计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202220924
61(6): 1545−1558, 2024

一种基于 SHVE 的连接查询动态对称可搜索加密方案

黄一才　郁　滨
（信息工程大学　郑州　450001）
（huangyicai3698@163.com）

Dynamic Searchable Symmetric Encryption Scheme for Conjunctive Queries Based on SHVE

Huang Yicai and Yu Bin
（*Information Engineering University*, *Zhengzhou* 450001）

Abstract　Searchable symmetric encryption (SSE) has been widely concerned because of its high search efficiency. Supporting conjunctive queries can effectively improve the usability of the scheme. However, most of the existing schemes with supporting conjunctive queries just work on static databases, which do not support dynamic updates of encrypted database, such as add or delete operation. Combined with the definition of symmetric hidden vectors encryption (SHVE), a new definition, called dynamic symmetric hidden vectors encryption (DSHVE), is proposed. On the basis of this definition, a DSHVE scheme supporting update operations is constructed. At the same time, by designing oblivious dynamic cross-tags and critical data access structures based on blinded exponentiations, a dynamic searchable symmetrical encryption scheme with supporting conjunctive queries is constructed. Then details about our proposal are introduced and a formal analysis of its security is presented. Theoretic analysis and experimental results show that, without significantly increasing the storage and computing costs, only documents identifications that match the query are revealed, which avoiding revealing more result patterns. In addition, by a single round of communication for conjunctive queries, the proposed scheme has lower communication overhead and higher search efficiency.

Key words　searchable symmetric encryption (SSE); conjunction queries; hidden vectors encryption; forward privacy; backward privacy

摘　要　对称可搜索加密（searchable symmetric encryption, SSE）因其较高的搜索效率得到了人们的广泛关注. 支持连接查询的对称可搜索加密方案可以提高方案的功能性. 然而现有大部分连接查询方案不支持对加密数据库的动态更新操作, 结合静态对称隐藏向量加密（symmetric hidden vectors encryption, SHVE）的定义, 提出支持动态数据更新的动态对称隐藏向量加密（dynamic SHVE, DSHVE）的定义, 并在此定义下构造了一种适用于可搜索加密应用场景的 DSHVE 方案. 在此基础上, 通过引入基于盲指数计算的不经意动态交叉标签, 设计具有前向和后向隐私的向量数据存取结构, 构造了支持连接查询的动态对称可搜索加密方案, 给出了方案的详细工作过程和正式的安全性分析. 理论分析和实验结果表明, 连接查询时, 该方案能够在不明显增加存储和计算开销的前提下, 有效避免泄露匹配文档索引之外的结果特征. 同时该方案能够仅通过单轮通信实现连接查询, 具有较低的通信开销和较高的搜索效率.

关键词　对称可搜索加密; 连接查询; 隐藏向量加密; 前向隐私; 后向隐私

收稿日期: 2022-11-03; **修回日期**: 2023-08-28
基金项目: 国家自然科学基金项目(61772547)
This work was supported by the National Natural Science Foundation of China (61772547).

中图法分类号 TP391

可搜索加密无需解密密文就能实现数据搜索. 其中对称可搜索加密(searchable symmetric encryption, SSE)方案因搜索过程采用对称算法，在数据量较大的云存储环境下往往具有更高的搜索效率，成为近年来的研究热点[1-2].

2000 年，Song 等人[3]首次提出了可搜索加密的概念，并基于对称密码算法，构造了一个对称可搜索加密方案. 文献 [4–8] 分别构造了正排索引、倒排索引、双向索引和树形索引结构的对称可搜索加密方案，从算法搜索效率、方案安全性等方面就对称可搜索加密方案进行了深入研究.

为提高方案的实用性，文献 [9] 设计了交叉索引结构，构造了支持连接查询的对称可搜索加密方案 OXT(oblivious cross tags). 该方案将连接查询请求分为sterm和xterm两部分，其中sterm用于实现单关键词查询，xterm用于从单关键词查询结果中去除多余的匹配文档索引. 在此基础上，人们从安全性和性能等方面构造了多种支持连接查询的可搜索加密方案[10-12]. 文献 [13] 基于 OXT 方案实现了一种支持析取操作的布尔关键词查询方案，进一步拓展了方案的场景适应能力.

然而，文献 [14] 研究指出，OXT 结构能够实现任意关键词之间的连接查询，但通常会向服务器泄露更多的结果特征. 理想情况下，连接查询对称可搜索加密仅向服务器泄露与查询关键词相匹配的整体结果特征(whole result pattern, WRP)，即与所有查询关键词匹配的文档数量和文档标识. 例如对于n个关键词的连接查询$w_1 \wedge \cdots \wedge w_n$(设$w_1$为“sterm”关键词)，关键词对结果特征(keyword-pair result pattern, KPRP)向服务器泄露了每个包含$(w_1, w_i)_{2\leqslant i\leqslant n}$关键词的文档集合$DB(w_1)\cap DB(w_i)$，但实际上仅需返回与所有$n$个查询关键词匹配的文档集合$\bigcup_{j=1}^{n} DB\left(w_j\right)$. 文献 [15] 利用 OXT 结构中泄露的结果特征信息恢复了所有连接查询关键词.

为此，文献 [14] 采用对称算法和伪随机函数替换复杂的公钥计算，定义了轻量级的对称隐藏向量加密(symmetric hidden vectors encryption, SHVE)原语，增强 OXT 方案在关联查询时的安全性. 但该方案仍是静态的，不支持密文数据库的动态更新(如添加、删除等)操作.

当前实现前向和后向隐私的动态可搜索加密方案大多仅支持单关键词查询. 文献 [16] 利用位图索引构建了基于双向索引结构的动态连接关键字查询方案. 文献 [17–18] 提出一种支持连接查询的对称可搜索加密方案，但方案仅具有前向隐私. 文献 [19] 基于已有动态对称可搜索加密方案，在 OXT 方案的基础上设计了不经意动态交叉标签(oblivious dynamic cross tags, ODXT)结构，构造了一种支持动态更新的对称可搜索加密方案，理论分析及实验结果表明，方案同时具有前向/后向隐私，且具有更小的通信开销. 但由于方案是基于 OXT 方案进行设计的，仍会泄露文献 [14] 指出的更多结果特征，文献 [20] 指出方案还存在查询结果不准确的问题.

本文在静态 SHVE 定义的基础上，给出了支持动态数据更新的动态对称隐藏向量加密(dynamic SHVE, DSHVE)的定义，并结合 ODXT 方案，构造了一种具有结果特征隐藏的连接查询动态对称可搜索加密方案 DSHVE-ODXT，重点在 3 个方面开展了相关研究：

1)构造了支持动态数据更新的 SHVE 方案. 结合静态 SHVE 的定义，给出了支持动态更新操作 SHVE 的定义. 在此基础上，构造了一种满足可搜索加密应用需求的 DSHVE 方案.

2)构造了支持动态连接查询的对称可搜索加密方案. 在 DSHVE 方案的基础上，通过引入 ODXT，设计了前/后向隐私的向量存取结构，构造了一种基于DSHVE的连接查询动态对称可搜索加密方案 DSHVE-ODXT. 方案保持了对称可搜索加密搜索效率高的特点，同时支持动态更新操作和连接查询，具有更好的实用性.

3)对 DSHVE-ODXT 方案进行了分析与测试. 结合该方案工作过程，从性能和安全性 2 个方面对该方案进行了理论分析，并在此基础上对方案的执行性能进行详细测试与分析.

1 基础知识

为方便描述，有关的通用符号定义如表 1 所示.

1.1 判定性 Diffie-Hellman 假设

定义 1. 判定性 Diffie-Hellman 假设. 设$\mathcal{G}$是阶为大素数p的群，g为群$\mathcal{G}$通过均匀采样确定的生成元. 判定性 Diffie-Hellman 假设定义为对所有多项式时间算法$\mathcal{A}$，有

$$|Pr\left[\mathcal{A}\left(g, g^{\alpha}, g^{\beta}, g^{\alpha\cdot\beta}\right)=1\right]-Pr\left[\mathcal{A}\left(g, g^{\alpha}, g^{\beta}, g^{\gamma}\right)=1\right]| \leqslant negl(\lambda),$$

其中$\alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p^*$，简称 DDH 假设.

Table 1 Definition of Notations

表 1 符号定义

符号	含义
λ	安全参数
id	文档标识
w	关键词
EDB	密文索引数据库
q	多关词连接查询语句
op	更新操作，$op\in\{\text{add},\text{del}\}$
$stoken$	单关键词搜索令牌
$xtoken$	交叉搜索令牌
$xtag$	XSet 中交叉标签
K	密钥
$negl(\lambda)$	安全参数λ上的可忽略函数
$[n]$	集合$\{1,2,\cdots,n\}$
$\lvert x\rvert$	x中元素的个数
$Pr[A]$	事件 A 发生的概率

1.2 对称加密算法

定义 2. 对称加密算法[21]. 设明文空间$\mathcal{M}$、密钥空间$\mathcal{K}$和密文空间$\mathcal{C}$, 称 3 个多项式时间算法组成的三元组(Gen,Enc,Dec)为对称加密算法, 其中:

1) $k\leftarrow Gen\left(1^{\lambda}\right)$. 输入秘密参数$\lambda$, 输出密钥$k\in\mathcal{K}$.

2) $ct\leftarrow Enc(k,mt)$. 输入密钥k、明文消息mt, 输出密文$ct\in\mathcal{C}$.

3) $mt\leftarrow Dec(k,ct)$. 输入密钥k、密文ct, 成功解密输出ct对应明文mt, 否则输出$\perp$.

定义 3. 正确性[21]. 若算法对任意明文$mt\in\mathcal{M}$, $k\leftarrow Gen\left(1^{\lambda}\right)$且$c\leftarrow Enc(k,mt)$, 满足

$$Pr[Dec(k,ct)=mt]=1,$$

则称对称加密算法满足正确性条件.

