计算机科学
COMPUTER SCIENCE

基于vORAM的前向和后向安全动态可搜索加密方案

邵通, 李川, 薛雷, 刘扬, 赵凝, 陈青

引用本文

邵通, 李川, 薛雷, 刘扬, 赵凝, 陈青. 基于vORAM的前向和后向安全动态可搜索加密方案[J]. 计算机科学, 2024, 51(6A): 230500098-9.

SHAO Tong, LI Chuan, XUE Lei, LIU Yang, ZHAO Ning, CHEN Qing. Forward and Backward Secure Dynamic Searchable Encryption Schemes Based on vORAM[J]. Computer Science, 2024, 51(6A): 230500098-9.

相似文章推荐（请使用火狐或 IE 浏览器查看文章）

Similar articles recommended (Please use Firefox or IE to view the article)

基于可更新加密的保护搜索模式的动态可搜索加密方案

Dynamic Searchable Symmetric Encryption Based on Protected Search Mode of Updatable Encryption

计算机科学, 2024, 51(3): 340-350. https://doi.org/10.11896/jsjkx.230100016

基于图卷积网络的甲型流感H3N2抗原性预测

Antigenicity Prediction of Influenza A/H3N2 Based on Graph Convolutional Networks

计算机科学, 2023, 50(11A): 230100113-6. https://doi.org/10.11896/jsjkx.230100113

基于多粒度对比学习的聊天对话摘要模型

Chat Dialogue Summary Model Based on Multi-granularity Contrastive Learning

计算机科学, 2023, 50(11): 192-200. https://doi.org/10.11896/jsjkx.230300241

基于主成分分析和函数机制的差分隐私线性回归算法

Differential Privacy Linear Regression Algorithm Based on Principal Component Analysis andFunctional Mechanism

计算机科学, 2023, 50(8): 342-351. https://doi.org/10.11896/jsjkx.220800255

类脑心智计算的科学技术和工程应用的研究与思考

Thoughts on Development and Research of Science,Technology and Engineering Application of Brain & Mind-inspired Computing

计算机科学, 2023, 50(2): 364-373. https://doi.org/10.11896/jsjkx.220500023

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230500098

基于 vORAM 的前向和后向安全动态可搜索加密方案

邵 通 李 川 薛 雷 刘 扬 赵 凝 陈 青
国网河北省电力有限公司保定供电公司 河北 保定 071000

摘 要 为了解决将敏感数据加密存储在云平台所带来的关键字检索问题,引进 vORAM 提出新的前向和后向安全动态可搜索加密方案——FBDSE-I 方案。该方案利用茫然数据结构的历史独立性以及安全删除性,实现了关键词/文件标识符对的直接删除,在保证数据更新安全的同时,简化了动态更新过程。为了实现更加高效的查询操作,进一步提出方案 FBDSE-II,利用映射字典结构解耦茫然原语与检索结果,减少查询过程中对 vORAM 的访问次数。给出了形式化的安全证明,证明了 FBDSE-I 和 FBDSE-II 方案在保证前向安全的同时,分别满足 Type-I 和 Type-III 后向安全。仿真实验结果显示,相对于同等安全级别的前向和后向安全动态可搜索加密方案,FBDSE-I 和 FBDSE-II 方案具有更高的查询和更新效率,而且数据集合规模越大,优势越显著。

关键词:动态可搜索加密;前向安全;后向安全;茫然数据结构;vORAM
中图分类号 TP309

Forward and Backward Secure Dynamic Searchable Encryption Schemes Based on vORAM

SHAO Tong,LI Chuan,XUE Lei,LIU Yang,ZHAO Ning and CHEN Qing
Baoding Electric Power Corporation,State Grid Hebei Electric Power Company,Baoding,Hebei 071000,China

Abstract To solve the problem of keyword retrieval caused by encrypting and storing sensitive data on the cloud platform,a forward and backward secure dynamic searchable encryption scheme FBDSE-I is proposed by introducing a new oblivious data structure. By using the history-independence and secure deletion of the oblivious data structure,FBDSE-I scheme realizes the direct deletion of keyword/file-identifier pairs,ensures the security of data updating,and simplifies the dynamic update process. Furthermore,an improved scheme,FBDSE-II,is proposed to achieve more efficient query operation. The map dictionary structure is used to decouple the oblivious primitives and search results,so as to reduce the number to access vORAM in the query process. In addition,the formal security proof is given. It is proved that FBDSE-I and FBDSE-II schemes respectively satisfy Type-I and Type-III backward security while ensuring forward security. Experimental results show that FBDSE-I and FBDSE-II schemes are more efficient than the forward and backward secure dynamic searchable encryption schemes at the same level. In particular,the larger the scale of data sets,the more significant the advantage becomes.

Keywords Dynamic searchable encryption,Forward secure,Backward secure,Oblivious data structure,vORAM

1 引言

由于业务的需求,有时需要将业务数据存储到共有云服务平台,随之带来数据隐私泄露问题[1]。一些数据是非常敏感的,这些数据虽然存储在公共云服务器上,但不希望数据被云服务提供商获取,造成敏感数据泄露。为了保障用户的数据隐私安全,防止数据泄露,常规方法是在外包数据之前对数据进行加密再上传到服务器。但这些复杂的加密技术严重阻碍了云数据的高效利用,特别是在明文数据上广泛使用的基于关键字的信息检索技术不能直接应用于加密数据。为了实现对加密数据的检索,Song 等[2]首次提出了可搜索加密(Searchable Encryption,SE)。自此,国内外学者在 SE 的效率、表现力和安全性方面进行了大量研究,但早期的大多数方案都是静态的,不支持加密数据的动态更新。

动态可搜索加密方案(Dynamic Searchable Encryption,DSE)[3]为用户提供密文上关键字搜索功能的同时,对加密数据集还支持添加和删除的更新能力。从安全的角度来看,设计安全的 DSE 方案是具有挑战性的,因为数据的添加和删除比搜索操作会泄漏更多信息,如 DSE 可能会泄露新添加到数据库的文件包含之前搜索过的关键字,或者已经删除的文件包含搜索关键字,避免添加文件造成泄露的方案称其具备前向安全。由于文献[4]利用文件注入攻击和更新泄露的信息可以成功攻击不具备前向安全的 DSE 方案,因此前向安全是所有 DSE 方案的关键属性,否则更新的能力在某种程度上是无意义的。后向安全是限制文件删除过程中的信息泄露,由 Stefanov 等[3]首次提出,Bost 等[5]正式定义,主要包含以下 3 种不同类型的泄漏:对于关键字 w,Type-I 泄露当前数据库中包含 w 的更新次数和包含 w 文件标识符及其存储时间,Type-II 额外泄露 w 之前所有更新的时间和类型(添加或删除),Type-III 额外泄露删除操作与其对应的添加操作的时间。

为了防止动态更新中的信息泄露,现有的 DSE 方案引入茫然随机访问机(Oblivious Random Access Machine,ORAM)隐藏访问模式,实现更高级别的后向安全,尤其是

通信作者:邵通(15613668123@163.com)

Chamani 等[6-7]将 AVL 树和 ORAM 结构结合设计了系列方案，其中 AVL 树表示平衡二叉查找树，命名来源于其发明作者 Adelson-Velsky 和 Landis。但是由于 ORAM+AVL 结构通信开销非常高，导致 DSE 方案的实用性很差。比如，当数据库大小为 1 MB 以及查询结果集的文件数量为 100 时，文献[6]中 ORION 方案传输所有必需的 ORAM 块需要 1.7 MB。此外，AVL+ORAM 结构本身只具备茫然性质，为了达到 DSE 方案更新安全的目标，还需通过标志删除状态以及利用插入操作覆盖等额外操作来实现，从而增加了更新操作的复杂度。

