基于双向索引的高效连接关键字查询动态可搜索加密方案

杜瑞忠[1,2]，张玉晴[1]，李明月[3]

（1. 河北大学网络空间安全与计算机学院，河北 保定 071000；
2. 河北省高可信信息系统重点实验室，河北 保定 071000；3. 南开大学计算机学院，天津 300071）

摘 要：为了解决现有动态可搜索加密方案更新过程操作复杂、信息泄露以及查询方式单一等问题，提出了一种前向安全和后向安全的高效连接关键字查询动态可搜索加密方案——BPC-DSSE 方案。该方案利用位图索引构建了双向索引结构来简化动态更新过程，并通过具有加法同态性质的对称加密隐藏访问模式。同时，由于添加和删除操作均通过模加法完成，可通过隐藏更新类型减少更新过程的泄露。此外，为了解决现有方案查询方式不灵活的问题，引入内积匹配算法实现了高效的连接关键字查询。安全分析表明，BPC-DSSE 方案实现了前向安全以及 Type-Ⅰ的后向安全。仿真结果表明，相对于其他连接关键字查询的方案，BPC-DSSE 方案具有更高的更新、查询效率。

关键词：动态对称可搜索加密；连接关键字查询；前向安全；后向安全

中图分类号：TP309

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2022099

Efficient dynamic searchable encryption scheme for conjunctive queries based on bidirectional index

DU Ruizhong[1,2], ZHANG Yuqing[1], LI Mingyue[3]

1. School of Cyberspace Security and Computer Science, Hebei University, Baoding 071000, China
2. Hebei Provincial Key Laboratory of High Trusted Information System, Baoding 071000, China
3.Computer College, Nankai University, Tianjin 300071, China

Abstract: To solve the problems of complicated operation, information leakage, and inflexible query method in the update process of existing dynamic searchable encryption scheme, an efficient dynamic searchable encryption scheme (named BPC-DSSE) for conjunctive query with forward and backward privacy was proposed. A bitmap index was used to construct a bidirectional index structure to simplify the dynamic update process, and the access pattern was hidden through symmetric encryption with homomorphic addition. At the same time, since the addition and deletion operations were completed by modulo addition, the leakage of the update process could be reduced by hiding the update type. Security analysis shows that the BPC-DSSE scheme achieves forward and Type-Ⅰ backward privacy. The simulation results show that the BPC-DSSE scheme has higher update and retrieval efficiency than other conjunctive query schemes.

Keywords: dynamic searchable symmetric encryption, conjunctive query, forward privacy, backward privacy

0 引言

云计算的出现允许个人和组织将大量数据的存储和处理外包给第三方服务器。然而，这导致了隐私问题，用户通常不相信服务提供者会尊重其数据的机密性[1]。这种信任的缺乏往往会受到来自内部恶意用户和外部攻击者的威胁。为了解决这个问题，Song 等[2]首次提出了可搜索加密（SE, searchable

收稿日期：2021−08−16；**修回日期：**2021−11−17

通信作者：张玉晴，yqzhang@hbu.edu.cn

基金项目：国家自然科学基金资助项目（No.61972073）；河北省自然科学基金重点资助项目（No.F2019201290）；河北省自然科学基金资助项目（No.F2018201153）

Foundation Items: The National Natural Science Foundation of China (No.61972073), The Key Program of Natural Science Foundation of Hebei Province (No.F2019201290), The Natural Science Foundation of Hebei Province (No.F2018201153)

encryption）技术。可搜索加密分为对称可搜索加密（SSE, searchable symmetric encryption）[3]和公钥可搜索加密（PEKS, public-key encryption with keyword search）[4]，SSE 算法简单并且更加高效，所以得到更广泛的应用。然而，早期的 SSE 方案都是静态的。

动态对称可搜索加密（DSSE, dynamic searchable symmetric encryption）技术[5]的提出扩展了 SSE 的应用，其不仅支持数据检索，还支持数据动态更新。然而，DSSE 方案在动态更新的过程中，服务器可能会学习到一些信息，即导致信息泄露，泄露的信息包括数据库大小、查询模式和访问模式。为提出更安全的方案，前向安全和后向安全[6]的概念被提出。前向安全是指无法确认新添加的文件是否包含已搜索的关键字。后向安全是指如果一个关键字/文件对(w, f)被添加到数据库中，然后被删除，则对 w 的后续搜索查询不会显示文件 f。Bost 等[7]根据泄露程度的不同定义了 3 种后向安全：对于关键字 w，Type-I 泄露了当前匹配关键字 w 的文件和文件的插入时间，以及 w 更新的总次数；Type-II 额外泄露了更新发生的时间；Type-III 额外泄露了已删除文件被添加的时间，以及被哪次更新删除。

大多数前向安全和后向安全的 DSSE 方案都是利用茫然随机访问机（ORAM, oblivious random access machine）避免访问模式的泄露。然而，ORAM 结构检索和更新效率低，通信开销和计算开销大，并且不支持包含大量文件的数据库，所以需要采用新的索引结构去提升检索和更新效率，并且保证方案的前向安全和后向安全。同时，现有的同时满足前向安全和后向安全的 DSSE 方案只支持单关键字查询，表达能力受到严重限制且不满足实际应用的需求。因此，任何 SSE 方案要真正实用，至少应该支持连接关键字查询，即给定一组关键字$(w_1, \cdots, w_n)$，该方案能够找到并返回包含所有这些关键字的文档集。

为了解决上述问题，本文主要研究工作如下。

1) 采用正排索引和倒排索引结合的思想，利用位图索引设计双向索引结构，添加和删除操作仅需要一个模加法运算，不仅简化了更新过程的操作，而且防止了访问模式的泄露。

2) 利用双向索引设计了连接关键字查询方案 BPC-DSSE。该方案使用内积匹配算法计算连接关键字查询结果，查询过程只需要两轮交互，提高了查询效率，同时实现了 Type-I 后向安全。

3) 理论分析和仿真实验结果表明，BPC-DSSE 方案与使用 ORAM 结构和其他结构实现前向安全和后向安全的连接关键字方案相比，具有更好的查询与更新性能，并且保证了更新过程的安全性。

1 相关工作

SSE 首先由 Song 等[2]提出，对于长度为 n 的文件，该方案加密和搜索算法需要流密码和分组密码，操作时间复杂度为 $O(n)$。为了支持动态更新，DSSE[5]被提出。早期的 DSSE 方案很容易受到潜在攻击的威胁[8]，如文件注入攻击[9]。为了减轻这种攻击造成的威胁，前向安全和后向安全的概念被提出[6]。2016 年，Bost[10]给出了前向安全的正式定义，并给出了一种前向安全的 DSSE 方案 Σοφος，该方案使用简单的密码工具（只有伪随机函数和陷门原语），不依赖 ORAM 结构，效率和安全性都有一定的提升。2017 年，Bost 等[7]根据泄露的不同程度从高到低定义了 3 种后向安全类型，并给出了 4 种前向安全和后向安全的方案。第一种方案 Fides 实现了 Type-II 的后向安全。第二种方案 Diana 非常高效，但只实现了 Type-III 的后向安全，Diana 被修改成只需要 2 种往返的后向安全方案 $\text{Diana}_{\text{del}}$。第三种方案 Janus 是具有单次往返的后向安全方案，但该方案也只实现了 Type-III 的后向安全。第四种方案 Moneta 实现了 Type-I 的后向安全，但此方案是基于 TWORAM（ORAM in two rounds）结构的，因此计算开销和通信开销都非常大。自此，一系列前向安全和后向安全的方案被提出[11-17]。

