1}^m d_u)}, T_c = g_1^{t(\sum_{u=1}^m e_u)}$$

其中, $t \in Z_p^*$ 。最后, $\mathcal{B}$ 将陷门 T 反馈给 $\mathcal{A}$ 。

(3) 挑战阶段 Challenge

敌手 $\mathcal{A}$ 选择一个关键词集 Q^* 并将其发送给敌手 $\mathcal{B}$, 敌手 $\mathcal{B}$ 另外选择一个关键词集 R^* 。关键词集 R^* 与关键词集 Q^* 不能在之前敌手 $\mathcal{A}$ 的查询中出现。 $\mathcal{B}$ 设置 $W_0 = Q^*, W_1 = R^*$, 并选择一位随机比特 $\beta \in \{0, 1\}$ 得到关键词集合 W_β 。

$$W_\beta = \{w_{\beta,1}, w_{\beta,2}, \dots, w_{\beta,m}\}$$

之后, $\mathcal{B}$ 将 W_β 中的所有关键词逐一向随机预言机询问并返回 H_{list} 的对应元组, 若不存在 $\text{coin}_{\beta,u} = 1$, 则 $\mathcal{B}$ 弃权。反之, $\mathcal{B}$ 生成挑战密文 S_β 。

$$S_\beta = \{A, B, C_{\beta,1}, \dots, C_{\beta,u}\}$$

S_β 中各元素定义如下所示。

$$A = v_1, B = v_2^\eta, C_{\beta,u} = L_{W_\beta}(c_{\beta,u})$$

若 $\text{coin}_{\beta,u} = 0$, $c_{\beta,u}$ 执行如下计算。

$$c_{\beta,u} = v_1^{d_{\beta,u}} v_2^{e_{\beta,u}\eta} = g_1^{ad_{\beta,u}} g_2^{d_{\beta,u}\eta} = h_{\beta,u}^a f_{\beta,u}^{\eta b}$$

若 $\text{coin}_{\beta,u} = 1$, $c_{\beta,u}$ 执行如下计算。

$$c_{\beta,u} = v_3^{d_{\beta,u}} = g_3^{(a+b)d_{\beta,u}} = g_3^{ad_{\beta,u}} (g_3^{d_{\beta,u}/\eta})^{b\eta} = (h_{\beta,u})^a (f_{\beta,u})^{b\eta}$$

最后 $\mathcal{B}$ 将三元组 (W_0, W_1, S_β) 发送给敌手 $\mathcal{A}$ 。

(4) 挑战后陷门查询 QueryPhase2

敌手 $\mathcal{A}$ 继续进行一系列关键词集合的陷门查询, 且查询的关键词集合不能为 W_0 和 W_1 。 $\mathcal{B}$ 执行与 QueryPhase1 相同的动作, 输出陷门 T 并反馈给敌手 $\mathcal{A}$ 。

(5) 猜测阶段 Guess

最终, $\mathcal{A}$ 输出猜测结果 $\beta' = 0$ 或 $\beta' = 1$ 。若 $\beta = \beta'$, 则敌手 $\mathcal{B}$ 输出对 DLDH 问题的判断结果 $v_3 = g_3^{a+b}$; 反之, $\mathcal{B}$ 输出结果 $v_3 = z$ 。

敌手 $\mathcal{B}$ 想要获得成功, 前提是需要完整地进行该安全游戏的全过程, 即在挑战前陷门查询阶段和挑战阶段不能弃权。基于敌手 $\mathcal{A}$ 最多可以进行 q_t 次陷门查询, k 为查询请求中最大的关键词个数, $\mathcal{B}$ 在挑战前陷门查询阶段和挑战阶段不弃权的概率分别为

$$\Pr(\mathcal{B} \text{ pass Queryphase 1 and 2}) = (1 - 1/kq_t)^{kq_t}$$

$$\Pr(\mathcal{B} \text{ pass Challenge}) = [1 - (1 - 1/kq_t)^k]$$

由上文可知, $kq_t \in [2, +\infty)$, $(1 - 1/kq_t)^{kq_t}$ 在该区间内为单调递增函数, 其最小值为 $(1 - 1/kq_t)^{kq_t} \geq 1/4$ 。而 $[1 - (1 - 1/kq_t)^k] \geq [1 - (1 - 1/kq_t)] = 1/kq_t$, 因此 $\mathcal{B}$ 完成游戏的概率为

$$\Pr(\mathcal{B} \text{ pass the game}) =$$

$$(1 - \frac{1}{kq_t})^{kq_t} \cdot [1 - (1 - \frac{1}{kq_t})^k] \geq \frac{1}{4kq_t}$$

