支持审计者更换和数据动态的云数据完整性审计方案

杨 帆, 袁艺林, 张邓凡, 李子臣

北京印刷学院 信息工程学院 北京 中国 102600

摘要 由于用户希望在传输和存储过程中的云端数据没有被篡改或损坏, 因此需要定期审计云端数据的完整性。为便于用户摆脱云端数据的审计负担, 目前, 云端数据完整性验证研究中通常采用审计者替代用户执行审计工作的模式。然而, 单一且固定的审计者, 可能因资源限制、服务到期、安全风险等原因导致审计失败, 从而造成用户权益受损。针对此问题, 本文提出了支持审计者更换和数据动态的云数据完整性审计方案。首先, 本文提出以雾节点充当审计者, 雾节点的低延迟、高自主、离线支持等特性可使审计服务不受地理、时间或其他客观因素影响, 且支持在审计服务到期或维修故障时随时退出审计服务。其次, 为了防止退出审计服务的雾节点利用原有签名和密钥发起不正当的行为, 方案加入标签与密钥更新技术, 根据代替原有雾节点继续执行审计服务的新雾节点的信息重新生成标签与密钥。另外, 保证云端数据新鲜性有助于提升用户的体验感, 因此方案引入一种数据结构--可调节分治表(Adjustable divide and conquer table, ADCT), 用于辅助用户完成数据动态操作。安全性分析证明方案在 Computational Diffie-Hellman(CDH)、Discrete Logarithm(DL)假设下具备安全性, 实现了标签不可伪造与审计者更换, 实验评估表明本方案的总计算开销较小, 综合性能更为优越, 兼具安全性与高效性。

关键词 云存储; 完整性验证; 雾节点; 审计者更换; 数据动态

中图法分类号 TP309 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2025.05.12

Cloud Data Integrity Audit Scheme That Supports Auditor Replacement and Data Dynamics

YANG Fan, YUAN Yilin, ZHANG Dengfan, LI Zichen

School of Information Engineering, Beijing Institute of Graphic Communication, Beijing 102600, China

Abstract Since users want data in the cloud to not be tampered with or corrupted during transmission and storage, the integrity of the data in the cloud needs to be audited regularly. In order to facilitate users to break free from the audit burden of cloud data, currently, in research on cloud data integrity verification, auditors are usually used to perform audit work on behalf of users. However, a single and fixed auditor may fail audits due to resource constraints, service expiration, security risks, and other reasons, resulting in damage to user rights. In response to this issue, this article proposes a cloud storage data integrity audit scheme that supports auditor replacement and dynamic data updates. Firstly, this article proposes using fog nodes as auditors. The characteristics of fog nodes such as low latency, high autonomy, and offline support enable audit services to be unaffected by geographical, temporal, or other objective factors, and can be exited at any time when the service expires or maintenance malfunctions occur. Secondly, in order to prevent the fog nodes that have exited the audit service from using the original signatures and keys to initiate improper behaviors, the label and key update technology is added to regenerate the label and key according to the information of the new fog node that continues to execute the audit service instead of the original fog node. In addition, ensuring the freshness of cloud data helps to enhance the user experience. Therefore, the solution introduces a data structure - Adjustable Divide and Conquer Table (ADCT) used to assist users in completing data dynamic operations. The safety analysis proves that the scheme has security under the assumption of Computational Diffie-Hellman(CDH) and Discrete Logarithm(DL), and realizes the non-forgery of the label and the replacement of the auditor, and the experimental evaluation shows that the total computing overhead of the scheme is small, the comprehensive performance is superior, and it has both security and efficiency.