Cash 等[18]提出基于茫然交叉标签（OXT, oblivious cross tag）协议设计有效的 SSE 协议，支持单作者阅读框架中的连接关键字查询。虽然 OXT 协议通过许多专门的数据结构来提供高性能，但它向服务器泄露“部分”数据库信息。同时，该方案需要三轮交互，增加了通信开销。2018 年，Lai 等[19]提出了一种新的 SSE 协议，称为隐藏交叉标签（HXT, hidden cross tag），它消除了连接关键字查询的关键字对结果模式（KPRP, keyword pair result pattern）泄露。2019 年，Wu 等[20]提出以自上而下的方式存储索引元素到虚拟二叉树（VBTree, virtual binary tree）结构中，而不存储任何树枝和树节点，该树只存在于逻辑视图中，并且所有的元素实际上都存储在一个哈希表中，但是该方案只实现了前向安全。2020 年，Zuo 等[21]提出了一个扩展位图索引结构，比 VBTree 更高效地实现了连接关键字查询，

并且实现了前向安全和后向安全。2021 年，Patranabis 等[22]提出了茫然动态交叉标签（ODXT, oblivious dynamic cross tag），ODXT 通过有效地支持在大型数据库上的快速更新和连接关键字查询，在性能和安全性之间提供了现实的折中，同时根据现有的前后隐私概念，只对服务器造成适度的访问模式泄露。

2　DSSE 定义及其安全模型

本节给出了必要的背景知识，包括符号定义、DSSE 方案的定义及其安全模型、前向安全和后向安全。

2.1　符号定义

本文中使用的主要参数如表 1 所示。

表 1　主要参数

参数	含义
λ	安全参数
F	安全的伪随机函数
n	位串的长度
W	关键字空间
f_i	第 i 个文件（$1\leqslant i\leqslant n$）
DB	数据库
EDB	加密数据库
ST_c	当前的搜索令牌
CT	存储当前搜索令牌 ST_c 和更新的次数 c
bs	位图索引对应的位串
E	加密后的位串
Sum_E	加密后的位串的和
sk	一次性安全密钥
Sum_{sk}	一次性安全密钥的和

2.2　DSSE 方案的定义

数据库 DB 为关键字/文件标识符对的列表；f_i 为文档标识符；w_i 为 f_i 中包含的所有关键字集合，$w_i\subseteq\{0,1\}^*$；n 为数据库 DB 中所有文件的数量。那么数据库 DB 可以表示为 $\mathrm{DB}=(f_i,w_i)_{i=1}^n$。|DB|为关键字/文件标识符对的总数量。$W$ 为数据库 DB 中所有关键字的集合，$W=\bigcup_{i=1}^n w_i$。

DSSE 方案包括算法 Setup 以及运行在客户端和服务器的协议 Search 和 Update，即 DSSE=(Setup, Search, Update)，具体描述如下。

(EDB, σ)←Setup(1^λ,DB)。输入安全参数λ、数据库 DB，输出当前的状态σ以及加密数据库 EDB，其中，σ存储在客户端，EDB 上传到云服务器。

(Γ, ⊥)←Search(s, σ, EDB)。对于状态σ，客户端发出搜索请求 s，与拥有加密数据库的服务器进行交互。执行协议结束后，客户端输出一个匹配搜索请求 s 的文件标识符的集合Γ，服务器不输出任何内容。

(σ', EDB′)←Update(σ, op, in; EDB)。对于状态σ，操作 op∈{add, del}，集合 in=(f, w)是文件/关键字对，客户端向拥有加密数据库的服务器发出添加或删除 in 中文件/关键字对的请求。协议结束后，返回给客户端新的状态σ'，并更新服务器的加密数据库为 EDB′。

2.3　DSSE 方案安全模型

给定一个 2.2 节定义的 DSSE 方案，本文将定义现实博弈 REAL 和理想博弈 IDEAL 并给出其安全模型，REAL 反映了原始的 DSSE 方案的行为，IDEAL 反映了模拟器 S 的行为，该模拟器将原始的 DSSE 的泄露作为输入。对于敌手 A，泄露函数定义为 $L=(L^{\mathrm{stup}}, L^{\mathrm{srch}}, L^{\mathrm{updt}})$，$L^{\mathrm{stup}}$ 是在执行 Setup 算法时 A 可以学到的信息，L^{srch} 是在执行 Search 协议时 A 可以学到的信息，L^{updt} 是在执行 Update 协议时 A 可以学到的信息。概率博弈 REAL 和 IDEAL 定义如下。

$\mathrm{REAL}_A^{\mathrm{DSSE}}(\lambda)$。首先运行 Setup($\lambda$, DB)，得到加密数据库 EDB，$A$ 发起一系列搜索查询 q 或者是更新查询(op, in)，最后，A 返回实验结果 b, $b\in\{0,1\}$。

$\mathrm{IDEAL}_{A,S}^{\mathrm{DSSE}}(\lambda)$。模拟器 S 首先执行泄露函数 $L^{\mathrm{stup}}(\lambda, \mathrm{DB})$，针对 A 发起一系列搜索查询或更新查询的不同(op, in)，模拟器 S 分别输入 L^{srch} 和 L^{updt}，并将输出结果返回给 A。最后，A 返回实验结果 b, $b\in\{0,1\}$。

定义 1　机密性。给定一个动态可搜索加密方案 DSSE 和以上规则，如果对于任意概率多项式的敌手 A，存在高效的模拟器 S 和输入 L 满足 $\left|\Pr\left[\mathrm{REAL}_A^{\mathrm{DSSE}}(\lambda)=1\right]-\Pr\left[\mathrm{IDEAL}_{A,S}^{\mathrm{DSSE}}(\lambda)=1\right]\right|\leqslant\mathrm{negl}(\lambda)$，则这种 DSSE 方案是 L-适应性安全的。

2.4　前向安全

前向安全由 Stefanov 等[6]提出。前向安全保证了 DSSE 方案在更新期间不会泄露新插入的文件与之前的搜索查询匹配的信息，前向安全的正式定义[7]如下。

定义 2　前向安全。如果一种 L-适应安全的 DSSE 方案是前向安全的，那么存在一个无状态的函数 L'，其更新的泄露函数 L^{updt} 为

$$L^{\text{updt}}(\text{op},\text{in}) = L'\left(\text{op},\left\{\left(f_i,\mu_i\right)\right\}\right) \tag{1}$$

