基于可更新加密的保护搜索模式的动态可搜索加密方案

徐承志, 徐磊, 许春根

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230100016

基于可更新加密的保护搜索模式的动态可搜索加密方案

徐承志[1]　徐　磊[2]　许春根[2]
1 南京理工大学计算机科学与工程学院　南京 210094
2 南京理工大学数学与统计学院　南京 210094
(xcz@njust.edu.cn)

摘　要　动态可搜索对称加密(Dynamic Searchable Symmetric Encryption,DSSE)技术作为静态可搜索加密技术的拓展,因解决了数据密态场景下的安全检索问题并支持数据动态更新而备受关注。众所周知,目前大多数DSSE方案会泄露一些额外的信息以寻求更好的效率,如搜索模式与访问模式。最近的研究表明,这些泄露的信息面临着严重的安全问题,拥有数据库背景知识的敌手可能利用这些泄露信息恢复查询或重构数据库。由于这些泄露是伴随着查询的过程泄露出来的,因此不少学者提出在搜索时更新加密数据库来降低上述潜在的风险,即用户下载搜索到的密文数据到本地,解密后重新加密再上传到云服务器端。但这种方法会导致巨大的客户端通信、存储和计算开销。针对这一问题,提出了一种基于可更新加密的保护搜索模式的DSSE方案,该方案可以在不泄露数据隐私的情况下直接在服务器端进行数据更新,从而降低传统更新方法的通信开销以及客户端的计算开销。安全性分析表明,所提方案能有效保护搜索模式泄露;性能分析表明,所提方案相比传统利用更新密文方法保护搜索模式的方案能有效降低通信开销。在关键词匹配100个文档的情况下,与下载到本地重加密重传方式相比,所提方案的通信开销降低了70.92%。

关键词:动态可搜索加密;可更新加密;前向安全;搜索模式
中图分类号　TP309

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

XU Chengzhi[1],XU Lei[2] and XU Chungen[2]
1 School of Computer Science and Technology,Nanjing University of Science and Technology,Nanjing 210094,China
2 School of Mathematics and Statistics,Nanjing University of Science and Technology,Nanjing 210094,China

Abstract　Dynamic searchable symmetric encryption(DSSE) technology,as an extension of static searchable encryption,has attracted much attention because it solves the problem of secure retrieval over encrypted data and supports data dynamicity. For practicality concerns,most current DSSE schemes leak extra information(e.g.,search patterns and access patterns) to fast search. Recent studies show that this leaked information poses serious security problems,the adversary with background knowledge of the database may exploit the leaked information to recover the query or reconstruct the database. Since this information reveals along with the query process,scholars propose to refresh the encrypted database after the query to reduce the above potential risks. However,this approach leads to huge client-side communication,storage,and computation overheads. Because the client needs to download the results locally,decrypt them,re-encrypt them and finally upload them to the cloud. To address this problem,this paper proposes a new updatable DSSE scheme that hides all the above information including access pattern,search pattern. The scheme can update data directly at the server side without disclosing data privacy,thus reducing the communication overhead of traditional update methods of the client side. The security analysis shows that this scheme can hide the search pattern effectively. In addition,the communication cost of the proposed scheme is also significantly degraded when compared with the traditional scheme that executes ciphertext refreshing by the client. For example,in the case of keywords matching 100 documents,compared with downloading to local re-encryption and retransmission,the communication overhead of this scheme is reduced by 70.92%.

Keywords　Dynamic searchable encryption,Updatable encryption,Forward secure,Search pattern

到稿日期:2023-01-03　返修日期:2023-05-23
基金项目:国家自然科学基金(62202228,62072240);江苏省自然科学基金(BK20210330)
This work was supported by the National Natural Science Foundation of China(62202228,62072240) and Natural Science Foundation of Jiangsu Province(BK20210330).
通信作者:徐磊(leixu@njust.edu.cn)

1 引言

随着互联网技术的飞速发展，海量数据的存储和管理需求日益增加，越来越多的用户选择将自己的数据存储到云服务器中。但是明文数据直接存放到云服务器上会面临敏感数据泄露、数据被篡改等安全隐患。为了保护数据的安全性，数据加密存储成为了共识。但是密文数据会影响用户的搜索和应用能力，如果想查询指定数据，用户需要下载解密后再进行查询，这会导致巨大的通信与计算开销。为了解决这个问题，可搜索加密(Searchable Encryption，SE)[1]的概念被提出。

可搜索机密是一种能够让用户在密文上进行检索的技术。2000 年，Song 等[1]提出了第一个 SE 方案，该方案中的数据拥有者对文档中的每个关键词进行加密，然后将加密的文档发送给云服务器。检索时数据拥有者将关键词的陷门发送给云服务器，云服务器将陷门与文档中的每个密文进行匹配，如果匹配成功，则返回该密文文档。Boneh 等[2]于 2004 年提出了第一个公钥可搜索加密(Public Key Encryption with Keyword Search，PEKS)方案，可以实现密钥管理等更多的功能[3]。但 PEKS 搜索的复杂度随密文数量线性增长，计算开销较大。早期的 SSE 方案都是静态的，用户只能对关键词进行检索，无法动态更新数据。为了更好地支持更新，研究者进而又提出了动态可搜索对称加密(Dynamic Searchable Symmetric Encryption，DSSE)[4]。DSSE 扩展了 SSE 的应用，用户不仅能在密文数据上进行检索，还可以执行更新操作，如添加新的数据或删除已有的数据。

值得注意的是，为了达到亚线性的搜索效率，Curtmola 等[5]提出的 DSSE 构造允许在搜索过程中泄露一些额外信息，如访问模式与搜索模式。访问模式揭露了查询访问的具体文档，而搜索模式指两个查询是否对应同一个关键词。搜索模式包括显式和隐式两种，显式搜索模式指关键词 w 每次查询时生成的搜索令牌相同，隐式搜索模式指敌手根据返回的密文标识符信息是否相同来推测对应的关键词。早期，人们并没有意识到这些泄露函数对密文数据安全性的影响，直到 Islam 等[6]提出了第一个利用访问模式的攻击方案 IKK。随后 Liu 等[7]和 Oya 等[8]利用搜索模式泄漏的信息对 DSSE 方案进行攻击，成功恢复了部分查询关键词信息。这种攻击的主要思想是根据搜索模式信息生成关键词的查询频率，并与现实世界中关键词的查询信息进行对比，找到对应关系。除了访问模式和泄露方式，动态可搜索加密方案还额外泄露更新模式，Zhang 等[9]针对更新泄露模式提出了文件注入攻击。敌手向用户数据库中注入特殊的文件，然后观察用户查询返回的文件信息，从而推测用户查询的关键词。

为了抵御基于上述泄露的攻击，不少防御手段被提出来。例如，针对访问模式中结果长度和共现关系的计数攻击，Islam 等[6]提出了填充的方法。他们将关键词按照频率的高低划分为不同的簇，然后填充文件使得每个簇中的所有关键词匹配相同数量的文档标识符。但是这种防御手段仍有待改进，因为敌手可以从搜索模式进行突破，通过搜索模式的泄露统计关键词的查询频率并发起频率分析攻击[7]。即使是前向安全算法有伴随更新刷新搜索令牌的功能也无法完全隐藏这种搜索模式的泄露，因为敌手可以通过返回结果的相似度去判断两次查询是否对应相同的关键词。为了强化这些弱防御方法，学者们提出了更新密文数据的方法。具体来说，在搜索时将密文数据下载到本地并删除原有的密文，数据持有者在本地解密后重新加密再上传到云服务器。通过这种方式，搜索模式和查询之间的共现模式就可以被隐藏，只需要隐藏结果集的大小就可以有效地隐藏那些致命的安全泄露。但这种方法会带来较高的客户端计算和通信开销，对于网络和计算资源受限的用户来说是难以接受的。为解决上述问题，本文重点关注并设计了一种更高效的支持密文更新的可搜索加密方案并同时支持前向安全，以起到降低密文更新时的计算和通信开销作用。

