计算机科学
COMPUTER SCIENCE

多云多副本的远程动态数据完整性检验方案

谈诗懿，王化群

引用本文

谈诗懿，王化群. 多云多副本的远程动态数据完整性检验方案[J]. 计算机科学，2025，52(5)：345-356.

TAN Shiyi, WANG Huaqun. Remote Dynamic Data Integrity Checking Scheme for Multi-cloud and Multi-replica [J]. Computer Science, 2025, 52(5): 345-356.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240300027

多云多副本的远程动态数据完整性检验方案

谈诗懿　王化群
南京邮电大学计算机学院　南京 210023
(294650575@qq.com)

摘　要　随着云存储服务的快速发展,越来越多的数据拥有者愿意将数据存储到云服务器中,从而减小自己在本地的存储负担。然而,一旦数据拥有者上传数据至云服务器,本地将不保存数据,数据拥有者将失去对数据的直接控制权。为了保证保存在云服务器上远程数据的完整性,数据完整性检验是必不可少的。它可以使得数据拥有者在不下载全部数据的情况下验证外包数据是否完整。为了提高外包数据的可用性和持久性,数据拥有者将多个副本存储在多个云服务器上。由于云服务器不是完全可信的,在公共云环境下保护数据拥有者的身份隐私是有必要的。当数据拥有者想要更改存储在云服务器上的数据文件时,数据动态操作如数据修改、数据删除、数据插入具有重要意义。因此,提出了在多云多副本环境下的远程动态数据完整性检验方案。该方案将环签名算法结合多云多副本环境,有效保护了数据拥有者的身份隐私,使得数据拥有者不用担心身份暴露问题。同时在多云环境下引入一种新的数据结构-分治邻接表实现数据动态操作,分治邻接表通过索引搜索指定数据并通过修改指针完成数据的插入和删除,相比其他数据结构如 Merkle 树等,提高了更新效率。基于标准困难问题,提出的方案是安全的。所提方案利用基于身份的公钥密码体制,消除了复杂的证书管理。通过性能分析和安全性分析,所提方案满足无条件匿名性、动态性和远程数据完整性验证。

关键词:云计算;可证明数据持有;动态数据;匿名性;多云服务器
中图分类号　TP309

Remote Dynamic Data Integrity Checking Scheme for Multi-cloud and Multi-replica

TAN Shiyi and WANG Huaqun
School of Computer Science,Nanjing University of Posts and Telecommunications,Nanjing 210023,China

Abstract　More and more data owners would like to store their data to cloud servers in order to reduce their local storage burden along with rapid development of cloud servers. However,data owners will lose the direct control over their data after uploading to cloud servers. Data integrity checking is essential to ensure the integrity of remote data stored on cloud servers. It allows data owners to verify the integrity of the outsourced data without downloading all the data. To improve the availability and durability of outsourced data,data owners store multiple copies on multiple cloud servers. It is necessary to protect data owners' identity privacy in public cloud environment because public cloud servers are not completely trustworthy. When data owners want to modify the data stored on the cloud servers,data dynamic operations such as data modification,data deletion,and data insertion are of great significance. Therefore,a remote dynamic data integrity checking scheme in a multi-cloud and multi-replica environment is proposed. The scheme combines the ring signature algorithm with a multi-cloud and multi-replica environment to effectively protect the privacy of data owners' identity,so that data owners do not have to worry about the problems due to identity exposure. At the same time,a new data structure,divide and conquer adjacency table,is introduced to implement dynamic operations of data in multi-cloud environment. The divide and conquer adjacency table searches the specified data through indexes and completes the insertion and deletion of data by modifying the pointers,which enhances updating efficiency compared to other data structures such as Merkle tree. The proposed scheme is secure based on the standard difficulty problem. This scheme makes use of identity-based public key cryptosystem and eliminates complex certificate management. Through performance and security analysis,the scheme satisfies unconditional anonymity,dynamics,and remote data integrity verification.

Keywords　Cloud computing,Provable data possession,Dynamic data,Anonymity,Multi-cloud servers

---

到稿日期:2024-03-04　返修日期:2024-07-21
基金项目:国家自然科学基金(U23B2002)
This work was supported by the National Natural Science Foundation of China(U23B2002).
通信作者:王化群(whq@njupt.edu.cn)

1 引言

随着云计算的迅速发展，数据拥有者越来越倾向于将大量数据存放在提供存储和计算服务的云平台上，如腾讯云、阿里云等。这种方式不仅使数据拥有者能够充分利用云平台丰富的软件、硬件和带宽资源，在任何时间和地点通过互联网访问其数据，还使得云平台通过向数据拥有者收取租金而获得经济效益。此外，与复杂的本地存储管理相比，云平台具有高可用性和低成本的优势。因此，将数据存储在云平台逐渐成为一种主流趋势。

然而，将数据存储在云平台上可能会带来严重的安全问题，其中一个主要问题是云服务器不是完全可信的，云服务器可能会在数据拥有者不知情的情况下删除或篡改不经常使用的数据，以节省存储空间。此外，一旦数据上传到云服务器上，数据拥有者会失去对数据的直接控制。因此，确保数据在云服务器上的完整性成为一个具有挑战性的问题[1-3]。

在此背景下，可证明数据持有(Provable Data Possession，PDP)成为一项重要的技术，它使得数据拥有者能够在不下载数据的情况下，远程验证云服务器上的数据是否被正确维护。为了确保结果的公正性，通常需要引入受云服务器和数据拥有者都信任的第三方审计机构(Third-party Auditor，TPA)进行数据完整性验证。TPA 通过向云服务器发起挑战来验证数据的完整性。在接收到挑战后，云服务器根据数据和相应的数据块标签计算完整性证明。如果证明通过了验证者的验证，则可以确定数据的完整性得到了保证。

一旦数据文件遭到破坏，数据拥有者会永久性地失去数据文件。为了提高数据的持久性和可用性，数据拥有者可以生成多个副本文件存储在云服务器上。如果一个副本文件被篡改或丢失，则数据拥有者可以从其他副本文件中恢复数据。为了进一步降低风险，数据拥有者将副本文件外包给不同的云服务器存储，即使一个云服务器上的所有副本丢失，数据拥有者也可以从其他云服务器上恢复数据。

大多数现有方案[4-6]只考虑单个文件的数据完整性，对于多个副本，它们必须运行 N 次才能验证通过 N 个副本的完整性，效率较低。为了提高验证效率，针对多副本的 PDP 方案[7-8]，可以通过一次挑战响应验证所有副本，但它们只支持所有副本存储在一个云服务器上的情况。此外，方案[7-8]是基于公钥基础设施(Public Key Infrastructure，PKI)技术的。

由于传统的 PKI 技术需要复杂的证书管理，因此与传统的 PKI 相比，基于身份的密码(Identity-based Cryptography，IBC)[9-10]提供了一种替代方案。IBC 使用用户唯一的身份作为公钥，无需证书即可验证公钥的有效性，从而避免繁琐的证书管理。匿名性被视为保护用户身份隐私的一项重要密码学技术。环签名[11-13]可以实现给定范围内的无条件匿名性，它是数字签名的一种形式，通过指定一组可能的签名者，使得验证者无法辨别真实签名者。

数据的动态性在云存储中十分重要。数据拥有者可以通过插入、修改或删除数据来更新外包数据，并且依然能够通过数据完整性检验来确保数据的完整性。因此，在多云多副本环境下设计一种基于身份的可保护用户身份隐私以及实现数据动态操作的 PDP 方案具有重要意义。

PDP 方案在 2007 年由 Ateniese 等[14]首次提出，正式形式化了 PDP 模型。该方案将所有数据文件分为若干块，通过随机抽样技术，可以高概率地得到整个数据文件的完整性状态。近年来，多种 PDP 方案不断被提出。为了支持数据的动态性，Ateniese 等[15]扩展了静态模型[14]。为了提高效率和灵活性，Sebe 等[16]和 Erway 等[17]分别提出了基于认证表和大整数分解的 PDP 方案。然而，这些方案都是通过私有验证实现的，只有数据拥有者才能够检查数据的完整性。为了实现公共验证，Wang 等[18]提出了一种公开的动态 PDP 方案，其优点在于任何人都可以挑战云服务器上存储数据的正确性。随后，Yang 等[19]提出了一种新的公共验证 PDP 方案，该方案支持索引表中的动态数据。

为了实现匿名性，Wang 等[20]提出了一种名为 Oruta 的群组远程数据完整性验证方案，该方案利用匿名技术对每个数据块进行签名，以保护用户的隐私。Oruta[20]基于 PKI 技术实现，会造成证书管理成本高昂。为了简化证书管理的复杂性，Yu 等[21]提出了一种基于身份的、保护数据隐私的公共 PDP 方案。Wang 等[22]提出了一种基于激励和无条件匿名的公共可证明数据完整性方案。Wang 等[22]只考虑了单云环境下用户隐私性的问题，并未考虑在多个云服务器环境下保护用户隐私性。

Zhu 等[23]提供了一种 PDP 方案，旨在协同验证存储在多个云服务器中的数据。为了提高效率，Wang 等[24]提出了一种应用于多云环境下的 PDP 方案。Zhu 等[23]和 Wang 等[24]考虑了多个云服务器的情况，但提出的方案不支持多副本。Curtmola 等[25]首次提出了一种用于云服务器中多个副本完整性检验的方案，但 Curtmola 等[25]提出的方案需要逐个检查副本，导致验证效率不高。随后，Hao 等[26]提出了一种具有隐私保护的多副本公共 PDP 方案。在这些多副本方案中，所有副本都存储在一个云服务器上。当数据副本分布在多个云服务器上时，它们无法解决数据完整性验证问题。

为了支持数据动态性，Barsoum 等[27]提出了一种基于映射表的动态多副本 PDP 方案，在查找操作上效率高，但在插入、删除操作时效率较低。Zhang 等[28]提出了一种基于 Merkle 哈希树的动态多副本 PDP 方案。Zhu 等[29]提出了一种基于索引哈希表的 PDP 方案。这些方案都是在单一云服务器下实现数据的动态性，并没有考虑在多个云服务器上数据的动态性问题。