其中，$\{(f_i, \mu_i)\}$是与更新文件 f_i 的修改关键字数量 μ_i 配对的修改文件集。

本文中，泄露函数为 $L^{\text{updt}}(\text{op}, \{(W, \text{bs}_w), (f, \text{bs}_f)\})=L'(\text{op}, \text{bs}_w, \text{bs}_f)$, 其中，$W$ 和 f 是更新操作关键字和文件集合，bs_w 和 bs_f 是对应的更新索引。

2.5 后向安全

后向安全由 Stefanov 等[6]提出。对同一关键字的两次查询期间，后向安全保证不会泄露之前插入且之后删除的文件的信息。Bost[10]定义了 3 种不同泄露程度的后向安全：Type-I、Type-II 和 Type-III。Zuo 等[16]定义了 Type-Ⅰ˙级别的后向安全。后向安全泄露信息对比如表 2 所示，其中，√表示泄露相应信息，—表示不泄露相应信息。

表 2 后向安全泄露信息对比

后向安全	当前匹配关键字 w 的文件	文件插入时间	w 更新总次数	w 更新时间	插入后被删除文档的插入删除时间
Type-I	√	√	√	—	—
Type-II	√	√	√	√	—
Type-III	√	√	√	√	√
Type-Ⅰ˙	√	—	√	√	—

例如，发生以下一组操作序列，顺序为{1, search, w}, {2, add, (w, f_1)}, {3, add, (w, f_2)}, {4, add, (w, f_3)}, {5, delete, (w, f_1)}, {6, search, w}，Type-I 后向安全会泄露当前匹配关键字 w 的文档有 f_2 和 f_3，分别在时间 3 和时间 4 插入，以及发生在 w 上的更新一共有 4 次；Type-II 额外泄露 w 的更新发生在时间 2～时间 5；Type-III 额外泄露 w 在时间 2 的插入在时间 5 被删除。而 Type-Ⅰ˙后向安全仅泄露一共有 4 次更新，分别发生在时间 2～时间 5。

本文中基于动态填充算法的BPC-DSSE方案采用基于位图索引的技术，添加和删除使用同一个模块，所以不会泄露文件插入时间；并且查询结果返回位串，隐藏了访问模式。综上，BPC-DSSE 方案仅泄露更新次数以及更新时间，实现了 Type-Ⅰ˙后向安全。

为了正式定义 Type-Ⅰ˙，对于搜索查询列表 Q，本文定义了搜索模式 $\text{sp}(w)=\{t:(t, w)\in Q\}$，其中 t 为时间戳。该搜索模式会泄露对关键字 w 的搜索查询重复的次数。本文还定义了新的泄露函数 TimePDB，对于一个关键字 w，TimePDB(w)存储了关键字 w 所有更新时间 t 的列表。对于更新查询 Q'，有

$$\text{TimeBPC}(w) = \left\{t:\left(t,\text{op},(w,\text{BS}_w)\in Q'\right)\right\} \tag{2}$$

定义 3 后向安全。对于一个 L-适应性安全的 DSSE 方案，如果存在 2 个无状态的函数 L'和 L''，使搜索和更新的泄露函数可以写为

$$L^{\text{updt}}(\text{op}, w, \text{BS}) = L'(\text{op}) \tag{3}$$

$$L^{\text{srch}}(w) = L''\left(\text{sp}(w), \text{TimeBPC}(w)\right) \tag{4}$$

则称这种 DSSE 方案是 Type-Ⅰ˙后向安全的。

3 BPC-DSSE 方案

本节给出了 BPC-DSSE 的方案设计。首先介绍了方案的设计理念，然后给出了方案涉及的详细步骤。

3.1 设计理念

BPC-DSSE 方案是具有前向安全和 Type-Ⅰ˙后向安全的连接关键字 DSSE 方案，主要思想是基于位图索引[16]构建双向索引结构，利用基于同态加法的对称加密[17]对索引进行加密，并通过基于内积的匹配算法[23]实现连接关键字查询。

更新操作。构建索引采用位图索引，索引结构及其运算规则如图 1 所示。假设数据库文件数量最大为 n，关键字空间 W 的每个关键字都维持一个 n 位的比特串，用来表示文件是否包含此关键字，如果包含则将相应位置设置为 1，否则设置为 0。对倒排索引以相同的方式进行处理，为每个文件维持一个$|W|$大小的比特串。如图 1(a)所示，数据库的文件最大数量为 6，关键字空间大小为 4，即 n=6，$|W|$=4，对应的比特串$(000100)_2$ 以及$(0010)_2$ 表示(w_1, f_2)存在。下面以关键字索引演示位图索引的计算规则，文件索引计算规则相同。如图 1(b)所示，如果添加文件 f_3，需要生成比特串$(001000)_2$，并将其加到初始比特串$(000100)_2$上，得到结果$(001100)_2$。如图 1(c)所示，如果删除文件 f_2，需要生成比特串 $-(000100)_2=(111100)_2 \mod 2^6$ 并加到初始比特串$(000100)_2$上，得到结果$(000000)_2$。

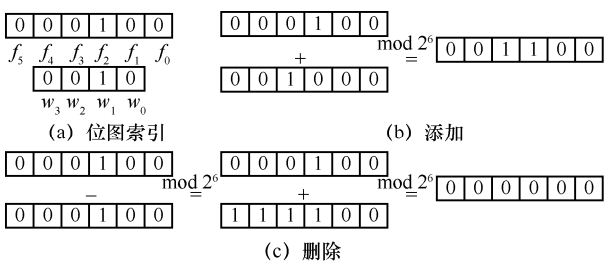

图 1 索引结构及其运算规则

连接关键字查询实现。其主要思想是采用正排索引和倒排索引结合，利用位图索引实现高效的连接关键字查询。首先，搜索频率最低的关键字 w_1 得到相应倒排索引的位串，将其解密后得到与 w_1 匹配的文件标识符。然后，根据文件标识符获取相应的正排索引对应的位串。最后，利用基于内积的匹配思想将位串解密并计算与查询位串的内积，通过内积结果过滤得到最终的结果集合。

方案主要步骤如图 2 所示，以明文形式进行说明。此时的索引结构如左上角位图索引所示，关键字空间大小为 6，文件数量为 8。查询关键字 w_0、w_1 和 w_5，即 q=(100011)$_2$。首先，访问倒排索引得到频率最低的关键字 w_0 对应的位串为(10101010)$_2$，解密后得到包含关键字 w_0 的文件标识符集合为 f={f_1, f_3, f_5, f_7}，访问正排索引获取 f 中文件对应的位串分别为(100111)$_2$、(111010)$_2$、(100111)$_2$、(101011)$_2$，将其分别与查询位串 q 内积，并将内积结果与查询关键字的个数（此例为 3）进行比较，将比较结果相同的文件标识符加入结果集合，得到最终结果集合 result={f_1, f_5, f_7}。