受 Boneh 等[10]提出的可更新加密(Updatable Encryption，UE)的启发，我们提出利用可更新加密在云服务器端更新密文数据。与上述传统的下载到本地重加密重传的方式不同，在可更新加密方案中，用户只需根据旧密钥和新密钥生成重加密令牌，并发送给云服务器，云服务器使用重加密令牌更新所有为新密钥加密的密文数据，这避免了密文数据的传输，并且主要的密文更新计算开销放在了云服务器端。基于此思想，本文构建了隐藏搜索模式的前向安全的 DSSE 方案 UEDSSE(Updatable Encryption DSSE)。在 UEDSSE 方案中，使用 UE 加密文档标识符，每次在搜索时发送加密的搜索令牌，服务器端更新搜索到的密文数据并返回，使得即使用相同关键词进行查询，返回数据的密文也不同，有效保护了搜索模式。与传统保护搜索模式的方案相比，UEDSSE 能有效减小通信开销。最后，安全性分析表明，本文方案能有效保护搜索模式的泄露；性能分析表明，本文方案相比传统利用更新密文方法保护搜索模式的方案能有效降低通信开销。例如，在关键词匹配 100 个文档的情况下，与下载到本地重加密重传方式相比，本文方案的通信开销降低了 70.92%。

2 相关工作

2.1 可搜索加密

近年来，可搜索加密得到了广泛的关注，其发展也非常迅速。Goh 等[11]为了提高检索效率，构造了基于正排索引的 SSE 方案，并提出了一种安全模型，即针对自适应性选择关键词攻击的语义安全。为了进一步提高效率，Curtmola 等[5]提出了一种基于倒排索引的方案，执行检索操作时，云服务器只需搜索相应的陷门去查找文档。随后，更多的 SSE 方案被提出，提高了 SSE 方案的性能和安全性，并具有更多的查询功能。

为了满足数据动态更新的需求，Kamara 等[4]提出了第一个 DSSE 方案。但是，数据的更新中往往会泄露重要信息。Zhang 等[9]针对添加操作中泄露的信息，提出了一种文件注入攻击。该攻击通过向云服务器注入特定的文件来获得搜索令牌与关键词之间的关系。为了抵御文件注入攻击，DSSE 方案需满足前向安全性。前向安全是指旧的搜索令牌无法搜到新添加的文件。Stefanov 等[12]首次给出了前向安全的定义，并基于不经意随机存储(Oblivious Random Access Memory，ORAM)[13]构造了一种前向安全的 DSSE 方案。然而，

ORAM需要较大的计算与存储开销，难以在实际中应用。2020年，He等[14]基于鱼骨链构造了两个方案，减少了客户端的存储开销，但更新次数受到限制。2021，Chen等[15]基于缓存技术构建了更高效的前向安全的DSSE方案。

针对文件删除中的泄漏信息，Stefanov等[12]提出了后向安全的概念，它指DSSE无法搜索到已删除文件的信息。Bost等[16]构造了第一个满足后向安全的DSSE方案，并根据删除过程中泄露信息的不同，给出了3种不同强度的后向安全定义。但是，文献[16]中的可穿刺加密是基于公钥构造的，执行删除操作时效率较低。Sun等[17]利用伪随机函数等构造了对称可穿刺加密，提高了检索效率。2021年，Sun等[18]在文献[17]的基础上，提出了新的DSSE方案Auro，减小了计算开销。此外，各种新的具有更多功能的DSSE方案也被提出[19]，如支持多用户搜索[20]、连接关键词查询[21]、结果可验证[22]等。

近年来，一种新的利用数据挖掘思想的攻击方法给可搜索加密的安全性带来了新的挑战[22]。虽然可搜索加密协议中数据库和检索请求都是加密的，但掌握背景知识的敌手仍然能通过观察用户发送的请求和接收的数据来提取关键信息并恢复用户的信息，这种攻击也叫作泄露滥用攻击(Leakage Abuse Attack，LAA)。泄露滥用攻击主要利用了搜索模式、访问模式以及共现模式等泄露信息[23]。搜索模式揭露了两个搜索查询的关键词是否是同一个，访问模式揭露了查询关键词匹配的文档信息，而共现模式则揭露了不同关键词出现在相同文档的频率信息。2012年，Islam等[6]提出了第一个利用这种泄露信息的攻击方案IKK，IKK利用共现模式进行攻击，当敌手拥有较多的辅助数据信息时，IKK攻击具有相当高的成功率。此后，陆续有新的利用各种泄露模式的攻击方案[24-26]被提出。

为了抵御泄漏滥用攻击，学者们也提出了一些安全防护方案。2018年，Chen等[27]提出了一种利用差分隐私隐藏搜索模式的方法。Kamara等[28]将分区和填充两种方法结合，构造了一种通用转换器，能够将一般的SSE方案进行转换，使其能隐藏结果大小。Grubbs等[29]提出了平滑频率的思想来隐藏泄露信息，并结合选择性复制、添加假查询以及批处理3种方法构造了Pancake方案。Zhao等[30]将差分隐私引入DSSE中，构造了隐藏多种模式的前后向安全的DSSE方案。此外，学者们也利用可信硬件来构造安全的SSE方案[31]。现有方案在降低泄露信息的同时都会增加通信开销，这在客户端网络资源受限的情况下是难以接受的。本文将可更新加密与DSSE相结合，虽然增加了加解密的计算开销，但能有效保护搜索模式，同时保持原有的通信开销。

2.2 可更新加密

当数据以加密形式存储在云服务器中时，为了防止侧信道攻击、软件泄露以及内部人员破坏导致的密钥泄露问题，数据加密密钥需要定期更新。传统的更新方法将所有数据下载到本地，解密后使用新密钥加密并上传到云服务器，通信和计算开销较大。为了解决这个问题，Boneh等[2]提出了可更新加密的概念，它可以周期性地生成一个新的密钥和重加密令牌，重加密令牌可以将旧的密文转换为新密钥加密的密文。

可更新加密有两种类型，分别为密文相关[32]与密文无关。密文相关的方案需要下载所有的密文，为每个密文生成一个重加密令牌，然后将重加密令牌上传到云服务器来对所有密文执行重加密操作。密文无关的方案中，重加密令牌的生成只依赖旧的密钥与新的密钥，无需密文参与，更具有实用性。Lehmann等[33]在密文无关的场景下提出了两个安全定义：不可区分加密(Indistiguishability of Encryptions，IND-ENC)和不可区分更新(Indistiguishability of Updates，IND-UPD)，并基于ElGamal方案构建了一个满足上述安全定义的方案RISE。其中IND-ENC指敌手无法区分密文c是由明文m_0，m_1中的哪一个加密得到，IND-UPD指敌手无法区分c'是由密文c_0，c_1中的哪一个重加密得到。Klooß等[34]在文献[33]的基础上提出了更强的安全模型，即选择明文安全以及明文和密文的完整性保护，并利用加密-认证和Naor-Yung范式构建了满足该安全模型的可更新加密方案。Boyd等[35]构建了一个新的高效的可更新加密方案，可满足更强的安全性定义，即敌手无法区分加密算法生成的密文和重加密算法生成的密文。

3 预备知识

本章给出了必要的背景知识，包括符号定义、DSSE方案定义及其安全性模型、公钥加密算法和可更新加密方案等。

3.1 符号定义

本文使用的符号及其含义如表1所列。文中用$x \xleftarrow{\$} \boldsymbol{X}$表示从集合$\boldsymbol{X}$中均匀随机采样获取一个元素，$\{0,1\}^l$表示长度为$l$的字符串，$\{0,1\}^*$表示任意长度的字符串。对于集合$\boldsymbol{D}$，$|\boldsymbol{D}|$表示$\boldsymbol{D}$的基数。对于两个字符串$S_1$和$S_2$，$S_1 \| S_2$表示对两个字符串进行拼接。用$\emptyset$表示空集或空字符串，$\perp$表示空值。

数据集表示为$\boldsymbol{DB}=(id_i,\boldsymbol{W}_i)_{i=1}^{|\boldsymbol{DB}|}$，其中$id_i$表示第$i$个文档的文档标识符，$\boldsymbol{W}_i$表示该文档包含的关键词集合。用$\boldsymbol{W}=\bigcup_{i=1}^{|\boldsymbol{DB}|}\boldsymbol{W}_i$表示数据库$\boldsymbol{DB}$的关键词集合，$DB(w)=\{id_i \mid w\in\boldsymbol{W}_i\}$表示包含关键词$w$的文档标识符集合。$a_w$表示关键词$w$当前的更新次数，$n_w$表示关键词$w$的搜索结果中元素的数量，即去掉删除文档后匹配的文档标识符数。

表1 符号定义

Table 1 Symbol definition

符号	定义说明
λ	安全参数
id	文件标识符
w	关键词
a_w	w的更新次数
n_w	w的搜索结果个数
$\boldsymbol{DB}$	明文数据库
$\boldsymbol{EDB}$	加密数据库
F	安全的伪随机函数
PKE	公钥加密算法
h,H_1,H_2	安全的哈希函数

