中国科学 : 信息科学 2026 年 第 56 卷 第 7 期: 1730–1747
SCIENTIA SINICA Informationis
论文

《中国科学》杂志社
SCIENCE CHINA PRESS
CrossMark click for updates

支持多数据拥有者的动态可搜索加密方案

李洪军, 何德彪*, 冯琦, 罗敏

武汉大学国家网络安全学院, 空天信息安全与可信计算教育部重点实验室, 武汉 430072
* 通信作者. E-mail: hedebiao@163.com

收稿日期: 2025–10–01; 修回日期: 2025–12–16; 接受日期: 2026–01–19; 网络出版日期: 2026–06–30

国家自然科学基金 (批准号: 624B2108, 62325209, U23A20302) 资助项目

摘要 随着 5G 网络与云计算技术的快速发展, 多源数据融合共享已成为提升数据价值的重要手段. 然而, 传统数据管理模式难以有效应对数据融合共享过程中日益凸显的安全与隐私挑战. 动态可搜索加密作为一种新颖的密码原语, 支持在密态数据上执行高效搜索操作的同时允许用户动态地添加和删除数据. 针对多源数据环境下的安全需求, 本文设计了一种支持多数据拥有者/单数据使用者的动态可搜索加密方案. 该方案允许多个数据拥有者独立地对加密数据库进行批量更新, 并保证方案具备基于纪元的前向安全性与 Type-II 后向安全性. 一方面引入集合限制伪随机函数与密钥同态伪随机函数等技术, 降低数据使用者在授权查询中的通信与存储开销; 另一方面, 构建新颖的链式索引结构与批量更新算法, 支持检索过程并行化, 显著提升检索效率. 最后, 本文从理论分析与实验测试两个层面对方案的安全性、正确性与性能进行验证与评估, 结果表明该方案在多源数据共享场景中具有良好的应用前景.

关键词 可搜索加密, 隐私保护, 多数据拥有者, 前向安全性, 并行检索

1 引言

随着大数据时代的兴起, 各种结构化与非结构化类型的数据不断涌现 [1]. 根据国际数据公司 IDC 预测, 到 2028 年全球数据量将增长至 393.8 ZB [2], 其中超过 60% 的数据会存储在云服务器端. 云计算通过互联网按需付费的定价方式为用户提供充足的计算与存储资源, 降低客户端的储存成本与维护负担. 根据国际研究机构 Gartner 统计, 2024 年全球云服务市场规模达 1717.55 亿美元 [3]. 尽管云计算的出现为人们的工作与生活带来了极大的便利, 然而存储在远端云服务器的数据脱离了用户的管控, 其数据安全与隐私问题越来越引起人们的重视. 云服务器由于系统故障、黑客攻击等多种原因, 可能主动或被动地泄露用户的隐私数据, 造成个人财产的损失与恶劣的社会影响. 为确保机密数据不被泄露, 用户在上传数据之前通常进行加密处理, 确保敏感数据的安全性与隐私性 [4]. 然而, 传统加密算法会破坏数据的一些原始属性, 比如可检索性、可统计性、可计算性等. 尽管用户可以将密态数据下载到本地解密并检索, 但会导致云服务器与客户端之间巨大的通信开销. 与此同时, 随着我国信息化与数字化进程的不断推进, 多源数据融合已成为智慧医疗、数字政府等典型应用场景中的迫切需求. 在智慧医疗环境中, 不同医疗机构分别掌握着电子病历、医学影像和基因组信息等大量异构数据; 在数字政

引用格式: 李洪军, 何德彪, 冯琦, 等. 支持多数据拥有者的动态可搜索加密方案. 中国科学: 信息科学, 2026, 56: 1730–1747, doi: 10.1360/SSI-2025-0420
Li H J, He D B, Feng Q, et al. Dynamic searchable encryption scheme supporting multiple data owners. Sci Sin Inform, 2026, 56: 1730–1747, doi: 10.1360/SSI-2025-0420

© 2026《中国科学》杂志社

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

府建设中, 各职能部门亦拥有分散的政务数据资源. 通过对多数据拥有者的数据进行协同分析, 有助于提升疾病预测、辅助决策和公共治理的科学性与稳健性. 然而, 如何在保障数据隐私与安全的前提下, 实现跨主体、跨域的数据高效共享与联合利用, 仍然面临诸多关键技术挑战. 可搜索加密技术 [5], 作为一种新型密码原语, 允许用户在密态数据上进行检索操作, 兼顾数据的安全性、隐私性与可检索性. 其基本框架包含以下三类角色: 数据拥有者、数据使用者以及云服务器. 数据拥有者从文档集合中提取由关键词与文件标识符组成的键值对索引, 利用自己持有的密钥生成密文并上传到云服务器. 数据使用者利用密钥和待检索关键词生成搜索陷门并上传到云服务器. 随后, 云服务器根据加密索引与关键词搜索陷门进行检索并将匹配的文件标识符返回给数据使用者.

可搜索加密技术主要分为公钥可搜索加密与对称可搜索加密. 公钥可搜索加密 [6] 在理论上支持多数据拥有者环境, 但在实际应用中仍面临着诸多问题. 一方面, 该类方案通常依赖复杂且耗时的密码学运算, 严重制约在多用户、高频查询场景中的检索效率; 另一方面, 公钥可搜索加密机制难以有效抵御关键词猜测攻击, 严重威胁用户数据的安全性与隐私性. 对称可搜索加密使用哈希函数、伪随机函数等轻量级密码运算, 其检索效率较高但仅支持单数据拥有者/单数据使用者场景或者数据拥有者与数据使用者为同一用户场景. 尽管单用户可搜索加密方案可以通过一些简单直接的方法拓展到多数据拥有者场景, 比如所有数据拥有者共享同一个加密密钥 [7] 或者数据使用者和每一个数据拥有者进行密钥协商 [8]. 然而, 这些方式会降低可搜索加密系统的安全性, 造成较高的通信开销. 此外, 在实际数据管理中数据通常不是静态的而是随时间不断发生更新演化, 数据拥有者会增加或者删除某些文件. 相比于静态可搜索加密, 数据动态更新过程会增加额外的信息泄露量. Zhang 等 [9] 提出一种名为文件注入攻击的方法, 敌手可以通过插入少量预设置文件, 观察返回结果来恢复出用户的查询信息. 为降低添加和删除文件时的信息泄露量, 在设计动态可搜索加密方案时需要考虑前向安全性和后向安全性. 前向安全性确保新插入的文档无法被之前的关键词查询陷门相关联. 后向安全性确保先添加后删除的文档不能被之后的关键词查询陷门检索. 在多数据拥有者环境下, 各个数据拥有者独立维护自身数据集合, 数据更新行为频繁且异步, 进一步提高了此场景下动态可搜索加密方案设计复杂度. 因此, 如何在多数据拥有者场景下实现高效、可扩展且具备前向与后向安全性的动态可搜索加密方案, 是当前急需解决的问题.

为解决上述问题, 本文设计一个支持批量更新的动态可搜索加密方案, 并基于集合限制伪随机函数、密钥同态伪随机函数等密码学工具, 实现前向安全性和后向安全性的同时, 适配多数据拥有者/单数据使用者场景. 本文的主要贡献总结为以下几点.

- 针对数据融合中多数据拥有者/单数据使用者场景, 基于链式索引结构与批量更新技术, 设计了一个具有纪元前向安全性和 Type-II 后向安全性的动态可搜索加密方案, 支持检索过程并发执行, 有效提高检索效率.
- 基于集合限制伪随机函数进行密钥管理与分发, 降低数据使用者与数据拥有者之间的通信开销, 通信复杂度降低为 $O(1)$. 此外, 基于密钥同态伪随机函数技术, 对多个数据拥有者的加密索引进行安全合并, 进一步提升检索效率, 并降低数据使用者的存储开销.
- 对支持多数据拥有者的动态可搜索加密方案的安全性模型与定义进行全面刻画, 并基于安全性模型给出了详细的安全性证明. 此外, 对方案的计算与存储复杂度进行理论分析与实验评估, 展现所提方案与其他方案在检索效率和存储开销等方面的优势.

2　相关工作

根据本文的研究主题, 接下来主要从动态可搜索加密与多用户可搜索加密两个方面对近些年来的研究工作进行分析与总结.

(1) 动态可搜索加密 (dynamic searchable encryption, DSE). Kamara 等 [10] 正式提出了 DSE 方案的安全性定义, 并构建了第一个安全的 DSE 方案, 具有选择关键词攻击下的自适应安全性. Zhang 等 [9] 提出了一种针对 DSE 方案的攻击方法, 名为文件注入攻击, 敌手使用少量的文件就可以恢复出客户完整的查询信息. 为了缓解这一攻击带来的信息泄露, Bost [11] 基于陷门置换函数和伪随机函数, 提出了第一个具备前向安全性的 DSE 方案, 在更新和搜索过程中具有较低的计算与通信开销. Song 等 [12] 指出 Bost 方案采用复杂的公钥密码原语构

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

建, 其 I/O 性能较差, 严重降低了方案的检索效率, 并基于对称密码原语提出了两个具有前向安全性的 DSE 方案, 与 Bost 方案具有相同的通信效率且显著提高了 I/O 效率. 随后, Bost 等 [13] 正式定义了 3 种级别的后向安全性, 并基于范围限制伪随机函数与可穿刺加密技术, 提出了多种同时满足前向安全性和不同级别后向安全性的 DSE 方案. Sun 等 [14] 提出了一种新颖的对称可撤销加密原语, 构建了具备前向安全性和 Type-II 后向安全性的非交互式 DSE 方案, 结合可穿刺伪随机函数与布隆过滤器技术, 支持高效的删除操作并减少更新过程中的信息泄露量. 基于树型索引结构, Chamani 等 [15] 设计了两种存在删除操作下具备最优或近似最优搜索效率的 DSE 方案, 其中第一个方案具有与结果大小线性相关的渐近最优搜索时间; 第二个方案达到子对数级的搜索复杂度, 进一步提升了删除效率并降低了服务器存储开销. He 等 [16] 提出了一种由关键词链与文档链组成的新型索引结构 “鱼骨链”, 确保前向与后向安全性的同时无需为每个关键词存储对应的状态, 实现了恒定客户端存储开销. 基于 “循环加密链” 结构, 陈等 [17] 设计了一种支持查询结果隐藏的高效 DSE 方案, 确保前向安全性的同时有效降低了计算与存储开销. 基于误操作报警机制和不经意映射技术, Zheng 等 [18] 设计了一个具备鲁棒性和恒定客户端存储开销的 DSE 方案, 具有前向安全性和 Type-III 后向安全性. Song 等 [19] 提出一个 DSE 通用框架, 采用集合范围约束伪随机函数并结合全局计数器, 确保前向安全性和后向安全性的同时, 将客户端存储开销压缩为常数级.

