软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007216] [CSTR: 32375.14.jos.007216]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

# 支持高效数据所有权共享的动态云存储审计方案*

殷新春[1,2,3], 王经纬[1], 宁建廷[4,5]

[1](扬州大学 信息工程学院, 江苏 扬州 225127)

[2](扬州大学广陵学院, 江苏 扬州 225000)

[3](广东省信息安全技术重点实验室, 广东 广州 510275)

[4](福建师范大学 计算机与网络空间安全学院, 福建 福州 350007)

[5](信息安全国家重点实验室 (中国科学院 信息工程研究所), 北京 100093)

通信作者: 宁建廷, E-mail: jtning@fjnu.edu.cn

**摘　要**: 云存储审计技术的出现为存储在云中的数据提供了可靠的安全保障, 数据拥有者可以轻易地验证存储在云中数据的完整性. 然而, 云服务器中可能存储着海量的数据, 目前的云存储审计方案在进行数据完整性验证以及数据所有权变更时均需花费大量的计算开销. 为了缓解该问题并提高云存储审计方案的实用性, 提出一种支持高效数据所有权共享的动态云存储审计方案. 在计算开销方面, 构造一种高效的验证结构可以聚合数据验证信息, 免去大量计算开销较高的双线性配对运算. 基于变色龙哈希函数易于制造新碰撞的特性设计高效的数据所有权共享机制, 共享数据所有权只需更新对应用户的密钥即可, 无需修改云服务器中存储的密文. 此外, 方案还提供了数据细粒度共享、密态数据验证以及数据动态修改功能. 安全性分析和性能分析表明, 方案可以在保证数据安全的同时不对方案的运行效率产生影响, 具有较高的实用性.

**关键词**: 云存储审计; 云计算; 数据所有权共享; 批量验证; 数据共享

**中图法分类号**: TP309

中文引用格式: 殷新春, 王经纬, 宁建廷. 支持高效数据所有权共享的动态云存储审计方案. 软件学报. http://www.jos.org.cn/1000-9825/7216.htm

英文引用格式: Yin XC, Wang JW, Ning JT. Dynamic Cloud Storage Auditing Scheme with Efficient Data Ownership Sharing. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7216.htm

## Dynamic Cloud Storage Auditing Scheme with Efficient Data Ownership Sharing

YIN Xin-Chun[1,2,3], WANG Jing-Wei[1], NING Jian-Ting[4,5]

[1](College of Information Engineering, Yangzhou University, Yangzhou 225127, China)

[2](Guangling College of Yangzhou University, Yangzhou 225000, China)

[3](Guangdong Provincial Key Laboratory of Information Security Technology, Guangzhou 510275, China)

[4](College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350007, China)

[5](State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences), Beijing 100093, China)

**Abstract**: Cloud storage auditing guarantees the security of data stored in the cloud, enabling data owners to easily verify the integrity of data. However, a vast amount of data in the cloud can lead to significant computational overhead during cloud storage auditing when verifying data integrity and modifying data ownership. To solve this problem as well as provide practical solutions, this study proposes a dynamic auditing scheme for cloud storage with efficient data ownership sharing. An efficient validation structure is constructed to

* 基金项目: 国家自然科学基金 (62102090, 62032005, 61972094, 61902070); 福建省科协第二届青年人才托举工程; 广东省信息安全技术重点实验室开放基金 (2020B1212060078)
收稿时间: 2023-02-13; 修改时间: 2023-12-03; 采用时间: 2024-04-23; jos 在线出版时间: 2024-12-11

aggregate information for data verification, avoiding a large number of bilinear pairing operations that incur high computational costs. An efficient mechanism for data ownership sharing is designed based on the chameleon hash function's ability to generate new collisions. It allows updating the secret key of the corresponding user for shared data ownership without modifying the ciphertexts stored in the cloud. In addition, the proposed scheme achieves fine-grained data sharing, encrypted data auditing, and dynamic data modification. The security and performance analyses show that the proposed scheme ensures the security of data in the cloud without affecting its performance, which means it is a practical scheme.

**Key words**: cloud storage auditing; cloud computing; data ownership sharing; batch auditing; data sharing

近年来, 云存储技术得到了快速的发展, 作为其中最为普遍的应用之一, 云数据共享服务为人们的生活带来了极大的便利. 通过云数据共享, 人们可以将自己存储的数据与网络上的任何人进行共享, 因此在金融、医疗、交通运输等行业中具有广阔的应用前景[1–4]. 然而, 当数据被存储到云服务器中之后, 用户就失去了对数据的掌控能力, 一旦发生诸如黑客攻击、服务器错误等意外情况, 用户的数据可能直接丢失, 从而损害用户的利益. 为了保护数据的安全并防止上述情况的出现, 研究者们提出了云存储审计的概念[5–7]. 在云存储审计方案中, 数据拥有者在上传数据之前需要通过个人私钥为自己的数据生成标签. 当发起验证挑战后, 云服务器通过这些标签来证明所存储数据的完整性. 虽然这些方案可以保证用户存储数据的完整性, 但在实际应用场景中, 云存储审计方案还面临着很多挑战[8].

首先, 目前大多数的云存储审计方案仅支持对明文数据进行完整性验证[9]. 然而由于负责存储数据的第三方云服务器是半可信的, 他们可以轻易地从存储的明文数据中窥探用户的隐私, 从而威胁用户数据的安全[10]. 不仅如此, 在存储明文数据的情况下, 恶意的攻击者或任意可以访问云服务器的人员均可获取云服务器中存储的内容, 因此如果不采取措施对存储在云服务器中的数据进行保护, 用户存储在云中的数据将面临巨大的安全风险. 其次, 用户在使用云存储服务时往往同时希望可以将自己的数据与他人共享, 而传统的云存储审计方案仅支持群组形式的用户共享. 群组中用户上传的数据必须对组内所有成员公开, 不支持更细粒度的数据共享机制, 因此限制了云存储审计方案的应用场景. 此外, 在使用云存储服务的过程中, 数据不是一成不变的, 用户经常需要对存储在云中的数据进行添加、修改或删除操作. 如果能同时支持云存储数据的细粒度共享与动态数据管理, 可以极大地提高云存储审计方案的实用性. 最后, 目前的云存储审计方案均没有考虑数据所有权共享的情况. 在云数据共享的背景下, 数据拥有者邀请系统中的其他用户合作维护数据内容的场景十分常见, 这要求云存储审计方案支持用户将数据所有权共享给其他用户以便对数据进行修改维护. 然而, 现存的云存储审计方案仅支持数据所有权的"一对一"转移, 即通过重签名密钥直接将撤销用户对其文件的签名转为其他非撤销用户的签名, 无法满足多用户合作维护文件场景下数据所有权"一对多"共享的需求. 此外, 目前的数据所有权"一对一"转移过程中涉及的计算开销会随撤销用户拥有文件的数量增加而线性增长. 尽管这部分计算开销可以由云服务器来承担, 但考虑到用户可能在云服务器中存储海量的数据以及用户数据的所有权常常需要变更, 这将会为云服务器的计算能力带来极重的负担. 基于上述讨论, 本文提出了一种支持高效数据所有权共享的动态云存储审计方案, 具体贡献如下.

(1) 本文方案支持用户将数据的所有权共享给其他用户. 通过使用变色龙哈希函数, 本文方案在共享用户数据的所有权时只要为新的用户更新密钥即可, 无需修改存储在云服务器中的文件, 因此效率更高. 为了保证数据的机密性和隐私性, 本文方案中的数据均以密文的形式存储在云服务器中且只有拥有数据所有权的用户才能对数据进行动态修改和完整性审计.

(2) 本文方案支持对存储在云中的数据进行动态修改. 数据用户可以通过增加、删除、修改操作灵活地修改自己存储在云服务器中的数据, 上述 3 种对数据的动态修改既不会影响系统的其他功能, 也不涉及对云服务器中其他数据的修改.

(3) 本文方案通过聚合计算的方式将原本需要执行的大量双线性配对计算替换为乘积计算, 缓解了云服务器和可信中心在数据完整性验证过程中的计算压力.

(4) 安全性分析表明本文方案可以保证数据安全性以及云存储审计的正确性、可检测性和合理性. 此外, 我们还通过模拟实验的方式对本文方案的性能进行评估, 实验结果表明本文方案在保证效率的前提下具有更好的实用性.

## 1 相关工作

在传统的云数据存储方案中, 如果数据拥有者需要验证所存储数据的完整性, 需要将所有数据从云服务器中下载到本地进行验证, 从而带来了巨大的计算和通信开销. 为此, Ateniese 等人于 2007 年在文献 [11] 中首次提出了可证明数据所有权 (provable data possession, PDP) 的方案, 允许数据拥有者通过同态数据标签对存储在云中的数据进行抽样验证, 免去了数据拥有者下载所有数据到本地的负担. Juels 等人在文献 [12] 中提出了一种基于双线性映射的数据完整性验证方案, 与文献 [11] 相比, 该方案不仅支持对存储在云服务器中的数据进行完整性验证, 而且提供了数据检索的功能. 基于他们的研究, 研究人员分别从计算效率、功能、安全性等方面进行探索, 提出了许多云数据存储审计方案[13–21].

在云数据存储审计方案中, 用户的数据往往不是一成不变的, 当数据更新之后, 依然需要保证更新后数据的完整性. 为了解决该问题, Ateniese 等人在文献 [13] 中基于哈希函数设计了一种支持数据动态修改的数据审计方案, 然而在系统建立阶段, 该方案需要为所有数据块执行预计算操作, 因此限制了数据动态操作的灵活性. 此外, 当数据修改完成之后, 该方案需要更新所有未修改的数据块, 实用性较低. Wang 等人在文献 [14] 中提出了一种动态数据审计协议, 该协议允许云服务器代替用户对存储的数据块进行增加、删除和修改操作. 然而该方案存在可能导致数据隐私泄露的问题. Rao 等人[15]基于 Merkle 哈希树设计了一种云存储审计方案, 可以保证数据拥有者存储在云中的数据不受不可信第三方的威胁, 然而该方案存在数据审计计算开销过大的问题. 文献 [16] 基于哈希表设计了一种动态数据审计方案, 但该方案存在与文献 [15] 同样的问题. Yang 等人在文献 [17] 中提出了一种高效的云存储审计协议, 除了实现数据的动态修改, 该协议还在保证数据隐私的前提下缓解了审计机构的计算和通信开销.