为了解决上述问题，本文引进了一个新的茫然数据结构[8]，该结构通过使用指针技术在 vORAM 上存储 HIRB 树实现，与 ORAM+AVL 结构相比，有以下 3 个优势：1）具备更多的隐私保证，即历史独立性和安全删除性；2）降低了交互轮数，具有更低的通信开销。如当数据集大小为 2^{10} 时，HIRB 和 AVL 每个操作的交互轮数分别为 4 和 2 096，通信开销分别为 286.7 kB 和 8.6 MB；3）节省巨大的填充成本，HIRB 树的每个操作至多需要访问 $2\times\log_{\beta}N$ 个节点，而 AVL 树必须填充到访问 $3\times1.44\times\log N$ 个节点。综上，本文旨在利用该 HIRB+AVL 结构设计具有高安全级别（具备前向和后向安全）且实用的 DSE 方案，主要的研究工作总结如下：

1）利用 HIRB+vORAM 结构的历史独立性和安全删除性设计了 FBDSE-I 方案，在查询和更新过程中，通过隐藏用户真正访问位置，达到 Type-I 后向安全，同时实现准最优搜索时间[1)]。

2）为了减少查询时 vORAM 的访问次数，提出了 FBDSE-II 方案。该方案通过解耦茫然原语与检索结果，提高了 FBDSE-I 方案的查询效率，且经过安全分析，其满足 Type-III 后向安全。

3）仿真结果表明，FBDSE-I 方案和 FBDSE-II 方案与基于 AVL+ORAM 结构的 DSE 方案相比，在保证数据更新安全的同时，具有更好的搜索与更新性能。

2 相关工作

自 Song 等[2]首次提出了基于密文扫描的可搜索加密方案，国内外学者提出了一系列方案[9-12,21-22]，用于解决检索效率和安全性的问题，但早期的大多数方案都是静态的，不支持数据的动态更新。直到 2013 年，Kamara 等[13]提出了第一个满足亚线性复杂度的动态可搜索加密方案。该方案是非交互的高效对称可搜索加密方案，可以处理任意的析取和布尔查询。但该方案没有考虑数据的前向安全和后向安全。

Stefanov 等[3]最早提出了一个由 ORAM 启发的前向安全对称可搜索加密方案，但由于使用 TW-ORAM 的结构，导致通信和计算开销大。随后，Bost 等[14]正式给出了动态可搜索加密的前向安全定义，并提出了陷门置换技术。客户端只存储一个 ST_c，发生更新时，客户端根据陷门的私钥计算更新后的 ST_{c+1}。需要查询时，客户端将最新的 ST_{c+1} 发送给云服务器。云服务器根据陷门的公钥以及 ST_{c+1}，可以很容易得到所有的 ST_0 到 ST_c，而云服务器不能根据 ST_c 计算得到 ST_{c+1}，文件的前向安全得到保证。自此，提出了一系列使用对称加密原语实现前向安全的方法[15-17]。后向安全同样由 Stefanov 等[3]首次提出，由 Bost 等[5]正式定义，其目的是限制执行搜索关键字时服务器可以学习到已经被删除条目的信息。文献[5]介绍了后向安全的 3 种不同的泄露类型。自此，出现了一系列有关后向安全的 DSE 方案[6-7,18-20]。尤其是 2018 年，Chamani 等[6]根据上述的 3 种不同类型泄露，在保证方案前向安全的同时，提出了 3 个新的不同后向安全级别的 DSE 方案。第一个方案 MITRA，利用映射字典存储加密记录，除了泄漏更新时间，不会向服务器泄露任何其他添加和删除相关的信息，在提供前向安全的同时，实现了 Type-II 后向安全。第二个方案 ORION 利用 AVL+ORAM 结构，通过隐藏更新位置极大地减少了更新信息的泄漏，实现了前向安全和最高级的 Type-I 后向安全。第三个方案 HORUS 同样使用了 AVL+ORAM 结构，但是其将 ORION 中的部分功能通过简单的非递归 Path-ORAM 结构实现，从而减少了交互轮数，提高了检索效率，提供了前向安全和 Type-III 后向安全。2020 年，Chamani 等[7]也使用该茫然结构设计了两个满足前向和后向安全的 DSE 方案，即 SDd 方案和 QOS 方案，但是 AVL+ORAM 结构本身就存在性能问题，需在本质上通过更改使用的基本结构来提高 DSE 性能。

3 预备知识

3.1 符号定义

本文使用的主要符号的说明如表 1 所列。

表 1 符号说明

Table 1 Symbol description

参数	含义
W	关键字空间 $W=\{w_1,\cdots,w_{\lvert W\rvert}\}$
D	文件集合 $D=\{d_1,\cdots,d_{\lvert D\rvert}\}$
$\lvert W\rvert$	关键词数量
$\lvert D\rvert$	文件数量
ID	文件标识符集合 $ID=\{id_1,\cdots,id_{\lvert D\rvert}\}$
DB	数据库 DB 包含所有的 (w,id) 对
$\lvert DB\rvert$	数据库中包含 (w,id) 对的数量
EDB	加密数据库
$DB(w)$	包含关键字 w 的文档集
n_w	包含关键字 w 的文档数量
d_w	关键字 w 的删除次数

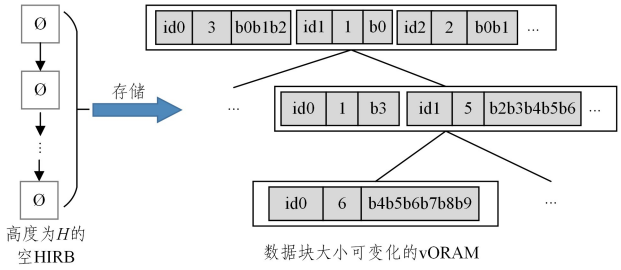
3.2 茫然数据结构

茫然数据结构（Oblivious Data Structure）旨在隐藏数据结构上执行的一系列操作（例如插入、删除或查找）的内容。对于两个长度相同的操作序列，在服务器上产生的访问模式是不可区分的。

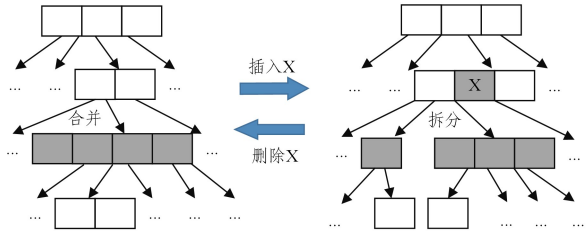
本文的 FBDSE-I 和 FBDSE-II 方案都使用文献[8]中基于键值映射构建的茫然数据结构来隐藏访问模式，该结构通过在 vORAM 上部署 HIRB 树来实现，即每个 HIRB 树的节点都对应一条 vORAM 路径且每个节点的数据被分成子数据块填充在该路径上。如图 1 所示，vORAM 的不同之处在于其桶中数据块的大小可变化。比如，根节点中标识符为 id_0 和 id_1 块的大小可以不同。此外，每个标识符的部分块填充到 vORAM 根至叶子的路径上。HIRB 树是具有历史独立性的 B 树，其定义了 Setup，Lookup，Insert 和 Remove 4 个协议来初始化结构、检索给定的键值对以及插入或删除指定的键值对。

1) 如果 DSE 方案的查询时间复杂度是 $O(nw)$，则具有最优查询时间；如果是 $O(nw\,\mathrm{polylog}(N))$，则具有准最优查询时间。

