软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007251] [CSTR: 32375.14.jos.007251]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

鲁棒的前后向隐私联合对称可搜索加密方案*

张文琪[1], 李 雄[1,2], 尹智明[3], 梁 伟[3], 黄 可[1], 张小松[1,2]

[1](电子科技大学 计算机科学与工程学院, 四川 成都 611731)

[2](电子科技大学 (深圳) 高等研究院, 广东 深圳 518110)

[3](湖南科技大学 计算机科学与工程学院, 湖南 湘潭 411201)

通信作者: 李雄, E-mail: lixiong@uestc.edu.cn

摘 要: 动态对称可搜索加密允许用户安全地搜索和动态更新存储在半可信云服务器中的加密文档, 近年来备受关注. 然而, 现有多数对称可搜索加密方案仅支持单关键词搜索, 无法在实现联合搜索的同时满足前向和后向隐私. 此外, 多数方案不具有鲁棒性, 即无法处理客户端重复添加或删除某个关键词/文件标识符对或删除不存在的关键词/文件标识符对等不合理更新请求. 针对上述挑战, 提出一个鲁棒的前后向隐私联合动态对称可搜索加密方案 RFBC. 在该方案中, 服务器为每个关键词建立两个布隆过滤器, 分别用于存储所要添加和删除的关键词/文件标识符对的相关哈希值. 当客户端发送更新请求时, 服务器利用两个布隆过滤器进行判断, 过滤不合理请求, 以满足方案的鲁棒性. 此外, 利用多关键词中最低频关键词的状态信息, 结合布隆过滤器与更新计数器, 筛选掉不包含其余关键词的文件标识实现联合查询. 通过定义方案的泄露函数, 经过一系列的安全性游戏证明 RFBC 支持前向隐私与 Type-III 后向隐私. 实验分析表明相较于相关方案, RFBC 较大幅度提高了计算和通信效率. 具体来说, RFBC 更新操作的计算开销分别为 ODXT 和 BDXT 的 28% 和 61.7%, 搜索操作的计算开销分别为 ODXT 和 BDXT 的 21.9% 和 27.3%, 而搜索操作的通信开销分别为 ODXT 和 BDXT 的 19.7% 和 31.6%. 而且, 当不合理更新的比例逐渐增加时, 搜索效率的提升明显高于 BDXT 与 ODXT.

关键词: 对称可搜索加密; 前向隐私; 后向隐私; 鲁棒性; 联合搜索

中图法分类号: TP309

中文引用格式: 张文琪, 李雄, 尹智明, 梁伟, 黄可, 张小松. 鲁棒的前后向隐私联合对称可搜索加密方案. 软件学报. http://www.jos.org.cn/1000-9825/7251.htm

英文引用格式: Zhang WQ, Li X, Yin ZM, Liang W, Huang K, Zhang XS. Robust Scheme for Conjunctive Symmetric Searchable Encryption with Forward and Backward Privacy. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7251.htm

Robust Scheme for Conjunctive Symmetric Searchable Encryption with Forward and Backward Privacy

ZHANG Wen-Qi[1], LI Xiong[1,2], YIN Zhi-Ming[3], LIANG Wei[3], HUANG Ke[1], ZHANG Xiao-Song[1,2]

[1](School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China)

[2](Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China)

[3](School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China)

Abstract: Dynamic searchable symmetric encryption has attracted much attention because it allows users to securely search and dynamically update encrypted documents stored in a semi-trusted cloud server. However, most searchable symmetric encryption schemes only support single-keyword search, failing to achieve conjunctive search while protecting forward and backward privacy. In addition, most schemes are not robust, which means that they cannot handle irrational update requests from a client, such as adding or deleting a certain

* 基金项目: 国家自然科学基金 (62332018, 62072078, 62271128); 四川省自然科学基金 (2022NSFSC0550)
收稿时间: 2023-08-28; 修改时间: 2024-02-20; 采用时间: 2024-07-08; jos 在线出版时间: 2024-12-04

keyword/file identifier pair, or deleting non-existent keywords/file identifier pairs. To address these challenges, this study proposes a robust scheme for conjunctive dynamic symmetric searchable encryption that preserves both forward and backward privacy, called RFBC. In this scheme, the server constructs two Bloom filters for each keyword, which are used to store the relevant hash values of the keyword/file identifier pair to be added and deleted, respectively. When the client sends update requests, the server uses the two Bloom filters to determine and filter irrational update requests, so as to guarantee the robustness of the scheme. In addition, by combining the status information of the lowest frequency keywords among multiple keywords, the Bloom filters, and the update counter, RFBC realizes conjunctive search by filtering out file identifiers that do not contain the rest keywords. Finally, by defining the leakage function, RFBC is proved to be forward private and Type-III backward private through a series of security analyses. Experimental results show that compared with related schemes, RFBC greatly improves computation and communication efficiency. Specifically, the computational overhead of update operations in RFBC is about 28% and 61.7% of that in ODXT and BDXT, respectively. The computational overhead of search operations in RFBC is about 21.9% and 27.3% of that in ODXT and BDXT, respectively. The communication overhead of search operations in RFBC is about 19.7% and 31.6% of that in ODXT and BDXT, respectively. Moreover, as the proportion of irrational updates gradually increases, RFBC exhibits significantly higher improvement in search efficiency compared to both BDXT and ODXT.

Key words: symmetric searchable encryption; forward privacy; backward privacy; robustness; conjunctive search

1 引 言

1.1 研究背景及意义

云存储是一种由云计算服务商提供的在线数据存储服务. 利用云存储, 用户可以将数据存储在远程服务器上, 并通过网络进行在线访问. 云存储服务具有存储空间动态可扩展、数据访问方式灵活、低成本高效率和数据自动化管理等优势. 然而, 云服务器往往并不是完全可信的, 云存储中用户数据控制权的减弱和数据访问的不透明使用户难以对数据进行监管, 存在用户隐私数据泄露的风险. 为了保护用户的数据隐私, 一种解决办法是客户端对数据加密处理后上传至服务器, 当用户使用数据时下载所有加密的文件, 但该方法通信和时间开销巨大.

对称可搜索加密 (symmetric searchable encryption, SSE)[1]是一种支持用户在密文上进行高效关键词查询的密码学原语, 能够确保用户查询关键词和文件等数据的隐私. 传统的可搜索加密是静态的, 不支持文件增删, 更无法保证更新时的安全性. 然而, 在工业互联网等实际应用中, 云服务器上的数据往往需要频繁更新. 针对静态对称可搜索加密方案在实际应用中的局限性, 学者们提出了动态对称可搜索加密 (dynamic symmetric searchable encryption, DSSE)[2]. 然而, 一些 DSSE 方案存在隐私泄露风险. 具体来讲, 用户更新数据时, 可能会泄露关键词、操作类型 (添加或删除) 及文件标识符等信息, 攻击者则可以利用这些信息实施文件注入等攻击[3], 造成安全威胁.

前向隐私[4]能够有效地抵抗文件注入攻击, 它能够防止搜索历史泄露未来的更新信息, 即使攻击者获取了过去的加密搜索日志, 也无法确定之后新添加的数据. 此后, 为了更好地维护历史删除数据的机密性和完整性, 学者们提出了后向隐私[4]的概念. 后向隐私能够确保后续的搜索无法检索到过去被删除的数据, 根据安全性的强度, 后向隐私又被分为 Type-I、Type-II 和 Type-III 后向隐私这 3 个等级. 这两个安全性概念的提出, 使学者们逐渐关注前后向隐私的可搜索加密方案.

许多前后向隐私可搜索加密方案着眼于保护用户和数据的隐私, 但大多方案不支持多关键词的联合搜索. 若用户想要通过多个关键词精准搜索某个文件, 但方案仅支持单关键词查询, 则用户需要通过多次查询及筛选才能获得想要的文件. 因此, 单关键词可搜索加密方案的搜索功能十分受限, 且效率低下.

此外, 现有的大多数 DSSE 方案都默认客户端提交的更新请求是合理的, 不会出现重复添加文件和删除不存在的文件这两种情况. 当响应大量不合理更新请求时, 会导致服务器存储空间的浪费和性能的下降, 必然会降低系统的鲁棒性, 甚至导致隐私泄露. 例如, 攻击者掌握某些文件信息, 那么它可以构造并上传大量无用文件, 以达到降低系统搜索效率的目的. 可见, 方案的鲁棒性也是可搜索加密的重要特性之一.

据我们所知, 当前少有能在满足前后向隐私的同时, 实现多关键词联合查询和处理不合理的更新请求的对称可搜索加密方案. 因此, 实现鲁棒的前后向隐私联合可搜索加密成为当前云计算安全领域的一个新的挑战. 针对可搜索加密方案鲁棒性 (robustness)、前后向隐私 (forward and backward privacy) 和联合查询 (conjunctive) 问题, 基

于布隆过滤器与轻量级对称密码原语, 本文提出了一个称为 RFBC 的具有鲁棒性的前后向隐私 DSSE 方案. RFBC 能够在客户端发送不合理更新请求时进行有效处理, 保证鲁棒性, 并能够同时兼顾前向安全与后向安全, 本文的主要贡献如下.

(1) 提出了一个具有鲁棒性的前后向隐私安全的动态联合对称可搜索加密方案 RFBC. 该方案能够满足前向隐私, 并在兼顾效率与安全的情况下, 满足 Type-III 后向隐私.

(2) 本文所提出的方案支持多关键词联合查询, 同时, 本文所提出的方案具有鲁棒性, 即方案在保证前后向隐私的情况下, 能够处理客户端的一些不合理的操作请求, 如对已添加过的文件进行重复添加或删除不存在的文件等.

(3) 与其他满足鲁棒性的前后向安全的 DSSE 方案相比, 本文所提出的方案计算开销显著降低, 并且通信开销较其他方案也具有一定优势.

1.2 相关工作

可搜索加密有许多不同的分支, 主要包括静态对称可搜索加密、动态对称可搜索加密、前向隐私的对称可搜索加密、后向隐私的对称可搜索加密以及多功能的前后向隐私对称可搜索加密等[5−8].

Song 等人[1] 最早提出了 SSE 方案, 但该方案在执行搜索时因需逐个遍历文件而导致高昂的计算与通信开销. 为解决这一问题, Goh 等人[9]首次提出了基于正排索引的 SSE 方案, 该方案虽使每个文件与多个关键词相匹配, 却仍需遍历所有文件以完成搜索, 效率提升有限. 随后, Curtmola 等人[10]基于倒排索引构建了具有亚线性计算复杂度的 SSE 方案, 并明确界定了 SSE 的安全性标准. 自此, 倒排索引结构成为众多 SSE 方案的设计核心. Cash 等人[11]进一步拓展了 SSE 的功能边界, 提出了支持布尔查询的 SSE 方案. Lai 等人[12]利用 OXT 结构[11], 提出了一个可隐藏搜索结果的 SSE 方案.

静态的 SSE 方案既不支持加密数据集的更新, 也无法避免更新过程所造成的隐私泄露. 因此, 这些方案并不实用. 为了支持加密数据集的更新, 学者们提出了动态对称可搜索加密 DSSE, 它允许客户端在服务器上任意地添加或删除关键词对应的文件. Kamara 等人[2]提出了首个具有亚线性搜索效率的 DSSE 方案, 但该方案牺牲了安全性, 在更新阶段会泄露关键词信息. 后来 Kamara 等人[13]借助红黑树结构增强了方案的隐私保护能力. 2016 年, Zhang 等人[3]提出了文件注入攻击, 上述两个方案均不能抵抗该攻击. 该攻击方案通过利用过往的搜索记录伪造含有相同查询令牌的文件, 一旦客户端下载了这些伪造文件, 攻击者即可逆推出查询关键词, 乃至部分原始明文内容, 且仅需少量伪造文件即可高效实施该攻击. 针对上述威胁, 学者们提出了“前向隐私”概念, 旨在抵御上述侵犯隐私的攻击行为, 成为衡量 DSSE 方案安全性的重要维度.

Stefanov 等人[4]对前向隐私进行了形式化定义, 并提出了第 1 个前向隐私 DSSE 方案, 但该方案由于需要在更新阶段重新组织数据结构而导致较大的计算开销. Bost 等人[14]随后引入了陷门排列函数, 显著提高了前向隐私 DSSE 方案的搜索效率, 但该方案中基于公钥密码原语的陷门仍限制了其搜索效率.此后的研究中涌现了许多高效的前向隐私 DSSE 方案. Kim 等人[15]设计了基于双重字典的前向隐私 DSSE 方案并实现了真正的文件删除. Etemad 等人[16]则通过在每次搜索之后更新已暴露给服务器的密钥来保证前向隐私. 这些方案大多使用了公钥密码原语, 导致性能较低. 因此, Song 等人[17]在 Bost 等人[14]的方案基础上, 巧妙利用对称密码替换了原有的公钥密码陷门, 提出了 FAST 与 FASTIO 方案, 进一步优化了 DSSE 方案的更新与搜索流程. 值得注意的是, 多数前向隐私 DSSE 方案均假设服务器为诚实且好奇的参与者, Zhang 等人[18]则在考虑服务器恶意行为的情况下, 利用多集合哈希函数构造了一个可验证的前向隐私 DSSE 方案. Yang 等人[19]提出的多客户端访问控制 DSSE 方案, 融合了对称隐向量加密 (symmetric hidden vector encryption, SHVE) 与布隆过滤器技术, 同时利用 ODXT[20]来保证前向隐私. Yuan 等人[21]的方案则增加了对不合理更新请求的验证, 维护了搜索结果的准确性. Mei 等人[22]提出了一个针对多客户端的 DSSE 方案, 同时保证了客户端和服务器的前向隐私. Kamara 等人[23]基于 OXT 结构[11]提出了多关键词布尔查询方案, 虽然该方案能够保证前向隐私, 但由于其依赖模幂运算因而产生较大的计算开销. Wang 等人[24]在 Bost 等人[14]方案的基础上, 设计了联合搜索方案, 虽然该方案能够支持联合查询, 但其使用的公钥密码原语仍

限制了其计算效率. 上述 DSSE 方案虽然能实现前向隐私, 但并没有考虑后向隐私. 因此, 仍会造成查询隐私的泄露.

