计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2020.20200439
57(10): 2104-2116, 2020

一种增强的多用户前向安全动态对称可搜索加密方案

卢冰洁 周 俊 曹珍富
(上海市高可信计算重点实验室(华东师范大学) 上海 200062)
(xiaoyao9697@163.com)

A Multi-User Forward Secure Dynamic Symmetric Searchable Encryption with Enhanced Security

Lu Bingjie, Zhou Jun, and Cao Zhenfu
(*Shanghai Key Laboratory of Trustworthy Computing* (*East China Normal University*), *Shanghai* 200062)

Abstract Dynamic symmetric searchable encryption has been widely used in cloud storage due to its functionality of dynamic encrypted data search. However, recent studies have shown that dynamic searchable encryption is vulnerable to file injection attacks. In order to resist such attacks, several forward secure symmetric searchable encryption schemes have been proposed. Unfortunately, most of the existing forward secure symmetric searchable solutions only work in the single user setting. In NSS 2018, Wang et al. proposed a multi-user forward secure dynamic searchable encryption scheme (MFS), by introducing a semi-honest proxy server that does not collude with the cloud server. However, we found that the forward security of the scheme can be compromised by the adversary who observes the association between the new update and the previous search tokens through eavesdropping attacks or replay attacks. To address this issue, a multi-user forward secure dynamic searchable symmetric encryption scheme EMFS is proposed with enhanced security, by exploiting user authentication mechanism without the need of state information transfer. We also adopt a new index structure to improve the efficiency. Finally, we give formal security proof to show that our scheme can resist the two attacks mentioned above, while maintaining forward security. Compared with Wang et al's scheme, our construction provides a higher level of practical efficiency by reducing the complexity of deletion from $O(n_w)$ to $O(1)$, where n_w denotes the number of matching documents for keyword w.

Key words dynamic symmetric searchable encryption; cloud storage; proxy server; forward security; multi-user

摘 要 动态对称可搜索加密由于其具有良好的动态密文数据搜索功能而在云存储中得到了广泛的应用,但最近研究表明,动态可搜索加密很容易遭受文件注入攻击.为了抵抗这种攻击,前向安全的对称可搜索加密方案被相继提出.可是,现有的前向安全对称可搜索方案大多只支持单用户.最近,Wang 等人在 NSS 2018 上提出了多用户环境下的前向安全动态可搜索加密方案(multi-user forward secure dynamic searchable encryption scheme, MFS),通过引入一个半诚实且不与云服务器合谋的代理服务

收稿日期:2020-06-12;**修回日期**:2020-07-27
基金项目:上海市自然科学基金项目(20ZR1418400);国家自然科学基金项目(61602180,61702187,61632012,61672239,U1636216);中央高校基本科研业务费专项资金;中国博士后科学基金项目(2017M611502)
This work was supported by Shanghai Natural Science Foundation (20ZR1418400), the National Natural Science Foundation of China (61602180, 61702187, 61632012, 61672239, U1636216), the Fundamental Research Funds for the Central Universities, and the China Postdoctoral Science Foundation (2017M611502).
通信作者:周俊(jzhou@sei.ecnu.edu.cn)

器,解决了多用户查询的问题.但是,发现敌手可以通过窃听攻击或重放攻击找出更新文件与旧的搜索令牌之间的关联,从而破坏 MFS 方案的前向安全性.为了解决这个问题,提出了一个增强的多用户前向安全动态可搜索加密方案 EMFS,通过去除用户和代理服务器之间的状态值传递和用户身份验证来抵抗窃听攻击和重放攻击.该方案采用了一个新的索引结构,能够有效地提升删除效率.最后,给出了形式化的安全证明,证明了 EMFS 方案在保证前向安全同时,能够抵抗上述 2 种攻击,并且把删除的复杂度从 $O(n_w)$降低到 $O(1)$,其中 n_w 表示匹配关键字 w 的文件个数.

关键词 动态对称可搜索加密;云存储;代理服务器;前向安全;多用户

中图法分类号 TP309

随着大数据时代的到来,数据的飞速增长使得维护、管理和利用数据逐渐变得困难.因此,越来越多的企业和个人都倾向于将数据外包到云上以节省本地存储和维护数据的开销.为了保护敏感数据,数据拥有者往往会选择将数据加密后再上传至云服务器,然而这会导致检索数据变得困难.因此,如何在加密的数据库上实现安全且高效的关键字搜索成为一个亟待解决的问题.

对称可搜索加密技术(symmetric searchable encryption, SSE)[1]是一种允许第三方对用户上传的加密数据进行检索,同时不会泄露除搜索模式和搜索结果外任何信息的一种密文技术.早期的对称可搜索加密往往仅适用于静态环境,即加密数据一旦上传至云服务器就不再进行修改与更新.但在实际应用中,用户往往希望能够实现数据库的更新,例如文件的插入、删除等,为此提出了动态对称可搜索加密的概念[2].然而在动态 SSE 环境下,数据的添加和删除将会比搜索泄露更多的隐私信息.最近,Zhang 等人[3]提出了一种针对动态可搜索加密方案的攻击,敌手可以通过向服务器注入少量文件来破坏用户的查询隐私.为了抵抗这种攻击,前向安全[4-5]的定义和方案相继被提出.这种方案可以防止旧的搜索令牌对新添加的密文文档进行搜索,因此大大降低了暴露令牌和数据集隐私的可能性.

目前大部分的动态对称可搜索加密方案仅支持单用户,不适用于多用户环境.这是由于这些方案往往要求客户端在本地维护关键字的状态信息以生成最新的限门.在这种设计下,其他用户只能向数据拥有者询问最新的关键字状态信息才能够生成限门,这就要求数据拥有者始终保持在线状态.为了解决这个问题,最近,Wang 等人[6]提出了一种支持多用户且前向安全的动态可搜索加密方案 MFS,该方案通过引入代理服务器来存储关键字信息和用户的访问控制权限,解决了前向安全的多用户加密数据搜索问题.然而,MFS 方案中仅给出了简单的安全性分析,缺乏形式化的安全性定义和证明;并且,我们注意到 MFS 方案中并未考虑数据拥有者和用户在与代理服务器通信时产生的信息泄露,主要包括 2 点:

1) 敏感信息泄露.为了使代理服务器得到最新的关键字信息,数据拥有者需要在每次文件更新发生时向服务器发送一些辅助信息,但是这些信息会泄露更新文件中包含的关键字与旧的搜索令牌之间的关联,使方案无法保持前向安全性.

2) 搜索令牌可关联.一个用户对于同一个关键字生成的搜索令牌始终是固定值.

针对上述存在的问题,我们提出了一个增强的多用户前向安全动态可搜索加密方案.本文的主要贡献包括 3 个方面:

1) 指出了 MFS 方案中存在敏感信息泄露和搜索令牌可关联的安全问题,并基于这 2 个安全问题提出了攻击的方法.

2) 提出了增强的多用户前向安全动态可搜索加密方案 EMFS,通过增加用户与代理服务器之间的通信密钥,保证了搜索令牌的不可关联性.并且,我们还将关键字状态信息生成全权授权给代理服务器,避免了状态信息在传递中造成的泄露.此外,我们的方案还采用了一种新的索引结构,能够大大提升删除的效率.

3) 给出了 EMFS 方案的安全性证明和性能对比,实验结果表明尽管我们的方案在查找复杂度上有略微增加,但删除复杂度从 $O(n_w)$降低到 $O(1)$,因此在实际应用中具有更高的效率.

1 相关工作

可搜索对称加密是一种非常有效的加密工具,能在实现隐私保护的同时保证可搜索性[7].具体来说,一个 SSE 方案允许数据所有者对自己的数据进行

加密,并将加密的数据库外包给云服务器,之后云服务器在接收到有效的查询请求时可以对密文进行搜索操作.首个对称可搜索方案由 Song 等人[8]提出,通过顺序扫描技术来实现搜索操作.此后,诸多 SSE 方案被不断提出[9-15],大大丰富 SSE 的查询表达性,提升了方案安全性和性能.