Shen 等[30]和 Li 等[31]提出了一种面向多云和多副本的可证明数据持有方案，实现了所有副本在不同云服务器上可以一次性通过验证完整性检查。然而，文献[30]中的方案提出的分治表在插入、删除操作时效率仍然不高。文献[31]中的方案并不具备数据动态性，无法满足用户需要更新存储在云服务器上的数据的需求。本文在文献[22]和文献[31]的基础上进一步创新，提出了一种具备隐私保护和动态性的数据完整性检验方案。该方案利用基于身份的公钥密码学，将环签名与多云服务器相结合，有效保护了用户身份隐私，并在多云环境下引入了分治邻接表，使得所有副本数据可以实现同步更新，提高了更新效率。在多云多副本环境下，该方案可以

一次性验证不同云服务器上的所有副本的完整性，提高了所提方案的验证效率。另外，该方案采用随机抽样技术，有效保证了数据完整性验证，提高了计算效率和通信效率。具体而言，该方案利用环签名构造同态认证器，使得验证者能够在不检索整个数据集的情况下验证远程数据的完整性。无论是对于云服务器还是验证者，都无法获知数据拥有者的身份信息。此外，该方案还引入了分治邻接表来实现数据的动态性。数据拥有者可以对已上传的数据进行修改、删除、增加等动态操作，使得数据能够在多云多副本环境下保持数据的更新和灵活性。

2 预备知识

2.1 双线性对

设 G_1 和 G_2 是乘法循环群，G_1 的生成元为 g，阶为素数 q。假设在群 G_1 和 G_2 中离散对数问题是困难的，映射 e：$G_1\times G_1\rightarrow G_2$ 为双线性对，并满足以下特性。

1）双线性性：$\forall a,b\in Z_q^*$，以下公式成立：

$e(g^a,g^b)=e(g,g)^{ab}$

2）非退化性：$\exists g_1,g_2\in G_1$，满足 $e(g_1,g_2)\neq 1$。

3）可计算性：$\forall g_1,g_2\in G_1$，存在有效的算法来计算 $e(g_1,g_2)$。

定义 1（G_1 上的 CDH 困难问题） 给定三元组 $(g,g^a,g^b)\in G_1^3$，其中 a,b 未知，计算 $g^{ab}\in G_1$。

2.2 分治邻接表

分治邻接表（Divide and Conquer Adjacency Table，DCAT）如图 1 所示。DCAT 主要由两个数据结构组成：表头数组和链表。表头数组中的每个节点由 4 个部分构成：Lmin 表示对应链表的最小数据块索引号；Lmax 表示对应链表的最大数据块索引号；Rear 表示对应链表最后一个节点指针；First 表示对应链表的头指针。链表由 3 部分构成：LI_{ij} 表示数据块索引号；Time 表示数据块最近一次的更新时间；Next 表示下一节点的指针。

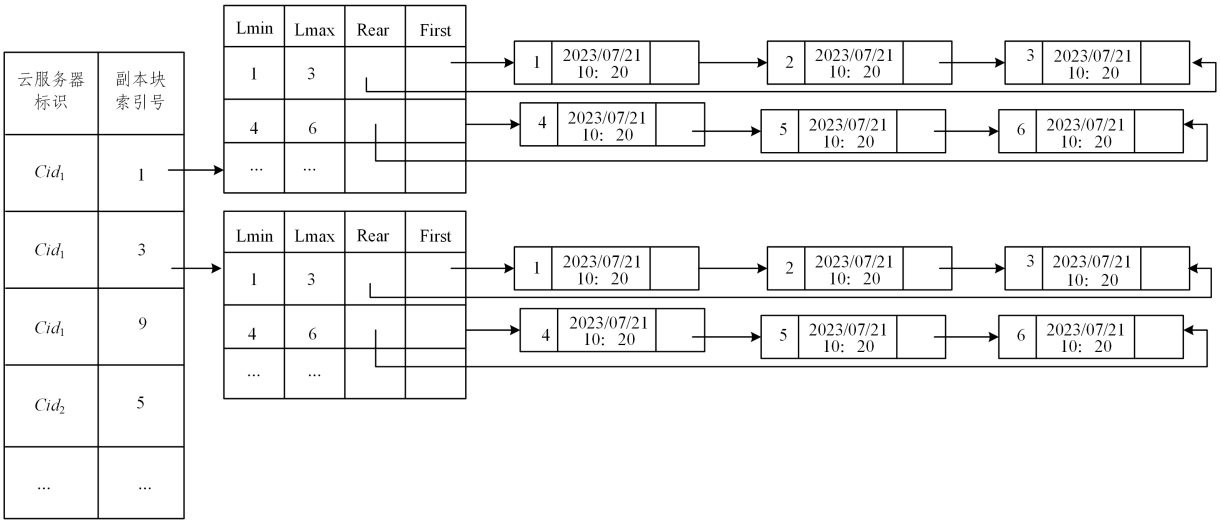

图 1 分治近邻表结构图

Fig. 1 Divide and conquer adjacency table structure

在本方案中有 N 个副本文件，每个副本文件都会构造一个分治邻接表，因此就有 N 个分治邻接表。数据拥有者在上传数据到云服务器之前，为每个副本文件生成对应的 DCAT 数据结构。当进行动态操作时，只需要对指针进行修改就可以实现数据块的插入、删除等操作。其中，分治表表示云服务与副本文件之间的关系，表头数组表示每个副本文件被分为 n 个数据块，这 n 个数据块根据数组中的 Lmin 以及 Lmax 进行链式存储。

2.3 基于身份的密码体制

为了降低公钥系统中密钥管理和使用的复杂性，Shamir 于 1984 年提出了基于身份的密码体制[9]。在这种系统中，用户的身份被作为用户的公钥。在这种情况下，用户不需要申请和交换证书，而是使用身份信息进行密码运算，从而解决了公钥真实性问题，简化了密钥系统管理的复杂性。系统中的用户私钥由系统中受信任的第三方（密钥生成中心）通过用户身份计算得出用户私钥。

2.4 环签名

环签名的概念最早由 Rivest 等[11]于 2001 年提出。使用环签名时，验证者确信签名是使用组成员的私钥之一计算的，但验证者无法确定是哪一个。具体地说，给定一个环签名和一组 d 个用户，验证者不能以大于 $1/d$ 的概率区分签名者的身份。这个特性可用于对验证者保留签名者的身份。

3 M2MD-PDP 系统模型和安全模型

3.1 系统模型

M2MD-PDP 的系统模型如图 2 所示。

1）数据拥有者（Data Owner，DO）与私钥生成器（Key Generation Center，KGC）交互并获得它的私钥；2）用户计算得出文件副本；3）用户将副本数据上传至云管理器（Cloud Organizer，CO），云管理器将副本分配给不同的云服务器（Cloud Storage，CSS）；4）第三方验证者（TPA）提出完整性挑战；5）云管理器将验证者发来的挑战发给对应的云服务器；6）云服务器分别计算保存在本地的副本验证值；7）云管理器聚合所有云服务器发来的验证值；8）第三方验证者验证数据完整性。

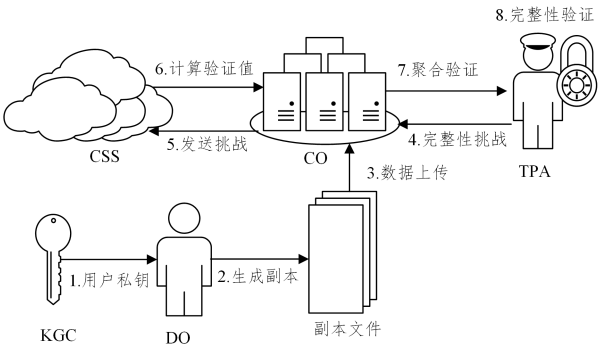

图 2 系统模型

Fig. 2 System model

M2MD-PDP 协议包括 5 个不同的实体，这 5 个不同的实体描述如下。

1)KGC：生成主私钥和主公钥，接收到用户身份后，KGC 输出该用户相应的私钥。

2)DO：数据的拥有者。数据拥有者租用云存储服务，将海量文件存储在多个云服务器上。它可以生成外包文件的不同副本并存储到不同的云服务器上。

3)CSS：具有大量的存储空间和计算资源，用于保存用户上传的文件数据以及生成挑战验证值。

4)TPA：将远程数据完整性挑战发送给云管理器，在收到云管理器的证明后可以对证明进行验证。DO 和 CSS 都相信 TPA 能诚实地执行验证工作。

5)CO：负责管理云服务器，在存储副本文件时，数据拥有者首先将副本发送给 CO，CO 根据用户的请求将不同的副本分发给对应的 CSS。当验证完整性时，TPA 首先向 CO 发送挑战请求，CO 将挑战分发给相应的 CSS，当从 CSS 收到所有的证明后，CO 将它们聚合为完整的证明并发送给 TPA。

系统参数和符号如表 1 所列。

表 1 系统参数和符号描述

Table 1 System parameters and symbol description

参数	含义
G_1	q 阶乘法循环群
G_2	q 阶乘法循环群
g	G_1 的生成元
Z_q^*	$\{1,2,\cdots,q-1\}$
H	$H:\{0,1\}^* \rightarrow G_1$
h	$h:\{0,1\}^* \rightarrow Z_q^*$
f	伪随机函数
π	伪随机排列
KGC	私钥生成器
ID_i	用户 i 的身份
N	文件副本总数
n	文件块总数
Δ	环成员总数
LI_{ij}	副本文件块的逻辑地址
$F_i=\{m_{ij}\},1\leqslant i\leqslant N,1\leqslant j\leqslant n$	m_{ij} 表示第 i 个副本的第 j 个块
(i,j,Fid)	i 表示文件的第 i 个副本，j 表示文件副本 F_i 的第 j 个块，Fid 表示文件的唯一标识
$\tau_{ij}=Fid \parallel LI_{ij} \parallel Time$	第 i 个副本的第 j 个块的动态数据
Cid_i	云服务器 CSS 的身份标识
FS_i	文件 Fid 对应副本号
CT_i	云服务器 CSS 上的副本 FS_i 的索引
(m_{ij},T_{ij})	第 i 个副本的第 j 个块的标签对
(c,k_1,k_2)	发出的挑战，c 表示挑战的块数，k_1 和 k_2 分别是 f 和 π 的随机键