1.3 动态对称可搜索加密方案

定义 4. 动态对称可搜索加密方案. 设数据库由$DB=(id_i,W_i)_{i=1}^{d}$文档地址和关键词集构成, 其中id_i为第i文档, W_i为文档i包含的关键词集合, d为数据库中文档的数量, 称三元组概率多项式时间算法$(Setup,Update,Search)$为一个动态对称可搜索加密方案, 其中:

1)初始化算法$(K,\sigma,EDB)\leftarrow Setup\left(1^{\lambda},DB\right)$. 算法输入安全参数$\lambda$和明文索引数据库$DB$, 输出密钥$K$、关键词状态$\sigma$和密文的检索索引数据库$EDB$. 在动态对称可搜索加密方案中, 通常设$EDB=\varnothing$.

2)更新算法$(EDB',\sigma')\leftarrow Update(K,\sigma,w,ind,op;EDB)$. 输入密钥$K$、关键词状态$\sigma$、更新关键词$w$及对应索引$ind$, 更新操作类型$op$, 其中$op\in\{\text{add},\text{del}\}$, 输出更新后密文索引数据库$EDB'$、关键词状态$\sigma'$.

3)查询算法$DB(w)\leftarrow Search(K,\sigma,w;EDB)$. 输入密钥$K$、关键词状态$\sigma$和检索关键词$w$, 输出包含检索关键词$w$的所有文档标识$DB(w)$. 通常该算法包含客户端生成查询令牌和服务器根据令牌搜索查询结果 2 部分.

定义 5. 正确性[19]. 对任意数据库DB, 任意查询q, $Search$算法除可忽略概率外, 均输出$DB(q)$, 则称动态对称可搜索加密方案满足正确性条件.

2 动态对称隐藏向量加密（DSHVE）

隐藏向量加密能够实现整体结果特征隐藏, 且支持灵活的连接查询和较高的查询效率. 在正式开始对称可搜索加密方案构造前, 首先给出支持动态更新隐藏向量加密的形式化定义, 进而详细介绍一种动态对称可搜索加密的构造方法.

2.1 形式化定义

设Σ为属性的有限集, $*$为不包含在集合Σ的通配符, 表示“不关心”的值. 设$\Sigma_*=\Sigma\cup\{*\}$, 其中Σ的典型值为有限域$\mathbb{Z}_p$(p为素数).

设$\mathcal{P}^{\text{SHVE}}:\Sigma^m\rightarrow\{0,1\}$表示一个谓词族, 在所有$\boldsymbol{v}=(v_1,v_2,\cdots,v_m)\in\Sigma_*^m$中存在谓词$P_{\boldsymbol{v}}^{\text{SHVE}}\in\mathcal{P}^{\text{SHVE}}$, 满足对属性向量$\boldsymbol{x}=(x_1,x_2,\cdots,x_m)\in\Sigma^m$, 有

$$P_{\boldsymbol{v}}^{\text{SHVE}}(\boldsymbol{x})=\begin{cases}1, \forall 1\leqslant i\leqslant m\left(v_i=x_i\text{或}v_i=*\right),\\ 0, \text{其他},\end{cases}$$

即在所有非通配符$*$对应向量$\boldsymbol{x}$与$\boldsymbol{v}$匹配. 参数m代表 SHVE 的宽度.

定义 6. DSHVE. DSHVE 定义为 4 个概率多项式时间(probabilistic polynomial-time, PPT)算法的集合:

1) $msk\leftarrow Setup(\lambda)$. 输入安全参数$\lambda$, 输出主密钥$msk$, 并定义消息空间$\mathcal{M}$, 初始化属性向量$\boldsymbol{x}$和密文集合$c$.

2) $c'\leftarrow UpdateEnc(msk,\mu\in\mathcal{M},c,p,op\in\{\text{add},\text{del}\})$. 输入主密钥$msk$、消息$\mu$、当前密文集合$c$、更新元素在索引向量$\boldsymbol{x}$中的位置$p$以及更新操作$op$, 输出更新后对应密文$c'$.

3) $\boldsymbol{s}\leftarrow KeyGen(msk,\boldsymbol{v}\in\Sigma_*^m)$. 输入谓词向量$\boldsymbol{v}$, 主密钥$msk$, 输出解密密钥$\boldsymbol{s}$.

4) $\mu\leftarrow Query(\boldsymbol{s},c)$. 输入属性向量$\boldsymbol{x}$对应密文$c$, 谓词向量$\boldsymbol{v}$对应解密密钥$\boldsymbol{s}$, 如果$P_{\boldsymbol{v}}^{\text{SHVE}}(\boldsymbol{x})=1$, 输出消息$\mu$.

定义 7. 正确性. 设 DSHVE 表示一个动态对称隐藏向量加密方案, 若对所有安全参数λ, 所有$(\mu,\boldsymbol{x})\in$

$\mathcal{M}\times\Sigma^m$和谓词向量$\boldsymbol{v}$，首先执行$DSHVE.Setup(\lambda)$获得$msk$，反复运行$UpdateEnc(msk,\mu\in\mathcal{M},c,p,op\in\{\text{add},\text{del}\})$后最终获得$c$，$KeyGen(msk,\boldsymbol{v}\in\Sigma_*^m)$获得$\boldsymbol{s}$，如果$P_v^{\text{SHVE}}(\boldsymbol{x})=1$，则$Query(\boldsymbol{s},c)=\mu$，否则

$$Pr\left[Query(\boldsymbol{s},c)=\perp\right]=1-negl(\lambda),$$

则称DSHVE满足正确性条件.

DSHVE的安全性是通过真实实验和理想实验的不可区分性来定义的. 结合定义5，给出真实实验和理想实验的详细过程.

1)真实实验. 实验包括挑战者和自适应PPT攻击者$\mathcal{A}$，其交互过程为：

①Setup阶段. 挑战者选择安全参数λ，运行$Setup(\lambda)$输出msk和明文空间$\mathcal{M}$. 并将$\mathcal{M}$发送给攻击者$\mathcal{A}$.

攻击者$\mathcal{A}$自适应选择执行更新阶段或查询阶段1.

②更新阶段. 攻击者$\mathcal{A}$自适应选择位置$p(p\in[m])$以及更新操作op，并将元组(p,op)发送给挑战者. 挑战者接收(p,op)，执行$UpdateEnc$，返回攻击者$\mathcal{A}$更新后的密文c'.

③查询阶段1. 攻击者$\mathcal{A}$自适应选择谓词向量$\boldsymbol{v}_j$，$j\in\left[q_1\right]$. 挑战者利用$\boldsymbol{v}_j$和msk运行$KeyGen$，返回相应的解密密钥$\boldsymbol{s}_j$.

设在这一阶段攻击者$\mathcal{A}$一共执行了q次查询操作，接着攻击者$\mathcal{A}$执行挑战阶段和查询阶段2.

④挑战阶段. 攻击者$\mathcal{A}$输出消息$\mu\in\mathcal{M}$. 挑战者利用msk，$\boldsymbol{x}$，μ运行$UpdateEnc$，得到密文c并发送给$\mathcal{A}$.

⑤查询阶段2. 攻击者$\mathcal{A}$运行与查询阶段1相同的协议，收到$\boldsymbol{s}_j$，$q+1\leqslant j\leqslant q_2$.

设$r_{\mathcal{A}}$表示$\mathcal{A}$在真实实验中生成的随机比特，使用符号$View_{\mathcal{A},\text{Real}}$表示在上述真实实验中攻击者$\mathcal{A}$所获得的有用信息$(\mathcal{M},c,\left\{\boldsymbol{v}_j\right\}_{j\in q_2},r_{\mathcal{A}})$.

2)理想实验. 实验包括PPT模拟器$\mathcal{S}$和自适应PPT攻击者$\mathcal{A}$，模拟器$\mathcal{S}$与攻击者$\mathcal{A}$交互过程为：

①Setup阶段. 挑战者选择安全参数λ，模拟器$\mathcal{S}$均匀随机选择$msk\xleftarrow{\$}\{0,1\}^\lambda$，定义明文空间$\mathcal{M}$，并将$\mathcal{M}$发送给攻击者$\mathcal{A}$.

攻击者$\mathcal{A}$自适应选择执行更新阶段或查询阶段1.

②更新阶段. 攻击者$\mathcal{A}$自适应选择位置$p(p\in[m])$以及更新操作op. 模拟器$\mathcal{S}$接收(p,op)，返回攻击者$\mathcal{A}$新的密文c'.

③查询阶段1. 攻击者$\mathcal{A}$自适应选择谓词向量$\boldsymbol{v}_j$，$j\in\left[q_1\right]$. 模拟器$\mathcal{S}$返回对应的解密密钥$\boldsymbol{s}_j$.

设在这一阶段攻击者$\mathcal{A}$一共执行了q次查询操作，接着攻击者$\mathcal{A}$执行挑战阶段和查询阶段2.

④挑战阶段. 攻击者$\mathcal{A}$输出消息$\mu\in\mathcal{M}$. 模拟器$\mathcal{S}$返回$\boldsymbol{x}$和μ对应的挑战密文c.

⑤查询阶段2. 攻击者$\mathcal{A}$运行与查询阶段1相同的协议，收到$\boldsymbol{s}_j$，$q+1\leqslant j\leqslant q_2$.

同样，设$r_{\mathcal{A}}$表示$\mathcal{A}$在理想实验中生成的随机比特，使用符号$View_{\mathcal{A},\mathcal{S}}$表示在理想实验中攻击者$\mathcal{A}$所获得的有用信息$(\mathcal{M},c,\left\{\boldsymbol{v}_j\right\}_{j\in q_2},r_{\mathcal{A}})$.

定义8. 自适应安全. 设DSHVE表示一个动态对称隐藏向量加密方案，对任意PPT攻击者$\mathcal{A}$分辨真实实验和理想实验中所取得的攻击优势为$Adv_{\mathcal{A}}^{\text{DSHVE}}(\lambda)=\left|Pr[Real_{\mathcal{A}}(\lambda)=1]\right|-\left|Pr[Idea_{\mathcal{A},\mathcal{S}}(\lambda)=1]\right|$，若存在一个模拟器$\mathcal{S}$，对所有PPT攻击者$\mathcal{A}$，函数$Adv_{\mathcal{D},\mathcal{A}}^{\text{DSHVE}}(\lambda)$在安全参数$\lambda$下是可忽略的，则称DSHVE是自适应安全的.