Stefanov 等人[4]最先引入了后向隐私的概念, 强调后续查询不能揭示先前已删除的文件信息, 但并未提供具体的方案. 其后, Bost 等人[25]根据不同的安全层次, 利用泄露函数详细界定了 3 种类型的后向隐私, 即 Type-I、Type-II 与 Type-III, 并以此构造相应的 DSSE 方案. 其中 Type-I 提供了最高等级的安全保障, 仅透露文件标识符、其插入时间和关键词更新频次; Type-II 相比 Type-I 安全性较低, 额外暴露了关键词的所有更新时间; 而 Type-III 在 Type-II 的基础上, 进一步泄露了每个删除标识符对应的添加标识符及其更新时间. 此后, 后向隐私的 DSSE 方案相继问世. Sun 等人[26]构建了一个满足 Type-III 后向隐私 DSSE 方案 Janus++, 该方案利用对称穿刺加密技术来提高通信效率, 但其计算开销与删除文件的数量成正比, 大量删除操作会造成开销陡增. Chamani 等人[27]根据上述 3 种不同的安全强度, 基于随机映射技术构造 Mitra、Orion 和 Horus 这 3 个方案. 其中 Mitra 满足 Type-II 后向隐私, 但在 “清除” (客户端在经过一次搜索后需要对未删除过的关键词与文件进行重新加密) 阶段存在巨大计算开销; Orion 满足最高安全等级的 Type-I 后向隐私, 但其搜索和更新效率较低; Horus 改善了 Orion 的通信效率, 但该方案仅满足 Type-III 后向隐私, 并且存在较大计算开销. Hoang 等人[28]构造了 DSSE 方案 IM-$DSSE_{II}$ 和 IM-$DSSE_{I+II}$, 两者均支持 Type-III 后向隐私, 但均存在较大的计算与通信开销. He 等人[29]与 Demertzis 等人[30]着眼于降低客户端的存储开销, 提出了具有常数级客户端存储开销的后向隐私 DSSE 方案, 但两者更新过程中的通信开销较大. Sun 等人[31]为了降低通信开销, 提出了一个满足前向隐私和 Type-II 后向隐私的 DSSE 方案 Aura, 仅需一次客户端-服务器交互即可完成搜索, 但撤销密钥计算成本偏高. 此外, 一些研究探索了可信执行环境在实现后向隐私中的应用[32−35]. Zuo 等人[36]利用同态加法和位图索引的对称加密技术构造了 Type-I 后向隐私 DSSE 方案. Wu 等人[37]利用 ORAM[38]与 LL-tree 构造了一个前后向隐私 DSSE 方案, 但受 ORAM 结构限制, 计算与通信开销较大. 现有 DSSE 方案大多依赖复杂的密码原语, 导致更新和搜索效率低下, 且局限于单关键词搜索, 缺乏联合搜索能力. 此外, 研究者们也提出了支持更多功能的前后向隐私方案. Li 等人[39]利用分区和隐藏指针技术, Xu 等人[40]利用可更新密钥的伪随机函数, 设计出支持更多功能的前后向隐私方案, 但分别面临计算开销与删除文件数量呈线性关系和单关键词搜索局限性的问题.

Patranabis 等人[20]于 2021 年提出 BDXT 和 ODXT, 均为满足前后向隐私的联合可搜索加密方案, 其中 ODXT 基于 OXT[11]的框架实现前后向隐私, OXT 架构包含“*TSet*” 与 “*XSet*” 两个加密数据库, 分别用于获取联合查询中最低频关键词对应的文件标识符以及检测文件是否包含联合查询中的剩余关键词. 但是, Zuo 等人[41]指出 OXT 结构在联合查询的前向隐私方面存在一定缺陷, 进而揭示了 ODXT[20]在特定情况下也不支持联合查询的前向隐私.

大多数可搜索加密方案并未实现真删除, 仅在搜索时忽略特定文件标识符件. Chen 等人[42]提出的 Bestie 方案打破了这一局限, 实现了真正意义上的删除操作, 但该方案仅满足 Type-III 后向隐私. Zuo 等人[43]基于二叉树结构设计了一个实现范围查询的前后向隐私 DSSE 方案. Wang 等人[44]通过结合几何前缀编码的倒排索引技术和加密位图技术, 提出了空间关键词查询的 DSSE 解决方案. Xu 等人[45]则利用局部差分隐私构造了一个能够抵抗泄露-滥用攻击的 DSSE 方案. Chamani 等人[46]聚焦于不同删除比率下高效搜索的问题, 并提出了 OSSE 和 LLSE 两个方案, 但这两个方案在更新文件时的多轮交互导致通信开销较大. Jiang 等人[47]利用变长的布隆过滤器、matryoshka 过滤器以及在线密码提出了一个结果模式隐藏的联合查询方案. Xu 等人[48]针对联合关键词搜索下的结果模式泄露, 提出了一个强隐私保护的联合关键词搜索方案 ESP-CKS. Jiang 等人[47]和 Xu 等人[48]分别通过创新的过滤器技术和强隐私保护模型, 解决了联合查询中的结果模式隐藏问题. 尽管上述研究在前后向隐私的基础上实现了诸多附加功能, 但普遍缺乏对客户端不合理更新请求的处理, 导致存储资源浪费的同时还影响了搜索效率. 其他方案则不支持联合搜索, 极大限制了搜索功能的灵活性和实用性.

本文第 2 节介绍方案所涉及的基础知识. 第 3 节针对 RFBC 进行了详细分析, 介绍方案 RFBC 的安全目标、主要思想、详细步骤以及安全性证明. 第 4 节分别从计算开销和通信开销两方面对方案 RFBC 的性能进行分析. 最后, 第 5 节总结全文.

2 基础知识

本节就本文提出的 RFBC 方案中所涉及的相关概念和基本知识进行介绍.

2.1 伪随机函数

定义 1. 函数 f: $\{0,1\}^k \times X \to Y$ 表示在安全参数 k 下集合 X 映射到集合 Y 的函数. 如果对于任意多项式时间敌手 A, 等式

$$ADF_{f,A}^{\text{prf}}(k) = |\Pr[A^{f_K(\cdot)} : K \leftarrow \{0,1\}^k] - \Pr[A^{s(\cdot)} : s \leftarrow F(x,y)]| \leqslant negl(k)$$

都不成立, 那么函数 f 是伪随机函数 (pseudo-random function, PRF), 其中 $ADF_{f,A}^{\text{prf}}(k)$ 表示在安全参数 k 下, 敌手 A 成功攻破函数 f 的概率, F 是随机函数, $negl(k)$ 表示敌手 A 能够区分函数 f 的值与随机函数 F 的值的概率是可忽略的.

2.2 布隆过滤器

布隆过滤器 (Bloom filter)[49]是一种空间和时间高效的数据结构, 常用于快速检测一个元素是否在某个集合中. 布隆过滤器由 m 比特的数组和 j 个独立的哈希函数构成, 在初始化阶段, 数组的每一位均置 0. 给定集合 $Y = \{b_1, b_2, \ldots, b_n\}$, 调用哈希函数 $H_0, H_1, \ldots, H_{j-1}$ 计算值 $H_i(b_k) \in [1, m]$, 其中 $i \in [0, j-1]$, $k \in [1, n]$, 并通过将数组中第 $H_i(b_k)$ ($i \in [0, j-1]$, $k \in [1, n]$) 位设置为 1 将集合 Y 插入到布隆过滤器中. 当要检测元素 p 是否在集合中, 调用 j 个哈希函数得到 $H_i(p)$, $i \in [0, j-1]$, 若数组中所有哈希值对应的位均为 1, 则元素 p 在集合 Y 中; 否则, 元素 p 不在集合 Y 中. 需注意的是, 布隆过滤器存在假阳性 (false positive), 即由于哈希函数的碰撞性以及位数组大小的限制, 布隆过滤器判定属于集合的元素可能不在集合中, 但其能准确判断某个元素一定不在集合中. 布隆过滤器的误判率 $y = (1 - e^{(-jn)/m})^j$ 与集合的大小 n、哈希函数个数 j 及为数组大小 m 有关, 故可通过选取合适的参数获得可接受的错误率.

2.3 对称加密

对称加密是一种加密和解密使用相同密钥的密码体制算法中, 常用的对称加密算法有 AES (advanced encryption standard) 等. 在同等安全级别下, 对称加密算法的计算速度快于非对称加密, 被广泛应用于可搜索加密中.

一个对称加密算法 SKE 主要包括密钥生成算法 Gen、加密算法 Enc 与解密算法 Dec, 即 $SKE = (Gen, Enc, Dec)$, 其中,

1) $SKE.Gen$: 给定安全参数 λ, 生成加密密钥 sk.

2) $SKE.Enc$: 在密钥 sk 的控制下对明文 m 进行加密, 生成密文 c, 即 $c = Enc_{sk}(m)$.

3) $SKE.Dec$: 在密钥 sk 的控制下对密文 c 进行解密, 恢复相应的明文 m, 即 $m = Dec_{sk}(c)$.

2.4 对称可搜索加密

本节主要介绍对称可搜索加密的形式化定义、安全性定义及前向隐私和后向隐私的形式化定义.

(1) 对称可搜索加密的形式化定义

在对称可搜索加密方案中, 客户端首先将包含关键词的文件加密后上传到服务器, 当客户端需要搜索文件时, 发送相应的关键词令牌给服务器, 服务器根据令牌返回对应的文件, 最后, 客户端解密返回的文件.

对称加密算法主要包括密钥生成算法 Gen、加密算法 Enc、陷门算法 $Trap$、搜索算法 $Search$ 与解密算法 Dec, 即 $SSE = (Gen, Enc, Trap, Search, Dec)$, SSE 过程如图 1 所示, 详细说明如下.

1) $K \leftarrow Gen(\lambda)$: 算法由客户端执行, 接收安全参数 λ, 该算法输出密钥 K.

2) $(I, c) \leftarrow Enc(K, F)$: 算法由客户端执行, 接收密钥 K 和包含关键词的文件集合 $F = \{F_1, F_2, \ldots, F_n\}$, 该算法输出一个加密索引 I 和加密文件集合 $c = \{c_1, c_2, \ldots, c_n\}$.

3) $tkn \leftarrow Trap(K, w)$: 算法由客户端执行, 接收密钥 K 和关键词 w, 该算法输出陷门 tkn.

4) $R \leftarrow Search(I, tkn)$: 算法由服务器执行, 接收一个文件集合 F、陷门 tkn 以及安全索引 I, 该算法输出文件标识符的集合 R.

5) $F_i \leftarrow Dec(K, c_i)$: 算法由客户端执行, 接收一个密钥 K 和密文 c_i, 该算法输出对应的文件明文 F_i.

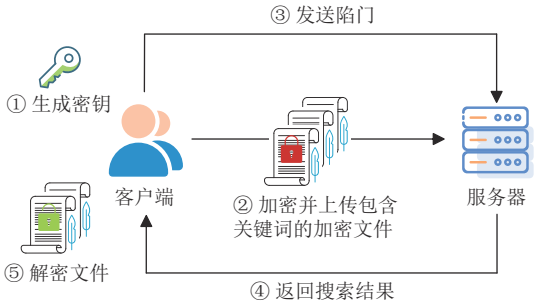

图 1 SSE 流程

(2) 安全性定义

在对称可搜索加密方案中, 通常利用理想世界与现实世界之间对抗的形式来描述方案的自适应安全[10]. 给一个对称可搜索加密方案 $SSE = (Gen, Enc, Trap, Search, Dec)$, A 和 S 分别表示敌手和模拟器, 泄露函数定义为 $L = \{L_{Enc}, L_{Search}, L_{Update}\}$. 根据现实世界 $Real_A^{SSE}(\lambda)$ 和理想世界 $Ideal_{A,S}^{SSE}(\lambda)$, 定义如下两个概率游戏.

现实世界 $Real_A^{SSE}(\lambda)$: 敌手 A 随机选择一个文件集合 DB, 游戏执行加密算法 $Enc(K, DB, W)$, 生成关键词索引 I 和文件密文集合 C, 再将其发送给敌手 A. 然后敌手 A 发出自适应查询 q, 若 q 是对关键词进行搜索操作, 那么该游戏执行搜索算法 $Search(tkn, st, I, C)$, 并将搜索结果返回给敌手 A. 若 q 是对关键词进行更新操作, 那么该游戏执行更新算法 $Update(K, w, ind, op, st, I, C)$, 并返回更新结果. 最后, 敌手 A 输出一个比特 $b \in \{0, 1\}$.

理想世界 $Ideal_{A,S}^{SSE}(\lambda)$: 敌手 A 随机选择一个文件集合 DB 并将其发送给模拟器 S. S 通过已知的泄露函数 L_{Enc} 生成对应关键词索引 I 和文件密文集合 C, 再将其发送给敌手 A. 然后敌手 A 根据收到的信息发出自适应查询 q, 如果 q 是对关键词进行搜索操作, 那么该游戏执行搜索算法 $S(L_{Search})$ 并将搜索结果返回给敌手 A. 如果 q 是对关键词进行更新操作, 那么该游戏执行更新算法 $S(L_{Update})$ 并返回更新结果. 最后, 敌手输出一个比特 $b \in \{0, 1\}$.