随后,为了满足数据动态性的要求,文献[2]提出了首个动态可搜索加密方案,实现了搜索和更新操作.但是,动态可搜索加密在数据添加和数据删除过程中会泄露额外的信息.例如敌手可以分析新添加或删除的文档是否包含先前搜索的关键字.Cash 等人[16]对动态 SSE 方案的泄露进行了研究,表明即使是最小的泄露也能够被攻击者利用来揭示用户查询的内容,导致泄露滥用攻击.最近 Zhang 等人[3]提出的一种恶意攻击称为文件注入攻击,通过向数据库中注入少量文件来破坏用户的查询隐私.为了抵抗上述攻击,Stefanov 等人[4]首先提出了 2 个安全性概念,即前向安全和后向安全,并提出了一个类 ORAM 构造的前向安全的动态可搜索加密方案.随后,Bost[5]在 2016 年提出了一个基于陷门原语支持增添前向安全动态可搜索加密方案,并正式给出了前向安全的定义.2017 年 Bost 等人[17]给出了后向安全由强到弱 3 种正式定义,并给出了一个支持单关键字搜索且满足前向和后向安全的动态可搜索加密方案.

在前向安全方面,文献[18]提出了可验证的前向安全动态可搜索加密方案,使方案适用于恶意服务器环境.文献[19]通过树状结构实现了一种支持范围搜索的前向安全动态可搜索加密方案.文献[20-21]提出了一种支持多关键字搜索的前向安全动态可搜索加密方案.

在后向安全方面,文献[22]提出了一种实现了第 3 类后向安全的对称可搜索加密方案,作者构造了一种新的密码学原语称为对称可穿刺加密,大大降低了通过穿刺实现后向加密所需的计算开销.文献[23]采用了一次一密的方式构造了一种后向安全的对称可搜索加密方案,该方案不仅高效,还能实现比第 1 类后向安全更高的安全性,缺陷是无法支持大量数据.文献[24]从第 2 类、第 3 类后向安全入手,分别提出了 2 种满足后向安全的对称可搜索加密方案,表明了实现后向安全的成本大大低于实现前向安全的成本.

尽管多用户在动态对称可搜索加密方案中并不少见,如文献[25]实现了一个支持多用户的动态对称可搜索加密方案,通过密钥分发和重新加密技术避免了密钥共享带来的一系列问题.文献[26]将客户端的授权信息合并到搜索令牌和索引中提出了一个支持布尔型查询的动态多用户可搜索方案.但是,前向安全和后向安全方案在目前大多仅支持单用户模型.为了支持多用户环境,Wang 等人[6]首次从单用户模型进行扩展,提出了一种支持多用户的前向安全动态可搜索加密方案 MFS,这为现有的单用户前向安全可搜索方案扩展提供了一种新的思路.然而,MFS 方案中仍然存在无法抵抗窃听攻击或重放攻击等安全缺陷,且搜索与删除效率也有待进一步提高.我们认为这是单用户扩展至多用户前向安全动态对称可搜索加密方案时必须解决的关键问题之一,为此我们提供了一种解决思路并给出了一个新的方案.

2 预备知识

2.1 双线性映射

设 G_1,G_2 均为阶为素数 p 的乘法循环群,其中 g 是 G_1 的生成元. 双线性映射 $e:G_1\times G_1\rightarrow G_2$ 具有 3 条性质:

1) 双线性.对于任意 $u,v\in G_1$,$a,b\in Z_p$,有 $e(u^a,v^b)=e(u,v)^{ab}$.

2) 非退化性.$e(g,g)\neq 1$.

3) 可计算性.对于任意 $u,v\in G_1$,$e(u,v)$的计算应该是高效的.

2.2 伪随机函数

定义函数 $F:\{0,1\}^\lambda\times\{0,1\}^m\rightarrow\{0,1\}^n$,我们称其为伪随机函数,如果对于所有概率性多项式时间的区分器 D 有:$|Pr[D^{F(k,\cdot)}=1\,|\,k\leftarrow\{0,1\}^\lambda]-Pr[D^g=1\,|\,g\leftarrow\{Func[m,n]\}]|\leqslant negl(\lambda)$,其中 $negl(\lambda)$是一个可忽略函数.

2.3 伪随机置换

给定密钥空间 $\mathcal{K}$、定义域 $\mathcal{X}$ 和值域 $\mathcal{Y}$,一个伪随机置换包含 2 个函数 $F:(\mathcal{K}\times\mathcal{X})\rightarrow\mathcal{Y}$ 以及 $F^{-1}:(\mathcal{K}\times\mathcal{Y})\rightarrow\mathcal{X}\cup\{\perp\}$.其中,函数 F 和 F^{-1} 满足 4 种性质:

1) $F^{-1}(k,F(k,x))=x$,$\forall x\in\mathcal{X}$.

2) $F^{-1}(k,y)=\perp$ 当且仅当 $y\notin\mathcal{Y}$.

3) F 和 F^{-1} 可以通过确定性多项式算法高效地计算.

4) $F(k,.)$和 $F^{-1}(k,.)$是 $\mathcal{X}$ 到 $\mathcal{Y}$,$\mathcal{Y}$ 到 $\mathcal{X}$ 的单射函数.

为了便于后续方案中的使用,我们额外定义了一个算法 $GenIPRF(1^\lambda)$,给定安全参数 λ,输出为从密钥空间 $\mathcal{K}$ 中随机均匀选取的值 k.

我们称伪随机置换是安全的,如果对于所有概率性多项式时间的区分器 D 有:$|Pr[D^{F(k,\cdot),F^{-1}(k,\cdot)}=1|k\leftarrow GenIPRF(1^\lambda)]-Pr[D^{f(\cdot),f^{-1}(\cdot)}=1|k\leftarrow\{Func[\mathcal{X},\mathcal{Y}]\}]|\leqslant negl(\lambda)$,其中 $negl(\lambda)$ 是一个可忽略函数.

3 系统模型

方案中涉及 4 类实体:数据拥有者(data owner, DO)、数据使用者(data user, DU)、云服务器(cloud server, CS)和代理服务器(proxy server, PS),具体系统模型如图 1 所示.

1) 数据拥有者负责为需要上传的文档选取对应关键字,加密并上传文档,并为文档生成辅助信息.其中,密文发送到云服务器,辅助信息发送到代理服务器.最后,数据拥有者还会为授权用户生成私钥,并通过安全信道将私钥分发给授权用户.

2) 云服务器是"诚实且具有好奇心的",用于存储加密文件、对应的索引以及用户信息,同时对代理服务器提交的查询请求进行响应,并将查询结果返回给对应用户.

3) 代理服务器是"诚实的",负责为关键字生成状态信息,存储每个关键字对应的最新状态值和用户验证信息,负责验证数据使用者的身份,并为数据使用者生成搜索令牌提交给云服务器.

4) 数据使用者是"诚实的",可以为需要搜索的关键字生成验证令牌,并将令牌提交给代理服务器,最终得到云服务器返回的搜索结果.

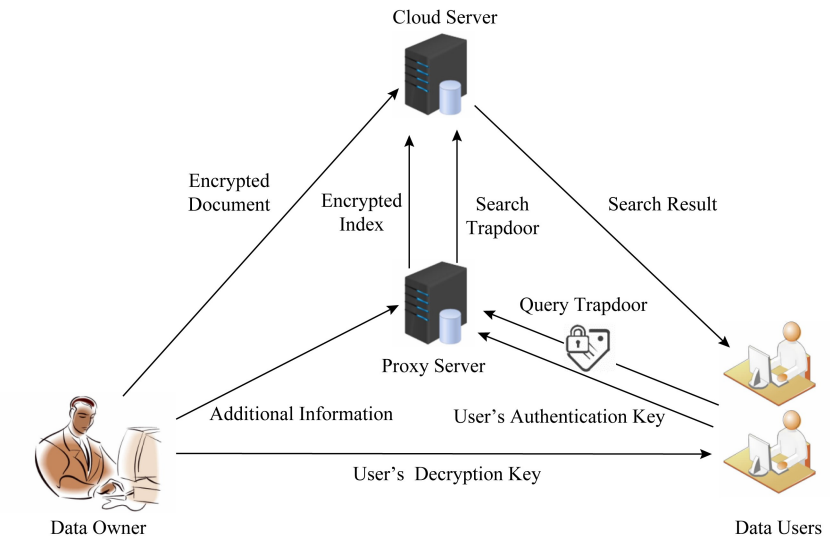

Fig. 1 System model

图 1 系统模型

4 动态可搜索加密

4.1 形式化定义

在本文的构造中,数据拥有者能够与多个用户共享文件,但是,只有数据所有者可以对数据集进行添加和删除.下面,我们给出了动态对称可搜索加密方案的定义:

定义 1. 动态对称可搜索加密(dynamic symmetric searchable encryption, DSSE)方案由多项式时间算法构成:

$Setup(1^\lambda,DB)\rightarrow(SK_o,st_q,EDB)$.数据拥有者选择安全参数 1^λ 和数据库 DB 作为输入,输出为数据拥有者的私钥 SK_o、关键字状态值 st_q 和加密数据库 EDB.其中,关键字状态值由代理服务器管理.