2.2 详细构造

设λ为安全参数，$F_0:\{0,1\}^\lambda\times\{0,1\}^{\lambda+\text{lb}\lambda}\to\{0,1\}^{\lambda+\text{lb}\lambda}$为一个伪随机函数(pseudo random function, PRF)，$(Sym.Enc,Sym.Dec)$表示一个IND-CPA安全的对称加密方案，其中密钥空间和明文空间均为$\{0,1\}^{\lambda+\text{lb}\lambda}$. $\boldsymbol{x}=(x_1,x_2,\cdots,x_m)\in\Sigma_*^m$表示一个宽度为$m$的属性向量，$op\in\{\text{add},\text{del}\}$表示对属性向量的添加和删除操作. 定义与$\boldsymbol{x}$对应的密文集合$c=\left\{\{c_l\}_{op,l\in[m]}\right\}=\{\{c_l\}_{\text{add},l\in[m]}\cup\{c_l\}_{\text{del},l\in[m]}\}$，即集合$c$由添加操作密文集合$\{c_l\}_{\text{add},l\in[m]}$和删除更新的密文集合$\{c_l\}_{\text{del},l\in[m]}$组成，最多可包含$2m$个元素.

在动态对称可搜索加密的应用场景中，文档索引的更新过程通常是先执行添加操作，然后在之后的某个时间执行删除操作. 因此不妨设向量$\boldsymbol{x}$中的元素$x_l(l\in[m])$满足的删除操作总是在添加操作之后发生. 支持动态更新操作的DSHVE的详细构造过程为：

1)$(msk,\mathcal{M})\leftarrow Setup(1^\lambda,m)$. 输入安全参数$\lambda$，SHVE宽度$m$，通过算法均匀采样得到$msk\xleftarrow{\$}\{0,1\}^\lambda$，并定义明文空间$\mathcal{M}=\{\text{'True'}\}$，$\boldsymbol{x}=(x_1,x_2,\cdots,x_i,\cdots,x_m)$，其中$x_{i\in[m]}=*$，$c=\left\{\{c_l=\perp\}_{l\in[2m]}\right\}$，输出$(msk,\mathcal{M})$.

2)$c'\leftarrow UpdateEnc(msk,\mu=\text{'True'},c=\{\{c_l\}_{op,l\in[2m]}\},p\in[m],op\in\{\text{add},\text{del}\})$. 输入主密钥$msk$、消息$\mu$、当前密文$c$和待更新的索引位置$p$、更新操作$op$，计算：

$$c_{op,p}=\begin{cases}F_0(msk,1\|p),\text{若}op=\text{add},\\F_0(msk,1\|p)\oplus F_0(msk,0\|p),\text{若}op=\text{del},\end{cases}$$

输出更新后的密文$c'=\left\{\{c_l\}_{op,l\in[m]\setminus p}\right\}\cup\left\{c_{op,p}\right\}$，即相比输入密文$c$，$c'$更新索引位置$p$处的元素为$c_{op,p}$，其他元素保持不变，且从$c_p$的计算过程可以看出，$c_p$取值

与该元素的历史状态无关. 未对索引向量 $\boldsymbol{x}\in\Sigma_*^m$ 对应位置进行插入操作时, 其对应密文元素为 $\perp$.

3) $\boldsymbol{s}\leftarrow KeyGen(msk,\boldsymbol{v}\in\Sigma_*^m)$. 输入谓词向量 $\boldsymbol{v}=(v_1,\cdots,v_m)$ 和主密钥 msk, 若用 $S=\{l_j\in[m]\,|\,v_{l_j}\neq *\}$ 表示 $\boldsymbol{v}$ 中不包含通配符的位置集合, 设这些位置中 $l_1<l_2<\cdots<l_{|S|}$, 算法均匀样本 $K\xleftarrow{\$}\{0,1\}^{\lambda+\mathrm{lb}\lambda}$, 计算

$$d_0=\oplus_{j\in[|S|]}\left(F_0(msk,v_{l_j}\|l_j)\right)\oplus K,$$

$$d_1=Sym.Enc\left(K,0^{\lambda+\mathrm{lb}\lambda}\right),$$

算法最终输出解密密钥:

$$\boldsymbol{s}=(d_0,d_1,S).$$

4) 'True'或 $\perp\leftarrow Query(\boldsymbol{s},c)$. 查询算法输入为密文和解密密钥, 并对变量进行编译:

$$c=\left\{\{c_l\}_{op,l\in[m]}\right\},$$

$$\boldsymbol{s}=(d_0,d_1,S),$$

其中 $S=\{l_1<l_2<\cdots<l_{|S|}\}$. 计算:

$$K'=\left(\oplus_{j\in[|S|]}c_{op,l_j}\right)\oplus d_0,$$

然后解密算法计算

$$\mu'=Sym.Dec(K',d_1),$$

若 $\mu'=0^{\lambda+\mathrm{lb}\lambda}$, 算法输出'True', 否则输出 $\perp$.

从解密计算过程可以看出, μ' 与元素在密文集合 c 的位置无关.

正确性证明过程为:

证明. 设与密文 $c=\left\{\{c_l\}_{op,l\in[m]}\right\}$ 对应的索引向量 $\boldsymbol{x}=(x_1,x_2,\cdots,x_m)$, 且 $\boldsymbol{s}=(d_0,d_1,S)$ 为谓词向量 $\boldsymbol{v}=(v_1,v_2,\cdots,v_m)$ 对应的解密密钥. 则根据 *UpdateEnc* 算法工作过程可知:

若 $x_i=*$, 则 $c_{\mathrm{add},i}=c_{\mathrm{del},i}=\perp$;

若 $x_i=1$, 则 $c_{\mathrm{del},i}=\perp$, 因此 $c_{\mathrm{add},i}\oplus c_{\mathrm{del},i}=F_0(msk,1\|i)=F_0(msk,x_i\|i)$;

若 $x_i=0$, 则 $c_{\mathrm{add},i}\oplus c_{\mathrm{del},i}=F_0(msk,1\|i)\oplus F_0(msk,1\|i)\oplus F_0(msk,0\|i)=F_0(msk,0\|p)=F_0(msk,x_i\|i)$,

即 x_i 经过添加和删除操作后对应实际密文元素 c_i 满足

$$c_i=c_{\mathrm{add},i}\oplus c_{\mathrm{del},i}=\begin{cases}F_0(msk,x_i\|i), & \text{若 } x_i\neq *,\\ \perp, & \text{若 } x_i=*,\end{cases}$$

设 $S=\{l_1,l_2,\cdots,l_{|S|}\}$, 则:

1) 若 $P_v^{\mathrm{SHVE}}(\boldsymbol{x})=1$, 则对所有 $j\in[|S|]$ 必须满足 $v_{l_j}=x_{l_j}$, 那么

由于 $x_{l_j}=v_{l_j}\neq *$, 则对所有 $j\in[|S|]$ 均有 $c_{l_j}=F_0(msk,x_{l_j}\|l_j)=F_0(msk,v_{l_j}\|l_j)$,

即 $K'=\left(\oplus_{j\in[|S|]}c_{op,l_j}\right)\oplus d_0=\left(\oplus_{j\in[|S|]}c_{l_j}\right)\oplus d_0=K$.

因此, $\mu'=Sym.Dec(K',d_1)=0^{\lambda+\log\lambda}$ 成立.

2) 若 $P_v^{\mathrm{SHVE}}(\boldsymbol{x})=0$, 则存在 $j\in[|S|]$, 满足 $v_{l_j}\neq x_{l_j}$, 则

若 $x_{l_j}=*$, 则 $c_{l_j}=\perp\neq F_0(msk,v_{l_j}\|l_j)$;

若 $x_{l_j}\neq *$ 且 $x_{l_j}\neq v_{l_j}$, 则 $c_{l_j}=F_0(msk,x_{l_j}\|l_j)\neq F_0(msk,v_{l_j}\|l_j)$ 成立, 即存在 $j\in[|S|]$, 使 $\left(\oplus_{j\in[|S|]}c_{\mathrm{add},l_j}\right)\oplus\left(\oplus_{j\in[|S|]}c_{\mathrm{del},l_j}\right)\neq F_0(msk,v_{l_j}\|l_j)$ 成立, 进而使得解密时 $K'\neq K$. 因此除非发生可忽略的事件, 否则 $\mu'\neq 0^{\lambda+\mathrm{lb}\lambda}$, 解密算法返回失败标志 $\perp$.

即 DSHVE 满足正确性条件. 证毕.

3 方案构造

结合 DSHVE 方案的详细构造过程, 设计一种支持连接查询的动态对称可搜索加密方案 DSHVE-ODXT, 本节从该方案总体结构、TSet 和 XSet 结构等方面介绍方案的设计思想, 详细描述方案的工作过程.

3.1 方案总体结构

DSHVE-ODXT 由一个代表数据拥有者和使用者的客户端和一个服务器组成. 其中客户端能够将用户数据加密成密文, 并动态更新至密文服务器. 服务器根据用户请求, 执行密文搜索操作, 并向客户端返回搜索结果. 假设服务器为"诚实且好奇"的, 即服务器能够忠实运行客户的请求, 但会尽可能窥探用户的隐私信息.

DSHVE-ODXT 以关键词 w 为索引的 TSet 结构和以文档 id 为索引的 XSet 结构组成, 其中 TSet 主要用于实现安全的动态单关键词查询, XSet 则主要用于实现连接查询. 图 1 所示为一个连接查询工作过程示意图.

客户端一次接受用户输入多个连接查询关键词序列 $q=(w_1\wedge w_2\wedge\cdots\wedge w_n)$, 从查询序列中选择更新次数最少的关键词, 不失一般性, 可设关键词序列 q 中更新次数最少的关键词为 w_1, 则客户端为 w_1 生成 TSet 查询令牌 *stoken*, 并为其他关键词 $(w_2,w_3,\cdots,w_n)$ 分别生成 XSet 添加和删除操作查询令牌 $(xtoken_2,xtoken_3,\cdots,xtoken_i,\cdots)$, 将 *stoken* 与 $(xtoken_2,xtoken_3,\cdots,xtoken_i,\cdots)$ 合并发送至服务器. 服务器根据相应令牌执行搜索算法, 计算满足条件的结果文件集合, 并将结果返回客户端.