Key words cloud storage; integrity verification; fog nodes; auditor replacement; data dynamics

通讯作者: 袁艺林, 博士, 讲师, Email: yuanyilin@bigc.edu.cn。

收稿日期: 2023-09-26; 修改日期: 2023-12-26; 定稿日期: 2025-04-10

1 引言

云计算是一种通过互联网将计算资源, 如服务器、存储空间和软件, 提供给用户的计算模型, 它具有灵活、可扩展和按需使用的能力, 使用户能够通过网络去访问和管理数据以及应用程序, 而无需自己拥有并维护基础设施。海量数据时代的到来, 为满足用户对数据存储和管理的需求, 云计算技术逐渐衍生出了一项重要的附加服务——云存储。云存储解决了传统存储方式所面临的诸多问题, 例如硬盘容量受限、备份和恢复困难等。越来越多的组织和个人将数据存储到云端, 以享受云存储带来的便利和高效。尽管云存储兼具可扩展性、安全性、自动备份等优势, 但所带来的数据安全问题却日渐凸显。由于用户在云存储环境下失去对数据的直接控制权, 不可信的云服务商可能出于商业利益的考虑, 删除或篡改使用频率较低的数据。为了应对这一安全问题, 云数据完整性[1-2]审计应运而生。云数据完整性审计服务, 将定期对云存储中的数据进行检查和验证, 确保数据的安全性、完整性和准确性, 这项技术提供了独立可靠的数据验证机制, 用户无需直接参与便可确保云端数据的完整性, 便可在发现异常时及时接收到反馈。

支持审计者更换有助于提升用户对云服务的满意度, 并可更好地保障数据安全, 且更换审计者可以减少资源集中的情况, 确保审计工作的高效进行。此外, 可通过调整审计团队的构成, 确保审计人员具备相关技术和领域的知识, 从而更好地适应不同的审计需求。然而, 在传统的云数据安全审计方案中, 只有一个固定的审计者或审计团队负责对云中数据进行审计和监控, 此方式可确保审计过程的一致性和稳定性。但是, 出于资源限制、合同到期、安全风险等原因, 可能会面临需更换现有审计者或引入新的审计者来继续对数据执行后续审计工作的问题。现有方案大多以提高计算难度与增加通讯开销为代价, 并不侧重于审计者更换问题, 且难以保证审计服务的标准和一致性, 进而造成数据泄漏和滥用, 影响审计质量。传统方案中审计者更换容易因审计标准或方法的不同而导致审计结果存在差异, 又可能因审计服务涉及用户敏感数据, 造成更换过程中的信息泄露或滥用。目前有方案提出利用雾节点[3-7]担任审计者, 可在一定程度上解决上述问题。相比于地域受限的第三方审计者, 雾节点具有更强的灵活性和可拓展性, 适合代替用户与云服务器进行审计工作交互。当旧雾节点退出审计服务时, 新雾节点可借助相应的更新手段, 保证用户数据的完整性与安全性, 同时实现审计服务的无缝衔接。

实际应用时, 支持云端数据更新的审计方案有助于消除数据错误、降低数据损坏或丢失的风险。数据更新可以及时反映用户和组织最新的业务状态, 确保数据的一致性, 既可便于后续数据安全审计的展开, 也可增强方案的实用性与可移植型。然而, 由于用户失去了存储在云端数据的控制权, 使得数据操作的难度增大。为了简化繁琐的数据更新操作, 设计一种辅助查询数据位置、便于后续更新的数据结构尤为重要。

基于上述考虑, 本文在云存储环境下, 提出一种支持安全高效的数据更新与审计者更换的云安全审计方案。主要工作归纳为以下三点:

(1) 保证云端数据的新鲜性有助于提升用户对云存储服务的满意度。但因数据的控制权转移至云服务器, 当用户有数据修改需求时, 如何保证数据的同步更新需要考虑。为了满足用户修改外包数据的需求, 本文提出一种新的数据结构——可调节分治表(Adjustable divide and conquer table, ADCT)来辅助用户完成云端数据的动态更新。借助于 ADCT, 可简化繁琐的数据动态操作, 实现云端文件的更新、插入与删除。

