

# 智能合约辅助下满足前后向安全的动态可搜索加密方案

丁晓晖<sup>1</sup>, 曹素珍<sup>1</sup>, 王彩芬<sup>2</sup>

(1.西北师范大学 计算机科学与工程学院,兰州 730070; 2.深圳技术大学 大数据与互联网学院,广东 深圳 518118)

**摘要:** 动态可搜索加密过程易受文件注入攻击和信息泄露导致的信息滥用攻击,同时现有基于公钥密码体制构造的动态可搜索加密方案往往涉及大量双线性对运算,不能满足实际应用的效率要求。通过引入智能合约,提出一种满足前后向安全的动态可搜索加密方案。以智能合约取代传统的搜索服务器进行关键字陷门匹配测试,解决传统搜索服务器必须满足诚实且好奇的设定问题,在此过程中避免大量使用双线性对运算,而是只执行一些简单的哈希操作,从而提升密文数据搜索阶段的计算效率。该方案满足前向安全性和后向安全性,即旧的搜索陷门不能用于搜索更新后的文件,且后续搜索不会泄露已删除文件所对应的索引信息。分析结果表明,与现有公钥密码体制下的动态可搜索加密方案相比,该方案在安全性和计算效率方面更具优势,适用于大数据通信环境。

**关键词:** 前向安全性;后向安全性;动态可搜索加密;智能合约;公钥密码体制

开放科学(资源服务)标志码(OSID):



**中文引用格式:** 丁晓晖,曹素珍,王彩芬.智能合约辅助下满足前后向安全的动态可搜索加密方案[J].计算机工程,2022,48(7):141-150.

**英文引用格式:** DING X H, CAO S Z, WANG C F. Smart contract-assisted dynamically searchable encryption scheme with forward and backward security[J]. Computer Engineering, 2022, 48(7): 141-150.

## Smart Contract-Assisted Dynamically Searchable Encryption Scheme with Forward and Backward Security

DING Xiaohui<sup>1</sup>, CAO Suzhen<sup>1</sup>, WANG Caifen<sup>2</sup>

(1.College of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, China;

2.College of Big Data and Internet, Shenzhen Technology University, Shenzhen, Guangdong 518118, China)

**[Abstract]** The dynamic searchable encryption process is vulnerable to file injection attacks and information abuse attacks caused by information leakage. Furthermore, the existing dynamic searchable encryption schemes based on public key cryptosystem often involve a large number of bilinear pairings, which are inefficient with respect to practical applications. Through the introduction of Smart Contract (SC), a dynamic searchable encryption scheme with forward and backward security is proposed. The keyword trapdoor matching test is carried out by replacing the traditional search server with SC, which solves the setting problem of honesty and curiosity that a traditional search server must pass. Furthermore, it avoids the extensive use of bilinear pairings in this process; to the contrary, it only performs some simple hashing operations to improve the computational efficiency of the ciphertext data search phase. The scheme meets the forward and backward securities: namely, the old search trapdoor cannot be used to search the updated files, and the subsequent search will not disclose the index information corresponding to the deleted files. The analysis results show that, compared with existing dynamic searchable encryption schemes under the public key cryptosystem, this scheme offers some advantages in security and computational efficiency, and it is more suitable for the big data communication environment.

**[Key words]** forward security; backward security; dynamic searchable encryption; Smart Contract (SC); public key cryptosystem

DOI: 10. 19678/j. issn. 1000-3428. 0062919

### 0 概述

随着云计算技术的发展<sup>[1-3]</sup>,越来越多的数据用户(Data User, DU)选择将数据存储云服务器

(Cloud Server, CS)上。但由于云服务器是半可信的,因此不能保证上传数据的机密性与完整性。虽然数据属主(Data Owner, DO)可以先将数据进行加密再上传到云服务器以保证数据的机密性,但数据

基金项目:国家自然科学基金(61662069,61662071)。

作者简介:丁晓晖(1997—),男,硕士研究生,主研方向为密码学、信息安全;曹素珍,副教授;王彩芬,教授、博士。

收稿日期:2021-10-11 修回日期:2021-12-02 E-mail:576342353@qq.com

用户如何从云服务器处搜索密文是一大难题。使用既可保证数据机密性又具有良好密文检索性的可搜索加密技术,能够有效解决这一问题<sup>[4-5]</sup>。在现有的可搜索加密研究中,搜索云服务器通常被认为是诚实且好奇的<sup>[6-7]</sup>,然而在现实生活中,搜索云服务器可能会因为商业利益等原因返回部分搜索结果,存在不诚实行为,而且由于搜索云服务器可以无限制地执行密文搜索陷门的匹配测试,因此许多可搜索加密方案无法抵抗内部关键字猜测攻击。研究者考虑到智能合约(Smart Contract, SC)技术具有透明性以及不可修改性,陆续提出一些基于智能合约的可搜索加密方案用来抵抗内部关键字猜测攻击。然而,智能合约技术的透明性对用户隐私构成了较大威胁。例如,文献[8]提出的方案在运行智能合约时可能会泄露关键字搜索密钥,这就导致了攻击者可以获取数据用户所进行的查询信息,甚至可以根据泄露的关键字信息获得加密数据的部分信息,造成严重的信息泄露。在可搜索加密中,信息泄露通常会引发一系列的安全问题<sup>[9]</sup>。2012年,ISIAM等<sup>[10]</sup>分析了在可搜索解密中信息泄露可能带来的后果,并指出即使是很小的泄露也可能造成严重的安全隐患。2016年,ZHANG等<sup>[11]</sup>指出,即便是泄漏率很低的可搜索加密方案,也可以通过简单的文件注入攻击来完整地揭示用户在客户端的查询。因此,为了抵抗泄露滥用攻击以及文件注入攻击,近年来提出的可搜索加密方案强调前向安全与后向安全<sup>[12]</sup>这2种新的安全属性。

文献[13]基于对称密码体制提出了一个满足前向安全的可搜索加密方案,该方案相对于文献[14]提出的方案具有更高的输入输出效率。文献[15]等基于键控区块链技术构造了具有前向安全性的可搜索加密方案,该方案在搜索阶段和更新及陷门生成阶段的效率分别是文献[14]方案的4倍和300倍。多数方案往往会忽略后向安全对一个动态可搜索加密方案的重要性,直到2017年BOST等在ACM SIGSAC会议上给出了关于动态可搜索加密后向安全的正式定义<sup>[16]</sup>,其将后向安全的等级分为3类,分别是插入模式下的后向安全、更新模式下的后向安全和弱后向安全。基于文献[16]的工作,文献[12,17]等分别提出了满足后向安全的动态可搜索加密方案,其在构造方式上较文献[16]有一定的改进。然而,以上提出的这些具有前向或者后向安全性的方案大多是基于对称密码体制的,而对称密码体制在部分实际应用环境中存在密钥管理的局限性。2004年,BONEH等<sup>[18]</sup>提出了基于公钥密码体制的可搜索加密方案,此后,很多公钥可搜索加密方案陆续被提出<sup>[19-20]</sup>。然而,上述这些公钥可搜索加密方案均不满足前后向安全性。2016年,BOST等<sup>[14]</sup>在CCS会议上提出了一个满足前向安全的公钥可搜索加密方案,但该方案涉及大量的双线性对运算操作,方案效率较低。文献[21]提出的公钥可搜索加密方案虽然效率较文献[14]提出的方案有所提

高,但其仅仅满足前向安全性。由此可见,设计可抵抗内部关键字猜测攻击且满足前后向安全性的动态公钥可搜索加密方案是很有必要的。

现有多数可搜索加密方案主要存在以下3个方面的缺陷:首先,由于搜索服务器可以无限制地执行搜索陷门匹配测试,导致大部分方案不能抵抗内部关键字猜测攻击;其次,为了提高密文检索的效率,动态的可搜索加密方案或多或少都会泄露一定的数据信息,但过多的信息泄露会导致信息泄露滥用攻击以及文件注入攻击,应在满足密文检索效率的基础上尽可能地减少数据信息的泄露,实现前向安全与后向安全;最后,基于公钥密码体制构造的可搜索加密方案往往涉及大量的双线性对运算操作,从而导致方案效率较低,不适用于大数据通信的现实环境。

针对上述问题,本文提出一种智能合约辅助的动态可搜索加密方案FBSEBS。该方案只根据输入返回正确且不可变的结果,并使用智能合约代替搜索服务器进行关键字陷门匹配测试,可抵抗内部关键字猜测攻击,并且其满足前向安全性与后向安全性,能够抵抗信息泄露滥用攻击以及文件注入攻击。在本文方案的构造过程中避免使用大量的双线性对操作,从而提高搜索效率。