(2) 多用户可搜索加密 (multi-user searchable encryption, MUSE). 多用户可搜索加密方案根据应用场景的不同可分为单数据拥有者/多数据使用者、多数据拥有者/单数据使用者, 以及多数据拥有者/多数据使用者 3 种场景. 在单数据拥有者/多数据使用者场景下, Shi 等 [20] 提出一种结合 B+ 树与计数布隆过滤器的高效 MUSE 方案, 支持安全验证、动态更新与多用户查询等功能, 不仅提升了查询与验证效率, 还通过引入时间戳机制防止重放攻击. Kermanshahi 等 [21] 基于可随机化的分布式密钥同态伪随机函数, 提出了一种支持多关键词合取查询、用户撤销和高效密钥更新等功能的 MUSE 方案, 在确保数据安全性与隐私性的同时, 可抵抗多方合谋攻击, 降低密钥泄露风险. Wang 等 [22] 将对称可搜索加密与公钥可搜索加密相结合, 基于可聚合身份基加密与基于纪元的前向安全 DSE 方案, 设计了一种支持多数据拥有者/单数据使用者的混合可搜索加密方案, 支持恒定陷门尺寸与不同数据拥有者子集检索. Zhang 等 [23] 提出一种具备前向与后向安全性、支持非交互式的动态 MUSE 方案, 采用轻量级存储链统一管理关键词状态以降低客户端的存储开销, 并引入 “隐藏密钥” 技术与时间范围查询实现前向安全性. Li 等 [24] 结合函数秘密共享与安全多方计算技术, 设计一种支持数据动态更新与任意布尔查询的 MUSE 方案, 隐藏搜索、访问与体积等模式. 在多数据拥有者/单数据使用者场景下, 侯等 [25] 提出了一种多数据拥有者场景下具有访问和搜索模式隐藏的 MUSE 方案, 用户只需要向服务器发送一个关键词搜索陷门, 就能在不同密钥加密的文件上进行检索. Le 等 [26] 利用分布式点函数、分布式伪随机函数等技术, 构建了一个新颖的 MUSE 方案, 支持读写分离与用户权限撤销, 在降低用户开销的同时隐藏所有统计信息. 在多数据拥有者/多数据使用者场景下, Mei 等 [27] 提出了一种支持数据动态更新的 MUSE 方案, 将前向安全性进一步细分成客户端前向安全性与服务器前向安全性, 并利用集合限制伪随机函数减少授权和查询的交互开销. Wang 等 [28] 利用非承诺身份基加密与同态加密技术提出了一种具备前向安全性的 MUSE 方案, 采用零知识证明检测恶意数据用户. 本文方案和之前方案的对比如表 1 [12, 16, 19, 22, 27, 29] 所示.

3　预备知识

3.1　密钥同态伪随机函数

给定密钥空间 $\mathcal{K}$、输入空间 $\mathcal{X}$ 以及输出空间 $\mathcal{Y}$, 对于给定的 $F(k_1, x)$ 和 $F(k_2, x)$, 如果存在一个高效的算法计算 $F(k_1 \oplus k_2, x) = F(k_1, x) \otimes F(k_2, x)$, 那么就说一个伪随机函数 $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ 具有密钥同态性质. 一个多方密钥同态伪随机函数 (key homomorphic pseudo-random function, KHF) 包含以下几个子算法.

- $k \leftarrow KHF.Gen(1^\lambda)$: 给定一个安全参数 λ, 算法输出一个密钥 $k \in \mathcal{K}$.
- $y \leftarrow KHF.Eval(k, x)$: 给定一个密钥 $k \in \mathcal{K}$ 和一个随机字符串 $x \in \{0,1\}^*$, 算法输出函数评估结果 $y =$

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

表 1　本文方案和之前方案的对比.

Table 1　Comparison between this scheme and previous schemes.

Scheme	Multiple data owners	Forward security	Backward security	Parallel search	Storage overhead
[12]	×	Regular	×	×	$O(\|W\|)$
[16]	×	Regular	Type-II	×	$O(1)$
[19]	×	Regular	Type-II	✓	$O(1)$
[22]	✓	Epoch	×	×	$O(\|W\|)$
[27]	✓	Regular	×	×	$O(1)$
[29]	×	Regular	Type-III	×	$O(\|W\|)$
Our	✓	Epoch	Type-II	✓	$O(1)$

$KHF(k,x)\in\mathcal{Y}$.

• $KHF(k_1\oplus...\oplus k_m,x)\leftarrow KHF(k_1,x)\otimes...\otimes KHF(k_m,x)$: 给定多个伪随机函数评估结果 $\{KHF(k_1,x),...,KHF(k_m,x)\}$, 算法输出 $KHF(k_1\oplus...\oplus k_m,x)$.

假设 $\mathbb{G}$ 是一个阶为素数 q 的循环群, H 是一个哈希函数: $\mathbb{G}\leftarrow\mathcal{X}$, 在随机谕言机模型中, 一个密钥同态伪随机函数 $KHF_{DDH}:\mathbb{G}\leftarrow\mathbb{Z}_q\times\mathcal{X}$ 被定义为 $KHF_{DDH}(k,x)\leftarrow H(x)^k$. 我们可以看到 $KHF_{DDH}(k_1+k_2,x)=KHF_{DDH}(x)^{k_1}\cdot KHF_{DDH}(x)^{k_2}$. Naor 等[30] 在随机谕言机模型下已经证明, 如果 DDH 假设成立, 那么 KHF_{DDH} 是一个安全的伪随机函数. 此外, 基于带错误学习问题可以构造一个标准模型下安全的密钥同态伪随机函数[31].

3.2　集合限制伪随机函数

给定密钥空间 $\mathcal{K}$、输入空间 $\mathcal{X}$ 以及输出空间 $\mathcal{Y}$, 一个伪随机函数 $F:\mathcal{Y}\leftarrow\mathcal{K}\times\mathcal{X}$ 可以通过一个确定多项式时间的函数计算, 给定输入 $(k,x)\in\mathcal{K}\times\mathcal{X}$, 输出 $F(k,x)\in\mathcal{Y}$. 一个集合限制伪随机函数 (set-constrained pseudo-random function, SCF) 是被限制在集合 $\mathcal{S}\subseteq 2^{\mathcal{X}}$ 上的[32], 如果存在一个额外的密钥空间 $\mathcal{K}_C$ 和以下两个额外的子算法 $SCF.Con$ 与 $SCF.Eval$.

• $K_S\leftarrow SCF.Con(k,S)$: 一个随机多项式时间的算法, 以密钥 $k\in\mathcal{K}$ 和集合 $S\in\mathcal{S}$ 为输入, 输出限制密钥 $K_S\in\mathcal{K}_C$, 其可以评估所有的 $x\in S$ 的伪随机函数值.

• $F(k,x)/\perp\leftarrow SCF.Eval(K_S,x)$: 一个确定多项式时间的算法, 以限制密钥 $K_S\in\mathcal{K}_C$ 和 $x\in\mathcal{X}$ 为输入. 对于伪随机函数密钥 $k\in\mathcal{K}$, 如果 K_S 是 $SCF.Con(k,S)$ 输出, $SCF.Eval(K_S,x)$ 输出

$$SCF.Eval(K_S,x)=\begin{cases}F(k,x), & \text{if}\quad x\in S,\\ \perp, & \text{otherwise.}\end{cases}$$

3.3　对称加密算法

给定密钥空间 $\mathcal{K}$、明文空间 $\mathcal{M}$ 与密文空间 $\mathcal{C}$, 一个对称加密算法 (symmetric encryption, SE) 包含以下 3 个多项式时间的子算法 $(SE.Gen,SE.Enc,SE.Dec)$.

• $k\leftarrow SE.Gen(1^\lambda)$: 密钥生成算法以安全参数 λ 为输入, 输出一个密钥 $k\in\mathcal{K}$.

• $c\leftarrow SE.Enc(k,m)$: 加密算法以密钥 $k\in\mathcal{K}$ 和明文 $m\in\mathcal{M}$ 为输入, 输出一个密文 $c\in\mathcal{C}$.

• $m\leftarrow SE.Dec(k,c)$: 解密算法以密钥 $k\in\mathcal{K}$ 和密文 $c\in\mathcal{C}$ 为输入, 输出一个明文 m 或者 $\perp$.

定义1 (SE 的 PCPA 安全性)　对于任意概率多项式时间内的敌手 $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$, 其具备加密谕言机查询权限 $\mathcal{O}_k^{SE.Enc(\cdot)}$, 如果存在一个可忽略函数 $negl(\lambda)$, 使得在伪随机选择明文攻击 (pseudo-randomness chosen-

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

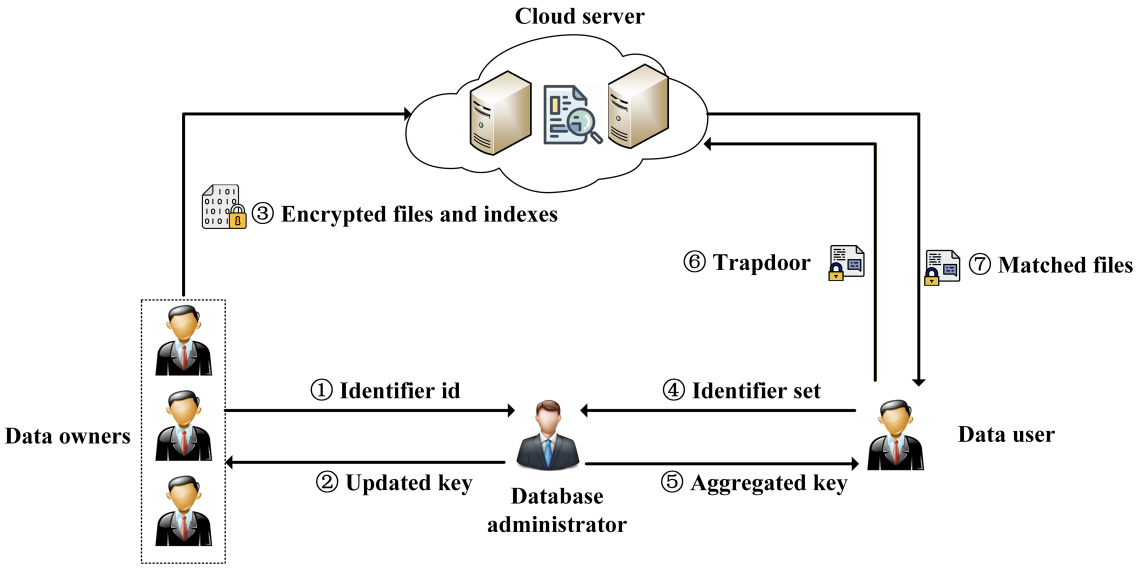


图 1　(网络版彩图) 系统模型.