除了数据动态修改, 数据共享同样是数据拥有者经常需要使用的功能. 然而目前提出的大多数云存储审计方案在这方面的研究却十分有限. 在文献 [18] 中, Zhang 等人提出了一种支持高效用户撤销的云存储审计方案, 其中所有的用户都归属于不同的用户群, 用户上传的数据可以在群内共享. 当需要撤销群中的某个用户时, 为了保证撤销用户上传的数据不受用户撤销的影响, 该方案允许群管理者在不修改数据的情况下将撤销用户的数据转移给群中的其他用户. 在文献 [19] 中, Su 等人同样提出了一种基于用户群的云存储审计方案, 具备数据隐私保护、数据可追溯和可撤销等特性. 文献 [20] 基于身份基加密设计了一种不依赖证书的云存储审计方案, 用户私钥由密钥生成中心根据用户身份计算得出, 从而避免了证书管理的问题. 该方案中数据用户只需要将加密后的数据和标签发送给群管理者, 由群管理者负责数据的上传与完整性验证等操作. 然而, 上述方案均只支持群内的数据共享, 无法实现更细粒度的数据共享方式. 文献 [21] 设计了一种同时支持数据细粒度共享、动态数据更新、批量审计以及属性撤销的云存储审计方案, 其中数据共享通过密文策略属性基加密实现. 虽然该方案在功能上更为全面, 但每次执行属性撤销之后, 该方案都需要更新系统中与撤销用户相关的密文, 计算开销较高.

## 2 基础知识

### 2.1 访问结构

令 $S=\{u_1, u_2, \ldots, u_n\}$表示属性空间, 则称一个非空的属性集合 $\mathbb{A}\subseteq 2^S/\{\varnothing\}$为访问结构. 如果 $\mathbb{A}$ 满足: 对 $\forall B, C\subseteq S$, 如果 $B\in\mathbb{A}$ 且 $B\subseteq C$, 有 $C\in\mathbb{A}$, 则称 $\mathbb{A}$ 是单调的访问结构. 对于 $\forall D\in\mathbb{A}$, 称 $D$ 为授权集合.

### 2.2 双线性映射

令 $\mathbb{G}$ 和 $\mathbb{G}_T$ 表示阶为质数 $p$ 的双线性循环群, $g$ 为群 $\mathbb{G}$ 的一个生成元. 双线性映射 $e$ 具有以下特性:

(1) 双线性: 对于任意 $u, v\in\mathbb{G}$ 以及 $a, b\in\mathbb{Z}_p$, $e(u^a, v^b)=e(u, v)^{ab}$.

(2) 非退化性: 存在 $u, v\in\mathbb{G}$ 使得 $e(u, v)\neq 1$.

(3) 可计算性: 对于任意 $u, v\in\mathbb{G}$, 都可以有效的计算 $e(u, v)$.

### 2.3 属性基加密

一个基本的密文策略属性基加密方案包括 *Setup*、*KeyGen*、*Enc* 以及 *Dec* 这 4 个算法, 具体定义如下.

*Setup*($1^\lambda$)→{*pp*, *msk*}: 算法输入安全参数 $\lambda$, 输出系统公共参数 *pp* 和系统主私钥 *msk*.

*KeyGe*n(*pp*, *msk*, *S*)→*sk*: 算法输入系统公共参数 *pp*、系统主私钥 *msk*、属性集合 *S*, 输出用户密钥 *sk*.

*Enc*(*pp*, *m*, (***M***, $\rho$))→*CT*: 算法输入系统公共参数 *pp*、消息 *m*、访问策略 (***M***, $\rho$), 输出密文 *CT*.

*Dec*(*pp*, *sk*, *CT*)→*m* or ⊥: 算法输入系统公共参数 *pp*、用户密钥 *sk*、密文 *CT*, 如果用户密钥 *sk* 中的属性集合可以满足密文中的访问策略, 则算法输出消息 *m*, 否则算法输出⊥表示停止.

### 2.4 变色龙哈希函数

一个变色龙哈希函数包括 *KeyGen*、*Hash*、*Verify* 以及 *Adapt* 这 4 个算法, 具体定义如下.

*KeyGen*($1^\lambda$)→{*pk*, *sk*}: 算法输入安全参数 $\lambda$, 输出公钥 *pk* 以及私钥 *sk*.

*Hash*(*pk*, *m*)→{*hash*, *r*}: 算法输入公钥 *pk* 以及消息 *m*, 输出一个哈希值 *hash* 以及随机数 *r*.

*Verify*(*pk*, *m*, *hash*, *r*)→0 or 1: 算法输入公钥 *pk*、消息 *m*、哈希值 *hash* 以及随机数 *r*, 如果验证通过, 算法输出 1, 否则算法输出 0.

*Adapt*(*sk*, *m*, *m*′, *hash*, *r*)→*r*′: 算法输入私钥 *sk*、原消息 *m*、新消息 *m*′、与原消息 *m* 对应的哈希值 *hash* 以及随机数 *r*, 输出与新消息相应对的随机数 *r*′.

为了便于理解, 我们还提供了一个变色龙哈希函数的实例.

*KeyGen*($1^\lambda$): 根据安全参数 $\lambda$ 生成群参数 $D=(\mathbb{G}, \mathbb{G}_T, p, e)$, 令 $g$ 为群 $\mathbb{G}$ 的生成元, 选择随机数 $x \in \mathbb{Z}_p$, 计算 $h=g^x$, 则 $pk=\{g, h\}$, $sk=x$.

*Hash*(*pk*, *m*): 对于给定的消息 *m*, 选择随机数 $r \in \mathbb{Z}_p$, 计算消息 *m* 的哈希 $hash=g^m h^r$.

*Verify*(*pk*, *m*, *hash*, *r*): 验证等式 $hash=g^m h^r$ 是否成立, 如果成立输出 1, 否则输出 0.

*Adapt*(*sk*, *m*, *m*′, *hash*, *r*): 根据 $m+xr=m'+xr'$来计算新的随机值 *r*′, 并验证 $hash=g^{m'}h^{r'}$.

### 2.5 数据完整性审计

一个数据完整性审计方案包括 *Setup*、*KeyGen*、*TagGen*、*Chall*、*Prove* 以及 *Verify* 这 6 个算法, 具体定义如下.

*Setup*($1^\lambda$)→{*pp*, *msk*}: 算法输入安全参数 $\lambda$, 输出系统公共参数 *pp* 和系统主私钥 *msk*.

*KeyGen*(*pp*, *uid*)→*sk*: 算法输入公共参数 *pp* 和用户身份 *uid*, 输出用户密钥 *sk*.

*TagGen*(*pp*, *sk*, *M*)→*T*: 算法输入公共参数 *pp*、用户密钥 *sk* 和数据 *M*, 对于数据块 $m_i \in M$, 计算数据标签 $t_i$, 算法输出数据标签的集合 *T*.

*Chall*(*m*, *T*)→*chal*: 算法输入数据 *m* 和数据标签集合 *T*, 输出一个挑战 *chal*.

*Prove*(*m*, *T*, *chal*)→*P*: 算法输入数据 *m*, 数据标签集合 *T* 以及挑战 *chal*, 输出一个对该挑战的证明 *P*.

*Verify*(*pp*, *P*)→0 or 1: 算法输入公共参数 *pp* 和证明 *P*, 如果审计通过, 算法输出 1, 否则算法输出 0.

### 2.6 困难性问题

**定义 1 (CDH 假设)**. 挑战者生成群参数 $\delta=(p, \mathbb{G}, \mathbb{G}_T, e)$, $g$ 表示群 $\mathbb{G}$ 的生成元, 对于 $x,y \in \mathbb{Z}_p$, 挑战者向攻击者发送 $g$, $v=g^x$, $g^y \in \mathbb{G}$, 攻击者输出 $v^y$.

如果不存在任何概率多项式时间的攻击者能以不可忽略的优势赢得上述游戏, 则 CDH 假设成立.

**定义 2 (DL 假设)**. 挑战者生成群参数 $\delta=(p, \mathbb{G}, \mathbb{G}_T, e)$, $g$ 表示群 $\mathbb{G}$ 的生成元, 对于 $x \in \mathbb{Z}_p$, 挑战者向攻击者发送 $g$, $g^x \in \mathbb{G}$, 攻击者输出 $x$.

如果不存在任何概率多项式时间的攻击者能以不可忽略的优势赢得上述游戏, 则 DL 假设成立.

**定义 3 (*q*–1 假设)**. 挑战者生成群参数 $\delta=(p, \mathbb{G}, \mathbb{G}_T, e)$, 令 $g$ 表示群 $\mathbb{G}$ 的生成元, 挑战者随机选择 $a, s, b_1, \ldots, b_q \in \mathbb{Z}_p$, 向攻击者发送 $\delta$ 以及以下参数:

$$g, g^s, \{g^{a^i}, g^{b_j}, g^{sb_j}, g^{a^i b_j}, g^{a^i/b_j^2}\}_{i,j\in[q,q]}, \left\{g^{a^i b_j/b_{j'}^2}\right\}_{i,j,j'\in[2q,q,q],j\neq j'}, \left\{g^{a^i/b_j}\right\}_{i,j\in[2q,q],i\neq q+1}, \left\{g^{sa^i b_j/b_{j'}}, g^{sa^i b_j/b_{j'}^2}\right\}_{i,j,j'\in[q,q,q],j\neq j'}.$$

挑战者随机选择 $b\in\{0, 1\}$, 如果 $b=0$, 挑战者向攻击者发送 $e(g,g)^{sa^{q+1}}$, 否则挑战者向攻击者发送一个随机元素 $R \in \mathbb{G}_T$. 随后, 攻击者输出对 $b$ 的猜测 $b'$, 令攻击者猜测成功的优势为 $\varepsilon=\Pr[b'=b]-1/2$.