## 1 预备知识

### 1.1 符号说明

FBSEBS方案中主要符号的说明如表1所示。

表1 FBSEBS方案中的符号说明

Table 1 Symbol description in FBSEBS scheme	
符号	符号说明
$\lambda$	系统参数
$SK_{du}$	数据用户私钥
$PK_{du}$	数据用户公钥
$F/F^{-1}$	伪随机置换函数
$G[]$	关键字状态键值函数
$T[]$	数据库更新键值函数
$D$	数据库
$O$	插入与删除操作
$FC$	文件集合
$W$	关键字集合
$W_i$	文件包含的关键字
$D(W_i)$	包含关键字 $W_i$ 的文件索引集合
$FC_j^{W_i}$	第 $j$ 个包含关键字 $W_i$ 的文件索引
$(st_c^{W_i}, c)$	关键字 $W_i$ 的当前状态为 $c$
$C_{dv}$	数据库当前状态
$EI_{FC_j^{W_i}}$	文件索引 $FC_j^{W_i}$ 的加密索引集
$\langle K_{st_{c,i}^{W_i}}, V_{st_{c,i}^{W_i}} \rangle$	关键字 $W_i$ 的状态密文
$\langle K_{FC_j^{W_i}}, V_{FC_j^{W_i}} \rangle$	文件索引 $FC_j^{W_i}$ 的索引密文
$\langle K_{CE_{W_i}}, V_{CE_{W_i}} \rangle$	关键字 $W_i$ 的授权密文
$T_{W_i}$	关键字陷门

## 1.2 双线性映射

设 $q$ 为一个素数, $G_1, G_2$ 是阶为 $q$ 的两个循环群。定义一个双线性映射 $e: G_1 \times G_1 \rightarrow G_2$ ,其具有以下性质:

- 1) 双线性性: 对任意 $g \in G_1, a, b \in \mathbb{Z}_q^*$ , 有 $e(g^a, g^b) = e(g, g)^{ab}$ 。
- 2) 非退化性: 存在 $g \in G_1$ , 使得 $e(g, g) \neq 1$ 。
- 3) 可计算性: 对于每一个 $g \in G_1$ , 存在一个有效的算法计算 $e(g, g)$ 。

## 1.3 困难问题

本文方案存在的困难问题DBDH<sup>[22]</sup>描述如下: 存在一个双线性映射 $e: G_1 \times G_1 \rightarrow G_2, g_1$ 为群 $G_1$ 的生成元, 那么不存在概率多项式算法能够以不可忽略的优势 $\varepsilon$ 区分给定的数组 $(g_1, g_1^a, g_1^b, g_1^c, e(g_1, g_1)^{abc}), (g_1, g_1^a, g_1^b, g_1^c, e(g_1, g_1)^z)$ , 其中 $a, b, c, z$ 随机选择于有限域 $\mathbb{Z}_q^*$ 中。

## 1.4 伪随机置换函数

文献[23]给出了伪随机置换函数的概念, 其为一个多项式时间的可计算函数, 本文定义一个函数 $F: \{0, 1\}^L \times \{0, 1\}^L \rightarrow \{0, 1\}^L$ 是伪随机置换函数, 其满足以下定义: 1) 对于任何的 $A \leftarrow \{0, 1\}^L$ , 函数 $F$ 是一个从 $\{0, 1\}^L$ 到 $\{0, 1\}^L$ 的映射; 2) 对于任何多项式时间内的敌手, 都无法准确地区分伪随机置换函数与随机函数; 3) 对于任意的 $A \leftarrow \{0, 1\}^L, x \leftarrow \{0, 1\}^L$ , 存在一个有效的算法计算 $F_A(x): \{0, 1\}^L \rightarrow \{0, 1\}^L$ 。此外, 如果一个函数满足 $F_A(x)=y$ , 且 $F_A^{-1}(y)=x$ , 则称 $F^{-1}$ 为伪随机置换函数 $F$ 的逆函数。

## 1.5 泄露函数

为使动态可搜索加密方案具有更高的搜索效率, 通常允许向云服务器泄露一定的信息。根据文献[13]的工作, 定义一个泄露函数 $L=(L^{\text{Setup}}, L^{\text{Search}}, L^{\text{Update}})$ 来说明允许向敌手泄露的信息。因此, 一个动态可搜索加密方案的机密性可以表述为泄露的信息不多于泄露函数所允许泄露的信息。

## 1.6 前向安全性与后向安全性

### 1.6.1 前向安全性

前向安全性<sup>[16]</sup>是指在进行更新操作时不会泄露任何有关关键字的信息, 或者旧的搜索陷门不能用于搜索更新后的文件, 具体定义如下:

一个自适应安全的动态可搜索加密方案, 如果其更新泄露函数可以写为 $L^{\text{Update}}(O, W_i, FC_j) = L^{\text{Update}}(O, FC_j)$ , 则其满足前向安全性。其中: $L^{\text{Update}}$ 是一个无状态函数。

### 1.6.2 后向安全性

后向安全性是指搜索算法不应泄露已经删除的文件标识符, 即后续搜索不会泄露已删除文件所对应的索引信息。本文方案满足文献[16]提出的第3类后向隐私。在给出具体定义之前, 首先定义以下函数:

1)  $\text{TimeD}(W_i)$ : 返回已经添加到数据库中且随后没有被删除的关键字 $W_i$ 的所有时间戳/文件标识符的集合。

$$\text{TimeD}(W_i) = \{(u, FC_j) | (u, \text{add}, (W_i, FC_j)) \in FC \text{ and } (u', \text{del}, (W_i, FC_j)) \notin FC\}$$

其中: $u$ 表示时间戳。

2)  $\text{DleHist}(W_i)$ : 向敌手返回所有已经删除的历史条目。条目中包含插入时间戳 $u^{\text{add}}$ 和删除时间戳 $u^{\text{del}}$ , 此外, 明确表示哪个删除对应哪个添加。

$$\text{DleHist}(W_i) = \{(u^{\text{add}}, u^{\text{del}}) | \exists FC_j: (u^{\text{add}}, \text{add}, (W_i, FC_j)) \in FC \text{ and } (u^{\text{del}}, \text{del}, (W_i, FC_j)) \in FC\}$$

基于上述定义, 一个自适应安全的动态可搜索加密方案, 若其更新泄露函数可以表示为 $L^{\text{Update}}(O, W_i, FC_j) = L^{\text{Update}}(O, FC_j)$ , 搜索泄露函数可以表示为 $L^{\text{Search}}(W_i) = L^{\text{Search}}(\text{TimeD}(W_i), \text{DleHist}(W_i))$ , 则算法满足后向安全性。其中: $L^{\text{Update}}, L^{\text{Search}}$ 是无状态函数。

## 2 系统模型

在本文提出的FBSEBS方案中, 共存在5个实体, 分别是数据属主(DO)、可信授权中心(Trusted Authorization, TA)、数据用户(DU)、智能合约(SC)以及云服务器(CS)。DO受限于自身的存储及计算能力, 首先会选择将加密的文件索引集以及对应的关键字密文上传到SC, 然后将文件密文以及对应的文件索引集上传到CS。当DU想要从CS处搜索准确有效的数据文件时, 会使用自己的私钥以及关键字构造关键字陷门上传到SC进行搜索, 搜索完成后DU会获得相应的文件加密索引, 利用自己的私钥进行解密后便可根据文件索引从CS中获得DO外包的加密数据文件。FBSEBS系统模型架构如图1所示, 具体步骤如下:

1) 密钥生成: 如图1中第①步所示, TA会计算生成系统的主公私钥对以及DU的公私钥对, DU在收到公私钥对后, 会秘密保存自己的私钥, 并将自己的公钥公开。

2) 生成关键字密文: 如图1中第②步所示, DO会生成关键字密文及其对应的加密文件索引集, 并将他们上传到SC供DU搜索。

3) 加密的数据文件外包: 如图1中第③步所示, DO将加密的文件密文以及对应的文件索引集外包给CS。本文方案主要侧重于通过文件索引的构建来实现前向与后向安全性, 因此, 对于文件的加密与解密没有进行详细的描述。

4) 关键字搜索: 如图1中第④步所示, DU使用自身私钥以及关键字生成关键字陷门, 并将其上传到SC进行搜索, 搜索完成后SC返回加密的文件索引集给DU。

5) 请求与返回数据: 如图1中第⑤步所示, DU解密由第④步获得的加密文件索引集, 并根据文件索引集从CS处获得对应的由DO外包的加密文件。



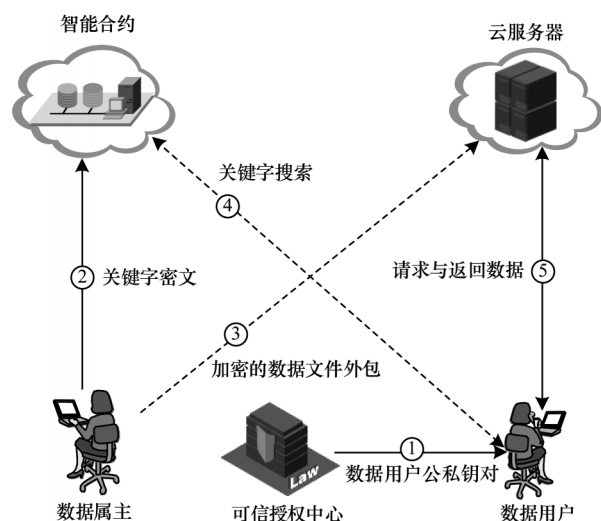


图1 FBSEBS系统模型架构

Fig.1 Structure of FBSEBS system model

### 3 算法描述

FBSEBS方案由以下6个算法构成:

1) 系统建立算法  $\text{Setup}(\lambda)$ : 该算法由TA执行, TA输入系统安全参数 $\lambda$ , 计算得出系统参数 $\text{para}$ 。

2) 密钥生成算法  $\text{KeyGen}(\text{para})$ : 该算法由TA执行, TA输入系统参数, 计算得出系统的主公私钥对 $(s, PK)$ 以及DU的公私钥对 $(SK_{du}, SK_{du})$ 。

3) 数据更新阶段算法  $\text{Update}(\text{para}, D, PK_{du}, W_i)$ : 该算法由DO执行, DO输入系统参数 $\text{para}$ 、数据库 $D$ 、DU公钥 $PK_{du}$ 以及关键字 $W_i$ , 输出一个加密的数据库 $E_D$ 。

4) 陷门生成算法  $\text{Trapdoor}(\text{para}, SK_{du}, W_i)$ : 该算法由DU执行, DU输入系统参数 $\text{para}$ 、自己的私钥 $SK_{du}$ 以及关键字 $W_i$ , 生成关键字陷门 $T_{W_i}$ 。

5) 搜索阶段算法  $\text{Search}(\text{para}, T_{W_i}, E_D)$ : 该算法由SC执行, SC输入系统参数 $\text{para}$ 、关键字陷门 $T_{W_i}$ 以及加密的数据库 $E_D$ , 输出加密的文件索引集 $EI$ 。

6) 解密阶段算法  $\text{Decrypt}(\text{para}, SK_{du}, W_i, EI)$ : 该算法由DU执行, DU输入系统参数 $\text{para}$ 、自己的私钥 $SK_{du}$ 、关键字 $W_i$ 以及加密的文件索引集 $EI$ , 输出解密的文件索引集。

本文方案主要侧重于通过文件索引的构建来实现前向与后向安全性, 因此, 对于文件的加密与解密没有进行详细的描述。

#### 3.1 系统建立算法

系统建立算法表示为  $\text{Setup}(\lambda) \rightarrow \text{para}$ , 具体描述如下:

TA输入一个系统安全参数 $\lambda$ , 并建立两个循环群 $G_1, G_2$ , 其阶均为素数 $q$ 。定义双线性映射 $e: G_1 \times G_1 \rightarrow G_2$ ,  $P$ 为群 $G_1$ 的生成元。选取9个安全的哈希函数:  $H_1: (0, 1)^* \rightarrow G_1$ ;  $H_2: G_1 \times (0, 1)^* \rightarrow \{0, 1\}^{2\lambda+1}$ ;  $H_3, H_4: G_2 \rightarrow \{0, 1\}^{2\lambda}$ ;  $H_5: (0, 1)^* \rightarrow \mathbb{Z}_q^*$ ;  $h_1: \{0, 1\}^\lambda \rightarrow \{0, 1\}^{2\lambda}$ ;  $h_2:$

$\{0, 1\}^\lambda \rightarrow \{0, 1\}^{\lambda + \text{lb } N_{\max}}$ ;  $h_3: \{0, 1\}^\lambda \times \{0, 1\}^{\lambda + \text{lb } N_{\max}} \rightarrow \{0, 1\}^{2\lambda}$ ;  $h_4: \{0, 1\}^\lambda \times \{0, 1\}^{\lambda + \text{lb } N_{\max}} \rightarrow \{0, 1\}^{\lambda + \text{lb } N_{\max} + 1}$ 。选取一个伪随机置换函数:  $F: \{0, 1\}^\lambda \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$ , 并且 $F^{-1}$ 是函数 $F$ 的逆运算。选取两个键值函数:  $G[\text{key}] = \text{value}$ ,  $T[\text{key}] = \text{value}$ , 其中,  $G[\text{key}] = \text{value}$ 用于存储关键字的状态,  $T[\text{key}] = \text{value}$ 在数据属主更新数据库时使用。随后, TA公开系统参数:  $\text{para} = (\lambda, q, G_1, G_2, P, e, H_1, H_2, H_3, H_4, H_5, h_1, h_2, h_3, h_4, F, F^{-1})$ 。

#### 3.2 密钥生成算法

密钥生成算法表示为  $\text{KeyGen}(\text{para}) \rightarrow (PK_{du}, SK_{du})$ , 具体描述如下:

输入系统参数 $\text{para}$ , TA随机选择 $s \leftarrow \mathbb{Z}_q^*$ 作为其系统主私钥, 计算 $PK = s \cdot P$ 作为系统主公钥。随机选择 $a, b_1, b_2 \leftarrow \mathbb{Z}_q^*$ , 计算 $SK_1 = H_5(b_1 || PK)$ ,  $SK_2 = H_5(b_2 || PK)$ , 计算 $SK_{du} = a(SK_1 + SK_2)$ 作为DU的私钥, 计算 $PK_{du} = SK_{du} \cdot P$ 作为DU的公钥。

#### 3.3 数据更新阶段算法

数据更新阶段算法表示为  $\text{Update}(\text{para}, D, PK_{du}, W_i) \rightarrow E_D$ , 具体描述如下:

DO执行此算法, 输入系统参数 $\text{para}$ 、DU的公钥 $PK_{du}$ 以及数据库 $D$ 。其中:  $D: \{O, FC, W\}$ ;  $O: \{\text{add}, \text{del}\}$ 表示加入和删除操作;  $FC: \{FC_1, FC_2, \dots, FC_j, \dots\}$ 表示文件集合;  $W: \{W_1, W_2, \dots, W_i, \dots\}$ 表示文件包含的关键字集合;  $D(W_i): \{FC_1^{W_i}, FC_2^{W_i}, \dots, FC_j^{W_i}\} \subset FC$ 表示所有包含关键字 $W_i$ 的文件索引的集合。随后, DO通过以下步骤来生成一个加密的数据库:

1) DO随机选取随机数 $r, u \leftarrow \mathbb{Z}_q^*$ , 计算当前的数据库状态 $C_{DV} = r \cdot P$ , 并将其公布。

2) 对于每个关键字 $W_i \in W$ :

(1) 从关键字状态集合 $G[W_i]$ 中检索 $(st_c^{W_i}, c)$ 是否存在, 如果 $(st_c^{W_i}, c) = \perp$ , 令 $st_0^{W_i} \leftarrow \{0, 1\}^\lambda$ ,  $c \leftarrow 0$ ; 否则随机选取 $s_{c+1}^{W_i} \leftarrow \{0, 1\}^\lambda$ , 计算 $st_{c+1}^{W_i} = F(s_{c+1}^{W_i}, st_c^{W_i})$ , 最后更新关键字状态集合。

(2) 关键字状态密文算法: 计算 $K_{st_{c+1}^{W_i}} = h_1(st_{c+1}^{W_i})$ ,  $V_{st_{c+1}^{W_i}} = (PK_{du} || u || s_{c+1}^{W_i}) \oplus h_2(st_{c+1}^{W_i})$ , 其中,  $PK_{du}$ 为DU的公钥。因此,  $\langle K_{st_{c+1}^{W_i}}, V_{st_{c+1}^{W_i}} \rangle$ 为关键字 $W_i$ 的状态密文。

(3) 文件索引密文算法: 对于包含关键字 $W_i$ 的第 $j$ 个文件的索引 $FC_j^{W_i}$ , 首先计算其加密索引 $EI_{FC_j^{W_i}} = H_1(rPK_{du}, W_i || PK_{du} || u || s_{c+1}^{W_i}) \oplus (O || FC_j^{W_i})$ , 然后计算 $K_{FC_j^{W_i}} = h_3(st_{c+1}^{W_i}, PK_{du} || u || s_{c+1}^{W_i})$ ,  $V_{FC_j^{W_i}} = (PK_{du} || u || s_{c+1}^{W_i} || EI_{FC_j^{W_i}}) \oplus h_4(st_{c+1}^{W_i}, PK_{du} || u || s_{c+1}^{W_i})$ , 则 $\langle K_{FC_j^{W_i}}, V_{FC_j^{W_i}} \rangle$ 为索引 $FC_j^{W_i}$ 的文件索引密文。