3.2 动态对称可搜索加密

动态对称可搜索加密方案主要由以下几个协议组成。

1)$(\sigma,\boldsymbol{EDB})\leftarrow$Setup$(\lambda)$：用于初始化系统，输入安全参数

λ，输出当前的状态 σ 并将其存储在客户端，云服务器初始化数据结构 $\boldsymbol{EDB}$ 并保存更新信息。

2)$(\sigma',\boldsymbol{EDB}')\leftarrow$Update$(\sigma,op,(w,id),\boldsymbol{EDB})$：是客户端与云服务器之间的协议。输入状态 σ，操作 $op\in\{add,del\}$ 和关键词-文档标识符对(w,ind)，输出更新令牌。云服务器接收到更新令牌后，更新密文数据库。协议执行结束后，客户端的状态变为 σ'，云服务器上的加密数据库更新为 $\boldsymbol{EDB}'$。

3)$(\sigma',\boldsymbol{DB}(w),\boldsymbol{EDB}')\leftarrow Search(\sigma,\mathrm{K},\mathrm{w},\boldsymbol{EDB})$：客户端和云服务器之间的协议。客户端使用密钥 K、状态 σ 和想要搜索的关键词 w 生成搜索令牌，然后将搜索令牌发送给云服务器。云服务器收到搜索令牌后执行匹配操作，将匹配的文件标识符集 $\boldsymbol{DB}(w)$ 返回给客户端，然后更新密文数据库 $\boldsymbol{EDB}$ 为 $\boldsymbol{EDB}'$，客户端更新状态 σ 为 σ'。

定义 DSSE 方案适应性安全的方法通常是定义 DSSE 在理想模型和现实模型中的不可区分性。在两个模型中，敌手都可以自适应地发起更新或搜索请求。现实模型中反映了真实 DSSE 方案的行为，而理想模型中，通过一个模拟器 S 根据泄露信息模拟 DSSE 方案的执行。给定一个 DSSE 方案 $\Pi=$(Setup，Search，Update)，用泄露函数$=(\mathcal{L}^{\mathrm{Setup}},\mathcal{L}^{\mathrm{Update}},\mathcal{L}^{\mathrm{Search}})$来表示敌手在系统初始化阶段、更新阶段以及查询阶段获取到的泄露信息。使用 $\mathcal{A}$ 和 $\mathcal{S}$ 来分别表示敌手和模拟器，定义两个概率性实验。

1)$\mathrm{Real}_{\mathcal{A}}^{\Pi}(\lambda)$：敌手 $\mathcal{A}$ 选择初始明文数据库 $\boldsymbol{DB}$，运行 Setup 算法得到加密数据库 $\boldsymbol{EDB}$。敌手 $\mathcal{A}$ 可以自适应地发出输入为$(op,(w,id))$的 Update 请求或输入为 w 的 Search 查询，并接收云服务器返回的结果。最后，敌手 $\mathcal{A}$ 输出一个比特 $b\in\{0,1\}$ 作为输出。

2)$\mathrm{Ideal}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\Pi}(\lambda)$：敌手 $\mathcal{A}$ 选择初始明文数据库 $\boldsymbol{DB}$，模拟器根据泄露信息 $\mathcal{L}^{\mathrm{Setup}}$ 运行模拟算法 $\mathcal{S}(\mathcal{L}^{\mathrm{Setup}})$ 并返回加密数据库 EDB 给敌手 $\mathcal{A}$。敌手 $\mathcal{A}$ 可以发出与现实模型中相同的请求，模拟器 $\mathcal{S}$ 根据泄漏函数 $\mathcal{L}^{\mathrm{Update}},\mathcal{L}^{\mathrm{Search}}$ 运行相应的模拟器 $\mathcal{S}(\mathcal{L}^{\mathrm{Update}}),\mathcal{S}(\mathcal{L}^{\mathrm{Search}})$，并返回结果给敌手 $\mathcal{A}$。最后，敌手 $\mathcal{A}$ 输出一个比特 $b\in\{0,1\}$ 作为输出。

定义 1($\mathcal{L}$-自适应安全) 给定泄露函数 $\mathcal{L}=(\mathcal{L}^{\mathrm{Setup}},\mathcal{L}^{\mathrm{Update}},\mathcal{L}^{\mathrm{Search}})$，如果对于任意足够大的安全参数 $\lambda\in\boldsymbol{N}$ 和敌手 $\mathcal{A}$，存在一个有效的模拟器 $\mathcal{S}$ 使得：

$$|Pr[\mathrm{Real}_{\mathcal{A}}^{\Pi}(\lambda)=1]-Pr[\mathrm{Real}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\Pi}(\lambda)=1]|\leqslant negl(\lambda)\quad(1)$$

则称一个 DSSE 方案 Π 是 $\mathcal{L}$-自适应安全的。

在 DSSE 中，前向安全指旧的搜索令牌无法搜索到新添加的文档，即更新操作不会泄露任何有关关键词的信息，只会泄露文档的更新次数。正式定义如下：

定义 2(前向安全) 对于一个 $\mathcal{L}$-自适应安全的动态对称可搜索加密方案 Π，如果其更新协议的泄露函数 $\mathcal{L}^{\mathrm{Update}}$ 可以写为：

$$\mathcal{L}^{\mathrm{Update}}(op,w,id)=(op,id)\quad(2)$$

则称方案 Π 是前向安全的。

3.3 公钥加密

与对称密码算法中数据发送方和接收方共享相同密钥不同，公钥加密算法中数据发送方和接收方具有不同的密钥。公钥加密正式的定义如下。

定义 3(公钥加密算法) 公钥加密算法 PKE 由一组多项式时间的算法(GenKey，Enc，Dec)组成：

1)$(pk,sk)\leftarrow$PKE.GenKey(λ)：密钥生成算法，输入安全参数 λ，输出公私钥对(pk,sk)，并将 pk 公布出去。

2)$c\leftarrow$PKE.Enc(pk,m)：加密算法，输入公钥 pk 和消息 m，算法输出密文 c。

3)$m\leftarrow$PKE.Dec(sk,c)：解密算法，输入私钥 sk 和密文 c，算法输出明文 m。

3.4 可更新加密

可更新加密是一种对称加密方案，它提供了一种重加密功能，能够将旧密钥加密的密文转换为新密钥加密的密文。加密密钥随着轮次 e 的推移而变化，当从轮次 e 变化到轮次 $e+1$ 时，首先创建一个新的密钥 k_{e+1}，然后根据旧密钥 k_e 和新密钥 k_{e+1} 构建一个重加密令牌 Δ_{e+1}，重加密算法使用重加密令牌 k_{e+1} 将轮次 e 中的密文更新为轮次 $e+1$ 中的密文，可更新加密正式的定义如下。

定义 4(可更新加密) 可更新加密方案 UE 由一组多项式时间的算法(GenKey，GenTok，Enc，Dec，ReEnc)组成：

1)$k\leftarrow$UE.GenKey(λ)：密钥生成算法，输入安全参数 λ，输出密钥 $k\in K$。

2)$\Delta_{e+1}\leftarrow$UE.GenTok(k_e,k_{e+1})：重加密令牌生成算法，输入密钥 k_e 和 k_{e+1}，算法输出重加密令牌 Δ_{e+1}。

3)$c_e\leftarrow$UE.Enc(k_e,m)：加密算法，输入密钥 k_e 和明文 $m\in M$，输出密文 c_e。

4)$m\leftarrow$UE.Dec(k_e,c_e)：解密算法，输入密钥 k_e 和密文 c_e，输出明文 m 或 $\perp$。

5)$c_{e+1}\leftarrow$UE.ReEnc(Δ_{e+1},c_e)：重加密算法，输入密文 c_e 和重加密令牌 Δ_{e+1}，输出新的密文 c_{e+1}。

其中，$\boldsymbol{K}$ 表示密钥空间，$\boldsymbol{M}$ 表示消息空间。

给定 UE，SKE$=$(GenKey，Enc，Dec)是 UE 的底层加密方案。如果 SKE 是正确的，并且 $\forall k^{\mathrm{old}},k^{\mathrm{new}}\leftarrow$GenKey$(\lambda)$，$\forall\Delta\leftarrow$GenTok$(k^{\mathrm{old}},k^{\mathrm{new}})$，$\forall c\in\boldsymbol{C}$，有 Dec$(k^{\mathrm{new}},$ReEnc$(\Delta,c))=Dec(k^{\mathrm{old}},c)$，那么称 UE 是正确的。