如果不存在任何概率多项式时间的攻击者能以不可忽略的优势赢得上述游戏, 则 $q$–1 假设成立.

## 3 形式化定义及设计目标

### 3.1 系统架构

如图 1 所示, 本文提出的支持高效数据所有权共享的动态云存储审计方案由数据拥有者 (data owner, DO)、数据用户 (data user, DU)、云服务器 (cloud server, CS)、可信中心 (trusted authority, TA) 这 4 个实体组成.

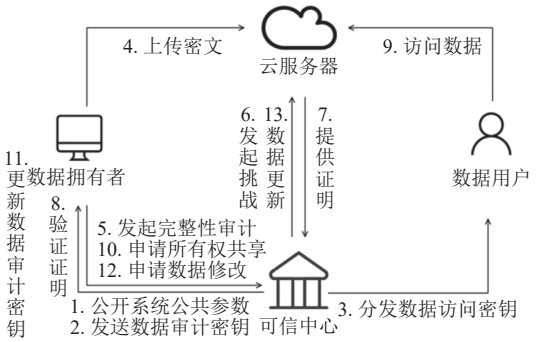


图 1 系统架构图

● 数据拥有者需要将自己存储在 CS 中的数据与他人共享. 出于安全性考虑, DO 在上传数据之前会使用属性基加密算法对数据进行加密. 此外, 本文方案还结合了变色龙哈希函数实现了数据的动态修改和数据所有权共享. DO 可以通过 TA 向 CS 发起数据完整性验证来随时检查自己存储在 CS 中数据的完整性. 在本文方案中, DO 是可信的.

● 数据用户可以访问 DO 共享的数据. 本文方案基于属性基加密实现了数据的细粒度共享, 只要 DU 自身拥有的属性集合可以满足密文中的访问策略, DU 即可完成解密. 如果 DO 将数据的所有权共享给 DU, 则对于该数据, DU 将拥有和 DO 相同的权限 (数据完整性验证和数据动态修改). 在本文方案中, DU 是半可信的.

● 云服务器拥有丰富的计算和存储资源, 主要负责存储 DO 上传的加密数据. 由于对 DO 来说, CS 是半可信的机构, 可能会为了其自身利益欺骗 DO, 因此 CS 需要在收到完整性验证挑战时生成一个数据完整性的证明. 在本文方案中, CS 是半可信的.

● 可信中心负责生成系统公共参数、为 DU 生成密钥以及协助 DO 对其存储在 CS 中的数据进行完整性验证. 在完整性验证的过程中, TA 首先向云服务器发送一个审计挑战, 通过验证 CS 对于该挑战的证明, TA 可以判断 CS 保存的数据是否完好. 在本文方案中, TA 是可信的.

### 3.2 算法定义

本文提出的支持高效数据所有权共享的动态云存储审计方案主要包括 9 个算法, 分别是 *Setup*、*KeyGen*、*Upload*、*Chall*、*Proof*、*Verify*、*Access*、*Share* 以及 *Update*, 具体定义如下.

(1) $Setup(1^{\lambda})\rightarrow\{pp, msk\}$: 系统建立算法, 算法输入安全参数 $\lambda$, 输出系统公共参数 $pp$ 和系统主私钥 $msk$.

(2) $KeyGen(pp, msk, S, uid)\rightarrow\{sk_1, sk_2\}$: 密钥生成算法, 算法输入系统公共参数 $pp$、系统主私钥 $msk$、属性集合 $S$ 以及用户身份 $uid$, 输出数据访问密钥 $sk_1$ 以及数据审计密钥 $sk_2$.

(3) $Upload(pp, ck, (\boldsymbol{M}, \rho), sk_2, F)\rightarrow CT$: 数据上传算法, 算法输入系统公共参数 $pp$、对称密钥 $ck$、访问策略 $(\boldsymbol{M}, \rho)$、数据审计密钥 $sk_2$、数据文件 $F$, 输出密文 $CT$.

(4) $Chall(sk_2)\rightarrow chal$ or $\perp$: 挑战算法, 算法输入数据审计密钥 $sk_2$, TA 首先验证用户是否具有该数据的所有权, 如果验证通过, 算法输出挑战 $chal$, 否则算法输出 $\perp$.

(5) $Proof(CT, chal)\rightarrow\{P, tag\}$: 证明算法, 算法输入密文 $CT$ 和挑战 $chal$, 输出一个证明 $P$ 以及文件标签 $tag$.

(6) *Verify*(*pp*, *chal*, *P*, *tag*)→0 or 1: 验证算法, 算法输入系统公共参数 *pp*、挑战 *chal*、证明 *P* 以及文件标签 *tag*, 如果验证通过, 算法输出 1, 否则算法输出 0.

(7) *Access*($sk_1$, *CT*)→*F* or ⊥: 数据访问算法, 算法输入数据访问密钥 $sk_1$、密文 *CT*, 如果数据访问密钥中的属性能够满足密文中的访问策略, 算法输出文件 *F*, 否则算法输出⊥.

(8) *Share*(*pp*, *uid*, $sk_2$, *uid*′, $sk'_2$)→ $sk'_2$ : 数据所有权共享算法, 算法输入系统公共参数 *pp*、用户身份 *uid* 及对应的数据审计密钥 $sk_2$、用户身份 *uid*′以及对应的用户审计密钥 $sk'_2$, 输出更新后的用户审计密钥 $sk'_2$.

(9) *Update*(*pp*, *uid*, $sk_2$, *CT*)→*CT*: 更新算法, 算法输入系统公共参数 *pp*、用户身份 *uid* 及对应的数据审计密钥 $sk_2$、密文 *CT*, 输出更新后的密文 *CT*.

### 3.3 设计目标

(1) 云存储审计的正确性: 如果本文方案中的可信中心及数据拥有者是可信的, 云服务器是半可信的, 且他们能按照方案设计的步骤对存储在云服务器中的数据进行审计, 则可信中心可以验证通过任意存储在云服务器中的有效数据.

(2) 错误数据的可检测性: 如果本文方案中的可信中心及数据拥有者是可信的, 云服务器是半可信的, 且他们能按照方案设计的步骤对存储在云服务器中的数据进行审计, 那么可信中心可以检测出云服务器中存在的损坏数据.

(3) 云存储审计的合理性: 如果本文方案中的可信中心及数据拥有者是可信的, 且他们能按照方案设计的步骤对存储在云服务器中的数据进行审计, 那么恶意的云服务器无法通过数据完整性审计.

(4) 数据机密性: 如果本文方案中的可信中心及数据拥有者是可信的, 云服务器是半可信的, 且他们能按照方案设计的步骤对存储在云服务器中的数据进行加密和存储, 则恶意的数据用户无法推测出数据的内容.

(5) 高效性: 为了提高方案的执行效率, 我们将数据完整性验证过程中大量耗时的双线性配对计算替换为乘积计算. 此外, 在进行数据所有权共享时, 本文方案仅需更新新数据拥有者的密钥, 与无需修改存储在云中的数据. 因此可以为云服务器节省大量的计算开销.

(6) 灵活性: 为了实现灵活的数据共享、保障用户的隐私, 本文方案使用属性基加密对存储在云服务器中的数据进行细粒度访问控制, 只有满足访问条件的用户才能完成解密获取数据.

(7) 数据安全: 为了保证用户存储在云中数据的安全性, 本文方案要求用户在上传数据到云服务器之前需要对数据进行加密, 因此无论是云服务器还是恶意的攻击者均无法获取数据的内容.

## 4 方案设计

### 4.1 技术概述

为了降低数据完整性验证过程中的计算开销、实现高效的数据所有权共享、保证数据的安全、提供数据细粒度共享以及动态数据修改的功能, 本文提出一种支持高效数据所有权共享的动态云存储审计方案. 具体来说, ① 在数据完整性验证方面, 我们构造了一种高效的验证结构, 将所有用于验证数据完整性的数据块信息以乘积的方式进行聚合. 通过这样的方式, 我们只需执行两次双线性配对计算即可完成数据完整性验证, 提高了数据完整性验证时的计算效率. ② 在数据所有权共享方面, 我们基于变色龙哈希函数在已知陷门 (陷门的特殊性质允许一些数学问题的求解在已知某些信息的情况下变得容易, 而在没有这些信息的情况下则变得极其困难) 的情况下可以快速制造新碰撞的特性设计了一种高效的数据所有权共享机制, 用户可以将自己对某数据的所有权共享给其他用户. 在本文方案中, 数据所有权通过用户的数据审计密钥体现, 只有数据审计密钥与数据文件中内容相匹配的用户才拥有数据的所有权, 可以对数据进行完整验证和数据动态修改操作. 数据审计密钥中包含一个由变色龙哈希函数生成的随机数, 在数据所有权共享时, 通过变色龙哈希函数的 *Adapt* 算法 (详见第 2.4 节) 为新的数据拥有者生成与该数据匹配的新随机数作为陷门, 因此数据所有权共享只需更新新数据拥有者的密钥即可, 无需修改云服务器中存储的密文. ③ 在数据安全性方面, 用户在上传数据到云服务器之前需要对数据进行加密, 加密密钥由属性基加密保护. 只有数据访问密钥中属性满足密文中访问策略的用户才能成功获取密钥解密数据. ④ 在数据动态修

改方面, 本文方案的数据动态修改功能是基于变色龙哈希函数的 *Verify* 算法 (详见第 2.4 节) 实现. 在数据修改时, 可信中心需要验证数据的所有权是否归属于该用户, 当用户通过验证之后才能对数据进行增加、删除和修改操作.

### 4.2 支持高效数据所有权共享的动态云存储审计方案介绍

在本节中, 我们给出支持高效数据所有权共享的动态云存储审计方案[22]的具体介绍. 该方案由 9 个算法组成, 令属性空间为 $U=\mathbb{Z}_p$, 待共享的文件为 $F=(fn, \{m_i\}_{i\in[f]})$, 其中 $fn$ 为文件名, $m_i$ 表示文件 $F$ 的第 $i$ 个数据块. 方案中使用的变色龙哈希函数采用第 2.3 节中提供的方案, 属性基加密采用文献 [23] 中的方案. 由于本文方案的构造具有通用性, 因此使用的变色龙哈希函数和属性基加密均可采用其他方案的构造. 此外, 我们还使用了一个身份基签名方案 *SSig* 来保证文件名和验证参数的安全, *ssk* 为身份基签名方案 *SSig* 的私钥.