3.2 M2MD-PDP 的定义

输入安全性参数 k，Setup 阶段输出系统参数 $params$；输入 $params$，ID，msk，mpk，Extract 阶段输出用户私钥 sk_{ID}；RepGen 阶段生成副本文件；TagGen 阶段生成块标签对；Challenge 阶段产生挑战值 $chal$；ProofGen 阶段每个云服务器生成副本的证明；ProofAggre 阶段云管理器聚合 ProofGen 阶段的证明；CheckProof 阶段 TPA 进行完整性检验；Data Dynamic 阶段用户提交更新请求并提交更新数据给云管理器，其中 Data Dynamic 分为 Insert，Modify，Delete 这 3 个阶段。

定义 2(M2MD-PDP) M2MD-PDP 协议是 11 种算法的集合，包括 Setup，Extract，RepGen，TagGen，Challenge，ProofGen，ProofAggre，CheckProof，Insert，Modify，Delete。这 11 种算法的描述如下：

$Setup(1^k)$：输入安全参数 k，输出系统参数 $params$，KGC 生成主私钥 msk 和主公钥 mpk。

$Extract(1^k,params,mpk,msk,ID)$：输入 $params$，mpk，msk 以及用户身份 ID，输出用户的私钥 sk_{ID}。

$RepGen(F,N)$：输入文件副本数 N 和文件 F，输出所有文件副本 $F_i=\{m_{ij}\},1\leqslant i\leqslant N,1\leqslant j\leqslant n$。

$TagGen(U_\Delta,sk_{ID},Cid_i,F_i)$：输入用户环($U_\Delta=(ID_1,ID_2,\cdots,ID_\Delta)$)、用户私钥 sk_{ID}、文件 F_i 要存储的 CSS 身份 Cid_i，输出数据块、标签对 (m_{ij},T_{ij})。

$Challenge(Fid)$：输入唯一文件名 Fid，输出 Fid 对应的 challenge 值。

$ProofGen(FS_i,TS_i,chal)$：每个云服务器根据 FS_i 生成完整性证明。输入文件副本集 FS_i、挑战集 $chal$，以及对应的标签集 TS_i，输出完整性证明 V_i。

$ProofAggre(V_i,1\leqslant i\leqslant L)$：云管理器聚合存储挑战数据的云服务器的所有完整性证明。输入 $V_i,1\leqslant i\leqslant L$，$V_i$ 是来自云服务器的完整性证明集合，输出最终的完整性证明 V。

$CheckProof$：由 TPA 根据云服务器生成的 V 验证远程数据完整性。它将输出成功或失败。

$Insert$：数据拥有者发送插入请求，将需要插入的数据块以及对应的标签发送给云管理器，云管理器验证数据合法性后将指令下发给云服务器。

$Modify$：数据拥有者发送修改请求，将需要修改的数据块以及对应的标签发送给云管理器，云管理器验证数据合法性后将指令下发给云服务器。

$Delete$：数据拥有者发送删除请求，将需要删除的数据块发送给云管理器，云管理器验证数据合法性后将指令下发给云服务器。

3.3 M2MD-PDP 的安全模型

协议应满足的安全需求如下。

1)正确性：数据文件的每个副本文件都应该由对应的云服务器完整存储，否则云服务器无法生成通过验证的证明。

2)隐私性：真实数据拥有者在整个验证过程中不会被验证者和云服务器发现其真实身份，数据文件内容不会被验证者知晓。

3)防伪造：在不使用指定副本文件数据块的情况下，任何人，包括云服务器都无法伪造验证证明来通过完整性验证。

4）防替换：云服务器无法用未被挑战的数据块替换被挑战的数据块来生成有效的证明。

5）动态性：用户在远程服务器上存储数据后，允许用户在不改变其他数据块的情况下执行块级别的动态操作，如块插入、块删除和块修改。同时，数据块更新后，块标签也需要更新。

基于以上安全目标，提出形式化的安全定义。

定义 3（不可伪造性） 如果对于任何（概率多项式时间）敌手 A（即 CSS），A 赢得 M2MD-PDP 游戏的概率可以忽略不计，则 M2MD-PDP 协议是不可伪造的。假设挑战者为 C，在挑战者 C 和敌手 A 之间 M2MD-PDP 的游戏如下。

Setup：C 执行 Setup 算法获得系统参数 $params$、主私钥 msk 和主公钥 mpk。C 将 msk 保密，并将 $params$ 发送给 A。将整个用户集设定为 $U_\Delta=(ID_1,ID_2,\cdots,ID_\Delta)$。

RepGen：C 执行 RepGen 获得原始文件的所有副本。

询问：A 自适应地向 C 提交一些随机预言机询问，如 Extract，Hash，TagGen。C 回应这些询问。C 与 A 之间的交互如下：

1）Extract 询问：A 发送身份 ID_i 来询问私钥。C 通过 Setup 阶段生成的系统参数 $params$ 计算私钥 sk_i，并响应给 A。

2）Hash 询问：A 自适应地选择向 C 提交此询问，C 返回 A 询问的哈希值。

3）TagGen 询问：A 自适应地选择任意副本的任意块 m_{ij}（它的索引为 i 和 j）、唯一文件名 Fid 以及 $U_i\subseteq U_\Delta$ 的用户环给 C 询问它的标签。C 计算对应的标签 T_{ij}。令 (U_i,i,j,m_{ij},T_{ij}) 是被查询的用户环、索引、块标记元组，设定 $UTF_1=(U_i,i,j,m_{ij},T_{ij})$。

Challenge：C 提交一个挑战验证值 $chal=(c,k_1,k_2)$ 给 A。$chal$ 确定有序索引集合 $\{v_1,v_2,\cdots,v_c\}$。假设这 c 个块标记是使用同一个用户环生成的，并且用户环不与 Extract 中的用户重合。A 提供块、标签对集合 $\{(m_{iv_1},T_{iv_1}),(m_{iv_2},T_{iv_2}),\cdots,(m_{iv_c},T_{iv_c})\}$ 的数据完整性证明。

ProofCheck：A 执行 ProofGen 和 ProofAggre，为文件块生成完整性证明。所有这些块都是 A 在 TagGen 阶段生成的。A 将证明发送给 C，C 执行验证算法来检查这些证明并将结果返回给 A。

OutPut：A 为文件块输出完整性证明 V。如果 V 通过完整性验证且不等于真实证明，则 A 获胜。

在定义 3 中，对于被挑战的用户环、块、标记三元组，恶意云服务器无法产生数据完整性证明。由于需要保护真实数据拥有者的隐私信息，M2MD-PDP 协议需要满足定义 4 中的无条件匿名性。

定义 4（匿名性） 给定任意身份集 $U_\Delta=(ID_1,ID_2,\cdots,ID_\Delta)$、消息 m 以及环签名 σ。一个敌手（不是用户环的成员）不能以高于 $\frac{1}{\Delta}$ 的概率辨别出数据拥有者的身份。如果敌手（不是数据拥有者）是用户环成员之一，那么他不能以高于 $\frac{1}{\Delta-1}$ 的概率辨别出数据拥有者的身份。

4 本文方案

本文提出了一种远程动态数据完整性检验方案。该方案将基于身份的环签名技术与多云多副本环境相结合，可以一次性通过同态验证数据完整性并且保护用户身份隐私，并引入数据结构 DCAT，高效地更新外包数据。该方案的具体算法包括系统建立算法 Setup、用户密钥生成算法 KeyGen、副本生成算法 RepGen、块标签生成算法 TagGen、挑战算法 Challeng、证明生成阶段 ProofGen、聚合阶段 ProofAggre 以及验证阶段 CheckProof。动态操作包括 Insert，Modify，Delete 这 3 个阶段。

假设数据拥有者计划将文件 F 外包给云服务器。根据文件大小，数据拥有者将文件分成 n 块，即 $F=\{m_1,m_2,\cdots,m_n\}$，m_j 表示文件的第 j 个块。

算法的详细描述如下。

$Setup(1^k)$：输入安全参数 k，KGC 选择两个阶为 q 的乘法循环群 G_1 和 G_2，令 g 为 G_1 的生成元，$e:G_1\times G_1\to G_2$ 是双线性映射。KGC 选择两个不同的哈希函数，定义如下：

$H:\{0,1\}^*\to G_1$

$h:\{0,1\}^*\to Z_q^*$

令 f 和 π 分别表示伪随机函数和伪随机排列，它们的定义如下：

$f:Z_q^*\times\{1,\cdots,n\}\to Z_q^*$

$\pi:Z_q^*\times\{1,\cdots,n\}\to\{1,\cdots,n\}$

KGC 随机选择主私钥 $x\in Z_q^*$，并公开主公钥 $Y=g^x$。KGC 公开系统参数 $params=(q,g,G_1,G_2,e,Y,h,H,f,\pi)$，

$Extract(1^k,params,Y,x,ID)$ 收到用户的身份 ID。KGC 按以下步骤生成用户的私钥。

1）KGC 随机选择 $r\in Z_q^*$，计算：

$R=g^r$，$d_{\mathrm{ID}}=r+xh(ID,R)\bmod(q-1)$

2）KGC 将 $sk_{\mathrm{ID}}=(R,d_{\mathrm{ID}})$ 通过安全信道发给用户 ID。

3）用户收到后，验证等式是否成立：$g^{d_{\mathrm{ID}}}=RY^{h(ID,R)}$。

4）如果等式成立，则将 R 公开，把 d_{ID} 作为用户私钥，否则拒绝并重新向 KGC 查询私钥。

$RepGen(F,N)$：为了预防云服务器之间可能存在的串通，将文件 F 复制为 N 个不同的副本文件。通过使用一种对称加密算法将每个块数据以及副本号一起加密，使得生成的每个块都不相同，从而得到 N 个不同的副本。例如：文件 $F=(F_1,F_2,\cdots,F_n)$，文件 F 被分为 n 个块。将每个块进行加密，即 $m_{ij}=E_k(i\parallel F_j)$。$E_k$ 表示一种对称加密算法，如 AES 或 SM4。因此，最终得到的文件副本为 $F_i=\{m_{ij}\}$，$1\leqslant i\leqslant N$，$1\leqslant j\leqslant n$，并计算每个副本文件块的逻辑地址 LI_{ij}。表 2 为记录副本地址索引的记录表。