具体地，Setup 初始化高度为 H、分支因子为 1 的空 HIRB 树，并存储在 vORAM 中。Lookup 协议需要沿着搜索路径从 vORAM 获取每个 HIRB 节点，并返回匹配的值。Insert 和 Remove 协议是从 HIRB 中插入或删除一个元素，分别沿着搜索路径从该元素所在高度节点到叶子节点拆分或合并节点。图 1(b)给出了 HIRB 树的更新过程，可以看出更新过程中只涉及拆分合并节点，相比 AVL 需要旋转节点等操作，简化了更新过程，减小了填充成本。



(a)将初始化的 HIRB 树存储到 vORAM



(b)在 HIRB 中插入、删除 X 的过程

图 1　HIRB +vORAM 结构

Fig. 1　HIRB +vORAM structure

HIRB+vORAM 结构具有以下安全特性：

1)茫然性：可以隐藏客户端-服务器端通信数据和访问模式。

2)安全删除性和历史独立性：在面对灾难性攻击时，确保之前删除的数据、以前存在数据的事实或现有数据被访问的顺序不会泄露给攻击者。

3.3　伪随机函数

令 $Gen(1^\lambda)\in\{0,1\}^\lambda$ 为密钥生成函数，$G:\{0,1\}^\lambda\times\{0,1\}^\ell\to\{0,1\}^{\ell'}$ 为伪随机函数。如果对于所有的多项式时间敌手，满足 $|Pr[K\leftarrow Gen(1^\lambda);A^{G^K(\cdot)}(1^\lambda)=1]-Pr[A^{R(\cdot)}(1^\lambda)=1]|\leqslant negl(\lambda)$，则 G 是安全的。其中，$G_K(\cdot)$ 代表 $G(K,\cdot)$，$R:\{0,1\}^\ell\to\{0,1\}^{\ell'}$ 是真随机函数，$negl(\cdot)$ 为可忽略函数。

3.4　方案定义

图 2 给出了本文 DSE 方案的设计框架，主要由以下 3 个算法组成。

Setup(λ)：该算法输入安全参数 λ，输出 (K,σ,EDB)，其中 K 为客户端的密钥，σ 为客户端状态(包括 HIRB 树的根节点标识符、本地存储 stash 等)。EDB 为要发送到云服务器的(空的)加密数据库。

Search($K,q,\sigma;EDB$)：该算法用于搜索加密数据库，由客户端输入查询 q、密钥 K 以及本地状态，服务器端输入加密数据库 EDB，输出搜索结果 $DB(q)$。本文只考虑单关键字查询(即 $q=w$)，如果 $q\notin W$，则 $DB(q)$为空查询操作是由 vORAM 定义的 Lookup 函数实现，查询结束需要更新客户端状态 σ、密钥 K 以及加密数据库 EDB。

Update($K,op,in,\sigma;EDB$)：该算法是向加密数据库添加或者从加密数据库删除条目的协议。op 表示更新类型，可以是添加或者删除，in 表示输入由关键字和文件标识符组成 (w,id)对。同样，该算法也需要更新客户端状态和加密数据库。

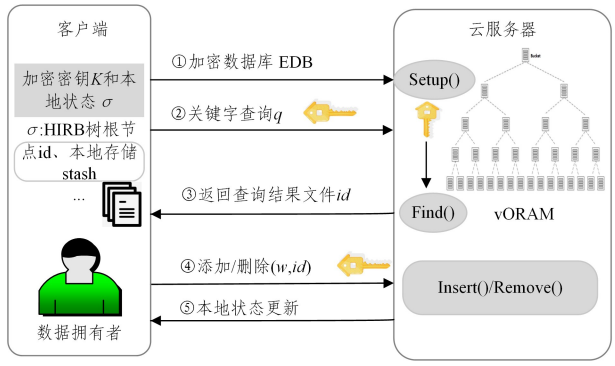


图 2　密文检索的系统架构

Fig. 2　System architecture of ciphertext retrieval

正确性：给定一个上述 DSE 方案 Σ，如果对于安全参数 λ 以及由 Setup(λ)生成的密钥 K 和加密数据库 EDB，密文查询(搜索和更新操作)的响应与相应的明文查询相同，则方案 Σ 是正确的。

3.5　安全模型

DSE 方案的安全性主要通过 $Real^{DSE}$ 和 $Ideal^{DSE}$ 两个游戏来制定，其中 $Real^{DSE}$ 的行为与 DSE 方案完全相同，$Ideal^{DSE}$ 则以泄漏函数 $L=(L_{Stp},L_{Srch},L_{Updt})$为输入，反映模拟器 S 的行为。L_{Stp}对应于初始化期间泄露给敌手服务器的信息，L_{Srch}对应于查询期间的泄漏，L_{Updt} 对应于更新期间的泄漏。两个游戏的正式定义如下。

$Real_A^{DSE}(\lambda,q)$：首先运行 Setup($\lambda,|DB|$)，得到加密数据库 EDB 发送给敌手 A。随后，A 发起一系列查询：搜索查询 q 或者是更新查询(op,in)。最后，A 返回一个字符 $b,b\in\{0,1\}$。

$Ideal_{A,S,L}^{DSE}(\lambda,q)$：模拟器 S 首先执行泄露函数 $L_{Stp}(\lambda,DB)$。对 A 发起一系列搜索查询 q 和更新查询(op,in)，S 分别利用泄露函数 $L_{Srch}(q)$和 $L_{Updt}(op,in)$输出结果返回给 A。最后，A 返回一个字符 $b,b\in\{0,1\}$。

定义 1　如果对于任何一个多项式时间敌手 A 发起的多项式数量查询 q，都存在一个多项式时间的模拟器 S 使得 $|Pr[Real_A^{DSE}(\lambda,q)=1]-Pr[Ideal_{A,S,L}^{DSE}(\lambda,q)=1]|\leqslant neal(\lambda)$，那么称 DSE 方案 Σ 对泄漏函数 L 自适应安全。

4　FBDSE-I 方案

本章详细描述了 Type-I 后向安全的 FBDSE-I 方案。尤其为了利用 vORAM 使 FBDSE-I 只泄露与特定关键字 w 相关的更新的数量。我们设计了一种新的存储方法，其主要思想是利用 3 个基于 vORAM 的茫然数据结构 ODS_{upd}、ODS_{src} 和 ODS_{cnt}来隐藏用户的访问模式。如图 3 所示，ODS_{upd}用来存储(w,id)对与其对应插入次数 i 的映射，ODS_{src}用来存储(w,i)与其对应文件标识符 id 的映射，ODS_{cnt}则用来记录 w 的总插入次数 i_w以及其当前包含 w 的所有文件的插入次数集合 S_w。

对于添加操作，用户先访问 ODS_{cnt}获得 w 当前的总插入

次数 i_w，再调用 *Insert* 函数将(w,id)对到插入次数 i_w 的映射添加到 ODS_{upd} 中，将(w,i_w)到文件标识符 id 的映射添加到 ODS_{src} 中。

对于删除操作，客户端则直接调用 *ODS* 提供的 *Remove* 函数依次删除 ODS_{upd} 中的(w,id)对到插入次数 i 的映射以及 ODS_{src} 中(w,i)到文件标识符 id 的映射，以上步骤结束后，客户端将插入次数 i 从 ODS_{cnt} 的 S_w 集合中删除。

对于搜索操作，客户端先访问 ODS_{cnt} 中的 S_w，得到当前包含 w 的所有文件 id 对应的插入次数，利用插入次数访问 ODS_{src} 得到对应文件标识符。