下面给出 Boyd 等[35]的 UE 方案 SHINE0 的具体构造(SHINE0 只能加密短消息)：

1)SHINE0.GenKey(λ)：$k\leftarrow\boldsymbol{Z}_q^*$，从 $\boldsymbol{Z}_q^*$ 上随机选择一个元素。

2)SHINE0.GenTok(k_e,k_{e+1})：$\Delta_{e+1}\leftarrow k_{e+1}/k_e$，用新密钥除以旧密钥，得到重加密令牌。

3)SHINE0.Enc(k_e,m)：$\boldsymbol{\alpha}\xleftarrow{\$}\boldsymbol{\alpha}$，$c_e\leftarrow(\pi(\boldsymbol{\alpha}\parallel m\parallel 0^t))^{k_e}$，从随机噪声空间 $\boldsymbol{\alpha}$ 取向量 $\boldsymbol{N}$，拼接后使用 π 进行转置操作，然后计算其 k_e 次幂。

4)SHINE0.Dec(k_e,c_e)：$a\leftarrow\pi^{-1}(c_e^{1/k_e})$，加密的逆变换得到 a，然后分解 a 为 $\boldsymbol{Z}\parallel m\parallel\boldsymbol{Z}$，如果 $\boldsymbol{Z}=0^t$，则解密成功，否则解密失败。

5)SHINE0.ReEnc(Δ_{e+1},c_e)：$c_{e+1}\leftarrow(c_e)^{\Delta_{e+1}}$，计算旧密文 c_e 的 Δ_{e+1} 次幂。

4 方案设计

4.1 系统模型

基于可更新加密，本文提出了一种前向安全的高效的隐藏搜索模式的 DSSE 方案 UEDSSE。该方案的主要设计目标是减少查询中的搜索模式泄露并高效地执行检索和更新操作。我们主要考虑除云服务器以外的外部敌手，他们是恶意的，可以通过观察客户端与云服务器端的通信信息来推测用户的隐私信息。UEDSSE 方案的设计思想如下：更新时使用可更新加密算法加密文档标识符，在搜索时生成一个新的加密密钥，并根据旧密钥和新密钥使用重加密令牌生成算法生成重加密令牌。执行搜索操作时将搜索令牌和重加密令牌加密后上传到云服务器，云服务器重加密检索到的文档标识符并返回。这使得即使用相同关键词进行查询每次返回的文档标识符密文数据也不相同，外部敌手无法推测两次查询是否对应同一个关键词。

4.2 详细构造

UEDSSE 方案由系统初始化算法 Setup、更新协议 Update 和搜索协议 Search 组成。该方案使用 1 个伪随机函数 $F:\{0,1\}^{\lambda}\times\{0,1\}^{*}\rightarrow\{0,1\}^{\lambda}$，3 个抗碰撞的安全哈希函数 h，$H_1:\{0,1\}^{*}\rightarrow\{0,1\}^{l}$，$H_2:\{0,1\}^{*}\rightarrow\{0,1\}^{2\lambda}$（$l$ 为映射位置的长度），1 个选择明文攻击下不可区分（Indistinguishability Under chosen-plaintext Attack，IND-CPA）安全的公钥加密算法 PKE=(GenKey, Enc, Dec) 和一个 IND-CPA 安全的对称加密算法 π=(GenKey, Enc, Dec)。图 1 给出了更新和搜索协议中的数据结构。下文将展示方案的具体构造。

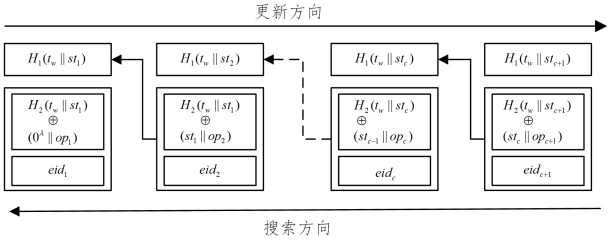

图 1 UEDSSE 中的存储结构

Fig. 1 Storage structure in UEDSSE

1) Setup(λ, ***DB***, $\perp$)。系统初始化算法，生成系统密钥和初始化存储云服务器和客户端数据的密文数据库。客户端输入安全参数 λ，生成一个密钥 k_s，当发起一个搜索或者更新查询时，客户端会从密钥 k_s 动态生成两个子密钥，这种方法避免了在客户端存储过多的密钥，减少了空间消耗。客户端生成一个空映射 ***WC***，保存关键词 w 的两个状态值 (st, k_w)，st 是一个随机比特串，在每次更新 w 的索引信息时随机生成，k_w 是文档标识符加密密钥，在每次搜索 w 时重新生成。云服务器端初始化一个空映射 ***T***，用于存储历史更新消息，然后运行 PKE 算法生成公私钥对 (pk, sk)，并将公钥 pk 发送给用户。

算法 1 Setup 算法

输入：安全参数 λ

输出：k_s，初始化后的存储结构 **WC** 和 **T**

客户端：

1. $k_s \xleftarrow{\$} \{0,1\}^{\lambda}$；
2. **WC**←empty map；
3. RETURN k_s, **WC**

云服务器端：

4. T←empty map；
5. (pk, sk)←PKE. GenKey()；
6. 发送 pk 给客户端。

2) Update(k_s, w, op, id, ***WC***, ***EDB***)：更新协议，客户端和云服务器之间进行交互。客户端可以向云服务器中添加/删除一个文档或添加/删除一个关键词-文档标识符对。当客户想要添加/删除一个关键词-文档标识符对 (w, id) 时，输入 (op, w, id)，其中 $op=add$ 或 del，系统会生成相应的更新令牌并发送给云服务器，云服务器保存更新数据。对于添加/删除文档的情况，客户端可以提取出文档的关键词 $(w_1, w_2, \cdots, w_t)$，构建多个关键词-文档标识符对 $\{(w_1, id), (w_2, id), \cdots, (w_t, id)\}$，对多个关键词-文档标识符对执行 Update 算法。算法 2 仅展示了单个关键词-文档标识符对的更新算法。

客户端首先生成 t_w 和新的状态信息 st_{c+1} 并查找 ***WC*** 中是否存在关键词 w 的信息，如果存在 w 的信息则计算 $u \leftarrow (st_c \| op) \oplus H_2(t_w \| st_{c+1})$，否则计算 $u \leftarrow (0^{\lambda} \| op) \oplus H_2(t_w \| st_{c+1})$，并生成密钥 k_w，其中 u 记录了更新操作信息。之后将新的状态信息 (st_{c+1}, k_w) 保存到 ***WC***[w] 并利用 t_w 计算索引 l，然后将 (l, u, eid) 发送到云服务器。云服务器将 (u, eid) 保存在 ***T***[l] 中。

算法 2 Update 算法

输入：k_s, op, w, id, **WC**, **T**

输出：更新后的存储结构 **WC** 和 **T**

客户端：

1. $t_w \leftarrow F(k_s, h(w))$；
2. $st_{c+1} \xleftarrow{\$} \{0,1\}^{\lambda}$；
3. $(st_c, k_w) \leftarrow$ **WC**[w]；
4. IF $(st_c, k_w) = \perp$ THEN
5. $k_w \xleftarrow{\$} \mathbf{Z}_q^{*}$；
6. $u \leftarrow (0^{\lambda} \| op) \oplus H_2(t_w \| st_{c+1})$；
7. ELSE
8. $u \leftarrow (st_c \| op) \oplus H_2(t_w \| st_{c+1})$；
9. END IF
10. $\mathbf{N} \xleftarrow{\$} \mathbf{N}$；
11. $eid \leftarrow \pi.\mathrm{Enc}(k_w, N \| id \| 0^t)_{k_w}$；
12. **WC**[w]←(st_{c+1}, k_w)；
13. $l \leftarrow H_1(t_w \| st_{c+1})$；
14. 将 (l, u, eid) 发送给云服务器。

云服务器端：

15. **T**[l]←(u, eid)；

3) Search(k_s, w, ***WC***, ***T***)：搜索协议，在客户端和云服务器之间执行。协议主要包括以下几个步骤：客户端生成关键词 w 的搜索陷门和重加密令牌，并上传到云服务器；云服务器执行搜索操作，重加密匹配的文件标识符并返回；客户端接收并解密文件标识符。具体步骤如算法 3 所示。