$Register(u,Ucqt,Udqt)\rightarrow(SK_u,Ucqt',Udqt')$.用户注册算法,密钥由数据拥有者和用户共同生成,最终用户保留密钥 SK_u,代理服务器更新用户搜索表 $Ucqt$,云服务器更新用户解密表 $Udqt$.

$Add(SK_o,f,st_q,EDB)\rightarrow(st_q',EDB')$.数据拥有者运行该算法将文件 f 添加到数据集中,代理服务器中的状态值和云服务器中的索引及密文会相应更新.

$Delete(SK_o,id(f),st_q,EDB)\rightarrow(EDB')$.数据拥有者运行此算法从数据集中删除文件 f,存储在云服务器上的索引和密文会相应地更新.

$Search(w, SK_u, st_q, EDB) \rightarrow (rst_w)$.用户运行此算法搜索包含关键字 w 的文件,搜索令牌由代理服务器生成提交到云服务器,云服务器将结果集返回给用户.

$Decrypt(rst, SK_u) \rightarrow F$.用户收到云服务器返回的数据集 rst 后,可以用私钥进行解密,得到满足查询结果的文件集 F.

4.2 泄露函数

本节给出 DSSE 方案泄露函数的一般定义:

1) $sp(w)=\{t:(t,w)\in Q\}$.搜索模式存储了查询和关键字之间的映射,该映射用于判断查询是否针对同一个关键字,其中 t 代表时间戳.

2) $rp(w)=rst_t$.访问模式被定义为每个搜索查询的结果,其中 rst_t 表示当前匹配关键字 w 的所有文件.

3) $UpHist(w)$.以 (t,op,ind) 的形式记录了关键字 w 上的所有更新操作,其中 t 为时间戳,op 为操作符,ind 为文件索引.

4.3 安全性定义

DSSE 方案的安全性主要通过 $REAL_A^{\Sigma}(\lambda)$ 和 $IDEAL_{A,S}^{\Sigma}(\lambda)$ 这 2 个游戏来制定.其中 $REAL_A^{\Sigma}(\lambda)$ 由 DSSE 执行,$IDEAL_{A,S}^{\Sigma}(\lambda)$ 通过 DSSE 的泄露来模拟.我们将 DSSE 方案的泄露函数定义为 $\mathcal{L}=(\mathcal{L}^{Setup},\mathcal{L}^{Search},\mathcal{L}^{Add},\mathcal{L}^{Delete})$,它描述了敌手 $\mathcal{A}$ 在方案运行期间能获得的信息.如果敌手 $\mathcal{A}$ 不能区别这 2 个游戏,那么我们认为泄露的信息仅仅局限于泄露函数 $\mathcal{L}$.下面给出 2 个游戏的具体定义:

1) $REAL_{\mathcal{A}}^{\Sigma}(\lambda)$.输入敌手 $\mathcal{A}$ 选定数据库 DB,通过运行 $Setup(\lambda,DB)$ 算法输出 EDB 并发送给敌手 $\mathcal{A}$.敌手 $\mathcal{A}$ 可以重复执行 3 种查询:搜索查询 w、添加查询 f_1 以及删除查询 $id(f_2)$.其中 DB 中至少存在一个文件 f 包含关键字 w,$f_1\notin DB$,$f_2\in DB$.游戏分别运行搜索和更新算法生成搜索令牌 τ_w、添加令牌 τ_a 以及删除令牌 τ_d 返回给敌手 $\mathcal{A}$.最终,$\mathcal{A}$ 输出一个字符.

2) $IDEAL_{\mathcal{A},S}^{\Sigma}(\lambda)$.敌手 $\mathcal{A}$ 选定数据库 DB,使用模拟器 $\mathcal{S}(\mathcal{L}^{Setup}(\lambda,DB))$ 将 EDB 返回给敌手 $\mathcal{A}$.敌手 $\mathcal{A}$ 可以重复执行 3 种查询:搜索查询 w、添加查询 f_1 以及删除查询 $id(f_2)$.其中 DB 中至少存在一个文件 f 含有包含关键字 w,$f_1\notin DB$,$f_2\in DB$.根据敌手 $\mathcal{A}$ 提交的查询,模拟器 $\mathcal{S}$ 使用泄露函数 $\mathcal{L}^{Search}(w)$,$\mathcal{L}^{Add}(f)$ 和 $\mathcal{L}^{Delete}(id(f))$ 模拟生成令牌返回给敌手 $\mathcal{A}$.最终,$\mathcal{A}$ 输出一个字符.

定义 2. 我们称方案 Σ 是 $\mathcal{L}$-自适应安全的,如果对于所有多项式时间的敌手,存在一个高效的模拟器 $\mathcal{S}$,使得:$|Pr[REAL_{\mathcal{A}}^{\Sigma}(\lambda)=1]-Pr[IDEAL_{\mathcal{A},S}^{\Sigma}(\lambda)=1]|\leqslant negl(\lambda)$.

4.4 前向安全

DSSE 方案的前向安全性要求旧的搜索令牌无法匹配到新的更新,换而言之,数据的更新操作不能泄露有关关键字的任何信息.我们定义前向安全 SSE 如下:

定义 3. 称一个 $\mathcal{L}$-自适应安全的 DSSE 方案是前向安全的,如果方案中关于更新操作的泄露函数可以分别描述为

$$\mathcal{L}^{Add}(f)=\mathcal{L}'(id(f),|f|,op,\mu_f),$$

$$\mathcal{L}^{Delete}(id(f))=\mathcal{L}'(id(f),op,\mu_f),$$

其中,$|f|$ 代表文件 f 的大小,op 代表操作符(在 $\mathcal{L}^{Add}$ 中 $op=add$,在 $\mathcal{L}^{Delete}$ 中 $op=del$),μ_f 表示文件 f 包含的关键字个数.

5 MFS 方案分析

为了更好地阐述我们的观点,本节首先介绍 Wang 等人[6]提出的多用户前向安全动态可搜索加密方案 MFS,并对方案中存在的安全性问题进行了分析,最后介绍了攻击的方法.

5.1 MFS 方案

5.1.1 初始化算法 $Setup(1^{\lambda})$

给定安全参数 λ,从 Z_p^* 中选取随机数 (wk,ek) 作为数据拥有者的搜索和加密密钥,随机选取 sk_1,sk_2,$k_s\in\{0,1\}^{\lambda}$.给定双线性映射 $\hat{e}:G_1\times G_1\rightarrow G_2$,$g_1$,$g_2$ 为分别为 G_1,G_2 的生成元,Hash 函数 h_1,h_2,h_3,H_1,H_2.服务器端初始化索引 T,代理服务器初始化索引 W.数据拥有者保存私钥 $SK=(wk,ek,sk_1,sk_2,k_s)$,公布公共参数 $pp=(p,g_1,g_2,G_1,G_2,h_1,h_2,h_3,H_1,H_2)$.