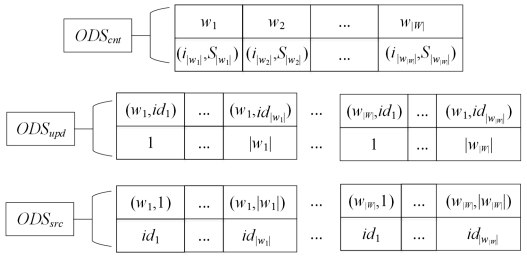


图 3 ODS_{upd}，ODS_{src} 和 ODS_{cnt} 的存储内容

Fig. 3 Storage contents of ODS_{upd}, ODS_{src} and ODS_{cnt}

由于 ODS 的茫然性、历史独立性以及安全删除性，以上搜索和更新的过程敌手服务器只能观察到一系列的 vORAM 路径和位置。根据不同操作访问的 ORAM 路径数量不同，服务器只能推断出更新的数量，从而达到 Type-I 后向安全。

FBDSE-I 方案由 3 种 PPT 算法 Setup，Update 和 Search 组成。

1)Setup。如算法 1 所示，客户端首先初始化 3 个茫然数据结构 ODS_{upd}，ODS_{src} 和 ODS_{cnt}（见算法 1 中的第 1－3 行），即高度为 H 的空树 T_1，T_2 和 T_3，其根节点标识符分别为 $rootID_1$，$rootID_2$ 和 $rootID_3$。ODS_{src} 维护映射$(w,i)\rightarrow id$，即在输入关键字和插入次数时，返回相应的文件标识符。ODS_{upd} 存储$(w,id)\rightarrow i$ 的映射，即在输入关键字和文件标识符时，输出相应的插入次数。ODS_{cnt} 为每个关键字存储总插入次数以及当前包含 w 的所有文件 id 的插入次数集合。因此，ODS_{upd}，ODS_{src} 和 ODS_{cnt} 的大小分别是 $|DB|$，$|DB|$ 和 $|W|$。*EDB* 由 3 个茫然数据结构组成，σ 由 ODS_{upd}，ODS_{src} 和 ODS_{cnt} 的状态参数组成。

算法 1 FBDSE-I Setup(λ, N)

输入：安全参数 λ，容量 N

输出：加密数据库 EDB，客户端状态 σ，密钥 K

1. $(T_1, rootID_1) \leftarrow ODS_{src}.Setup(1^\lambda, |DB|)$
2. $(T_2, rootID_2) \leftarrow ODS_{upd}.Setup(1^\lambda, |DB|)$
3. $(T_3, rootID_3) \leftarrow ODS_{cnt}.Setup(1^\lambda, |W|)$
4. $EDB \leftarrow \{T_1, T_2, T_3\}$
5. $\sigma \leftarrow (rootID_1, rootID_2, rootID_3)$
6. $K \leftarrow K_{ODS}$
7. 发送 EDB 至服务器

2)Update。如算法 2 所示，客户端首先访问 ODS_{upd} 得到(w,id)对应的插入次数 i，再针对添加和删除的情况分别执行。对于添加操作，客户端首先根据 i 的值判断之前是否添加过(w,id)。如果 $i=0$，则没有添加过，先将 ODS_{cnt} 中的 w 的总插入次数 i_w 的值更新为 i_{w+1}，再将集合 S_w 更新为 $S_w \cup i_w$（见算法 2 中的第 5－8 行）。更新 ODS_{cnt} 之后，客户端利用茫然数据结构提供的 *Insert* 函数分别向 ODS_{upd} 和 ODS_{src} 中插入映射$(w,id)\rightarrow i_w$ 和映射$(w,i_w)\rightarrow id$（见算法 2 中的第 10－13 行）。对于删除操作，客户端首先根据 i 的值判断数据库中是否存在(w,id)，如果 $i>0$ 则存在，那么利用茫然数据结构提供的 *Remove* 函数直接删除 ODS_{upd} 和 ODS_{src} 中的映射$(w,id)\rightarrow i_w$ 和映射$(w,i_w)\rightarrow id$（见算法 2 中的第 16－17 行），最后更新 ODS_{cnt} 中 S_w 的值（见算法 2 中的第 18－20 行）。

算法 2 FBDSE-I Update(K, op, (w,id), σ; EDB)

输入：加密密钥 K，操作类型 op，更新数据(w，id)，加密数据库 EDB，客户端状态 σ

输出：更新后的加密数据库 EDB，客户端状态 σ

1. in＝(w，id)
2. $(rootID_2, i) \leftarrow ODS_{upd}.Lookup(in, rootID_2)$
3. IFop＝add THEN//添加操作
4. IF i＝NULL THEN//没有添加过(w，id)
5. $(i_w, S_w) \leftarrow ODS_{cnt}.remove(w, rootID_3)$
6. $i_w = i_w + 1$
7. $S_w = S_w \cup \{i_w\}$
8. $rootID_3 \leftarrow ODS_{cnt}.insert(w, (i_w, S_w), rootID_3)$
9. END IF
10. data＝((w，id)，i_w)
11. $rootID_2 \leftarrow ODS_{upd}.Insert(data, rootID_2)$
12. data＝((w，i_w)，id)
13. $rootID1 \leftarrow ODS_{src}.Insert(data, rootID_1)$
14. ELSE IF op＝del THEN//删除操作
15. IF i ＞ 0 THEN //数据库存在(w，id)
16. $(rootID_2, i) \leftarrow ODS_{upd}.remove(in, rootID_2)$
17. $(rootID_1, id) \leftarrow ODSsrc.remove(w, i, rootID_1)$
18. $(i_w, S_w) \leftarrow ODScnt.remove(w, rootID_3)$
19. $S_w = S_w - \{i\}$
20. $rootID_3 \leftarrow ODS_{cnt}.insert(w, (i_w, S_w), rootID_3)$
21. END IF
22. END IF

3)Search。如算法 3 所示，对于搜索操作客户端可以通过每个(w,id)的插入次数 i 在 ODS_{src} 进行查询。在查询之前首先需要在 ODS_{cnt} 得到 w 包含所有 id 的插入次数集合 S_w（见算法 3 中的第 2 行）。假设 $|S_w|=n_w$，客户端需要对 ODS_{src} 执行 n_w 次访问得到搜索结果集合（见算法 3 中的第 3－6 行）。由于 HIRB＋vORAM 支持并行访问，这 n_w 次访问可批量执行，使搜索查询的时间复杂度为 $O(n_w \log_\beta N \log N)$，即实现准最优搜索时间。

算法 3 FBDSE-I Search(K，w，σ；EDB)

输入：密钥 K，关键词 w，加密数据库 EDB，客户端状态 σ

输出：搜索结果集合 RList

1. RList＝空集合
2. $(rootID_3, i_w, S_w) \leftarrow ODS_{cnt}.Lookup(w, rootID_3)$
3. FOR each i in S_w do //并行执行
4. $(rootID_2, id) \leftarrow ODS_{src}.Lookup((w,i), rootID_2)$
5. RList＝RList∪{id}
6. END FOR
7. returnRList

5 FBDSE-II 方案

FBDSE-II 方案是 FBDSE-I 的改进方案。虽然 FBDSE-I 方案可以达到准最优搜索时间的检索效率，但是都将检索结

果 $DB(w)$所需的 n_w 访问替换为 vORAM 访问，影响搜索效率。而 FBDSE-II 方案相比 FBDSE-I 方案减少了搜索过程中的通信和计算开销，同时实现了 Type-III 后向安全。FBDSE-II 方案的主要思想是使用映射字典 Dict 代替茫然数据结构 ODS_{src}，解耦茫然原语与检索结果，减少查询时 vORAM 的访问，同时保证隐私数据的更新安全。

FBDSE-II 方案同样由 Setup，Update 和 Search 3 个 PPT 算法组成。与 FBDSE-I 方案不同之处在于搜索期间只进行一次 vORAM 访问，可提高查询效率。