(2) 方案引入兼具高效性与安全性的雾节点担任审计方。借助于雾节点, 可在本地完成审计而无需将数据发送到云服务提供商, 减少数据传输延迟, 同时降低用户在本地端的开销。当旧雾节点需要退出审计服务时, 因其不受地域限制, 可直接选择一个新的雾节点替代。为保证更换安全性, 方案增加了云服务商对雾节点身份认证的设计, 可确保退出审计服务的旧雾节点不能获得新的签名密钥, 有效防止外来攻击者冒充旧雾节点向云服务器发出完整性挑战。

(3) 安全性分析证明本方案满足正确性与不可伪造性。性能分析进一步证明了标签和密钥更新阶段仅需少量的计算与通信开销, 便可达到确保审计数据和相关信息的无缝转移, 更具优越性。

1.1 相关工作

2007 年, Ateniese 等人[8]提出了“可证明数据占有”(Provable data possession, PDP)方案, 既支持概率验证, 也可保证不可信云服务器上的数据完整性。同年, Shacham 和 Waters[9]提出了可检索性证明(Proof of retrievability, POR)方案, 该方案利用双线性映射产生同态认证器并支持公开验证。基于 PDP 和 POR 方案, 更多公开审计方案被提出。2015 年, Li 等人[10]

提出一种可实现审计者更换的云安全审计方案, 通过引入名为 SecCloud 的审计实体在用户上传云数据之前生成标签, 但该方案增加了数据上传时的计算开销。而付安民等人[11]通过多种密钥共同计算的签名设计了一种支持密钥更新与审计者更换的云安全审计方案, 通过雾节点代替第三方审计员(Third party auditor, TPA)进行审计服务, 但这种方式并不适用频繁更新云存储数据的场景。Yang 等人[12]通过使用雾节点和智能合约进行了审计者替换, 将审计过程生成的证据存储在区块链中, 防止各实体的不诚实行为。Qin 等人[13]利用了雾计算的特点和优势进行解决审计者更换的问题, 但对于物理设备的要求较高。2019 年, Xue 等人[14]通过利用区块链中的随机数构建不可预测的随机挑战, 使用户只需批量验证审计证据, 便可确保存储在云中的数据的完整性, 从而防止恶意的审计者伪造审计证据以欺骗用户。Zhang 等人[15]利用区块链技术, 还提出审计者将每个验证的结果记录到区块链上的交易中, 利用区块链在时间上对于交易的敏感, 使得用户能够检查审计者是否在规定时间内执行验证, 防止审计者拖延。此外, Huang 等人[16]为了解决数据所有者和云服务提供商之间的信任问题, 提出了一种用于云数据存储的协作审计区块链框架, 利用区块链中所有共识节点替换单一的第三方审计者执行审计服务并永久记录, 从而防止实体相互欺骗。而 Fan 等人[17]通过设计一种新型以太坊智能合约取代 TPA, 使得任何人都可以从以太坊获得审计结果而不必担心审计者更换。在引入区块链技术来保证审计服务的安全性的研究中, Jia 等人[18]利用去中心化的区块链网络替换中心化 TPA 进行审计服务同时利用去中心化自治组织防止审计方与恶意的区块链矿工勾结。Weng 等人[19]构建了一个轻量级的隐私保护可委派存储证明方案, 通过加快标签生成过程更新标签以实现审计者更换。Wang 等人[20]通过设计基于区块链的公共云存储审计公平支付智能合约采用区块链取代 TPA, 但以上几种利用区块链技术进行审计者更换的方案常常会引入更大代价且不便于云存储数据进行动态更新。为支持数据动态更新, Wang 等人[21]提出了基于构造块标记认证的动态公开审计方案。Erway 等人[22]提出了一种完全动态审计方案,利用基于秩的认证跳表来支持存储数据的可证明更新。2015 年, Wang 等人[23]在原方案的基础上提出了 Panda, 该方案支持多任务批量审计, 极大地提高了公共审计效率。然而以上几种支持动态更新的方案中设计的结构并不便于用户进行检查且会产生额外的开销。为了解决用户及公私钥变化的问题, Han 等人[24]提出一种轻量级的支持用户可动态撤销及存储数据动态更新的云审计方案。Ma 等人[25]采用变色龙认证树提出了一种新的审计方案, 显著降低了数据动态操作的计算开销且有效防止恶意的云服务商的欺诈行为。Singh 等人[26]提出一种第三方审计协议, 通过修改后的变色龙身份验证树对存储在云上的数据执行高效的块级和细粒度动态数据更新操作。