3.2 不经意动态交叉标签

在构造支持动态连接查询方案前, 首先给出一种记录更新状态的数据结构, 即动态交叉标签 (dynamic cross-tags, DXT).

定义 9. DXT. 设关键词 w、文档标识 id 通过每个索引关键词对 (id,w) 能够计算得到一个唯一的存储

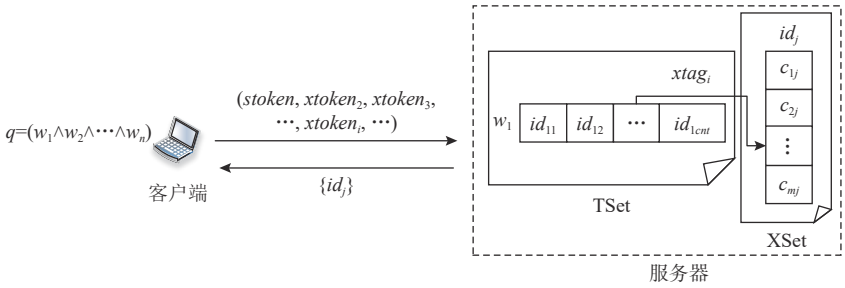


Fig. 1 The process of conjunctive queries

图 1 连接查询过程

标签$xtag_{id,w}$，且标签$xtag_{id,w}$代表的位置存储 3 种值：

① $(\perp,\perp)$表示(id,w)没有插入和删除操作，

② $(c_{\text{add}},\perp)$表示(id,w)插入后未删除，

③ $(c_{\text{add}},c_{\text{del}})$表示$(id,w)$插入后删除.

其中$\perp$表示对应标签取值为空，c_{add}和c_{del}分别表示插入和删除操作时写入的操作标志. 这种记录数据操作标签数值对称为 DXT.

为了隐藏数据不同操作类型之间的对应关系，即隐藏“删除操作删除了哪个添加操作”信息，为 DXT 中每类操作标志$c_{op}\in\{c_{\text{add}},c_{\text{del}}\}$定义一个对应的数据标签$xtag_{id,w,op}$.

设g是素数阶p的群$\mathcal{G}$上的生成元，F_p是$\mathbb{Z}_p^*$上的 PRF，K_X和K_Y是均匀采样得到的 PRF 伪随机函数F_p的密钥，操作$op\in\{\text{add},\text{del}\}$表示数据插入和删除操作，关键词$w$的第$cnt$次更新操作$(op,(id,w))$对应数据标签$xtag_{id,w,op}$计算方式为

$$xtag_{id,w,op}=g^{F_p(K_X,w)F_p(K_Y,id\|op)},$$

这里$xtag_{id,w,op}$的计算过程与$(op,(id,w))$相关. 文档关键词对(id,w)的添加数据标签和删除数据标签分别记为$xtag_{id,w,\text{add}}$和$xtag_{id,w,\text{del}}$.

在此基础上，引入盲指数(blinded exponentiations)，定义交叉搜索令牌$xtoken$和盲因子α，设计一种标签盲化的不经意动态交叉标签，进一步减少交叉搜索时的通信次数和需要传递和通信令牌数量. 设计的核心思想是使交叉搜索令牌$xtoken$的计算仅与搜索关键词相关，而与具体的文档无关. 这样通过统一的交叉搜索令牌$xtoken$结合存储在服务器上的盲因子α，计算得到所有文档的实际数据标签.

若客户端在插入新元素时记录并更新当前关键词的插入次数$UpdateCnt$，服务器插入关键词信息时，在 TSet 结构中存储盲元素$\alpha_{id,w,op}=F_p(K_Y,id\|op)\cdot(F_p(K_Z,w\|cnt))^{-1}$. 若用$q=(w_1\wedge w_2\wedge\cdots\wedge w_n)$表示连接查询关键词，且$w_1$对应更新次数$cnt$最少，查询时客户端根据查询关键词计算查询令牌

$$xtoken_{i,cnt}=g^{F_p(K_X,w_1)F_p(K_Z,w_i\|cnt)},$$

其中$i\in\{2,3,\cdots,n\}$. 这里$xtoken_{i,cnt}$计算与具体文档无关.

服务器接收到查询令牌$xtoken_{i,cnt}$后，计算

$$xtag_{id,w,op}=\left(xtoken_{i,cnt}\right)^{\alpha_{id,w,op}}=g^{F_p(K_X,w)F_p(K_Y,id\|op)},$$

恢复动态交叉标签中对应的数据地址$xtag_{id,w,op}$.

3.3 关键结构

DSHVE-ODXT 方案采用 2 个专门设计的数据存取结构实现单关键词和交叉多关键词查询，即 TSet 和 XSet 结构.

1）TSet 设计

TSet 为一个以关键词为索引组成的“倒排索引”结构，实现前向/后向隐藏的动态单关键词搜索，其核心数据为一个数据集合，记为$tset$. 图 2 所示为服务器上存储的以关键词w_i为索引，更新次数为cnt的文档列表，其中$stoken$表示客户端发送的与查询关键词对应的搜索令牌，每个文档节点id_{ij}除存储单关键词搜索相关的信息val外，还需要额外存储支持 XSet 查询的盲因子$\alpha_{j,op}$.

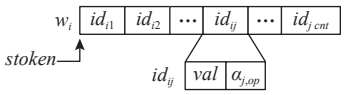


Fig. 2 The storage structure of TSet data

图 2 TSet 数据存储结构

关键词w_i的第j次更新文档对应盲因子$\alpha_{j,op}$按如下方式计算：

$$\alpha_{j,op}=\alpha_{id_j,w_i,op}=\begin{cases}F_p(K_Y,id_j\|\text{add})(F_p(K_Z,w_i\|cnt))^{-1}, & op=\text{add},\\ F_p(K_Y,id_j\|\text{del})(F_p(K_Y,id_j\|\text{add}))^{-1}, & op=\text{del}.\end{cases}$$

2）XSet 实现

XSet 设计为一个存储标签-键值$(xtag,c)$对的存储空间，记为$xset$，其中$xset\left[xtag\right]=c$表示数据c写入数据集合$xset$，$c=xset\left[xtag\right]$表示从数据集$xset$中读取标签$xtag$对应数据c.

方案中实际存储数据的$xtag$即为 ODXT 中的 DXT. 连接查询时，首先通过在 XSet 中恢复出 ODXT 值，然后运用动态对称隐藏向量加密搜索与整体匹配的结果文档集合，并返回匹配的结果集合. 图 3 所示为服务器利用 XSet 进行连接查询时的搜索过程.

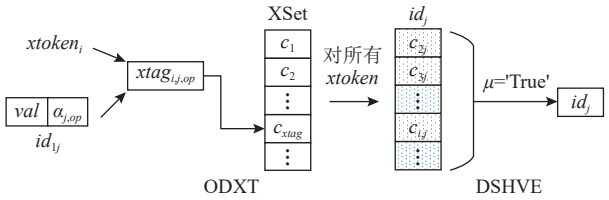


Fig. 3 The search process when the server uses XSet for conjunctive queries

图 3 服务器利用 XSet 进行连接查询时的搜索过程

设$xtoken_i$表示连接查询关键词序列$q=(w_1\wedge w_2\wedge\cdots\wedge w_n)$中第$i(i\in\{2,3,\cdots,n\})$个关键词$w_i$的交叉查询令牌，$id_{1j}$表示 TSet 中关键词$w_1$对应列表的第$j(1\leqslant j\leqslant cnt$，$cnt$表示$w_1$的更新次数$)$个文档节点，$\alpha_{j,op}$表示 XSet 中$id_{1j}$节点上存储的盲因子，则根据 ODXT 构造过程，可得

$$xset\left[xtag_{i,j,op}\right]=\begin{cases}c_{\text{add}}\text{或}c_{\text{del}}, \text{若} xtag_{i,j,op}\in S_{xset},\\ \perp, \text{其他},\end{cases}$$

其中$xtag_{i,j,\text{add}}=(xtoken_i)^{\alpha_{j,\text{add}}}$，$xtag_{i,j,\text{del}}=\left(xtag_{i,j,\text{add}}\right)^{\alpha_{j,\text{del}}}$.

针对id_{1j}节点上存储的盲因子$\alpha_{j,op}$计算所有$xtoken_i(i\in\{2,3,\cdots,n\})$，并最终得到与文档索引$id_{1j}$对应的所有 DXT 值.

3.4 方案描述

基于上述思想，DSHVE-ODXT 由 3 个 PPT 算法 *Setup*, *Update*, *Search* 组成.

算法 1. *Setup* 算法：$EDB\leftarrow Setup(\lambda)$.

输入：安全参数λ；

输出：空的密文数据库 EDB.

客户端：

① 均匀随机采样得到 PRF F的密钥K_{T}；

② 均匀随机采样得到 PRF F_p的密钥K_X, K_Y, K_Z；

③ 均匀随机采样得到 DSHVE 的密钥K_{H}；

④ 初始化st, $tset$, $xset$为$\varnothing$；

⑤ 客户端保存$sk=\{K_{\text{T}},K_X,K_Y,K_Z,K_{\text{H}}\}$, st；

⑥ 令$EDB=\{tset,xset\}$；

⑦ 发送EDB至服务器.

客户端接受输入安全参数λ，分别通过均匀随机采样，生成方案需要使用的密钥集$sk=\{K_{\text{T}},K_X,K_Y,K_Z,K_{\text{H}}\}$，置关键词状态集$st$、密文索引数据库$EDB=(tset,xset)$为$\varnothing$. sk, st直接存储在客户端，EDB发送至服务器保存.

算法 2. *Update* 算法：$(st',EDB')\leftarrow Update(sk,st,op,(id,w);EDB)$.

输入：客户端输入为方案密钥sk，关键词状态集st，更新操作类型op，文档关键词对(id,w)，服务器输入为密文索引数据库EDB；

输出：更新后的关键词状态集st'和密文索引数据库EDB'.