1）Setup。如算法 4 所示，客户端首先初始化一个空的映射 Dict 以及两个茫然数据结构 ODS_{upd} 和 ODS_{cnt}（见算法 4 中的第 2－4 行）。其中，ODS_{upd} 和 ODS_{cnt} 的大小、存储内容及其作用与方案 FBDSE-I 相同。EDB 由两个茫然数据结构和一个加密映射字典组成。σ 由 ODS_{upd} 和 ODS_{cnt} 的状态参数组成，密钥 K 由映射字典 $Dict$ 的加密密钥 K_{Dict} 和茫然数据结构的密钥 K_{ODS} 组成。

算法 4　FBDSE-II Setup(λ，N)

输入：安全参数 λ，容量 N

输出：加密数据库 EDB，客户端存储参数 σ

1. K_{Dict} ← Gen(1^{λ})
2. Dict←空映射
3. (T_1，$rootID_1$)←ODS_{upd}. Setup(1^{λ}，|DB|)
4. (T_2，$rootID_2$)←ODS_{cnt}. Setup(1^{λ}，|W|)
5. EDB←{T_1，T_2，Dict}
6. σ←($rootID_1$，$rootID_2$)
7. K←(K_{ODS}，K_{Dict})
8. 发送 EDB 至服务器

2）Update。算法 5 为更新算法，客户端同样先访问 ODS_{upd} 得到 (w,id) 对应的插入次数 i，再针对添加和删除的情况分别执行。对于添加操作，与 *FBDSE-I* 方案的不同之处在于对映射 $Dict$ 的更新。客户端为 (w,id) 运行两次随机函数 G（密钥为 K_{Dict}）生成 $G_{KDict}(w,i_w \| 0)$ 和 $G_{KDict}(w,i_w \| 1)$，进而得到 $(addr,val)$ 对，并将其发送到服务器端（见算法 5 中的第 11－13 行）。服务器端收到后将之存储在 $Dict$ 中（见算法 5 中的第 22 行）。对于删除操作，FBDSE-II 方案无需更改映射 $Dict$，对于 ODS_{upd} 和 ODS_{cnt} 的操作与 FBDSE-I 相同。

算法 5　FBDSE-II Update(K，op，(w，id)，σ；EDB)

输入：加密密钥 K，操作类型 op，更新数据(w，id)，加密数据库 EDB，客户端状态 σ

输出：更新后的加密数据库 EDB，客户端状态 σ

1. ($rootID_2$，i_w)←ODS_{upd}. Lookup((w，id)，$rootID_2$)
2. IF op＝add THEN//添加操作
3. IF i＝NULL THEN//没有添加过(w，id)
4. 　　($rootID_2$，i_w，S_w)←ODS_{cnt}. remove(w，$rootID_3$)
5. i_w＝i_w＋1
6. S_w＝S_w∪{i_w}
7. $rootID_2$←ODS_{cnt}. insert(w，(i_w，S_w)，$rootID_2$)
8. END IF
9. data＝((w，id)，i_w)
10. $rootID_2$←ODS_{upd}. Insert(data，$rootID_2$)
11. addr＝G_{KDict}(w，i_w ‖ 0)
12. val＝id⊕G_{KDict}(w，i_w ‖ 1)
13. 发送(addr，val) 至服务器
14. ELSE IF op＝del THEN//删除操作
15. 　IF i ＞ 0 THEN //数据库存在(w，id)
16. 　($rootID_1$，i_w)←ODS_{upd}. remove((w，id)，$rootID_1$)
17. 　(i_w，S_w)←ODS_{cnt}. remove(w，$rootID_2$)
18. S_w＝S_w－{i}
19. $rootID_2$←ODS_{cnt}. insert(w，(i_w，S_w)，$rootID_2$)
20. END IF
21. END IF

服务器端：

22. 设置 Dict[addr] ＝val

3）Search。如算法 6 所示，查询之前，客户端从 ODS_{cnt} 得到要查询次数集合 S_w，对集合 S_w 中每一项生成其对应的 $addr$（即 $G_{KDict}(w,i \| 0)$），并发送到服务器进行查询。服务器收到之后，在 $Dict$ 中进行查询，得到每个 $addr$ 对应的 val，返回到客户端。客户端利用存储在本地的密钥 K_{Dict} 对每个 val 进行解密，得到最终的搜索结果 $RList$。可以看出，在搜索过程中，客户端仅使用了一次茫然原语，即访问 ODS_{cnt} 得到集合 S_w。

算法 6　FBDSE-II Search(K，w，σ；EDB)

输入：密钥 K，关键词 w，加密数据库 EDB，客户端状态 σ

输出：搜索结果集合 RList

客户端：

1. T_d＝空集合
2. ($rootID_2$，i_w，S_w)←ODS_{cnt}. Lookup(w，$rootID_2$)
3. FOR each i in S_w do //并行执行
4. 　　t_i＝G_{KDict}(w，i ‖ 0)
5. 　　T_d＝T_d∪{t_i}
6. END FOR
7. 发送 T_d 到服务器

服务器端：

8. L_w＝空集合
9. FOR each i in T_d. size do
10. 　L_w＝L_w∪Dict[T_d[i]]
11. END FOR
12. 发送 L_w 到客户端

客户端：

13. RList＝空集合
14. FOR i＝1 to L_w. size do
15. 　id＝L_w[i]⊕G_{KDict}(w，i ‖ 1)
16. 　RList＝RList∪{id}
17. END FOR
18. return RList

6　安全定义与安全分析

6.1　安全定义

在更新期间，如果敌手无法将正发生的更新与之前的操作建立联系，则 DSE 方案满足前向安全。

定义 2（前向安全）[5]　对于 L-自适应安全的 DSE 方案 Σ（仅支持添加或删除单个关键字/文档对），如果更新泄漏函数 L_{Updt} 可以写为 $L_{Updt}(op,w,id)=L'_{Updt}(op,id)$，则 Σ 是前向安全的。其中，L' 是无状态函数，op 是插入或删除，id 是文件标识符。

后向安全限制服务器通过对 w 的搜索查询可以学习到 w 的更新信息。如果当 (w,id) 对被添加到数据库中随后被删除时，在 w 上的后续搜索查询不会显示 id，那么 SSE 方案就是后

向安全的。以下给出3种不同类型后向安全的正式定义。

定义3(后向安全)[5] L-自适应安全的DSE方案Σ满足如下条件。

Type-I后向安全:如果更新泄露函数$L_{\text{Updt}}(op,w,id)=L'(op)$,搜索泄露函数$L_{\text{Srch}}(w)=L''(TimeDB(w),a_w)$。

Type-II后向安全:如果更新泄露函数$L_{\text{Updt}}(op,w,id)=L'(op,w)$,搜索泄露函数$L_{\text{Srch}}(w)=L''(TimeDB(w),Updates(w))$。

Type-III后向安全:如果更新函数$L_{\text{Updt}}(op,w,id)=L'(op,w)$,搜索泄露函数$L_{\text{Srch}}(w)=L''(TimeDB(w),DelHist(w))$。

其中,a_w是关键字w的更新次数,L'和L''是无状态函数,$TimeDB(w)$是所有匹配w的文档的列表(不包括已删除的文档),以及插入到数据库中的时间戳,$Updates(w)$是w上更新的时间戳列表,$DelHist(w)$是w的所有删除操作的时间戳列表,以及插入这些删除项的时间戳。对于关键字w的$TimeDB(w)$,$Updates(w)$和$DelHist(w)$3个函数的形式定义如下。

1) $TimeDB(w)=\{(u,id)\mid(u,add,(w,id))\in Q$且$\forall u',(u',del,(w,id))\notin Q\}$。