上述方案虽然能够很好地解决云存储过程中的数据完整性问题, 但多数只考虑了云服务器方在不可信情况下的安全性, 几乎不考虑审计者更换时的安全风险, 仍然存有一定局限性。因此, 设计一种支持数据更新和审计者更换的审计方案尤为重要。

1.2 组织

本文其余部分组织如下。第 2 节介绍预备知识。第 3 节给出具体方案设计。第 4 节中分析方案的正确性和安全性。第 5 节进行性能评估。第 6 节进行全文总结。

2 预备知识

本节介绍了方案的系统模型、使用的符号以及相关密码学知识。

2.1 系统模型

方案模型图如图 1 所示, 包含四个实体, 介绍如下:

1. Data Owner(DO): 用户端, 持有大量数据的实体, 希望将数据上载到云服务器进行存储。在本方案中, DO 负责对外包数据完成加密以及将数据上传至云端存储, 并委托雾节点完成云端数据的审计工作, 必要时接收雾节点的审计反馈。

2. Cloud Service Provider(CSP): 提供云存储服务的不可信实体。CSP 在所提方案中负责存储与管理 DO 的外包数据, 响应雾节点的发起的完整性挑战。另外, CSP 内置 ADCT, 可应对 DO 的数据更新的需求, 还可以帮助用户重新计算标签与签名, 完成密钥更新。

3. Key Generation Center(KGC):可信实体, KGC 在所提方案中负责在初始化阶段生成 DO 的身份密钥与雾节点的时间密钥。

4. 雾节点: 担任审计者的实体。本方案引入雾节点作为公开审计阶段的验证者, 其作为用户和云服务器之间的“桥梁”, 接受用户的审计委托, 并在审计阶段与 CSP 交互完成数据完整性校验。另外, 雾节点负责与云服务器进行审计交互以及对分块后的原始文件进行块标签与文件签名的计算。