表 2 副本地址索引的记录

Table 2 Record of replica address index

云服务器身份标识	副本块索引号
Cid_1	1，3，9
Cid_2	2，4
Cid_3	5
…	…

$TagGen(U_\Delta, sk_{ID_s}, Cid_i, F_i)$：用户环 $U_\Delta=(ID_1, ID_2, \cdots, ID_\Delta)$，文件副本 $F_i=\{m_{ij}\}$，$1\leqslant i\leqslant N$，$1\leqslant j\leqslant n$，真实数据拥有者为 U_s，U_s 的计算过程如下。

1）对于所有 $t\neq s$ 的用户：U_s 随机选择 $a_{ij,t}\in Z_q^*$，$t=\{1, 2, \cdots, \Delta\}$，并计算 $\bar{\sigma}_{ij,t}=g^{a_{ij,t}}$。

2）U_s 计算 $\tau_{ij}=Fid\parallel LI_{ij}\parallel Time$，$\bar{\sigma}_{ij,s}=\left(\dfrac{H(Cid_i\parallel i\parallel j\parallel \tau_{ij}\parallel U_\Delta)g^{m_{ij}}}{\prod\limits_{t\neq s}(R_tY^{h(ID_t,R_t)})^{a_{ij,t}}}\right)^{1/d_{ID_s}}$。

生成的标签为 $T_{ij}=(\bar{\sigma}_{ij,1}, \bar{\sigma}_{ij,2}, \cdots, \bar{\sigma}_{ij,\Delta})$，其中 Cid_i 为储存第 i 个副本的 CSS 的标识。

$\tau_{ij}=Fid\parallel LI_{ij}\parallel Time$，$Fid$ 是文件的唯一标识符，LI_{ij} 是副本块的索引号，$Time$ 为创建块的时间或者最近一次更新块的时间。初始 $Time$ 均相同。用户将 $\{\tau_{ij}\}$，$1\leqslant i\leqslant N$，$1\leqslant j\leqslant n$ 发送给 TPA。

真实数据拥有者重复以上标签生成 $N*n$ 次，确保所有的副本上的块都有对应的块标签，则数据块、标签对可表示为 (m_{ij}, T_{ij})。

为了使得 TPA 验证时确保是环中某一成员所拥有的文件所进行的完整性检验而不是恶意敌手提出的请求，因此需要生成一个签名在完整性检验前进行验证。可使用任意一个签名算法如 BLS 签名算法进行签名：$T_{Fid}=sig_{sk_\Delta}(F)$。

最后，将生成的块、标签对、文件签名 $\{(m_{ij}, T_{ij}), T_{Fid}\}$ 以及记录表发送给 CO。CO 根据记录表将文件副本以及标签发送给对应的云服务器 CSS。另外，每一个标签都可由如下等式验证：

$$\prod_{t=1}^{\Delta}e(R_tY^{h(ID_t,R_t)}, \bar{\sigma}_{ij,t})=e(H(Cid\parallel i\parallel j\parallel \tau_{ij}\parallel U_\Delta)\cdot g^{m_{ij}}, g)$$

$Challenge(Fid)$：为了验证所有副本名称为 Fid 的文件，TPA 向 CO 发送 $chal=(c, k_1, k_2)$ 以及需要挑战的文件标识 Fid，CO 收到后搜索记录表并向至少含有一个副本的 CSS 发送 $chal$，假设有 L 个 CSS，那么 CO 会将 $chal$ 分别发送给 L 个 CSS。

$ProofGen(FS_i, TS_i, chal)$：身份为 Cid_i 的 CSS 收到 CO 发送过来的 $chal=(c, k_1, k_2)$，开始计算验证标签。如果在 Cid_i 上的副本 FS_i 的索引值为 CT_i，那么 $\sum\limits_{\ell=1}^{L}|CT_\ell|=N$ 显然成立。CSS 的计算过程如下：

for $t=1, 2, \cdots, \Delta$：

1）for $1\leqslant j\leqslant c$：$v_j=\pi_{k_1}(j)$，$a_j=f_{k_2}(j)$

2）$\forall\beta\in CT_i$，$\bar{\sigma}_{\beta,t}=\prod\limits_{j=1}^{c}\bar{\sigma}_{\beta v_j,t}^{a_j}$，$M_\beta'=\sum\limits_{j=1}^{c}a_jm_{\beta v_j}$

3）$\bar{\sigma}_{i,t}=\prod\limits_{\beta\in CT_i}\bar{\sigma}_{\beta,t}$，$M_i=\sum\limits_{\beta\in CT_i}M_\beta'$

则生成的证明为 $V_i=\{T_i, M_i\}$。

$ProofAggre(V_i, 1\leqslant i\leqslant L)$：CO 从 L 个 CSS 上收到所有证明 V_i，CO 进行聚合 V_i，$1\leqslant i\leqslant L$。证明：for $t=1, 2, \cdots, \Delta$：$\sigma_t=\prod\limits_{i=1}^{L}\bar{\sigma}_{i,t}$，$M=\sum\limits_{i=1}^{L}M_i$。

CO 生成聚合证明 $V=\{T, M, T_{\text{Fid}}\}$ 发送给 TPA。

$CheckProof$：收到来自 CO 的证明 V，以及根据 $chal$ 和 $\{Cid_i\}$，$1\leqslant i\leqslant L$，TPA 开始验证：

1）TPA 首先验证 T_{Fid} 是否是 $U_\Delta=(ID_1, ID_2, \cdots, ID_\Delta)$ 对文件 F 的一个有效签名，若成立则往下执行，否则立即停止。

2）for $1\leqslant j\leqslant c$，$c_j=\pi_{k_1}(j)$，$a_j=f_{k_2}(j)$。

3）TPA 计算 $h_{ij}=H(Cid_i\parallel i\parallel c_j\parallel \tau_{ij}\parallel U_\Delta)$。

TPA 验证等式是否成立：

$$\prod_{t=1}^{\Delta}e(R_tY^{h(ID_t,R_t)}, \sigma_t)=e(\prod_{i=1}^{N}\prod_{j=1}^{c}h_{ij}^{a_j}\cdot g^M, g)$$

动态操作（$Insert$，$Modify$，$Delete$）：数据拥有者 DO 在将数据文件上传至 CO 之前，为每个文件生成对应的 DCAT 数据结构。为了实现数据动态更新，设计了数据修改 Modify、数据插入 Insert 与数据删除 Delete 3 个数据更新算法。

1）数据修改算法 Modify 的执行过程如下：

（1）根据要修改的数据块编号 j，首先查找表头节点，若 $L_{\min}\leqslant j\leqslant L_{\max}$，则根据 First 头指针遍历找到对应节点 Node。

（2）修改节点 Nod_j 的 $Time=Time'$，其中 $Time'$ 为当前时间。

（3）DO 运行 TagGen 算法生成新的标签 $T_{ij}'=(\bar{\sigma}_{ij,1}', \bar{\sigma}_{ij,2}', \cdots, \bar{\sigma}_{ij,\Delta}')$，其中对于所有 $t\neq s$ 的用户，随机选择 $a_{ij,t}'\in Z_q^*$，$t=\{1, 2, \cdots, \Delta\}$，并计算 $\bar{\sigma}_{ij,t}'=g^{a_{ij,t}'}$。

（4）U_s 计算 $\tau_{ij}=Fid\parallel LI_{ij}\parallel Time'$，$\bar{\sigma}_{ij,s}'=\left(\dfrac{H(Cid\parallel i\parallel j\parallel \tau_{ij}\parallel U_\Delta)g^{m_{ij}^*}}{\prod\limits_{t\neq s}(R_tY^{h(ID_t,R_t)})^{a_{ij,t}'}}\right)^{1/d_{ID_s}}$。

（5）DO 将 $\{modify, Fid, m_{ij}^*, T_{ij}'\}$ 发送给 CO。CO 将要修改的文件以及索引号发送给记录表中的 CSS，CSS 根据索引号更新文件和标签，同时修改分治邻表上的 $Time$ 值。