● $Setup(1^{\lambda})\rightarrow\{pp, msk\}$

(1) TA 根据安全参数 $\lambda$ 生成群参数 $D=(\mathbb{G}, \mathbb{G}_T, p, e)$, 然后选择随机元素 $g, h, u, v, w\in\mathbb{G}$, $\alpha, x\in\mathbb{Z}_p$.

(2) TA 选择 2 个哈希函数 $H_0: \mathbb{G}\rightarrow\mathbb{Z}_p$, $H_1: \{0, 1\}^*\rightarrow\mathbb{G}$, TA 计算 $Y=g^x$, $e(g, g)^{\alpha}$.

(3) TA 公开系统公共参数 $pp=\{D, g, h, u, v, w, Y, e(g, g)^{\alpha}, H_0, H_1\}$, 秘密保存系统主私钥 $msk=\{\alpha, x\}$.

● $KeyGen(pp, msk, S=\{A_1, \ldots, A_k\}\in\mathbb{Z}_p, uid)\rightarrow\{sk_1, sk_2\}$

(1) 令 $k=|S|$, TA 随机选择 $k+1$ 个元素 $o, o_1, o_2, \ldots, o_k\in\mathbb{Z}_p$, 然后计算 $K_0=g^{\alpha}w^{o}$, $K_1=g^{o}$. 对于 $\tau\in[k]$, TA 计算 $K_{\tau,2}=g^{o_\tau}$, $K_{\tau,3}=(u^{A_\tau}h)^{o_\tau}v^{-o}$, 令数据访问密钥为 $sk_1=\{S, K_0, K_1, \{K_{\tau,2}, K_{\tau,3}\}_{\tau\in[k]}\}$.

(2) TA 为 *uid* 选择随机数 $r_{uid}\in\mathbb{Z}_p$, 然后计算 $R_{uid}=g^{r_{uid}}$, $\sigma_{uid}=r_{uid}+xH_0(R_{uid})$. TA 选择随机数 $r_1\in\mathbb{Z}_p$, 计算 $R_{uid}$ 的变色龙哈希 $h_1=g^{H_0(R_{uid})}h^{r_1}$, 令数据审计密钥为 $sk_2=\{R_{uid}, r_{uid}, \sigma_{uid}, h_1, r_1\}$.

(3) TA 将 $\{sk_1, sk_2\}$ 秘密发送给用户.

● $Upload(pp, ck, (\boldsymbol{M}, \rho), sk_2, F)\rightarrow CT$

(1) DO 随机选择向量 $\boldsymbol{y}=(s, y_2, \ldots, y_n)^{\mathrm{T}}\in\mathbb{Z}_p$, 其中 $s$ 是待共享的秘密值. 对于 $i\in[l]$, 令 $\boldsymbol{M}_i$ 为矩阵 $\boldsymbol{M}$ 的第 $i$ 行, DO 计算 $\lambda_i=\boldsymbol{M}_i\boldsymbol{y}$. DO 选择 $l$ 个随机数 $t_1, t_2, \ldots, t_l\in\mathbb{Z}_p$, 计算 $C=cke(g, g)^{\alpha s}$, $C_0=g^{s}$, 对于 $j\in[l]$, 计算 $C_{j,1}=w^{\lambda_j}v^{t_j}$, $C_{j,2}=(u^{\rho(j)}h)^{-t_j}$, $C_{j,3}=g^{t_j}$.

(2) 对于 $i\in[f]$, DO 使用 $ck$ 加密数据块 $m_i$ 获得对称密文 $e_i$. DO 计算密文 $e_i$ 的标签 $T_i=(H_1(fn\|i)u^{e_i})^{\sigma_{uid}}$ 以及文件 $F$ 的标签 $tag=fn\|R_{uid}\|SSig_{ssk}(fn\|R_{uid})$.

(3) DO 将密文 $CT=\{(\boldsymbol{M}, \rho), C, C_0, \{C_{j,1}, C_{j,2}, C_{j,3}\}_{j\in[l]}, \{e_i, T_i\}_{i\in[f]}, tag\}$ 上传至 CS.

(4) CS 收到密文 $CT$ 后, 首先需要验证文件的正确性. CS 需要验证: ① 文件标签 $tag$ 的正确性: CS 验证签名 $SSig_{ssk}(fn\|R_{uid})$ 是否正确; ② 数据块标签的正确性: CS 验证等式 $e\left(\prod_{i\in[f]}T_i, g\right)=e\left(\prod_{i\in[f]}H_1(fn\|i)u^{e_i}, R_{uid}Y^{H_0(R_{uid})}\right)$ 是否成立. 如果上述①②验证均通过, CS 保存密文, 否则 CS 拒绝为用户存储密文.

● $Chall(sk_2)\rightarrow chal$ or $\perp$

(1) DO 可以向 TA 发起完整性验证请求, TA 收到请求后, 首先需要验证 DO 是否拥有数据的所有权, TA 验证等式 $h_1=g^{H_0(R_{uid})}h^{r_1}$ 是否成立, 如果成立则继续执行.

(2) TA 随机选择包含 $c$ 个元素的集合 $I\in[c]$. 对于 $i\in I$, TA 输入随机元素 $v_i\in\mathbb{Z}_p$, 将挑战 $chal=\{i, v_i\}_{i\in I}$ 发送给 CS.

● $Proof(CT, chal)\rightarrow\{P, tag\}$

CS 收到挑战 $chal$ 后, 计算 $T=\prod_{i\in I}T_i^{v_i}$, $\widehat{e}=\sum_{i\in I}v_ie_i$. 随后 CS 将 $P=(T, \widehat{e})$ 以及文件标签 $tag$ 发送给 TA 进行认证.

● $Verify(pp, chal, P, tag)\rightarrow 0$ or 1

(1) TA 验证签名 $SSig_{sk}(fn\|R_{uid})$ 是否正确, 如果验证通过, 解析 $fn$ 以及 $R_{uid}$.

(2) TA 通过等式 $e(T, g)=e\left(\prod_{i\in I}(H_1(fn\|i)^{v_i})u^{\widehat{e}}, R_{uid}Y^{H_0(R_{uid})}\right)$ 验证证明 $P$ 是否正确. 如果等式成立, 算法输出 1, 否则算法输出 0.

● $Access(pp, sk_1, CT)\rightarrow F$ or $\perp$

如果数据访问密钥 $sk_1$ 中的属性无法满足密文 $CT$ 中的访问策略, DO 停止进行访问, 否则对于 $i\in[l]$, DO 计算

常数 $w_i \in \mathbb{Z}_p$ 使得 $\sum_{i\in I} w_i M_i = (1,0,\ldots,0)$, 然后计算

$$B = \frac{e(C_0, K_0)}{\prod_{i\in I}(e(C_{i,1}, K_1)e(C_{i,2}, K_{i,2})e(C_{i,3}, K_{i,3}))^{w_i}}.$$

DO 通过计算 *C*/*B* 获取 *ck*, 随后 DO 通过 *ck* 解密 $e_i$ 获取 $m_i$, 将所有的数据块 $m_i$ 组合成文件 *F*.

● *Share*(*pp*, *uid*, $sk_2$, *uid*′, $sk_2'$)→$sk_2$

(1) 当数据拥有者希望与他人共同维护某文件时, 可以将该文件的所有权共享给其他用户. 令 $DO_1$ 为文件原拥有者, 其身份为 *uid*, 数据审计密钥为 $sk_2=\{R_{uid}, r_{uid}, \sigma_{uid}, h_1, r_1\}$, $DO_2$ 为新的文件拥有者, 其身份为 *uid*′, 数据审计密钥为 $sk_2' = \{R_{uid'}, r_{uid'}, \sigma_{uid'}, h_1', r_1'$. TA 首先验证 $DO_1$ 是否为系统中的合法用户. TA 验证 $h_1 = g^{H_0(R_{uid})}h^{r_1}$ . 如果验证通过, TA 执行后续步骤.

(2) TA 根据 $H_0(R_{uid})+xr_1=H_0(R_{uid'})+xr_2$ 来计算新的随机值 $r_2$, 随后将 $r_2$ 添加到 $DO_2$ 的数据数据审计密钥中, 则 $DO_2$ 新的数据审计密钥为 $sk_2' = \{R_{uid'}, r_{uid'}, \sigma_{uid'}, h_1', r_1', r_2\}$ .

● *Update*(*pp*, *uid*, $sk_2$, *CT*)→*CT*

(1) 为了实现文件动态修改, 本方案提供了 3 种类型的操作, 分别为增加、修改和删除.

(2) 增加: 令 *D*=(*fn*, $\{m_i\}_{i\in[f']}$) 为待增加的文件, 对于 *i*∈[*f*′], DO 使用 *ck* 加密数据块 $m_i$ 获得对称密文 $e_i$, 计算密文 $e_i$ 的标签 $T_i = (H_1(fn\|i)u^{e_i})^{\sigma_{uid}}$ . DO 将增加请求 $req_{\text{add}}$=$\{sk_2, fn, \{e_i, T_i\}_{i\in[f']}\}$发送给 TA. TA 收到请求后, 首先验证 DO 是否拥有该修改数据的所有权, TA 验证 $h_1 = g^{H_0(R_{uid})}h^{r_1}$ 是否成立. 如果验证通过, TA 将 *fn* 以及$\{e_i, T_i\}_{i\in[f']}$发送给 CS, CS 将$\{e_i, T_i\}_{i\in[f']}$添加到对应于 *fn* 的密文中, 则更新后的密文为 *CT*=$\{(\boldsymbol{M}, \rho), C, C_0, \{C_{j,1}, C_{j,2}, C_{j,3}\}_{j\in[l]}, \{e_i, T_i\}_{i\in[f+f']}, tag\}$.