由定理中的条件可知, 敌手 $\mathcal{A}$ 猜测成功的优势为 ε , ε 不可忽略, 由此可推出敌手 $\mathcal{B}$ 获得胜利的优势为 $\varepsilon/4kq_t$ 。

$$\text{Adv}_{\mathcal{B}}^{\text{DLDH}}(1^\lambda) \geq 1/4kq_t |\Pr(\beta = \beta') - 1/2| = \varepsilon/4kq_t$$

因此, 敌手 $\mathcal{B}$ 求解 DLDH 问题优势不可忽略, 本文方案在随机预言机模型下是 $(t', q_t, q_h, 4kq_t\varepsilon)$ IND-CKA 安全的。

4.2 功能分析

接下来从是否支持多关键词搜索、文档及索引的动态添加和删除, 以及 IND-CKA 安全方面, 将本文方案与文献[12-15]提出的方案进行对比, 如表 2 所示。

表 2 不同可搜索加密方案的功能性比较
Table 2 Functional comparison of different searchable encryption schemes

方案	多关键词搜索	动态添加	动态删除	IND-CKA
文献[12]				√
文献[13]	√			√
文献[14]		√		√
文献[15]	√	√	√	√
本文	√	√	√	√

由表 2 可知, 相较于其他方案, 本文方案在支持动态更新功能上具有一定优势, 并支持多关键词搜索模式, 适用于多用户场景下数据的安全共享。

4.3 性能分析

对不同可搜索加密方案使用 Python 编程语言进行仿真实验, 使用开源的 Pycryptodome 和 pypbc 模块构建双线性对映射和群上的乘法运算

及幂运算; 哈希函数使用 hashlib 模块中的 SHA3_256 算法; 使用 numpy 模块构建基于拉格朗日插值多项式的映射函数^[16]; 使用 mmh3 非加密哈希算法模块构建布谷鸟过滤器中的哈希运算。实验在 Windows 10 操作系统下进行, 处理器为 Intel Core i7-10510U CPU @ 1.80 GHz。分析单次密码学基础运算的执行时间, 设定群 G 中的元素位数为 160 bit, 对各运算重复执行 1 000 次并取平均值, 执行耗时如表 3 所示。

由表 3 可知, 执行单次双线性对耗时为 9.9 ms, 远高于其他基础运算。进行一次群 G 中的幂运算及哈希运算的时间分别为 0.028 ms 和 0.017 ms, 比群中的乘法运算和加法运算要占用更多的计算开销。将本文方案与文献[12-14]中提出的方案从构建索引耗时、生成陷门耗时、密文搜索耗时 3 个

方面进行性能上的理论比较。假设某文档集合共有 m 个文档, 每个文档包含 n 个关键词, 数据使用者想要对 k 个关键词进行查询, 关键词计数器最小值为 l 。各方案在构建索引阶段、生成陷门阶段及密文搜索阶段执行密码学基础运算的耗时比较如表 4 所示。

表 3 单次密码学基础运算执行耗时
Table 3 Single cryptographic base operation execution time

符号	运算类型	执行耗时 ms
t_p	双线性对运算	9.9
t_e	群 G 中的幂运算	0.028
t_m	群 G 中的乘法运算	0.003
t_a	群 G 中的加法运算	0.001
t_h	哈希运算	0.017

表 4 可搜索加密方案各阶段耗时比较

Table 4 Comparison of time consuming in each phase of the searchable encryption schemes