客户端：

① 编译得到$sk=\{K_{\text{T}},K_X,K_Y,K_Z,K_{\text{H}}\}$, st；

② 若$st[w]=\varnothing$，则令$st[w]=(len(st),0)$；

③ 编译$st[w]$，得到w_{id},w_{cnt}；

④ $w_{cnt}=w_{cnt}+1$；

⑤ $st[w]=(w_{id},w_{cnt})$；

⑥ $addr=F(K_{\text{T}},w\|w_{cnt}\|0)$；

⑦ $val=(id\|op)\oplus F(K_{\text{T}},w\|w_{cnt}\|1)$；

⑧ $\alpha=F_p(K_Y,id\|op)(F_p(K_Z,w\|w_{cnt}))^{-1}$；

⑨ $xtag=g^{F_p(K_X,w)F_p(K_Y,id\|op)}$；

⑩ if op=add

$c=F(K_H,1\|w_{id})$；

⑪ else if op=del

计算$c=F(K_{\text{H}},1\|w_{id})\oplus F(K_{\text{H}},0\|w_{id})$；

end if

⑫ end if

⑬ 发送$(addr,val,\alpha,(xtag,c))$至服务器.

服务器：

① 编译得到$EDB=\{tset,xset\}$；

② $tset[addr]=(val,\alpha)$；

③ $xset\left[xtag\right]=c$.

客户端通过编译本地保存的密钥集sk和关键词集st，恢复方案各参数. 然后通过行⑥~⑧生成 TSet 中存储数据，即$tset[addr]=(val,\alpha)$，行⑨~⑫按照 ODXT 设计思想生成 XSet 中 DXT，即$xset\left[xtag\right]=c$. 服务器接收到客户端发过来的相关参数后，直接更新密文索引数据库EDB.

算法 3. *Search*算法：$DB(q)\leftarrow Search(sk,st,q;EDB)$.

输入：客户端输入为方案密钥sk，关键词状态集st，连接查询q，服务器输入为密文索引数据库EDB；

输出：匹配的文档索引集$DB(q)$.

Step 1. 查询令牌生成.

客户端：

① 编译得到$sk=\{K_{\text{T}},K_X,K_Z,K_{\text{H}}\}$, st；

② 将谓词向量集合S初始化为$\varnothing$；

③ 通过st找到更新次数最少的关键词(不失一般性，设为w_1)；

④ 若查询序列$q=(w_1\wedge w_2\wedge\cdots\wedge w_n)$中存在关键词不在$st$列表中，返回查询结果为$\varnothing$，否则继续执行；

⑤ 通过编译st，得到每一个查询关键词w_i的$w_{i,id}$，并添加至向量集S；

⑥ 将$stokenList$初始化为$\varnothing$；

⑦ 将$xtokenList_1,\cdots,xtokenList_{UpdateCnt[w_1]}$初始化为空列表；

⑧ 编译$st[w_1]$，得到$w_{1,id},w_{1,cnt}$；

⑨ for j=1 to $w_{1,cnt}+1$

$saddr_j=F(K_T,w_1\|j\|0)$；

$stokenList=stokenList\cup\{saddr_j\}$；

for $i=1$ to n

$xtoken_{i,j}=g^{F_p(K_X,w_i)F_p(K_Z,w_1\|j)}$；

$xtokenList_j=xtokenList_j\cup\{xtoken_{i,j}\}$；

end for

随机选择一个位置插入$xTagList_j$；

⑩ end for

⑪ 均匀随机采样得到对称加密密钥K；

⑫ 计算$d_0=\oplus_{j\in[|S|]}\left(F_0(msk,v_{l_j}\|l_j)\right)\oplus K$；

⑬ 计算$d_1=Sym.Enc\left(K,0^{\lambda+\mathrm{lb}\lambda}\right)$；

⑭ 发送到$(stokenList,xtokenList_1,\cdots,xtokenList_{[w_1]},d_0,d_1)$服务器.

Step 2. 密文搜索.

服务器：

① 编译得到$EDB=(tset,xset)$和$stokenList,xtokenLists,\boldsymbol{d}_0,\boldsymbol{d}_1$；

② 初始化$sEOpList$为空列表；

③ for $j=1$ to $stokenList.size$

$cnt_j=1$；

$(sval_j,\alpha_j)=tset[stokenList[j]]$；

初始化集合S为$\varnothing$；

for $i=1$ to n

$xtoken_{i,j}=xtoken_j[i]$；

计算$xtag_{i,j}=\left(xtoken_{i,j}\right)^{\alpha_j}$；

$S=S\cup\{xtag_{i,j}\}$；

end for

计算$K'=\left(\oplus_{j\in[|S|]}c_{op,l_j}\right)\oplus d_0$；

计算$\mu'=Sym.Dec(K',\boldsymbol{d}_1)$；

if $\mu'=0^{\lambda+\mathrm{lb}\lambda}$

$sEOpList=sEOpList\cup\{j,sval_j\}$；

end if

④ end for

⑤ 发送$sEOpList$到客户端.

Step 3. 最终结果输出.

客户端：

① 初始化$IdList$为空列表；

② for $\ell=1$ to $sEOpList.size$

令$(j,sval_j)=sEOpList[\ell]$；

恢复$(id_j\|op_j)=sval_j\oplus F(K_T,w_1\|j\|1)$；

$IdList=IdList\cup\{id_j\}$；

③ end for

④ 输出$IdList$.

多关键词的连接查询过程仅需进行1轮交叉就可实现，主要包括客户端生成查询令牌、服务器执行密文搜索和客户端输出最终结果3个步骤.

1)查询令牌生成(Step 1). 客户端接受用户输入连接查询关键词q，利用本地保存的关键词状态集st，找出更新次数最少的关键词(不妨设为w_1). 若发现某个查询关键词w_i不更新列表，即$w_i\notin st$，则说明w_i之前未执行过更新操作，因此直接返回查询结果为空(行④).

通过st恢复连接查询q中对应的每个关键词w_i的惟一标识，记为$w_{i,id}$，存入谓词向量集合S中(行⑤)，并在行⑪~⑬生成DSHVE解密密钥向量(d_0,d_1)，与行⑨⑩生成的查询令牌一起，发送至服务器.

2)密文搜索(Step 2). 服务器接收查询令牌和解密密钥向量后，按照ODXT设计思想，分别从XSet对应位置取出存储索引密文$sval_j$和盲因子α_j，并与对应交叉令牌$xtoken_{i,j}$恢复需要读取的TSet数据存储标签$xtag$，加入谓词向量集合S，并通过行③，执行DSHVE查询操作，返回符合查询条件的文档在TSet中的位置和密文数据.

3)最终结果输出(Step 3). 客户端利用服务器返回的信息恢复出真实文档索引集合，并计算最终结果$IdList$，即$DB(q)$.

4 方案分析

结合方案详细构造，从性能与安全性2个方面对其进行分析，其中性能分析包括服务器及客户端存储开销、通信开销和计算开销等，而安全性着重围绕方案更新泄露和搜索泄露的信息展开.

4.1 性能分析

存储开销、通信开销和计算开销是影响方案性

能的重要因素，下面从这 3 个方面对方案的性能进行分析.

1）存储开销

方案的存储开销包括服务器和客户端 2 部分.

服务器在方案初始化阶段，实际存储的数据集 $tset$ 和 $xset$ 为 $\varnothing$. 随后每执行一个更新操作，服务器需要在集合 $xset$ 中存储 $(addr,(val,\alpha))$，在集合 $tset$ 中存储 $(xtag,c)$，占用空间大小为方案执行更新操作次数的整数倍，即服务器存储开销约为 $O\left(N_{Update}\right)$，N_{Update} 为方案执行更新操作的次数.

客户端仅需要存储相关密钥集 sk 和关键词更新状态集 st，其中 st 在方案初始化时为 $\varnothing$，每新增一个索引关键词，需要为其分配一定存储空间. 因此客户端存储空间实际随密文索引关键词个数线性增加，即客户端存储空间开销约为 $O(N_w)$，其中 N_w 为关键词个数.

2）通信开销

方案仅需经过 1 轮通信就可以完成支持连接查询的密文搜索，通信开销与连接查询中更新次数最少关键词 w_1 的更新次数和查询关键词个数相关，即通信开销可表示为 $O(n|Update(w_1)|)$，其中 n 表示查询关键词个数，$|Update(w_1)|$ 表示更新次数最少关键词 w_1 的更新次数.

真实应用场景中，使用频率最低的关键词与更新次数最少关键词通常是一致的，因此在数据量较大时，通过选择更精确的查询关键词有望大大提高大数据集上的查询效率.

DSHVE 构造过程中，服务器执行 $DSHVE.Search$ 的查询算法，可以直接得到匹配查询的文档索引密文，不需要返回所有信息，这极大提高了返回时的通信量. 实际上在多关键词查询应用场景下，返回的文档索引数最大值为更新次数最少关键词 w_1 的匹配文档数，即服务返回查询结果的通信开销上限可表示为 $O(|Update(w_1)|)$.

3）计算开销

由于在 DSHVE 中查询谓词向量 $\boldsymbol{v}$ 中非通配符位置相对索引向量 $\boldsymbol{x}$ 是稀疏排列的，即在大多数应用场景下，连接查询关键词的个数远小于当前数据库中关键词的个数. 因此 DXT 数据盲化处理和标签恢复时，指数运算次数仅为当前连接查询关键词数量的 2 倍（添加和删除操作），计算开销通常不大，具有较高的查询效率.

4.2 安全性

下面分别对 DSHVE 和 DSHVE-ODXT 的安全性进行分析.

4.2.1 DSHVE 方案安全性

定理 1. 设 F_0 是安全的伪随机函数，对称加密方案 $(Sym.Enc,Sym.Dec)$ 为一个理想的加解密算法，且属性向量删除更新操作总是在添加操作之后发生，则在随机预言机模型下 DSHVE 是自适应安全的.

证明. 通过一系列模拟实验，运用构造法证明 DSHVE 的安全性.