(4) 关键字授权密文算法: 首先计算 $CE_{W_i} = e(H_0(W_i), rPK_{du})$ , 然后计算 $K_{CE_{W_i}} = H_2(CE_{W_i})$ ,  $V_{CE_{W_i}} = H_3(CE_{W_i}) \oplus st_{c+1}^{W_i}$ , 则 $\langle K_{CE_{W_i}}, V_{CE_{W_i}} \rangle$ 为关键字 $W_i$ 的授权密文。

3) 将加密后的数据库  $E_D = \{ \langle K_{st_{c+1}^{w_i}}, V_{st_{c+1}^{w_i}} \rangle, \langle K_{FC_j^{w_i}}, V_{FC_j^{w_i}} \rangle, \langle K_{CE_{w_i}}, V_{CE_{w_i}} \rangle \}$  存储到智能合约, 智能合约采用 Key-value 的存储模式:  $T[K_{CE_{w_i}}] = V_{CE_{w_i}}, T[K_{FC_j^{w_i}}] = V_{FC_j^{w_i}}, T[K_{st_{c+1}^{w_i}}] = V_{st_{c+1}^{w_i}}$ 。

### 3.4 陷门生成算法

陷门生成算法表示为  $\text{Trapdoor}(\text{para}, \text{SK}_{\text{du}}, W_i) \rightarrow T_{W_i}$ , 具体描述如下:

该算法由 DU 执行, 当输入  $\text{para}, \text{SK}_{\text{du}}, W_i$  时, DU 通过计算  $T_{W_i} = e(H_0(W_i)^{\text{SK}_{\text{du}}}, C_{\text{DV}}) = e(H_0(W_i), r\text{SK}_{\text{du}}P)$  生成关键字陷门  $T_{W_i}$ , 并将其上传到 SC。

### 3.5 搜索阶段算法

搜索阶段算法表示为  $\text{Search}(\text{para}, T_{W_i}, E_D) \rightarrow \text{EI}$ , 具体描述如下:

该算法由 SC 执行, 当 SC 接收到 DU 上传的关键字陷门后, 输入系统参数  $\text{para}$ 、关键字陷门  $T_{W_i}$  以及加密的数据库  $E_D$ , 进行以下计算获得加密的文件索引集。

1) 计算:  $K_{CE_{w_i}} = H_2(T_{W_i}) = H_2(e(H_0(W_i), r\text{SK}_{\text{du}}P)) = H_2(CE_{w_i}); V_{CE_{w_i}} = T[K_{CE_{w_i}}]; st_c^{w_i} = V_{CE_{w_i}} \oplus H_3(CE_{w_i})$ 。

2) 由于第 1) 步计算得到  $st_c^{w_i}$ , 因此计算  $K_{st_c^{w_i}} = h_1(st_c^{w_i}), V_{st_c^{w_i}} = T[K_{st_c^{w_i}}]$ ; 又因为已知  $V_{st_c^{w_i}}$ , 所以计算  $(PK_{\text{du}} \| u \| s_c^{w_i}) = V_{st_c^{w_i}} \oplus h_2(st_c^{w_i})$ ; 再计算  $K_{FC_j^{w_i}} = h_3(st_c^{w_i}, PK_{\text{du}} \| u \| s_c^{w_i}), T[K_{FC_j^{w_i}}] = V_{FC_j^{w_i}}$ , 由此, 可计算出  $(PK_{\text{du}} \| u \| s_c^{w_i} \| \text{EI}_{FC_j^{w_i}}) = V_{FC_j^{w_i}} \oplus h_4(st_c^{w_i}, PK_{\text{du}} \| u \| s_c^{w_i})$ 。又因为  $PK_{\text{du}} \| u \| s_c^{w_i}$  已由计算获得, 所以可以得到加密的文件索引集  $\text{EI}_{FC_j^{w_i}}$ , SC 将加密的文件索引集返回给 DU。

3) 输入当前状态  $st_{c-1}^{w_i} = F^{-1}(s_c^{w_i}, st_c^{w_i})$ , 并重复执行上述操作, 获得该状态下的加密文件索引集。

### 3.6 解密阶段算法

解密阶段算法表示为  $\text{Decrypt}(\text{para}, \text{SK}_{\text{du}}, W_i, \text{EI}) \rightarrow \text{RI}$ , 具体描述如下:

该算法由 DU 执行, 当输入  $\text{para}, \text{SK}_{\text{du}}, W_i, \text{EI}$  时, DU 通过计算  $(O \| FC_j^{w_i}) = H_1(C_{\text{DV}} \cdot \text{SK}_{\text{du}}, W_i \| PK_{\text{du}} \| u \| s_{c+1}^{w_i}) \oplus \text{EI}_{FC_j^{w_i}}$  来获得文件索引  $(O \| FC_j^{w_i})$ , 若  $O = \text{del}$ , 则意味着该文件索引已经被删除或者不存在。最后, DU 根据文件索引从 CS 获得相应的加密文件。

数据更新和搜索算法示意图如图 2 所示。

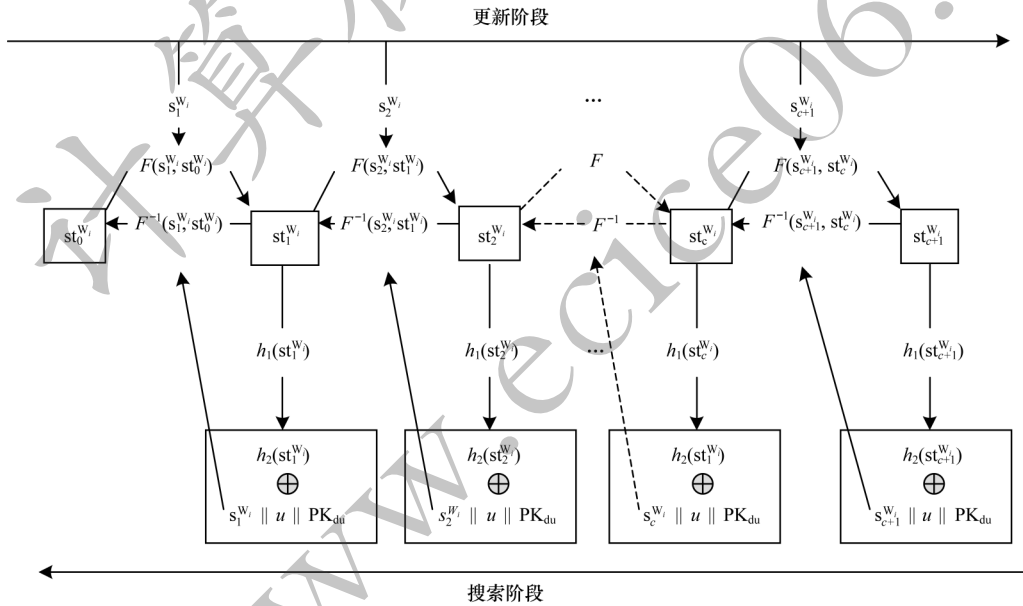


图2 更新与搜索算法示意图

Fig.2 Schematic diagram of update and search algorithms

### 3.7 智能合约设计

算法 2 详细地说明了智能合约的设计流程, 智能合约包括插入和搜索两个阶段。在插入阶段, DO 将加密的数据库  $E_D$  上传到智能合约。在搜索阶段, DU 通过发送关键字陷门来调用搜索合约完成搜索。尽管理论上智能合约可以执行任何公钥操作, 例如双线性对运算等, 但操作以及计算的复杂性会使其实现以及应用带来严重障碍。在本文方案中, 智能合约只执行一些简单的哈希操作, 计算以及通信开