假设 DO 需要修改的数据块为 F_2，图 3 是初始文件上传后生成的 DCAT 结构图，图 4 是修改数据块 F_2 后更新完成后的结构图。

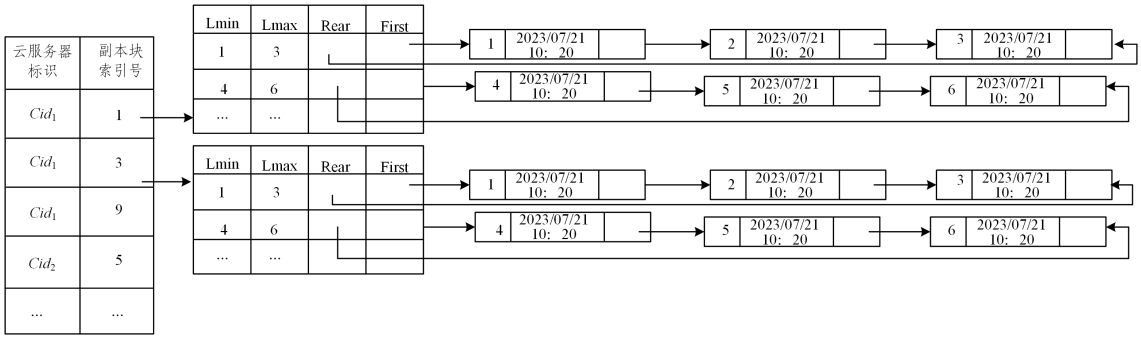

图 3 修改数据块前的分治邻表

Fig. 3 Divide and conquer adjacency table before modifing the data block

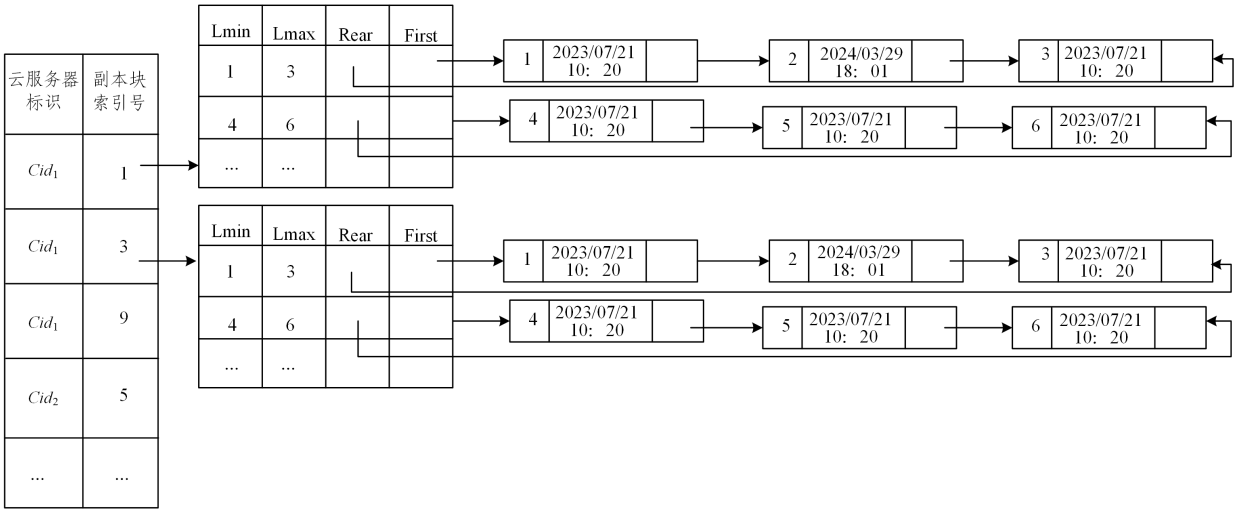

图 4 修改数据块后的分治邻表

Fig. 4 Divide and conquer adjacency table after modifing the data block

2)数据插入算法 Insert 的执行过程如下：

(1)根据 F_j 的数据块编号查找表头点，若满足 $L_{\min} \leqslant j \leqslant L_{\max}$，则在对应的链表找到对应的第 j 个数据块节点。

(2)创建一个新的链表节点，即 $LI_{ij}=MAX+1$，$Time=Time'$，然后设置 $Next_{j+1}=Next_j$，同时 $Next_j$ 指向新的插入节点。

(3)将当前表头节点的 $L_{\max}$ 和随后的表头节点 $L_{\min}$ 和 $L_{\max}$ 均加 1。

(4)DO 运行 TagGen 算法生成新的标签 $T'_{ij}=(\bar{\sigma}'_{ij,1}, \bar{\sigma}'_{ij,2}, \cdots, \bar{\sigma}'_{ij,\Delta})$，其中对于所有 $t \neq s$ 的用户，随机选择 $a'_{ij,t} \in Z_q^*$，$t=\{1,2,\cdots,\Delta\}$，并计算 $\bar{\sigma}'_{ij,t}=g^{a'_{ij,t}}$。

(5) U_s 计算 $\tau_{ij}=Fid \parallel LI_{ij} \parallel Time'$，$\bar{\sigma}'_{ij,s}=\left(\frac{H(Cid \parallel i \parallel j \parallel \tau_{ij} \parallel U_\Delta) g^{m_{ij}^*}}{\prod_{t \neq s}(R_t Y^{h(ID_t, R_t)})^{a'_{ij,t}}}\right)^{1/d_{ID_s}}$。

(6)DO 将 $\{Insert, Fid, m_{ij}^*, T'_{ij}\}$ 发送给 CO。CO 将要插入的文件以及索引号发送给记录表中的 CSS，CSS 根据索引号插入文件以及记录文件标签。

假如 DO 需要在数据块 F_2 后面插入一个块，图 5 是插入数据块 F_2 后更新完成后的结构图。

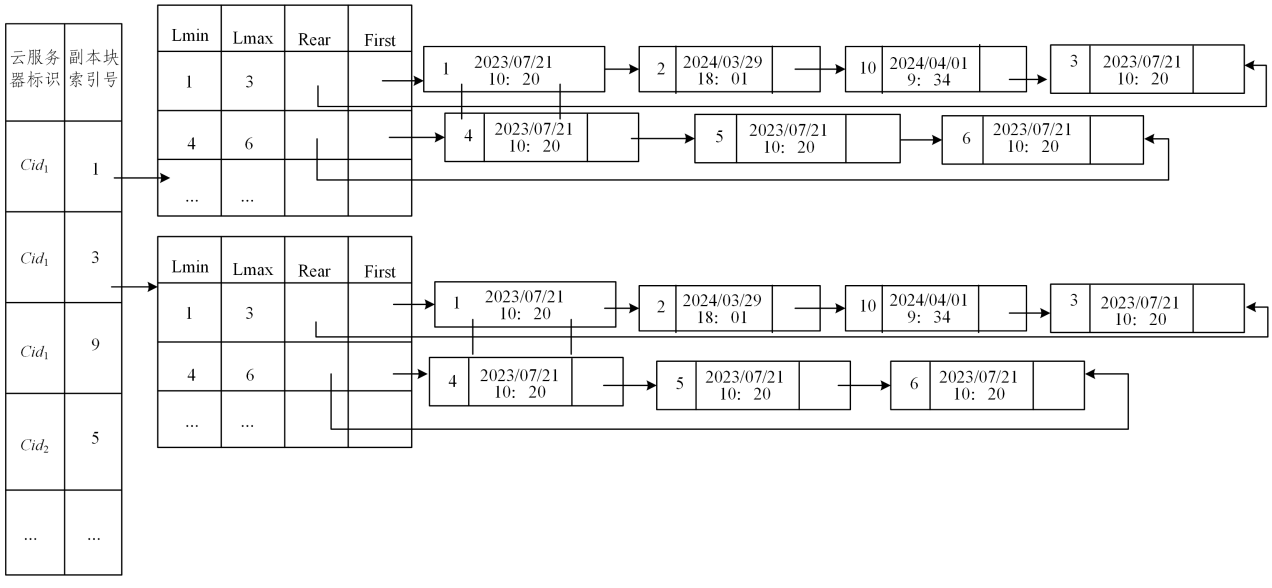

图 5 插入数据块后的分治邻表

Fig. 5 Divide and conquer adjacency table after inserting the data block

3)数据删除算法 Delete 的执行过程如下：

(1) 根据 F_j 的数据块编号查找表头节点，若满足 $L_{\min} \leqslant j \leqslant L_{\max}$，则在对应的链表找到对应的第 $j-1$ 个数据块节点。

(2) 若第 j 个数据块对应节点为链表第一个节点，则将表头节点域 $First=Next_j$。若第 j 个数据块对应节点为链表最后一个节点，则直接将表头节点域 $Rear$ 指向第 $j-1$ 个节点。若第 j 个节点既不是链表对应第一个节点也不是最后一个节点，则将第 $j-1$ 个节点对应的节点中 $Next_{j-1}=Next_j$、相应表头节点 $L_{\max}$，以及随后表头节点中的 $L_{\min}$ 和 $L_{\max}$ 均减 1。

(3) DO 将 $\{Delete, Fid, m_{ij}\}$ 发送给 CO，CO 将要删除的文件以及其索引号发送给记录表中的 CSS，CSS 根据索引号删除文件以及文件标签。

假设 DO 需要删除的数据块为 F_2，图 6 是删除数据块 F_2 后更新完成后的结构图。

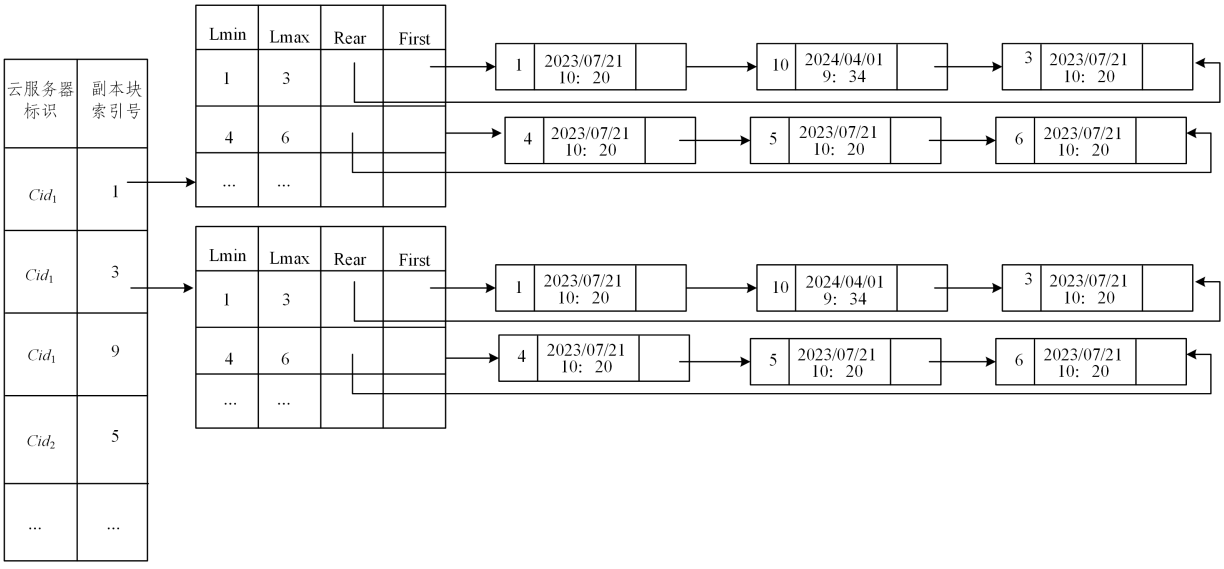

图 6 删除数据块后的分治邻表

Fig. 6 Divide and conquer adjacency table after deleting the data block

5 安全性分析

5.1 正确性分析

一个有效的 M2MD-PDP 协议必须满足正确性。

定理 1(正确性) 如果 KGC,CO,CSS,DO 和 TPA 是诚信的并且遵循协议的步骤,那么证明 V 可以通过验证者的检验。

证明:M2MD-PDP 协议的正确性来源于以下推导:

$$\begin{aligned}
&\prod_{t=1}^{\Delta} e(R_t Y^{h(ID_t,R_t)},\sigma_t)\\
&=\prod_{t=1}^{\Delta} e(R_t Y^{h(ID_t,R_t)},\prod_{\ell=1}^{L}\bar{\sigma}_{\ell,t})\\
&=\prod_{t=1}^{\Delta} e(R_t Y^{h(ID_t,R_t)},\prod_{\ell=1}^{L}\prod_{\beta\in CT_\ell}\bar{\sigma}_{\beta,t})\\
&=\prod_{t=1}^{\Delta} e(R_t Y^{h(ID_t,R_t)},\prod_{i=1}^{N}\bar{\sigma}_{i,t})\\
&=\prod_{t=1}^{\Delta} e(R_t Y^{h(ID_t,R_t)},\prod_{i=1}^{N}\prod_{j=1}^{c}\bar{\sigma}_{iv_j,t}^{a_j})\\
&=\prod_{t=1}^{\Delta}\prod_{i=1}^{N}\prod_{j=1}^{c} e(R_t Y^{h(ID_t,R_t)},\bar{\sigma}_{iv_j,t}^{a_j})\\
&=\prod_{i=1}^{N}\prod_{j=1}^{c}\prod_{t=1}^{\Delta} e(R_t Y^{h(ID_t,R_t)},\bar{\sigma}_{iv_j,t})^{a_j}\\
&=\prod_{i=1}^{N}\prod_{j=1}^{c} e\left(h_{ij} g^{m_{iv_j}},g\right)^{a_j}\\
&=e\left(\prod_{i=1}^{N}\prod_{j=1}^{c} h_{ij}^{a_j} g^{a_j m_{iv_j}},g\right)\\
&=e\left(\prod_{i=1}^{N}\prod_{j=1}^{c} h_{ij}^{a_j}\cdot\prod_{i=1}^{N} g^{\sum_{j=1}^{c} a_j m_{iv_j}},g\right)\\
&=e\left(\prod_{i=1}^{N}\prod_{j=1}^{c} h_{ij}^{a_j}\cdot g^{M},g\right)
\end{aligned}$$

5.2 匿名性

定理 2(匿名性) M2MD-PDP 协议满足无条件匿名性。

证明:从以下两种情况证明定理 2:对于每一个云服务器,它们不能从标签中识别出真正的数据拥有者;验证者无法从 ProofGen 阶段生成的响应中识别出真正的数据拥有者。

1)对于每一个块、标签对(m_{ij},T_{ij}),其中 $m_{ij}=E_k(i\|F_j)$,不包含任何关于 U_s 的信息。对于 $T_{ij}=(\bar{\sigma}_{ij,1},\bar{\sigma}_{ij,2},\cdots,\bar{\sigma}_{ij,\Delta})$,$\bar{\sigma}_{ij,t}$,$t\neq s$ 是随机选取的,不能从中获取任何关于 U_s 的信息。由于 $\bar{\sigma}_{ij,s}=\left(\dfrac{H(Cid\|i\|j\|\tau_{ij}\|U_\Delta)g^{m_{ij}}}{\prod_{t\neq s}(R_t Y^{h(ID_t,R_t)})^{a_{ij,t}}}\right)^{1/d_{ID_s}}$,$H(Cid\|i\|c_j\|\tau_{ij}\|U_\Delta)$是 H 的一个哈希值,在随机 oracle 模型中,很难从中获取关于 U_s 的信息。因此,云服务器无法从块标签中辨认出真实数据拥有者。

2)由于 TPA 收到的验证值为(T,M,Fid),其中 $T=(\sigma_1,\sigma_2,\cdots,\sigma_\Delta)$。从 1)中得出每个用户的签名 $\sigma_{ij,t}$,$i=1,2,\cdots,N$,$j=c_1,c_2,\cdots,c_c$,$1\leqslant t\leqslant\Delta$,都不包含任何真实数据拥有者的信息。此外,$(M,Fid)$与真实数据拥有者毫无关系。因此,验证者无法从 ProofGen 阶段生成的回应辨认出真实数据拥有者的身份。

综上,此方案具有匿名性。

5.3 不可伪造性

定理 3 (G_1,G_2)是(t',ε')-q 阶 GDH 群对。TagGen 阶段为$(t_A,\varepsilon,q_h,q_H,q_E,q_t)$,在自适应选择块攻击下是安全的即不存在伪造,对于所有的 t 和 ε 都满足 $\varepsilon'>\dfrac{\varepsilon}{e(q_t+1)}$以及 $t'\leqslant t_A+O(q_h)+O(q_H)+O(q_E)+O(q_t)$,其中 $e=2.718$,ε 表示 A 在时间 t_A 下赢得游戏的概率。基于 CDH 困难问题假设,本文方案的 TagGen 阶段是不可伪造的。

证明:假设索引、块对$\{(1,m_{i1}),(2,m_{i2}),\cdots(n,m_{in})\}$被处理并上传到云服务器上。假设用户环为 $U_\Delta=(ID_1,ID_2,\cdots,ID_\Delta)$。我们在挑战者 B 和敌手 A 之间进行一个博弈游戏。通过使用 A 的伪造作为子程序,B 可以打破 CDH 问题。基于 CDH 困难问题,所提方案被证明是安全的。输入$(g^a,g,g^b)\in G_1$,目标是输出 $g^b\in G_1$。B 与 A 的交互如下。

Setup:B 选择 $Y=g^a$ 作为主公钥,同时保持 a 的秘密性。

Extract:输入用户的身份 ID_t,B 作如下询问:

1)如果 $ID_t\notin U_\Delta$,表示 $c=0$。B 随机选择 $x_t,h_t\in Z_q^*$,令 $R_t=Y^{x_t-h_t}$。接着 B 存储记录(c,ID_t,R_t,x_t,h_t)在表 $Table_h$ 中,并且 B 拒绝给 A 回应。

注意：因为 $R_t=Y^{x_t-h_t}$，可以知道 $R_tY^{h_t}=Y^{x_t}=g^{ax_t}$，用 $\sigma_t=ax_t$ 表示。由于 a 是未知的，因此 σ_t 也是未知的。尽管 B 拒绝回应不在 U_Δ 中的用户的询问，但 B 依然可以为 U_Δ 的所有用户产生以上的记录并存储在 $Table_h$ 中。

2）如果 $ID_t\in U_\Delta$，表示 $c=1$。B 随机选择 $x_t,h_t\in Z_q^*$，接着计算 $R_t=g^{x_t}Y^{-h_t}$，B 存储记录 (c,ID_t,R_t,x_t,h_t) 在表 $Table_h$ 中。

H-Oracle：当 A 向 H-Oracle 提交询问 (ID_t,R_t)，B 在 $Table_h$ 中查找记录 $(*,*,R_t,x_t,*)$。如果在 $Table_h$ 中存在记录 $(*,ID_t,R_t,x_t,h_t)$，B 就把 h_t 回应给 A。否则，B 执行 Extract 阶段并获得记录 $(*,ID_t,R_t,x_t,h_t)$，再把 h_t 回应给 A。

H-Oracle：输入 $(Fid,Cid_i,\tau_{ij},i,j)$，B 开始掷硬币，以 μ 的概率掷 0 和以 $1-\mu$ 的概率掷 1。接着 B 随机选择一个 $r_t\in Z_q^*$。如果硬币显示为 0，B 就返回 $(g^{ab})^{r_t}g^{-m_{ij}}$ 给 A，否则返回 $(g^a)^{r_t}g^{-m_{ij}}$。由于 r_t 是在 Z_q^* 上随机选择的，$(g^{ab})^{r_t}g^{-m_{ij}}$ 和 $(g^a)^{r_t}g^{-m_{ij}}$ 有相同的分布，因此 A 不能从哈希查询的结果中区分抛硬币的结果。B 在表 $Table_h$ 中存储记录 $(U_\Delta,i,j,Fid,Cid_i,\tau_{ij},coin,r_t)$。

RepGen：B 执行算法 RepGen 获取原始文件的所有副本。B 随机设置这些副本的存储位置，并将信息提供给 A。

TagGen queries：在任何时候，A 都可以查询到任何副本中任何身份生成的块标签。当 A 提交 $(Fid,Cid_i,\tau_{ij},i,j,U_\Delta)$ 向 B 询问 m_{ij} 的块标签，由于 H 是个随机预言机，$(Fid,Cid_i,\tau_{ij},i,j)$ 已经向 B 询问过了，如果 $c=0$，那么 B 失败并退出；否则，B 返回 $H(Cid\|i\|j\|\tau_{ij}\|U_\Delta)g^{m_{ij}}=(g^a)^{r_t}$。对于在 U_Δ 中的用户 ID_t，B 在 $Table_h$ 中获得对应的 x_t。最后，B 获得所有的集 $\{x_1,x_2,\cdots,x_\Delta\}$。在本方案中，B 随机选择 $a_2,\cdots,a_\Delta$，计算 $a_1=x_1^{-1}(r_t-\sum_{t=2}^{\Delta}a_tx_t)$，其中 $r_t=\sum_{t=1}^{\Delta}a_tx_t$。因此返回标签 $T_{ij}=(g^{a_1},\cdots,g^{a_\Delta})$。

OutPut：最终，A 输出一个有效伪造 $T=(\sigma_1,\cdots,\sigma_\Delta)$。如果硬币由 B 抛到 1，B 失败；否则，有：

$$e(H(Cid\|i\|j\|\tau_{ij}\|U_\Delta)g^{m_{ij}},g)=\prod_{t=1}^{\Delta}e(R_tY^{h(ID_t,R_t)},\sigma_t)$$

$$e(H(Cid\|i\|j\|\tau_{ij}\|U_\Delta)g^{m_{ij}},g)=$$

$$\prod_{t=1}^{\Delta}e(Y^{x_t},\sigma_t)e((g^{ab})^{r_t},g)=e(g,\prod_{t=1}^{\Delta}\sigma_t^{ax_t})$$