如果对任意概率多项式 (probabilistic polynomial-time, PPT) 敌手 A, 存在 PPT 模拟器 S 使得不等式

$$|\Pr[Real_A^{SSE}(\lambda)] - \Pr[Ideal_{A,S}^{SSE}(\lambda)]| \leqslant negl(\lambda)$$

成立, 那么我们说 SSE 方案具有自适应安全性.

(3) 前向隐私和后向隐私的形式化定义

在 DSSE 方案中, 前向隐私和后向隐私的主要目的是限制更新和搜索过程中的信息泄露. 其中, 前向隐私意味如下情况: 当客户端对某个关键词进行查询后, 又添加了一个包含该关键词的文档, 那么此时服务器不能根据已知信息推断出客户端对该关键词进行过查询, 即服务器无法将当前对某个关键词的更新与过去对该关键词的查询链接起来, 除非客户端发送最新的搜索令牌. 前向隐私的安全性定义[4]如下.

定义 2. 一个 DSSE 方案支持前向隐私, 当且仅当其更新泄露函数 L_{Updt} 能够满足:

$$L_{Updt}(op, w, ind) = L'(op, ind),$$

其中, L'表示无状态函数.

类似的, 后向隐私是指后续的查询不能链接到过去更新的文件. 下面对后向隐私的一些概念进行定义.

泄露函数 L 记录了过去所发送的一系列查询 Q, 对于一个搜索查询, Q 用 (tsp, w) 来表示, 其中 tsp 表示时间戳, 其初始化值为 0 并随着后续的查询不断增加; 对于一个更新查询, Q 用 $(tsp, op, (w, ind))$ 来表示. 我们首先考虑搜索情况的泄露函数 $sp(w)$, 其由每个搜索查询的时间戳组成, 且能表示同一个关键词 w 所对应的查询, 可将其定义为:

$$sp(w) = \{tsp | (tsp, w) \in Q\}.$$

函数 $TimeDB(w)$ 输出包含关键词 w 的最新文件及这些文件添加到数据库的时间戳 (不包含删除过的文件), 可以将其定义为:

$$TimeDB(w)=\{(tsp,ind)\mid (tsp,add,(tsp,ind))\in Q\wedge \forall tsp',(tsp',del,(w,ind))\notin Q\}.$$

函数 $DelHist(w)$ 输出关键词 w 的删除历史, 它包含与 w 相关的文件被添加到数据库的时间戳以及被删除的时间戳, 可将其定义为:

$$DelHist(w)=\{(tsp^{add},tsp^{del})\mid \exists ind:(tsp^{add},add,(w,ind))\in Q\wedge (tsp^{del},del,(w,ind))\in Q\}.$$

基于上述泄露函数, 后向隐私[25]可定义如下.

定义 3. 一个 DSSE 方案满足后向隐私安全, 当且仅当更新泄露函数 L_{Updt} 与搜索泄露函数 L_{Srch} 可写为如下 3 种形式, 其中, a_w 表示所有添加的文件, L''表示无状态函数:

Type-I: $L_{Updt}(op,w,ind)=L'(op)$, $L_{Srch}(w)=L''(TimeDB(w),a_w)$.

Type-II: $L_{Updt}(op,w,ind)=L'(op,w)$, $L_{Srch}(w)=L''(TimeDB(w),Updates(w))$.

Type-III: $L_{Updt}(op,w,ind)=L'(op,w)$, $L_{Srch}(w)=L''(TimeDB(w),DelHist(w))$.

2.5 可证明安全

可证明安全 (provable security)[50]是一种基于规约的安全性证明方法, 用来证明一个方案或协议可在某个敌手模型下实现特定的安全目标. 具体来讲, 该方法的证明流程为: 首先确定协议的安全目标, 其次为协议提供正式的安全性定义, 然后根据敌手的攻击能力构建敌手模型, 最后, 将攻击者成功攻破目标的过程规约为解决极微本原, 即某些数学难题. 在此过程中, 如果敌手能够攻破目标, 就说明方案中某些困难问题已被破解, 也就是说, 只要我们相信这些极微本原在一定时间内无法被敌手攻破, 那么该协议就是安全的.

在安全性证明的过程中, 协议的安全可以形式化为攻击者与挑战者之间的游戏过程. 若攻击者攻破目标的概率是忽略不计的, 则证明该协议是安全的. 该过程首先定义一个实际游戏, 然后根据极微本原对游戏进行一系列改动, 得到一个游戏序列, 推断出两个相邻游戏等价或者以忽略不计的概率近似, 并根据该游戏序列的最后一个游戏来得出方案是否安全的结论.

一般来说, 在可证明安全中, 随机预言模型是指理想的哈希函数, 即该函数满足一致性、可计算性以及均匀分布性. 在用于证明协议安全时, 随机预言模型可以被看作是一个随机函数. 在构造安全方案时, 系统的所有参与者在初始化过程中都共享该模型. 方案构造完成后, 设计者利用理想的哈希函数来代替随机预言模型. 在证明过程中, 协议的参与者都能够对模型进行访问, 若攻击者能够在模拟环境中以不可忽略不计的概率攻破目标, 则攻击者就能解决某些困难问题, 但是这就造成了与证明的前提假设相矛盾, 从而证明协议是安全的.

3 具有鲁棒性的前后向隐私联合查询 DSSE 方案: RFBC

基于布隆过滤器和轻量级对称密码, 本节提出一个具有鲁棒性的前后向隐私联合查询 DSSE 方案 RFBC, 下面我们详述 RFBC 的设计目标、方案细节及其安全性分析.

RFBC 的结构模型如图 2 所示, 客户端使用私钥对关键词和包含该关键词的文件加密, 当客户端对服务器发起包含多个关键词的联合查询时, 服务器会根据该联合查询返回对应的文件, 最后, 客户端再用自己的私钥对其文件进行解密.

3.1 RFBC 的安全目标

首先, 我们介绍具有鲁棒性的前后向隐私联合查询方案的安全性及功能性目标. 一个动态的联合对称可搜索加密方案具有如下性质: (1) 前向隐私: 无法将当前的更新与过去的查询链接起来, 换句话说, 当前的更新不会泄露过去的联合查询的隐私, 也就是服务器不知道当前添加的关键字之前是否搜索过; (2) 后向隐私: 后续的联合查询不能链接到过去删除的文件, 即当前的搜索不会泄露过去更新 (添加或删除) 的隐私, 也就是说服务器不知道更新 (添加或删除) 历史; (3) 鲁棒性: 在添加或删除关键词对应的文件时, 服务器会筛选出客户端的不合理更新请求, 保

证更新操作的合理性和搜索结果的正确性; (4) 联合搜索: 方案不仅支持单关键词的搜索, 还支持多关键词的联合搜索; (5) 高效搜索: 进行多关键词的联合查询时, 无需对所有关键词对应的文件进行搜索, 仅需对包含文件数最小的关键词进行搜索. 下面给出具有鲁棒性的联合对称可搜索加密方案形式化的安全性定义.

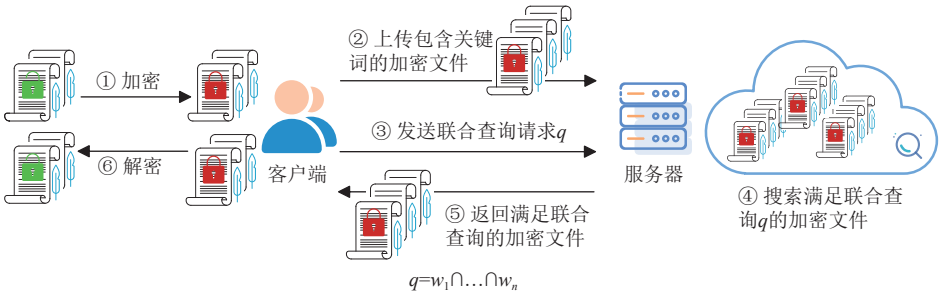

图 2 RFBC 的大致流程

假设 Q 是所有已发送的更新和搜索查询的列表, 更新查询用 $(v, op, (w, ind))$ 表示, 搜索查询用 (v, w) 表示, 其中 v 表示时间戳, w 表示所查询的关键词. 对于任意一个关键词 w, 函数 $TimeDB(w)$ 表示所有已添加但未从 EDB 中删除的关键字 w 的时间戳/文件标识符, $DelHist(w)$ 表示所有关键字 w 的插入/删除时间戳对, 下面对两个函数进行形式化定义:

$$TimeDB(w) = \{(v, ind) \mid (v, add, (w, ind)) \in Q \wedge \forall v', (v', del, (w, ind)) \notin Q\},$$

$$DelHist(w) = \{(v^{add}, v^{del}) \mid \exists ind : (v^{add}, add, (w, ind)) \in Q \wedge (v^{del}, del, (w, ind)) \in Q\}.$$

对于不合理的更新以及联合查询 $q = w_1 \wedge w_2 \wedge \ldots \wedge w_n$, 我们将上述函数 $TimeDB(w)$ 与 $DelHist(w)$ 拓展为函数 $cxTimeDB(w)$ 与 $cxDelHist(w)$, 定义分别如下:

$$cxTimeDB(q) = \{(\{v_i\}_{i\in[n]}, ind) \mid (v_i, add, (w_i, ind)) \in Q \wedge \forall v' > v_i, (v', del, (w_i, ind)) \notin Q\},$$

$$cxDelHist(q) = \{(\{v_i^{add}, v_i^{del}\}_{i\in[n]}) \mid \exists ind : (v_i^{add}, add, (w_i, ind)) \in Q \wedge (v_i^{del}, del, (w_i, ind)) \in Q\}.$$

对于任意关键词 w, 我们使用函数 $Upd(w)$ 表示所有更新操作的时间戳, 对其定义如下:

$$Upd(w) = \{v \mid \exists (op, ind) : (v, op, (w, ind)) \in Q\}.$$

对于任意一对关键词 (w_1, w_2), 函数 $Upd(w)$ 的定义如下:

$$Upd(w_1, w_2) = \{(v_1, v_2) \mid \exists (op, ind) : (v_1, op, (w_1, ind)) \in Q \wedge (v_2, op, (w_2, ind)) \in Q\}.$$

即函数 $Upd(w_1, w_2)$ 表示包含相同文件标识符的该对关键词 (w_1, w_2) 上的所有更新操作的时间戳. 当该查询为 $q = w_1 \wedge w_2 \wedge \ldots \wedge w_n$ 时, 假设关键词 w_1 对应的文件数量最少, 我们将更新函数进一步拓展并做如下定义:

$$Upd(q) = Upd(w_1) \vee \{Upd(w_1, w_i)\}_{i\in[2,n]}.$$

即函数 $Upd(q)$ 表示包含相同文件标识符的联合查询关键词 $(w_1, w_2, \ldots, w_n)$ 上的所有更新时间戳和关键词 w_1 的所有的更新时间戳.

定义 4. 一个自适应安全的联合 DSSE 方案满足前向隐私、Type-III 后向隐私以及鲁棒性, 当且仅当初始化函数 L_{Setup}、更新泄露函数 L_{Updt} 与搜索泄露函数 L_{Srch} 满足下列等式:

$$L_{Setup} = \perp,$$

$$L_{Updt}(w, op, ind) = L'(op),$$

$$L_{Srch}(q) = L''(cxTimeDB(q), cxDelHist(q), Upd(q)),$$

其中, L'与 L''都为无状态函数.

3.2 RFBC 的主要构造思想

为了实现方案的鲁棒性, 我们在服务器端为每个关键词设立了两个布隆过滤器, 分别用于追踪文件的添加操

作和文件的删除操作. RFBC 的更新操作流程如图 3 所示. 客户端进行更新操作时, 服务器端使用上述两个布隆过滤器实现方案的鲁棒性, 令计算值 h 包含关键词以及对应的文件标识信息, BF_1、BF_2 分别表示两个布隆过滤器. 更新主要包括两种操作: (1) 添加操作: 当客户端想要添加关键词 w 对应的文件 ind 时, 服务器端先判断 h 是否已插入到布隆过滤器 BF_1 中, 若 h 值已在 BF_1 中, 则表示该关键词对应的文件已被添加, 操作重复. 否则, 将 h 值插入到 BF_1 中, 服务器将信息存储在本地, 完成文件添加; (2) 删除操作: 当客户端想要删除关键词 w 对应的文件 ind 时, 服务器端先判断 h 是否已插入到布隆过滤器 BF_2 中, 若值 h 已在 BF_2 中, 则表示该关键词对应的文件已被删除, 操作重复. 若值 h 不在 BF_2 中, 服务器端还需要判断值 h 是否在 BF_1 中 (旨在防止删除实际上并不存在的文件), 若值 h 已在 BF_1 中, 则表示该关键词对应的文件存在, 客户端可以执行删除操作; 否则意味着该关键词对应的文件并不存在, 删除失败.