方案	构建索引	生成陷门	密文搜索
文献[12]	$mn(2t_h + 3t_m + 3t_a) = 0.046mn$	$k(t_p + 2t_m + 2t_a + 3t_h) = 9.959k$	$m(2t_p + t_h + 3t_m + 2t_a) = 19.828m$
文献[13]	$2mt_e + mn(2t_h + 2t_a) = 0.056m + 0.036mn$	$k(t_a + t_h) + 3t_e = 0.018k + 0.084$	$3mt_p = 29.7m$
文献[14]	$2mt_e + mn(t_h + t_m) = 0.056m + 0.02mn$	$k(t_m + t_h + t_e) = 0.048k$	$3mt_p + t_e = 29.7m + 0.028$
本文	$2mt_e + mn(3t_h + t_m) = 0.056m + 0.054mn$	$t_e + k(3t_h + 2t_m) = 0.051k + 0.028$	$(l + 2)t_p + nlt_e = 9.9l + 0.028nl + 18.8$

在构建索引阶段, 文献[14]提出的方案构建索引的耗时最短, 其次为文献[13]和文献[12]提出的方案。由于本文方案同时构建了正向索引和倒排索引, 因此在构建索引阶段的计算耗时略高于其他方案。而在生成陷门阶段, 所有方案的计算耗时均与查询关键词总数线性相关。若为单关键词搜索, 文献[14]提出的方案生成陷门的耗时最短, 其次为本文方案和文献[13]提出的方案, 而文献[12]提出的方案由于在构造陷门时使用了一次双线性对运算, 因此其耗时最高。而对于多关键词搜索模式, 文献[13]提出的方案生成陷门的耗时最短, 本文方案的陷门生成耗时略高于文献[14]提出的方案, 但远小于文献[12]提出的方案生成陷门的耗时。对于搜索阶段, 其他方案的搜索耗时均与文档数 m 线性相关, 而本文方案只与各文档关键词数 n 和关键词计数器 l 有关。因此, 文献[14]提出的方案在搜索阶段的计算开销最高。通常云服务器存储文档的数量较多, 即 $mn(2t_h +$

$3t_m + 3t_a) = 0.046m$, 因此本文方案在密文搜索阶段的计算效率比其他方案更具优势。

为更加直观地分析本文方案使用倒排索引与正向索引构建在可搜索加密方案各阶段带来的性能变化, 设计一个构建方式为只使用正向索引的方案, 其索引构建细节与本文方案构建正向索引的过程一致, 但该方案不使用倒排索引。对两个方案从构建索引、生成陷门、密文检索 3 个阶段分别进行仿真实验。比较在上述各阶段中引入倒排索引前后带来的计算开销差异。仿真实验使用的数据集为互联网工程任务组 (IETF) 发布的 RFC (request for comments) 官方文档, 该文档集主要记录互联网发展阶段的相关信息。从总共 9 196 份文档中分别随机抽取出 50、100、150、200、250 份文档作为 5 个不同大小的数据集。

依次读取各数据集的索引关键词信息, 生成加密的倒排索引及正向索引。两种方式的索引构建耗时如图 3 所示。两种方式的索引构建耗时均

呈线性增长的趋势。当文档数目为 50 个时,使用倒排索引和未使用倒排索引的索引构建耗时较为接近,分别为 38.1 ms 和 27.3 ms。随着文档数量的增加,两种方式索引构建耗时的差距不断增加。当文档数量达到 250 个,两种方式的索引构建耗时分别是 198.1 ms 和 148 ms。整体来看,倒排索引与正向索引结合的方式比只构建正向索引在计算开销上高出约 37%。