3.2　方案涉及的详细步骤

在详细介绍算法之前，首先给出具有加法同态性质的对称加密算法[14]，主要由以下 4 种算法组成。

初始化 Setup(1^λ)。其输入为安全参数λ，输出为公共参数消息空间 N。如果一种方案可以支持的最大文件数量为 n，那么消息空间 N=2^n。

加密算法 Enc(m, sk, N)。其输入为明文消息 m、随机的安全密钥 sk 以及公共参数 N，输出为加密后的密文 c。其中，0≤m<N，0≤sk<N，密文 c=sk+m (mod N)。

解密算法 Dec(c, sk, N)。其输入为密文 c、随机的安全密钥 sk 以及公共参数 N，输出为解密后的明文 m。其中，明文 m=c−sk (mod N)。

同态加法 Add(c_1, c_2, N)。其输入为 2 个密文 c_1 和 c_2 以及公共参数 N，输出为 2 个密文 c_1 和 c_2 之和 c。对于密文 c_1= Enc(m_1, sk$_1$, N)，c_2= Enc(m_2, sk$_2$, N)，c= c_1+c_2= Enc(m_1+m_2, sk$_1$+ sk$_2$, N)。

BPC-DSSE 方案由一种算法Setup 和 2 个协议 Update 和 Search 组成。

Setup 算法如算法 1 所示。客户端首先初始化 2 个空映射 M_w 和 M_f，分别存储关键字 w 对应的 bs$_w$ 和文件 f 对应的 bs$_f$，然后初始化 2 个空的映射 CT$_w$ 用来存储当前搜索令牌 ST$_{cw}$ 以及每个关键字更新次数 cw，CT$_f$用来存储当前搜索令牌 ST$_{cf}$ 以及每个关键字更新次数 cf。最后计算 EDB←{M_w, M_f}，将 EDB 以及状态σ发送给服务器。

算法 1　BPC-DSSE Setup(1^λ, DB)

输入　安全参数λ，数据库 DB

输出　EDB，状态σ=(n_w, n_f, K, CT$_w$, CT$_f$)

1)　$K \xleftarrow{\$} \{0,1\}^\lambda$, n_w, n_f← setup(1^λ)

2) CT$_w$, CT$_f$, M_w, M_f←空映射

3) EDB←{M_w, M_f }

4) return (EDB, σ=(n_w, n_f, K, CT$_w$, CT$_f$))

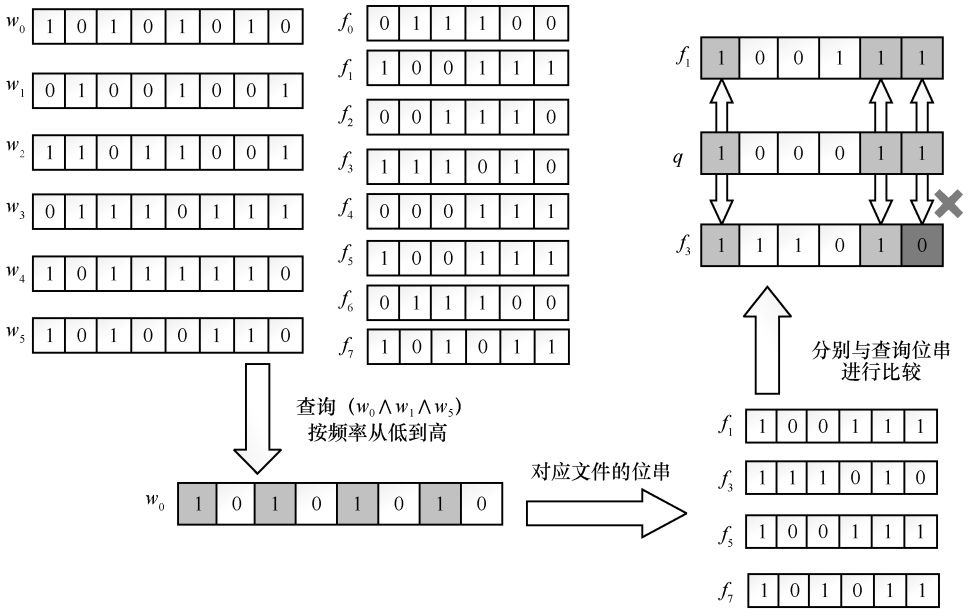

图 2　方案主要步骤

Update 算法如算法 2 所示。客户端根据更新的 $(w,\ \mathrm{id})$生成对应的位串 bs_w 和 bs_f，将其加密后得到加密位串 Ew 和 Ef。首先访问 CT_w 得到关键字 w 对应的搜索令牌 $\mathrm{ST}_{\mathrm{cw}}$ 和更新次数 cw，并访问 CT_f 得到文件 f 对应的搜索令牌 $\mathrm{ST}_{\mathrm{cf}}$ 和更新次数 cf，接下来生成密钥。如果数据库为空，则初始化搜索令牌 $\mathrm{ST}_{\mathrm{cw}}$、$\mathrm{ST}_{\mathrm{cf}}$ 和更新次数 cw、cf。用哈希函数 H_1 生成更新令牌 UW 和 UF 来更新服务器端的 EDB，并用 H_2 混淆之前的搜索令牌，用哈希函数 H_3 生成一次密钥 $\mathrm{sk}_{\mathrm{cw}}$ 和 $\mathrm{sk}_{\mathrm{cf}}$。将 UW、UF、Ew、Ef 和混淆后的搜索令牌 C 发送给服务器，并且更新 CT 得到新的状态 σ'。服务器更新映射对应的位串，并加到原始位串上。

算法 2 BPC-DSSE Update(w, f, bs_w, bs_f, σ, EDB)

输入 关键字 w 和文件 f, 对应的更新位串 bs_w 和 bs_f，状态 σ

输出 更新后的加密数据库 EDB′

客户端

1) $K_w \| K'_w \leftarrow F_K(w), K_f \| K'_f \leftarrow F_K(f)$

2) $(\mathrm{ST}_{\mathrm{cw}}, \mathrm{cw}) \leftarrow \mathrm{CT}_w[w]$，$(\mathrm{ST}_{\mathrm{cf}}, \mathrm{cf}) \leftarrow \mathrm{CT}_f[f]$

3) if $(\mathrm{ST}_{\mathrm{cw}}, \mathrm{cw}) = \perp$ then

4) $\quad \mathrm{cw} \leftarrow -1, \mathrm{ST}_{\mathrm{cw}} \leftarrow \{0,1\}^{\lambda}$

5) end if

6) if $(\mathrm{ST}_{\mathrm{cf}}, \mathrm{cf}) = \perp$ then

7) $\quad \mathrm{cf} \leftarrow -1, \mathrm{ST}_{\mathrm{cf}} \leftarrow \{0,1\}^{\lambda}$

8) end if

9) $\mathrm{ST}_{\mathrm{cw+1}} \leftarrow \{0,1\}^{\lambda}, \mathrm{ST}_{\mathrm{cf+1}} \leftarrow \{0,1\}^{\lambda}$