5.1.2 文件添加算法 $Add(SK,f,W,EDB)$

1) $DataOwner(SK,f)$:给定一个文件 f,文件标识符 ind_f,文档中包含的所有关键字 $W(f)$,从 G_1 中随机选取 r_f,计算文件加密密钥 $ek_f=h_3(\hat{e}(r_f,g_2)^{ek})$,密文 $C_f=AES.Encrypt(ek_f,f)$,将 (C_f,r_f) 发送给云服务器.此外,对于每个关键字 $w\in W(f)$,执行操作:

① $addr(w,ind_f)=H_1(sk_1,w\|ind_f)$;

② $k(w,ind_f)=H_2(sk_2,w\|ind_f)$;

③ $t_w=F(k_s,w)$;

④ $Tr(w)=h_2(\hat{e}(h_1(w),g_2)^{wk})$;

⑤ $e_2=ind_f\oplus F(k(w,ind_f),t_w)$;

⑥ $AddTuple\leftarrow(Tr(w),addr(w,ind_f),k(w,ind_f),t_w,e_2)$;

⑦ 将 $AddTuple$ 发送给代理服务器.

2) $ProxyServer(AddTuple,W)$:代理服务器收到 $AddTuple=(Tr(w),addr(w,ind_f),k(w,ind_f),t_w,e_2)$后,执行操作:

① if $W[Tr(w)]=$null then

② $s\leftarrow\{0,1\}^\lambda$;

③ $e_1=\langle 0^\mu\parallel s\rangle\oplus\langle Tr(w)\parallel F(k(w,ind_f),addr(w,ind_f))\rangle$;

④ else

⑤ $(addr(w,ind_c),k(w,ind_c),t_w)\leftarrow W[Tr(w)]$;

⑥ $e_1=\langle addr(w,ind_c)\parallel k(w,ind_c)\rangle\oplus\langle Tr(w)\parallel F(k(w,ind_f),addr(w,ind_f))\rangle$;

⑦ end if

⑧ $W[Tr(w)]\leftarrow(addr(w,ind_f),k(w,ind_f),t_w)$;

⑨ $AddTuple\leftarrow(addr(w,ind_f),e_1,e_2)$;

⑩ 将 $AddTuple$ 发送给云服务器.

3) $CloudServer(AddTr,EDB)$:云服务器更新 T,使得 $T[addr(w,ind_c)]\leftarrow(e_1,e_2)$.

5.1.3 用户注册算法 $Register(u)$

给定用户 u 的身份 $uid\in\{0,1\}^\lambda$,从 Z_p^* 中选取随机数(qk_u,dk_u)作为用户的搜索及文件加密密钥,计算 $qk_c=g_2^{wk/qk_u}$,$dk_c=g_2^{ek/dk_u}$. 其中,(uid,qk_c)发送给代理服务器,(uid,dk_c)发送给云服务器,(uid,qk_u,dk_u)发送给注册用户 u.

5.1.4 搜索算法 $Search((w,qk_u),(qk_c,st_q),(dk_c,EDB))$

1) $DataUser(w,qk_u)$:计算 $Tr_u(w)=h_1(w)^{qk_u}$,将$(uid,Tr_u(w))$发送给代理服务器.

2) $ProxyServer(Tr_u(w),uid,qk_c,W)$:给定搜索用户的 uid 以及查询令牌 $Tr_u(w)$,执行操作:

① if $Ucqt[uid]\neq\perp$ then

② $qk_c\leftarrow Ucqt[uid]$;

③ else

④ 非授权用户,程序结束;

⑤ end if

⑥ $Tr(w)=h_2(\hat{e}(Tr_u(w),qk_c))$;

⑦ if $W[Tr(w)]\neq\perp$ then

⑧ $(addr(w,ind_f),k(w,ind_f),t_w)\leftarrow W[Tr(w)]$;

⑨ $SearchTrapdoor\leftarrow(Tr_u(w),addr(w,ind_f),t_w,k(w,ind_f))$;

⑩ 将 $SearchTrapdoor$ 和 uid 发送给云服务器;

⑪ else

⑫ 返回“none”给用户 u;

⑬ end if

3) $CloudServer(SearchTrapdoor,uid,qk_c,EDB)$:初始化 2 个集合 $ResultSet$ 和 rst 用于存放查询结果及数据,并执行操作:

① while $addr(w,ind_c)\neq 0^\mu$ do

② $ind=T[addr(w,ind_c)].e_2\oplus F(k(w,ind_c),i_w)$;

③ $\langle addr(w,ind_c)\parallel k(w,ind_c)\rangle\leftarrow T[addr(w,ind_c)].e\oplus\langle Tr(w)\parallel F(k(w,ind_f),addr(w,ind_f))\rangle$;

④ $ResultSet\leftarrow ResultSet\cup ind$;

⑤ end while

⑥ $dk_c=U[uid]$;

⑦ for each $ind\in ResultSet$ do

⑧ $cds_{ind}=\hat{e}(r_{ind},dk_c)$;

⑨ $rst\leftarrow rst\cup(cds_{ind},C_{ind})$;

⑩ end for

⑪ 将数据集 rst 返回给用户.

5.1.5 解密算法 $Decrypt(rst,dk_u)$

用户收到数据集 rst 后执行操作:

① $F=$null;

② for each $(cds_{ind},C_{ind})\in rst$

③ $dk_{ind}=h_3(cds_{ind}^{dk_u})$;

④ $f=AES.Decrypt(dk_{ind},C_{ind})$;

⑤ $F=F\cup f$;

⑥ end for

最终得到所有符合查询的结果.

5.2 安全性问题

MFS 方案是首个在单用户模型下扩展至多用户模型的前向安全动态可搜索加密方案,为了解决前向安全和多用户合并的问题,作者引入了第三方代理服务器来存储关键字信息和用户的访问控制权限.然而由于作者将代理服务器设置为半可信,使得数据拥有者不得不泄露某些信息以达到托管关键字信息的目的.由于代理是半可信的,本质上这是把部分泄露转移到了代理服务器上,违背前向安全的性质.因此尽管 MFS 方案中考虑了诸多安全因素,其中依然存在 2 项严重的安全性问题,最终致使 MFS 方案无法保证前向安全性.

1) 敏感信息泄露.在文件添加算法 Add 中,我们注意到用户将 $AddTuple$ 直接发送给代理服务器,其中包含了关键字标识符 t_w,这是极度不安全的,因为旧的搜索令牌中同样包含了 t_w,而前向安全要求隐藏更新文件与旧的搜索令牌之间的关系.

2) 搜索令牌可关联.对于某一用户,他搜索同一关键字时向代理服务器提交的令牌始终是一个固定值,这使敌手可以轻易伪装成任何用户.

5.3 攻击方法

根据 5.2 节分析的安全性问题,我们给出 2 种攻击方法:

1) 窃听攻击.敌手通过监听数据拥有者向代理服务器发送的信息来维护一个关键字/文件表,每当数据拥有者向代理服务器发送更新令牌 $AddTuple$ 时,敌手将 $AddTuple$ 中的关键字标识符 t_w 和文件 ind 存放到表中(ind 可以通过 $AddTuple$ 中的其他值计算).对于窃听敌手来说,尽管他并不知道 t_w 对应的关键字,但是他能轻而易举地将搜索令牌与更新文件进行关联(搜索令牌中同样包含 t_w),这对保证方案的前向安全性是十分不利的.

2) 重放攻击.敌手维护一个用户令牌表,存放用户向代理服务发送的令牌,一旦发现数据拥有者更新了一个新的文件,敌手就将他存储的所有令牌发送给代理服务器.在这种情况下,云服务器不断获得用户搜索过的所有关键字对应的最新搜索令牌.如果令牌包含了最新的更新索引,那么云服务器就掌握了更新文件所包含的关键字.注意此时没有任何用户提交有关该文件的搜索请求,云服务器应对该文件中包含的关键字保持未知.