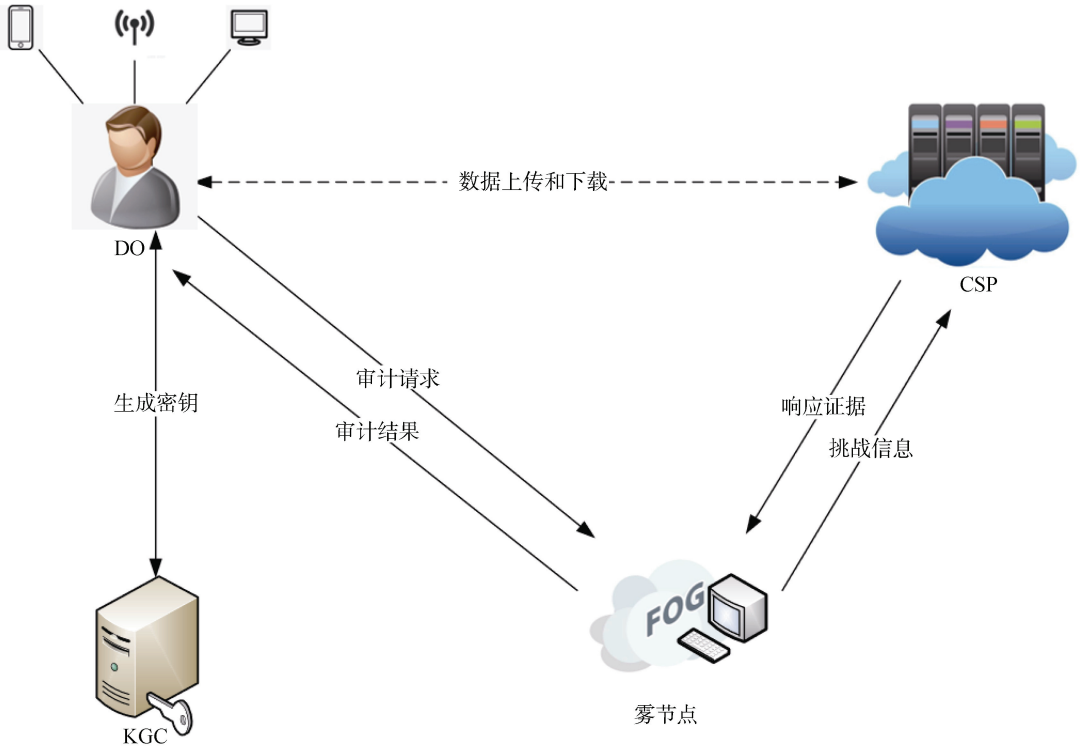

图 1　系统整体结构图

Figure 1　Overall system structure diagram

2.2　符号

本文所使用的符号说明详见表 1。

表 1　符号说明表

Table 1　Notation

符号	含义
$\alpha,\beta,\gamma,\delta$	大整数
G,G_1	乘法循环群
g	群 G_1 的生成元
Z_p^*	非零元素的素数域
H	加密散列函数
pp	系统公开参数
q	一个大素数
Uid_k,Uid	用户身份密钥与身份信息
T_k	时间密钥
σ_i	第 i 个区块的标签, $1\le i\le m$
F_l,Fid_l	雾节点和雾节点标识符
mpk,msk	主公钥和主密钥

2.3　密码学知识

2.3.1　双线性映射

设 G,G_1 为 q 阶的乘法循环群, g 是 G 的生成元。双线性映射 $e:G\times G\to G_1$, 满足下列性质:

(1) 双线性: $\forall u,v\in G$ 且 $\forall a,b\in Z_p^*$, $e(u^a,v^b)=e(u,v)^{ab}$;

(2) 非退化性: $e(g_1,g_2)\neq 1$, 其中 g_1,g_2 分别为 G 的生成元;

(3) 可计算性: 存在一个有效的算法来计算 e ;

2.3.2　安全性假设

计算 Diffie-Hellman 假设。对于未知的 $\forall a,b\in Z_p^*$, 给定 g,g^a 和 g^b 作为输入, 输出 $g^{ab}\in G$ 。

定义 1 (CDH 假设) 概率多项式时间 (Probabilistic polynomial time, PPT)算法在解决 G 中定义的 CDH 问题时的优势可以忽略不计。

$$AdvCDH_A=\Pr[A(g,g^a,g^b)=g^{ab}:a,b\xleftarrow{R}Z_p^*]$$

离散对数假设. 对于未知 $x\in Z_p^*$, 给定 g 和 g^x 作为输入, 输出 x 。

定义 2 (DL 假设) PPT 算法在解决 G 中定义的 DL 问题时的优势可以忽略不计。

$$AdvDL_A=\Pr[A(g,g^x)=x:x\xleftarrow{R}Z_p^*]$$

3　具体方案

3.1　方案设计

Setup: 系统初始化算法, 由 KGC 运行, 以安全参数 k 为输入, 输出系统公共参数 pp 、主公钥 mpk

和主密钥 msk。其中公共参数 pp 与主公钥 mpk 公开, 主密钥 msk 由 KGC 保密存储。

KeyGen: 密钥生成算法, 包含由用户执行的身份密钥生成与由雾节点执行的时间密钥生成两种。其中, 身份密钥生成以公共参数 pp、用户身份信息 Uid 为输入, 输出用户身份密钥 Uid_k。时间密钥生成以公共参数 pp、主公钥 mpk 以及雾节点的身份标识符 Fid_l 作为输入, 输出时间密钥 T_k。