10) $\mathrm{CT}_w[w] \leftarrow (\mathrm{ST}_{\mathrm{cw+1}}, \mathrm{cw+1}), \mathrm{CT}_f[f] \leftarrow (\mathrm{ST}_{\mathrm{cf+1}}, \mathrm{cf+1})$

11) $\mathrm{UW}_{\mathrm{cw+1}} \leftarrow H_1(K_w, \mathrm{ST}_{\mathrm{cw+1}}), \mathrm{UF}_{\mathrm{cf+1}} \leftarrow H_1(K_f, \mathrm{ST}_{\mathrm{cf+1}})$

12) $C_{\mathrm{ST}_{\mathrm{cw}}} \leftarrow H_2(K_{\mathrm{w}}, \mathrm{ST}_{\mathrm{cw+1}}) \oplus \mathrm{ST}_{\mathrm{cw}}$, $C_{\mathrm{ST}_{\mathrm{cf}}} \leftarrow H_2(K_f, \mathrm{ST}_{\mathrm{cf+1}}) \oplus \mathrm{ST}_{\mathrm{cf}}$

13) $\mathrm{sk}_{\mathrm{cw+1}} \leftarrow H_3(K'_w, \mathrm{cw+1}), \mathrm{sk}_{\mathrm{cf+1}} \leftarrow H_3(K'_f, \mathrm{cf+1})$

14) $\mathrm{Ew}_{\mathrm{cw+1}} \leftarrow \mathrm{Enc}(\mathrm{sk}_{\mathrm{cw+1}}, \mathrm{bs}_w, n_w), \mathrm{Ef}_{\mathrm{cf+1}} \leftarrow \mathrm{Enc}(\mathrm{sk}_{\mathrm{cf+1}}, \mathrm{bs}_f, n_f)$

15) send $(\mathrm{UW}_{\mathrm{cw+1}}, \mathrm{UF}_{\mathrm{cf+1}}, \mathrm{Ew}_{\mathrm{cw+1}}, \mathrm{Ef}_{\mathrm{cf+1}}, C_{\mathrm{ST}_{\mathrm{cw}}}, C_{\mathrm{ST}_{\mathrm{cf}}})$

服务器

16) $M_w[\mathrm{UW}_{\mathrm{cw+1}}] \leftarrow (\mathrm{Ew}_{\mathrm{cw+1}}, C_{\mathrm{ST}_{\mathrm{cw}}})$

17) $M_f[\mathrm{UF}_{\mathrm{cf+1}}] \leftarrow (\mathrm{Ef}_{\mathrm{cf+1}}, C_{\mathrm{ST}_{\mathrm{cf}}})$

Search 算法如算法 3 所示。客户端首先根据频率最低的 w_1 生成搜索令牌并发送给服务器。服务器访问 M_w 得到加密位串 bs_w，并返回客户端。客户端解密得到对应文件 id，并发送给服务器。服务器访问 M_f 得到对应的位串，将其逐一与查询位串内积，并将内积结果和查询关键字的个数相同的文件 id 加入结果集合。

算法 3 BPC-DSSE Search(q, σ, EDB)

输入 查询连接关键字 $q=w_1 \wedge w_2 \wedge \cdots \wedge w_k$，状态 σ

输出 结果集合 result

客户端

1) $K_w \| K'_w \leftarrow F_k(w_1), (\mathrm{ST}_{\mathrm{cw}}, \mathrm{cw}) \leftarrow \mathrm{CT}_w[w_1]$

2) if $(\mathrm{ST}_{\mathrm{cw}}, \mathrm{cw}) = \perp$ then

3) $\quad$ 返回∅

4) end if

5)将$(K_w, \mathrm{ST}_{\mathrm{cw}}, \mathrm{cw})$发送给服务器

服务器

6) $\mathrm{Sum}_{\mathrm{Ew}} \leftarrow 0$

7) for i=cw to 0 do

8) $\quad \mathrm{UW}_{\mathrm{cw}} \leftarrow H_1(K_w, \mathrm{ST}_{\mathrm{cw}})$

9) $\quad (\mathrm{Ew}_{\mathrm{cw}}, C_{\mathrm{STcw}}) \leftarrow M_W[\mathrm{UW}_{\mathrm{cw}}]$

10) $\quad \mathrm{Sum}_{\mathrm{Ew}_{\mathrm{cw}}} \leftarrow \mathrm{Add}(\mathrm{Sum}_{\mathrm{Ew}_{\mathrm{cw}}}, \mathrm{Ew}_{\mathrm{cw}}, n)$

11) $\quad$ 移除 $M_w[\mathrm{UW}_{\mathrm{cw}}]$

12) $\quad$ if $C_{\mathrm{ST}_{\mathrm{cw-1}}} = \perp$ then

13) $\quad\quad$ break

14) $\quad$ end if

15) $\quad \mathrm{ST}_{\mathrm{cw-1}} \leftarrow H_2(K_f, \mathrm{ST}_{\mathrm{cf+1}}) \oplus C_{\mathrm{STcw-1}}$

16) end for

17) $M_w[\mathrm{UW}_{\mathrm{cw+1}}] \leftarrow (\mathrm{Sum}_{\mathrm{Ew}_{\mathrm{cw}}}, \perp)$

18)将 $\mathrm{Sum}_{\mathrm{Ew}_{\mathrm{cw}}}$ 发送给客户端

客户端

19) $\mathrm{Sum}_{\mathrm{sk}} \leftarrow 0$

20) for i=cw to 0 do

21) $\quad \mathrm{sk}_i \leftarrow H_3(K'_w, \mathrm{cw})$

22) $\quad \mathrm{Sum}_{\mathrm{sk}} \leftarrow \mathrm{Sum}_{\mathrm{sk}} + \mathrm{sk}_i \bmod n_w$

23) end for

24) $\mathrm{bs}_w \leftarrow \mathrm{Dec}(\mathrm{Sum}_{\mathrm{sk}}, \mathrm{Sum}_{\mathrm{Ew}_{\mathrm{cw}}}, n_w)$

25)将 bs_w 转换为文件标识符 f 放入集合 File

26) $\mathrm{Sum}_{\mathrm{sk}} \leftarrow 0$

27) for i=cf to 0 do

28) $\quad \mathrm{sk}_i \leftarrow H_3(K'_w, \mathrm{cf})$

29) $\quad \mathrm{Sum}_{\mathrm{sk}}[i] \leftarrow \mathrm{Sum}_{\mathrm{sk}} + \mathrm{sk}_i \bmod n_f$