Figure 1　(Color online) System model.

plaintext attack, PCPA) 安全实验中区分真实加密与随机密文的优势满足

$$Adv_{\mathcal{A},SE}^{PCPA} = \left| Pr\left[\begin{array}{l} k \leftarrow SE.Gen(1^{\lambda}), b \leftarrow \{0,1\}; \\ (m^*, st) \leftarrow \mathcal{A}_1^{\mathcal{O}_k^{SE.Enc(\cdot)}}(1^{\lambda}); \\ \text{If } b = 0, c^* \leftarrow SE.Enc(k, m^*); \\ \text{Else } c^* \leftarrow \mathcal{C}; \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_k^{SE.Enc(\cdot)}}(st, c^*); b = b' \end{array} \right] - \frac{1}{2} \right| \leqslant negl(\lambda),$$

称该 SE 方案在 PCPA 下是计算不可区分的.

4　问题描述

4.1　系统模型

如图 1 所示, 系统模型中共包含 4 个实体: 数据库管理员 (database administrator, DA)、云服务器 (cloud server, CS)、多个数据拥有者 (data owners, DOs) 与数据使用者 (data user, DU).

- DA: 拥有主密钥 K, 可以根据数据拥有者的身份 id 为其生成相应的加密密钥, 并可以根据选定的数据拥有者身份标识集合为数据使用者生成聚合密钥 AK.
- CS: 存储加密数据库, 根据更新陷门更新加密数据库, 根据搜索陷门检索并更新加密数据库.
- DOs: 将每个纪元内更新文件标识/关键词集合 IN 加密生成相应更新陷门并上传到云服务器. 此外, 为避免 DA 掌握全部密钥, 每一个数据拥有者会生成部分私钥并上传至云服务器.
- DU: 根据关键词生成搜索陷门并上传云服务器, 解密检索结果获得匹配的文件标识符集合.

方案的整体执行流程如下: DO 向 DA 提交自己的身份标识 id_j, DA 根据主密钥 K 生成对应的更新密钥 sk_j. 每个 DO 利用 sk_j 生成状态信息 st, 并根据批量更新集合 IN 和部分私钥 psk_j 生成加密更新索引, 将密文索引与部分私钥上传到 CS 上, 对加密数据库 EDB 进行更新. 在进行检索时, DU 向 DA 提交待检索的 DO 身份标识集合 ID, DA 根据主密钥生成对应的聚合密钥 AK. DU 根据 AK, 待检索关键词 w 以及当前纪元信息

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

e, 生成关键词检索陷门并上传到 CS 进行检索, CS 将匹配文件标识符结果与对应的部分私钥返回给 DU 进行解密并更新数据库.

在本文的威胁模型中将 CS 视为半诚实的实体, 这意味着 CS 会诚实地执行协议, 遵循算法规范来提供可靠的存储服务, 但会尝试从更新陷门、搜索陷门以及检索结果中学习关于 DO 和 DU 的一些隐私信息. DA, DO 与 DU 被视为诚实的实体, 其没有动机发起任何的恶意行为. 本文通过防止敌手 $\mathcal{A}$ 学习超出其应该知道的内容来保护诚实的实体 DA, DU 以及 DO. 在具有集中化管理需求的系统中, 找到一个可信第三方 DA 是可行的, 比如政务系统的数据交换平台和大型企业内部数据共享平台, 可以基于可信执行环境构建密钥分发服务器, 进行密钥管理与分发, 这种架构已在金融、政务、医疗等领域被广泛采用.

4.2　批量更新的动态可搜索加密

$DB = (f_i, W_i)_{i\in[1,n]}$ 表示 n 个文件标识/关键词集合, 其中 f_i 代表第 i 个文件标识符, W_i 代表第 i 个文件中包含的关键词集合. $|DB|$ 和 $|W|$ 分别代表文件标识符数量和关键词数量, $DB(w) = \{f_i | w \in W_i\}$ 代表包含关键词 w 的文件标识符集合, $IN = \{(op, f, w)\}$ 代表批量更新集合. 一个支持批量更新的动态可搜索加密方案包含以下几个多项式时间算法.

- $((sk, st); EDB) \leftarrow Setup(1^\lambda, DB; \perp)$: 以安全参数 λ 和明文数据库 DB 为输入, 输出一个私钥 sk 和一个状态值 st 给客户端, 一个空的加密数据库 EDB 给云服务器.
- $(st'; EDB') \leftarrow Update(sk, st, IN; EDB)$: 以客户端密钥 sk, 状态 st 以及更新集合 IN 和云服务器端的加密数据库 EDB 为输入, 输出一个新的状态值 st' 和新的加密数据库 EDB'.
- $((st', DB(w)); EDB') \leftarrow Search(sk, st, w; EDB)$: 以客户端的密钥 sk、状态 st, 关键词 w 以及云服务器端的加密数据库 EDB 为输入, 输出一个更新的状态 st' 与包含关键词 w 的文件标识符集合 $DB(w)$ 给客户端, 一个更新的加密数据库 EDB' 给云服务器.

4.3　安全性定义

本方案遵循 “真实 – 理想” 范式, 将信息泄露量控制在泄露函数 $\mathcal{L} = \{\mathcal{L}_{Setup}, \mathcal{L}_{Update}, \mathcal{L}_{Search}\}$ 之内. “真实 – 理想” 模拟范式中分为真实世界和理想世界. 在真实世界 $Real^{\Pi}_{\mathcal{A}}$ 中, 协议按照预定步骤执行, 敌手 $\mathcal{A}$ 可以观察到真实的交互记录. 在理想世界 $Ideal^{\Pi}_{\mathcal{A},\mathcal{S}}$, 模拟器 $\mathcal{S}$ 根据泄露函数模拟交互记录并返回给敌手 $\mathcal{A}$. 定义 $Real^{\Pi}_{\mathcal{A}}$ 与 $Ideal^{\Pi}_{\mathcal{A},\mathcal{S}}$ 的流程, 如下所示.

- $Real^{\Pi}_{\mathcal{A}}$: 敌手 $\mathcal{A}$ 选择一个数据库 DB, 挑战者 $\mathcal{C}$ 运行 $Setup(1^\lambda, DB; \perp)$ 并返回加密的数据库 EDB 给 $\mathcal{A}$. 然后, 敌手 $\mathcal{A}$ 可以自适应选择更新集合 IN_i, 挑战者 $\mathcal{C}$ 运行 $(st'; EDB') \leftarrow Update(sk, st, IN_i; EDB)$ 并返回给敌手 $\mathcal{A}$; $\mathcal{A}$ 可以自适应选择查询关键词 w_i, 挑战者 $\mathcal{C}$ 运行 $((st', DB(w_i)); EDB') \leftarrow Search(sk, st, w_i; EDB)$ 并返回给 $\mathcal{A}$. 最后, $\mathcal{A}$ 输出一个比特值 $b \in \{0, 1\}$.
- $Ideal^{\Pi}_{\mathcal{A},\mathcal{S}}$: 敌手 $\mathcal{A}$ 选择一个数据库 DB, 给定泄露函数 $\mathcal{L}_{Setup}(DB)$, 模拟器 $\mathcal{S}$ 生成一个加密的数据库 $EDB \leftarrow \mathcal{S}(\mathcal{L}_{Setup}(DB))$. 然后, 敌手 $\mathcal{A}$ 可以自适应选择更新集合 IN_i, 模拟器 $\mathcal{S}$ 运行 $\mathcal{S}(\mathcal{L}_{Update}(IN_i))$ 并返回给 $\mathcal{A}$; $\mathcal{A}$ 可以自适应选择查询关键词 w_i, 模拟器 $\mathcal{S}$ 运行 $\mathcal{S}(\mathcal{L}_{Search}(w_i))$ 并返回给 $\mathcal{A}$. 最后, $\mathcal{A}$ 输出一个比特值 $b \in \{0, 1\}$.

定义2 ($\mathcal{L}$- 自适应安全性)　如果对于任意概率多项式时间内的敌手 $\mathcal{A}$, 存在一个概率多项式时间内的模拟器 $\mathcal{S}$, 使得 $|Pr[Real^{\Pi}_{\mathcal{A}}(\lambda) = 1] - Pr[Ideal^{\Pi}_{\mathcal{A},\mathcal{S}}(\lambda) = 1]| \leqslant negl(\lambda)$, 可搜索加密方案 Π 是 $\mathcal{L}$- 自适应安全的.

定义 Q 表示迄今为止发起所有的查询列表, 搜索查询表示为 (μ, w), 更新查询表示为 (μ, op, f, w), 其中 μ 表示每回查询的时间戳, 每查询一次都会自增一次. 关键词 w 的搜索模式 $sp(w)$ 定义为 $sp(w) = \{\mu | (\mu, w) \in Q\}$. 关键词 w 的访问模式 $ap(w)$ 定义为 $ap(w) = DB(w)$. 包含关键词 w 且未被删除的文件标识符列表 $TimeDB(w)$ 定义为 $TimeDB(w) = \{(\mu, f) | (\mu, add, f, w) \in Q \wedge \forall \mu', (\mu', del, f, w) \notin Q\}$. 关键词 w 的更新历史 $Update(w)$ 定义为 $Update(w) = \{\mu | (\mu, add, f, w) \in Q \vee (\mu, del, f, w) \in Q\}$, 表示所有更新操作发生的时间戳. 关键词 w 的删除历

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

表 2　符号定义.

Table 2　Symbol definition.

Symbol	Definition	Symbol	Definition
λ	Security parameter	K	Master key
e	Epoch information	AK	Aggregated key
id_j	Identifier of DO_j	ID	Identifier set of DO
sk_j	Updated key of id_j	psk_j	Partial private key of id_j
H_1, H_2, H_3	Three hash functions	SCF	Set-constrained pseudo-random function
KHF	Key homomorphic pseudo-random function	F	Pseudo-random function
SE	Symmetric encryption algorithm	ek_j^w	File encryption key
$IN_j^{w,e}$	Batch update set	R	Matching file identifier set

$IN_j^{w_1,e}$	(op, f_1, w_1)	(op, f_2, w_1)	(op, f_3, w_1)
$IN_j^{w_2,e}$	(op, f_2, w_2)	(op, f_3, w_2)	(op, f_4, w_2)
$IN_j^{w_3,e}$	(op, f_1, w_3)	(op, f_2, w_3)	(op, f_4, w_3)

$IN_j^{w_1,e'}$	(op, f_5, w_1)	(op, f_6, w_1)
$IN_j^{w_2,e'}$	(op, f_6, w_2)	(op, f_7, w_2)
$IN_j^{w_3,e'}$	(op, f_3, w_3)	(op, f_5, w_3)

图 2　(网络版彩图) 批量更新集合.

Figure 2　(Color online) Batch update set.

史 $Del(w)$ 定义为 $Del(w) = \{(\mu^{add}, u^{del}) | \exists f : (\mu^{add}, add, f, w) \in Q \wedge (\mu^{del}, del, f, w) \in Q\}$, 表示删除操作发生的时间戳以及其对应添加操作发生的时间戳.