(3) 删除: 令 *D*=(*fn*, $\{m_i\}_{i\in[f']}$) 为待删除的文件, DO 将删除请求 $req_{\text{del}}$=$\{sk_2, fn, [f']\}$发送给 TA. TA 收到请求后, 首先验证 DO 是否拥有该修改数据的所有权, TA 验证 $h_1 = g^{H_0(R_{uid})}h^{r_1}$ 是否成立. 如果验证通过, TA 将 *fn* 以及 [*f*′] 发送给 CS, CS 删除对应密文的 $e_i$ 和 $T_i$, 则更新后的密文为 *CT*=$\{(\boldsymbol{M}, \rho), C, C_0, \{C_{j,1}, C_{j,2}, C_{j,3}\}_{j\in[l]}, \{e_i, T_i\}_{i\in[f-f']}, tag\}$.

(4) 修改: 令 *D*=(*fn*, $\{m_i\}_{i\in[f']}$) 为待修改的文件, 用于替换 CS 存储的旧密文, 对于 *i*∈[*f*′], DO 使用 *ck* 加密数据块 $m_i$ 获得对称密文 $e_i$, 计算密文 $e_i$ 的标签 $T_i = (H_1(fn\|i)u^{e_i})^{\sigma_{uid}}$ . DO 将修改请求 $req_{\text{mod}}$=$\{sk_2, fn, \{e_i, T_i\}_{i\in[f']}\}$发送给 TA. TA 收到请求后, 首先验证 DO 是否拥有该修改数据的所有权, TA 验证 $h_1 = g^{H_0(R_{uid})}h^{r_1}$ 是否成立. 如果验证通过, TA 将 *fn* 以及$\{e_i, T_i\}_{i\in[f']}$发送给 CS, CS 替换对应密文的 $e_i$ 以及 $T_i$, 则更新后的密文为 *CT*=$\{(\boldsymbol{M}, \rho), C, C_0, \{C_{j,1}, C_{j,2}, C_{j,3}\}_{j\in[l]}, \{e_i, T_i\}_{i\in[f]}, tag\}$.

## 5 方案分析

### 5.1 正确性及安全性分析

(1) 云存储审计的正确性

如果 TA、DO 是可信的, CS 是半可信的, 那么任意合法的证明都可以通过 TA 的验证. 由双线性映射 (详见第 2.2 节) 的特性可知, 如果以下验证等式成立, 则本方案可以保证云存储审计的正确性.

$$e(T,g) = e\left(\prod_{i\in I} T_i^{v_i}, g\right) = e\left(\prod_{i\in I}(H_1(fn\|i)u^{e_i})^{\sigma_{uid}v_i}, g\right) = e\left(\prod_{i\in I}(H_1(fn\|i)u^{e_i})^{v_i}, g^{\sigma_{uid}}\right) = e\left(\prod_{i\in I}(H_1(fn\|i)^{v_i}u^{e_iv_i}), g^{\sigma_{uid}}\right)$$

$$= e\left(\prod_{i\in I}(H_1(fn\|i)^{v_i})u^{\sum_{i\in I}e_iv_i}, g^{r_{uid}+xH_0(R_{uid})}\right) = e\left(\prod_{i\in I}(H_1(fn\|i)^{v_i})u^{\widehat{e}}, g^{r_{uid}}g^{xH_0(R_{uid})}\right) = e\left(\prod_{i\in I}(H_1(fn\|i)^{v_i})u^{\widehat{e}}, R_{uid}Y^{H_0(R_{uid})}\right).$$

(2) 错误数据的可检测性

在本文方案中, 如果云服务器中存储的某个文件分为 *n* 个数据块, 其中有 *m* 个损坏的数据块, 当挑战数据块数量为 *c* 时, 检测到这些损坏的数据块的概率至少为 $1-((n-m)/m)^c$. 令 *X* 为挑战数据块中损坏数据块的数量, $P_X$ 为检测到损坏数据块的概率:

$$P_X = P\{X \geqslant 1\} = 1 - P\{X = 0\} = 1 - \frac{n-m}{n} \cdot \frac{n-1-m}{n-1} \cdot \ldots \cdot \frac{n-c+1-m}{n-c+1},$$

即, $P_X \geqslant 1-((n-m)/m)^c$.

(3) 云存储审计的合理性

**定理 1**. 若 DL 假设成立, 则不存在概率多项式时间的攻击者能在没有挑战数据的情况下伪造证明通过 TA 的数据完整性验证.

证明: 如果 CS 在没有存储相应挑战数据的情况下通过了 TA 的数据完整性验证, 那么我们就可以通过构造一个知识抽取器与本文方案进行多次交互来获取正确的挑战数据块, 我们通过一系列游戏来进行证明.

● 游戏 0: 挑战者运行 *Setup* 和 *KeyGen* 来生成公共参数 *pp* 以及用户 *uid* 的私钥$\{sk_1, sk_2\}$, 挑战者公开 *pp*. 攻击者选择某文件的数据块 $m_1, \ldots, m_n$ 提交给挑战者, 挑战者生成相应的文件标签 *tag*. 挑战者向攻击者发起审计挑战, 攻击者返回一个证明 $P = (T, \widehat{e})$. 如果证明能通过挑战者的验证, 则攻击者赢得该游戏.

● 游戏 1: 游戏 1 基本与游戏 0 相同, 唯一的区别在于挑战者通过一个列表记录所有生成的文件标签 *tag*. 如果攻击者在完整性验证中提交了一个并非由挑战者生成的合法 *tag*, 则挑战者拒绝进行验证.

分析: 如果挑战者在游戏 1 中能以不可忽略的优势识别出恶意的完整性验证请求, 那么说明攻击者可以有效地伪造 *SSig* 签名. 这与 *SSig* 签名是一个安全的身份基签名算法相矛盾. 因此, 攻击者无法伪造文件标签 *tag* 中的 *fn* 与验证信息 $R_{uid}$.

● 游戏 2: 游戏 2 基本与游戏 1 相同, 唯一的区别在于挑战者通过一个列表记录所有对攻击者询问的响应. 由于挑战者可以获取攻击者发起的所有询问的内容, 如果挑战者发现聚合签名 $T$ 与 $\prod_{i \in I} T_i^{v_i}$ 不相等, 则说明攻击者获胜.

分析: 假设 $P = (T, \widehat{e})$ 是一个可以通过验证的合法证明, 那么可以推出下列等式是成立的:

$$e(T, g) = e\left(\prod_{i \in I} (H_1(fn \| i)^{v_i}) u^{\widehat{e}}, R_{uid} Y^{H_0(R_{uid})}\right).$$

假设攻击者提供一个伪造的证明 $P' = (T', \widehat{e'})$, 当伪造成功时, 则下列验证等式依然成立:

$$e(T', g) = e\left(\prod_{i \in I} (H_1(fn \| i)^{v_i}) u^{\widehat{e'}}, R_{uid} Y^{H_0(R_{uid})}\right).$$

明显可得 $\widehat{e'} = \widehat{e}$ (否则可说明 $T$=$T'$, 与之前的假设相矛盾). 令 $\Delta\widehat{e} = \widehat{e'} - \widehat{e}$ $(\Delta\widehat{e} \neq 0)$, 如果攻击者能以不可忽略的优势使得挑战者停止游戏, 则我们可以构造一个模拟者打破 CDH 假设.

给定 $g, g^a, w \in \mathbb{G}$, 模拟者的目标为输出 $w^a$. 模拟者选择两个随机数 $a, b \in \mathbb{Z}_p$, 计算 $u$=$g^a w^b$. 规定模拟者的能力与游戏 1 中的挑战者基本一致.

在 *Setup* 和 *KeyGen* 中, 挑战者设置 $Y$=$g^a$, 这表示挑战者不知道 $x$ 和 $\sigma_{uid}$ 的值.

对于挑战 *chal* 中的每一项 $i$, 模拟者选择一个随机数 $r_i \in \mathbb{Z}_p$, 令随机预言机的输出为:

$$H_1(fn \| i) = \frac{g^{r_i}}{g^{ae_i} w^{be_i}},$$

则

$$H_1 = (fn \| i) u^{e_i} = \frac{g^{r_i}}{(g^{ae_i} w^{be_i})} \cdot u^{e_i} = \frac{g^{r_i}}{(g^{ae_i} w^{be_i})} \cdot g^{ae_i} w^{be_i} = g^{r_i}.$$

此时, 模拟者可以计算 $T_i = (H_1(fn \| i) u^{e_i})^{\sigma_{uid}} = (g^{r_i})^{\sigma_{uid}} = (R_{uid} Y^{H_0(R_{uid})})^{r_i}$. 随后计算:

$$\begin{aligned} e(T', g)/e(T, g) &= e(T'/T, g) = e(u^{\Delta\widehat{e}}, R_{uid} Y^{H_0(R_{uid})}) = e((g^a, g^{r_{uid}} Y^{H_0(R_{uid})}) = e(g^{a\Delta\widehat{e}} w^{b\Delta\widehat{e}}, g^{r_{uid}} Y^{H_0(R_{uid})}) \\ &= e(g^{a\Delta\widehat{e}}, g^{r_{uid}} Y^{H_0(R_{uid})}) e(w^{b\Delta\widehat{e}}, g^{r_{uid}} Y^{H_0(R_{uid})}) = e(g, g^{a\Delta\widehat{e} r_{uid}} Y^{a\Delta\widehat{e} H_0(R_{uid})}) e(w^{b\Delta\widehat{e} r_{uid}}, g) e(w^{b\Delta\widehat{e}}, Y^{H_0(R_{uid})}) \\ &= e(g, g^{a\Delta\widehat{e} r_{uid}} Y^{a\Delta\widehat{e} H_0(R_{uid})} w^{b\Delta\widehat{e} r_{uid}}) e(w^{b\Delta\widehat{e}}, Y^{H_0(R_{uid})}). \end{aligned}$$

由此可得:

$$e\left((T'/T) \cdot g^{-a\Delta\widehat{e} r_{uid}} Y^{-a\Delta\widehat{e} H_0(R_{uid})} w^{-b\Delta\widehat{e} r_{uid}}, g\right) = e\left(w^{b\Delta\widehat{e}}, Y^{H_0(R_{uid})}\right) = e(w, Y)^{b\Delta\widehat{e} H_0(R_{uid})} = e(w^a, g)^{b\Delta\widehat{e} H_0(R_{uid})}.$$

由以上等式可知, $w^a = \left((T'/T) \cdot g^{-a\Delta\widehat{e} r_{uid}} Y^{-a\Delta\widehat{e} H_0(R_{uid})} w^{-b\Delta\widehat{e} r_{uid}}\right)^{1/b\Delta\widehat{e} H_0(R_{uid})}$. 也就是说, 我们解决 CDH 问题概率与 $b\Delta\widehat{e} H_0(R_{uid})$

$=0$ 的概率等价. 而 $b\Delta\widehat{e}H_0(R_{uid})=0$ 的概率为 $1/q$, 因此是可忽略. 这表明如果存在一个攻击者以不可忽略的优势赢得游戏 1 与游戏 2, 那么我们可以构造一个模拟者来解决 CDH 问题.