销极低, 因此可以很好地保证实用性。

#### 算法 1 数据属主执行算法

1.  $\text{Update}(\text{para}, D, \text{PK}_{\text{du}}, W_i) \rightarrow E_D$
2.  $r, u \xleftarrow{R} \mathbb{Z}_q^*$
3. Publish the status of the database  $C_{\text{DV}} = r \cdot P$
4. for each keyword  $W_i \in W$  do
5.  $(st_c^{w_i}, c) \leftarrow G[W_i]$
6. if  $G[W_i] = \perp$  then
7.  $st_0^{w_i} \xleftarrow{R} \{0, 1\}^\lambda, c \leftarrow 0$

```

8.end if
9. $s_{c+1}^{w_i} \xleftarrow{R} \{0, 1\}^\lambda$ 
10. $st_{c+1}^{w_i} = F(s_{c+1}^{w_i}, st_c^{w_i})$ 
11. $G[W_i] = (st_{c+1}^{w_i}, c+1)$ 
12. $K_{st_{c+1}^{w_i}} = h_1(st_{c+1}^{w_i})$ 
13. $V_{st_{c+1}^{w_i}} = (PK_{du} || u || s_{c+1}^{w_i}) \oplus h_2(st_{c+1}^{w_i})$ 
14.for each index  $FC_j^{w_i}$  do
15.  $EI_{FC_j^{w_i}} = H_1(rPK_{du}, W_i || PK_{du} || u || s_{c+1}^{w_i}) \oplus (O || FC_j^{w_i})$ 
16. $K_{FC_j^{w_i}} = h_3(st_{c+1}^{w_i}, PK_{du} || u || s_{c+1}^{w_i})$ 
17. $V_{FC_j^{w_i}} = (PK_{du} || u || s_{c+1}^{w_i} || EI_{FC_j^{w_i}}) \oplus h_4(st_{c+1}^{w_i}, PK_{du} || u || s_{c+1}^{w_i})$ 
18.end for
19. $CE_{w_i} = e(H_0(W_i), rPK_{du})$ 
20. $K_{CE_{w_i}} = H_2(CE_{w_i})$ 
21. $V_{CE_{w_i}} = H_3(CE_{w_i}) \oplus st_c^{w_i}$ 
22. $E_D = \{ \langle K_{st_{c+1}^{w_i}}, V_{st_{c+1}^{w_i}} \rangle, \langle K_{FC_j^{w_i}}, V_{FC_j^{w_i}} \rangle, \langle K_{CE_{w_i}}, V_{CE_{w_i}} \rangle \}$ 
23.end for
//Smart contract
24.for each ciphertext of  $W_i \in W$  do
25.  $T[K_{st_{c+1}^{w_i}}] = V_{st_{c+1}^{w_i}}$ 
26.  $T[K_{FC_j^{w_i}}] = V_{FC_j^{w_i}}$ 
27.  $T[K_{CE_{w_i}}] = V_{CE_{w_i}}$ 
28.end for

```

## 算法2 智能合约执行算法

```

1.Search(para,  $T_{w_i}, E_D$ )  $\rightarrow$  EI
2.Initialize an empty set EI
3.Compute  $K_{CE_{w_i}} = H_2(T_{w_i})$ 
4.if  $T[K_{CE_{w_i}}] = \perp$  then
5.return EI
6.else
7. $V_{CE_{w_i}} = T[K_{CE_{w_i}}]$ 
8. $st_c^{w_i} = V_{CE_{w_i}} \oplus H_3(CE_{w_i})$ 
9.end if
10. $K_{st_c^{w_i}} = h_1(st_c^{w_i})$ 
11.while  $T[K_{st_c^{w_i}}] \neq \perp$  do
12.  $V_{st_c^{w_i}} = T[K_{st_c^{w_i}}]$ 
13.  $(PK_{du} || u || s_c^{w_i}) = V_{st_c^{w_i}} \oplus h_2(st_c^{w_i})$ 
14.  $K_{FC_j^{w_i}} = h_3(st_c^{w_i}, PK_{du} || u || s_c^{w_i})$ 
15.  $T[K_{FC_j^{w_i}}] = V_{FC_j^{w_i}}$ 
16.  $(PK_{du} || u || s_c^{w_i} || EI_{FC_j^{w_i}}) = V_{FC_j^{w_i}} \oplus h_4(st_c^{w_i}, PK_{du} || u || s_c^{w_i})$ 
17.  $EI = EI \cup (EI_{FC_j^{w_i}})$ 
18.  $st_{c-1}^{w_i} = F^{-1}(s_c^{w_i}, st_c^{w_i})$ 
19.  $st_c^{w_i} = st_{c-1}^{w_i}$ 
20.end while

```

## 4 安全性分析

### 4.1 正确性分析

从数据用户获取文件索引的角度来验证本文方案的正确性。数据用户想要从智能合约处获得加密的文件索引,首先需要向其提交关键字搜索陷门:  $T_{w_i} = e(H_0(W_i)^{SK_{du}}, C_{DV}) = e(H_0(W_i), rSK_{du}P)$ , 其中包含数据用户自己的私钥  $SK_{du}$ , 因此其他用户是无法伪造该搜索陷门的。智能合约在获得陷门后, 计算  $K_{CE_{w_i}} = H_2(T_{w_i}) = H_2(e(H_0(W_i), rSK_{du}P))$ , 又因为数据属主是将授权密文通过键值函数  $T[K_{CE_{w_i}}] = V_{CE_{w_i}}$  的形式存储在智能合约上的, 因此可得  $V_{CE_{w_i}}$ , 并由此可得到:  $st_c^{w_i} = V_{CE_{w_i}} \oplus H_3(CE_{w_i})$ ,  $K_{st_c^{w_i}} = h_1(st_c^{w_i})$ ,  $V_{st_c^{w_i}} = T[K_{st_c^{w_i}}]$ ,  $(PK_{du} || u || s_c^{w_i}) = V_{st_c^{w_i}} \oplus h_2(st_c^{w_i})$ ,  $K_{FC_j^{w_i}} = h_3(st_c^{w_i}, PK_{du} || u || s_c^{w_i})$ ,  $T[K_{FC_j^{w_i}}] = V_{FC_j^{w_i}}$ ,  $(PK_{du} || u || s_c^{w_i} || EI_{FC_j^{w_i}}) = V_{FC_j^{w_i}} \oplus h_4(st_c^{w_i}, PK_{du} || u || s_c^{w_i})$ 。

智能合约将计算得到的加密索引  $EI_{FC_j^{w_i}}$  发送给数据用户, 数据用户获得加密索引  $EI_{FC_j^{w_i}}$  后, 通过计算  $(O || FC_j^{w_i}) = H_1(C_{DV} \cdot SK_{du}, W_i || PK_{du} || u || s_{c+1}^{w_i}) \oplus EI_{FC_j^{w_i}}$  来获得文件索引  $FC_j^{w_i}$ , 因为数据用户是使用自己的私钥解密的加密索引, 即使恶意用户可以获得加密索引  $EI_{FC_j^{w_i}}$ , 也无法解密获得文件索引  $FC_j^{w_i}$ , 所以只要每个实体按照要求执行, 就能保证方案是安全有效的。

### 4.2 前后向安全性分析

对前向安全性做一个简要说明, 具体的安全性证明见4.3节。在本文方案中, 当数据用户想要获得文件索引时, 需要生成一个关键字陷门  $T_{w_i}$ 。智能合约根据该陷门可以计算获得授权密文  $\langle K_{CE_{w_i}}, V_{CE_{w_i}} \rangle$ , 并据此计算出当前状态  $st_c^{w_i} = V_{CE_{w_i}} \oplus H_3(CE_{w_i})$ , 进而可以进一步计算出加密的文件索引, 但是当更新发生时, 新的状态会根据  $st_{c+1}^{w_i} = F(s_{c+1}^{w_i}, st_c^{w_i})$  计算得到, 根据伪随机置换函数的安全性, 敌手不可以通过过去的状态  $st_c^{w_i}$  来推断出新的状态  $st_{c+1}^{w_i}$ , 因此, 敌手便不可以通过  $st_c^{w_i}$  来获得  $st_{c+1}^{w_i}$  状态下的加密文件索引, 即旧的搜索陷门不能用来获取更新后的文件, 实现了前向安全。

同样对后向安全性做一个简要说明, 具体的安全性证明见4.3节。在本文方案中, 智能合约中存放的是使用数据用户公钥加密的加密索引  $EI_{FC_j^{w_i}}$  且  $EI_{FC_j^{w_i}} = H_1(rPK_{du}, W_i || PK_{du} || u || s_{c+1}^{w_i}) \oplus (O || FC_j^{w_i})$ 。换言之, 即使发生泄漏, 没有数据用户的私钥敌手, 依然无法从加密索引中获得有关文件索引的任何信息, 后续搜索也就不会泄露已删除文件所对应的索引信息, 实现了后向安全。

### 4.3 安全性证明

**定义1** 如果  $F$  是一个安全的伪随机置换函数, 哈希函数是抗冲突的散列函数, 且 DBDH 困难问题



是成立的,那么本文提出的方案是满足自适应安全的公钥可搜索加密方案。

**定义2** 对于一个动态更新的公钥可搜索加密方案,如果其泄露函数  $L=(L^{\text{Setup}}, L^{\text{Search}}, L^{\text{Update}})$  可以定义为:  $L^{\text{Setup}}=\perp, L^{\text{Update}}(\text{O}, W_i, \text{FC}_j)=L^{\text{Update}}(\text{O}, \text{FC}_j), L^{\text{Search}}(W_i)=L^{\text{Search}}(\text{TimeD}(W_i), \text{DleHist}(W_i))$ , 那么该公钥可搜索加密方案满足前向与后向安全性。

**定理** 令  $F$  是一个安全的伪随机置换函数, 哈希函数是抗冲突的散列函数, 且 DBDH 困难问题是成立的。定义泄露函数  $L=(L^{\text{Setup}}, L^{\text{Search}}, L^{\text{Update}})$ :  $L^{\text{Setup}}=\perp, L^{\text{Update}}(\text{O}, W_i, \text{FC}_j)=L^{\text{Update}}(\text{O}, \text{FC}_j), L^{\text{Search}}(W_i)=L^{\text{Search}}(\text{TimeD}(W_i), \text{DleHist}(W_i))$ , 那么称方案是具有前向与后向安全性的 L-adaptively-secure 的公钥可搜索加密方案。