前向安全性意味着添加操作不会泄露先前搜索的关键词是否在新添加的文件中. 而基于纪元的前向安全性意味着当前纪元中添加的文件不会匹配到先前纪元搜索的关键词, 其正式定义如下.

定义3 (纪元前向安全性)　更新泄露函数可以表示为 $\mathcal{L}_{Update} = \mathcal{L}'(|IN|, (op_i, f_i)_{i\in[IN]})$. 那么, 一个动态可搜索加密算法 Π 满足纪元前向安全性.

定义4 (后向安全性)　后向安全性意味着对于关键词 w 的搜索查询不会泄露已经删除的文件标识. Bost 等 [11] 正式定义了后向安全性, 并根据信息泄漏量的不同将后向安全性分为 3 个等级.

- **Type-I 后向安全性**: 如果泄露函数可以表示成 $\mathcal{L}_{Update}(op, f, w) = \mathcal{L}'(op)$ 和 $\mathcal{L}_{Serach}(w) = \mathcal{L}''(TimeDB(w), a_w)$, 其中 a_w 代表关键词 w 的更新次数.
- **Type-II 后向安全性**: 如果泄露函数可以表示成 $\mathcal{L}_{Update}(op, f, w) = \mathcal{L}'(op, w)$ 和 $\mathcal{L}_{Serach}(w) = \mathcal{L}''(TimeDB(w), Updates(w))$, 意味着泄露关键词 w 的更新次数以及每次更新的时间戳.
- **Type-III 后向安全性**: 如果泄露函数可以表示成 $\mathcal{L}_{Update}(op, f, w) = \mathcal{L}'(op, w)$ 和 $\mathcal{L}_{Serach}(w) = \mathcal{L}''(TimeDB(w), Del(w))$, 意味着泄露关键词 w 的更新次数, 每次添加的时间戳以及删除对应的添加操作的时间戳.

5　方案设计

5.1　整体框架

设计的方案适用于多数据拥有者/单数据使用者场景, 其中每个数据拥有者会批量更新文件标识符/关键词键值对, 其更新集合如图 2 所示. 这种批量更新技术已经在多个数据库系统中进行应用, 比如 MYSQL 中的 replace into 命令和 Amazon DynamoDB 中的 BatchWriteItem 命令, 可以有效减少数据库与应用程序的交互次数, 提高数据处理的吞吐量. 本文的符号说明如表 2 所示.

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

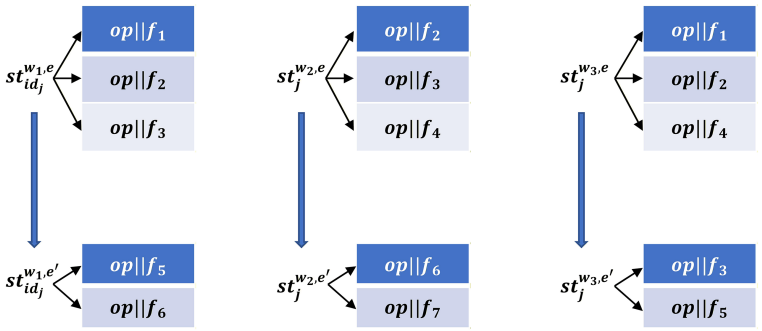


图 3　(网络版彩图) 加密索引结构.

Figure 3　(Color online) Encrypted index structure.

本方案解决的第一个难点就是如何在支持多数据拥有者的同时降低各个实体之间的通信开销与数据使用者的存储开销. 在多数据拥有者/单数据使用场景下, 尽管数据使用者可以和每一个数据拥有者之间进行密钥协商获得可搜索加密密钥. 然而这种方式需要与数据使用者进行多轮交互, 通信开销较大. 本方案通过引入一个可信的数据库管理者, 并基于集合限制伪随机函数进行密钥分发. 集合限制伪随机函数可以将多个数据拥有者的密钥聚合生成一个密钥, 该密钥可以根据数据拥有者的身份标识生成对应的密钥. 数据库管理员只需和数据使用者进行一轮交互, 且交互中数据使用者只需要提交待检索数据拥有者的身份标识集合, 数据库管理员仅需返回一个聚合密钥. 此外, 为避免数据库管理员掌握全部密钥, 每个数据拥有者生成各自的部分私钥并上传至云服务器, 进行辅助检索. 为了确保数据使用者可以检索到上一次检索结果, 需要存储每个关键词在上一次搜索时所对应的状态信息. 在本方案中为了降低数据使用者端的存储开销, 采用密钥同态伪随机函数确保数据使用者仅仅只需保存上次检索发生的纪元 e'、待检索的身份标识集合以及聚合密钥 AK.

本方案解决的第二个难点就是如何确保可搜索加密方案的前向安全性和后向安全性. 在本方案中, 将纪元 e 作为批量更新的基础单位, 其长度可以根据用户需求进行设定. 比如以 “一天” 作为纪元单位, 数据拥有者会在每一天结束时对这一天之内本地更新的文件标识符/关键词集合加密更新到云服务器上. 本方案通过引入了纪元 e 作为批量更新与同步控制的时间粒度基础, 以保证多个 DO 在高并发环境下对索引结构更新的原子性与一致性. 以统一纪元标识 e、相应的更新关键词集合, 以及每个 DO 持有的更新密钥与部分私钥作为输入, 通过密钥同态伪随机函数生成唯一的更新状态值. 如图 3 所示, 纪元 e 和数据拥有者对应的密钥生成状态信息用以更新加密数据库, 确保前一个纪元的搜索陷门不能匹配当前纪元下更新的数据记录, 实现基于纪元的前向安全性. 每个纪元之间通过链式结构进行链接, 即当前纪元中的初始地址中存储有上一个纪元的状态信息. 本方案中加密数据库的结构可以表示为 $EDB = EDB_1 \cup ... EDB_j \cup ... \cup EDB_m$, 其中 $EDB_j = EDB_j^1 \cup ... \cup EDB_j^e$ 表示数据拥有者 id_j 在纪元 e 之前所有批量更新集合的并集. 每个加密数据库中存储的数据类型为键值对 $(addr, value)$, 其中 $addr$ 代表存储地址, $value$ 代表加密内容.

本方案解决的第三个难点是如何实现并行检索, 提高检索效率. 本方案主要采用以下两种方法. 首先, 对于数据拥有者 id_j 来说, 在纪元 e 中的初始地址对应的值中, 将匹配标识符的数量作为加密索引的一部分进行额外加密. 数据使用者在进行检索时, 可以提前获得匹配文件标识符的数量并分配多线程进行并行检索. 其次, 可以将多个数据拥有者批量更新的子数据库进行合并, 提高检索效率并降低通信开销. 如图 4 所示, 以两个数据拥有者 id_i 和 id_j 在两个纪元 e 和 e' 的更新加密索引为例. 首先, 将每个数据拥有者在两个更新纪元中的所有更新记录进行合并. 其次, 根据两个数据拥有者在纪元 e 的关键词状态信息 $st_i^{w_1,e}$ 和 $st_j^{w_1,e}$, 分别检索截至当前纪元的所有更新记录并将相应值存储在两个集合中. 当检索完成时, 云服务器利用 $st_i^{w_1,e}$ 和 $st_j^{w_1,e}$ 生成当前纪元的新的状态信息 $st^{w_1,e}$, 并根据新状态信息重新生成加密索引用以下一次检索.

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

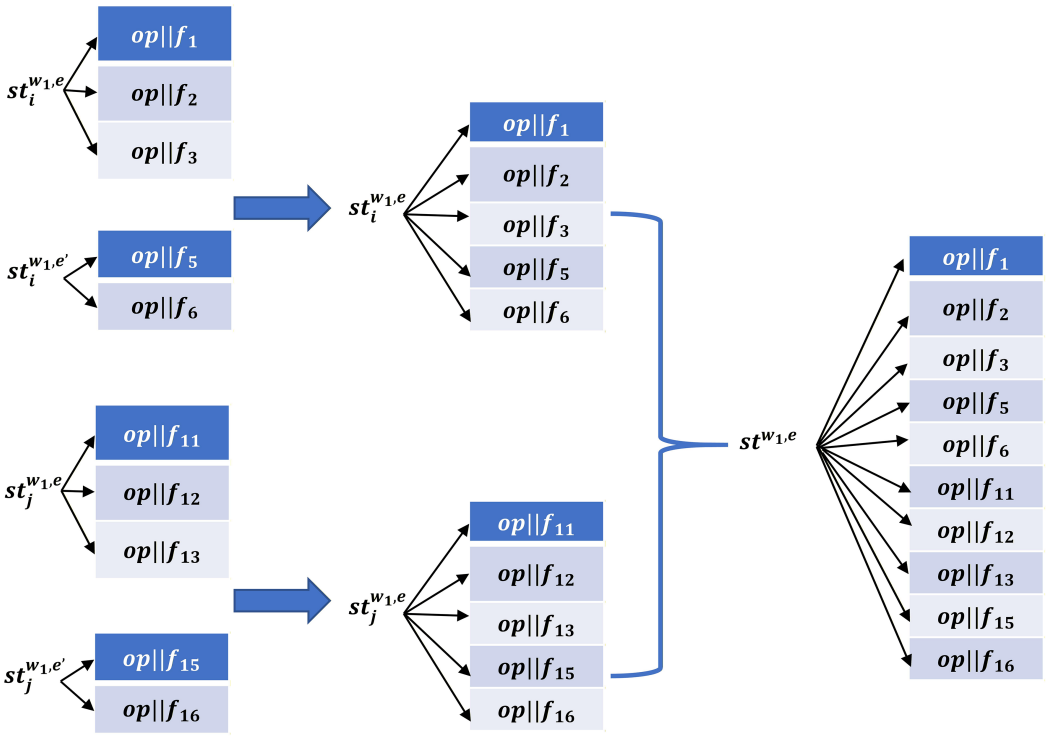


图 4　(网络版彩图) 搜索后加密索引结构.

Figure 4　(Color online) Index structure after search.

5.2　算法构建

DA 随机选择主密钥 K, 并根据主密钥 K 和 DO_j 发送的身份标识 id_j, 利用集合限制伪随机函数 $SCF: \{0,1\}^\lambda \times \{0,...,m\} \to \{0,1\}^\lambda$ 生成对应的更新密钥 sk_j. 每一个 DO_j 初始化映射集合 OWT_j, 存储关键词 w 上次更新时的状态值. DO_j 生成一个部分私钥 $\{0,1\}^\lambda \to psk_j$ 并发送给 CS. DU 初始化变量 UWT, 存储关键词 w 上次搜索发生的纪元. 纪元 e 是一个 λ 比特长的字符串, 由 DO 和 DU 双方共同从系统时间导出, 用以确保方案的前向安全性. 初始化算法的具体过程如算法 1 所示.

算法 1 初始化算法.