● 游戏 3: 游戏 3 基本与游戏 2 相同, 唯一的区别在于如果挑战者发现聚合密文 $\widehat{e}$ 与预期的聚合结果不同, 则挑战者宣布攻击者获胜.

分析: 假设 $P=(T,\widehat{e})$ 是一个可以通过验证的合法证明, 那么可以推出下列等式是成立的:

$$e(T,g)=e\left(\prod_{i\in I}(H_1(fn\|i)^{v_i})u^{\widehat{e}},R_{uid}Y^{H_0(R_{uid})}\right).$$

假设攻击者提供了一个伪造的证明 $P'=(T',\widehat{e'})$, 当伪造成功时, 则下列验证等式依然成立:

$$e(T',g)=e\left(\prod_{i\in I}(H_1(fn\|i)^{v_i})u^{\widehat{e'}},R_{uid}Y^{H_0(R_{uid})}\right).$$

根据游戏 2 的证明, 我们知道 $T=T'$. 令 $\Delta\widehat{e}=\widehat{e'}-\widehat{e}\ (\Delta\widehat{e}\neq 0)$, 如果攻击者能以不可忽略的概率使得挑战者停止游戏, 则我们可以构造一个模拟者打破 DL 假设.

给定 $g,w\in\mathbb{G}$, 模拟者的目标为输出 $x$ 使得 $w=g^x$. 模拟者选择两个随机数 $a,b\in\mathbb{Z}_p$, 计算 $u=g^aw^b$, 由之前的等式可推出 $e\left(\prod_{i\in I}(H_1(fn\|i)^{v_i})u^{\widehat{e}},R_{uid}Y^{H_0(R_{uid})}\right)=e(T,g)=e(T',g)=e\left(\prod_{i\in I}(H_1(fn\|i)^{v_i})u^{\widehat{e'}},R_{uid}Y^{H_0(R_{uid})}\right)$, 可以得出 $u^{\widehat{e}}=u^{\widehat{e'}}$, 因此 $1=u^{\Delta\widehat{e}}=(g^aw^b)^{\Delta\widehat{e}}=g^{a\Delta\widehat{e}}w^{b\Delta\widehat{e}}$, 从而得出 $\Delta\widehat{e}=0 \mod q$, 即 $\widehat{e'}=\widehat{e} \mod q$, 但这与前述假设冲突. 此时, 我们可以通过计算 $w=g^{-a\Delta\widehat{e}/b\Delta\widehat{e}}=g^{-a/b}$ 来解决 DL 问题. 由于 $b$ 仅有 $1/q$ 的概率为 0, 是可忽略的, 我们有 $1-1/q$ 的概率找到 DL 问题的解, 这与 DL 问题是困难问题矛盾. 令一个攻击者赢得游戏 2 与游戏 3 的概率分别为 $P_2$ 和 $P_3$, 如果$|P_2-P_3|$是不可忽略的, 则我们可以通过上述方法构造一个模拟者来解决 DL 问题. 因此上述游戏之间的差异均为可忽略的.

综上, 我们可以构造一个知识抽取器, 通过选择 $c$ 个不同的系数 $v_i$ ($i\in I$, $|i|=c$) 并对数据块 $m_i$ ($i\in I$, $|i|=c$) 进行 $c$ 次挑战来获取挑战的数据块 $m_i$. 此时, 知识抽取器可以获取 $c$ 个关于 $m_i$ 的独立线性等式. 通过解这些等式, 知识抽取器可以计算并生成 $m_i$. 这表明如果 CS 可以通过 TA 的验证, 那它必然完整地存储了用户的数据.

(4) 数据机密性

由于本文方案采用了文献 [23] 中的密文策略属性基加密方案. 因此如果文献 [23] 中的方案在 $q$–1 假设下满足选择明文不可区分安全. 那么本文方案同样在 $q$–1 假设下满足明文不可区分安全, 即可以保证数据的机密性.

### 5.2 理论分析

(1) 功能对比

在本节中, 我们将本文方案与文献 [18,21] 中的方案从方案功能的角度进行对比, 对比情况如表 1 所示. 本文方案与文献 [18,21] 均支持数据批量验证, 但只有本文方案支持数据所有权“一对多”共享, 文献 [18,21] 仅支持数据所有权“一对一”转移. 在数据共享方面, 本文方案和文献 [21] 都基于密文策略属性基加密机制实现了细粒度的数据共享, 然而文献 [18] 仅支持群数据共享, 因此本文方案与文献 [21] 更加实用. 在文献 [18] 中, 所有的数据均以明文的形式存储在云服务器中, 任意可以访问云服务器的用户都可以访问数据的内容, 无法保证数据的机密性. 相比而言, 本文方案与文献 [21] 均支持对加密后的数据进行完整性验证, 可以更好地保障方案的安全性. 本文方案限制只有数据的拥有者才能对数据完整性发起验证, 因此相比于文献 [21] 可以防止由于数据完整性验证导致的隐私泄露. 此外, 本文方案与文献 [21] 还提供了数据动态修改的功能, 以便于数据拥有者对自己上传的数据进行调整. 总体而言, 本文方案的功能更加全面, 实用性更高.

表 1 方案功能对比

| 方案 | 批量验证 | 数据共享 | 加密验证 | 隐私保护 | 动态修改 | 所有权转移模式 |
|---|---|---|---|---|---|---|
| 文献[18] | √ | 群数据共享 | × | × | × | 一对一 |
| 文献[21] | √ | 细粒度共享 | √ | × | √ | 一对一 |
| 本文方案 | √ | 细粒度共享 | √ | √ | √ | 一对多 |

(2) 计算开销对比

在本节中, 我们将本文方案与文献 [18,21] 中的方案从计算开销的角度进行对比. 令 $G$ 表示一次群 $\mathbb{G}$ 上的幂运算, $G_T$ 表示一次群 $\mathbb{G}_T$ 上的幂运算, $e$ 表示一次双线性配对计算, $k$ 表示用户密钥中属性的数量, $n$ 表示文件中数据块的数量, $p$ 表示完整性验证中数据块的数据, $t$ 表示密文中属性的数量, 变色龙哈希函数采用第 2.3 节提供的方案, 属性基加密采用文献 [23] 中的方案. 在表 2 中, 我们分别对比了密钥生成、服务器验证、完整性验证、数据访问以及所有权转移/共享 5 个部分的计算开销, 我们为计算开销最优的方案标注了底色以便于区分.

表 2　方案计算开销对比

| 计算内容 | 文献[18] | 文献[21] | 本文方案 |
|---|---|---|---|
| 密钥生成 | $8G$ | $(5k+2)G$ | $(4k+8)G$ |
| 服务器验证 | $7nG+2ne$ | $4ne$ | $(3n+2)G+2e$ |
| 完整性验证 | $(p+7)G+2e$ | $pG+2e$ | $(p+3)G+2e$ |
| 数据访问 | — | $G+3G_T+(2t+1)e$ | $(4t+1)G_T+(3t+1)e$ |
| 所有权转移/共享 | $4G$ | $(4t+2)G+2G_T$ | $3G$ |

在密钥生成部分, 本文方案与文献 [21] 方案的计算开销分别为 $(4k+8)G$ 和 $(5k+2)G$, 而文献 [18] 仅为 $8G$. 这是由于本文方案与文献 [21] 的密钥包括数据审计密钥和数据访问密钥两部分, 其中数据审计密钥负责对数据完整性进行验证, 数据访问密钥用于实现对数据的细粒度访问控制. 文献 [18] 由于不提供数据细粒度共享的功能, 因此密钥生成时只需计算数据完整性验证部分的数据审计密钥. 在本文方案中, 生成数据审计密钥的计算开销为 $4G$, 生成数据访问密钥的计算开销为 $(4k+4)G$, 因此在数据审计密钥生成方面, 本文方案比文献 [18] 更加高效. 与文献 [21] 相比, 本文方案的计算开销更低且文献 [21] 中的数据访问控制方案是基于一个二叉树构建的, 系统中用户数量受二叉树叶子结点数量的限制存在上限, 存在实用性方面的不足. 服务器验证的计算开销是指服务器收到数据用户上传的数据时需要在存储前对收到的数据进行验证, 本文方案与文献 [18,21] 方案在这部分的计算开销分别为 $(3n+2)G+2e$、$7nG+2ne$ 和 $4ne$. 由于本文方案支持批量验证, 因此云服务器在验证时的计算开销更少. 类似地, 在完整性验证的过程中, 本文方案与文献 [21] 都采用了批量验证的方式, 因此计算开销均优于文献 [18]. 虽然在数据访问部分, 本文方案的计算开销为 $(4t+1)G_T+(3t+1)e$, 高于文献 [21] $G+3G_T+(2t+1)e$ 的计算开销, 但本文方案在执行数据所有权共享时的计算开销仅为 $3G$, 而文献 [21] 执行数据所有权转移时的计算开销为 $(4t+2)G+2G_T$, 与密文中属性的数量相关. 这是由于文献 [21] 不仅需要更新用户的密钥, 同时还需要对系统中的密文进行更新, 而本文方案借助变色龙哈希函数, 免去了密文更新的步骤, 仅需常数级的计算开销即可在不影响数据完整性验证的情况下共享数据的所有权, 与用户所拥有的数据数量无关. 总体而言, 本文方案在不影响计算效率的情况下提供了更加丰富的功能, 使方案更加实用, 在数据验证以及数据有权共享时的计算开销相比于其他方案有显著的优化.