计算可得：

$$(g^{ab})^{r_t}=\prod_{t=1}^{\Delta}\sigma_t^{ax_t}$$

$$g^b=(\prod_{t=1}^{\Delta}\sigma_t^{x_t})^{r_t^{-1}}$$

可能性分析如下：上述模拟游戏中，如果游戏没有失败，B 将输出 g^b 并打破 CDH 问题，因此可以得出：

$$\begin{aligned}P_r(B\rightarrow g^b)&=Pr(Awins)Pr(coin=1)^{q_t}Pr(coin=0)\\&=\varepsilon(1-\mu)^{q_t}\mu\\&=\varepsilon\left(1-\frac{1}{q_t+1}\right)^{q_t}\frac{1}{q_t+1}>\frac{\varepsilon}{e(q_t+1)}\end{aligned}$$

在上面的模拟中，假设 A 在 Extract 阶段发出 q_E 询问，在 H-Oracle 发出 q_H 询问，在 TagGen-oracle 发出 q_t 询问，那么总共时间花费为 $t_A+O(q_h)+O(q_H)+O(q_E)+O(q_t)$，有 $t'\leqslant t_A+O(q_h)+O(q_H)+O(q_E)+O(q_t)$。

定理 3 指出不可信的云服务器可以伪造出单个标签的概率可以忽略不计。定理 4 将证明不可信的云服务器可以伪造响应的概率也是可以忽略不计的。

定理 4 如果哈希函数是抗碰撞的，并且所选择的签名算法是安全的，那么在没有真实数据的情况下伪造完整证明通过验证的概率可以忽略不计，即本文提出的方案在随机 oracle 模型中是不可伪造的。

证明：由完整证明 $V=\{T,M,T_{Fid}\}$（其中 $T=(\sigma_1,\cdots,\sigma_\Delta)$）可以看到，如果签名算法是安全的，那么 T_{Fid} 是不可被伪造的，可以被伪造的只能是 (T,M)，$T=(\sigma_1,\cdots,\sigma_\Delta)$。$chal=(c,k_1,k_2)$ 是挑战信息，通过计算 $v_j=\pi_{k_1}(j)$，$a_j=f_{k_2}(j)$ 得到被挑战块的索引和参数。

假设伪造的聚合证明为 $V'=(T',M')$，由于伪造的块可以通过验证，则有：

$$e(\prod_{i=1}^{N}\prod_{j=1}^{c}h_{ij}^{a_j}\cdot g^{M'},g)=\prod_{t=1}^{\Delta}e(R_tY^{h(ID_t,R_t)},\sigma_t')$$

其中，$h_{ij}=H(Cid\|i\|j\|\tau_{ij}\|U_\Delta)$。

假设真正的挑战值 $chal=(c,k_1,k_2)$ 对应的真实证明为 $V=\{T,M\}$，可以得到：

$$e(\prod_{i=1}^{N}\prod_{j=1}^{c}h_{ij}^{a_j}\cdot g^{M},g)=\prod_{t=1}^{\Delta}e(R_tY^{h(ID_t,R_t)},\sigma_t)$$

对比两个等式，很容易得到如果 $M=M'$，那么每一个 $\sigma_t=\sigma_t'$，这与提出的假设相反。因此，至少有一个 $M\neq M'$。在本方案中，根据定理 3，由于单个块不能被伪造，因此必须使得每一个 $\sigma_t=\sigma_t'$；否则，只需要挑战一个块就能得到伪造块标签。为了使得两个等式成立，必须有每一个 $\sigma_t=\sigma_t'$ 且至少有一个 $M\neq M'$。对比以上两个式子，发现 $g^M=g^{M'}$ 有 $M-M'=0$，根据假设得出至少有一个 $M\neq M'$，因此等式与上述假设相矛盾。

由于定理 3 是基于 CDH 困难问题的，因此定理 4 由此得出。定理 4 证明了 M2MD-PDP 是不可伪造的。

6 性能分析与实验结果

6.1 性能分析

1）计算成本。令 C_p，$C_{\exp}$，C_{mul}，C_{div} 各自表示群 G_1 上的配对、幂、乘法运算和除法运算。由于在 Z_q 上的哈希、加法和乘法等其他操作的成本可忽略不计，因此省略。假设数据拥有者在 L 个云服务器上总共存储 N 个副本，每个 CSS 的副本编号为 $|CT_i|$，每一个副本有 n 个块，TPA 挑战 c 个块。算法 KeyGen 需要耗费一个 C_{mul} 来生成用户的私钥。生成副本算法 RepGen 需要运行 N 次加密算法 E_K。假设 E_K 的计算成本是 C_E，RepGen 的计算成本是 $N\cdot C_E$，生成所有副本的所有标签，算法 TagGen 花费 $N\cdot n\cdot((3\Delta-1)C_{\exp}+(\Delta-1)C_{\mathrm{mul}}+1C_{\mathrm{div}})$。算法 Challenge 的计算成本可忽略。每一个 CSS 运行 ProofGen 算法生成完整性证明。假设 CSS 存储 L 个副本，那么 ProofGen 的计算成本是 $L\cdot c\cdot(C_{\exp}+C_{\mathrm{mul}})$。CO 运行算法 ProofAggre 来聚合所有的证明，计算成本为 $L\cdot C_{\mathrm{mul}}$。为了验证文件完整性，TPA 运行 Verify 算法，需要的计算成本为 $\Delta\cdot C_p+N\cdot c\cdot C_{\exp}+N\cdot c\cdot C_{\mathrm{mul}}$。表 3 列出了 MB-PMDDP 方案[27]、MCMS-PDP 方案[30] 与本文方案的计算成本。

表 3 实验结果分析

Table 3 Experimental results analysis

方案	TagGen	ProofGen	ProofAggre	CheckProof
MB-PMDDP[27]	$(s+1)C_{exp}+(s-1)C_{mul}$	$c\cdot C_{exp}+(c-1)C_{mul}$	—	$2C_p+(N\cdot c+s+1)\cdot C_{exp}+(N\cdot c+s-1)C_{mul}$
MCMS-PDP[30]	$2C_{exp}+C_{mul}$	$(c+t)\cdot C_{exp}+(c+t-2)C_{mul}$	$(L-1)\cdot C_{mul}$	$2C_p+(N\cdot c+s)\cdot C_{exp}+(N\cdot c+s)C_{mul}$
本文方案	$(3\Delta-1)C_{exp}+(\Delta-1)C_{mul}+1C_{div}$	$c\cdot C_{exp}+c\cdot C_{mul}$	$L\cdot C_{mul}$	$\Delta\cdot C_p+N\cdot c\cdot C_{exp}+N\cdot c\cdot C_{mul}$

2)通信成本。在对数据文件进行处理后,数据文件被一次性上传给 CO。为了验证云服务器上文件的完整性,我们只考虑方案中传输挑战 $chal$ 和证明 $proof$ 的通信成本。TPA 向 CO 提交挑战信息 $chal=(c,k_1,k_2)$ 以及文件名 Fid,因此挑战信息的通信成本为 $(\log c+2|q|+|Fid|)$bits。CO 向 TPA 返回证明 $V=\{T_{1\leqslant t\leqslant\Delta},M,T_{Fid}\}$,因此挑战回应的通信成本为 $(|q|+\Delta|G_1|+|T_{Fid}|)$bits。

6.2 实验结果

为了直观地显示本文方案与 MB-PMDDP 方案[27]及 MCMS-PDP 方案[30]时间成本的比较,我们在两种不同的环境下对 TagGen,ProofGen,ProofAggre 和 CheckProof 算法进行了实验。

MB-PMDDP 方案与 MCMS-PDP 方案以及本文方案 M2MD-PDP 都是在基于身份密码体制下实现的,MB-PMDDP 方案和 MCMS-PDP 方案中都将数据拥有者身份暴露,并没有考虑用户匿名问题,且在 MB-PMDDP 方案中数据拥有者需要与所有的云服务器交互,使得数据拥有者的通信成本增加。虽然 MCMS-PDP 方案中增加了 CO 即云管理器,数据拥有者只需要与 CO 进行交互,降低了通信负担,但仍没有考虑匿名性问题。而本文方案在 TagGen 阶段使用环签名算法使得数据拥有者避免了身份暴露问题。图 7—图 10 分别给出了 3 个方案在标签生成 TagGen 阶段、验证值生成 ProofGen 阶段、证明聚合 ProofAggre 阶段和验证时间 CheckProof 阶段的比较结果。

在 TagGen 阶段,我们在 Windows11 操作系统上使用 2.6 GHz Intel(R) Core(TM)i7 处理器和 16 GB RAM 进行实验。采用 PBC 的 A 型椭圆曲线,组序为 160 位即 $|q|=160$,在椭圆曲线上,该点有 x 坐标和 y 坐标,长度为 512 位,其安全级别可等同于 1 024 位的 RSA。在本实验中,选取的文件大小为 4 MB 并生成 20 个副本,每个副本有 500 个块。在标签生成阶段,本文方案与 MB-PMDDP 方案以及 MCMS-PDP 的对比如图 7 所示,可以看到在生成相同的块标签时 MCMS-PDP 以及 MB-PMDDP 的时间消耗比本文方案 M2MD-PDP 更多。

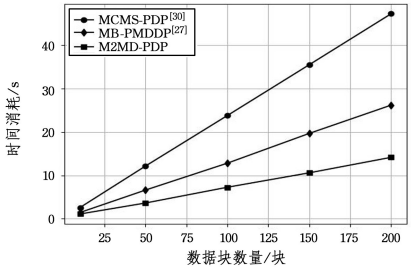

图 7 块标签生成时间

Fig. 7 Generation time of block tag

对于由 CSS,CO,TPA 执行的 3 种算法 ProofGen,ProofAggre 和 CheckProof,我们在 VMware Workstation Pro 上使用 windows10 系统,以 1 CPU,4 GB RAM 和 60 GB ROM 进行实验。VMware Workstation Pro 安装在 windows11 操作系统上,使用 2.7 GHz Intel(R) Core(TM)i5 处理器和 8 GB RAM。在实验中,我们同样采用 PBC 的 A 型椭圆曲线,组序为 160 位。为了不失一般性,选取的文件仍然是 4 MB,生成 20 个文件副本,每个副本有 500 个块。