输入: 安全参数 λ, 数据拥有者数量 m 以及相应的标识集合 $ID = \{id_1, ..., id_m\}$;
输出: 更新密钥集合 $\{sk_1, ..., sk_m\}$;
1: DA 初始化密钥同态伪随机函数 KHF 系统参数与集合限制伪随机函数 SCF 系统参数;
2: DA 随机选择 $K \leftarrow \{0,1\}^\lambda$ 并作为系统主密钥;
3: **for** $id_j \in ID$ **do**
4:　DO_j 将身份标识 id_j 发送给 DA;
5:　DA 计算更新密钥 $sk_j = SCF(K, id_j)$ 并发送给 DO_j;
6:　DO_j 初始化加密数据库 $EDB_j = \emptyset$ 与关键词状态映射集合 $OWT_j = \emptyset$;
7:　DO_j 生成一个部分私钥 $psk_j \leftarrow \{0,1\}^\lambda$ 并发送给 CS;
8: **end for**
9: DU 初始化关键词状态变量 UWT;
10: 所有 DO_j 与 DU 将从系统时间导出的历元作为状态值 e.

根据更新密钥 sk_j 和 DO 的部分私钥 psk_j, 利用伪随机函数 $F: \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda$ 生成关键词的文件加密密钥 ek_j^w. F 是一个定义在 $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$ 上的确定性函数, 其中 $\mathcal{K}$ 是密钥空间, $\mathcal{X}$ 是输入空间, $\mathcal{Y}$ 是输出空间. 给定一个密钥 $k \in \mathcal{K}$ 和一个数据块 $x \in \mathcal{X}$, 输出 $y = F(k, x) \in \mathcal{Y}$. 具体来说, 基于每一个数据拥有者的更新

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

密钥 sk_j 和 DO 的部分私钥 psk_j 以及对应的关键词 w 作为数据块, 输出一个伪随机函数值 ek_j^w 作为文件密钥. 本文中伪随机函数实现是通过迭代伪随机发生器生成伪随机值.

利用哈希函数 $H_1 : \{0,1\}^\lambda \to \mathbb{Z}_q$ 和密钥同态伪随机函数 $KHF : \mathbb{Z}_q \times \{0,1\}^* \to \mathbb{G}$, 生成在当前纪元 e 下关于关键词 w 的状态信息 $st_j^{w,e}$. 在批量更新集合 $IN_j^{w,e}$ 中, 对于一个关键词 w 来说可能对应多个文件标识符 f. DO 计算部分私钥 psk_j 和状态值 $st_j^{w,e}$ 的伪随机函数值, 进而基于自增的计数器 i 并通过哈希函数 $H_2 : \{0,1\}^* \to \{0,1\}^\lambda$ 计算加密索引中的地址值 $addr$, 对应的值 $value$ 则用对称加密算法 $SE.Enc()$ 加密对应的操作 op 和文件标识符 f. 每个数据拥有者根据当前纪元 e 是否是第一个纪元 (定义 EOF 为初始纪元值), 计算关键词 w 在纪元 e 中的初始地址 $addr$, 对应 $value$ 的加密密钥是从状态值 $st_j^{w,e}$ 和部分私钥 psk_j 通过伪随机函数 F 和哈希函数 $H_3 : \{0,1\}^* \to \{0,1\}^\lambda$ 演化得到的, 其包含上一次更新状态 $st_j^{w,e'}$ 以及当前关键词的更新次数 i. 通过检索初始地址获得关键词 w 在纪元 e 中更新操作的次数, 进而支持检索过程并行化, 提高密态数据搜索效率. 每个数据拥有者更新对应的 OWT_j 中的存储值. 索引更新算法具体过程如算法 2 所示. 当批量更新集合 $IN_j^{w,e}$ 为空集时, 数据拥有者也需要计算 $addr = H_2(F(psk_j, st_j^{w,e})$ 和 $value = SE.Enc(K_t, st_j^{w,e'}||0)$ 并上传到云服务器上, 以确保之后的检索能够顺利进行.

算法 2 索引更新算法.

输入: 当前纪元 e, 每个数据拥有者身份标识符 id_j, 待更新文件标识/关键词集合 $IN_j^{w,e}$;
输出: 更新数据库集合 $\{EDB_1^e, ..., EDB_m^e\}$ 与关键词状态集合 $\{OWT_1, ..., OWT_m\}$;

每一个数据所有者 DO_j 生成关键词 w 对应的文件加密密钥 $ek_j^w = F(sk_j||psk_j, w)$;
for $id_j \in ID$ **do**
  for $w \in W^e$ **do**
    DO_j 生成在当前纪元 e 下关键词 w 的状态值 $st_j^{w,e} = KHF(H_1(sk_j), w||e)$;
    DO_j 设置随机数 $i = 1$;
    for $(op, f, w) \in IN_j^{w,e}$ **do**
      DO_j 计算 $addr = H_2(F(psk_j, st_j^{w,e}), i)$ 与 $value = SE.Enc(ek_j^w, op||f)$;
      DO_j 计算 $i = i + 1$ 并更新 $EDB_j^e = EDB_j^e \cup (addr, value)$;
    end for
    DO_j 计算 $addr = H_2(F(psk_j, st_j^{w,e}), 0)$ 与 $K_t = H_3(F(psk_j, st_j^{w,e}))$;
    if $e = EOF$ **then**
      DO_j 计算 $value = SE.Enc(K_t, 0||i)$ 并设置 $OWT_j[w] = st_j^{w,e}$;
    else
      DO_j 设置 $st_j^{w,e'} \leftarrow OWT_j[w]$ 并计算 $value = SE.Enc(K_t, st_j^{w,e'}||i)$;
      DO_j 设置 $OWT_j[w] = st_j^{w,e}$;
    end if
    CS 更新 $EDB_j^e = EDB_j^e \cup (addr, value)$;
  end for
end for

DA 根据选定的 DO 标识集合 ID 计算对应的聚合密钥 AK, DU 利用聚合密钥 AK, 根据 DO 的身份标识 id_j, 分别生成检索密钥 sk_j、状态信息 $st_j^{w,e}$. 此外, DU 根据本地存储的上次检索的纪元信息 e' 生成检索状态信息 $st^{w,e'}$. CS 可以根据搜索陷门检索每个 DO_j 在当前纪元 e 与上回检索纪元 e' 之间所有的更新记录, 具体如下: 对于每一个加密数据库 EDB_j, CS 初始化集合 V_j, 根据检索陷门 st_j^w 和部分私钥 psk_j 计算得到初始地址 $addr$, 解密对应的 $value$ 获得纪元 e 内关键词 w 的更新次数 i 以及上一纪元的状态信息 $st_j^{w,e-1}$, 并将相应的记录从原始数据库集合中删除; 之后根据获得的更新次数 i 以及检索陷门 st_j^w 和部分私钥 psk_j 并行检索数据库 EDB_j, 将检索结果 $value$ 添加进集合 V_j 中并将相应的记录从数据库中删除. 将当前的状态值 st_j^w 和加密数据库 EDB_j 设置为上一纪元的状态信息 $st_j^{w,e-1}$ 和加密数据库 EDB_j^{e-1}, 继续进行检索直至加密数据库 EDB_j 为空. 另一方面, CS 可以根据检索状态信息 $st^{w,e'}$ 检索纪元 e' 之前的更新记录, 具体过程如下: CS 初始化集合 V', 根据检索陷门 $st^{w,e'}$ 计算得到初始地址 $addr$, 解密对应的 $value$ 获得纪元 e 内关键词 w 的更新次数 $|EDB^{e'}|$; 之后根据获得的更新次数 i 与检索陷门 $st^{w,e'}$ 并行检索数据库 $EDB^{e'}$, 将检索结果 $value$ 添加进集合 V' 中. CS

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

获得最终检索结果 V 并返回给 DU. 此外, CS 对关键词 w 在当前纪元 e 之前的所有记录进行合并, 其状态信息利用密钥同态伪随机函数更新为 $st^{w,e}$, 部分私钥设置为 psk, 进而生成初始地址 $addr$ 和对称加密密钥 K_t, 加密更新次数并更新加密数据库 EDB^e, 将之前所有的更新记录添加进新数据库 EDB^e 中. 在获得匹配记录 V 之后, DU 利用相应的文件密钥 ek_j^w 解密对应的值获得文件标识符 f 和操作 op, 根据 $op = add/del$ 确定最后的匹配结果. 密文检索算法具体过程如算法 3 所示.

算法 3 密文检索算法.

输入: 当前纪元 e, 待查询数据拥有者身份标识集合 $ID = \{id_1, .., id_{m'}\}$, 待查询关键词 w;
输出: 更新数据库 EDB^e, $UWT[w]$ 以及匹配文件标识集合 R;

DA 利用集合限制伪随机函数 SCF 生成聚合密钥 $AK = SCF.Con(K, ID)$ 并发送给 DU;
for $id_j \in ID$ **do**
　DU 计算 $sk_j = SCF.Eval(AK, id_j)$;
　DU 计算 $st_j^{w,e} = KHF(H_1(sk_j), w||e)$;
end for
$e' = UWT[w]$;
if $e' = \emptyset$ **then**
　$st^{w,e'} = 0$;
else
　DU 计算 $st^{w,e'} = KHF(H_1(sk_1) \oplus ... \oplus H_1(sk_{m'}), w||e')$;
end if
DU 将 $\{st_1^{w,e}, ..., st_{m'}^{w,e}, st^{w,e'}\}$ 发送给 CS;
CS 设置 $psk = 0$;
for $j \in [m']$ **do**
　CS 设置 $EDB_j = EDB_j^e$ 和 $st_j^w = st_j^{w,e}$;
　$psk = psk \oplus psk_j$;
　while $EDB_j \neq \emptyset$ **do**
　　CS 初始化集合 $V_j = \emptyset$;
　　CS 计算 $addr = H_2(F(psk_j, st_j^w), 0)$ 与 $K_t = H_3(F(psk_j, st_j^w))$;
　　CS 检索 $value = EDB_j[addr]$ 并解密得到 $st_j^{w,e-1}||i = SE.Dec(K_t, vlaue)$;
　　CS 更新 $EDB_j = EDB_j/(addr, value)$;
　　for $t \in [i]$ **do**
　　　CS 并行计算 $addr = H_2(F(psk_j, st_j^w), t)$;
　　　CS 并行检索 $value = EDB_j[addr]$;
　　　CS 将 $value||j$ 添加进 $V_j = V_j \cup value||j$;
　　　CS 更新加密数据库 $EDB_j = EDB_j/(addr, value)$;
　　end for
　　CS 设置状态 $st_j^w = st_j^{w,e-1}$ 和加密数据库 $EDB_j = EDB_j^{e-1}$;
　end while
end for
CS 初始化集合 $V' = \emptyset$;
if $st^{w,e'} \neq 0$ **then**
　CS 计算 $addr = H_2(F(psk, st^{w,e'}), 0)$ 与 $K_t = H_3(F(psk, st^{w,e'}))$;
　CS 检索获得 $v = EDB^{e'}[addr]$;
　CS 计算 $|EDB^{e'}| = SE.Dec(K_t, v)$;
　for $c \in [|EDB^{e'}|]$ **do**
　　CS 计算 $addr = H_2(F(psk, st^{w,e'}), c)$;
　　CS 并行检索 $v = EDB^{e'}[addr]$;
　　CS 将 v 添加进 $V' = V' \cup v$;
　end for
end if
CS 计算 $V = V_1 \cup ... \cup V_{m'} \cup V'$;
CS 计算 $st^{w,e} = st_1^{w,e} \otimes ... \otimes st_{m'}^{w,e}$;
CS 计算 $addr = H_2(F(psk, st^{w,e}), 0)$ 与 $K_t = H_3(F(psk, st^{w,e}))$;
CS 计算 $v = SE.Enc(K_t, |V|)$;
CS 更新数据库 $EDB^e = EDB^e \cup (addr, v)$;
CS 初始化 $c = 1$;
for $v \in [V]$ **do**
　CS 计算 $addr = H_2(F(psk, st^{w,e}), c)$;
　CS 更新数据库 $EDB^e = EDB^e \cup (addr, v)$;
　CS 计算 $c = c + 1$;
end for
CS 发送 V 和相应的 psk_j 给 DU;
DU 计算对应的文件加密密钥 $ek_j^w = F(sk_j||psk_j, w)$;
DU 使用相应的 ek_j^w 解密 V 获得文件标识符集合 R;
DU 设置 $UWT = e$.