客户端首先获取关键词 w 的状态信息，如果为空，则说明不存在该关键词对应的文件，返回空。若不为空，则生成令牌 t_w 和新密钥 k_w'，并根据 k_w 和 k_w' 生成重加密令牌 Δ。为了隐藏搜索模式，客户端生成随机字符串 $mask$，加密 t_w，st_c 和 Δ。然后客户端使用公钥 pk 加密随机字符串 $mask$ 得到 $emask$，并将 $(t_w, st_c, \Delta, emask)$ 发送给云服务器。

云服务器初始化两个空集合 ***ID*** 和 ***Del***，分别用来保存未删除和已删除的文件标识符 eid。然后云服务器用私钥 sk 解密 $emask$ 得到 $mask$，再用 $mask$ 解密 t_w，st_c 和 Δ。接着，云服务器计算索引 l，从 T 中获取密文 (u, eid) 并解密 u 为 $(st_c \| op)$。如果 op 为 del，则将对应的 eid 添加到 ***Del*** 中；如果 op 为 add，则判断 ***Del*** 中是否包含 eid，如果 ***Del*** 中包含则从 ***Del*** 中删除 eid，表示先前的添加操作被后面的删除操作删除，如果 ***Del*** 中不包含 eid 则运行重加密算法更新 eid 的密文为 eid'，并添加 eid' 到集合 ***ID*** 中。当 $st_c = 0^\lambda$ 时，表明遍历结束，返回 ***ID*** 给客户端。客户端使用密钥 k_w' 解密所有文件标识符获得明文文档标识符，搜索协议结束。

算法 3 Searh 算法

输入：k_s，w，**WC**，**T**

输出：R

客户端：

1. $(st_c, k_w) \leftarrow$ **WC**[w]；
2. IF $(st_c, k_w) = \perp$
3. RETURN $\emptyset$；
4. END IF
5. $t_w \leftarrow F(k_s, h(w))$；
6. $k_w' \xleftarrow{\$} \mathbf{Z}_q^*$；
7. $\Delta \leftarrow k_w'/k_w$；
8. mask$\leftarrow \{0,1\}^{2\lambda}$；
9. $t_w \leftarrow t_w \oplus$ mask；
10. $st_c \leftarrow st_c \oplus$ mask；
11. $\Delta \leftarrow \Delta \oplus$ mask；
12. emask$\leftarrow$PKE. Enc(pk，mask)；
13. 将$(t_w, st_c, \Delta$，emask)发送给云服务器。

云服务器端：

14. 初始化空集合 **ID**，R$\leftarrow\emptyset$；
15. mask$\leftarrow$PKE. Dec(sk，emask)；
16. $t_w \leftarrow t_w \oplus$ mask；
17. $st_c \leftarrow st_c \oplus$ mask；
18. $\Delta \leftarrow \Delta \oplus$ mask；
19. WHILE $st_c != 0^\lambda$ DO
20. $l \leftarrow H_1(t_w \| st_c)$；
21. $(u, eid) \leftarrow T[l]$；
22. $(st_c \| op) \leftarrow u \oplus H_2(t_w \| st_c)$；
23. IF op=del THEN
24. **Del**$\leftarrow$**Del**$\cup$eid；
25. ELSE IF op=add THEN
26. IF eid$\in$**Del** THEN
27. **Del**$\leftarrow$**Del**$-$eid；
28. ELSE
29. $eid' \leftarrow eid^{\Delta}$；
30. **ID**$\leftarrow$**ID**$\cup$eid′；
31. END IF
32. END IF
33. END WHILE
34. 返回 **ID** 给客户端

客户端：

35. **WC**[w]$\leftarrow(st_c, k_w')$；
36. FOR eid$\in$ID
37. r$\leftarrow\pi$. Dec$(eid^{1/k_w'})$；
38. R=R$\cup$r；
39. END FOR
40. RETURN R

值得注意的是，上述方案主要建立在诚实且不好奇的服务器安全模型下。即方案中的云服务器是诚实的，不会作为敌手对数据库发起被动攻击。此时系统的威胁主要是除服务器以外的外部敌手。后续的工作中，我们将进一步弱化该假设，考虑诚实且好奇的云服务器安全模型。该场景下，云服务器可能对用户的查询内容感兴趣，并尝试利用所观察到的查询信息学习用户的查询内容。

5 安全分析及证明

本章将分析本文方案的安全性。

表 2 列出了 UEDSSE 方案与其他方案的安全性对比结果，包括前向安全、后向安全、搜索模式和访问模式。从表 2 中可以发现，现有 DSSE 方案无法有效保护搜索模式、访问模式等泄露信息。本文方案弥补了上述方案的不足，提供了更高的安全性能。

表 2 UEDSSE 与其他方案的安全性对比

Table 2 Security comparison between UEDSSE and other schemes

方案	前向安全	后向安全	搜索模式	访问模式
方案[15]	√	√	×	×
方案[20]	√	√	×	×
文献[36]	√	×	×	×
UEDSSE	√	×	√	√

由于尚未有利用后向安全攻击 DSSE 方案的方法，因此本文为了效率原因未考虑 DSSE 方案的后向安全，UEDSSE 也易改造成具有后向安全的 DSSE 方案。

下面证明 UEDSSE 方案的安全性。我们用 Q 来表示执行的一系列操作，每个操作的形式为 (t, w)（搜索）或 (t, op, w, id)（更新），t 为时间戳。定义两个泄露模式：1）搜索模式 $sp(w) = \{t | (t, w) \in Q\}$ 返回查询关键词 w 的时间戳；2）更新历史 $\text{UpdHis}(w) = \{(t, op, id) | (t, op, w, id) \in Q\}$，表示文档标识符更新的时间。

定理 1 若 F 是一个伪随机函数，PKE 是 IND-CPA 安全的公钥加密算法，π 是 IND-CPA 安全的对称加密算法，H_1，H_2，h 是模拟为随机预言机的哈希函数，定义泄露函数 $= (\mathcal{L}^{\text{Setup}}, \mathcal{L}^{\text{Search}}, \mathcal{L}^{\text{update}})$ 为：

$$\mathcal{L}^{\text{Setup}} = (|\boldsymbol{DB}|)$$

$$\mathcal{L}^{\text{Search}}(w) = (\text{sp}(w), \text{UpdHis}(w))$$

$$\mathcal{L}^{\text{Update}} = (op, id)$$

则本文的 UEDSSE 是具有 $\mathcal{L}$-自适应安全的保护搜索

模式的前向安全的 DSSE 方案。

证明：我们采用理想-现实模型来证明本文方案的安全性。我们构造一系列的游戏和模拟器 S 模拟真实世界到理想世界的一条路径，通过证明游戏间的不可区分性来证明真实世界与理想世界的不可区分性。

$Game_0$：$Game_0$ 即真实世界的 DSSE 游戏，因此有：

$$|Pr[Game_0=1]-Pr[Real_A^{\Pi}=1]|=0 \tag{3}$$

$Game_1$：与$Game_0$ 使用伪随机函数生成 t_w 不同，$Game_1$ 维护一个查找表 $\boldsymbol{MF}$ 并存储值(w,t_w)。当需要 t_w 时，首先根据 w 在查找表中查找 t_w 是否存在，如果存在，则 $t_w \leftarrow \boldsymbol{MF}[w]$；否则 $t_w \xleftarrow{\$} \{0,1\}^{\lambda}$，并将 t_w 存储到 $\boldsymbol{MF}[w]$中。因为真随机函数和伪随机函数是不可区分的，所以有：

$$|Pr[Game_1=1]-Pr[Game_0=1]|\leqslant negl(\lambda) \tag{4}$$

$Game_2$：更新协议中，$Game_2$ 使用随机字符串代替密文 eid，并用 $\boldsymbol{E}$ 保存。由于加密方案是 IND-CPA 安全的，模拟的随机字符串和真实的密文在随机预言机下有相同的分布，因此：

$$|Pr[Game_2=1]-Pr[Game_1=1]|\leqslant negl(\lambda) \tag{5}$$

算法 4 Game_3

Setup

客户端：

1. $k_s \xleftarrow{\$} \{0,1\}^{\lambda}$；
2. **WC**，**MF**，L←empty map；

Update

客户端：

1. IF **MF**[w]=⊥ THEN
2. $\quad t_w \xleftarrow{\$} \{0,1\}^{\lambda}$；
3. $\quad$ **MF**[w]←t_w；
4. ELSE
5. $\quad t_w \leftarrow$ **MF**[w]；
6. END IF；
7. $st_{c+1} \xleftarrow{\$} \{0,1\}^{\lambda}$；
8. $(st_c, k_w) \leftarrow$ **WC**[w]；
9. IF(st_c, k_w)=⊥ THEN
10. $\quad k_w \xleftarrow{\$} \mathbf{Z}_q^*$；
11. $\quad u \leftarrow (0^{\lambda} \| op) \oplus H_2(t_w \| \| st_{c+1})$；
12. ELSE
13. $\quad u \leftarrow (st_c \| op) \oplus H_2(t_w \| st_{c+1})$；
14. END IF；
15. $eid \xleftarrow{\$} \{0,1\}^{\lambda}$；
16. **WC**[w]←(st_{c+1}, k_w)；
17. $l \xleftarrow{\$} \{0,1\}^{\lambda}$；
18. $L[t_w \| st_{c+1}] \leftarrow l$；
19. 将(l,u,eid)发送到云服务器。