1）实验 0. 该实验与真实实验完全相同，即挑战者分别执行 $DSHVE.UpdateEnc$，$DSHVE.KeyGen$，$DSHVE.Query$ 算法实现更新、解密密钥生成和查询操作.

由于执行过程与真实方案完全相同，所以对 PPT 的分辨器 $\mathcal{D}$ 有

$$|Pr[\mathcal{D}\left(View_{\mathcal{A},E_0}\right)=1]|=|Pr[\mathcal{D}\left(View_{\mathcal{A},\mathrm{Real}}\right)=1]|.$$

2）实验 1. 在实验 0 的基础上，每次执行更新操作时，挑战者将 $DSHVE.UpdateEnc$ 中的 F_0 用均匀采样的随机数代替. 设攻击者 $\mathcal{A}$ 对位置为 p 的属性 x_p（$p\in[m]$）执行操作 op（$op\in\{\mathrm{add},\mathrm{del}\}$），记为 (op,p).

若 $op=\mathrm{add}$，挑战者直接通过均匀采样生成随机数 $s_{\mathrm{add}}\xleftarrow{\$}\{0,1\}^{\lambda}$，并置 $c_{\mathrm{add},p}=s_{\mathrm{add}}$；

若 $op=\mathrm{del}$，挑战者首先查找 c_{add}，得到 $s_{\mathrm{add}}=c_{\mathrm{add},p}$，由于属性向量删除更新操作总是在添加操作之后，因此 $s_{\mathrm{add}}\neq\perp$，同时，挑战者均匀采样生成随机数 $s_{\mathrm{del}}\xleftarrow{\$}\{0,1\}^{\lambda}$，计算 $c_{\mathrm{del},p}=s_{\mathrm{add}}\oplus s_{\mathrm{del}}$. 即

$$c_{op,p}=\begin{cases}s_{\mathrm{add}},\text{若}op=\mathrm{add},\\ s_{\mathrm{add}}\oplus s_{\mathrm{del}},\text{若}op=\mathrm{del},\end{cases}$$

其中 s_{add}，s_{del} 挑战者通过均匀采样生成 2 个随机数 $s_{\mathrm{add}}\xleftarrow{\$}\{0,1\}^{\lambda}$，$s_{\mathrm{del}}\xleftarrow{\$}\{0,1\}^{\lambda}$. 最后挑战者计算密文集合 $c'=\{\{c_l\}_{op,l\in[m]\setminus p}\}\cup\{c_{op,p}\}$，并将 c' 发送至攻击者 $\mathcal{A}$.

设实验 1 中将 PRF F_0 输出用均匀采样的随机数代替，攻击者 $\mathcal{A}$ 通过模拟器 $\mathcal{B}1$ 所取得的优势记为 $Adv^{\mathrm{PRF}}_{\mathcal{B}1,\mathcal{A}}(\lambda)$，即在实验 1 中，攻击者所取得的攻击优势可表示为

$$|Pr[\mathcal{D}\left(View_{\mathcal{A},E_1}\right)=1]-|Pr[\mathcal{D}\left(View_{\mathcal{A},E_0}\right)=1]|\leqslant Adv^{\mathrm{PRF}}_{\mathcal{B}1,\mathcal{A}}(\lambda).$$

3）实验 2. 在实验 1 的基础上，解密密钥生成算法 $SHVE.KeyGen$ 和 $DSHVE.Query$ 算法中的挑战者将对称加密方案 $(Sym.Enc,Sym.Dec)$ 用一个快速查询列表代替. 设攻击者 $\mathcal{A}$ 自适应地选择谓词 $P^{\mathrm{SHVE}}_{v_j}$，$j\in\left[q_2\right]$，生成解密密钥时，挑战者随机采样 $K\xleftarrow{\$}\{0,1\}^{\lambda+\mathrm{lb}\lambda}$. 然后，对 $i\in[|S_j|]$，计算

$$d_{j,0}=\left(\oplus_{i\in[|S_j|]}c_{l_i}\right)\oplus K,$$

$$d_{j,1} \xleftarrow{\$} \{0,1\}^{\lambda+\mathrm{lb}\lambda},$$

最后挑战者向攻击者$\mathcal{A}$返回解密密钥

$$\boldsymbol{s}_j = (d_{j,0}, d_{j,1}, S_j).$$

由于对称加密方案$(Sym.Enc, Sym.Dec)$为一个理想的加解密算法，算法的加解密输出与均匀采样随机数在计算上是不可区分的. 设对称加密方案输入用均匀采样随机数替换，攻击者$\mathcal{A}$通过模拟器$\mathcal{B}2$所取得的优势为$Adv^{Sym}_{\mathcal{B}2,\mathcal{A}}(\lambda)$，则在实验 2 中

$$|Pr[\mathcal{D}(View_{\mathcal{A},E_2}) = 1]| - |Pr[\mathcal{D}(View_{\mathcal{A},E_1}) = 1]| \leqslant Adv^{Sym}_{\mathcal{B}2,\mathcal{A}}(\lambda).$$

4)实验 3. 按实验 2 方式构造模拟器$\mathcal{S}$并执行理想实验，可以看出攻击者在实验 3 中所取得的攻击优势与理想实验中相同，即

$$|Pr[\mathcal{D}(View_{\mathcal{A},\mathcal{S}}) = 1]| = |Pr[\mathcal{D}(View_{\mathcal{A},E_3}) = 1]| = |Pr[\mathcal{D}(View_{\mathcal{A},E_2}) = 1]|.$$

综合模拟器$\mathcal{B}1$，$\mathcal{B}2$可以构造出高效的模拟器$\mathcal{S}$，使得对于 PPT 的分辨器$\mathcal{D}$下，满足

$$Adv^{\mathrm{DSHVE}}_{\mathcal{D},\mathcal{A}}(\lambda) = |Pr[\mathcal{D}(View_{\mathcal{A},\mathrm{Real}}) = 1]| - |Pr[\mathcal{D}(View_{\mathcal{A},\mathcal{S}}) = 1]| \leqslant Adv^{\mathrm{PRF}}_{\mathcal{B}_1,\mathcal{A}}(\lambda) + Adv^{Sym}_{\mathcal{B}_2,\mathcal{A}}(\lambda),$$

即高效模拟器在安全参数λ下使攻击者$\mathcal{A}$在真实实验和理想实验中所取得的优势是可忽略的. 根据定义 8 可知，在随机预言机模型下 DSHVE 是自适应安全的. 证毕.

4.2.2 DSHVE-ODXT 方案安全性

方案初始化时，服务器存储信息为$\varnothing$，不会泄露任何信息，即$\mathcal{L}^{Setup} = \perp$，泄露主要发生在数据动态更新和连接查询阶段，即方案的泄露函数$\mathcal{L}$可表示为

$$\mathcal{L}_{\text{DSHVE-ODXT}} = (\mathcal{L}^{Update}, \mathcal{L}^{Search}).$$

下面分别对 2 个阶段的信息泄露给出定性分析.

1)更新泄露$\mathcal{L}^{Update}$

执行更新算法 *DSHVE-ODXT.Update*实现对服务器上密文索引数据库 *EDB* 的*xset*和*tset*更新. 方案采用标签数据对的方式存储*xset*与*tset*中的数据，数据存储标签均采用 PRF 生成，既能够节省存储开销，也可以实现更新操作，与*xset*和*tset*的历史状态无关. 因此方案是前向隐私安全的.

设相关文档索引不会出现相同的更新操作，则利用 PRF 生成的数据标签也不会重复. 同时对于不同的更新操作，服务器均采用相同的处理过程，服务器无法通过操作过程和存储的数据差异分辨更新操作的类型.

2)搜索泄露$\mathcal{L}^{Search}$

从方案的搜索算法执行过程可以看出，通过设计 DSHVE 方案，使得连接查询结果中仅包含与连接查询相匹配的所有文档集合，有效避免了更多的 KPRP 信息泄露，有效提高了连接查询时方案的安全性. 同时，相比单关键词搜索方案，连接查询泄露$\mathcal{L}^{Search}$还包括：

①TSet 访问泄露. 泄露更新次数最少关键词w_1的更新操作次数及发生时间，且服务器通过跟踪 TSet 查询时访问的数据地址标签，可以发现 2 次查询中是否包含相同的关键词w_1.

②XSet 访问泄露. 泄露与当前查询相关的所有更新操作的次数及发生时间，以及通过监控 XSet 访问标签，发现 2 次不同查询结果是否包含相同的文档信息.

③输出泄露. 泄露与当前查询匹配的文档索引集合.

其中①③对单关键词查询方案也同样存在.

引理 1. ODXT 的安全性[19]. 假设F和F_p为安全的伪随机函数，且群$\mathcal{G}$上满足 DDH 假设，则 ODXT 是对于泄露函数$\mathcal{L}_{\text{ODXT}}$自适应安全的. 其中泄露函数$\mathcal{L}_{\text{ODXT}}$定义为

$$\mathcal{L}_{\text{ODXT}} = (\mathcal{L}^{Setup}_{\text{ODXT}}, \mathcal{L}^{Search}_{\text{ODXT}}, \mathcal{L}^{Update}_{\text{ODXT}}),$$

其中：

$$\mathcal{L}^{Setup}_{\text{ODXT}} = \perp;$$

$$\mathcal{L}^{Update}_{\text{ODXT}}(op, (id, w)) = \perp;$$

$$\mathcal{L}^{Search}_{\text{ODXT}}(q) = (TimeDB(q), Update(q)).$$

推论 1. DSHVE-ODXT 安全性. 假设F和F_p为安全PRFs且群$\mathcal{G}$上满足 DDH 假设，F_0是安全的伪随机函数，且在随机预言机模型下 DSHVE 和 ODXT 分别是自适应安全的，则 DSHVE-ODXT 对泄露函数$\mathcal{L}_{\text{DSHVE-ODXT}}$是自适应安全的.

从 DSHVE-ODXT 中连接查询的工作过程可以看出，方案可以分别构造 2 个模拟器$\mathcal{B}_{\text{DSHVE}}$和$\mathcal{B}_{\text{ODXT}}$，并将算法中其他 PRF 用均匀采样的随机数代替，进而构造出方案的高效模拟器$\mathcal{S}$，使推论得以证明，详细证明过程在此不再赘述.