6 EMFS 方案

6.1 数据结构

我们的方案采用了状态链构造,如图 2 所示,每个关键字对应一条状态链,所有匹配关键字 w 的文件标识符都存放在链中,当客户端想要搜索关键字 w 时,他向服务器发送最后一个状态 st_{c+1},服务器可以从 st_{c+1} 开始反向遍历状态链获得所有先前状态 $st_c, st_{c-1}, \cdots, st_1$,最终获得所有查询结果.需要注意的是,服务器无法从当前状态 st_{c+1} 获取下一个状态 st_{c+2},这也确保了方案的前向安全性.为了进一步提高搜索和更新的效率,我们采用随机生成的方式来生成状态值.

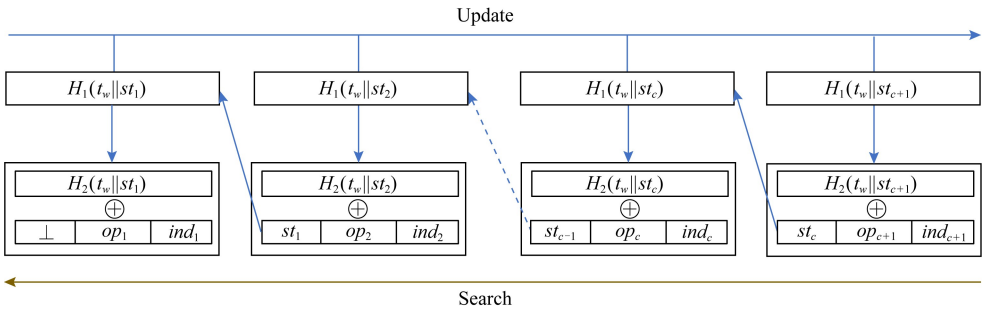

Fig. 2 Update and search in our scheme

图 2 更新和查找图示

为了使方案适应多用户环境,我们选择引入一个诚实的代理服务器来维护关键字的状态值,并将状态值的生成全权及交给代理服务器,避免用户生成状态值传递给服务器导致的信息泄露.需要注意的是,为了保证方案的非交互性,代理服务器除了记录关键字最新的状态信息外,还会额外保存各关键字对应的文件数和用户的搜索次数用于用户验证(令牌的生成与这些信息相关).虽然敌手可能会获知这些信息,例如某用户的查询次数,但我们认为在未持有密钥的情况下,敌手伪造令牌的概率仅为一个可忽略值.

6.2 具体构造

6.2.1 初始化算法 $Setup(1^\lambda)$

给定安全参数 λ,从 Z_p^* 中选取随机数 ek,随机选取 $k_s \in \{0,1\}^\lambda$.给定双线性映射 $\hat{e}: G_1 \times G_1 \to G_2$,$g_1, g_2$ 分别为 G_1, G_2 的生成元,伪随机置换 R_1, R_2,Hash 函数 H_1, H_2, H_3,运行算法 $GenIPRF(1^\lambda)$ 生成伪随机置换 R_1 的密钥 K_G.数据拥有者初始化更新集合 $Files$,服务器端初始化索引集合 T,代理服务器初始化映射集合 W. 数据拥有者保存私钥 $SK_o=(ek, k_s, K_G)$,代理服务器秘密保存 K_G,公布公共参数 $pp=(p, g_1, g_2, G_1, G_2, H_1, H_2, H_3, R_1, R_1^{-1}, R_2, R_2^{-1})$.

6.2.2 文件添加算法 $Add(SK_o, f, W, EDB)$

1) $DataOwner(SK_o, f)$:给定一个文件 f,文件标识符 ind_f,文档中包含的所有关键字 $W(f)$,从 G_1 中随机选取 r_f,计算文件加密密钥 $ek_f = H_3(\hat{e}(r_f, g_2)^{ek})$,密文 $C_f = AES.Encrypt(ek_f, f)$,将 (C_f, r_f) 发送给云服务器. 此外,初始化一个集合 τ_{upd},对于每个关键字 $w \in W(f)$,执行操作:

① $Files[w] = Files[w] + 1$;

② $t_w = F(k_s, w)$;

③ $upd_w = R_1(K_G, t_w \| add \| ind \| Files[w])$;

/ * $op = add$ 代表文件插入 * /

④ $\tau_{upd} = \tau_{upd} \bigcup upd_w$.

然后,将 τ_{upd} 发送给代理服务器.

2) $ProxyServer(K_G, W, \tau_{upd})$:代理服务器收到 τ_{upd} 后,初始化一个集合 τ_{td},并执行操作:

① for each $upd_w \in \tau_{upd}$ do

② $t_w \| op \| ind \| Files[w] = R_1^{-1}(K_G, upd_w)$; / * 根据 op 的不同分别代表文件插入和删除 * /

③ $(st_c, c) \leftarrow W[t_w]$;

④ if $Files[w] \neq c + 1$ then

⑤ return "无效令牌";

⑥ else if $c = 0$ then

⑦ $st_{c+1} \leftarrow \{0,1\}^{\lambda}$;

⑧ $e = H_2(t_w \| st_{c+1}) \oplus (\perp \| op \| ind)$;

⑨ else

⑩ $st_{c+1} \leftarrow \{0,1\}^{\lambda}$;

⑪ $e = H_2(t_w \| st_{c+1}) \oplus (st_c \| op \| ind)$;

⑫ end if

⑬ $c = c + 1$;

⑭ $u = H_1(t_w, st_{c+1})$;

⑮ $W[t_w] \leftarrow (st_{c+1}, c)$;

⑯ $\tau_{td} = \tau_{td} \bigcup (u, e)$;

⑰ end for

⑱ 将 τ_{td} 发送给云服务器.

3) $CloudServer(\tau_{td}, EDB)$:云服务器根据收到的 τ_{td} 执行操作:

① for each $(u, e) \in \tau_{td}$ do

② $T[u] = e$;

③ end for

6.2.3 用户注册算法 $Register(u)$

给定用户 u 的身份 $uid \in \{0,1\}^{\lambda}$,数据拥有者从 Z_p^* 中选取随机数 dk_u 作为用户的文件解密密钥,计算 $dk_c = g_2^{ek|dk_u}$,用户运行算法 $GenIPRF(1^{\lambda})$ 生成伪随机置换 R_2 的密钥 qk_u,计数器 $s = 0$.其中,代理服务器保存 (uid, qk_u, s),云服务器保存 (uid, dk_c),用户保存 $SK_u = (uid, qk_u, k_s, dk_u, s)$.

6.2.4 搜索算法 $Search((w, SK_u), (qk_u, W), (dk_c, EDB))$

1) $DataUser(w, SK_u)$:给定搜索的关键字 w,执行操作:

① $(uid, qk_u, k_s, dk_u, s) \leftarrow SK_u$;

② $t_w = F(k_s, w)$;

③ $s = s + 1$;

④ $\tau_u = R_2(qk_u, t_w \| s)$;

⑤ $SK_u \leftarrow (uid, qk_u, k_s, dk_u, s)$.

将 (uid, τ_u) 发送给代理服务器.

2) $ProxyServer(uid, W, Ucqt, \tau_u)$:给定搜索用户的 uid 以及令牌 τ_u,执行操作:

① $(qk_u, s_c) \leftarrow Ucqt[uid]$;

② $t_w \| s = R_2^{-1}(qk_u, \tau_u)$;

③ if $s \neq s_c + 1$ then

④ return "无效令牌";

⑤ end if

⑥ $(st_c, c) \leftarrow W[t_w]$;

⑦ $\tau_w \leftarrow (t_w, st_c)$;

⑧ if $st_c = \perp$ then

⑨ 返回消息"none"给用户 u;

⑩ else

⑪ 将 (uid, τ_w) 发送给云服务器;

⑫ end if

⑬ $s_c = s_c + 1$;

⑭ $U[uid] \leftarrow (qk_u, s_c)$.