30) end for

31) for i=0 to cw do

32)　(ST_{cf}, cf)←CT_f[File[i]]
33)　UF[i]←$H_1(K_f, ST_{cf})$
34) end for
35)将 UF 返回给服务器
服务器
36)for i=0 to cw do
37)　$M[i]$←M_f[UF[i]]
38)end for
39)将 M 返回给客户端
客户端
40)result=集合
41)将查询 q 转换为位串 bs_q
42)for i=0 to cw do
43)　(ST_{cf},cf)←CT_f[File [i]]
44)　for i=cf to 0
45)　　sk_{cf}←$H_3(K'_f$, cf)
46)　　Sum_{sk}←Sum_{sk} + sk_{cf} mod n_f
47)　end for
48)　$D[i]$←Dec($Sum_{sk}[i]$, $M[i]$, k)
49)　if $bs_q D[i]=k$ then
50)　　result= result∪File[i]
51)　end if
52)end for
53) 返回查询结果 result

4　安全性分析

本节给出 BPC-DSSE 的安全分析。

4.1　前向安全

BPC-DSSE 是前向安全的。在更新期间，使用 3 个哈希函数，H_1 通过随机选取的密钥来生成更新令牌 UW 和 UF，由于加上计数 c，每个更新令牌都是不相同的；H_2 通过新生成的搜索令牌和密钥将之前的搜索令牌进行更新，如果拥有之前的搜索令牌，就可以通过哈希函数计算得到之后的搜索令牌，而拥有最新的搜索令牌却无法得知之前的搜索令牌，保证了前向安全性；H_3 用来生成一次性密钥。综上所述，BPC-DSSE 是前向安全的。

4.2　后向安全

BPC-DSSE 是 Type-Ⅰ后向安全的。由于使用位图索引，并且添加和删除操作只需要一个模加法运算，因此不会泄露插入时间以及文件被插入后删除对应的插入和删除时间；同时由于 F 的伪随机性，每次更新时都包含了不同的输入，对于两次查询期间添加随后又删除的文件，服务器是无法学习到其相关信息的。综上所述，BPC-DSSE 是 Type-Ⅰ后向安全的。

定理 1　BPC-DSSE 的前向安全和 Type-Ⅰ后向安全。F 是安全的伪随机函数，RO_1、RO_2 和 RO_3 是随机预言机。定义泄露函数 $L_{BPC\text{-}DSSE}=(L^{srch}, L^{updt})$，如果泄露函数 $L^{srch}(w)=(sp(w),\ rp(w),\ Time(w))$，$L^{updt}(op,\ (w,\ bs_w),\ (f,\ bs_f))=\perp$，那么 BPC-DSSE 是 $L_{BPC\text{-}DSSE}$−适应前向安全和 Type-Ⅰ后向安全的。

证明　与 Chamani 等[12]类似，从现实博弈 REAL 和理想博弈 IDEAL 之间构造了一系列博弈 Game，连续 2 个 Game 之间存在细微的不同，但不可区分，从而得到 REAL 与 IDEAL 不可区分的结论。最后，用定理 1 中定义的泄露函数模拟了 IDEAL。

Game G_0。G_0 和现实博弈 $REAL_A^{DSSE}(\lambda)$ 完全一样。

Game G_1。G_1 与 G_0 相同，除了不使用 F 生成关键字 w 的密钥，而是以相同的概率随机选择密钥，并将密钥存储在表 Key 中。如果从来没有搜索过 w，则生成 w 的密钥并存储在表中；如果搜索过 w，则返回表中关键字 w 的密钥。由于伪随机函数的安全性，G_1 与 G_0 是不可区分的。

Game G_2。G_2 与 G_1 相同，除了在更新过程中，使用随机预言机 RO_1 代替 H_1 进行运算。对于更新协议，更新令牌是随机生成的并被存储在表 UT 中。当搜索协议被调用时，通过计算 $RO_1(K, ST)$生成更新令牌并存储在表 R_1 中，若$(K,\ ST)$已经在 R_1 中，则可以直接搜索 R_1 得到结果。对于同样的哈希函数 H_2 和 H_3，也同样使用随机预言机逐个替换，由于搜索令牌是随机选取的，并且每次输入的令牌都是不同的，G_2 与 G_1 是不可区分的。

Game G_3。G_3 与 G_2 相同，除了使用长度为 n 的全 0 的位串代替 bs。由于具有加法同态性质对称加密的安全性，G_3 与 G_2 是不可区分的。

模拟器。模拟器与 G_3 相同，只是用搜索模式 $sp(w)$代替搜索关键字 w 来模拟理想博弈。对于更新，在 G_3 中为每次更新选择了新的随机字符串。对于搜索，模拟器从当前搜索标记 ST_c 开始，并为之前的搜索标记选择一个随机字符串。对于关键字 w，模拟器使用新的时间戳 t'←minsp(w)，然后通过 RO_2 将其嵌入密文 C 中。此外，模拟器通过 RO_3 将 bs 嵌入 ST_c 并将所有 0 嵌入剩余的搜索标记中。因此，G_3 和模拟器是不可区分的。另外，模拟器中描述的本质上是 2.3 节定义的理想博弈 $IDEAL_{A,S}^{DSSE}(\lambda)$，

因此 G_3 与 $\mathrm{IDEAL}_{A,S}^{\mathrm{DSSE}}(\lambda)$ 是不可区分的。

综上，$\mathrm{IDEAL}_{A,S}^{\mathrm{DSSE}}(\lambda)$ 与 G_0 是不可区分的，即 $\mathrm{IDEAL}_{A,S}^{\mathrm{DSSE}}(\lambda)$ 与 $\mathrm{REAL}_{A}^{\mathrm{DSSE}}(\lambda)$ 是不可区分的。

5 仿真分析

本节主要对 BPC-DSSE 方案的查询、更新复杂度和存储开销进行分析，并给出仿真结果。BPC-DSSE 方案与已有前向安全和后向安全的动态可搜索加密方案（包含单关键字查询以及连接关键字查询）的性能对比如表 3 所示，其中，N 为关键字/文件标识符对数，n_w 为包含关键字 w 的文档数量，d_w 为关键字 w 被删除的条目数，a_w 为关键字 w 的更新次数，$|w|$为关键字的总数，$|D|$为文件总数，$\tilde{O}$ 符号隐藏了多对数因子。连接关键字中，n 为连接关键字查询的关键字个数，w_1 为其中频率最小的关键字，$|\mathrm{Upd}(w_1)|$为包含 w_1 的文件的数量更新操作，HT 为树的高度。经对比分析可以发现，实现连接关键字的方案中，VBTree[20]没有实现后向安全性，FBDSSE-CQ[21]实现了 Type-C 的后向安全性，ODXT[22]只实现了 Type-II 的后向安全性，因此，BPC-DSSE 方案实现了更强的后向安全性。

5.1 计算开销

在查询阶段，第一次交互服务器需要计算关键字 w_1 对应的位串并返回，时间复杂度为 $O(|\mathrm{Upt}(w_1)|)$，客户端需要计算出文件对应的更新令牌，复杂度相同；接下来服务器通过文件更新令牌返回文件对应的加密位串，时间复杂度为 $O(|\mathrm{Upt}(w_1)|)$，客户端解密出文件的位串，并与查询位串进行内积，时间复杂度为 $O(|\mathrm{Upt}(w_1)|)$，综上，查询过程时间复杂度为 $O(|\mathrm{Upt}(w_1)|)$。在更新阶段，客户端服务器只需要固定地进行搜索令牌更新以及更新令牌生成操作，时间复杂度为 $O(1)$。