TagGen: 标签生成算法, 由雾节点运行, 以公共参数 pp、用户身份信息 Uid、雾节点的身份标识符 Fid_l 以及时间密钥 T_k 作为输入, 输出块标签和文件签名。随后用户将块标签和文件签名上传至 CSP 并删除本地存储。

ProofGen: 审计证据生成算法, 首先由雾节点以用户授权信息 W 和随机数据块集合 Q 为输入, 产生完整性挑战 $chal$ 并发送至 CSP。随后, CSP 根据挑战信息产生并返回审计证据 P 至雾节点。

ProofVerify: 审计证据验证算法, 由雾节点运行, 当雾节点收到 CSP 的审计证据 P 后通过等式验证, 若验证失败则立即通知用户终止服务。

UpdateKeyAndTag: 密钥与标签更新算法, 以公共参数 pp、主公钥 mpk、用户身份信息 Uid、新雾节点的身份标识符 Fid_l'' 为输入, 输出新的块标签和时间密钥和文件签名, 并将旧雾节点的身份信息和签名删除。

3.2 可调整分治表

在云安全审计过程中, 保持数据的完整性至关重要。通过设计能够实现数据更新的数据结构, 可以有效地记录云存储数据的变化, 确保更新后的数据在审计过程中得到正确的验证, 在防止数据的篡改和损坏的同时简化用户在数据更新时繁琐的操作。此外, 由于每次更新操作都可被明确地记录, 包括操作者的身份信息、操作内容等, 在出现数据异常或安全问题时有助于实现数据的可追溯, 即可追踪到数据更新的源头。因此, 添加 ADCT 有助于减少恶意方非法操作或用户误操作带来的风险, 提高数据操作的可控性。

本方案设计一种新的数据结构——可调整分治表(Adjustable divide and conquer table, ADCT)来辅助完成数据动态操作, 其结构如表 2 所示。

ADCT 存储在 CSP 端, 包含数据的逻辑索引号 N_0、块号 B_n、版本号 V_n、以及数据块存储时间 $Time$。

数据的每次增加、删除或修改均可被如实记录至 ADCT 中, 借助于 ADCT, 除实现数据动态外, 还可回溯数据的更新和修改历史, 获取数据的真实状态。ADCT 各字段的具体解释如下:

1. N_0 示逻辑索引号, 记录文件的逻辑位置。

2.文件上传云端前, DO 为文件进行分块处理, 划分后的数据块用 B_n 表示。

3. V_n 为版本号, 记录数据块的更新次数, 初始值设置为 1, 每进行一次数据更新操作, V_n 增加 1。

4. $Time$ 记录数据块存储时间。每执行一次更新操作, 需根据文件标识符 id 以及更新时间 t, 重新计算 $F_{id}<t>$; 由于 $F_{id}<t>$ 随 t 与 id 变化, 因此可作为该文件的唯一标识符。

表 2 ADCT 示意表

Table 2 ADCT schematic table

ADCT			
N_0	B_n	V_n	$Time$
1	1	1	$F_{id}<t_1>, F_{id}<t_2>$
2	10	1	$F_{id}<t_3>, F_{id}<t_4>, F_{id}<t_5>$
3	3	2	$F_{id}<t_6>$

3.3 具体方案

所提方案包括六个算法:**Setup**, **KeyGen**, **TagGen**, **ProofGen**, **ProofVerify**, **UpdateKeyAndTag**。各算法描述如下:

Setup: KGC 选取随机数 $\alpha \in Z_q^*$, 设置私钥 $msk=(\alpha, ssk)$, 并根据 G_1 的生成元 g 计算公钥 $mpk=g^{\alpha}$。随后, KGC 保密存储 msk, 并公开参数 $pp=(G, G_1, q, g, H, mpk)$。其中 G 与 G_1 为阶数为 q 的乘法循环群, H 为哈希函数。