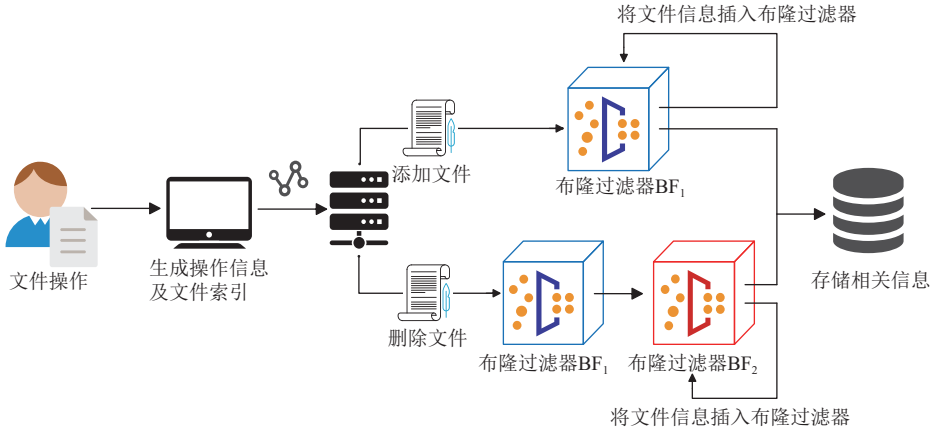

图 3 RFBC 更新操作流程

为了实现多关键词的联合查询, 我们为每个关键词都设置一个文件更新计数器, 当客户端对某个关键词成功执行添加操作后, 计数器的值加 1; 反之, 若成功删除与某关键词关联的文件, 计数器的值减 1. 当进行联合查询时, 客户端首先筛选出更新计数器值最小的关键词. 接下来, 搜索该关键词对应的文件. 在搜索过程中, 服务器根据从客户端接收到联合查询中所有关键词的令牌信息以及更新计数器值最小的关键词对应的最新状态, 检索该关键词所有添加过的文件, 并且利用布隆过滤器筛选掉已被联合查询中的关键词删除的加密文件标识符, 最后将筛选结果返回给客户端. 客户端收到这些加密文件标识符后执行解密操作, 得到搜索结果. 最后客户端还需进行一次 “清除” 操作, 即对更新计数器值最小的关键词, 清除掉已被删除的文件标识符, 对其进行一次整体更新, 这样就保证了前向隐私.

3.3 RFBC 的详细描述

本节详细介绍 RFBC 的具体算法, 其由初始化算法 *Setup*、搜索算法 *Search* 和更新算法 *Update* 组成, 用 $\Pi = (Setup, Search, Update)$ 表示. 下面分别介绍 RFBC 的 3 个子算法.

(1) 初始化算法 *Setup*

算法 1 详细描述了 RFBC 的初始化算法. 给定安全参数 λ, 客户端随机生成两个密钥 k_t 与 sk, 分别用来加密关键词生成关键词令牌 t_w 和加密文件索引. 然后, 初始化映射 *XSet* 和 *UpdCnt*, 其中, *XSet* 映射用于存储关键词 w 对应的最新文件索引密文和状态信息, *UpdCnt* 存储关键词 w 当前对应的文件数量. 服务器初始化算法初始化映射 *TSet* 和字典 *Dic*, 它们分别用来存储加密数据库和关键词 w 对应的两个布隆过滤器.

算法 1. RFBC.*Setup*.

输入: 安全参数 λ;

输出: $\perp$.

Client:

1. $k_t, sk, d \xleftarrow{\$} \{0,1\}^{\lambda}$
2. $XSet, UpdCnt \leftarrow$ Empty map

Server:

3. $TSet \leftarrow$ Empty map
4. $Dic \leftarrow$ Empty dictionary

(2) 更新算法 *Update*

算法 2 详细描述了 RFBC 方案针对关键词 w 的更新操作, 包括添加和删除 w 对应的文件.

算法 2. RFBC.*Update*.

输入: 密钥 k_t, sk, 关键词 w, 集合 *XSet*, 文件 *ind* 及对应操作 *op*, 集合 *TSet*;

输出: $\perp$.

Client:

1. $t_w \leftarrow F_1(k_t, w)$
2. $c_{i+1} = F_2(sk, ind)$
3. $h = H_1(t_w \| c_{i+1})$
4. $td = H_4(t_w, d)$
5. $(st_i, c_i) \leftarrow XSet[w]$
6. **IF** $op = add$ **THEN**
7. **IF** (st_i, c_i) = NULL **THEN**
8. $st_0 \xleftarrow{\$} \{0,1\}^{\lambda}$
9. $u \leftarrow H_2(t_w \| st_0)$
10. $e \leftarrow H_3(t_w \| st_0) \ominus (c_{i+1} \| \perp)$
11. **ELSE**
12. $st_{i+1} \xleftarrow{\$} \{0,1\}^{\lambda}$
13. $u \leftarrow H_2(t_w \| st_{i+1})$
14. $e \leftarrow H_3(t_w \| st_{i+1}) \ominus (c_{i+1} \| st_i)$
15. **END IF**
16. **SEND** (u, e, h, td) **TO Server**
17. **ELSE**
18. **SEND** $(\perp, \perp, h, td)$ **TO Server**
19. **END IF**

Server:

20. $(BF_1, BF_2) \leftarrow Dic[td]$
21. **IF** $Dic[td]$ = NULL **THEN**
22. Initialize two empty bloom filter BF_1 and BF_2 for td
23. $Dic[td] \leftarrow (BF_1, BF_2)$24. **IF** $u \neq \perp$ && $e \neq \perp$ **THEN**
25. **IF** $BF_1[h] \neq 1$ **THEN**
26. set $BF_1[h] = 1$
27. set $b = 1$

28. $TSet[u] = e$
29. **ELSE**
30. set $b = 0$
31. **END IF**
32. **ELSE**
33. **IF** $BF_2[h] \neq 1$ && $BF_1[h] = 1$ **THEN**
34. set $BF_2[h] = 1$
35. set $b = 1$
36. **ELSE**
37. set $b = 0$
38. **END IF**
39. **END IF**
40. **SEND** b **TO Client**

Client:

41. **IF** $op = add$ && $b = 1$ **THEN**
42. $UpdCnt[w] \leftarrow UpdCnt[w]+1$
43. $XSet[w] \leftarrow (st_{i+1}, c_{i+1})$
44. **IF** $op = del$ && $b = 1$ **THEN**
45. $UpdCnt[w] \leftarrow UpdCnt[w]-1$
46. **END IF**

当对 w 对应的文件 *ind* 进行操作时, 客户端首先利用不同的密钥计算出关键词令牌 t_w、文件索引密文 c_{i+1} 以及包含令牌和文件索引密文的哈希值 h, 客户端检查关键词 w 是否针对此文件执行过更新. 当添加包含关键词 w 的文件时, 客户端首先判断此次操作是否是首次为关键词 w 添加文件. 若从未添加过关键词为 w 的文件, 则客户端会随机初始化 w 的状态信息 td, 并以此计算 u 和 e, 需要注意的是, 若添加的文件为第 1 个文件, 值 e 包含一个状态终止标志; 若关键词 w 已添加过文件, 则进行正常计算, 此时值 e 包含了关键词 w 的状态信息. 之后, 客户端将 (u, e, h, td) 发送到服务器 (删除时仅发送 (h, td)).

对于服务器, 当所执行的操作为添加关键词 w 对应的文件时, 首先获取该关键词对应的添加布隆过滤器和删除布隆过滤器, 若获取失败 (表示该关键词为初次更新), 则创建两个空布隆过滤器 BF_1 和 BF_2, 分别表示该关键词对应的添加布隆过滤器和删除布隆过滤器. 接下来判断该文件是否已存在, 若不存在, 则将包含关键词令牌和文件索引密文的哈希值 h 插入到布隆过滤器 BF_1 中, 并保存该文件, 将 b 置为 1, 添加成功; 否则文件已存在, 将值 b 设为 0. 与添加类似, 当所执行的操作为删除关键词 w 对应的文件时, 服务器首先判断该文件是否已被添加且是否被删除, 若该文件已被添加且未被删除, 服务器将包含关键词令牌和文件索引密文的哈希值 h 插入到布隆过滤器 BF_2 中, 将 b 的值置为 1, 删除成功; 否则, 文件曾被添加但被删除或未被添加, 将 b 的值设为 0, 删除失败. 最后, 服务器将值 b 发送给客户端, 表示服务器端的操作结果.

客户端接收到服务器发来的 b, 若 $b = 1$ 且对该文件的操作为添加, 表示添加成功, 客户端将更新计数器 *UpdCnt* 的值加 1, 并更新映射 *XSet*, 若 $b = 1$ 且对该文件的操作为删除, 表示删除成功, 客户端将更新计数器 *UpdCnt* 的值减 1; 否则, 表示客户端对该文件的操作不合理.

(3) 搜索算法 *Search*

算法 3 详细描述了 RFBC 方案的搜索算法. 当进行联合关键词查询时, 客户端首先确定文件更新计数器 *UpdCnt* 中值最小的关键词 (假设是 w_1), 然后根据映射 *XSet* 得到关键词 w_1 最新的状态信息, 接着计算所有查询关键词的令牌以及每个关键词对应的布隆过滤器地址, 并将 w_1 最新的状态信息、令牌列表 *TokenList* 以及布隆过滤器的地

址列表 *TdList* 发送给服务器.

算法 3. RFBC.*Search*.

输入: 密钥 k_t, *sk*, 联合查询 *q*, 集合 *XSet* 及其对应的操作 *op*;
输出: 目标文件集合 *ID*.

Client:
1. $q = w_1 \wedge w_2 \wedge \ldots \wedge w_n$, *TokenList*, *TdList* ← Empty list
2. 确定当前文件最少的关键词 (假设是 w_1)
3. $(st_i, c_i) \leftarrow XSet[w_1]$
4. $j \leftarrow 1$
5. **REPEAT**
6. $t_{w_j} \leftarrow F_1(k_t, w_j)$
7. $TokenList \leftarrow TokenList \vee t_{w_j}$
8. $j \leftarrow j+1$
9. $td_{w_j} \leftarrow H_4(t_{w_j}, d)$
10. $TdList \leftarrow TdList \vee td_{w_j}$
11. **UNTIL** $(j = n)$
12. **SEND** *TokenList*, st_i, *TdList* **TO** Server
Server:
13. $R, I \leftarrow \varnothing$
14. **FOR** $j = 1$ **to** *TdList.size*
15. $(BF_{w_j,1}, BF_{w_j,2}) \leftarrow Dic[td_{w_j}]$
16. **END FOR**
17. **REPEAT**
18. $u \leftarrow H_2(t_{w_1} \| st_i)$
19. $e \leftarrow TSet[u]$
20. $(c_i, st_{w_{1,i-1}}) \leftarrow e \oplus H_3(t_{w_1} \| st_i)$
21. $h_{w_1} \leftarrow H_1(t_{w_1} \| c_i)$
22. **IF** $BF_{w_1,2}[h_{w_1}] \neq 1$ **THEN**
23. $I \leftarrow I \vee c_i$
24. **FOR** $j = 2$ **to** *TokenList.size*
25. **IF** $BF_{w_j,1}[h_{w_j}] = 1$ **&&** $BF_{w_j,2}[h_{w_j}] \neq 1$ **THEN**
26. $R \leftarrow R \vee c_i$
27. **END IF**
28. **END FOR**
29. **END IF**
30. **UNTIL** $(st_{w_{1,i}} = \perp)$
31. **SEND** *R*, *I* **TO** Client
Client:
32. *ID* ← ∅
33. $ID \leftarrow Dec(sk, R)$

34. **RETURN** *ID*

35. Client use *I* to CLENA-UP file identifiers for w_1

服务器接收到这些信息后, 根据布隆过滤器的地址列表取出每个关键词对应的添加布隆过滤器和删除布隆过滤器, 然后根据令牌列表中的首个元素, 也就是关键词 w_1 的令牌, 结合状态信息检索得到该关键词最新添加的文件对应的加密文件标识符以及前一个状态. 通过二者可检索到 w_1 过去添加的所有加密文件标识符, 每检索到一个加密文件标识符都对其进行如下判断: 若文件标识符不在关键词 w_1 的删除布隆过滤器中, 服务器则会将加密文件索引 c_i 添加到索引集合 I 中, 进一步判断加密文件索引 c_i 是否包含其他关键词, 若 c_i 并未被其他关键词的删除操作所删除, 则将 c_i 添加到恢复集合 R 中. 服务器再将集合 R 与集合 I 发送给客户端, 客户端收到这些信息后利用私钥解密得到联合查询的结果. 这里, 我们对为何要选出更新计数器 *UpdCnt* 中值最小的关键词做出说明. 主要原因是该策略能够减少查询过程所需的时间, 考虑如下情况, 假设关键词对应的更新计数器为 n_c, 联合查询的关键词个数为 n_w, 根据上述查询步骤可知, 服务器端对这 n_c 个文件, 进行 n_w 次布隆过滤器比较操作 (这里我们对两个布隆过滤器的操作视为一次操作), 故算法时间复杂度为 $O(n_c \times n_w)$. 因此, 当 n_w 固定时, 选择最小的 n_c 会最小化服务器端操作. 同时, 由于在查询过程的前半段, 即客户端侧, 所有操作的执行次数同样与 n_c 有关, 同理, 选择最小的 n_c 会最小化客户端的操作.

在进行一次搜索之后, 客户端需使用集合 I 对 w_1 的文件进行 “清除” 操作, 即服务器将 w_1 过去存储的文件全部删除, 并对未删除过的文件进行重新添加并初始化其对应的布隆过滤器.

3.4 安全性分析

下面我们证明本文方案 RFBC 满足前向隐私、Type-III 后向隐私以及鲁棒性.