**证明** 首先定义一个安全的伪随机置换函数  $F$ , 运行所有的哈希函数作为随机预言机, 每个哈希函数运行模式相同, 区别在于输出长度不同。每一个随机预言机维护一个哈希链表存储输入输出对。例如, 对于  $h_1$  预言机, 输入一个字符串  $x$ , 预言机首先检查  $h_1\text{-List}$  中是否存在  $x$  对应的值, 若不存在, 随机选择一个字符串  $y=h_1(x)$  作为输入输出对  $(x, y)$  存储在哈希链表  $h_1\text{-List}$  中。令  $S$  作为一个模拟器,  $A$  作为一个敌手, 定义以下两个游戏:

1)  $\text{Real}_A^{\Pi}(\lambda)$ 。该游戏运行系统建立算法  $\text{Setup}(\lambda)$  和密钥生成算法  $\text{KeyGen}(\lambda)$  生成  $(\text{para}, \text{PK}_{\text{du}}, \text{SK}_{\text{du}}, G[\text{key}]=\text{value})$ , 敌手  $A$  选择数据库  $D$  执行更新询问, 游戏执行更新算法  $\text{Update}(\text{para}, D, \text{PK}_{\text{du}}, W_i) \rightarrow E_D$  并将运行结果返回给敌手  $A$ 。敌手  $A$  随机选取一个关键字  $W_i$ , 执行陷门生成询问, 游戏执行陷门生成算法  $\text{Trapdoor}(\text{para}, \text{SK}_{\text{du}}, W_i) \rightarrow T_{W_i}$ , 并将结果返回给敌手  $A$ 。敌手选取一个关键字陷门  $T_{W_i}$  执行搜索询问, 游戏执行搜索算法  $\text{Search}(\text{para}, T_{W_i}, E_D) \rightarrow EI$ , 并将结果返回给敌手  $A$ 。以上询问敌手  $A$  可以在多项式时间内执行多次, 最终, 敌手输出  $b \in (0, 1)$ 。

2)  $\text{Ideal}_{A,S}^{\Pi}(\lambda)$ 。该游戏由模拟器  $S$  运行, 模拟器  $S$  生成的数据均由泄露函数生成。例如, 模拟器利用  $L^{\text{Setup}}$  生成  $(\text{para}, \text{PK}_{\text{du}}) \leftarrow S(L^{\text{Setup}})$ , 利用  $L^{\text{Update}}$  生成加密的数据库  $E_D$ , 并用其回复敌手  $A$  的更新询问。利用  $L^{\text{Search}}$  生成加密的索引集  $EI$ , 并用其回复敌手  $A$  的搜索询问。以上询问敌手  $A$  可以在多项式时间内执行多次, 最终, 敌手输出  $b \in (0, 1)$ 。

本文方案的安全性通过一系列游戏来证明, 从游戏  $\text{Real}_A^{\Pi}(\lambda)$  开始执行, 并生成一系列与前一个有细微差别的游戏, 确保各个游戏之间是不可区分的, 执行到游戏  $\text{Ideal}_{A,S}^{\Pi}(\lambda)$  时游戏结束。因为每一个游戏与其前一个游戏是不可区分的, 所以根据不可区分的传递性, 可以得到游戏  $\text{Real}_A^{\Pi}(\lambda)$  与  $\text{Ideal}_{A,S}^{\Pi}(\lambda)$  是不可区分的, 从而完成证明。

1)  $\text{HybridG}_1$ 。游戏  $G_1$  除了在生成状态  $\text{st}_c$  时不使用伪随机置换函数  $F$  外, 其余过程与游戏  $\text{Real}_A^{\Pi}(\lambda)$

相同。游戏维护一个状态列表  $\text{stList}[W_i, c]=\text{st}_c$ , 在数据更新阶段, 当需要使用  $\text{st}_c$  时, 会首先从  $\text{stList}$  中检索其是否存在, 若存在, 则返回相应的元组, 否则, 随机选择一个字符串  $\text{st}_c \leftarrow (0, 1)^i$  并将其插入状态列表  $\text{stList}$  中, 而不是通过伪随机置换函数  $\text{st}_c = F(\text{sc}, \text{st}_{c-1})$  计算得到状态  $\text{st}_c$ 。因为伪随机置换函数的安全性, 所以无法区分伪随机置换函数与随机函数, 因此, 游戏  $G_1$  与游戏  $\text{Real}_A^{\Pi}(\lambda)$  是不可区分的。

2)  $\text{HybridG}_2$ 。游戏  $G_2$  与游戏  $G_1$  略有不同, 在  $G_2$  中, 在执行哈希操作时, 使用随机的字符串访问预言机来取代哈希查询。例如, 当在数据更新阶段需要访问  $h_1$  时, 不计算  $K_{\text{FC}_{c+1}}^{\text{st}_{c+1}} = h_1(\text{st}_{c+1}^{W_i})$ , 而是执行以下操作: 随机选择  $K_{\text{FC}_{c+1}}^{\text{st}_{c+1}} \leftarrow (0, 1)^i, L_1[\text{st}_{c+1}^{W_i}] \leftarrow K_{\text{FC}_{c+1}}^{\text{st}_{c+1}}$ , 其中,  $L_i[]$  是游戏  $G_2$  维护的一个链表。然后在搜索阶段, 当需要访问  $h_1$  时, 进行操作  $h_1\text{-List}[\text{st}_{c+1}^{W_i}] \leftarrow L_1[\text{st}_{c+1}^{W_i}]$ , 其中,  $h_1\text{-List}[]$  是哈希函数  $h_1$  对应的哈希链表。同样的操作适用于  $h_3$ , 当在数据更新阶段需要访问  $h_3$  时, 不计算  $K_{\text{FC}_{c+1}}^{\text{st}_{c+1}} = h_3(\text{st}_{c+1}^{W_i}, \text{PK}_{\text{du}} \| u \| \text{st}_{c+1}^{W_i})$ , 而是执行以下操作: 随机选择  $K_{\text{FC}_{c+1}}^{\text{st}_{c+1}} \leftarrow (0, 1)^i, L_3[\text{st}_{c+1}^{W_i}, \text{PK}_{\text{du}} \| u \| \text{st}_{c+1}^{W_i}] \leftarrow K_{\text{FC}_{c+1}}^{\text{st}_{c+1}}$ , 其中,  $L_i[]$  是游戏  $G_2$  维护的一个链表。在搜索阶段, 当需要访问  $h_3$  时, 进行以下操作:  $h_3\text{-List}[\text{st}_{c+1}^{W_i}, \text{PK}_{\text{du}} \| u \| \text{st}_{c+1}^{W_i}] \leftarrow L_3[\text{st}_{c+1}^{W_i}, \text{PK}_{\text{du}} \| u \| \text{st}_{c+1}^{W_i}]$ , 其中,  $h_3\text{-List}[]$  是哈希函数  $h_3$  对应的哈希链表。采用同样的操作完成所有哈希函数的替换, 此处不再赘述。操作完成后, 可以明显得知游戏  $G_2$  与游戏  $G_1$  是不可区分的, 因为哈希函数是抗碰撞攻击的,  $|\Pr[G_2=1] - \Pr[G_1=1]|$ 。

3)  $\text{HybridG}_3$ 。游戏  $G_3$  与游戏  $G_2$  略有不同, 在  $G_3$  中, 当生成授权密文时, 不通过计算  $\text{CE}_{W_i} = e(H_0(W_i), r\text{PK}_{\text{du}})$ , 而是执行以下操作: 随机选择  $r\text{CE}_{W_i} \leftarrow G_2, \text{CEList}[W_i, C_{\text{DV}}, r\text{CE}_{W_i}] \leftarrow r\text{CE}_{W_i}$ , 其中,  $r$  是计算当前数据库状态  $C_{\text{DV}}$  时生成的随机数,  $\text{CEList}[W_i, C_{\text{DV}}, r\text{CE}_{W_i}]$  是游戏维护的一个链表用来回复敌手  $A$  进行的陷门询问。可以看出, 元组  $(P, rP, \text{PK}_{\text{du}}, H_0(W_i), \text{CE}_{W_i})$  是一个满足 DBDH 困难问题的元组, 而元组  $(P, rP, \text{PK}_{\text{du}}, H_0(W_i), r\text{CE}_{W_i})$  是一个随机生成的元组, 因此, 敌手若想区分游戏  $G_3$  与  $G_2$ , 必须解决 DBDH 困难问题, 所以有:  $|\Pr[G_3=1] - \Pr[G_2=1]| \leq \text{Adv}_A^{\text{DBDH}}(\lambda)$ 。又因为在多项式时间内解决困难问题 DBDH 的概率  $\text{Adv}_A^{\text{DBDH}}(\lambda)$  是可以忽略不计的, 所以游戏  $G_3$  与  $G_2$  是不可区分的。