6　安全性分析

定理1 ($\mathcal{L}$- 自适应安全性)　H_1, H_2 与 H_3 是 3 个可被模拟成随机谕言机的哈希函数, KHF 是一个密钥同态伪随机函数, SCF 是一个集合限制伪随机函数, SE 方案是一个 PCPA 安全的对称加密算法. 泄露函数 $\mathcal{L} = \{\mathcal{L}_{Setup}, \mathcal{L}_{Update}, \mathcal{L}_{Search}\}$ 定义为

$$\mathcal{L}_{Setup} = \perp;\ \mathcal{L}_{Search} = (sp(w), TimeDB(w), Update(w));\ \mathcal{L}_{Update} = (|IN|, \{op_i, f_i\}).$$

那么, 设计的方案是一个 $\mathcal{L}$- 自适应安全的动态可搜索加密方案.

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

证明　首先从 $Real_{\mathcal{A}}^{\Pi}$ 开始, 随后定义了一系列的模拟游戏, 逐步证明每两个游戏之间的不可区分性, 并将最后一个游戏设置为 $Ideal_{\mathcal{A},\mathcal{S}}^{\Pi}$. 通过证明 $Real_{\mathcal{A}}^{\Pi}$ 与 $Ideal_{\mathcal{A},\mathcal{S}}^{\Pi}$ 之间的计算不可区分性, 最终证明定理 1.

游戏 G_1: 游戏 G_1 与游戏 $Real_{\mathcal{A}}^{\Pi}$ 唯一不同的点是不再通过哈希函数 H_1 计算 $H_1(sk_j)$, 而是维护一个哈希状态表 T_{H_1}. 每当接收一个身份 id_j, 如果 id_j 存在于列表 T_{H_1} 中则返回相应的元素; 否则, 随机选择一个 λ 比特长度的随机字符串 h_1 返回并将 (id_j, h_1) 存储在列表 T_{H_1} 中.

游戏 G_2: 游戏 G_2 与游戏 G_1 唯一不同的点是不再通过密钥同态伪随机函数 KHF 为每一个 $w||e||j$ 生成相应的状态值 $st_{id_j}^{w,e}$, 而是维护一个映射关系表 T_F. 每当接收一个关于 $w||e||j$ 的查询, 如果 $w||e||j$ 存在于 T_F 就返回相应的元素; 否则, 随机从群 $\mathbb{Z}_q$ 中选择一个群元素 $st_{id_j}^{w,e}$ 并将 $(w||e||j, st_{id_j}^{w,e})$ 存储到列表 T_F 中. 很明显可以证明 G_2 与 G_1 是等价的, 否则可构建一个敌手 $\mathcal{B}_1$ 区分 KHF 和一个真正的伪随机函数. 因此, $|Pr[G_1 = 1] - Pr[G_2 = 1]| \leqslant negl(\lambda)$.

游戏 G_3: 游戏 G_3 与游戏 G_2 唯一不同的点是不再通过哈希函数 H_2 计算 $addr = H_2(st_{id_j}^{w,e}, i)$, 而是将 H_2 模拟成一个随机谕言机. 通过维护一个哈希状态表 T_{H_2}, 随机选择一个 λ 比特长度的字符串 h_2 并设置 $T_{H_2}[w||e||j||i] = h_2$. 在搜索阶段, 将哈希函数 H_2 编码成如下形式:

$$
\begin{aligned}
&i \leftarrow 0, h_2 \leftarrow T_{H_2}[w||e||j||i];\\
&\textbf{while } h_2 \neq \perp \textbf{ do}\\
&\quad H_2(st_{id_j}^{w,e}, i) \leftarrow h_2;\\
&\quad i \leftarrow i+1, h_2 \leftarrow T_{H_2}[w||e||j||i].
\end{aligned}
$$

游戏 G_3 除了在查询 H_2 时有极小概率存在不一致的情况, 游戏 G_3 与游戏 G_2 在敌手的视角上是行为一致的. 存在不一致的原因在于 H_2 并不是立刻更新的, 只是在搜索前被定义好的. 如果敌手 $\mathcal{A}$ 在下一次搜索之前查询了某些 $(st_{id_j}^{w,e}, i)$, 有很大的概率返回 $h_2' \neq h_2$, 因为 $T_{H_2}[w||e||j||i]$ 没有被更新, 在这种情况下会选择一个随机的字符串 h_2'. 对于下一次搜索查询, 敌手 $\mathcal{A}$ 查询 $H_2(st_{id_j}^{w,e}, i)$. $H_2(st_{id_j}^{w,e}, i)$ 的值被更新为 h_2, 并作为随机谕言机查询返回结果. 这就是游戏 G_3 与游戏 G_2 之间唯一的不同之处 (表示成 BAD 事件). 因此, 可以得到 $Pr[G_3 = 1] - Pr[G_2 = 1] \leqslant Pr[BAD]$. 因为 $st_{id_j}^{w,e}$ 是一个随机的 λ 比特长度的字符串, 敌手 $\mathcal{A}$ 查询 h_2 的概率等于 $2^{-\lambda}$. 敌手 $\mathcal{A}$ 会对 H_2 进行 q_2 次的查询. 因此, $Pr[BAD] \leqslant q_2/2^{\lambda} \leqslant negl(\lambda)$.

游戏 G_4: 游戏 G_4 与游戏 G_3 唯一不同的点是用从密文域 $\mathcal{C}$ 中随机选择字符串代替由 $SE.Enc$ 算法加密生成的密文. 如果敌手 $\mathcal{A}$ 能够区分 G_4 和 G_3, 那么存在一个算法 $\mathcal{B}_2$ 能够打破对称加密方案 SE 的 PCPA 安全性. 因此, $Pr[G_4 = 1] - Pr[G_3 = 1] \leqslant Adv_{SE,\mathcal{B}_2}^{PCPA}(\lambda) \leqslant negl(\lambda)$.

游戏 G_5: 游戏 G_5 与游戏 G_4 唯一不同的点是将 T_{H_2} 的索引标记由 $w||e||j||i$ 替换为 $e||j||i$. 具体而言, 将每一次更新定义为 $(e, j||i)$, 其中 e 代表更新纪元, j 代表第 j 个数据拥有者, i 代表在纪元 e 中的更新次数. 显而易见地, G_5 和 G_4 是不可区分的. 因此, $Pr[G_5 = 1] = Pr[G_4 = 1]$.

$Ideal_{\mathcal{A},\mathcal{S}}^{\Pi}$: 模拟器 $\mathcal{S}$ 可以访问被允许的泄露函数去生成和游戏 G_5 计算不可区分的视图. 和游戏 G_5 唯一的不同, 在模拟游戏 $Ideal_{\mathcal{A},\mathcal{S}}^{\Pi}$ 中不知道关键词 w, $\mathcal{S}$ 使用泄露函数一对一映射从 w 到计数器 $\overline{w||e||j} = sp(w||e||j)$. $\mathcal{S}$ 利用关键词 w 的更新历史 $Update(w)$ 编码哈希函数 H_2, 具体过程如算法 4 所示. 因此, $Pr[Ideal_{\mathcal{A},\mathcal{S}}^{\Pi} = 1] = Pr[G_5 = 1]$. 综上所述, 我们有 $|Pr[Real_{\mathcal{A}}^{\Pi} = 1] - Pr[Ideal_{\mathcal{A},\mathcal{S}}^{\Pi} = 1]| \leqslant negl(\lambda)$.

定理2 (前向安全性和后向安全性)　H_1, H_2 与 H_3 是被模拟成随机谕言机的三个哈希函数, KHF 是一个密钥同态伪随机函数, SCF 是一个集合限制伪随机函数, SE 是一个满足 PCPA 安全的加密算法, 则设计的方案具备前向安全性和 Type-II 后向安全性.

证明　在更新阶段, DO 将键值对 $(addr, value)$ 上传到 CS. 由于纪元 e 的存在, 敌手 $\mathcal{A}$ 在不知道更新密钥 sk_j 的情况下, 无法将关键词 w 在本纪元的更新陷门与之前任何纪元的搜索陷门相关联. 因此, 本方案能够确保前向安全性. 在搜索阶段, CS 返回所有匹配的记录. 对于给定的关键词查询, CS 可以获知所有更新或删除操作的时间戳. 然而, CS 无法判断某个标识符 f 是否在其被添加后, 在随后的某次更新操作中被删除, 因为 $op||f$ 是

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

算法 4 模拟器 $\mathcal{S}$.

S.Setup():

$e \leftarrow EOF$;
$T_{H_1}, T_{H_2}, T_F \leftarrow \emptyset$;

S.Update():

$EDB_j^e \leftarrow \emptyset$;
$e = e + 1$;
for $id_j \in ID$ **do**
  for $i \in [0, |IN_j^{w,e}|]$ **do**
    $h_{2,j}^i \leftarrow \{0,1\}^\lambda$;
    $T_{H_2}[e, j, i] \leftarrow h_{2,j}^i$;
    $v_j^i \leftarrow \mathcal{C}$;
    $EDB_j^e \leftarrow EDB_j^e \cup (h_{2,j}^i, v_j^i)$;
  end for
  发送 EDB_j^e 给云服务器 CS;
end for

S.Search():

$h_{1,j} \leftarrow T_{H_1}[id_j]$;
$\overline{w||e||j} \leftarrow sp(w||e||j)$;
$st_{id_j}^{\overline{w||e||j}} \leftarrow T_F[\overline{w||e||j}]$;
发送 $st_{id_j}^{\overline{w||e||j}}$ 给云服务器;
模拟器 $\mathcal{S}$ 在搜索之前编写哈希函数 H_2;
$t \leftarrow |Update(w)|$;
for $id_j \in ID$ **do**
  $\{e_c, j, i_c\}_{c\in[t]} \leftarrow Update(w)$;
  for $1 \leqslant c \leqslant t$ **do**
    $H_2(st_{id_j}^{\overline{w||e||j}}, i) \leftarrow T_{H_2}[e_c, j, i_c]$;
  end for
end for

经过 SE 算法加密的. 因此, 本方案能够保证 Type-II 后向安全性.