Search

客户端：

1. $(st_c, k_w) \leftarrow$ **WC**[w]；
2. IF(st_c, k_w)=⊥ THEN
3. $\quad$ RETURN $\bar{E}$
4. END IF
5. FOR i=0 to c DO
6. $\quad H_1[t_w \| st_{c+1}] \leftarrow L[t_w \| st_{c+1}]$；
7. END FOR
8. $t_w \leftarrow$ **MF**[w]；
9. $k_w' \xleftarrow{\$} \mathbf{Z}_q^*$；
10. $\Delta \leftarrow k_w'/k_w$；
11. $mask \xleftarrow{\$} \{0,1\}^{2\lambda}$；
12. $t_w \leftarrow t_w \oplus mask$；
13. $st_c \leftarrow st_c \oplus mask$；
14. $\Delta \leftarrow \Delta \oplus mask$；
15. emask←PKE.Enc(pk,mask)；
16. 将$(t_w, st_c, \Delta, emask)$发送到云服务器。

$Game_3$：$Game_3$ 与$Game_2$ 的不同之处在于$Game_3$ 使用随机字符串代替哈希函数 H_1，具体来说，就是客户端调用 H_1 的代码被替换为：

$l \xleftarrow{\$} \{0,1\}^{\lambda}$

$L[t_w \| st_c] \leftarrow l$

L 是一个映射。在 Search 协议云服务器搜索时增加：

$H_1[t_w \| st_c] \leftarrow L[t_w \| st_c]$

H_1是随机预言机。现在$Game_3$ 和$Game_2$ 完全相同，但敌手可能会观察到随机预言机的不一致性：在$Game_3$ 中，H_1 不会立即更新，而是在下一次搜索查询时添加到H_1中。如果敌手在下一次查询之前使用 $t_w \| st_c$ 询问H_1，得到值为 u'，由于此时H_1还未更新，因此有极大的概率 $u' \neq L[t_w \| st_c]$。将敌手观察到随机预言机不一致的时间记作 Bad，此时有：

$$|Pr[Game_3=1]-Pr[Game_2=1]|\leqslant Pr[\text{Bad}] \tag{6}$$

只有敌手用 $t_w \| st_c$ 查询预言机H_1时 Bad 才会发生。因为 $st_c \xleftarrow{\$} \{0,1\}^{\lambda}$，敌手随机选择 st_c 的概率是 $2^{-\lambda}$，但由于敌手最多只能尝试 $poly(\lambda)$次，因此 $Pr[\text{Bad}]\leqslant poly(\lambda)\cdot(2^{-\lambda})$，故：

$$|Pr[Game_3=1]-Pr[Game_2=1]|\leqslant negl(\lambda) \tag{7}$$

$Game_3$ 的伪代码如算法 4 所示。

$Game_4$：$Game_4$ 与$Game_3$ 类似，只是我们对 H_2 做了和$Game_3$ 中 H_1相同的修改，因此：

$$|Pr[Game_4=1]\text{-}Pr[Game_3=1]|\leqslant negl(\lambda) \tag{8}$$

算法 5 Game_5

Setup

客户端：

1. L，U，**E**，**TW**，**ST**，**DT**，**MS**←empty map；
2. v←0；
3. s←0；

Update

客户端：

1. v←v+1；
2. $L[v] \xleftarrow{\$} \{0,1\}^{\lambda}$；
3. $U[v] \xleftarrow{\$} \{0,1\}^{2\lambda}$；
4. $E[v] \xleftarrow{\$} \{0,1\}^{\lambda}$；
5. 将(L[v]，E[v]，U[v])发送到云服务器

Search

客户端：

1. s←s+1；
2. **TW**[s]$\xleftarrow{\$}$$\{0,1\}^{\lambda}$；
3. **ST**[s]$\xleftarrow{\$}$$\{0,1\}^{\lambda}$；
4. **DT**[s]$\xleftarrow{\$}$$\{0,1\}^{\lambda}$；
5. **MS**[s]$\xleftarrow{\$}$$\{0,1\}^{\lambda}$；
6. 将(**TW**[s]，**ST**[s]，**DT**[s]，**MS**[s])发送到云服务器。

$Game_5$：在$Game_5$ 中，客户端选择随机数作为搜索令牌并发送给云服务器。在$Game_4$ 中，用户的搜索令牌是使用随机数加密的，并且每次搜索时随机生成。而随机数使用公钥加密算法加密，根据公钥加密算法的安全性和一次一密加密算法的安全性，敌手无法破解密文。因此在敌手的视角里，搜索协议输出了 4 个随机数。在$Game_5$ 中，使用 s 记录搜索操作的次数，使用 ***TW***，***ST***，***DT***，***MS*** 分别记录随机数值。在更新协议中，使用 v 记录更新操作次数，从敌手的视角看，更新协议输出的是 3 个随机字符串，因此$Game_5$ 和$Game_4$ 表现相同，即：

$$|Pr[Game_5=1]-Pr[Game_4=1]|\leqslant negl(\lambda) \tag{9}$$

$Game_5$ 的伪代码如算法 5 所示。

$\mathrm{Idea}^{\Pi}_{A,S}(\lambda)$：由模拟器 S 基于泄露函数构造，仅使用方案中定义的泄漏信息作为输入，输出一个仿真的视图。

算法 6 模拟器 S

Setup

客户端：

1. L，U，**E**，**TW**，**ST**，**DT**，**MS**←empty map；
2. v←0；
3. s←0；

Update

客户端：

1. v←v+1；
2. L[v]$\xleftarrow{\$}$$\{0,1\}^{\lambda}$；
3. U[v]$\xleftarrow{\$}$$\{0,1\}^{2\lambda}$；
4. **E**[v]$\xleftarrow{\$}$$\{0,1\}^{\lambda}$；
5. 将(L[v]，**E**[v]，U[v])发送到云服务器

Search

客户端：

1. s←s+1；
2. **TW**[s]$\xleftarrow{\$}$$\{0,1\}^{\lambda}$；
3. **ST**[s]$\xleftarrow{\$}$$\{0,1\}^{\lambda}$；
4. **DT**[s]$\xleftarrow{\$}$$\{0,1\}^{\lambda}$；
5. **MS**[s]$\xleftarrow{\$}$$\{0,1\}^{\lambda}$；
6. 将(**TW**[s]，**ST**[s]，**DT**[s]，**MS**[s])发送到云服务器。

模拟器的算法如算法 6 所示。由于 UEDSSE 方案中关键词 w 在每次搜索时令牌都是随机加密的，并且每次搜索返回的都是重加密后的文档标识符，敌手无法根据搜索令牌和密文结果获取关键词的搜索模式信息，因此对于更新和搜索请求，模拟器模拟产生随机数作为相应的请求令牌。故模拟器产生的视图和 $Game_5$ 是不可区分的。

$$|Pr[\mathrm{Idea}^{\Pi}_{A,S}=1]-Pr[Game_5=1]|\leqslant negl(\lambda) \tag{10}$$

综上所述，可以证明定理 1 成立。

$$|Pr[\mathrm{Real}^{\Pi}_{A}=1]-Pr[\mathrm{Idea}^{\Pi}_{A,S}=1]|\leqslant negl(\lambda) \tag{11}$$

UEDSSE 方案中加密数据是安全的，根据加密方案的安全性，敌手无法根据观察到的重加密令牌和密文文件标识符来攻破加密方案。

6 实验

本章评估了 UEDSSE 方案的性能，并与 FAST[36] 方案进行了性能对比分析。本文采用 C++编写实验代码，基于 miracl 库实现 UE，采用 RocksDB 存储客户端和云服务器上的数据，基于 gRPC 库实现客户端和云服务器之间的通信。该方案中安全参数 λ 设置为 128，H_1，H_2 采用 SHA256 算法，F 采用 CTR 模式的 AES 算法，PKE 采用 ECC 算法。本文也比较了不同网络环境下的方案性能。在广域网(Wide Area Network，WAN)环境下，云服务器部署在具有四核 CPU、8 GB内存和 10 Mbps 带宽的腾讯云服务器上，客户端部署在具有 8 核 CPU(Intel Core i7-10700 2.9 GHz)、16 GB 内存的台式机电脑上，云服务器端和客户端的操作系统均为 Ubuntu 18.04LTS。在局域网(Local Area Network，LAN)环境下，云服务器端和客户端同时部署在台式机电脑上。为了避免偶然误差，每个实验重复 30 次，取平均值。