3) $CloudServer(uid, Udqt, \tau_w)$:给定搜索用户的 uid 以及搜索令牌 $\tau_w = (t_w, st_c)$,云服务器初始化 3 个集合 Δ,R 和 rst 并执行操作:

① $head = H_1(t_w \| st_c)$;

② $flag = $ false;

③ while $st_c \neq \perp$ do

④ $u = H_1(t_w \| st_c)$;

⑤ $e = T[u]$;

⑥ $(st_c \| op \| ind) = e \oplus H_2(t_w \| st_c)$;

⑦ if $op = del$ then

⑧ $\Delta = \Delta \bigcup ind$;

⑨ if $u \neq head$ then

⑩ 删除当前块;

⑪ else

⑫ $flag = $ true;

⑬ end if

⑭ else if $op = add$ then

⑮ if $ind \in \Delta$ && $flag$ = false then
⑯ $\Delta = \Delta \backslash ind$;
⑰ 删除当前块;
⑱ else if $ind \in \Delta$ && $flag$ = true then
⑲ $\Delta = \Delta \backslash ind$;
⑳ else
㉑ $ResultSet \leftarrow ResultSet \cup ind$;
㉒ end if
㉓ end if
㉔ end while
㉕ $dk_c \leftarrow Udqt[uid]$;
㉖ for each $ind \in ResultSet$ do
㉗ $cds_{ind} = \hat{e}(r_{ind}, dk_c)$;
㉘ $rst \leftarrow rst \cup (cds_{ind}, C_{ind})$;
㉙ end for
㉚ 将数据集 rst 返回给用户.

6.2.5 删除算法 $Delete(id(f), W(f), SK_o, P)$

1) $DataUser(id(f), W(f), SK_o)$:对于每个关键字 $w \in W(f)$,执行操作::

① $Files[w] = Files[w] + 1$;
② $t_w = F(k_s, w)$;
③ $upd_w = R_1(K_G, t_w \| del \| ind \| Files[w])$;
/ * $op = del$ 代表文件删除 * /
④ $\tau_{upd} = \tau_{upd} \cup upd_w$.

然后,将 τ_{upd} 发送给代理服务器.

2) $ProxyServer(K_G, W, \tau_{upd})$:代理服务器执行算法见 6.2.2 节.

3) $CloudServer(\tau_{td}, EDB)$:云服务器执行算法见 6.2.2 节.

6.2.6 解密算法 $Decrypt(rst, dk_u)$

用户收到数据集 rst 后执行操作:

① F = null;
② for each $(cds_{ind}, C_{ind}) \in rst$ do
③ $dk_{ind} = H_3(cds_{ind}{}^{dk_u})$;
④ $f = AES.Decrypt(dk_{ind}, C_{ind})$;
⑤ $F = F \cup f$;
⑥ end for

最终得到所有符合查询的结果.

7 安全性分析和证明

7.1 安全性分析

在给出 EMFS 方案的安全性证明前,我们首先从文件和索引的安全性、验证令牌的不可伪造性、搜索令牌的保密性来分析方案的安全性.

1) 文件安全性.文件由数据拥有者在本地加密后上传到云服务器,在本文中,我们选择用主流的对称加密算法 AES 来加密文件.AES 算法的安全性保证了在密钥不外泄的情况下,敌手无法区别一串 0 的加密与文件的加密,保证了文件的安全性.

2) 身份验证令牌的不可伪造性.方案中采用了可逆伪随机函数来生成身份验证令牌,所需的密钥由代理服务器和用户持有,并且生成的验证令牌仅仅能够使用一次.在密钥不外泄的情况下,可逆伪随机函数的安全性保证了敌手无法区分随机值和一个验证数据的加密,故我们认为敌手仅有可忽略的概率可以伪造验证令牌.

3) 搜索令牌的保密性.存储在云服务器端的索引中保存的并非是真正的关键字,而是关键字的标识符.在未持有密钥的情况下,我们认为云服务器无法根据标识符推测出其中的关键字的相关信息.

7.2 安全性证明

定理 1. 给定 CPA 安全的对称密钥加密方案 AES,伪随机函数 F,可逆伪随机函数 R,Hash 函数 H_1, H_2, H_3,双线性映射 e.如果 EMFS 方案的泄露函数 $\mathcal{L} = (\mathcal{L}^{Setup}, \mathcal{L}^{Search}, \mathcal{L}^{Add}, \mathcal{L}^{Delete})$ 满足定义:

$\mathcal{L}^{Setup}(\lambda) = \perp$,
$\mathcal{L}^{Search}(w) = \mathcal{L}'(sp(w), rp(w))$,
$\mathcal{L}^{Add}(f) = \mathcal{L}'(id(f), |f|, op, \mu_f)$,
$\mathcal{L}^{Delete}(f) = \mathcal{L}'(id(f), op, \mu_f)$,

则 EMFS 方案构造是 $\mathcal{L}$-自适应安全的.

证明. 构建一个模拟器 $\mathcal{S}$ 来模拟真实的算法,并且任何多项式时间的敌手都无法对真实算法和模拟算法进行区分.为了证明构造方案的安全性,我们使用了混合参数,模拟器 $\mathcal{S}$ 将通过泄露函数 $\mathcal{L}^{Setup}$, $\mathcal{L}^{Search}$, $\mathcal{L}^{Add}$ 和 $\mathcal{L}^{Delete}$ 模拟算法.

混淆 0:规定所有算法都按照协议运行.

混淆 1:模拟器 $\mathcal{S}$ 通过泄露函数 $\mathcal{L}^{Setup}(\lambda) = \perp$ 而非 $Setup(1^\lambda)$ 来模拟算法.

算法 1. 系统初始化模拟算法.

① $k \leftarrow AES.KeyGen(1^\lambda)$;
② 初始化字典 $Dict$ 存放模拟索引;
③ 初始化字典 $KeyStore$ 存放模拟关键字标识符;
④ 初始化字典 ST 存放模拟状态值;

混淆 2:和混淆 1 类似,模拟器 $\mathcal{S}$ 通过泄露函数 $\mathcal{L}^{Add}(f) = \mathcal{L}'(id(f), |f|, op, \mu_f)$ 模拟添加令牌.

算法 2. 添加令牌模拟算法.

① $\mathcal{L}^{Add}=$null;

② for $i=1$ to μ_f do

③ 随机生成(u,e)对;

④ 将$(id(f),(u,e))$添加到字典 $Dict$;

⑤ $\mathcal{L}^{Add}=\mathcal{L}^{Add}\cup(u,e)$;

⑥ end for

⑦ $c=ASE.Enc(k,0^{|f|})$;

⑧ $\tau_a(f)=(id(f),c,\mathcal{L}^{Add})$.

添加模拟算法允许模拟器 $\mathcal{S}$ 在初始化算法执行后用敌手 $\mathcal{A}$ 提供的文件保持字典的更新.在模拟算法中,我们采用能满足随机预言机模型的随机字符串来代替 Hash 函数 H 的输出.这个随机值会被存储在字典 $Dict$ 中,当下一次需要生成 Hash 值时,这个随机值就会被重用.模拟字典的大小与真实字典完全相同,因此敌手 $\mathcal{A}$ 无法进行区分.并且,我们使模拟器 $\mathcal{S}$ 模拟的令牌与真正的令牌同样保持完全相同的格式和大小,证明了仅知道 $\mathcal{L}^{Add}$ 就可以模拟添加令牌,即证明了我们的方案保持了前向安全性.

混淆 3:和混淆 1 类似,模拟器 $\mathcal{S}$ 通过泄露函数 $\mathcal{L}^{Search}(w)=\mathcal{L}'(sp(w),rp(w))$模拟搜索令牌.

算法 3. 搜索令牌模拟算法.

① $(sp(w),rp(w))\leftarrow\mathcal{L}^{Search}$;

② $rst\leftarrow rp(w)$;

③ if $KeyStore[w]=$null then

④ 随机生成 $KeyStore[w]$;

⑤ end if

⑥ $t_w=KeyStore[w]$;

⑦ if $ST[w]=$null then

⑧ 随机生成 st_c;

⑨ $ST[w]\leftarrow(rst,st_c)$;

⑩ else if $rst\not\subset ST[w].allrst$ then

/ * 搜索需要最新状态值 * /

⑪ 随机生成 st_c;

⑫ $ST[w]\leftarrow(allrst\cup rst,st_c)$;

⑬ end if

⑭ $(allrst,st_c)\leftarrow ST[w]$;

⑮ $\tau_w=(t_w,st_c)$.