### 5.3 实验分析

在本节中, 我们将本文方案分别与文献 [18] 和文献 [21] 进行了实验对比. 实验基于 Charm 框架 [24]使用 Python 3 编程实现, 其中双线性映射采用 Charm 框架中的默认曲线“SS512”. 实验的平台为一台 Macbook Pro 笔记本电脑, 处理器为 Inter Core i7 (2.7 GHz), 内存为 16 GB. 此外, 本文方案中密文策略属性基加密使用的访问策略通过 AND 相互连接, 实验中采用毫秒作为运行时间的统计单位.

(1) 与文献 [18] 的对比情况

由表 1 功能对比的情况可知, 文献 [18] 中的方案仅提供了对明文数据进行完整性验证的功能, 而本文方案在提供数据完整性验证功能基础上, 还基于密文策略属性基加密实现了数据加密和细粒度共享的功能. 出于公平性考虑, 我们在实验对比时, 仅关注这两个方案在数据标签生成、数据标签验证、审计挑战生成、证明生成以及证明验证阶段的运行时间. 实验测试文件的大小为 2 MB, 分为 10 000 个数据块. 我们在数据标签生成和验证阶段统计了运行时间随数据块在 200–2 000 的变化情况. 从图 2(a)、(b) 可以看出, 本文方案在数据标签生成阶段的计算开销与文献 [18] 基本一致, 但在数据标签验证阶段, 得益于所采用的聚合计算机制, 本文方案在计算开销方面具

有较为明显的优势. 在图 2(c)、(d)、(e) 中我们分别统计了在总共 10 000 块数据块的条件下, 本文方案与文献 [18] 在审计挑战生成、证明生成以及证明验证阶段运行时间随挑战数据块从 100–1 000 的变化情况. 本文方案在审计挑战生成阶段的计算开销大约比文献 [18] 中的方案多 2 ms 左右, 而在证明的生成和验证阶段, 本文方案与文献 [18] 中的方案基本保持一致. 总体而言, 上述结果充分说明本文方案采用的聚合计算机制有效地提升了数据验证的执行效率.

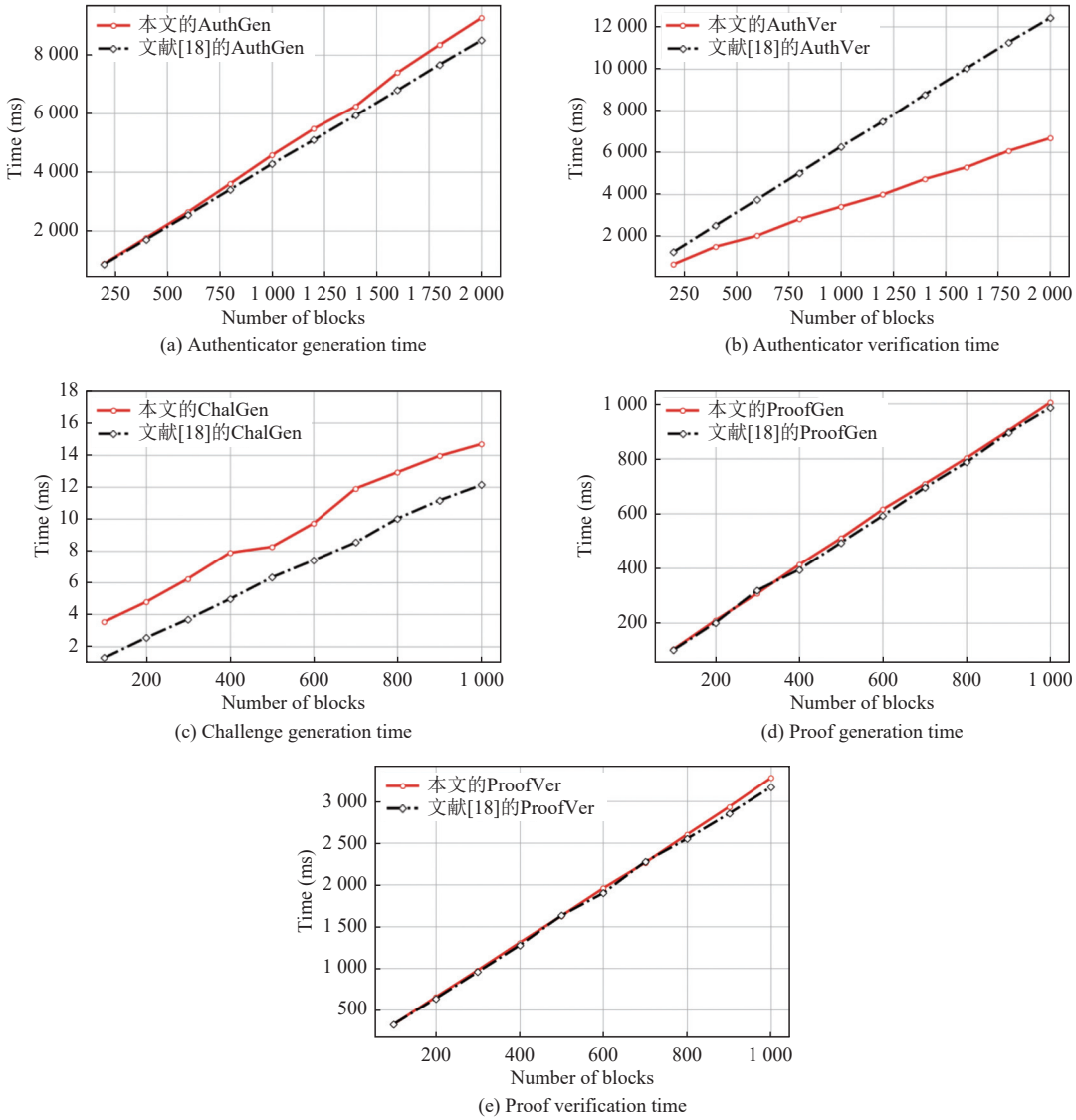


图 2　本文方案与文献 [18] 方案的时间开销对比

(2) 与文献 [21] 的对比情况

由于本文方案与文献 [21] 都实现了基于密文策略属性基加密的细粒度数据共享和对密态数据的完整验证, 因此我们主要关注这两个方案在密钥生成、服务器验证、完整性验证、数据访问以及所有权转移/共享阶段的运行时间. 我们在密钥生成和数据访问阶段均统计了运行时间随属性数量从 10–80 变化的情况, 在这几个阶段中我们约定数据块的数量为 10. 根据图 3(a)、(d) 所示, 本文方案与文献 [21] 在密钥生成阶段和数据访问阶段的运行

时间随属性数量线性增加, 本文方案在密钥生成阶段的计算开销更低, 但数据访问阶段的计算开销高于文献 [21]. 在图 3(b)、(c)、(e) 中, 我们分别统计了本文方案与文献 [21] 在服务器验证、完整性验证以及所有权转移/共享阶段运行时间随数据块数量从 10 变化到 80 的情况, 此时我们约定系统中属性的数量为 10. 在完整性验证阶段, 由于本文方案在服务器验证和完整性验证阶段均采用了批量验证的方法, 因此时间开销基本一致. 虽然文献 [21] 在云服务器验证阶段的计算开销更占优势, 但在所有权转移/共享阶段, 本文方案仅需更新用户的密钥即可, 无需对存储的密文进行任何修改, 因此运行时间为常数. 由表 2 可知, 文献 [21] 在所有权转移阶段的时间开销与密文中属性数量有关, 虽然实验中约定密文中属性数量为常数 10, 但文献 [21] 的时间开销依然比本文方案更高.