7　方案评估

7.1　理论分析

本节主要对提出的方案中各个实体的计算开销与存储开销以及实体间的通信开销进行理论分析. 设置数据拥有者数量为 m, 待查询的数据拥有者数量为 m'. 定义批量更新次数为 um, 关键词个数为 km, 每一次批量更新的关键词包含的文件标识的数量为 fm. 在本方案中, DA 主要负责生成每个数据拥有者的更新密钥 sk_j 以及为数据使用者生成聚合密钥 AK. 如果采用基于哈希的集合限制伪随机函数 SCF, 在更新阶段 DA 需要执行 m 次哈希操作生成 m 个更新密钥, 其通信开销为 $m \cdot \lambda$ 比特, 在检索阶段 DA 需要执行 $m - m'$ 次哈希操作生成一个聚合密钥 AK, 其通信开销为 λ 比特. 在更新阶段, 每一个 DO 需要为每一个关键词执行一次密钥同态伪随机函数, 执行 $fm+3$ 次哈希运算与伪随机函数操作以及 $fm+1$ 次对称加密算法. 因此, 在每一次批量更新中, DO 的计算开销为 km 次的密钥同态伪随机函数操作, $km(fm+3)$ 次哈希操作以及 $km(fm+1)$ 次对称加密操作. 每一个 DO 在每一次批量更新操作中的通信开销为 $km(fm+1)(\lambda+l)$ 比特, 其中 l 为对称加密中的密文长度. 在更新阶段, 每一个 DO 需要为每一个关键词存储相应的状态信息 $st_{id_j}^{w,e}$. 因此每一个 DO 的存储开销为 $|W|\lambda$, 其中 $|W|$ 代表关键词集合中元素的个数. 在密文检索阶段, DU 需要执行 m' 次集合限制伪随机函数, $m'+1$ 次密钥同态伪随机函数操作以及 $2m'$ 次哈希运算, 其通信开销为 $(m'+1)\lambda$ 比特, 其存储包含上一次检索的纪元信息 e' 以及聚合密钥 AK, 其存储开销为 2λ 比特. 此外, CS 将检索结果以及对应的部分私钥返回给 DU, 其通信开销为 $|DB(w)|l + m'\lambda$ 比特. 在解密阶段, DU 的计算开销为 m' 次伪随机函数运算.

7.2　实验验证

本节进行了若干实验测试与对比对方案的性能进行评估. 在实验中, 每个实体均使用笔记本电脑进行模拟, 其具体配置为: AMD Ryzen 9 6900HS with Radeon Graphics 3.30 GHz 处理器, 8 GB 内存, 操作系统为 Ubuntu 20.04.5 LTS. 实验使用的编程语言为 C++, 使用的密码学库为 MIRACL, 实验中所有哈希函数均采用 SHA-512 实现, 密态数据库用 Redis 库存储, 对称加密算法采用 AES.

在第一个实验中, 首先展现在 DO 数量为 10 个与关键词数量为 10 个的情况下, 设计的方案在不同更新次数与更新文件数量下各个阶段的时间开销, 其中索引更新阶段的时间开销为每一个 DO 的平均时间开销, 密文检索开销为 DU 检索某一个 DO 加密数据库的时间开销. 如表 3 所示, 在保持总体关键词/文件标识符数量不变的情况下, 随着批量更新次数的增加, 索引更新的时间开销与密文检索的时间开销显著增长. 因为其过程

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

表 3　不同更新次数与更新文件数量下各个阶段时间开销 (ms) ($km = 10$).

Table 3　Time cost (ms) of each stage under different update times and numbers of updated files ($km = 10$).

Phase	Index update (average)	Trapdoor generation	Ciphertext search (average)
$um = 1, fm = 10000$	9309.59	0.55	742.37
$um = 10, fm = 1000$	9337.08	0.56	749.46
$um = 20, fm = 500$	9465.11	0.57	804.69
$um = 100, fm = 100$	10514.07	0.55	834.96
$um = 500, fm = 20$	11700.97	0.56	876.89
$um = 1000, fm = 10$	13864.77	0.58	1060.23

表 4　不同关键词数量与更新文件标识符数量下各个阶段时间开销 (ms) ($um = 10$).

Table 4　Time cost (ms) of each stage under different numbers of keywords and updated files ($um = 10$).

Phase	Index update (average)	Trapdoor generation	Ciphertext search (average)
$km = 1, fm = 10000$	9222.63	0.58	9279.51
$km = 10, fm = 1000$	9511.76	0.59	765.37
$km = 20, fm = 500$	9622.11	0.56	381.59
$km = 100, fm = 100$	9820.07	0.53	75.00
$km = 500, fm = 20$	10878.52	0.57	16.01
$km = 1000, fm = 10$	11710.21	0.438	7.56

表 5　不同 DO 数量下各个阶段时间开销 (ms) ($um = 10, fm = 100, km = 10$).

Table 5　Time cost (ms) of each stage under different numbers of DO ($um = 10, fm = 100, km = 10$).

Phase	Index update (average)	Trapdoor generation	Ciphertext search	Ciphertext search (average)
DO=10	1024.48	0.63	951.58	95.16
DO=15	1045	0.66	1271.25	84.75
DO=20	1043.59	0.83	1772.78	88.64
DO=25	1018.73	0.97	2254.80	90.19
DO=30	1009.28	1.27	2581.50	86.05

中状态值生成次数与对称加密算法执行次数和更新频率线性相关. 陷门生成阶段的时间开销基本保持稳定, 在 0.55∼0.58 ms 之间波动, 因为其只和 DO 的数量相关. 其次, 在固定批量更新的次数 $um = 10$ 的前提下评估了在不同关键词数量与更新文件标识符数量组合下, 所提方案在索引更新、陷门生成与密文检索三个阶段的时间开销. 如表 4 所示, 在保持总体关键词/文件标识符数量不变的情况下, 随着关键词数量的增加与更新文件标识数量的减少, 索引更新阶段的时间开销呈现缓慢上升趋势. 例如, 关键词数量从 1 增加至 1000 时, 索引更新时间由 9222.63 ms 增加至 11710.21 ms. 这主要是因为更多的关键词数量导致更新过程中执行多次状态值计算以及对称加密算法, 从而引入额外的计算负担. 因为匹配文件的数量等于更新次数与批量更新文件数量之积, 密文检索阶段的时间开销则随着匹配文件数量的减少而显著下降, 从初始的 9279.51 ms 降至仅 7.56 ms, 这表明密文检索阶段性能高度依赖于待匹配文件的数量.

如表 5 所示, 在固定系统参数 $um = 10, fm = 100, km = 10$ 的条件下, 我们分析了不同 DO 数量对系统各阶段时间开销的影响. 实验结果表明, 单个 DO 的索引更新时间开销随 DO 数量变化基本保持稳定, 说明索引更新主要受批量更新数据规模的影响, 体现了方案在多源环境下的良好可扩展性. 相较而言, 陷门生成阶段的时间开销随 DO 数量呈现轻微线性增长. 此外, 密文检索阶段的时间开销随 DO 数量显著增加, 这是由于服务器需要处理更多来自不同 DO 的索引密文, 从而增加了匹配与解密操作的计算负担.

在不同的文件标识符数量与关键词数量下, 将本方案与 Song 等方案 [12]、He 等方案 [16] 以及 Chen 等方

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

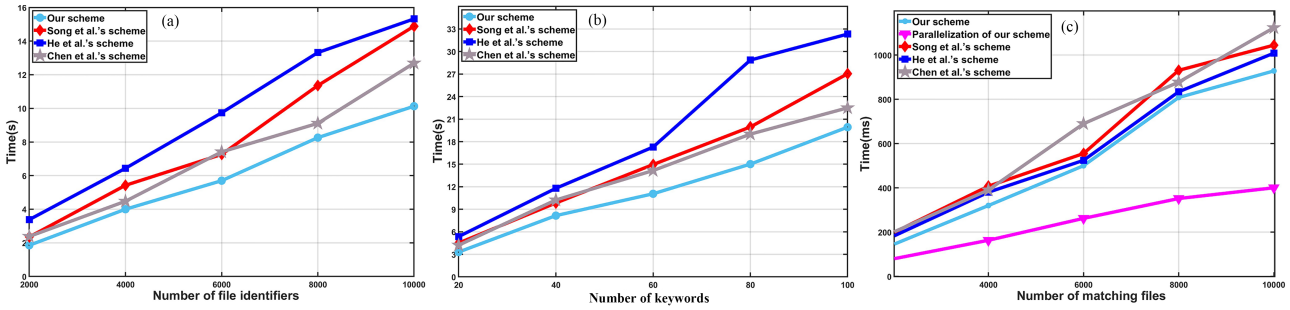


图 5　(网络版彩图) 本方案和其他方案的性能对比. (a) 不同文件数量下更新时间开销; (b) 不同关键词数量下更新时间开销; (c) 不同匹配文件数量下检索时间开销.

Figure 5　(Color online) Performance comparison of this solution and other solutions. (a) Update time cost with different numbers of file identifiers; (b) update time cost with different numbers of keywords; (c) search time cost with different numbers of matching files.

表 6　不同更新密文数量下各阶段性能开销对比 (ms) (DO=10, $km = 10$).

Table 6　Performance overhead (ms) comparison at different stages for different amounts of updated ciphertext (DO=10, $km = 10$).

Number of ciphertexts	Scheme	Ciphertext update (average)	Trapdoor generation	Ciphertext search
6000	PAEKS [33]	3684.00	44.84	14018.28
	Our	636.86	0.52	51.91
8000	PAEKS [33]	5114.4	47.67	20024.51
	Our	867.06	0.45	71.02
10000	PAEKS [33]	6158	46.195	23393.05
	Our	1027.61	0.42	84.91

案 [29] 中的更新索引时间进行对比. 固定关键词的数量为 10, 图 5(a) 中显示随着文件标识符数量的增加, 所有方案的时间开销均呈上升趋势, 但增长幅度存在显著差异. 所提出的方案在各个规模下均表现出优于其他对比方案的性能, 尤其在文件标识符数量达到 10000 时, 其更新时间仅为约 10.13 s, 显著低于 He 等方案的 15.32 s 和 Song 等方案的 14.88 s, 体现出更高的更新效率. 固定文件标识数量为 2000 (本方案中更新次数为 2, 每次更新的文件标识数量为 1000), 图 5(b) 显示随着关键词数量的增长, 各方案的索引更新时间整体呈上升趋势. He 等方案的时间开销增长最为显著, 在关键词数量为 100 时, 其更新时间达到 32.32 s, 明显高于其他方案. Song 等方案与 Chen 等方案在此设置下分别达到 27.04 和 22.50 s. 相比之下, 本文方案始终保持较低的更新时间开销, 即使在关键词数量为 100 的情况下, 更新时间也仅为 19.93 s, 展现出良好的关键词扩展能力.