在 RFBC 的更新中, 每次合理地添加关键词文件对信息都会生成一个随机状态, 因此服务器无法预测下一个状态信息, 每次合理地删除关键词文件对时都是通过哈希值进行操作, 因此服务器也无法得到任何关键词文件对的明文信息. 同时, 当更新不合理时, RFBC 会过滤掉这些不合理的请求. 从而, RFBC 保证了前向隐私与鲁棒性. 经过一次联合搜索后, RFBC 将包含更新计数器值最小的关键词的对应文件进行 “删除”, 即使服务器保存了过去的搜索令牌以及其他相关信息也查询不到过去删除过的文件. 因此, RFBC 保证了后向隐私. 下面详细地对方案 RFBC 的安全性进行形式化证明.

定理 1. 定义本文方案 $L^{\mathrm{RFBC}} = (L^{\mathrm{RFBC}}_{Setup} = \perp, L^{\mathrm{RFBC}}_{Updt}, L^{\mathrm{RFBC}}_{Srch})$, 其中,

$$L^{\mathrm{RFBC}}_{Setup} = \perp,$$

$$L^{\mathrm{RFBC}}_{Updt}(w, op, ind) = L'(op),$$

$$L^{\mathrm{RFBC}}_{Srch}(q) = L''(cxTimeDB(q), cxDelHist(q), Upd(q)),$$

那么 RFBC 是一个支持前向隐私、Type-III 后向隐私以及鲁棒性的自适应安全联合对称可搜索加密方案.

证明: 我们将通过下面的一系列游戏来证明定理 1.

游戏 G_0: 该游戏表示真实的安全游戏. 因此, 该游戏满足下列等式:

$$\Pr[Real^{\Pi}_{A}(\lambda) = 1] = \Pr[G_0 = 1].$$

游戏 G_1: 除使用随机字符串代替伪随机函数 F_1 与 F_2 外, 该游戏与 G_0 相同. 在该游戏中, 映射 **Tw** 记录了关键词及对应的令牌信息, 即 (w, t_w), 映射 **C** 记录了文件标识符及对应的文件标识符密文, 即 (ind, c). 当进行联合查询时, 该游戏判断映射 **Tw** 是否包含该联合查询的所有关键词, 若包含这些关键词, 则游戏返回对应的令牌; 否则, 随机选择一个令牌来代替 t_w, 并将元组 (w, t_w) 存储到映射 **Tw** 中. 对于联合搜索的结果, 映射 **C** 与映射 **Tw** 的操作类似. 显然, 敌手 β 无法以不可忽略不计的概率 λ 区分伪随机函数与随机函数生成的随机字符串, 因此游戏 G_1 与 G_0 是不可区分的, 因此该游戏满足:

$$\Pr[G_0 = 1] - \Pr[G_1 = 1] \leqslant Adv^{\mathrm{prf}}_{F_1, F_1, \beta}(\lambda).$$

游戏 G_2: 在该游戏的更新协议中, 我们使用随机预言机 $\mathbf{RO}_1$ 生成同样长度的字符串来代替哈希函数 H_1 的输

出, 并将该信息存储在映射 **A** 中, 算法 4 对游戏 G_2 进行了详细描述. 在游戏 G_2 中, 该字符串的生成与游戏 G_1 是相同的, 因此, 游戏 G_1 与 G_2 是不可区分的, 从而游戏 G_2 满足下列等式:

$$\Pr[G_1=1]-\Pr[G_2=1]=0.$$

游戏 G_3: 除了使用哈希函数 H_2, 该游戏与 G_2 没有区别, 推理过程与游戏 G_2 类似. 因此该游戏与 G_2 是不可区分的. 从而该游戏满足下列等式:

$$\Pr[G_2=1]-\Pr[G_3=1]=0.$$

游戏 G_4: 同理, 除了使用哈希函数 H_3, 该游戏与 G_3 是相同的, 因此该游戏与 G_3 是不可区分的. 从而该游戏满足下列等式:

$$\Pr[G_3=1]-\Pr[G_4=1]=0.$$

算法 4. 游戏 G_2.

输入: 密钥 k_t, sk, 联合查询 q, 集合 $XSet$ 及其对应的操作 op;
输出: $\perp$.

Setup(λ, $\perp$, $\perp$)
Client:
1. $XSet$, **A**, $TSet$, $UpdCnt$ ← Empty map
Update(k_t, sk, w, $XSet$, op, ind; $TSet$)
Client:
2. t_w ← **Tw**[w]
3. c_{i+1} ← **C**[ind]
4. $td = H_4(t_w, d)$
5. **A**[$t_w||c_{i+1}$] ←$\{0,1\}^s$
6. (st_i, c_i) ← $XSet$[w]
7. **IF** op = add **THEN**
8. **IF** (st_i, c_i) = NULL **THEN**
9. $st_0 \xleftarrow{\$} \{0,1\}^\lambda$
10. $u \leftarrow H_2(t_w||st_0)$
11. $e \leftarrow H_3(t_w||st_0) \oplus (c_{i+1}||\perp)$
12. **ELSE**
13. $st_{i+1} \xleftarrow{\$} \{0,1\}^\lambda$
14. $u \leftarrow H_2(t_w||st_{i+1})$
15. $e \leftarrow H_3(t_w||st_{i+1}) \oplus (c_{i+1}||st_i)$
16. **END IF**
17. **SEND** (u, e, **A**[$t_w||c_{i+1}$], td) **TO** Server
18. **ELSE**
19. **SEND** (**A**[$t_w||c_{i+1}$], td) **TO** Server
20. **END IF**
Server:
21. (BF_1, BF_2) ← Dic[td]
22. **IF** Dic[td] = NULL **THEN**
23. Initialize two empty bloom filter BF_1 and BF_2 for td

24. $Dic[td] \leftarrow (BF_1, BF_2)$
25. **IF** $op = add$ **THEN**
26. **IF** $BF_1[\mathbf{A}[t_w||c_{i+1}]] \neq 1$ **THEN**
27. set $BF_1[\mathbf{A}[t_w||c_{i+1}]] = 1$
28. set $b = 1$
29. $TSet[u] = e$
30. **ELSE**
31. set $b = 0$
32. **END IF**
33. **ELSE**
34. **IF** $BF_2[\mathbf{A}[t_w||c_{i+1}]] \neq 1$ && $BF_1[\mathbf{A}[t_w||c_{i+1}]] = 1$ **THEN**
35. set $BF_2[\mathbf{A}[t_w||c_{i+1}]] = 1$
36. set $b = 1$
37. **ELSE**
38. set $b = 0$
39. **END IF**
40. **END IF**
41. **SEND** b **TO** Client

Client:

42. **IF** $op = add$ && $b = 1$ **THEN**
43. $UpdCnt[w] \leftarrow UpdCnt[w]+1$
44. $XSet[w] \leftarrow (st_{i+1}, c_{i+1})$
45. **IF** $op = del$ && $b = 1$ **THEN**
46. $UpdCnt[w] \leftarrow UpdCnt[w]-1$
47. **END IF**

ServerSearch$(k_t, sk, q, XSet; TSet)$

Client:

48. $q = w_1 \wedge w_2 \wedge \ldots \wedge w_n$, *TokenList*, *TdList* ← Empty list
49. 确定当前文件最少的关键词 (假设是 w_1)
50. $(st_i, c_i) \leftarrow XSet[w_1]$
51. $j \leftarrow 1$
52. **REPEAT**
53. $t_{w_j} \leftarrow \mathbf{Tw}(w_j)$
54. $j \leftarrow j+1$
55. $TokenList \leftarrow TokenList \vee t_{wj}$
56. $td_{w_j} \leftarrow H_4(t_{w_j}, d)$
57. $TdList \leftarrow TdList \vee td_{wj}$
58. **UNTIL** $(j = n)$
59. **SEND** $(TokenList, st_i)$ **TO** Server

Server:

60. **FOR** $j = 1$ **to** *TdList.size*

61. $(BF_{w_j,1}, BF_{w_j,2}) \leftarrow Dic[td_{w_j}]$
62. **END FOR**
63. **REPEAT**
64. $u \leftarrow H_2(t_{w_1} \| st_i)$
65. $e \leftarrow TSet[u]$
66. $(c_i, st_{w_{1,i-1}}) \leftarrow e \oplus H_3(t_{w_1} \| st_i)$
67. $h_{w_1} \leftarrow \mathbf{A}(t_{w_1} \| c_i)$
68. **IF** $BF_{w_1,2}[h_{w_1}] \neq 1$ **THEN**
69. $R \leftarrow R \vee c_i$
70. **FOR** $j = 2$ **to** *TokenList.size*
71. $h_{w_j} \leftarrow \mathbf{A}[t_{w_j} \| c_i]$
72. **IF** $BF_{w_j,1}[h_{w_j}] = 1$ **&&** $BF_{w_j,2}[h_{w_j}] \neq 1$ **THEN**
73. $I \leftarrow I \vee c_i$
74. **END IF**
75. **END FOR**
76. **END IF**
77. **UNTIL** $(st_{w_{1,i}} = \perp)$
78. **RETURN** R, I **TO** Client

游戏 G_5: 同理可得, 除了使用哈希函数 H_4, 该游戏与 G_4 是相同的, 因此该游戏与 G_4 是不可区分的. 从而, 该游戏满足下列等式:

$$\Pr[G_4 = 1] - \Pr[G_5 = 1] = 0.$$

游戏 G_6: 在游戏 G_5 中, 每个关键词对应的 BF_1 与 BF_2 的值都是通过布隆过滤器、令牌以及加密文件标识符生成的. 在游戏 G_6 中, 该游戏通过随机预言机生成字符串来代替 BF_1 与 BF_2, 并分别存储在映射 BF_{add} 与 BF_{del} 中. 在进行更新操作时, 从敌手的视角看, 游戏 G_5 与 G_6 都是随机生成字符串. 因此, 游戏 G_5 与 G_6 是不可区分的. 从而, 该游戏满足:

$$\Pr[G_5 = 1] - \Pr[G_6 = 1] = 0.$$

模拟器 S: 算法 5 详细描述了模拟器游戏 S 的算法. 在更新与搜索算法中, 该游戏使用随机预言模型并保存时间戳与关键词令牌/文件标识符密文的映射关系. 显然, 模拟器 S 与 G_6 产生的信息分布相同, 因为该游戏使用了泄露函数 L^{RFBC} 来生成数据. 从而模拟器 S 满足:

$$\Pr[G_6 = 1] - \Pr[Ideal^{\mathrm{RFBC}}_{A,S,L^{\mathrm{RFBC}}}(\lambda) = 1] = 0.$$

结论: 通过组合上述所有游戏得到的结论, 存在敌手 β, 满足:

$$\Pr[Real^{\Pi}_{\mathrm{A}}(\lambda) = 1] - \Pr[Ideal^{\mathrm{RFBC}}_{\mathrm{A,\,S,\,L^{RFBC}}}(\lambda) = 1] \leqslant Adv^{\mathrm{prf}}_{F_1,F_2,\beta}(\lambda).$$

即本文所提出的方案 RFBC 满足前向隐私、Type-III 后向隐私以及鲁棒性.

算法 5. 模拟器 Simulator *S*.

Setup()

1. $v \leftarrow 0$
2. *XSet*, *UpdCnt*, *TSet*, **Tw** $\leftarrow$ Empty map
3. **A**, **B**, **C**, **P**, **ST** $\leftarrow$ Empty map
4. *Dic* $\leftarrow$ Empty dictionary

Update(*w*, *op*, *ind*)

Client:

5. $v \leftarrow v+1$

6. $\mathbf{C}[ind] \xleftarrow{\$} \{0,1\}^{\lambda}$

7. $\mathbf{A}[v] \xleftarrow{\$} \{0,1\}^{s}$

8. **IF** *op* = *add* **THEN**

9. $\quad$ $\mathbf{ST}[w], \mathbf{B}[v], \mathbf{P}[v], \mathbf{E}[v] \leftarrow \{0,1\}^{\lambda}, \{0,1\}^{p}, \{0,1\}^{z}, \{0,1\}^{\lambda+t}$

10. $\quad$ **SEND** ($\mathbf{A}[v], \mathbf{B}[v], \mathbf{E}[v], \mathbf{P}[v]$) **TO** Server

11. **ELSE**

12. $\quad$ **SEND** ($\mathbf{A}[v], \mathbf{P}[v]$) **TO** Server

13. **END IF**

Server:

14. $(\mathrm{BF}_{add}, \mathrm{BF}_{del}) \leftarrow Dic[\mathbf{P}[v]]$

15. **IF** *op* = *add* **then**

16. $\quad$ **IF** $\mathrm{BF}_{add}[\mathbf{A}[v]] \neq 1$ **THEN**

17. $\quad\quad$ $\mathrm{BF}_{add}[\mathbf{A}[v]] = 1$

18. $\quad$ **END IF**

19. **ELSE**

20. $\quad$ **IF** $\mathrm{BF}_{del}[\mathbf{A}[v]] \neq 1$ **&&** $\mathrm{BF}_{add}[\mathbf{A}[v]] = 1$ **THEN**

21. $\quad\quad$ $\mathrm{BF}_{del}[\mathbf{A}[v]] = 1$

22. $\quad$ **END IF**

23. **END IF**

Search(*Upd*(*q*), *cxTimeDB*(*q*), *cxDelHist*(*q*))

Client:

24. $\bar{w} \leftarrow \min(UpdCnt(q))$

25. $t_w \leftarrow \mathbf{Tw}[\bar{w}]$

26. $(st_i, c_i) \leftarrow XSet[\bar{w}]$

27. $(v_0, v_1, \ldots, v_i) \leftarrow Upd[q]$

28. **FOR** $(v_j, ind) \in \{cxTimeDB(q), cxDelHist(q)\}$ **DO**

29. $\quad$ program H_1: $H_1(t_w \| c_j) \leftarrow \mathbf{A}[v_j]$

30. $\quad$ program H_2: $H_2(t_w \| \mathbf{ST}[v_j]) \leftarrow \mathbf{B}[v_j]$

31. $\quad$ program H_3: $H_3(t_w \| st_j) \leftarrow \mathbf{E}[v_j] \oplus (\mathbf{C}[v_j] \| \mathbf{ST}[v_j])$

32. $\quad$ program H_4: $H_4(t_w \| d') \leftarrow \mathbf{P}[v_j]$

33. **END FOR**

34. $XSet[\bar{w}] \leftarrow (\mathbf{ST}[v_i], \mathbf{C}[v_i])$

35. **SEND** ($\mathbf{Tw}$, $\mathbf{P}$, $\mathbf{ST}[v_i]$) **TO** Server

4 分析与评估

本节对 RFBC 进行分析和评估, 并同其他相关方案 ROSE[40], 联合对称可搜索加密方案 ODXT[20]及 BDXT[20]进行比较. 本文所有实验均运行于 64 位的 Windows 10 操作系统的电脑 (16 GB 内存, CPU 为 Intel(R) Core(TM) i7-

12700F), 程序运行环境为 Python 3.9. 实验中伪随机函数 F_1 与 F_2 均采用 AES-128, 哈希函数 H_1–H_4 均采用 SHA-256. 为了实现效率与存储的平衡, 布隆过滤器中哈希函数的个数设置为 12, 错误率设为 10^{-4}. 为了评估更新与搜索效率, 我们构造了 2×10^5–2×10^6 个关键词文件数据对, 并且设置不同比例的不合理更新来检测不合理更新次数对搜索效率的影响. 此外, 所有实验结果均取自 10 次实验的去尾平均值.