2) $Updates(w)=\{u\mid(u,add,(w,id))$或者$(u,del,(w,id))\in Q\}$。

3) $DelHist(w)=\{(u^{\text{add}},u^{\text{del}})\mid\exists\ id:(u^{\text{del}},del,(w,id))\in Q$且$(u^{\text{add}},add,(w,id))\in Q\}$。

其中,Q是包含所有的搜索查询和更新查询的列表,搜索查询的形式是(u,w),更新查询的形式是$(u,op,(w,id))$,u是查询时间戳,$op=add/del$,id是修改后的文件。

为了明确这些概念之间的区别,将使用一个例子来说明。假设有以下系列更新操作,(add,id_1,w_1),(add,id_1,w_2),(add,id_2,w_1),(del,id_1,w_1),(add,id_3,w_2)。当查询关键词w_1时,Type-I方案泄漏id_1,以及其插入时间$time_1$,此外还泄露了w_1的更新次数为3次;Type-II方案额外泄露了w_1的更新发生在$time_1$,$time_2$和$time_3$;Type-III方案在Type-I和Type-II方案所泄露信息的基础上还额外泄露了w_1在$time_1$添加的索引在$time_3$被移除。

6.2 安全分析

6.2.1 FBDSE-I方案

由于茫然数据结构的使用,FBDSE-I方案的安全分析非常直接。在整个协议执行过程中,服务器对每次查询只能观察到固定数量随机选择的vORAM位置。对于更新查询,由于虚拟填充,服务器只能观察到$5\times2\times\log_\beta N$个此类位置,服务器甚至不知道执行的操作类型(添加或删除),即更新泄漏为空。这意味着FBDSE-I方案具备前向安全。对于搜索查询,服务器观察到$n_w\times5\times2\times\log_\beta N$个此类位置,即只泄露了$n_w$的值。特别地,由于每次更新后访问的ORAM条目都会重新映射到新选择的随机位置,因此服务器不可能将搜索期间访问的位置与特定的更新操作相匹配。这意味着FBDSE-I方案具有最小的泄漏,即Type-I后向安全。下面给出形式化证明。

定理1 假设ODS_{src},ODS_{upd}和ODS_{cnt}是由文献[8]中的茫然数据结构实例化,那么FBDSE-I方案是具备前向安全和Type-I后向安全的自适应安全DSE方案,即其更新泄露$L_{\text{Updt}}(op,w,id)=op$且搜索泄露函数$L_{\text{Srch}}(w)=TimeDB(w)$。

证明:假设$(o_1,o_2,\cdots,o_p)$为执行p个操作序列产生的访问内存地址序列。在Real^{SSE}游戏中,服务器对于FBDSE-I观察到的内容由vORAM位置和加密路径组成。具体地,如果第i个操作是添加操作,则o_i由$\lfloor5\times2\times\log_\beta N\rfloor$个vORAM位置和加密路径组成。这是因为每个HIRB树操作需要$2\times\log_\beta N$个vORAM操作,添加操作需要进行5次HIRB树操作。由于删除操作也要进行5次HIRB树操作,如果第i个操作是删除操作,o_i也由$\lfloor5\times2\times\log_\beta N\rfloor$个vORAM位置和加密路径组成。此外,如果第$i$个操作搜索结果大小为$n_w$的查询操作,$o_i$由集合$o_{i,j}$组成,其中$j=1,\cdots,\lfloor\log_\beta N\rfloor$,$|o_{i,j}|=\min\{2^j,n_w\}$。每个$o_{i,j}$由相同的vORAM位置和加密路径组成。模拟器$S$的描述如下:

S-Init$(st_S,L_{\text{Stp}}(\lambda,N))$:创建3个空的HIRB树,每个节点都填充上加密的0。令$EDB_0\leftarrow(T_1,T_2,T_3)$,将$\text{EDB}_0$发送给服务器,并设置$st_S$为空。

S-Update$(st_S,L_{\text{Updt}}(w_k),EDB_{k-1})$:无论是添加操作还是删除操作,都令$d=\lfloor5\times2\times\log_\beta N\rfloor$,随后模拟器$S$生成$d$个随机位置逐个发送到服务器,最后设$P_1$,$P_2$和$P_3$分别为由$T_1$,$T_2$和$T_3$的随机位置生成的路径,模拟器$S$为这些路径计算所有新加密的0,并将它们发送到服务器。

S-Search$(st_S,L_{\text{Srch}}(w_k),EDB_{k-1})$:模拟器$S$从当前数据库检索$n_w$个数据项。对于$j=1,\cdots,\lfloor\log_\beta N\rfloor$,$S$生成$\min\{2^j,n_w\}$个随机vORAM位置表示为$o_{k,j}$。随后模拟器$S$逐个将$o_{k,1},\cdots,o_{k-\lfloor\log_\beta N\rfloor}$发送到服务器。设$P_1$,$P_2$和$P_3$分别为由$T_1$,$T_2$和$T_3$的随机位置生成的路径,模拟器$S$为这些路径计算所有新加密的0,并将它们发送到服务器。令R_w是与搜索关键字对应的文档标识符集合,是由$TimeDB(w)$推导而来。最后,S将R_w发送到服务器。

通过以上分析,针对模拟器S模拟的$\text{Ideal}^{\text{DSE}}$游戏,其ORAM位置生成方式与$\text{Real}^{\text{DSE}}$游戏相同,使用的加密方案是语义安全的,并且进行了同等数量的ORAM访问,文档标识符数量也是相同的,因此$\text{Ideal}^{\text{DSE}}$游戏生成的访问内存地址序列与$\text{Real}^{\text{DSE}}$游戏过程中产生的不可区分。证毕。

6.2.2 FBDSE-II方案

FBDSE-II方案是前向安全的,因为在更新期间服务器观察以下两种类型的访问:1)固定数量随机选择的vORAM位置,敌手无法区分;2)$(addr,val)$对。由于G的伪随机性,以及PRF在每次更新时都包含了不同的输入,敌手观察到的$(addr,val)$这两个值与随机值是不可区分的。此外,如果发生更新,查询令牌T_d将发生变化,服务器无法将未来的更新与搜索之前的更新联系起来,即满足前向安全。对于后向安全,同样由于G的伪随机性以及PRF在每次更新时都包含了不同的输入,服务器无法学习到两次查询期间添加了随后又删除的文件包含关键字w,但是服务器可能会推测出某个删除操作抵消了之前的插入操作,即FBDSE-II方案具备Type-III后向安全。以下给出正式的形式化证明。

定理2 假设ODS_{cnt},ODS_{upd}是由文献[8]中的茫然数据结构实例化,函数G为语义安全的PRF,那么FBDSE-II方案是具备前向安全和Type-III后向安全的自适应安全DSE方

案，即其更新泄露 $L_{\text{Updt}}(op,w,id)=op$ 且搜索泄露函数 $L_{\text{Srch}}(w)=(Updates(w),DelHist(w))$。

证明：通过定义以下系列游戏来证明方案 FBDSE-II 的安全性。

Game-0：为 2.5 节定义的 Real^{DSE} 游戏。

Game-1：除了在初始化过程中，调用大小分别为 $|W|$ 和 $|N|$ 的 ODS 模拟器替代 ODS 初始化，以及未来所有 ODS 访问都是通过调用相应的访问模拟器来模拟以外，Game-1 与 Game-0 相同。由于茫然数据结构的安全性，Game-1 与 Game-0 不可区分。