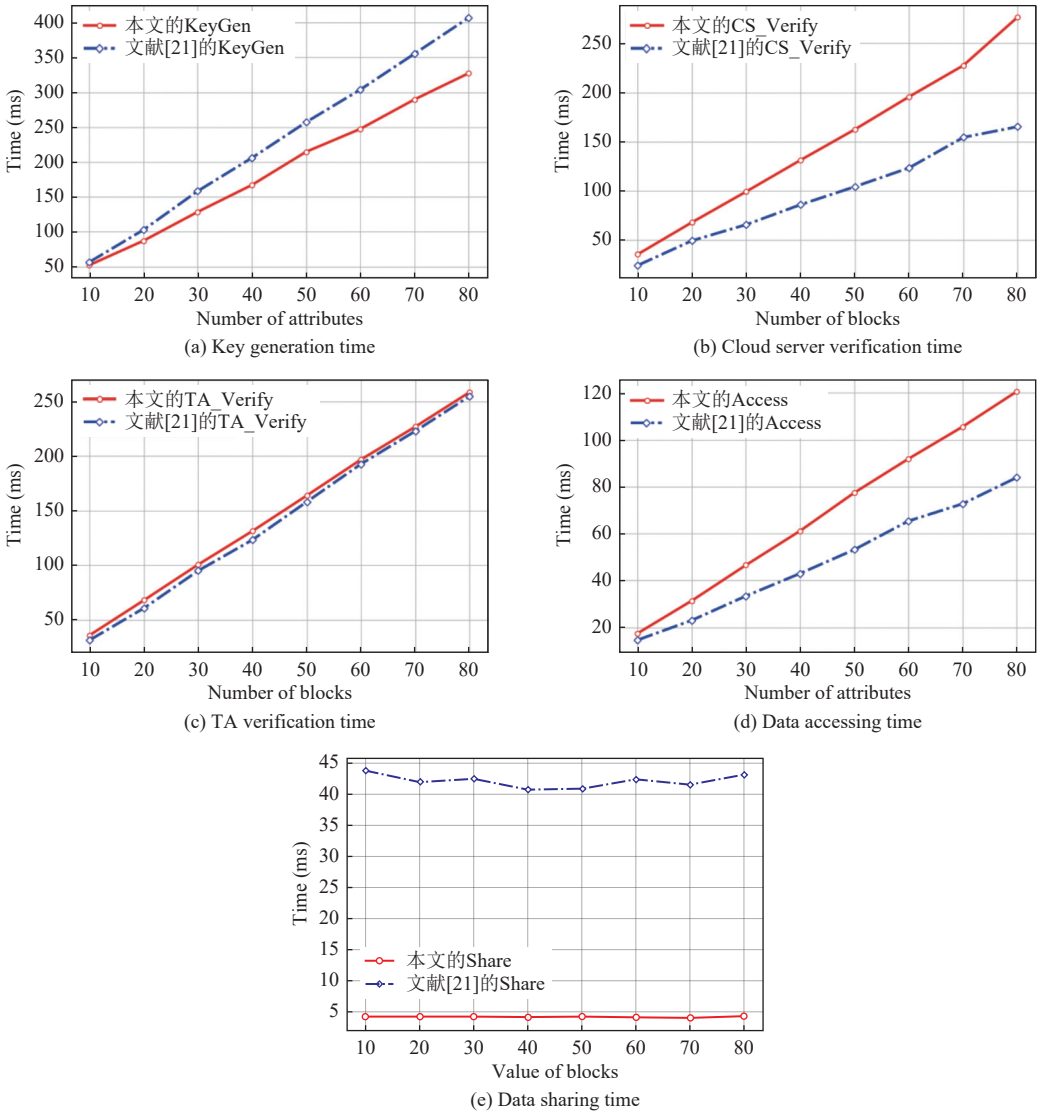


图 3 本文方案与文献 [21] 方案的时间开销对比

## 6 总 结

为了缓解数据审计过程中的计算开销大的问题, 提高云存储审计方案的实用性, 本文提出了一种支持高效数

据所有权共享的动态云存储审计方案. 在该方案中, 我们采用批量验证的方式将大量的双线性配对计算替换为计算开销较小的乘法计算, 有效地降低了数据审计过程中的计算开销. 通过结合属性基加密和变色龙哈希函数, 本文方案还实现了细粒度数据访问控制、密态数据审计、高效的数据所有权共享以及动态数据修改功能. 安全性分析表明, 本文方案可以保证数据审计的安全性以及数据的机密性. 同时, 在计算开销方面, 本文方案与其他方案相比占有一定优势.

**References:**


[1] Han J, Li YP, Yu Y, Ding Y. Cloud auditing scheme with dynamic revocation of users and real-time updates of data. Ruan Jian Xue Bao/Journal of Software, 2020, 31(2): 578–596 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5633.htm [doi: 10.13328/j.cnki.jos.005633]

[2] Ning JT, Huang XY, Susilo W, Liang KT, Liu XM, Zhang YH. Dual access control for cloud-based data storage and sharing. IEEE Trans. on Dependable and Secure Computing, 2022, 19(2): 1036–1048. [doi: 10.1109/TDSC.2020.3011525]

[3] Su Y, Li YP, Yang B, Ding Y. Decentralized self-auditing scheme with errors localization for multi-cloud storage. IEEE Trans. on Dependable and Secure Computing, 2022, 19(4): 2838–2850. [doi: 10.1109/TDSC.2021.3075984]

[4] Ning JT, Huang XY, Wei LF, Ma JH, Rong J. Tracing malicious insider in attribute-based cloud data sharing. Chinese Journal of Computers, 2022, 45(7): 1431–1445 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2022.01431]

[5] Shang T, Zhang F, Chen XY, Liu JW, Lu XX. Identity-based dynamic data auditing for big data storage. IEEE Trans. on Big Data, 2021, 7(6): 913–921. [doi: 10.1109/TBDATA.2019.2941882]

[6] Li X, Liu SP, Lu RX, Zhang XS. On security of an identity-based dynamic data auditing protocol for big data storage. IEEE Trans. on Big Data, 2021, 7(6): 975–977. [doi: 10.1109/TBDATA.2020.3026318]

[7] Duan HY, Du YF, Zheng LQ, Wang C, Au MH, Wang Q. Towards practical auditing of dynamic data in decentralized storage. IEEE Trans. on Dependable and Secure Computing, 2023, 20(1): 708–723. [doi: 10.1109/TDSC.2022.3142611]

[8] Wang HQ, Liu Z, He DB, Li JG. Identity-based provable data possession scheme for multi-source IoT terminal data in public cloud. Journal on Communications, 2021, 42(7): 52–60 (in Chinese with English abstract). [doi: 10.11959/j.issn.1000-436x.2021077]

[9] Fu Y, Li QD, Zhang ZH, Gao TG. Data integrity verification scheme for privacy protection and fair payment. Journal of Computer Research and Development, 2022, 59(6): 1343–1355 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.20210023]

[10] Ning JT, Xu J, Liang KT, Zhang F, Chang EC. Passive attacks against searchable encryption. IEEE Trans. on Information Forensics and Security, 2019, 14(3): 789–802. [doi: 10.1109/TIFS.2018.2866321]

[11] Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D. Provable data possession at untrusted stores. In: Proc. of the 14th ACM Conf. on Computer and Communications Security. Alexandria: ACM, 2007. 598–609. [doi: 10.1145/1315245.1315318]

[12] Juels A, Kaliski Jr BS. Pors: Proofs of retrievability for large files. In: Proc. of the 14th ACM Conf. on Computer and Communications Security. Alexandria: ACM, 2007. 584–597. [doi: 10.1145/1315245.1315317]

[13] Ateniese G, Di Pietro R, Mancini LV, Tsudik G. Scalable and efficient provable data possession. In: Proc. of the 4th Int'l Conf. on Security and Privacy in Communication Netowrks. Istanbul: ACM, 2008. 9. [doi: 10.1145/1460877.1460889]

[14] Wang Q, Wang C, Ren K, Lou WJ, Li J. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. on Parallel and Distributed Systems, 2011, 22(5): 847–859. [doi: 10.1109/TPDS.2010.183]

[15] Rao L, Zhang H, Tu TF. Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated Merkle hash tree. IEEE Trans. on Services Computing, 2020, 13(3): 451–463. [doi: 10.1109/TSC.2017.2708116]

[16] Tian H, Chen YX, Chang CC, Jiang H, Huang YF, Chen YH, Liu J. Dynamic-hash-table based public auditing for secure cloud storage. IEEE Trans. on Services Computing, 2017, 10(5): 701–714. [doi: 10.1109/TSC.2015.2512589]

[17] Yang K, Jia XH. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans. on Parallel and Distributed Systems, 2013, 24(9): 1717–1726. [doi: 10.1109/TPDS.2012.278]

[18] Zhang Y, Yu J, Hao R, Wang C, Ren K. Enabling efficient user revocation in identity-based cloud storage auditing for shared big data. IEEE Trans. on Dependable and Secure Computing, 2020, 17(3): 608–619. [doi: 10.1109/TDSC.2018.2829880]

[19] Su Y, Sun JM, Qin J, Hu JK. Publicly verifiable shared dynamic electronic health record databases with functional commitment supporting privacy-preserving integrity auditing. IEEE Trans. on Cloud Computing, 2022, 10(3): 2050–2065. [doi: 10.1109/TCC.2020.3002553]

[20] Yuan YL, Zhang JB, Xu WS, Li Z. Identity-based group user data integrity verification scheme. Ruan Jian Xue Bao/Journal of Software, 2022, 33(12): 4758–4770 (in Chinese with English abstract). [doi: 10.13328/j.cnki.jos.006360]

[21] Yeh LY, Chiang PY, Tsai YL, Huang JL. Cloud-based fine-grained health information access control framework for lightweight IoT devices with dynamic auditing and attribute revocation. IEEE Trans. on Cloud Computing, 2018, 6(2): 532–544. [doi: 10.1109/TCC.2015.2485199]

[22] Wang JW. Research on data controllability and data availability of attribute-based encryption for the cloud environment [Ph.D. Thesis]. Yangzhou: Yangzhou University, 2023 (in Chinese with English abstract). [doi: 10.27441/d.cnki.gyzdu.2023.000090]

[23] Rouselakis Y, Waters B. Practical constructions and new proof methods for large universe attribute-based encryption. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security. Berlin: ACM, 2013. 463–474. [doi: 10.1145/2508859.2516672]

[24] Akinyele JA, Garman C, Miers I, Pagano MW, Rushanan M, Green M, Rubin AD. Charm: A framework for rapidly prototyping cryptosystems. Journal of Cryptographic Engineering, 2013, 3(2): 111–128. [doi: 10.1007/s13389-013-0057-3]

## 附中文参考文献:

[1] 韩静, 李艳平, 禹勇, 丁勇. 用户可动态撤销及数据可实时更新的云审计方案. 软件学报, 2020, 31(2): 578–596. http://www.jos.org.cn/1000-9825/5633.htm [doi: 10.13328/j.cnki.jos.005633]

[4] 宁建廷, 黄欣沂, 魏立斐, 马金花, 荣静. 支持恶意用户追踪的属性基云数据共享方案. 计算机学报, 2022, 45(7): 1431–1445. [doi: 10.11897/SP.J.1016.2022.01431]

[8] 王化群, 刘哲, 何德彪, 李继国. 公有云中身份基多源 IoT 终端数据 PDP 方案. 通信学报, 2021, 42(7): 52–60. [doi: 10.11959/j.issn.1000-436x.2021077]

[9] 富瑶, 李庆丹, 张泽辉, 高铁杠. 支持隐私保护和公平支付的数据完整性验证方案. 计算机研究与发展, 2022, 59(6): 1343–1355. [doi: 10.7544/issn1000-1239.20210023]

[20] 袁艺林, 张建标, 徐万山, 李铮. 基于身份的组用户数据完整性验证方案. 软件学报, 2022, 33(12): 4758–4770. [doi: 10.13328/j.cnki.jos.006360]

[22] 王经纬. 云环境下属性基加密方案数据可控性与可用性研究 [博士学位论文]. 扬州: 扬州大学, 2023. [doi: 10.27441/d.cnki.gyzdu.2023.000090]

**殷新春**(1962－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为密码学, 高性能计算.

**宁建廷**(1988－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为密码学与数据安全, 区块链安全, 隐私保护技术.

**王经纬**(1993－), 男, 博士, 主要研究领域为属性基加密.