4.1 安全性与功能比较

方案 RFBC 与 ROSE[40]、ODXT[20]及 BDXT[20]的安全性和功能比较如表 1 所示. 由表 1 可知, ROSE[40]虽然具有鲁棒性且能实现前后向隐私, 但该方案不支持多关键词联合搜索; ODXT[20]与 BDXT[20]虽然在满足前后向隐私的同时实现了联合搜索的功能, 但都无法保证更新时的鲁棒性; 而本文所提出的方案 RFBC 能够同时保证前向隐私、后向隐私及鲁棒性, 并且支持联合搜索.

表 1 功能比较

功能	RFBC	ROSE[40]	ODXT[20]	BDXT[20]
前向隐私	√	√	√	√
后向隐私	√	√	√	√
鲁棒性	√	√	×	×
联合搜索	√	×	√	√

4.2 计算开销比较

在本节中, 我们通过分析更新时间和搜索时间来衡量 RFBC 及相关方案的计算效率. 我们设置了不同比例的不合理更新来评估其数量对整体效率的影响. 值得注意的是, 由于方案 ROSE 不具有多关键词联合搜索的能力, 因此我们选择具备联合搜索功能的方案 BDXT 和 ODXT 进行比较, 以全方位评估方案的性能.

(1) 合理更新下的更新时间

图 4 展示了 3 个方案在合理更新下的更新时间. 由图 4 可知, 3 个方案的更新时间均随关键词文件对的增加而增加, 但方案 RFBC 的计算效率优于 BDXT 和 ODXT. 特别的, 相对于 ODXT, RFBC 的计算效率提升了约 3 倍. 这是由于在添加关键词对应的文件时, 方案 ODXT 需要 6 次 AES 运算, 同时为了实现联合查询, 该方案还需要进行模幂运算, 显著增加了计算开销; 方案 BDXT 的更新计算开销虽较 ODXT 显著降低, 但仍略微高于 RFBC, 该方案主要在添加文件时进行 3 次 AES 运算. 而本文所提出的方案 RFBC 在执行相同操作时, 核心的计算操作仅为包括 2 次 AES 运算及哈希函数运算, 计算效率明显优于 ODXT 和 BDXT.

(2) 合理更新下的搜索时间

图 5 描述了在合理更新的情况下, 不同方案的联合搜索 $q = w_1 \wedge w_2$ 的效率, 这里我们假设关键词 w_1 的更新计数器值最小, 从图 5 中可以看出, RFBC 的搜索时间明显少于 BDXT 和 ODXT.

在进行联合搜索时, BDXT 方案要求客户端与服务器之间进行两次交互, 这不仅增加了通信开销, 也带来了较大的计算开销. 具体来讲, 在第 1 轮通信中, 客户端会将关键词 w_1 对应的文件地址信息发送给服务器, 然后服务器反馈所有含有关键词 w_1 的文件 (包含已被删除的文件); 在第 2 轮通信中, 客户端基于服务器的反馈, 筛选出未被删除的含关键词 w_1 的文件, 然后将其他关键词的文件标签发送给服务器供其在布隆过滤器中对这些标签进行检测, 服务器检测后将结果返回给客户端, 客户端据此剔除非目标文件. 上述过程中第 1 轮通信的复杂度为 $O(a_w+d_w)$, 其中 a_w 表示更新计数器值最小的关键词 w_1 添加的文件数量, d_w 表示关键词 w_1 删除的文件数量. 第 2 轮通信的复杂度为 $O(q)$, q 表示联合查询中关键词的个数.

相比之下, ODXT 虽降低了通信开销, 但计算开销显著增加. 与 BDXT 相同, ODXT 在进行联合搜索时, 客户端首先计算 w_1 对应的文件地址信息及其他关键词的令牌信息, 随后传输给服务器. 服务器基于这些信息计算各关键词令牌的标签, 以判断 w_1 对应的文件是否匹配其他关键词, 并统计匹配计数连同加密文件信息发送给客户端. 客户端接收到回传数据后, 通过计数及文件操作记录剔除掉已删除的文件, 完成搜索. 值得注意的是, RFBC 和

ODXT 的联合搜索的计算复杂度均为 $O(a_w+d_w+q)$, 但 ODXT 在计算令牌信息及标签信息时引入了模幂运算, 这带来了较大的计算开销.

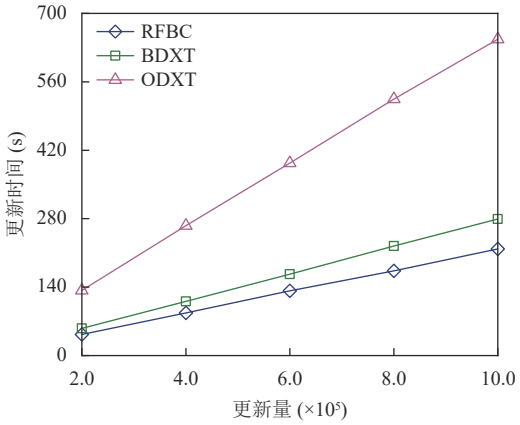

图 4 合理更新下更新时间开销

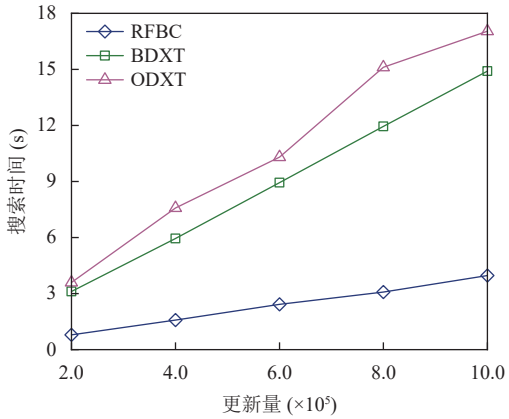

图 5 合理更新下搜索时间开销

(3) 不合理更新下的搜索时间

RFBC 在进行联合搜索操作时优化了这一流程, 无需模幂运算和多轮通信, 客户端只需将关键词 w_1 的最新状态及其他关键词令牌发送给服务器, 服务器检索出 w_1 对应的文件后, 利用布隆过滤器排除没有包含所有关键词的文件以及被删除的文件, 仅返回过滤后的加密文件索引, 最后客户端使用私钥解密以获得搜索结果. 因此, 在合理更新前提下, RFBC 的搜索效率高于 BDXT 和 ODXT.

图 6 展示了不同比例的不合理更新对联合搜索效率的影响. 图中, *iup* 代表不合理更新占关键词文件数据对总数的比例, 该比例从 0.1 开始, 以 0.1 为间隔递增至 0.4. 由图 6 可知, 随着不合理更新比例的提高, 在相同的更新量条件下, 3 种方案的联合搜索所需时间均有所缩短. 但值得注意的是, 由于方案 RFBC 具备鲁棒性, 其搜索效率显著优于 BDXT 和 ODXT.

因为 BDXT 和 ODXT 均无法处理不合理的更新, 导致这些不合理更新产生的无效密文仍会被存储到服务器中. 当进行联合查询时, 这些无效的密文仍会参与计算, 增加了计算开销, 因此二者在不同比例不合理更新下的搜索表现均差于 RFBC.

RFBC 方案在更新时会对不合理的更新操作进行过滤, 因此在进行联合查询时, RFBC 检索出来的文件都是合理添加或删除的 (不包括重复添加或删除的文件). 此外, RFBC 也无需对不合理的更新操作进行复杂计算. 以更新 1.0×10^6 个关键词文件对时为例, 假设此时不合理更新比例 *iup* 为 0.1, 那么 BDXT 和 ODXT 需要存储 1.0×10^6 条数据, 并对这些数据进行运算处理, 而 RFBC 通过布隆过滤器过滤掉其中的不合理的更新, 即过滤掉 1.0×10^5 条数据, 只需存储并处理剩余的 9.0×10^5 条合理的数据更新请求. 此外, 与 ODXT 和 BDXT 相比, RFBC 在关键词文件数据对逐渐增大的情况下优势更加明显 (斜率更低), 具体来讲, 当关键词文件数据对为 10^6 且不合理更新比例 *iup* 分别为 0.1、0.2、0.3 和 0.4 时, RFBC 的搜索操作开销分别约为 ODXT 和 BDXT 的 24.3% 和 26.8%、21.2% 和 27.4%、21.5% 和 27.1% 以及 20.8% 和 27.9%. 平均来看, RFBC 在不合理更新下的搜索时间开销分别约为 ODXT 和 BDXT 的 21.9% 和 27.3%.

此外, 图 7 进一步展示了不合理更新对方案 RFBC 联合搜索效率的影 响, 从图 7 中可以看出, 随着不合理更新比例的增加, RFBC 的搜索时间逐渐减少. 其原因如前所述, 随着不合理更新比例的增加, 服务器将过滤掉更多的不合理更新操作, 仅需处理合理更新的关键词文件数据对.

(4) 不同更新计数器数值下的搜索时间

图 8 显示了更新计数器值最小的关键词对应的文件数量对联合搜索效率的影响, 其中联合查询 $q=w_1\wedge w_2\wedge\ldots\wedge$

w_8, 我们假设 w_1 的更新计数器值最小且逐渐增大. 当进行联合搜索时, 搜索包含的关键词越多需要检查的文件就越多, 因此搜索时间会受关键词数量的影响. 同时, 由于添加的文件数量远大于检索关键词数量, 再结合上文我们对为何选取更新计数器值最小的关键词 w_1 做出的解释, 可知此时联合搜索时间主要受更新计数器值最小的关键词所更新的文件数量所影响.