KeyGen: 密钥生成算法分为身份密钥生成与时间密钥生成, 其中, 身份密钥算法由 KGC 执行, 用于为 DO 产生身份密钥; 时间密钥算法由 KGC 执行, 目的为生成签名密钥对。

1. 身份密钥生成

DO 设置身份标识符 Uid, 并提交身份密钥请求 $req_{Uid}=\{Uid\}$ 至 KGC。KGC 记录 DO 身份标识符, 计算并返回身份密钥 $Uid_k=H(Uid)^{\alpha}$ 给 DO。接收到身份密钥 Uid_k 后, DO 将身份密钥保密存储。

2. 时间密钥生成

本文引入雾节点充当审计者, 当审计过程中产生审计者更换的情形时, 雾节点需重新运行此算法为 DO 产生新时间密钥, 以保证更换审计者的安全

性。DO 根据云存储文件生成 $w \in \{0,1\}^*$，其中包括文件处理的时间和要求。雾节点 F_l （$l \in \{1,2,\cdots,L\}$，L 表示雾节点的个数)发送身份标识符 Fid_l 至 DO。为使得雾节点身份得到用户确认，DO 选择一个随机值 $\beta \in Z_q^*$，重新为该雾节点计算身份标识符 $Fid_l' = g^{\beta}$，并将授权信息 $W = H(w \| Uid \| Fid_l')^{\beta}$ 和新的标识符 Fid_l' 返回给雾节点 F_l，同时转存至 KGC。确保 KGC 存储了经过用户确认的雾节点的身份信息，可有效避免时间密钥分发至恶意的雾节点。雾节点 F_l 的身份标识符经过用户确认后也变为 Fid_l'。

接收到 DO 返回的信息后，雾节点 F_l 向 KGC 发出时间密钥请求 $req_t = \{t, W, Fid_l'\}$，希望得到由 KGC 计算生成的时间密钥，t 是根据当前时间产生的时间戳。此时，KGC 根据 DO 之前发送的信息对于雾节点的身份进行验证；验证成功则将时间密钥 $T_k = Uid_k \cdot H(W \| t)^{\alpha}$ 发送至雾节点 F_l。成功接收到时间密钥后，雾节点 F_l 据其生成签名密钥对 $sk = \{\gamma, T_k\}, pk = g^{\gamma}$。

TagGen: DO 首先将原始文件进行对称加密，然后将加密文件划分为 m 个数据块 $b_i (1 \le i \le m)$ 并发送至雾节点 F_l。由于雾节点 F_l 的身份在密钥生成阶段得到用户确认，因此可以完全信任，进行块标签及文件签名的计算。雾节点 F_l 选择一个随机值 $\delta \in G$ 计算块标签 $\sigma_i = (H(Fid_l \| i) \cdot \delta^{b_i})^{\gamma} \cdot T_k$，然后雾节点 F_l 根据签名密钥对计算 $Sig_0 = H(Uid \| Fid_l')^{\gamma}$ 得到文件签名 $Sig = Sig_0 \| W \| Fid_l'$，当雾节点 F_l 退出审计服务时，标签与签名的计算需使用新雾节点的身份标识符与密钥。

最后，将文件签名与块标签上传至 CSP 并删除本地数据。

ProofGen: 雾节点定期代替 DO 执行云数据的完整性校验。雾节点 F_l 产生完整性质询集 $chal = \{W, Q\}$ 并发送至 CSP。$Q = \{i, N_i\}, i \in I$，W 是授权信息，可验证发起挑战者的身份，i 是参与挑战的数据块的下标，N_i 是随机选取的随机数。随后，CSP 根据完整性质询集计算审计证据 $proof = \{\mu, \sigma\}$ 并返回给雾节点，等待雾节点进行验证，其中，$\mu = \sum_{i \in Q} N_i b_i, \sigma = \prod_{i \in Q} \sigma_i^{N_i}$。