在令牌模拟算法中,模拟器通过 $sp(w)$来判断查询是否针对同一个关键字,通过 $rp(w)$来维护字典 $allrst$,$allrst$ 中存储了每次查询发生时匹配关键字 w 的文件 ind.一旦 $rp(w)$中包含了新的 ind,模拟器 $\mathcal{S}$ 就会随机生成一个新的令牌,否则返回旧的令牌.同样地,对于以前从未出现在查询中的关键字,我们用一个随机值来替换伪随机函数 F 的输出.这个随机的值会被存储在一个字典 $KeyStore$ 中,当下一次再对此关键字做查询时,这个随机的值会被重用.需要注意的是,模拟的搜索令牌和真实的搜索令牌在大小和格式上是完全一样的,因此任何多项式时间的敌手 $\mathcal{A}$ 都无法进行区分.

混淆 4:和混淆 2 类似,模拟器 $\mathcal{S}$ 通过泄露函数 $\mathcal{L}^{Delete}(f)=\mathcal{L}'(id(f),op,\mu_f)$模拟删除令牌.

算法 4. 删除令牌模拟算法.

① $\mathcal{L}^{Delete}=$null;

② 从$Dict$ 中随机选取一个包含 μ_f 个关键字的文件 $id(f)$;

③ for $i=1$ to μ_f do

④ 从 $Dict$ 中取出$(id(f),(u,,e))$对;

⑤ 随机生成新的(u_{del},e_{del})对;

⑥ $\mathcal{L}^{Delete}=\mathcal{L}^{Delete}\cup(u_{del},e_{del})$;

⑦ 将字典 $Dict$ 的$(id(f),(u,,e))$替换为$(id(f),(u,v,u_{del},e_{del}))$;

⑧ end for

⑨$\tau_{del}=(\mathcal{L}^{Delete})$.

在删除令牌模拟算法中,模拟器 $\mathcal{S}$ 用字典 $Dict$ 来回答删除请求,同样地,模拟的删除令牌与实际的删除令牌具有相同的格式和大小.

通过以上混淆,我们构建了一个模拟器 $\mathcal{S}$ 模拟真实协议,任何多项式时间的敌手都无法区分真实协议和理想协议,证明了方案的安全性. 证毕.

8 性能分析

本节给出 EMFS 方案与 MFS 方案的性能对比.

为了更好地展现实验结果,我们首先给出方案的计算通信开销复杂度对比,如表 1 所示.其中 n_w 表示当搜索发生时匹配关键字 w 的文件个数,a_w 表示关键字 w 上更新次数之和.相比于 MFS 方案,我们方案的删除操作不需要遍历数据链,复杂度仅为 $O(1)$,但由于数据链中包含部分需要删除的块,因此搜索复杂度略微升高.

表 2 中给出了 2 个方案的性能对比,为了更简明地进行对比,我们主要考虑限门生成以及服务器查询链表所需的开销.其中,t_{BE}表示执行双线性配对(bilinear pairing)和指数运算(exponential operation)合计所需要的时间,t_{ma}表示执行异或运算(modular addition)所需要的时间,λ 表示安全参数,n 代表插入或删除文件中包含的关键字个数.可以观察到

我们的方案较 MFS 方案在计算和通信方面的开销都有所降低,这是由于我们通过伪随机置换来生成令牌,而 MFS 方案中生成令牌需要进行双线性和指数操作.此外,我们的方案为了减少信息的泄露,去除了部分不必要的数据传输,因此通信开销也优于 MFS 方案.

Table 1 Comparison of Search and Update Complexity Between MFS and EMFS

表 1 MFS 方案与 EMFS 方案的复杂度比较

Scheme	Computation			Communication	
	Search	*Add*	*Delete*	*Search*	*Add*/*Delete*
MFS[6]	$O(n_w)$	$O(1)$	$O(n_w)$	$O(n_w)$	$O(1)$
Our Scheme	$O(a_w)$	$O(1)$	$O(1)$	$O(n_w)$	$O(1)$

Table 2 Performance Comparison Between MFS and IMFS

表 2 MFS 方案与 EMFS 方案的性能比较

Scheme	Computation			Communication		
	Search	*Add*	*Delete*	*Search*	*Add*	*Delete*
MFS[6]	$O(n_w)\times t_{\mathrm{ma}}+t_{\mathrm{BE}}$	$O(1)\times t_{\mathrm{BE}}\times n$	$O(n_w)\times t_{\mathrm{BE}}\times n$	5λ	$8n\lambda$	$5n\lambda$
Our Scheme	$O(a_w)\times t_{\mathrm{ma}}$	$O(1)\times t_{\mathrm{ma}}\times n$	$O(1)\times t_{\mathrm{ma}}\times n$	3λ	$3n\lambda$	$3n\lambda$

我们在 Windows 7 操作系统上(单核的 Intel Core i5 4590 K 3.30 GHz CPU,内存 4 GB)进行了仿真实验,采用 Java 编程语言,并通过 jsCrypto 库实例化方案的加密操作.其中,伪随机函数的实现采用了 128 b HMAC-MD5,Hash 函数的实现用的是 160 b HMAC-SHA1,加密文档使用的是 256 b 密钥的对称加密算法 AES,关键词状态表则通过 Hash 表实现,以写入文件的方式进行存储,需要使用时再从文件中读取,加密索引的存储则采用了持久化的 Key-Value 数据库 Redis 以加快存取速度.

为了评估 2 个方案中用户端搜索关键字的效率,我们选取了一系列出现频率不同的关键词,将匹配数据集的大小从 10 增加到 10^5 分别进行搜索,并计算出检索匹配项所需的平均时间.如图 3 所示,当匹配文档数量增加时,平均搜索时间会随之降低,这是由于方案在搜索时需要执行一些一次性操作,譬如读取文件中存放的最新状态值和生成搜索令牌等等,这些操作耗费的时间会平均地分配到每个匹配结果项中.我们方案的搜索效率是优于 MFS 的,这是由于在 MFS 方案中生成搜索令牌需要通过双线性配对,耗时较大.而在我们的方案中搜索令牌则通过一个伪随机置换生成,搜索效率更高.

数据拥有者端在搜索关键字的效率如图 4 所示,需要注意的是,此时云服务器并不需要计算匹配项的文件密钥,开销相比于客户端大幅度降低.同样,

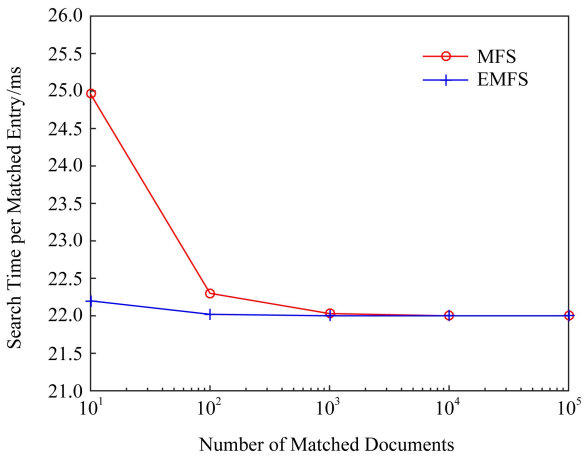

Fig. 3 Performance evaluation of search on client side

图 3 客户端搜索效率对比

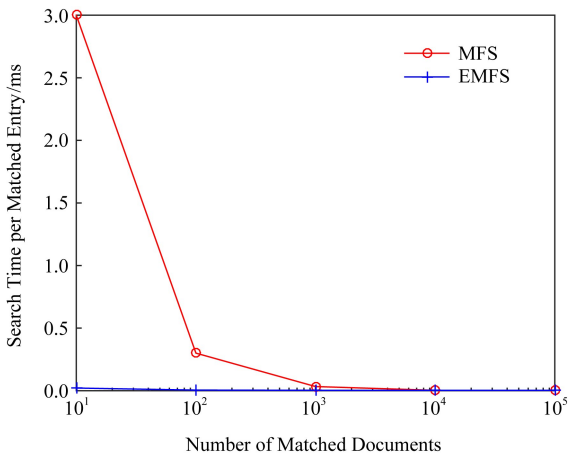

Fig. 4 Performance evaluation of search on data owner side

图 4 数据拥有者端搜索效率对比

我们的方案在用户端的搜索效率也是优于 MFS 方案的.