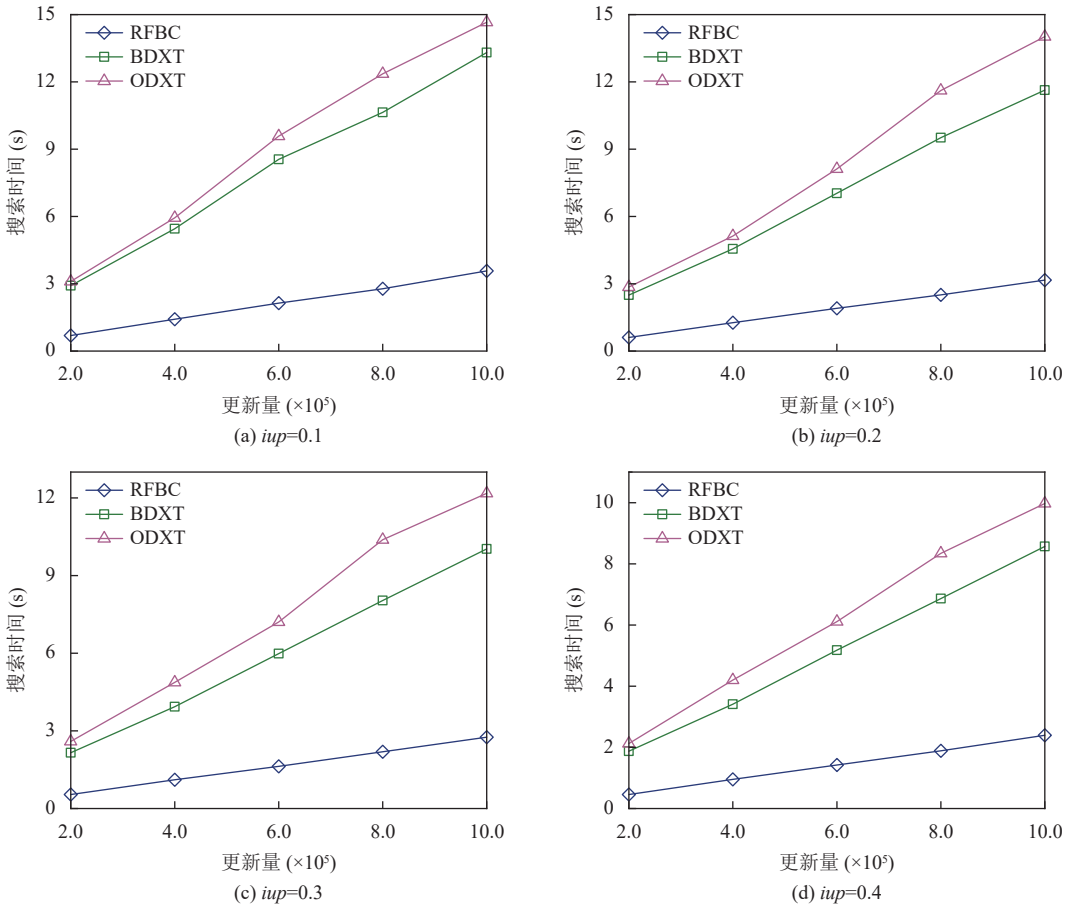

图 6　不合理更新下的搜索时间

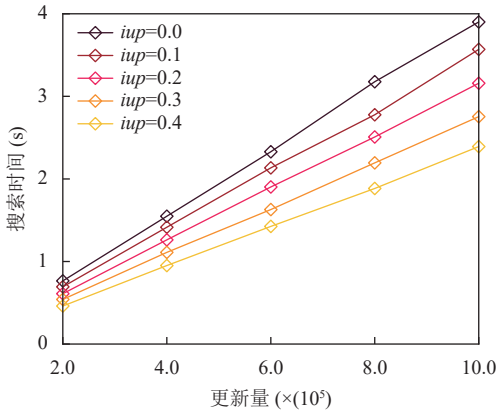

图 7　RFBC 在不同比例的不合理更新下的搜索时间

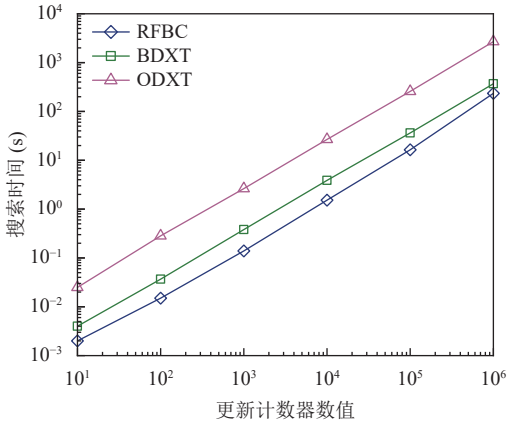

图 8　不同更新计数器数值下的搜索时间

实际上, 3 个方案的搜索时间主要受更新计数器值最小的关键词 (如前所述的 w_1) 所更新的文件数量的影响. 图中 RFBC 的搜索时间显然优于 BDXT 和 ODXT, 其原因如前所述, 当更新计数器数值固定时, ODXT 方案需要在客户端通过模幂运算计算出更新计数器值最小的关键词所更新的所有文件与每个查询关键词对应的 *xtoken*, 在服务端也要执行类似的操作, 因此 ODXT 方案的搜索时间要明显高于 BDXT 方案和 RFBC 方案. 而 BDXT 方案需要在 ClientRound2 阶段筛选出更新计数器值最小的关键词所更新的所有文件, 计算这些文件与每个查询关键词对应的两个 *xtag*, 并在 ServerRound2 阶段对这些 *xtag* 进行判断操作, 最后将判断结果返回给客户端, 客户端在 FinalOuoput 阶段根据服务器返回的判断结果判断在更新计数器值最小的关键词所更新的所有文件中保留哪些文件. 此过程需要客户端与服务器的多轮通信及多次筛选操作, 所以时间开销大于 RFBC. 而 RFBC 相较于 ODXT 不需要复杂的模幂运算, 相较于 BDXT 无需多轮通信, 也无需多次筛选操作, 只需简单的计算出 *token* 并将其发送至服务器端, 使用布隆过滤器的判断即可筛选出目标文件集合.

(5) 多关键词搜索时间开销

图 9 描述了不同关键词个数对搜索时间开销的影响. 从图 9 中我们可以看出, 在固定更新数据量的情况下, 随着多关键词联合查询的个数逐渐增加, RFBC 和 BDXT 的搜索时间基本不变, 也就是说, 联合查询的关键词个数对 RFBC 和 BDXT 没有影响. 同时, RFBC 的多关键词搜索时间不到 BDXT 的 1/2. ODXT 的搜索时间均高于 RFBC 和 BDXT, 且随着关键词个数的增加, ODXT 的查询时间逐渐增长. 其原因如上文所述, ODXT 需要对联合查询中的关键词执行模幂运算, 因此造成了计算开销的上升.

图 10 则描述了不同关键词长度对搜索时间开销的影响. 我们以基准关键词的不同倍长度为测试变量 (×*n* 表示 *n* 倍, *n* 为 1–8), 从图中我们可以看出, 在同样固定更新数据量的情况下, 随着联合查询中关键词的长度逐渐增加, 3 个方案的搜索时间均基本不变, 也就是说, 联合查询的关键词长度对三者均无影响. 本文的 RFBC 在此种情况下的搜索时间低于 BDXT 和 ODXT, 分别约为他们的 1/2 和 1/5.

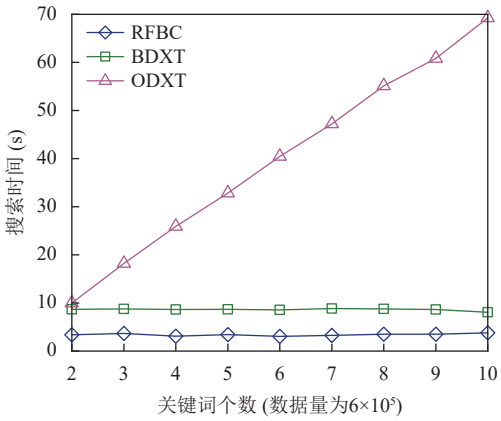

图 9 不同关键词个数的搜索时间开销

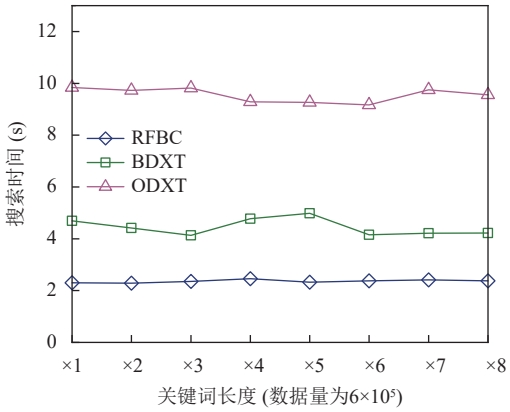

图 10 不同关键词长度的搜索时间开销

4.3 通信开销比较

我们主要通过比较 RFBC、BDXT 和 ODXT 在联合搜索过程中的通信开销来评估它们的通信效率. 与计算开销的评估方法相似, 我们同样设置不同比例的不合理更新来观察其对搜索过程通信开销的影响. 实验结果如图 11 所示, 其中横坐标代表更新数量, 纵坐标代表搜索过程的通信开销. 显然, 随着更新数量的上升, 相应的通信开销也随之上升, 但是随着不合理更新比例的增加, RFBC 的通信开销仍显著低于 BDXT 和 ODXT.

由于 RFBC 和 ODXT 仅需一轮通信就可得到联合搜索结果, 且客户端与服务器之间传输的数据较少, 而 BDXT 需要两轮通信才能完成联合搜索, 因此 RFBC 和 ODXT 的通信开销明显低于 BDXT. 具体来讲, RFBC 首先使用伪随机函数计算出每个关键词对应的关键词令牌, 再计算出关键词对应的布隆过滤器的存储地址, 最后将关键词令

牌集合、布隆过滤器的存储地址集合以及更新计数器值最小的关键词最新的状态信息发送给服务器. 服务器经过一系列筛选操作将加密的目标文件标识符集合返回给客户端. ODXT 同样首先筛选出更新计数器值最小的关键词, 然后使用伪随机函数计算出该关键词每次更新的相关地址信息, 并且计算该次更新和其他关键词关联的信息, 并将这些信息发送给服务器. 随后, 服务器利用接收到的信息, 计算所有包含相关文件标识符的 *sval* 值, 将其与其他信息一起返回给客户端, 客户端以此过滤掉不需要的文件的标识符, 得到最终搜索结果. 在 RFBC 和 ODXT 的一轮通信过程, ODXT 方案中客户端发送给服务器的数据量和服务器发送给客户端数据量均大于 RFBC 方案中所发送的数据量. 在 BDXT 方案第 1 轮通信中, 客户端通过计算地址信息, 和服务器交互获取 *TSet* 集合中的元素. 第 2 轮通信中, 客户端向服务器发送 *xtag* 列表, 服务器向客户端返回针对每个文件标识符的判断信息, 用于客户端最后的文件筛选. 如前所述, ODXT 相比于 BDXT 降低了通信轮数, 因此该方案的所需要的交互信息少于 BDXT, 通信开销相比 BDXT 也显著降低.

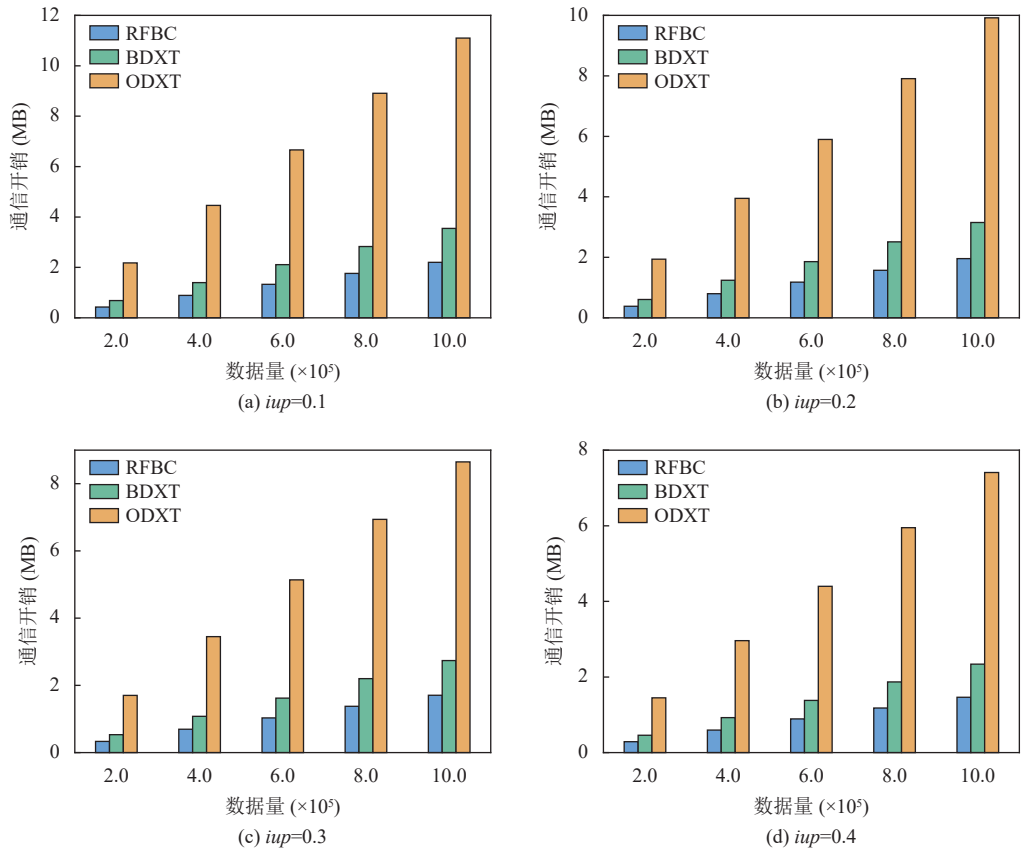

图 11　3 个方案在合理更新下的联合搜索通信开销

总的来说, 随着不合理更新比例的增加, 3 个方案的通信开销都逐渐减少. 并且因为 RFBC 在进行联合搜索操作时无需对不合理的更新文件进行搜索, 使得其通信开销的优势更为显著.

5 总　结

本文回顾了对称可搜索加密方案研究进展, 针对现有方案多不具有鲁棒性和不支持联合搜索问题, 提出了一个鲁棒的前后向隐私对称可搜索加密方案 RFBC. 它不仅满足前向隐私和 Type-III 后向隐私, 还能够通过设置布隆过滤器来支持联合查询和处理不合理的更新请求. 此外, 形式化安全分析证明本文方案 RFBC 具有前向隐私、

Type-III 后向隐私以及鲁棒性. 详细的实验分析和评估表明相较于相关工作 BDXT 和 ODXT, 方案 RFBC 是计算和通信高效的. 未来, 我们将在现有方案的基础上进一步提升方案的安全性能. 同时, 我们也将继续探索基于其他密码原语和工具, 以期在现有工作的基础上实现更多功能, 实现更为高效、安全的动态对称可加密搜索方案.

References:

[1] Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Proc. of the 2000 IEEE Symp. on Security and Privacy. Berkeley: IEEE, 2000. 44–55. [doi: 10.1109/SECPRI.2000.848445]

[2] Kamara S, Papamanthou C, Roeder T. Dynamic searchable symmetric encryption. In: Proc. of the 2012 ACM Conf. on Computer and Communications Security. Raleigh: ACM, 2012. 965–976. [doi: 10.1145/2382196.2382298]

[3] Zhang YP, Katz J, Papamanthou C. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In: Proc. of the 25th USENIX Conf. on Security Symp. Austin: USENIX Association, 2016. 707–720.

[4] Stefanov E, Papamanthou C, Shi E. Practical dynamic searchable encryption with small leakage. In: Proc. of the 21st Annual Network and Distributed System Security Symp. San Diego: NDSS, 2014. 23–26.