Game-2：除了更新过程，对于 $b=0$ 和 1，$G_K(w,i_w\|b)$ 生成的值被替换成从 G 的范围内均匀随机抽样的随机值以外，Game-2 与 Game-1 相同。设 Q 是具有 q 个查询的列表，如果第 i 个操作是插入操作，对于 $Q(j)=(addr,val,w,id)$，$j=1,\cdots,q$，其存储与输入 (w,id) 操作采样的随机值，否则存储 NULL。在搜索 w 过程中，Game-2 遍历列表 Q 标记出匹配 w 的项，发送给服务器对应的 $addr$ 值，并接收查询结果。然后再次扫描 Q 来推断当前包含 w 的文档集 R_w，并将 R_w 发送到服务器。在更新过程中，由于 PRF 输入从不相同以及 PRF 的安全性，Game-2 与 Game-1 不可区分。

Game-3：这与 Game-2 相同，但在更新时，客户端接收的是 op 而不是 (op,w,id)，以及在搜索时接收的是 $Updates(w)$ 和 $DelHist(w)$ 而不是 w，其按照如下方式填充列表 $TokensD$。对于时间戳 t 的搜索，客户端首先检查输入的更新历史 $Updates(w)$ 是否为先前发生在时间戳 i 的搜索的扩展。如果是，意味着时间戳 t 和 i 搜索的是相同关键字，则从 $TokensD$ 检索 T_d 作为最新的条目；否则，选择新的随机令牌 $\dot{T}_d$ 并将其填加到 $TokensD(t)$。因为在 Game-3 搜索过程中，更新后生成的 T_d 不相同，且 $id\oplus r$ 中的 r 也是随机均匀分布的，则 Game-3 与 Game-2 是不可区分的，即 Game-3 与 Game-0 是不可区分的。此外，Game-3 中所描述的本质是 2.5 节定义的 $\text{Ideal}^{\text{DSE}}$ 游戏的模拟器代码，因为其只将定理 2 中规定的泄漏作为输入，泄漏函数对应的值与在该游戏中计算的值相同。综上，$\text{Ideal}^{\text{DSE}}$ 游戏与在 Game-0 是不可区分的。证毕。

7　性能分析

本节主要对 FBDSE-I 和 FBDSE-II 方案的查询、更新复杂度以及存储开销进行分析。表 2 列出了与已有的基于 ORAM 的前向和后向安全 DSE 方案进行性能对比的结果。

FBDSE-I 和 FBDSE-II 方案的性能依赖于 HIRB＋vORAM 结构，对于具有 N 个节点的 HIRB 树，每次访问需要检索 $O(\log_\beta N)$ 个 HIRB 节点，每个节点都需要检索长度为 $O(\log N)$ 的 ORAM 路径，因此每次访问的时间复杂度为 $O(\log_\beta N\log N)$，其中 β 为 HIRB 树的分支因子，N 为节点个数。对于搜索结果大小为 n_w 的查询操作，FBDSE-I 需要并行执行 n_w 次访问，查询时间复杂度为 $O(n_w\log_\beta N\log N)$，FBDSE-II 由于只执行一次 HIRB 树操作，其时间复杂度为 $O(n_w+\log_\beta N\log N)$。对于更新操作，FBDSE-I 和 FBDSE-II 方案无论是添加还是删除操作，都会执行固定数量 HIRB 操作，如表 2 所列，更新时新时间复杂度都为 $O(\log_\beta N\log N)$。而 Moneta，ORION，HORUS 和 SD_d 由于使用了 TWORAM 结构以及 AVL＋ORAM 结构，具有更高的查询和更新时间复杂度，此外这些方案还有更高的查询通信开销，很难满足用户的查询需求。而 FBDSE-I 和 FBDSE-II 方案具有更低查询通信开销，且随着数据集以及 β 的增大，性能优势越明显。对于存储开销，用户希望客户端存储尽可能少的数据，通过方案的详细描述可知，FBDSE-I 和 FBDSE-II 本地存储内容为加密密钥、vORAM 的状态，造成的客户端存储开销为 O(1)。在服务器端，FBDSE-I 方案构造了 3 个容量分别为 $|DB|$，$|DB|$ 和 $|W|$ 的茫然数据结构 ODS_{src}，ODS_{upd} 和 ODS_{cnt}。FBDSE-II 方案需要构造两个容量分别为 $|DB|$ 和 $|W|$ 的茫然数据结构 ODS_{upd}，ODS_{cnt} 以及容量为 a_w 的 Dict 映射词典。因此，服务器端开销为 $O(N)$。

表 2　基于 ORAM 的前向和后向安全 DSE 方案性能对比表

Table 2　Performance comparison of ORAM-based forward and backward secure DSE schemes

方案	查询计算开销	更新计算开销	查询通信开销	客户端存储开销	前向安全	后向安全	ORAM 结构类型
Moneta[5]	$\tilde{O}(a_w\log N+\log^3 N)$	$\tilde{O}(\log^2 N)$	$\tilde{O}(a_w\log N+\log^3 N)$	O(1)	√	Type-I	TWORAM
ORION[6]	$O(n_w\log^2 N)$	$O(\log^2 N)$	$O(n_w\log^2 N)$	O(1)	√	Type-I	AVL＋Path-ORAM
HORUS[6]	$O(n_w\log d_w\log N)$	$O(\log^2 N)$	$O(n_w\log d_w\log N)$	O(1)	√	Type-III	AVL＋Path-ORAM
SDd[7]	$O(a_w+\log^2 N)$	$O(\log^3 N)$	$O(a_w+\log^2 N)$	O(1)	√	Type-II	AVL＋Path-ORAM
FBDSE-I	$O(n_w\log_\beta N\log N)$	$O(\log_\beta N\log N)$	$O(n_w\log_\beta N\log N)$	O(1)	√	Type-I	HIRB＋vORAM
FBDSE-II	$O(n_w+\log_\beta N\log N)$	$O(\log_\beta N\log N)$	$O(n_w+\log_\beta N\log N)$	O(1)	√	Type-III	HIRB＋vORAM

8　仿真分析

本文实验原型系统的开发和测试环境是基于 windows10（64 位）系统，具体硬件配置是 intel(R) Core(TM) i5-4570 (3.20 GHz)处理器，16 GB 内存。利用文献[8]中的 vORAM 框架[1)]，采用 Python 编程语言，进行了 FBDSE-I 和 FBDSE-II 方案的原型实现（可提供）。其中，伪随机函数的实现采用了 256b 密钥的对称加密算法 AES，通过 Crypto 库实例化方案的加密操作。仿真实验主要测试了 FBDSE-I 和 FBDSE-II 方案搜索和更新操作的耗时，多次执行每个实验以获得平均执行时间，并与同等后向安全水平的 ORION 和 HORUS 方案进行实验对比，以下为实验结果分析。

如图 4 所示，对于搜索操作，将数据集的大小 $|DB|$ 从 10^1 增加到 10^5 分别进行搜索，可以看出 FBDSE-I 和 FBDSE-II 方案的效率相对较高，其中 FBDSE-II 方案在搜索过程中客户端仅进行一次 vORAM 的访问，效率最高。此外，由测试结果可以看出，数据集 DB 的规模越大，FBDSE-I 和 FBDSE-II 方案的优势更明显。如测试到 10^5 时，由于使用 AVL＋Path-

1) https://github.com/dsroche/obliv

ORAM结构，我们测出ORION的查询时间为4 846.62 s，HORUS的查询时间为13 743.5 s，这是非常难以接受的。而FBDSE-I和FBDSE-II方案查询时间分别为97.089 s和3.817 s，优势更加显著。