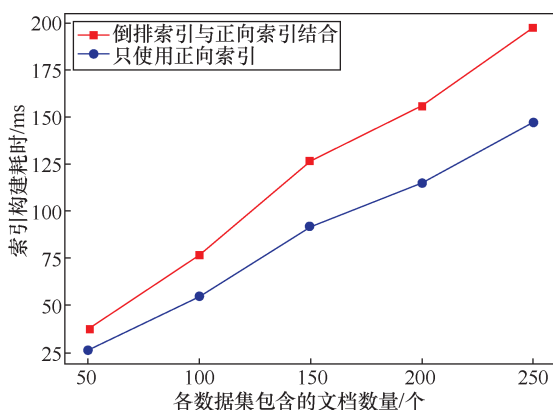


图3 两种方式的索引构建耗时

Figure 3 Index construction time of two ways

在生成陷门阶段,使用文档数目为 100 的数据集,并设定数据使用者分别使用不同数量的关键词进行搜索查询,每组实验从 2 个关键词逐个增加至 7 个关键词。比较倒排索引与正向索引结合与只构建正向索引在生成陷门阶段的计算开销,具体结果如图 4 所示。当数据使用者只查询 2 个不同的关键词时,两种方式在生成陷门的耗时并无太大差异。当查询的关键词数量为 7 个时,倒排索引与正向索引结合的陷门生成耗时为 0.74 ms,比只采用正向索引的生成陷门耗时多出约 0.09 ms。整体来看,倒排索引与正向索引结合的生成陷门耗时比只使用正向索引多出约 8.2%。但所有情况下生成陷门耗时均未超过 1 ms,因此本文方案生成陷门时无须付出较高的耗时。

在密文搜索阶段的仿真实验中,设定数据使用者对 3 个关键词进行联合搜索,即返回的全部文档都应包含这 3 个关键词。如图 5 所示,随着文档数量的增加,只使用正向索引构建的方案密文搜索耗时呈线性增长趋势,并且增长幅度较大。而倒排索引与正向索引结合的方案搜索耗时基本保持不变。当文档数量达到 250 个时,两者

的密文搜索耗时具有明显的差异。只使用倒排索引的检索耗时为 2.05 s,而倒排索引与正向索引结合的搜索耗时只有 0.3 s。本文方案将倒排索引与正向索引相结合,有效提升了密文搜索效率。

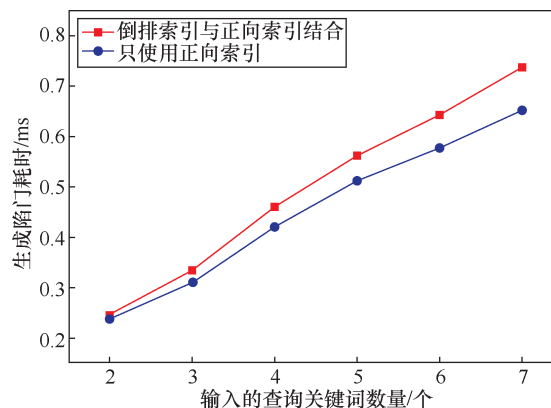


图4 两种方式生成陷门耗时

Figure 4 Generation time of trapdoor of two ways

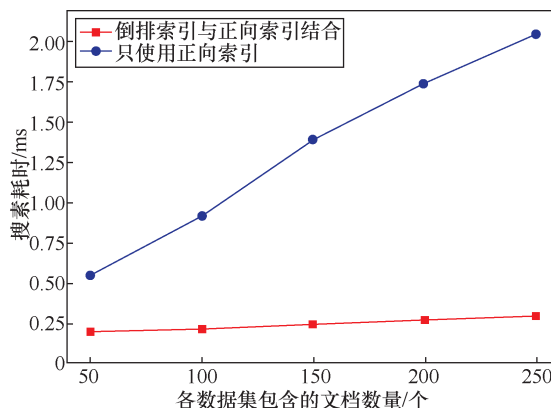


图5 两种方式在不同数据集上的搜索耗时

Figure 5 Search time of two ways in different data sets

在本文方案中,通过倒排索引与正向索引结合的构建方式,有效减少了云服务器进行的双线性对运算次数,提升了多关键词搜索效率。数据拥有者和数据使用者虽然会额外付出一定的计算开销,但缩短了密文搜索过程的时间,符合 3 种实体在云存储多方数据共享场景中的核心需求。