将本方案与 Song 等方案 [12]、He 等方案 [16] 以及 Chen 等方案 [29] 中的密文检索时间进行对比, 其中本方案并行化中采用双线程实现. 图 5(c) 显示, 随着匹配文件标识符数量的增加, 各方案的检索时间均呈线性增长趋势, 表明检索开销在很大程度上受匹配文件数量的影响. 所提方案整体性能优于 Song 等, He 等和 Chen 等 3 种方案, 尤其在匹配文件数为 10000 时, 其检索时间为 928.53 ms, 显著低于 Chen 等方案的 1123.49 ms 与 He 等方案的 1008.80 ms. 所提方案的并行化实现进一步降低检索延迟, 在相同条件下仅需 400.61 ms 即可完成密文检索, 验证方案多线程并发环境中的高效性.

如表 6 所示, 在固定参数 DO=10, $km = 10$ 的条件下, PAEKS 方案与本方案在不同更新密文规模下的性能开销随更新密文数量增加均呈上升趋势. 相比之下, 本方案在各阶段均表现出显著的性能优势. 在更新密文阶段, 本方案的时间开销随更新规模近似线性增长, 且始终远低于 PAEKS 方案. 在陷门生成阶段, 本方案保持亚毫秒级开销, 显著优于对比方案. 进一步地, 在密文搜索阶段, PAEKS 方案的时间开销随更新规模快速增长, 而本方案增长幅度较小, 始终维持在较低水平, 说明本方案在大规模动态更新场景下能够有效降低服务器端的匹配与解密开销.

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

8 总结

在面对多数据拥有者密态数据检索场景下, 公钥可搜索加密方案效率较低且易受关键词猜测攻击, 而传统对称可搜索加密方案交互轮数多且安全性较弱. 本文结合链式索引结构与密钥同态伪随机函数技术, 设计了一个支持多数据拥有者和批量更新的动态可搜索加密方案, 在确保基于纪元的前向安全性和 Type-II 后向安全的同时支持检索过程并行化, 可有效提高检索效率. 此外, 本方案基于集合限制伪随机函数进行密钥管理与分发, 将多个数据拥有者的密钥聚合成一个密钥, 可有效降低数据使用者的通信开销与存储开销. 最后, 通过详细的安全性分析与实验评估证明方案的安全性和高效性.

参考文献

1 Azad P, Navimipour N J, Rahmani A M, et al. The role of structured and unstructured data managing mechanisms in the Internet of Things. Cluster Comput, 2020, 23: 1185–1198
2 Wright A. Worldwide IDC global datasphere forecast, 2024–2028: AI everywhere, but upsurge in data will take time. International Data Corporation, 2024. https://my.idc.com/getdoc.jsp?containerId=US52712424
3 Gartner. Market share: services, worldwide, 2024. 2025. https://www.gartner.com/en/documents/6353779
4 Li J W, Sun J, Yang J Y, et al. Cloud-edge integrated secure storage architecture and technical challenges. Sci Sin Inform, 2025, 55: 516–527 [李经纬, 孙嘉, 杨劲远, 等. 云边融合安全存储架构及技术挑战. 中国科学: 信息科学, 2025, 55: 516–527]
5 Wang W, Lei S F, Liu D L, et al. Research on key leakage-verifiable searchable encryption with post-compromise security. Sci Sin Inform, 2025, 55: 2319–2338 [王蔚, 雷淑芳, 刘东立, 等. 后泄露安全下的密钥泄露可验证可搜索加密技术研究. 中国科学: 信息科学, 2025, 55: 2319–2338]
6 Kong X, Chen L, Zhu Y, et al. Efficient public-key searchable encryption scheme from PSI with scalable proxy servers. IEEE Trans Serv Comput, 2024, 17: 3527–3540
7 Liu C, Zhu L, Chen J. Efficient searchable symmetric encryption for storing multiple source dynamic social data on cloud. J Netw Comput Appl, 2017, 86: 3–14
8 Liang J, Qin Z, Xiao S, et al. Privacy-preserving range query over multi-source electronic health records in public clouds. J Parallel Distributed Computing, 2020, 135: 127–139
9 Zhang Y, Katz J, Papamanthou C. All your queries are belong to us: the power of file-injection attacks on searchable encryption. In: Proceedings of the 25th USENIX Security Symposium (USENIX Security), 2016. 707–720
10 Kamara S, Papamanthou C, Roeder T. Dynamic searchable symmetric encryption. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS), 2012. 965–976
11 Bost R. $\sum o\varphi o\varsigma$: Forward secure searchable encryption. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS), 2016. 1143–1154
12 Song X, Dong C, Yuan D, et al. Forward private searchable symmetric encryption with optimized I/O efficiency. IEEE Trans Dependable Secure Comput, 2018, 17: 912–927
13 Bost R, Minaud B, Ohrimenko O. Forward and backward private searchable encryption from constrained cryptographic primitives. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS), 2017. 1465–1482
14 Sun S F, Steinfeld R, Lai S, et al. Practical non-interactive searchable encryption with forward and backward privacy. In: Proceedings of the 28th Annual Network and Distributed System Security Symposium (NDSS), 2021
15 Chamani J G, Papadopoulos D, Karbasforushan M, et al. Dynamic searchable encryption with optimal search in the presence of deletions. In: Proceedings of the 31st USENIX Security Symposium (USENIX Security), 2022. 2425–2442
16 He K, Chen J, Zhou Q, et al. Secure dynamic searchable symmetric encryption with constant client storage cost. IEEE Trans Inform Forensic Secur, 2020, 16: 1538–1549
17 Chen H L, Ma C S. Circular encrypted chain based dynamic searchable symmetric encryption. J Cryptologic Res, 2022, 9: 949–968 [陈海龙, 马昌社. 基于循环加密链的高效动态可搜索对称加密方案. 密码学报, 2022, 9: 949–968]
18 Zheng Y, Xu P, Wang M, et al. Themis: robust and light-client dynamic searchable symmetric encryption. IEEE Trans Inform Forensic Secur, 2024, 19: 8802–8816
19 Song X, Zheng Y, Bai J, et al. Disco: dynamic searchable encryption with constant state. In: Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, 2024. 1724–1738
20 Shi Z, Fu X, Li X, et al. Esvsse: enabling efficient, secure, verifiable searchable symmetric encryption. IEEE Trans Knowl Data Eng, 2022, 34: 3241–3254
21 Kermanshahi S K, Liu J K, Steinfeld R, et al. Multi-client cloud-based symmetric searchable encryption. IEEE Trans Dependable

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

Secure Comput, 2019, 18: 2419–2437
22 Wang J, Chow S S. Omnes pro uno: practical multi-writer encrypted database. In: Proceedings of the 31st USENIX Security Symposium (USENIX Security), 2022. 2371–2388
23 Zhang H, Xu C, Lu R, et al. Non-interactive multi-client searchable symmetric encryption with small client storage. IEEE Trans Serv Comput, 2023, 16: 3972–3985
24 Li J, Wang J, Zhang R, et al. NEMO: practical distributed Boolean queries with minimal leakage. IEEE Trans Inform Forensic Secur, 2024, 19: 2594–2608
25 Hou X Y, Liu J L, Zhang X, et al. Symmetric searchable encryption with access and search pattern privacy in the multi-dataowner settings. J Cryptologic Res, 2023, 10: 1241–1265 [侯欣怡, 刘晋璐, 张茜, 等. 多数据所有者场景下具有访问模式和搜索模式隐私的对称可搜索加密方案. 密码学报, 2023, 10: 1241–1265]
26 Le T, Behnia R, Guajardo J, et al. MUSES: efficient multi-user searchable encrypted database. In: Proceedings of the 33rd USENIX Security Symposium (USENIX Security), 2024. 2581–2598
27 Mei L, Xu C, Xu L, et al. Practical multi-source multi-client searchable encryption with forward privacy: refined security notion and new constructions. IEEE Trans Dependable Secure Comput, 2023, 21: 63–77
28 Wang J, Chow S S. Unus pro omnibus: multi-client searchable encryption via access control. In: Proceedings of the 31st Annual Network and Distributed System Security Symposium (NDSS), 2024
29 Chen T, Xu P, Wang W, et al. Bestie: very practical searchable encryption with forward and backward security. In: Proceedings of the European Symposium on Research in Computer Security, 2021. 3–23
30 Naor M, Pinkas B, Reingold O. Distributed pseudo-random functions and kdcs. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, 1999. 327–346
31 Boneh D, Lewi K, Montgomery H, et al. Key homomorphic prfs and their applications. In: Proceedings of the Annual Cryptology Conference, 2013. 410–428
32 Boneh D, Waters B. Constrained pseudorandom functions and their applications. In: Proceedings of the Advances in Cryptology—ASIACRYPT 2013, 2013. 280–300
33 Huang Q, Li H. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Inf Sci, 2017, 403–404: 1–14

https://www.sciengine.com/doi/10.1360/SSI-2025-0420

Dynamic searchable encryption scheme supporting multiple data owners

Hongjun LI, Debiao HE*, Qi FENG & Min LUO

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China
* Corresponding author. E-mail: hedebiao@163.com

Abstract With the rapid advancement of 5G networks and cloud computing technologies, multi-source data integration and sharing have become key approaches to releasing the value of data. However, traditional data management paradigms are difficult to handle the growing security and privacy concerns associated with such integration and sharing. Dynamic searchable encryption, as a novel cryptographic primitive, enables efficient search operations over encrypted data while supporting dynamic updates such as data insertion and deletion. To address the security requirements in a multi-source data environment, this paper proposes a dynamic searchable encryption scheme that supports multiple data owners and a single data user. The proposed scheme allows multiple data owners to independently perform batch updates to an encrypted database while ensuring epoch-based forward security and Type-II backward security. To design the proposed scheme, advanced cryptographic tools such as set-constrained pseudo-random functions and key-homomorphic pseudo-random functions are employed to reduce communication and storage overhead during authorized queries by the data user. Additionally, a novel chained index structure and a batch update algorithm are introduced to enable parallelized search, significantly improving retrieval efficiency. Finally, comprehensive evaluations including theoretical analysis and experimental testing are conducted to show the scheme's security, correctness, and performance. The results demonstrate that the proposed scheme exhibits strong potential for practical deployment in multi-source data sharing scenarios.

Keywords searchable encryption, privacy protection, multiple data owners, forward security, parallel search

https://www.sciengine.com/doi/10.1360/SSI-2025-0420