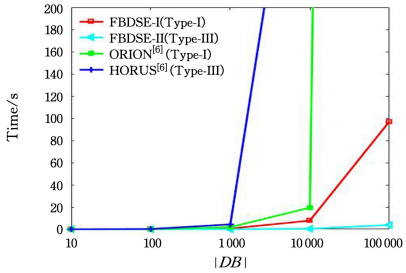


图4 查询时间对比

Fig. 4 Query time comparison

对于更新操作，我们在数据集合大小$|DB|$分别为10^1，10^2，…，10^5时，测试了插入和删除一个(w, id)对所消耗的时间。如图5所示，给出插入操作的实验对比结果，由于HIRB+vORA相对AVL+Path-ORAM降低了在插入数据时的通信开销以及虚拟填充开销，FBDSE-I和FBDSE-II方案相对同级别的DSE方案提高了添加更新效率。图6则给出了删除操作的实验对比结果，同样FBDSE-I和FBDSE-II方案效率更高，且相比添加操作效率提升的幅度更大一些，这是因为除了上述提到的HIRB+vORAM结构，降低了通信开销以及虚拟填充开销的特点以外，该结构提供的Remove函数可以直接删除关键词/文件标识符对，简化了DSE更新过程的原因。

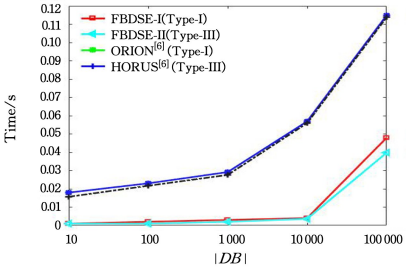


图5 插入操作性能对比

Fig. 5 Insert operation performance comparison

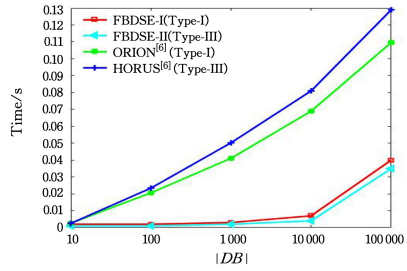


图6 删除操作性能对比

Fig. 6 Delete operation performance comparison

此外，我们在数据集合大小$|DB|$分别为10^5时对比了本文方案和ORION，HORUS以及SDd的通信开销。当返回结果集的大小为50时，FBDSE-I和FBDSE-II方案传输所有的数据块分别需要860.1 kB和623.8 kB，而ORION，HORUS以及SDd则分别需要17.5 MB，5.9 MB以及792.5 kB，可见由于引进HIRB+vORAM有效减少了通信开销。

结束语 本文基于vORAM构造了两个DSE方案。由于引入的茫然数据结构具有茫然性、历史独立性和安全删除性，FBDSE-I和FBDSE-II方案实现了关键字/文件标识符对的直接、安全的删除，简化了更新过程，减少搜索和更新期间的通信开销，提高了方案性能。此外，两个方案被证明在具备前向安全的前提下，分别满足Type-I和Type-III后向安全。仿真结果表明，本文的两个方案相对于同等安全级别的方案具有更好的查询和更新性能。由于现有的前向和后向安全的DSE方案大多只支持单用户，未来将工作扩展到多用户、多服务器环境。

参 考 文 献

[1] DU R Z, LIU N, LI M Y, et al. Block Verifiable Dynamic Searchable Encryption Using Redactable Blockchain[J]. Journal of Information Security and Applications, 2023, 75: 103504.

[2] SONG D X, WAGNER D, PERRIG A. Practical Techniques for Searches on Encrypted Data[C]// Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000: IEEE, 2000: 44-55.

[3] PAPAMANTHOU C, STEFANOV E, SHI E. Practical Dynamic Searchable Encryption with Small Leakage[C]// Proceedings of the 21st Annual Network and Distributed System Security Symposium. San Diego: ISOC, 2014: 23-26.

[4] ZHANG Y, KATZ J, PAPAMANTHOU C. All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption[C]// Proceedings of the 25th USENIX Conference on Security Symposium. Berkeley: USENIX Association, 2016: 707-720.

[5] BOST R, MINAUD B, OHRIMENKO O. Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives[C]// Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 1465-1482.

[6] GHAREH CHAMANI J, PAPADOPOULOS D, PAPAMANTHOU C, et al. New Constructions for Forward and Backward Private Symmetric Searchable Encryption[C]// Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 1038-1055.

[7] DEMERTZIS I, CHAMANI J G, PAPADOPOULOS D, et al. Dynamic Searchable Encryption with Small Client Storage[C]// Proceedings of the 27th Annual Network and Distributed System Security Symposium. San Diego: ISOC, 2020: 1-18.

[8] ROCHE D S, AVIV A, CHOI S G. A Practical Oblivious Map Data Structure with Secure Deletion and History Independence [C]// 2016 IEEE Symposium on Security and Privacy (SP). IEEE, 2016: 178-197.

[9] LIU Z L, LI T, LI P, et al. Verifiable Searchable Encryption with Aggregate Keys for Data Sharing System[J]. Future Generation Computer Systems, 2018, 78: 778-788.

[10] CHENG R, YAN J B, GUAN C W, et al. Verifiable Searchable Symmetric Encryption from Indistinguishability Obfuscation [C]// Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, 2015: 621-626.

[11] DU R Z, WANG Y, LI M Y. Dynamic Ciphertext Retrieval Scheme with Two-Way Verification[J]. Journal of Computer Research and Development, 2022, 59(11): 2635-2647.

[12] AMORIM I，COSTA I. Leveraging Searchable Encryption through Homomorphic Encryption：A Comprehensive Analysis [J]. Mathematics，2023，11(13)：2948.

[13] KAMARA S，PAPAMANTHOU C. Parallel and Dynamic Searchable Symmetric Encryption[C]// Financial Cryptography and Data Security：17th International Conference. Okinawa，Japan：Springer Berlin Heidelberg，2013：258-274.

[14] BOST R. Σοφος：Forward Secure Searchable Encryption[C]// Proceedings of the 23rd Conference on Computer and Communications Security. New York：Association for Computing Machinery，2016：1143-1154.

[15] LAI R W F，CHOW S S M. Forward-Secure Searchable Encryption on Labeled Bipartite Graphs[C]// International Conference on Applied Cryptography and Network Security. Cham：Springer International Publishing，2017：478-497.

[16] KIM K S，KIM M，LEE D，et al. Forward Se cure Dynamic Searchable Symmetric Encryption with Efficient Updates[C]// Proceedings of the 24th Conference on Computer and Communications Security. New York：Association for Computing Machinery，2017：1449-1463.

[17] LI M Y，JIA C F，DU R Z，et al. DSE-RB：A Privacy-Preserving Dynamic Searchable Encryption Framework on Redactable Blockchain[J]. IEEE Transactions on Cloud Computing，2023，11(3)：2856-2872.

[18] SUN S F，YUAN X L，LIU J K，et al. Practical Backward-Secure Searchable Encryption from Symmetric Puncturable Encryption [C]// Proceedings of the 25th Conference on Computer and Communications Security，New York：Association for Computing Machinery，2018：763-780.

[19] ZUO C，SUN S F，LIU J K，et al. Dynamic Searchable Symmetric Encryption with Forward and Stronger Backward Privacy [C]// Proceedings of the 24th European Symposium on Research in Computer Security. Berlin：Springer，2019：283-303.

[20] LI M Y，JIA C F，DU R Z，et al. Forward and Backward Secure Searchable Encryption Scheme Supporting Conjunctive Queries Over Bipartite Graphs[J]. IEEE Transaction on Cloud Computing，2023，11(1)：1091-1102.

[21] DU R Z，ZHANG Y Q，LI M Y. Efficient Dynamic Searchable Encryption Scheme for Conjunctive Queries Based on Bidirectional Index[J]. Journal on Communications，2022，43(5)：123-132.

[22] ZHANG H J，ZENG S K，YANG J L. Backward Private Dynamic Searchable Encryption with Update Pattern[J]. Information Science，2023，624：1-19.

SHAO Tong，born in 1993，master，engineer. His main research interests include network security and information security.