若用q表示包含$|q|$个关键词的连接关键查询语句，$|q|$表示q中包含关键词的个数，不妨设w_1表示q中更新次数最少的关键词.N_w表示关键词的个数，N^w_{Update}表示关键词w的更新次数，N_{Update}表示执行更新操作的总次数，n_{h}为 Bloom 过滤器中 Hash 函数的个数，DSHVE-ODXT 与同类方案在性能和安全性方面的对比结果如表 2 所示.

Table 2 Comparison of Schemes

表 2 方案对比

方案	动态更新	存储		通信轮数	计算	KPRP 隐藏	查询结果
		客户端	服务器				
OXT[9]	√		$O\left(N_{Update}\right)$	2	$O\left(\lvert q\rvert\times N_{Update}^{w_1}\right)$	×	准确
BDXT[19]	√		$O\left(N_{Update}\right)$	2	$O\left(\lvert q\rvert\times 2N_{Update}^{w_1}\right)$	×	准确
ODXT[19]	√		$O\left(N_{Update}\right)$	1	$O\left(\lvert q\rvert\times 2N_{Update}^{w_1}\right)$	×	不准确[15]
SHVE[14]	×		$\leqslant O\left(n_{\text{hash}}N_{Update}\right)$	2	$O\left(\lvert q\rvert\times N_{Update}^{w_1}\right)$	√	准确
DSHVE-ODXT（本文）	√	$O(N_w)$	$O\left(N_{Update}\right)$	1	$O\left(\lvert q\rvert\times 2N_{Update}^{w_1}\right)$	√	准确

5 实验及结果分析

DSHVE-ODXT 测试通过在物理机上安装的虚拟机实现，其中物理机处理器为 Intel® Core™ i7-8550U，内存为 8 GB，虚拟机硬件配置 4 核处理器、4 GB 内存，操作系统为 Ubuntu 18.08、编程语言为 Python3.9. 基于以上环境，从存储空间开销和搜索性能 2 个方面对方案进行测试. 其中涉及密码学算法均采用 pycryptodome 3.14 工具包实现，服务器密文索引数据库 TSet 和 XSet 采用支持标签访问的数据字典.

5.1 存储空间开销

Setup 算法执行完毕后，反复执行 *Update* 算法，向服务器中添加密文索引，以字节形式将客户端数据和服务器密文索引数据库以字节形式保存，记录保存的文件大小，得到方案存储空间开销随更新操作次数的变化关系如图 4 所示.

从图 4 可以看出，服务器存储空间开销与更新次数呈线性关系，相比于 ODXT 方案，DSHVE-ODXT 方案存储空间开销略大，但仍在可承受范围内. DSHVE-ODXT 客户端所需的存储空间开销较低，且与相同关键词的更新次数无关，而与索引中维护的关键词个数呈线性关系.

5.2 性能测试

DSHVE-ODXT 方案单关键词、多关键词连接查询以及删除时的搜索性能测试结果如图 5~7 所示.

1）单关键词查询

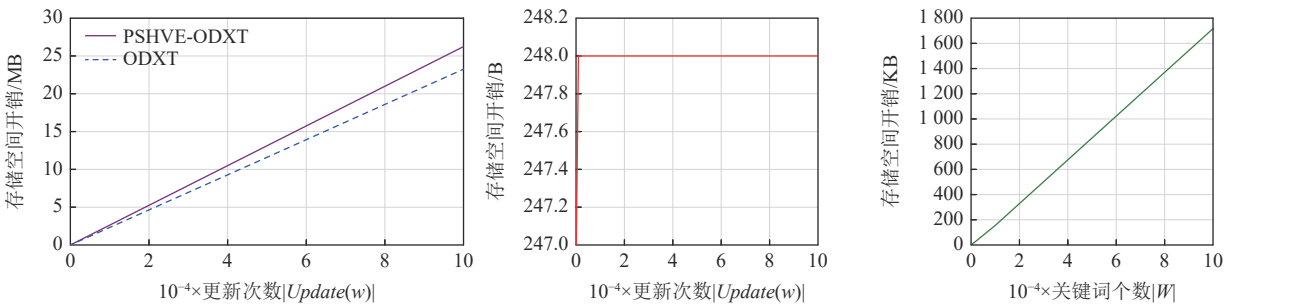
方案初始化完成后，按照先后顺序，通过更新算法向服务器分别添加关键词w_1，w_2，w_3，使得$|Update(w_i)|_{i\in\{1,2,3\}}=10^5$，$|Update(w_i)|$表示$w_i$的更新次数. 在更新过程中，对添加的关键词反复执行 10 次查询操作，记录结果返回时间，并计算查询平均时间$\bar{t}=\dfrac{\overline{ts}}{N}=\dfrac{ts/\left|r_{w_i}\right|}{N}$，其中$N=10$，$\left|r_{w_i}\right|$表示匹配文档索引集合中的元素个数，$ts$表示每次查询结果的总时间.

从图 5 可以看出，单关键词的查询响应总时间与该关键词的更新次数呈线性关系，与服务器中密文索引规模大小$|EDB|$无关.

2）多个关键词连接查询

方案初始化完成后，按照先后顺序，通过更新算法向服务器分别添加关键词w_1，w_2，w_3，w_4，使得$|Update(w_i)|_{i\in\{1,2,3,4\}}=10^5$，$|Update(w_i)|$表示$w_i$的更新次数. 在更新过程中，分别执行连接查询操作，反复执行 10 次记录查询结果返回时间，并按 1）的方式计算平均查询时间，得到测试结果如图 6 所示.

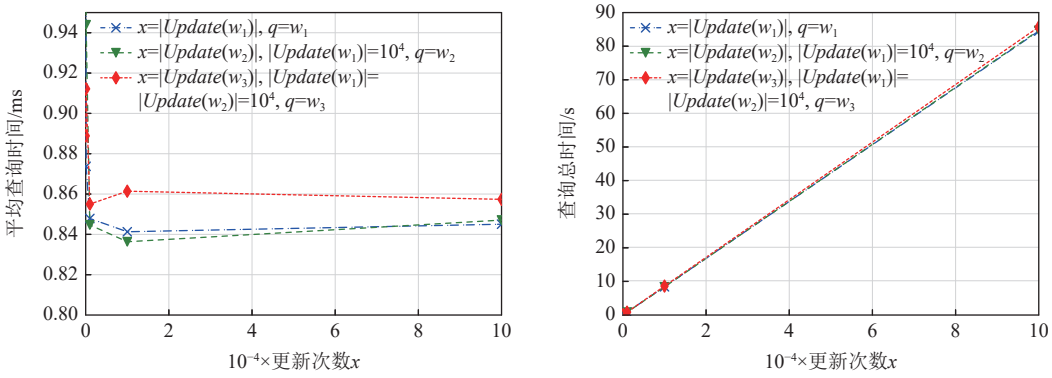
可以看出，多关键词连接查询的平均查询时间



(a) 更新次数对服务器存储空间开销的影响 (b) 更新次数对客户端存储空间开销的影响 (c) 关键词个数对客户端存储空间开销的影响

Fig. 4 Influencing factors of storage space overhead

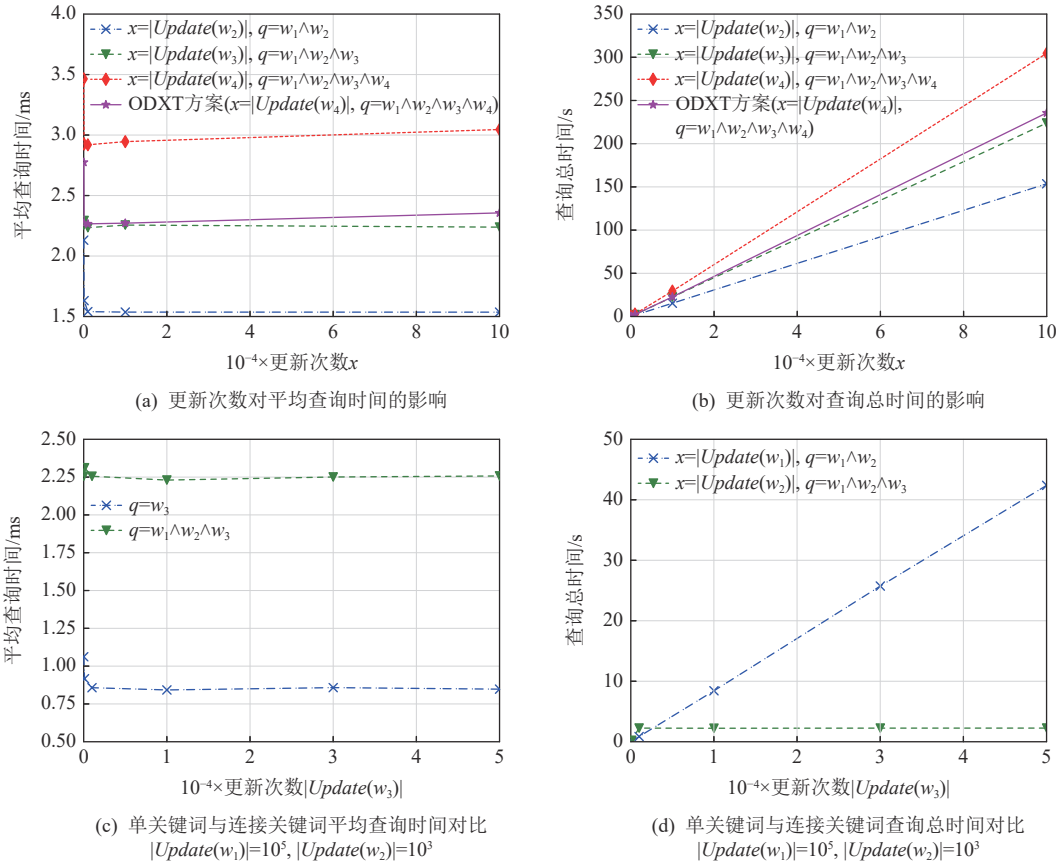
图 4 存储空间开销的影响因素



(a) 单关键词平均查询时间 (b) 单关键词查询总时间

Fig. 5 Performance of single keyword queries

图 5 单关键词查询性能



(a) 更新次数对平均查询时间的影响 (b) 更新次数对查询总时间的影响

(c) 单关键词与连接关键词平均查询时间对比 |$Update(w_1)$|=10^5, |$Update(w_2)$|=10^3

(d) 单关键词与连接关键词查询总时间对比 |$Update(w_1)$|=10^5, |$Update(w_2)$|=10^3

Fig. 6 Performance of conjunctive queries

图 6 连接查询性能

随连接关键词个数的增加而增大, 但连接查询结果文档包含10^5个文档索引时的 4 个关键词的连接查询, 其平均时间依然在 3 ms 左右, 略大于 ODXT 方案, 能够满足大部分应用场景对搜索性能的要求. 另外, 由于方案的查询时间与服务器中密文索引规模大小|*EDB*|无关, 从图 6(c)(d)可以看出, 连接查询时, 最终方案的总查询时间与文档更新操作最少的关键词的更新次数有关. 在真实应用场景下, 若能在连接查询中给出更精确的关键词, 可以进一步提高最终查询结果的响应时间, 提升方案的应用效果.