5 结束语

本文围绕云存储中多方数据的安全共享场景,提出了一种多关键词动态可搜索加密方案。本文方案采用混合加密思路进行设计,基于对称密码构建倒排索引,使用公钥密码构建正向索引,并在布谷鸟过滤器中进行存储,支持对索引的添

加及删除。同时结合双线性对与拉格朗日插值多项式,让所提方案支持多关键词的联合搜索模式。在 DLDH 假设下,本文方案满足 IND-CKA 安全。通过性能分析,方案倒排索引与正向索引相结合的构建方式能有效减少双线性对运算次数,提高密文检索效率。

参考文献:

- [1] CHAUDHARI S, KUMAR PATHURI S. A comprehensive survey on public auditing for secure cloud storage[J]. International Journal of Engineering & Technology, 2018, 7: 565-569.
- [2] SUN P J. Privacy protection and data security in cloud computing: a survey, challenges, and solutions[J]. IEEE Access, 2019, 7: 147420-147452.
- [3] SONG D X, WAGNER D, PERRIG A. Practical techniques for searches on encrypted data[C]//Proceedings of Proceeding 2000 IEEE Symposium on Security and Privacy. 2002: 44-55.
- [4] KAMARA S, PAPAMANTHOU C, ROEDER T. Dynamic searchable symmetric encryption[C]//Proceedings of the 2012 ACM Conference on Computer and Communications Security. 2012: 965-976.
- [5] STEFANOV E, PAPAMANTHOU C, SHI E. Practical dynamic searchable encryption with small leakage[EB]. Cryptology ePrint Archive, 2013.
- [6] BOST R, MINAUD B, OHRIMENKO O. Forward and backward private searchable encryption from constrained cryptographic primitives[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 1465-1482.
- [7] CHAMANI J G, PAPADOPOULOS D, PAPAMANTHOU C, et al. New constructions for forward and backward private symmetric searchable encryption[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 1038-1055.
- [8] DEMERTZIS I, CHAMANI J G, PAPADOPOULOS D, et al. Dynamic searchable encryption with small client storage[EB]. Cryptology ePrint Archive, 2019.
- [9] ZHONG H, LI Z F, CUI J, et al. Efficient dynamic multi-keyword fuzzy search over encrypted cloud data[J]. Journal of Network and Computer Applications, 2020, 149: 102469.
- [10] ZHANG Y, YOU L, LI Y. Tree-based public key encryption with conjunctive keyword search[J]. Security and Communication Networks, 2021, 2021: 1-16.
- [11] FAN B, ANDERSEN D G, KAMINSKY M, et al. Cuckoo filter: practically better than bloom[C]//Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies. 2014: 75-88.
- [12] HE D B, MA M M, ZEADALLY S, et al. Certificateless public key authenticated encryption with keyword search for industrial Internet of Things[J]. IEEE Transactions on Industrial Informatics, 2018, 14(8): 3618-3627.
- [13] ZHANG R, XUE R. Efficient keyword search for public-key setting[C]//Proceedings of MILCOM 2015-2015 IEEE Military Communications Conference. 2015: 1236-1241.
- [14] YE F H, DONG X L, SHEN J C, et al. A verifiable dynamic multi-user searchable encryption scheme without trusted third parties[C]//Proceedings of 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS). 2020: 896-900.
- [15] GUO C, ZHUANG R H, CHANG C C, et al. Dynamic multi-keyword ranked search based on bloom filter over encrypted cloud data[J]. IEEE Access, 2019, 7: 35826-35837.
- [16] SONG C G, LIU X, YAN Y L. Efficient Public key encryption with field-free conjunctive keywords search[C]//Proceedings of International Conference on Trusted Systems. 2015: 394-406.

[作者简介]



袁承昊 (1998-), 男, 山东东营人, 北京交通大学硕士生, 主要研究方向为云计算安全。



李勇 (1973-), 男, 山东无棣人, 北京交通大学副教授, 主要研究方向为数据安全和隐私保护、区块链安全。



任爽 (1981-), 男, 吉林长春人, 北京交通大学副教授, 主要研究方向为人工智能、区块链、虚拟现实技术。