在验证值生成阶段,如图 8 所示,可以发现 M2MD-PDP 的时间消耗相较于 MCMS-PDP 和 MB-PMDD 是稍高的,原因在于本文方案 M2MD-PDP 需要保护用户身份的秘密性。生成挑战验证值时 MCMS-PDP 中只存在一个用户即真实数据拥有者,生成挑战只需要通过副本以及真实数据拥有者生成挑战的验证值,但本文方案中存在的是一个用户群,群成员数量为 Δ,($\Delta=20,50,100,\cdots$),每个成员都需要计算挑战值,因此本文方案在验证值生成时相比于 MCMS-PDP 方案和 MB-PMDDP 方案时间效率会有所增加,但仍在可接受范围内。

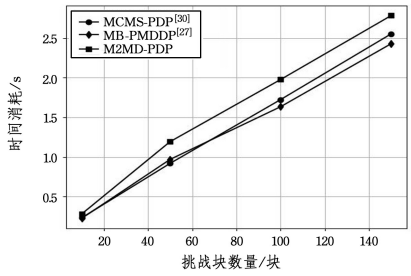

图 8 验证值生成时间

Fig. 8 Generation time of verification

在证明聚合阶段,如图 9 所示,由于 MB-PMDD 方案中没有实体 CO 为云服务器聚合证明,从而把 MB-PMDD 方案看作在单云服务器下实现,因此本文方案仅与 MCMS-PDP 方案相比较。与验证值生成阶段原因相同,由于 CO 在聚合块标签时需要将所有环签名成员所生成的标签进行聚合,因此在聚合时间上与 MCMS-PDP 方案中仅聚合数据拥有者的块标签相比,时间消耗有所增加,但仍在可接受范围内。

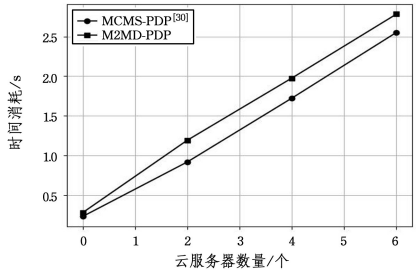

图 9 证明聚合时间

Fig. 9 Generation time of proof aggregation

在验证阶段,如图 10 所示,可以看到在相同副本以及块

数量的情况下，随着挑战块的增加，MCMS-PDP 方案与 MB-PMDDP 方案的执行时间是呈线性增加的，而本文方案 M2MD-PDP 的验证时间无太大变化。可以看出，本文方案 M2MD-PDP 的性能是优于 MB-PMDDP 方案和 MCMS-PDP 方案的。

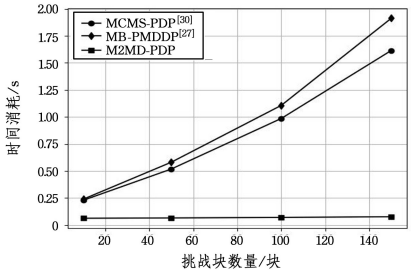

图 10 验证时间的比较

Fig. 10 Comparison of verification time

结束语 本文在多云的环境下提出了一种基于身份的可用于保护用户身份隐私性的 PDP 方案，用于检查分布式云服务器上多副本中的数据的完整性。本文方案设计在挑战者交互响应中可以一次验证多个云服务器上的多个副本的数据完整性，并且可以通过动态操作将需要更新的数据加入到各个副本中。本文方案是基于身份的，相比于传统的 PKI 模式所带来的证书管理问题，本文方案更加高效。此外，本文方案还支持公开验证。理论分析和实验结果表明，本文方案具有较高的效率和实用性。在未来，还需要继续优化本文方案，使其具有更好的用户隐私性，并提高其效率。

参 考 文 献

[1] ZUO C, SHAO J, LIU K J, et al. Fine-grained two factor protection mechanism for data sharing in cloud storage[J]. IEEE Transactions on Information Forensics and Security, 2018, 13(1): 186-196.

[2] LI J G, YAO W, HAN J G, et al. User Collusion Avoidance CP-ABE With Efficient Attribute Revocation for Cloud Storage[J]. IEEE Systems Journal, 2018, 12(2): 1767-1777.

[3] LI J G, YU Q H, ZHANG Y C, et al. Key-policy attribute-based encryption against continual auxiliary input leakage[J]. IEEE Journal of Selected Topics in Signal Processing, 2019, 470: 175-188.

[4] ZHU Y, AHN G J, HU H X, et al. Dynamic audit services for outsourced storage in clouds[J]. IEEE Transactions on Services Computing, 2013, 6(2): 227-238.

[5] BHARAT V, PATIL M. Advanced cooperative provable data possession based data integrity verification for multi-cloud storage[J]. International Journal of Computer Applications, 2013, 81(13): 24-27.

[6] YAN H, LI J G, HAN J G, et al. A novel efficient remote data possession checking protocol in cloud storage[J]. IEEE Transactions on Information Forensics and Security, 2017, 12(1): 78-88.

[7] HE J, ZHANG Y C, HUANG G Y, et al. Distributed data possession checking for securing multiple replicas in geographically-dispersed clouds[J]. Journal of Computer and System Sciences, 2012, 78: 1345-1358.

[8] ZHUO H, YU N H. A multiple-replica remote data possession checking protocol with public verifiability[C]// Second International Symposium on Data, Privacy, and E-Commerce. 2010: 84-89.

[9] SHAMIR A. Identity-based cryptosystems and signature schemes[J]. Springer, 1984, 196: 47-53.

[10] BONEH D, FRANKLIN M. Identity-based encryption from the weil pairing[C]// Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology. 2001: 213-229.

[11] RIVEST R L, SHAMIR A, TAUMAN Y. How to Leak a Secret [C]// Theory and Application of Cryptology and Information Security: Advances in Cryptology. 2001: 552-565.

[12] LIN C Y, WU T C. An identity-based ring signature scheme from bilinear pairings[C]// 18th International Conference on Advanced Information Networking and Applications. 2004: 182-185.

[13] MAN H A, JOSEPH K L, TSZ H Y, et al. ID-Based ring signature scheme secure in the standard model[J]. Information and Computer Security. 2006, 4266: 1-16.

[14] ATENIESE G, BURNS R, CURTMOLA R, et al. Provable data possession at untrusted stores[C]// Proceedings of the 14th ACM Conference on Computer and Communications Security. 2007: 598-609.

[15] ATENIESE G, PIETRO R D, MANCINI L V, et al. Scalable and efficient provable data possession[C]// Proceedings of the 4th International Conference on Security and Privacy in Communication Networks. 2008: 1-10.

[16] SEBE F, DOMINGO-FERRER J, MARTINEZ-BALLESTE A, et al. Efficient remote data possession checking in critical information infrastructures[J]. IEEE Transactions on Knowledge and Data Engineering, 2008, 20(8): 1034-1038.

[17] ERWAY C, PAPAMANTHOU C, TAMASSIA R. Dynamic provable data possession[J]. ACM Transactions on Information and System Security, 2009, 17(4): 213-222.

[18] WANG Q, WANG C, REN K, et al. Enabling public auditability and data dynamics for storage security in cloud computing [J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(5): 847-859.

[19] YANG K, JIA X H. An efficient and secure dynamic auditing protocol for data storage in cloud computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2013, 24(9): 1717-1726.

[20] WANG B Y, LI B C, LI H. Oruta: Privacy-preserving public auditing for shared data in the cloud[J]. IEEE Transactions on Cloud Computing, 2014, 2(1): 43-56.

[21] YU Y, AU M H, ATENIESE G, et al. Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage[J]. IEEE Transactions on Information Forensics and Security, 2017, 12(4): 767-778.

[22] WANG H Q, HE D B, YU J, et al. Incentive and unconditionally anonymous identity-based public provable data possession[J]. IEEE Transactions on Services Computing, 2019, 12(5): 824-835.

[23] ZHU Y, HONG X H, GAIL A, et al. Cooperative provable data possession for integrity verification in multicloud storage[J]. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(12): 2231-2244.

[24] WANG H Q. Identity-based distributed provable data possession in multicloud storage[J]. IEEE Transactions on Services Computing, 2015, 8(2): 328-340.

[25] CURTMOLA R, KHAN O, BANDAL R, et al. MR-PDP: Multiple-replica provable data possession[C]// The 28th International Conference on Distributed Computing Systems. 2008: 411-420.

[26] HAO Z, YU N H. A multiple-replica remote data possession checking protocol with public verifiability[C]// 2010 Second International Symposium on Data, Privacy, and E-Commerce. 2010: 84-89.

[27] BARSOUM A F, HASAN M A. Provable multicopy dynamic data possession in cloud computing systems[J]. IEEE Transactions on Information Forensics and Security, 2015, 10(3): 485-497.

[28] ZHANG Y F, NI J B, TAO X L, et al. Provable multiple replication data possession with full dynamics for secure cloud Storage [J]. Concurrency and Computation: Practice and Experience, 2016, 28: 1161-1173.

[29] ZHU Y, AHN G J, HU H X, et al. Dynamic audit services for integrity verification of outsourced storages in clouds[C]// Proceedings of the 2011 ACM Symposium on Applied Computing. 2011: 1550-1557.

[30] SHEN J Y, ZEN P, CHOO K. Multicopy and multiserver provable data possession for cloud-based IoT[J]. IEEE Internet of Things Journal, 2022, 9(14): 12300-12310.

[31] LI J G, YAN H, ZHANG Y C. Efficient Identity-Based Provable Multi-Copy Data Possession in Multi-Cloud Storage[J]. IEEE Transactions on Cloud Computing, 2022, 10(1): 356-365.

TAN Shiyi, born in 1999, master candidate. Her main research interests include cryptography and information security.

WANG Huaqun, born in 1974, Ph. D, professor. His main research interests include cryprtography, blockchain, and cloud computing security.

（责任编辑：喻藜）