3)删除操作对操作性能影响

向密文数据库中加入 3 个密文索引w_1, w_2, w_3, 其中|$Update(w_1)$|=10^5, |$Update(w_2)$|=7 000, |$Update(w_3)$|=5 000, 然后执行方案的删除操作, 并在删除后反复执行 10 次连接查询$q=(w_1 \wedge w_2 \wedge w_3)$, 记录查询时间并取其平均值, 得到测试结果如图 7 所示, 其中关键词

w_3删除 40% 的文档索引后总查询时间基本保持不变，因为继续删除 TSet 中 w_3对应列表文档索引后，$|Update(w_3)| > |Update(w_2)|$，查询时间不再受 w_3更新次数影响.

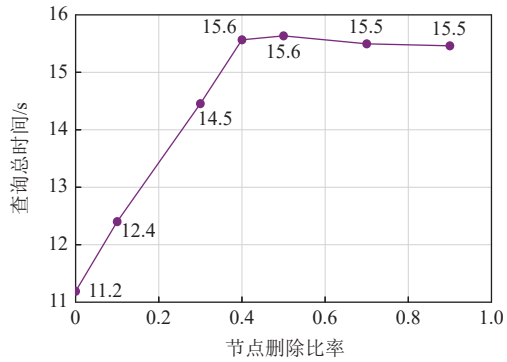


Fig. 7 Effect of deletion operation on search time

图 7 删除操作对搜索时间的影响

6 结 语

针对当前大部分连接查询方案仅支持静态密文数据库的问题，结合对称隐藏向量加密思想，设计了支持动态数据更新的动态对称隐藏向量加密（SHVE）方案，并在此基础上，通过设计动态连接查询的 TSet 和 XSet 结构，构造了支持连接查询的动态对称可搜索加密方案. 方案具有 3 个特点：

1）支持动态更新. 相对当前大多数静态连接查询方案，该方案支持密文数据库添加和删除操作，进一步提高了方案在真实应用场景下的可用性.

2）查询时仅需单轮交互. 方案引入基于盲指数计算的 ODXT，实现了单轮交互的连接查询. 相比其他 2 轮以上的交互方案，减少通信轮数可以有效降低真实网络环境下网络延时对搜索性能的影响，且更易于多客户端场景下的扩展.

3）安全性高. 通过构造 SHVE，结合 ODXT，使方案在单次查询中，仅泄露文档索引与查询条件的完全匹配结果，减少交叉查询信息泄露，进一步提高了安全性.

方案中的连接查询语句灵活性以及多客户端场景下方案适用性问题仍值得进一步深入研究.

作者贡献声明：黄一才提出了算法思路和实验方案，并撰写了论文；郁滨参与了实验方案制定并提出论文修改意见.

参 考 文 献

[1] Huang Yicai, Li Sensen, Yu Bin. A survey of symmetric searchable encryption in cloud environment[J]. Journal of Electronics & Information Technology, 2023, 45(3): 1134−1146(in Chinese)
（黄一才, 李森森, 郁滨. 云环境下对称可搜索加密研究综述[J]. 电子与信息学报, 2023, 45(3): 1134−1146）

[2] Sharma D. Searchable encryption : A survey[J]. Information Security Journal: A Global Perspective, 2023, 32(2): 76−119

[3] Song Xiaodong, Wagner D, Perrig A. Practical techniques for searches on encrypted data[C]//Proc of the 21st IEEE Symp on Security and Privacy 2000. Piscataway, NJ: IEEE, 2000: 44−55

[4] Goh E J. Secure indexes, 2003/216[EB/OL]. 2003[2021-03-01]. http://eprint.iacr.org/2003/216

[5] Naveed M, Prabhakaran M, Gunter C A. Dynamic searchable encryption via blind storage[C]//Proc of the 35th IEEE Symp on Security and Privacy 2014. Piscataway, NJ: IEEE, 2014: 639−654

[6] Curtmola R, Garay J, Kamara S, et al. Searchable symmetric encryption: Improved definitions and efficient constructions[J]. Journal of Computer Security, 2011, 19(5): 895−934

[7] Hahn F, Kerschbaum F. Searchable encryption with secure and efficient updates[C]//Proc of the ACM Conf on Computer and Communications Security. New York: ACM, 2014: 310−320

[8] Ghareh C J, Papadopoulos D, Papamanthou C, et al. New constructions for forward and backward private symmetric searchable encryption[C]//Proc of the 2018 ACM SIGSAC Conf on Computer and Communications Security(CCS'18). New York: ACM, 2018: 1038−1055

[9] Cash D, Jarecki S, Jutla C, et al. Highly-scalable searchable symmetric encryption with support for boolean queries[G]//LNCS 8042: Proc of the 33rd Symp on CRYPTO 2013. Berlin: Springer, 2013: 353−373

[10] Sun S F, Liu J K, Sakzad A, et al. An efficient non-interactive multi-client searchable encryption with support for Boolean queries[C]//LNCS 9878: Proc of the 21st Symp on ESORICS 2016. Berlin: Springer, 2016: 154–172

[11] Wang Yunling, Wang Jianfeng, Sun S F, et al. Towards multi-user searchable encryption supporting Boolean query and fast decryption[C]//LNCS 10592: Proc of Symp on ProvSec 2017. Berlin: Springer, 2017: 24–38

[12] Li Mingyue, Jia Chunfu, Du Ruizhong, et al. Forward and backward secure searchable encryption scheme supporting conjunctive queries over bipartite graphs[J]. IEEE Transactions on Cloud Computing, 2023, 11(1): 1091−1102

[13] Bag A, Talapatra D, Rastogi A, et al. TWo-IN-one-SSE: Fast, scalable and storage-efficient searchable symmetric encryption for conjunctive and disjunctive Boolean queries[J]. Proceedings on Privacy Enhancing Technologies, 2023, 2023(1): 115−139

[14] Lai S, Patranabis S, Sakzad A, et al. Result pattern hiding searchable encryption for conjunctive queries[C]//Proc of the 2018 ACM SIGSAC Conf on Computer and Communications Security(CCS'18). New York: ACM, 2018: 745−762

[15] Zhang Y, Katz J, Papamanthou C. All your queries are belong to us: The power of file-injection attacks on searchable encryption[C]//Proc of the 25th Symp on USENIX Conf on Security (SEC'16). Berkeley, CA: USENIX Association, 2016: 707−720

[16] Du Ruizhong, Zhang Yuqing, Li Mingyue. Efficient dynamic searchable encryption scheme for conjunctive queries based on bidirectional index[J]. Journal on Communications, 2022, 43(5): 123−132 (in Chinese)
(杜瑞忠, 张玉晴, 李明月. 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[J]. 通信学报, 2022, 43(5): 123−132)

[17] Kamara S, Moataz T. Boolean searchable symmetric encryption with worst-case sub-linear complexity[EB/OL]. 2017[2023-07-20]. https://eprint.iacr.org/2017/126.pdf

[18] Xu Wanshan, Zhang Jianbiao, Yuan Yilin, et al., Symmetric searchable encryption with supporting search pattern and access pattern protection in multi-cloud[J/OL]. Concurrency and Computation Practice and Experience, 2023[2023-07-20]. https://onlinelibrary.wiley.com/doi/10.1002/cpe.7651

[19] Patranabis S, Mukhopadhyay D. Forward and backward private conjunctive searchable symmetric encryption[C/OL]//Proc of the Network and Distributed System Security (NDSS) Symp. 2021[2023-07-20].https://www.ndss-symposium.org/wp-content/uploads/ndss2021_2C-3_23116_paper.pdf

[20] Zhang Lanlan, Cao Weidong, Wang Huaichao. A multi-user dynamic symmetric searchable encryption scheme supporting conjunctive search[J]. Journal of Computer Research and Development, 2022, 59(10): 2309−2322 (in Chinese)
(张蓝蓝, 曹卫东, 王怀超. 一种支持联合搜索的多用户动态对称可搜索加密方案[J]. 计算机研究与发展, 2022, 59(10): 2309−2322)

[21] Sun S F, Steinfeld R, Lai S, et al. Practical non-interactive searchable encryption with forward and backward privacy[C/OL]//Proc of the Network and Distributed System Security (NDSS) Symp. 2021[2023-07-20].https://www.ndss-symposium.org/wp-content/uploads/ndss2021_2C-4_24162_paper.pdf



Huang Yicai, born in 1985. PhD. His main research interests include IoT, secure cloud storage system, and searchable encryption.
黄一才, 1985 年生. 博士. 主要研究方向为物联网、安全云存储系统、可搜索加密.



Yu Bin, born in 1964. PhD, professor, PhD supervisor. His main research interests include the design and analysis of algorithms, visual cryptography, and network security.
郁　滨, 1964 年生. 博士, 教授, 博士生导师. 主要研究方向为算法设计与分析、视觉密码、网络安全.