4)  $\text{HybridG}_4$ 。在最后一个游戏  $G_4$  中, 模拟器  $S$  通过使用泄露函数生成一个视图, 而不是通过真实的更新和查询。泄露函数包括搜索模式和更新历史, 搜索模式显示过去哪些查询是关于关键字  $W_i$  的查询, 更新历史会显示过去哪些查询是关于关键字  $W_i$  的更新查询, 以及更新的类型和文档标识符。在敌手看来, 游戏  $G_4$  与游戏  $G_3$  的表现是完全一致的, 它们是不可区分的:  $\Pr[G_4=1] = \Pr[G_3=1] = \text{Ideal}_{A,S}^{\Pi}(\lambda)$ , 所以有  $|\Pr[\text{Real}_A^{\Pi}(\lambda)=1] - \Pr[\text{Ideal}_{A,S}^{\Pi}(\lambda)=1]| \leq \text{Adv}_A^{\text{DBDH}}(\lambda)$ 。

又因为在多项式时间内解决困难问题 DBDH 的概率  $\text{Adv}_{\Lambda}^{\text{DBDH}}(\lambda)$  是可以忽略不计的,所以游戏  $\text{Real}_{\Lambda}^{\Pi}(\lambda)$  与  $\text{Ideal}_{\Lambda,S}^{\Pi}(\lambda)$  是不可区分的。因此,可证明本文提出的方案是具有前向与后向安全性的 L-adaptively-secure 的公钥可搜索加密方案。

5 性能分析

将本文方案与文献[5,14,21]方案进行性能对比,其中符号说明如表2所示,比较结果如表3所示。

在本文方案中,更新阶段主要分为3个部分,分别为生成状态密文、生成索引密文以及生成授权密文。在生成状态密文时,方案需要维护一个映射以及运算伪随机置换函数和哈希函数,该阶段的计算开销为  $T_{\text{map}} + F + 2T_h$ ,同理可计算出在生成索引密文时的计算开销为  $N \times 2T_h + T_{\text{sm}}$ ,生成授权密文的计算开销为  $T_e + 2T_h + T_{\text{sm}}$ ,因此,整个更新阶段的计算开销为  $T_{\text{map}} + F + 4T_h + T_e + N \times 2T_h + 2T_{\text{sm}}$ 。同理可得,文献[5]方案加密阶段的计算开销为  $T_{\text{map}} + T_e + N \times (T_h + T_{\text{Mul}_{G_2}}) + 2T_{\text{sm}}$ ,文献[14]方案加密阶段的计算开销为  $T_h + N \times (2T_h + T)$ ,文献[21]方案加密阶段的计算开销为  $N \times F + N(T_e + 2T_{\text{sm}} + 2T_{H_{G_2}})$ 。在陷门生成阶段,本

文方案的计算开销为  $T_{\text{map}} + T_{\text{sm}} + T_e$ ,文献[5,14,21]方案的计算开销分别为  $T_{\text{map}} + T_{\text{sm}} + T_h + T_{\text{sm}}$ 。在搜索阶段,本文方案避免了诸如双线性映射等复杂计算操作,具有较高的搜索效率,计算开销为  $N \times 2T_h + 4T_h$ ,文献[5,14,21]方案的计算开销分别为  $T_e + (N+1)h + N \times T_{\text{Div}_{G_2}} + N \times (2T_h + T^{-1}) + N \times F^{-1} + T_e + T_{H_{G_2}}$ 。

表2 性能对比中的符号说明

Table 2 Symbol description in performance comparison

符号	符号说明
$T_{\text{sm}}$	群 $G_1$ 中标量乘法计算时间
$T_e$	双线性对计算时间
$T_{\text{map}}$	维护映射计算时间
$T_{H_{G_2}}$	群 $G_2$ 中哈希运算计算时间
$T_h$	哈希运算计算时间
$T/T^{-1}$	函数 $T/T^{-1}$ 计算时间
$F/F^{-1}$	函数 $F/F^{-1}$ 计算时间
$T_{\text{Mul}_{G_2}}$	群 $G_2$ 中乘法运算计算时间
$T_{\text{Div}_{G_2}}$	群 $G_2$ 中除法运算计算时间
$N$	加密索引的个数

表3 各方案的计算开销和安全性对比

Table 3 Comparison of computing overhead and security of each scheme

方案	计算开销			安全性		
	加密(更新)	陷门生成	搜索阶段	前向	后向	密码体制
文献[5]方案	$T_{\text{map}} + T_e + N \times (T_h + T_{\text{Mul}_{G_2}}) + 2T_{\text{sm}}$	$T_{\text{map}} + T_{\text{sm}}$	$T_e + (N+1)h + N \times T_{\text{Div}_{G_2}}$	×	×	公钥
文献[14]方案	$T_h + N \times (2T_h + T)$	$T_h$	$N \times (2T_h + T^{-1})$	√	×	对称
文献[21]方案	$N \times F + N(T_e + 2T_{\text{sm}} + 2T_{H_{G_2}})$	$T_{\text{sm}}$	$N \times F^{-1} + T_e + T_{H_{G_2}}$	√	×	公钥
FBSEBS 方案	$T_{\text{map}} + F + 4T_h + T_e + N \times 2T_h + 2T_{\text{sm}}$	$T_{\text{map}} + T_{\text{sm}} + T_e$	$N \times 2T_h + 4T_h$	√	√	公钥

在安全性方面:文献[5]方案虽然基于公钥密码体制提出了一个轻量化的可搜索加密方案,但该方案的安全性明显不足,不满足前后向安全性,无法很好地抵抗文件注入攻击;文献[14]提出的对称可搜索加密方案满足前向安全性,且在陷门生成阶段具有良好的效率,但是由于该方案基于对称密码体制,因此不适用于现实生活中大多数的应用环境,如车联网环境等;文献[21]基于公钥密码体制提出了一个具有前向安全性的可搜索加密方案,解决了文件注入攻击问题;本文方案在搜索效率以及后向安全性上,相较于文献[21]方案均有一定的优势。

为更清楚地对比各方案的计算效率,在配备 Intel Core i5-7500 处理器、3.0 GHz 主频,以及 8 GB 的内存环境下进行模拟实验,将本文方案同文献[5,14,21]方案进行计算效率对比,结果如图3~图5所示,结果表明,本文方案在加密算法阶段效率要高于文献[5]提出的轻量化公钥可搜索加密方案以及文献[14]提出的具有前向安全的对称可搜索加密方案,与文献[21]方案基本相当。在陷门生成阶段,本文方案的计算效率要略差与其他方案。在搜索算法阶段,由于本文方案避免了大量的双线性对计算操作,因此计算效率远高于其他方案,且随着

索引数量的不断增加,其在搜索阶段的计算效率优势会更加明显。综上所述,本文提出的公钥可搜索加密方案在加密阶段以及搜索阶段的计算效率表现是比较有优势的,虽然在陷门生成算法阶段的计算效率略低于其他方案,但作为安全性的折中,这是可以接受的。此外,由于本文方案在搜索算法阶段避免了大量的双线性对运算操作,因此搜索效率要远高于其他方案,这也使得本文方案更适用于大数据通信环境。