[5] Huang YC, Li SS, Yu B. A survey of symmetric searchable encryption in cloud environment. Journal of Electronics & Information Technology, 2023, 45(3): 1134–1146 (in Chinese with English abstract). [doi: 10.11999/JEIT211572]

[6] Wang YL, Chen XF. Research on searchable symmetric encryption. Journal of Electronics & Information Technology, 2020, 42(10): 2374–2385 (in Chinese with English abstract). [doi: 10.11999/JEIT190890]

[7] Liu WX, Gao Y. A survey on security development of searchable symmetric encryption. Journal of Cyber Security, 2021, 6(2): 73–84 (in Chinese with English abstract). [doi: 10.19363/J.cnki.cn10-1380/tn.2021.03.05]

[8] Asharov G, Segev G, Shahaf I. Tight tradeoffs in searchable symmetric encryption. Journal of Cryptology, 2021, 34(2): 9. [doi: 10.1007/s00145-020-09370-z]

[9] Goh EJ. Secure indexes. 2003. https://crypto.stanford.edu/~eujin/papers/secureindex/secureindex.pdf

[10] Curtmola R, Garay J, Kamara S, Ostrovsky R. Searchable symmetric encryption: Improved definitions and efficient constructions. In: Proc. of the 13th ACM Conf. on Computer and Communications Security. Alexandria: ACM, 2006. 79–88. [doi: 10.1145/1180405.1180417]

[11] Cash D, Jarecki S, Jutla C, Krawczyk H, Roşu MC, Steiner M. Highly-scalable searchable symmetric encryption with support for Boolean queries. In: Proc. of the 33rd Annual Cryptology Conf. Santa Barbara: Springer, 2013. 353–373. [doi: 10.1007/978-3-642-40041-4_20]

[12] Lai SQ, Patranabis S, Sakzad A, Liu JK, Mukhopadhyay D, Steinfeld R, Sun SF, Liu DX, Zuo C. Result pattern hiding searchable encryption for conjunctive queries. In: Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security. Toronto: ACM, 2018. 745–762. [doi: 10.1145/3243734.3243753]

[13] Kamara S, Papamanthou C. Parallel and dynamic searchable symmetric encryption. In: Proc. of the 17th Int'l Conf. on Financial Cryptography and Data Security. Okinawa: Springer, 2013. 258–274. [doi: 10.1007/978-3-642-39884-1_22]

[14] Bost R. ∑οφος: Forward secure searchable encryption. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 1143–1154. [doi: 10.1145/2976749.2978303]

[15] Kim KS, Kim M, Lee D, Park JH, Kim WH. Forward secure dynamic searchable symmetric encryption with efficient updates. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 1449–1463. [doi: 10.1145/3133956.3133970]

[16] Etemad M, Küpçü A, Papamanthou C, Evans D. Efficient dynamic searchable encryption with forward privacy. Proc. on Privacy Enhancing Technologies, 2018, 2018(1): 5–20. [doi: 10.1515/popets-2018-0002]

[17] Song XF, Dong CY, Yuan DD, Xu QL, Zhao MH. Forward private searchable symmetric encryption with optimized I/O efficiency. IEEE Trans. on Dependable and Secure Computing, 2020, 17(5): 912–927. [doi: 10.1109/TDSC.2018.2822294]

[18] Zhang ZJ, Wang JF, Wang YL, Su YP, Chen XF. Towards efficient verifiable forward secure searchable symmetric encryption. In: Proc. of the 24th European Symp. on Research in Computer Security. Luxembourg: Springer, 2019. 304–321. [doi: 10.1007/978-3-030-29962-0_15]

[19] Yang JJ, Liu F, Luo XY, Hong JN, Li J, Xue KP. Forward private multi-client searchable encryption with efficient access control in cloud storage. In: Proc. of the 2022 IEEE Global Communications Conf. Rio de Janeiro: IEEE, 2022. 3791–3796. [doi: 10.1109/GLOBECOM48099.2022.10001146]

[20] Patranabis S, Mukhopadhyay D. Forward and backward private conjunctive searchable symmetric encryption. 2021. https://www.ndss-symposium.org/wp-content/uploads/ndss2021_2C-3_23116_paper.pdf

[21] Yuan DD, Cui SJ, Russello G. We can make mistakes: Fault-tolerant forward private verifiable dynamic searchable symmetric encryption. In: Proc. of the 7th IEEE European Symp. on Security and Privacy (EuroS&P). Genoa: IEEE, 2022. 587–605. [doi: 10.1109/EuroSP53844.2022.00043]

[22] Mei L, Xu CG, Xu L, Yuan XL, Liu JK. Practical multi-source multi-client searchable encryption with forward privacy: Refined security notion and new constructions. IEEE Trans. on Dependable and Secure Computing, 2024, 21(1): 63–77. [doi: 10.1109/TDSC.2023.3245638]

[23] Kamara S, Moataz T. Boolean searchable symmetric encryption with worst-case sub-linear complexity. In: Proc. of the 36th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Paris: Springer, 2017. 94–124. [doi: 10.1007/978-3-319-56617-7_4]

[24] Wang YL, Wang JF, Sun SF, Miao MX, Chen XF. Toward forward secure SSE supporting conjunctive keyword search. IEEE Access, 2019, 7: 142762–142772. [doi: 10.1109/ACCESS.2019.2944246]

[25] Bost R, Minaud B, Ohrimenko O. Forward and backward private searchable encryption from constrained cryptographic primitives. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 1465–1482. [doi: 10.1145/3133956.3133980]

[26] Sun SF, Yuan XL, Liu JK, Steinfeld R, Sakzad A, Vo V, Nepal S. Practical backward-secure searchable encryption from symmetric puncturable encryption. In: Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security. Toronto: ACM, 2018. 763–780. [doi: 10.1145/3243734.3243782]

[27] Chamani JG, Papadopoulos D, Papamanthou C, Jalili R. New constructions for forward and backward private symmetric searchable encryption. In: Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security. Toronto: ACM, 2018. 1038–1055. [doi: 10.1145/3243734.3243833]

[28] Hoang T, Yavuz AA, Guajardo J. A secure searchable encryption framework for privacy-critical cloud storage services. IEEE Trans. on Services Computing, 2021, 14(6): 1675–1689. [doi: 10.1109/TSC.2019.2897096]

[29] He K, Chen J, Zhou QX, Du RY, Xiang Y. Secure dynamic searchable symmetric encryption with constant client storage cost. IEEE Trans. on Information Forensics and Security, 2021, 16: 1538–1549. [doi: 10.1109/TIFS.2020.3033412]

[30] Demertzis I, Chamani JG, Papadopoulos D, Papamanthou C. Dynamic searchable encryption with small client storage. In: Proc. of the 27th Annual Network and Distributed System Security Symp. San Diego: NDSS, 2020.

[31] Sun SF, Steinfeld R, Lai SQ, Yuan XL, Sakzad A, Liu JK, Nepal S, Gu DW. Practical non-interactive searchable encryption with forward and backward privacy. In: Proc. of the 28th Annual Network and Distributed System Security Symp. NDSS. 2021.

[32] Vo V, Lai SQ, Yuan XL, Nepal S, Liu JK. Towards efficient and strong backward private searchable encryption with secure enclaves. In: Proc. of the 19th Int'l Conf. on Applied Cryptography and Network Security. Kamakura: Springer, 2021. 50–75. [doi: 10.1007/978-3-030-78372-3_3]

[33] Vo V, Lai SQ, Yuan XL, Sun SF, Nepal S, Liu JK. Accelerating forward and backward private searchable encryption using trusted execution. In: Proc. of the 18th Int'l Conf. on Applied Cryptography and Network Security. Rome: Springer, 2020. 83–103. [doi: 10.1007/978-3-030-57878-7_5]

[34] Amjad G, Kamara S, Moataz T. Forward and backward private searchable encryption with SGX. In: Proc. of the 12th European Workshop on Systems Security. Dresden: ACM, 2019. 4. [doi: 10.1145/3301417.3312496]

[35] Huang YY, Lv SY, Liu ZL, Song XF, Li J, Yuan YL, Dong CY. Cetus: An efficient symmetric searchable encryption against file-injection attack with SGX. Science China Information Sciences, 2021, 64(8): 182314. [doi: 10.1007/s11432-020-3039-x]

[36] Zuo C, Sun SF, Liu JK, Shao J, Pieprzyk J. Dynamic searchable symmetric encryption with forward and stronger backward privacy. In: Proc. of the 24th European Symp. on Research in Computer Security. Luxembourg: Springer, 2019. 283–303. [doi: 10.1007/978-3-030-29962-0_14]

[37] Wu ZQ, Cai ZB, Tang XY, Xu YM, Deng T. A forward and backward private oblivious RAM for storage outsourcing on edge-cloud computing. Journal of Parallel and Distributed Computing, 2022, 166: 1–14. [doi: 10.1016/j.jpdc.2022.04.008]

[38] Stefanov E, Van Dijk M, Shi E, Chan THH, Fletcher C, Ren L, Yu XY, Devadas S. Path ORAM: An extremely simple oblivious RAM protocol. Journal of the ACM (JACM), 2018, 65(4): 18. [doi: 10.1145/3177872]

[39] Li J, Huang YY, Wei Y, Lv SY, Liu ZL, Dong CY, Luo WJ. Searchable symmetric encryption with forward search privacy. IEEE Trans. on Dependable and Secure Computing, 2021, 18(1): 460–474. [doi: 10.1109/TDSC.2019.2894411]

[40] Xu P, Susilo W, Wang W, Chen TY, Wu QH, Liang KT, Jin H. ROSE: Robust searchable encryption with forward and backward security. IEEE Trans. on Information Forensics and Security, 2022, 17: 1115–1130. [doi: 10.1109/TIFS.2022.3155977]

[41] Zuo C, Lai SQ, Yuan XL, Liu JK, Shao J, Wang HX. Searchable encryption for conjunctive queries with extended forward and backward privacy. IACR Cryptology ePrint Archive, 2021, 2021: 1585.

[42] Chen TY, Xu P, Wang W, Zheng YB, Susilo W, Jin H. Bestie: Very practical searchable encryption with forward and backward security. In: Proc. of the 26th European Symp. on Research in Computer Security. Darmstadt: Springer, 2021. 3–23. [doi: 10.1007/978-3-030-88428-4_1]

[43] Zuo C, Sun SF, Liu JK, Shao J, Pieprzyk J, Xu L. Forward and backward private DSSE for range queries. IEEE Trans. on Dependable and Secure Computing, 2022, 19(1): 328–338. [doi: 10.1109/TDSC.2020.2994377]

[44] Wang XY, Ma JF, Liu XM, Miao YB, Liu Y, Deng RH. Forward/backward and content private DSSE for spatial keyword queries. IEEE Trans. on Dependable and Secure Computing, 2023, 20(4): 3358–3370. [doi: 10.1109/TDSC.2022.3205670]

[45] Xu L, Duan HY, Zhou AX, Yuan XL, Wang C. Interpreting and mitigating leakage-abuse attacks in searchable symmetric encryption. IEEE Trans. on Information Forensics and Security, 2021, 16: 5310–5325. [doi: 10.1109/TIFS.2021.3128823]

[46] Chamani JG, Papadopoulos D, Karbasforushan M, Demertzis I. Dynamic searchable encryption with optimal search in the presence of deletions. In: Proc. of the 31st USENIX Security Symp. USENIX Association, 2022. 2425–2442.

[47] Jiang Q, Chang EC, Qi Y, Qi SY, Wu PF, Wang JF. Rphx: Result pattern hiding conjunctive query over private compressed index using Intel SGX. IEEE Trans. on Information Forensics and Security, 2022, 17: 1053–1068. [doi: 10.1109/TIFS.2022.3144877]

[48] Xu C, Wang RJ, Zhu LH, Zhang C, Lu RX, Sharif K. Efficient strong privacy-preserving conjunctive keyword search over encrypted cloud data. IEEE Trans. on Big Data, 2023, 9(3): 805–817. [doi: 10.1109/TBDATA.2022.3205668]

[49] Bloom BH. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 1970, 13(7): 422–426. [doi: 10.1145/362686.362692]

[50] Feng DG. Research on theory and approach of provable security. Ruan Jian Xue Bao/Journal of Software, 2005, 16(10): 1743–1756 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1743.htm

附中文参考文献:

[5] 黄一才, 李森森, 郁滨. 云环境下对称可搜索加密研究综述. 电子与信息学报, 2023, 45(3): 1134–1146. [doi: 10.11999/JEIT211572]

[6] 王赟玲, 陈晓峰. 对称可搜索加密技术研究进展. 电子与信息学报, 2020, 42(10): 2374–2385. [doi: 10.11999/JEIT190890]

[7] 刘文心, 高莹. 对称可搜索加密的安全性研究进展. 信息安全学报, 2021, 6(2): 73–84. [doi: 10.19363/J.cnki.cn10-1380/tn.2021.03.05]

[50] 冯登国. 可证明安全性理论与方法研究. 软件学报, 2005, 16(10): 1743–1756. http://www.jos.org.cn/1000-9825/16/1743.htm

张文琪(1999－), 男, 博士生, 主要研究领域为隐私计算.

梁伟(1978－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为网络安全, 区块链.

李雄(1984－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为应用密码学, 数据安全, 隐私计算.

黄可(1989－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为区块链, 应用密码学.

尹智明(1995－), 男, 硕士, 主要研究领域为密码协议, 隐私计算.

张小松(1968－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为网络安全, 卫星互联网安全, 区块链.