6.1 计算与通信复杂度分析

表 3 列出了 UEDSSE 与 FAST 方案的理论性能比较结果，其中计算复杂度与通信复杂度为云服务器端的复杂度。在更新操作中，UEDSSE 和 FAST 具有相同的理论计算开销，只存储单个关键词-文档标识符对。在搜索操作中，FAST 方案和 UEDSSE 方案都需要检索 a_w 个历史更新操作并丢弃其中已经被删除的文档标识符，返回 n_w 个结果。不同的是，UEDSSE 方案需要对检索到的结果进行重加密操作，因此搜索计算复杂度为 $O(a_w)+O(n_w)$。但由于本文 UEDSSE 方案使用的可更新加密性能较低，导致实际搜索和更新性能下降。此外，FAST 方案客户端需要为每个关键词存储一个状态位和关键词更新次数，UEDSSE 方案需要为每个关键词存储一个状态位和一个密钥，因此 UEDSSE 的客户端存储开销会稍高于 FAST 方案。

表 3 UEDSSE 与 FAST 理论性能分析

Table 3 UEDSSE and FAST theoretical performance analysis

符号	计算复杂度		通信复杂度		户端存储	隐藏搜索模式
	搜索	更新	搜索	更新		
FAST	$O(a_w)$	$O(1)$	$O(n_w)$	$O(1)$	$O(\lvert W\rvert(\lambda+\log(a_w)))$	×
UEDSSE	$O(a_w)+O(n_w)$	$O(1)$	$O(n_w)$	$O(1)$	$O(\lvert W\rvert(2\lambda))$	√

6.2 实验分析

实验首先测量了 UE 算法的加解密性能。UE 方案虽然是对称密码算法，但其在具体构造中使用了椭圆曲线上的点运算，使得其加解密算法的性能有所下降。实验随机生成了

不同数量级的文档标识符，然后统计加解密算法的平均计算开销。如图 2 所示，在 1000 个文档标识符情况下，平均加密每个文档标识符的计算时间为 0.399 ms，重加密每个文档的计算时间为 0.535 ms，而解密计算时间为 0.528 ms。UE 算法的解密算法和重加密算法的计算开销相差较小，但比加密算法高，这是因为加密算法中计算的指数较小。

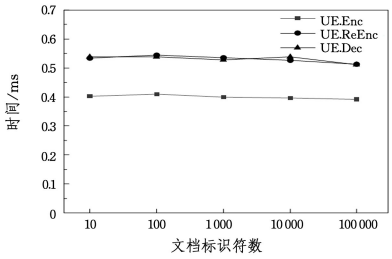

图 2 UE 算法的计算开销

Fig. 2 Computational overhead of UE algorithm

图 3 和图 4 给出了 UEDSSE 和 FAST 方案更新协议的计算开销，实验比较了不同关键词文档对数下的单个键值对的平均更新时间。

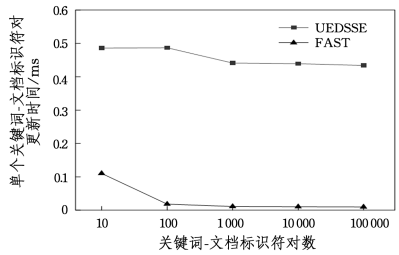

图 3 LAN 下的更新时间

Fig. 3 Update time under LAN

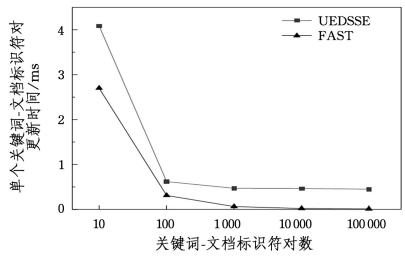

图 4 WAN 下的更新时间

Fig. 4 Update time under WAN

如图 3 所示，实验首先测量了 FAST 方案和 UEDSSE 方案在 LAN 环境下的更新效率。在 LAN 环境下，实验结果不包括网络传输导致的延迟误差，体现了方案本身的执行效率。观察图 3 可以发现，FAST 方案中，随着关键词-文档对数的增加，每个关键词-文档标识符对的平均搜索时间缩短。这是因为 FAST 方案在搜索时需要执行初始化操作，如建立数据库连接，从数据库中读取历史数据等。随着更新次数的增加，这些时间消耗被不断平摊，使得平均搜索时间缩短。但在 UEDSSE 方案中，加密计算是更新操作的计算瓶颈，因此 UEDSSE 方案中更新操作的平均时间开销只下降了一点。在 1000 个关键词-文档标识符的情况下，FAST 方案的平均更新时间开销为 0.011 ms，而 UEDSSE 方案的平均更新时间为 0.441 ms。在 WAN 环境下，两种方案的初始化开销都增大，并且因为网络延迟和设备性能等原因，实际性能表现比在 LAN 环境中稍差。

在搜索协议中，云服务器端需要逐条处理加密索引列表来查找匹配的文件标识符。实验统计了 10，100，1 000，10000，100000 这 5 个不同数量级下关键词-文档标识符对的平均搜索时间开销，即客户端生成查询令牌到接收并解密文档标识符的时间。图 5 和图 6 给出了不同网络环境下 FAST 和 UEDSSE 方案的平均搜索时间开销。与更新协议相同，FAST 方案搜索初始化的成本随着关键词-文档标识符数量的增加而分摊，平均搜索时间会缩短。而 UEDSSE 方案在生成搜索令牌时需要使用公钥算法 PKE 加密随机字符串，云服务器端接收令牌时需要使用 PKE 解密字符串，这使得计算开销在匹配文档数为 10 时显著增加。但随着匹配结果的增加，PKE 加密算法的开销被均摊到每个搜索操作中。

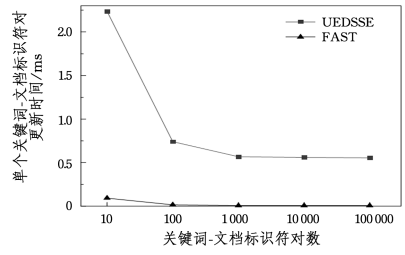

图 5 LAN 下的搜索时间

Fig. 5 Search time under LAN

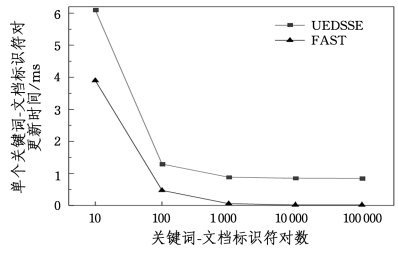

图 6 WAN 下的搜索时间

Fig. 6 Search time under WAN

在 UEDSSE 方案中，搜索协议中云服务器需要对每个密文文档标识符执行重加密算法，客户端需要对返回的文档标识符信息进行解密。由于解密算法的计算开销几乎与重加密算法相同，因此云服务器端可以在执行重加密算法后将新的密文数据返回给客户端，客户端执行解密操作。当云服务器端重加密完新的密文后，客户端将上一条数据解密完成，重加密算法和解密算法的计算开销成为 UEDSSE 方案的性能瓶颈。在单个关键词匹配 100 个文档的情况下，FAST 方案的单个关键词-文档标识符对的时间开销为 0.016 ms，而 UEDSSE 方案的时间开销为 0.74 ms。

实验将密文全部下载到本地解密，重加密后上传，从而隐藏搜索模式的方案记为 SIMPLE。表 4 列出了 UEDSSE 与 SIMPLE 的通信开销的比较结果。实验设置文件标识符密文长度为 32 字节，然后比较了关键词对应不同数量的文档标识符下的通信开销。在 UEDSSE 方案中，搜索只需要发送单个搜索令牌和重加密令牌，然后接收全部密文数据。在 SIMPLE，客户端还需要将所有数据加密后使用更新算法重新

上传到云服务器上，而更新操作需要为每个文档标识符生成索引 l、操作 u 和密文 eid。表 4 列出了关键词对应不同数量文档标识符下的搜索时通信开销比较结果，在返回 100 个文档标识符的情况下，UEDSSE 减少了 70.92%的通信开销。