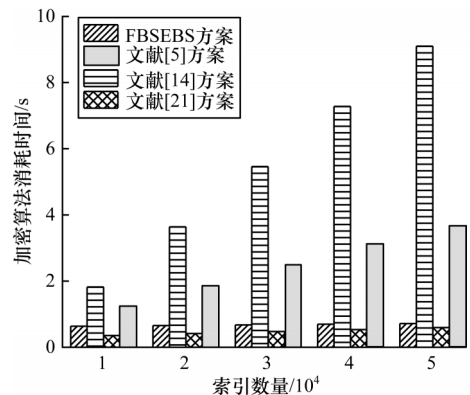


图3 更新(加密)算法消耗时间

Fig.3 Time consuming of updating(encryption) algorithm



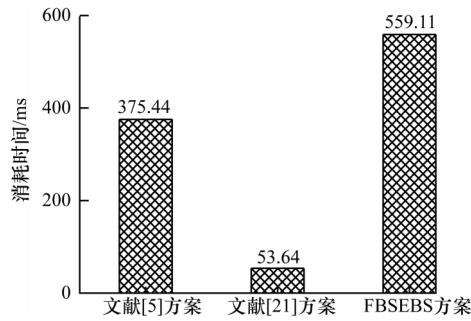


图4 陷门生成算法消耗时间

Fig.4 Time consuming of trapdoor generation algorithm

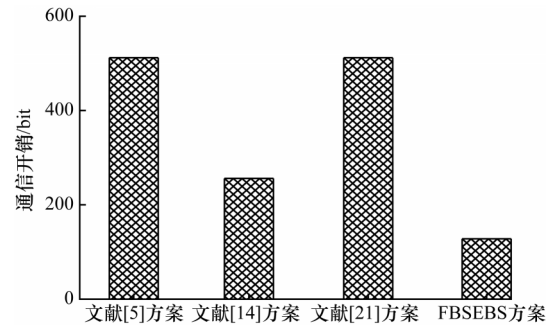


图7 陷门生成阶段通信开销对比

Fig.7 Comparison of communication overhead during trapdoor generation

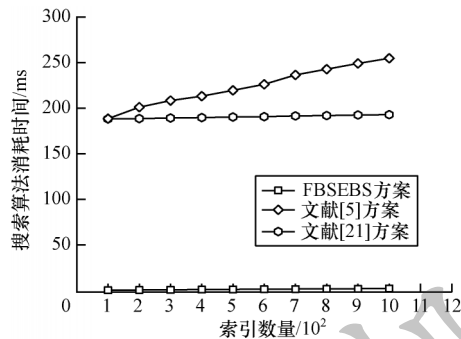


图5 搜索算法消耗时间

Fig.5 Time consuming of search algorithm

设置安全参数为 $\lambda=128$  bit且 $|G_1|=512$  bit,  $|G_2|=1536$  bit,通过比较索引密文生成阶段以及陷门生成阶段的通信开销大小来衡量本文方案在通信开销上面的表现,比较结果如表3所示。由表3可知,本文方案在计算索引密文时,主要的计算开销为 $N \times 2T_h + 4T_h$ ,因此,其主要通信开销为 $2N\lambda + 4\lambda$ ,其中 $N$ 为加密索引的个数。同理可知,文献[5,14,21]方案的通信开销分别为 $2N\lambda$ 、 $N(\lambda + |G_2|)$ 、 $N(\lambda + |G_1| + |G_2|)$ 。此外,本文生成一个搜索陷门的最小通信开销为 $\lambda$ ,相对应的文献[5,14,21]方案的通信开销为 $2\lambda$ 、 $|G_1|$ 、 $|G_1|$ ,如图6、图7所示。

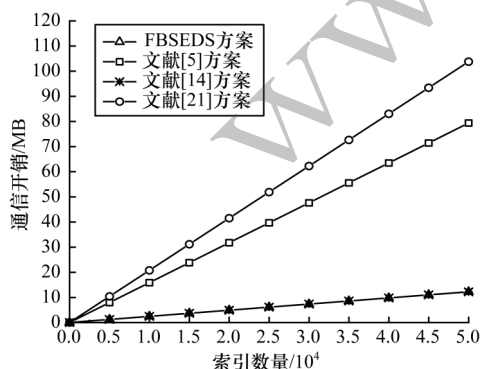


图6 索引密文生成阶段通信开销对比

Fig.6 Comparison of communication overhead during index ciphertext generation

## 6 结束语

在云计算环境下,采用可搜索加密技术能够在保证数据机密性的同时实现高效的密文检索,但当前多数动态可搜索加密方案因不满足前后向安全性导致难以抵抗泄露滥用攻击以及文件注入攻击。本文设计一个新的动态公钥可搜索加密方案,其在智能合约的辅助下满足前后向安全性,可抵抗内部关键字猜测攻击,并避免使用大量的双线性对运算操作,解决了现有公钥可搜索加密方案计算开销较大的问题。分析结果表明,本文方案在计算开销以及安全性上的表现优于文献[5,14,21]方案。然而,本文构造的动态可搜索加密方案仅支持单关键字搜索,下一步将构造一个支持多关键字搜索,同时满足前后向安全性的动态可搜索加密方案。

## 参考文献

- [1] NAMASUDRA S. Data access control in the cloud computing environment for bioinformatics[J]. International Journal of Applied Research in Bioinformatics, 2021, 11(1):40-50.
- [2] MOHAMMED SADEEQ M, ABDULKAREEM N M, ZEEBAREE S R M, et al. IoT and cloud computing issues, challenges and opportunities: a review [J]. Qubahan Academic Journal, 2021, 1(2): 1-7.
- [3] MANSOURI N, GHAFARI R, ZADE B M H. Cloud computing simulators: a comprehensive review [J]. Simulation Modelling Practice and Theory, 2020, 104: 1-10.
- [4] SONG D X, WAGNER D, PERRIG A. Practical techniques for searches on encrypted data[C]//Proceedings of 2000 IEEE Symposium on Security and Privacy. Washington D. C. , USA: IEEE Press, 2000: 44-55.
- [5] XU P, HE S H, WANG W, et al. Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks[J]. IEEE Transactions on Industrial Informatics, 2018, 14(8): 3712-3723.
- [6] STEFANOV E, PAPAMANTHOU C, SHI E. Practical dynamic searchable encryption with small leakage[C]//Proceedings of 2014 Network and Distributed System Security Symposium. Reston, USA: Internet Society, 2014: 72-75.
- [7] CASH D, JARECKI S, JUTLA C, et al. Highly-scalable

- searchable symmetric encryption with support for Boolean queries[C]//Proceedings of CRYPTO'13. Berlin, Germany: Springer, 2013:353-373.
- [ 8 ] HU S S, CAI C J, WANG Q, et al. Searching an encrypted cloud meets blockchain: a decentralized, reliable and fair realization[C]//Proceedings of 2018 IEEE Conference on Computer Communications. Washinton D. C. , USA: IEEE Press, 2018:792-800.
- [ 9 ] 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, USA: ACM Press, 2015:668-679.
- [ 10 ] ISLAM M S, KUZU M, KANTARCIOGLU M. Access pattern disclosure on searchable encryption: ramification, attack and mitigation [EB/OL]. [ 2021-08-10 ]. <https://personal.utdallas.edu/~mxk055100/publications/ndss2012.pdf>.
- [ 11 ] ZHANG Y P, KATZ J, PAPAMANTHOU C. All your queries are belong to us: the power of file-injection attacks on searchable encryption [C]//Proceedings of the 25th USENIX Conference on Security Symposium. [ S. l. ]: USENIX, 2016:707-720.
- [ 12 ] CHAMANI J G, PAPADOPOULOS D, PAPAMANTHOU C, et al. New constructions for forward and backward private symmetric searchable encryption [C]//Proceedings of 2018 ACM SIGSAC Conference on Computer and Communications Security. New York, USA: ACM Press, 2018:1038-1055.
- [ 13 ] SONG X F, DONG C Y, YUAN D D, et al. Forward private searchable symmetric encryption with optimized I/O efficiency [J]. IEEE Transactions on Dependable and Secure Computing, 2020, 17(5):912-927.
- [ 14 ] BOST R. Σσpos; forward secure searchable encryption [C]//Proceedings of 2016 ACM SIGSAC Conference on Computer and Communications Security. New York, USA: ACM Press, 2016:1143-1154.
- [ 15 ] WEI Y, LU S Y, GUO X J, et al. FSSE: forward secure searchable encryption with keyed-block chains [J]. Information Sciences, 2019, 500:113-126.
- [ 16 ] BOST R, MINAUD B, OHRIMENKO O. Forward and backward private searchable encryption from constrained cryptographic primitives[C]//Proceedings of 2017 ACM SIGSAC Conference on Computer and Communications Security. New York, USA: ACM Press, 2017:1465-1482.
- [ 17 ] SUN S F, YUAN X L, LIU J K, et al. Practical backward-secure searchable encryption from symmetric puncturable encryption [C]//Proceedings of 2018 ACM SIGSAC Conference on Computer and Communications Security. New York, USA: ACM Press, 2018:763-780.
- [ 18 ] BONEH D, DI CRESCENZO G, OSTROVSKY R, et al. Public key encryption with keyword search [C]//Proceedings of EUROCRYPT' 04. Berlin, Germany: Springer, 2004:506-522.
- [ 19 ] ZHANG Y L, LIU X Z, LANG X L, et al. VCLPKES: verifiable certificateless public key searchable encryption scheme for industrial Internet of things[J]. IEEE Access, 2020, 8:20849-20861.
- [ 20 ] YANG N B, XU S M, QUAN Z. An efficient public key searchable encryption scheme for mobile smart terminal[J]. IEEE Access, 2020, 8:77940-77950.
- [ 21 ] CHEN B W, WU L B, KUMAR N, et al. Lightweight searchable public-key encryption with forward privacy over IIoT outsourced data[J]. IEEE Transactions on Emerging Topics in Computing, 2021, 9(4):1753-1764.
- [ 22 ] CHEN B W, WU L B, WANG H Q, et al. A blockchain-based searchable public-key encryption with forward and backward privacy for cloud-assisted vehicular social networks[J]. IEEE Transactions on Vehicular Technology, 2020, 69(6):5813-5825.
- [ 23 ] BONEH D, WATERS B. Constrained pseudorandom functions and their applications [C]//Proceedings of International Conference on the Theory and Application of Cryptology and Information Security. Berlin, Germany: Springer, 2013:280-300.

编辑 金胡考