5.2 存储开销

在客户端，存放状态σ，由于需要找到最小频率的关键字，因此造成的客户端的存储开销为 $O(1)$。在服务器端，由于需要存储双向映射 M_w 和 M_f，因此服务器端存储开销为 $O(\max(|w|,|D|))$，其中 $\max(|w|,|D|)$为文件数量和关键字数量中的最大值。

5.3 性能对比

本节给出BPC-DSSE方案经过一系列测试后的性能评估。本文方案采用拥有 AMD Ryzen 7 4800U with Radeon Graphics 1.8 GHz 的处理器配置了 Windows 10(64 位)系统的机器，支持 16 GB RAM，使用 Java 语言编程实现。实验采用的是安然电子邮件（Enron）数据集，包括 50 万个不同的文件，将其解析为本文设计的双向索引结构，在索引中提取 190 万个不同的关键字。对于本文方案，将文件的最大数量设置为 50 万。在实验过程中，多次调整文件更新次数以及查询连接关键字的个数进行实验，以此模拟不同的情境下本文方案的性能。

首先，测试更新性能。选用实现连接关键字的方案中效率较高的 VBTree 方案[20]以及同样使用位图索引实现连接关键字的 FBDSSE-CQ 方案[21]来进行对比。通过变换更新次数来测试更新时间受更新次数的影响，根据现有论文对比实验的设计，将更新次数设置为 10～50 次，测试结果如图 3 所示。由于位图索引简化了更新操作并且方案基于双向索引结构，因此 BPC-DSSE 的更新时间比 VBTree 以及 FBDSSE-CQ 更短。由此可见，BPC-DSSE 实现了更好的更新性能。

表 3 不同方案的性能对比

方案	查询	更新	前向安全	后向安全	查询类型	客户端存储	交互轮数
Moneta[10]	$\tilde{O}(\log^2 N)$	$\tilde{O}(a_w \log N+\log^3 N)$	√	Type-I	单关键字	$O(1)$	3
Orion[12]	$O(\log^2 N)$	$O(\log^2 N)$	√	Type-I	单关键字	$O(1)$	$O(\log N)$
Horus[12]	$O(\log^2 N)$	$O(n_w \log d_w \log N+\log^2 \lvert W\rvert)$	√	Type-III	单关键字	$O(1)$	$O(\log d_w)$
SD_a[13]	$O(\log N)$	$O(a_w+\log N)$	√	Type-II	单关键字	$O(1)$	1
VBTree[20]	$O(n\lvert\mathrm{Upd}(w_1)\rvert\lvert D\rvert)$	$O(\mathrm{HT})$	√	×	连接关键字	$O(\lvert W\rvert)$	1
FBDSSE-CQ[21]	$O(n\lvert\mathrm{Upd}(w_1)\rvert)$	$O(1)$	√	Type-C	连接关键字	$O(\lvert W\rvert)$	1
ODXT[22]	$O(n\lvert\mathrm{Upd}(w_1)\rvert)$	$O(1)$	√	Type-II	连接关键字	$O(\lvert W\rvert \log N)$	1
BPC-DSSE	$O(\lvert\mathrm{Upd}(w_1)\rvert)$	$O(1)$	√	Type-I	连接关键字	$O(1)$	2

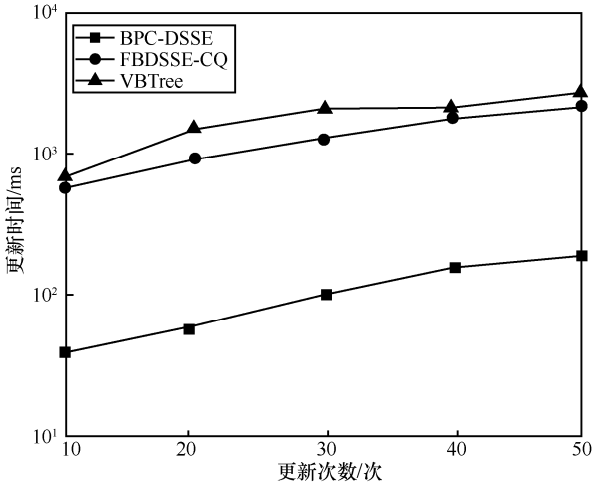

图 3　更新时间受更新次数的影响

接下来，测试查询性能。查询性能主要从两方面来考察：一方面是关键字频率最低的 w_1 的更新次数对查询时间的影响；另一方面是查询关键字个数对查询时间的影响。

对于第一方面，对 2 个关键字进行连接查询的测试，即 $q=w_1 \wedge w_2$，根据 Enron 数据集的文件数，将 w_1 的更新次数设置为 $10^1 \sim 10^5$ 进行测试，重复测试 20 次后取平均值，测试结果如图 4 所示，可见查询时间与 w_1 的更新次数呈线性相关。

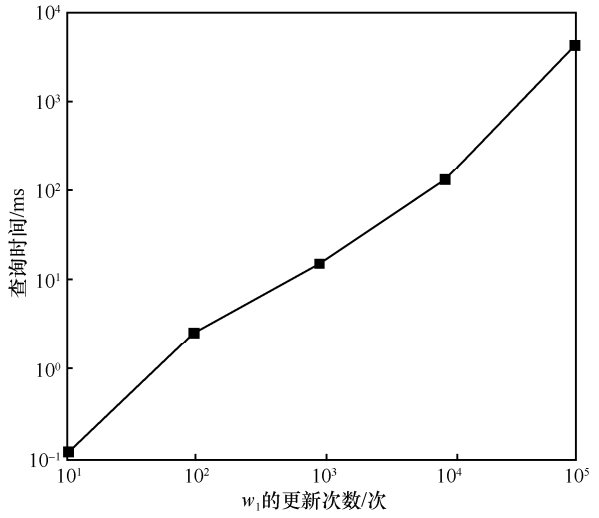

图 4　查询时间受 w_1 的更新次数的影响

对于第二方面，对比方案 VBTree[20] 和 FBDSSE-CQ[21]分别测试了 3 个和 5 个关键字的实验结果，由于 5 个关键字已满足实际应用的需求，因此本文将查询的连接关键字个数设置为 1～5，测试结果如图 5 所示。当查询关键字个数为 1 时，与 FBDSSE-CQ 相比，BPC-DSSE 的搜索时间大约是 FBDSSE-CQ 的 $\frac{1}{30}$；与 VBTree 相比，BPC-DSSE 的查询时间大约是 VBTree 的 $\frac{1}{2\times10^6}$。通过表 3 的查询时间复杂度分析可以看出，FBDSSE-CQ 和 VBTree 的查询时间复杂度与查询关键字个数呈线性相关，而 BPC-DSSE 的查询时间不受查询关键字个数的影响。所以随着查询关键字个数从 1 增加到 5，FBDSSE-CQ 的查询时间增长了 5 倍，VBTree 的查询时间增加了 2 倍，而 BPC-DSSE 的查询时间几乎不受影响。由此可见，BPC-DSSE 实现了更好的查询性能。