如图 5 所示,我们给出方案在云服务器端的删除效率对比.我们将删除的数据集的大小从 10^4 增加到 4×10^4,可以观察到随着文件数的增加,MFS 方案的响应时间以一种线性方式增加,这是因为 MFS 在每次删除时都需要遍历数次链表(遍历次数与当前文件包含的关键字数有关),而在 EMFS 方案中(我们考虑实际链表中块的删除,即删除操作与搜索绑定),我们认为删除最多需要遍历 N 条链表,其中 N 代表删除数据集中包含的关键字总数.

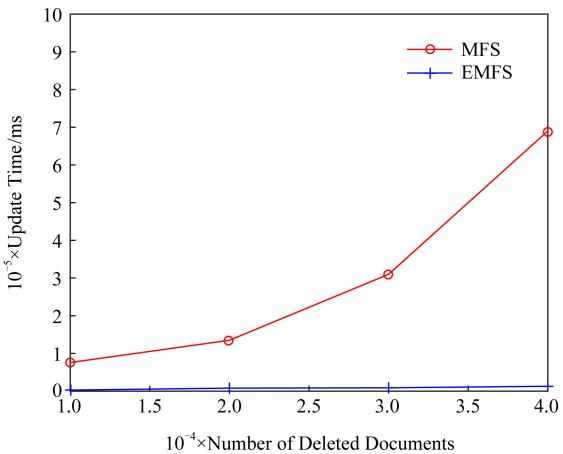

Fig. 5 Performance evaluation of delete on cloud server side

图 5 云服务器端删除算法效率对比

9 总 结

本文发现了 Wang 等人[6]提出的 MFS 方案存在的安全性问题,并针对这些存在问题提出了一个改进的方案 EMFS.通过避免关键字信息在数据拥有者和代理服务器之间的传递和增加用户验证机制,我们保证了在引入第三方代理服务器时,方案仍能保持前向安全性.此外,我们还给出了方案的安全分析和证明,表明我们的方案具有良好的安全性.最后,通过性能评估,证明我们的方案比 EMF 方案拥有更高的删除效率,在实际应用中效果更佳.

参 考 文 献

[1] 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

[2] Chang Yancheng, Mitzenmacher M. Privacy preserving keyword searches on remote encrypted data [C] //Proc of Int Conf on Applied Cryptography and Network Security. Berlin: Springer, 2005: 442-455

[3] Zhang Yupeng, 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 USENIX Security Symp (Security 16). Berkeley, CA: USENIX, 2016: 707-720

[4] Stefanov E, Papamanthou C, Shi E. Practical dynamic searchable encryption with small leakage [C] //Proc of NDSS. Reston, VA: Internet Society, 2014: 72-75

[5] Bost R. $\Sigma o\varphi os$: Forward secure searchable encryption [C] //Proc of the 2016 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2016: 1143-1154

[6] Wang Qiao, Guo Yu, Huang Hejiao, et al. Multi-user forward secure dynamic searchable symmetric encryption [C] //Proc of Int Conf on Network and System Security. Berlin: Springer, 2018: 125-140

[7] Dong Xiaolei, Zhou Jun, Cao Zhenfu. Research advances on secure searchable encryption [J]. Journal of Computer Research and Development, 2017, 54(10): 2107-2120 (in Chinese)
(董晓蕾, 周俊, 曹珍富. 可搜索加密研究进展[J]. 计算机研究与发展, 2017, 54(10): 2107-2120)

[8] Song D X, Wagner D, Perrig A. Practical techniques for searches on encrypted data [C] //Proc of 2000 IEEE Symp on Security and Privacy (S&P 2000). Piscataway, NJ: IEEE, 2000: 44-55

[9] Cash D, Jarecki S, Jutla C, et al. Highly-scalable searchable symmetric encryption with support for Boolean queries [C] //Proc of Annual Cryptology Conf. Berlin: Springer, 2013: 353-373

[10] Miao Meixia, Wang Jianfeng, Wen Sheng, et al. Publicly verifiable database scheme with efficient keyword search [J]. Information Sciences, 2019, 475: 18-28

[11] 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. New York: ACM, 2018: 745-762

[12] Ogata W, Kurosawa K. Efficient no-dictionary verifiable searchable symmetric encryption [C] //Proc of Int Conf on Financial Cryptography and Data Security. Berlin: Springer, 2017: 498-516

[13] Loh R, Zuo Cong, Liu J K, et al. A multi-client DSSE scheme supporting range queries [C] //Proc of Int Conf on Information Security and Cryptology. Berlin: Springer, 2018: 289-307

[14] Soleimanian A, Khazaei S. Publicly verifiable searchable symmetric encryption based on efficient cryptographic components [J]. Designs, Codes and Cryptography, 2019, 87(1): 123-147

[15] Sun Shifeng, Liu J K, Sakzad A, et al. An efficient non-interactive multi-client searchable encryption with support for Boolean queries [C] //Proc of European Symp on Research in Computer Security. Berlin: Springer, 2016: 154-172

[16] Cash D, Tessaro S. The locality of searchable symmetric encryption [C] //Proc of Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2014: 351-368

[17] Bost R, Minaud B, Ohrimenko O. Forward and backward private searchable encryption from constrained cryptographic primitives [C] //Proc of the 2017 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2017: 1465-1482

[18] Zhang Zhongjun, Wang Jianfeng, Wang Yunling, et al. Towards efficient verifiable forward secure searchable symmetric encryption [C] //Proc of European Symp on Research in Computer Security. Berlin: Springer, 2019: 304-321

[19] Zuo Cong, Sun Shifeng, Liu J K, et al. Dynamic searchable symmetric encryption schemes supporting range queries with forward (and backward) security [C] //Proc of European Symp on Research in Computer Security. Berlin: Springer, 2018: 228-246

[20] Hu Chengyu, Song Xiangfu, Liu Pengtao, et al. Forward secure conjunctive-keyword searchable encryption [J]. IEEE Access, 2019, 7(7): 35035-35048

[21] Wang Yunlin, Wang Jianfeng, Sun Shifeng, et al. Toward forward secure SSE supporting conjunctive keyword search [J]. IEEE Access, 2019, 7(7): 142762-142772

[22] Sun Shifeng, Yuan Xingliang, Liu J K, et al. Practical backward-secure searchable encryption from symmetric puncturable encryption [C] //Proc of the 2018 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2018: 763-780

[23] Zuo Cong, Sun Shifeng, Liu J K, et al. Dynamic searchable symmetric encryption with forward and stronger backward privacy [C] //Proc of European Symp on Research in Computer Security. Berlin: Springer, 2019: 283-303

[24] Chatterjee S, ParshuramPuria S K, Shah A. Efficient backward private searchable encryption [J]. Journal of Computer Security, 2020, 28(2): 229-267

[25] Guo Chen, Fu Xingbing, Mao Yaojun, et al. Multi-user searchable symmetric encryption with dynamic updates for cloud computing [J]. Information, 2018, 9(10): No.242

[26] Du Leilei, Li Kenli, Liu Qin, et al. Dynamic multi-client searchable symmetric encryption with support for Boolean queries [J]. Information Sciences, 2020, 506: 234-257

Lu Bingjie, born in 1996. PhD. Her main research interests include searchable encryption, cloud computing security and secure multiparty computation.

Zhou Jun, born in 1982. Received his PhD in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. Currently associate professor in East China Normal University. Member of IEEE and ACM. His main research interests include public key cryptography, secure multiparty computation, key theories for secure edge computing, privacy preserving, and applied cryptography in big data security, AI security, IoT security, 5G security and blockchain security.

Cao Zhenfu, born in 1962. PhD, distinguished professor in East China Normal University. Senior member of IEEE. His main research interests focus on number theory and new theories for cryptography and network security, including blockchain security, AI security, 5G security and privacy preserving.