表 4 通信开销比较

Table 4 Communication overhead comparison

方案	通信开销/kB				
	10	100	1 000	10 000	100 000
SIMPLE	1.13	10.97	109.41	1 093.78	10 937.54
UEDSSE	0.37	3.19	31.31	312.56	3 125.06

结束语 针对现有的动态对称可搜索加密方案在利用更新密文的方式隐藏搜索模式时通信与计算开销大的问题，本文构造了一种基于可更新加密的保护搜索模式的前向安全的动态可搜索加密方案。该方案采用可更新加密算法来加密文档标识符，执行搜索操作时客户端向云服务器发送加密的搜索令牌和重加密令牌，云服务器将重加密后的密文文档标识符返回给客户端。针对同一个关键词的查询，每次返回的密文文档标识符都不同，敌手无法根据这些密文信息推测出搜索模式信息。安全分析表明，本文方案具有自适应安全性和前向安全性并能保护搜索模式，实验结果表明本文方案相比传统利用更新密文方法保护搜索模式的方案能有效降低通信开销。

参 考 文 献

[1] SONG D X, WAGNER D, PERRIG A. Practical techniques for searches on encrypted data[C]// Proceedings of IEEE Symposium on Security and Privacy. California: IEEE Computer Society, 2000: 44-55.

[2] BONEH D, CRESCENZO G D, OSTROVSKY R, et al. Public key encryption with keyword search[C]// Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques. Interlaken: Springer, 2004: 506-522.

[3] XU L, XU C G, YU X L. Secure and Efficient Data Retrieval Scheme Using Searchable Encryption in Cloud[J]. Journal of Cryptologic Research. 2016, 3(4): 330-339.

[4] KAMARA S, PAPAMANTHOU C. Parallel and dynamic searchable symmetric encryption[C]// Proceedings of International Conference on Financial Cryptography and Data Security. Okinawa: Springer, 2013: 258-274.

[5] CURTMOLA R, GARAY J, KAMARA S, et al. Searchable symmetric encryption: improved definitions and efficient constructions[C]// Proceedings of ACM Conference on Computer and Communications Security. Alexandria: ACM, 2006: 79-88.

[6] ISLAM M S, KUZU M, KANTARCIOGLU M. Access pattern disclosure on searchable encryption: ramification, attack and mitigation[C]// Proceedings of Annual Network and Distributed System Security Symposium. San Diego: The Internet Society, 2012.

[7] LIU C, ZHU L, WANG M, et al. Search pattern leakage in searchable encryption: Attacks and new construction[J]. Information Sciences, 2014, 265: 176-188.

[8] OYA S, KERSCHBAUM F. Hiding the access pattern is not enough: Exploiting search pattern leakage in searchable encryption[C]// Proceedings of USENIX Security Symposium. USENIX Association, 2021: 127-142.

[9] 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 USENIX Security Symposium. Austin: USENIX Association, 2016: 707-720.

[10] BONEH D, LEWI K, MONTGOMERY H, et al. Key homomorphic PRFs and their applications[C]// Proceedings of Annual Cryptology Conference. Santa Barbara: Springer, 2013: 410-428.

[11] GOH E J. Secure indexes[J/OL]. http://eprint.iacr.org/2003/216.

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

[13] GOLDREICH O, OSTROVSKY R. Software protection and simulation on oblivious RAMs[J]. Journal of the ACM (JACM), 1996, 43(3): 431-473.

[14] HE K, CHEN J, ZHOU Q, et al. Secure dynamic searchable symmetric encryption with constant client storage cost[J]. IEEE Transactions on Information Forensics and Security, 2020, 16: 1538-1549.

[15] CHEN T, XU P, WANG W, et al. Bestie: Very practical searchable encryption with forward and backward security[C]// Proceedings of European Symposium on Research in Computer Security. Darmstadt: Springer, 2021: 3-23.

[16] BOST R, MINAUD B, OHRIMENKO O. Forward and backward private searchable encryption from constrained cryptographic primitives[C]// Proceedings of ACM SIGSAC Conference on Computer and Communications Security. Dallas: ACM, 2017: 1465-1482.

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

[18] SUN S F, STEINFELD R, LAI S, et al. Practical Non-Interactive Searchable Encryption with Forward and Backward Privacy[C]// Proceedings of Annual Network and Distributed System Security Symposium. The Internet Society, 2021.

[19] WANG Y L, CHEN X F. Research on Searchable Symmetric Encryption[J]. Journal of Electronics & Information Technology, 2020, 42(10): 2374-2385.

[20] LU B J, ZHOU J, CAO Z F. A multi-user forward secure dynamic symmetric searchable encryption with enhanced security[J]. Journal of Computer Research and Development, 2020, 57(10): 2104-2116.

[21] CASH D, JARECKI S, JUTLA C, et al. Highly-scalable searchable symmetric encryption with support for boolean queries[C]// Proceedings of Annual Cryptology Conference. Santa Barbara: Springer, 2013: 353-373.

[22] XU L, YUAN X L, ZHENG X Z, et al. Towards efficient cryptographic data validation service in edge computing[J]. IEEE Transactions on Services Computing, 2021, 16(1): 656-669.

[23] CASH D, GRUBBS P, PERRY J, et al. Leakage-abuse attacks

against searchable encryption[C]// Proceedings of ACM SIGSAC Conference on Computer and Communications Security. Denver: ACM, 2015: 668-679.

[24] OYA S, KERSCHBAUM F. IHOP: Improved Statistical Query Recovery against Searchable Symmetric Encryption through Quadratic Optimization[C]// Proceedings of USENIX Security Symposium. Boston: USENIX Association, 2022: 2407-2424.

[25] DAMIE M, HAHN F, PETER A. A Highly Accurate Query-Recovery Attack againstSearchable Encryption using Non-Indexed Documents[C]// Proceedings of USENIX Security Symposium. USENIX Association, 2021: 143-160.

[26] XU L, DUAN H, ZHOU A, et al. Interpreting and mitigating leakage-abuse attacks in searchable symmetric encryption[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 5310-5325.

[27] CHEN G, LAI T H, REITER M K, et al. Differentially private access patterns for searchable symmetric encryption[C]// Proceedings of IEEE Conference on Computer Communications. Honolulu: IEEE, 2018: 810-818.

[28] KAMARA S, MOATAZ T. Computationally volume-hiding structured encryption[C]// Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. Darmstadt: Springer, 2019: 183-213.

[29] GRUBBS P, KHANDELWAL A, LACHARITÉ M S, et al. Pancake: Frequency smoothing for encrypted data stores[C]// Proceedings of USENIX Security Symposium. USENIX Association, 2020: 2451-2468.

[30] ZHAO Z T, XU Y, SONG X F, et al. A Multi-Pattern Hiding Dynamic Symmetric Searchable Encryption Based on Differential Privacy[J]. Journal of Computer Research and Development, 2021, 58(10): 2287-2300.

[31] ZHENG W, DAVE A, BEEKMAN J G, et al. Opaque: An oblivious and encrypted distributed analytics platform[C]// Proceedings of USENIX Symposium on Networked Systems Design and Implementation. Boston: USENIX Association, 2017: 283-298.

[32] EVERSPAUGH A, PATERSON K, RISTENPART T, et al. Key rotation for authenticated encryption[C]// Proceedings of Annual International Cryptology Conference. Santa Barbara: Springer, 2017: 98-129.

[33] LEHMANN A, TACKMANN B. Updatable encryption with post-compromise security[C]// Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. Tel Aviv: Springer, 2018: 685-716.

[34] KLOOß M, LEHMANN A, RUPP A. (R) CCA secure updatable encryption with integrity protection[C]// Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. Darmstadt: Springer, 2019: 68-99.

[35] BOYD C, DAVIES G T, GJØSTEEN K, et al. Fast and secure updatable encryption[C]// Proceedings of Annual International Cryptology Conference. Santa Barbara: Springer, 2020: 464-493.

[36] SONG X F, DONG C, YUAN D, et al. Forward private searchable symmetric encryption with optimized I/O efficiency[J]. IEEE Transactions on Dependable and Secure Computing, 2018, 17(5): 912-927.

XU Chengzhi, born in 1998, postgraduate, is a student member of CCF(No. 96688G). His main research interests include cryptography and searchable encryption.

XU Lei, born in 1990, Ph. D, associate professor, is a member of CCF(No. 72575M). His main research interests include applied cryptography and information security.

(责任编辑:喻藜)