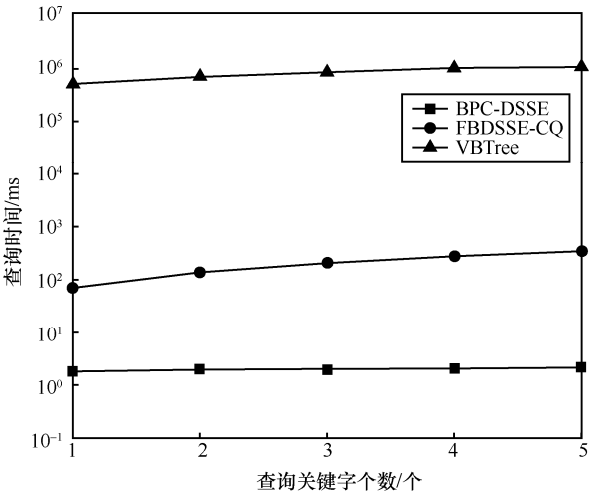

图 5　查询时间受查询关键字个数的影响

6　结束语

本文构造了一种前向安全和后向安全的连接关键字查询方案。基于位图索引设计了双向索引，简化了更新过程，同时防止了访问模式的泄露，提高了方案的安全性。此外，利用基于内积的匹配思想计算连接关键字的查询结果，具有更好的查询性能。安全分析表明，本文方案具有前向安全以及 Type-Ⅰ的后向安全。仿真结果表明，本文方案具有更好的更新性能以及查询性能。

参考文献：

[1] 董晓蕾, 周俊, 曹珍富. 可搜索加密研究进展[J]. 计算机研究与发展, 2017, 54(10): 2107-2120.
DONG X L, ZHOU J, CAO Z F. Research advances on secure searchable encryption[J]. Journal of Computer Research and Development, 2017, 54(10): 2107-2120.

[2] SONG D X, WAGNER D, PERRIG A. Practical techniques for

searches on encrypted data[C]//Proceedings of IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2000: 44-55.

[3] 吴志强, 李肯立, 郑蕙. 高效可扩展的对称密文检索架构[J]. 通信学报, 2017, 38(8): 79-93.
WU Z Q, LI K L, ZHENG H. Efficient and scalable architecture for searchable symmetric encryption[J]. Journal on Communications, 2017, 38(8): 79-93.

[4] 杜瑞忠, 谭艾伦, 田俊峰. 基于区块链的公钥可搜索加密方案[J]. 通信学报, 2020, 41(4): 114-122.
DU R Z, TAN A L, TIAN J F. Public key searchable encryption scheme based on blockchain[J]. Journal on Communications, 2020, 41(4): 114-122.

[5] KAMARA S, PAPAMANTHOU C, ROEDER T. Dynamic searchable symmetric encryption[C]//Proceedings of the 2012 ACM Conference on Computer and Communications Security. New York: ACM Press, 2012: 965-976.

[6] STEFANOV E, PAPAMANTHOU C, SHI E. Practical dynamic searchable encryption with small leakage[C]//Proceedings 2014 Network and Distributed System Security Symposium. Virginia: the Internet Society, 2014: 72-75.

[7] 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. New York: ACM Press, 2017: 1465-1482.

[8] CASH D, GRUBBS P, PERRY J, et al. Leakage-abuse attacks against searchable encryption[C]//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2015: 668-679.

[9] ZHANG Y P, KATZ J, PAPAMANTHOU C. All your queries are belong to us: the power of file-injection attacks on searchable encryption[R]. IACR Cryptology EPrint Archive, 2016.

[10] BOST R. ∑οφος: forward secure searchable encryption[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2016: 1143-1154.

[11] SUN S F, YUAN X L, LIU J K, et al. Practical backward-secure searchable encryption from symmetric puncturable encryption[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2018: 763-780.

[12] CHAMANI J G, PAPADOPOULOS D, PAPAMANTHOU C, et al. New constructions for forward and backward private symmetric searchable encryption[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2018: 1038-1055.

[13] RIZOMILIOTIS P, GRITZALIS S. Simple forward and backward private searchable symmetric encryption schemes with constant number of roundtrips[C]//Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop. New York: ACM Press, 2019: 141-152.

[14] 黄珂. 前向和后向安全的动态对称可搜索加密方案的研究[D]. 上海: 华东师范大学, 2019.
HUANG K. Research on forward and backward security of dynamic symmetric searchable encryption scheme[D]. Shanghai: East China Normal University, 2019.

[15] SARDAR L, RUJ S. FSPVDSSE: a forward secure publicly verifiable dynamic SSE scheme[C]//International Conference on Provable Security. Berlin: Springer, 2019: 355-371.

[16] ZUO C, SUN S F, LIU J K, et al. Dynamic searchable symmetric encryption with forward and stronger backward privacy[C]//European Symposium on Research in Computer Security. Berlin: Springer, 2019: 283-303.

[17] ZUO C, SUN S F, LIU J K, et al. Forward and backward private DSSE for range queries[J]. IEEE Transactions on Dependable and Secure Computing, 2022, 19(1): 328-338.

[18] CASH D, JARECKI S, JUTLA C, et al. Highly-scalable searchable symmetric encryption with support for Boolean queries[C]//Annual Cryptology Conference. Berlin: Springer, 2013: 353-373.

[19] LAI S Q, PATRANABIS S, SAKZAD A, et al. Result pattern hiding searchable encryption for conjunctive queries[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2018: 745-762.

[20] WU Z Q, LI K L. VBTree: forward secure conjunctive queries over encrypted data for cloud computing[J]. The VLDB Journal, 2019, 28(1): 25-46.

[21] ZUO C, SUN S, LIU J K, et al. Forward and backward private dynamic searchable symmetric encryption for conjunctive queries[R]. IACR Cryptology EPrint Archive, 2020.

[22] PATRANABIS S, MUKHOPADHYAY D. Forward and backward private conjunctive searchable symmetric encryption[C]//Proceedings 2021 Network and Distributed System Security Symposium. Virginia: the Internet Society, 2021: doi.org/10.14722/ndss.2021.23116.

[23] FU Z J, WU X L, GUAN C W, et al. Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(12): 2706-2716.

[作者简介]

杜瑞忠（1975− ），男，河北献县人，博士，河北大学教授、博士生导师，主要研究方向为可信计算与网络安全。

张玉晴（1997− ），女，河北廊坊人，河北大学硕士生，主要研究方向为可信计算与信息安全。

李明月（1993− ），女，河北阜平人，南开大学博士生，主要研究方向为信息安全、可搜索加密等。