ProofVerify: 雾节点 F_l 在收到 CSP 返回的审计证据后通过执行以下等式是否成立执行验证:

$$e(\sigma, g) = e(\prod_{i \in Q} H(Fid_l \| i)^{N_i} \cdot \delta^{\mu}, pk) \cdot e(\prod_{i \in Q} H(Uid)^{N_i}, mpk) \cdot e(\prod_{i \in Q} H(W \| t)^{N_i}, mpk) \quad (1)$$

若成功则代表云数据没有损坏或被恶意篡改，如果在这个阶段中出现错误或验证失败则意味着云存储数据遭到破坏，雾节点 F_l 需立即通知 DO 审计过程出现问题，及时做出决策。

UpdateKeyAndTag: 当雾节点 F_l 主动退出审计或用户发现 F_l 不诚实时，可选择一个符合根据云存储文件生成的 $w \in \{0,1\}^*$ 的新的雾节点 F_l''(身份标识为 Fid_l'')代替原来的雾节点 F_l。然后新雾节点 F_l'' 发送身份标识符 Fid_l'' 到 DO。DO 生成新的授权信息 $W' = H(Uid \| Fid_l'')^{\beta}$，发送 $\{Uid, W'\}$ 至 F_l'' 并提交 $\{Uid, Fid_l'', W'\}$ 到 KGC 更新存储信息。收到 DO 返回信息后，新雾节点 F_l'' 根据信息向 KGC 发出时间密钥请求 $req_t' = \{t', W', Fid_l''\}$。KGC 检查 W' 是否在存储信息中，存在则更新时间密钥并将新的时间密钥 T_k' 返回给 F_l''。随后，新雾节点 F_l'' 删除原来的时间密钥 T_k。

密钥更新完毕后 F_l'' 请求 KGC 计算新因子 $uf = (H(W' \| t') / H(W \| t))^{\alpha}$，并将更新因子发送给 CSP 进行标签更新。CSP 收到更新因子后通过公式进行标签更新，令 $U = H(Fid_l'') \| i) \cdot \delta'^{b_i})^{\gamma'}$，则根据更新因子更新后的标签为 $\sigma_i^{(t')} = \sigma_i' \times uf = H(Fid_l'') \| i) \cdot \delta'^{b_i})^{\gamma'} \cdot Uid_k \cdot H(W \| t)^{\alpha} \cdot (H(W' \| t') / H(W \| t))^{\alpha} = U \times T_k'$，其中 Fid_l''，δ'，γ' 均由新雾节点重新生成。这是由于当新雾节点 F_l'' 加入审计后，需要首先经过 DO 的同意，即产生新的授权信息 W'，获得 DO 给予的授权信息后，才可重新选择 δ 进行更新产生标签 $\sigma_i^{(t')}$。标签与密钥更新完成后由 CSP 存储新的标签，KGC 将新的时间密钥发送至新的雾节点 F_l''，审计服务继续执行。新雾节点 F_l'' 在执行审计服务时会重新运行 **KeyGen** 算法生成密钥，证据验证阶段所需要的 δ, pk 及身份标识符等参数均由新雾节点 F_l'' 重新生成并使用，与旧雾节点无任何关联。

3.4 动态操作

数据动态操作分为“数据更新”、“数据插入”、

“数据删除”三种类型, 而通过 ADCT, 可清晰地查看数据动态操作前后的变化。

3.4.1 数据更新

若需将第 i 个数据块 $f[i]$ 修改为 $f'[i]$, 运用 ADCT, 步骤描述如下(若不做特殊说明, 均默认只选取 ADCT 的某一部分作为示例, m 表示数据块的个数, $1 \leqslant i \leqslant m$)。

1. DO 在 ADCT 中进行搜索, 找到对应的 N_0 为 i 的数据块;

2. 首先 DO 运行对称加密算法对更新数据块加密(例如: AES, Blowfish 等), 将加密结果发送至 CSP, 由 CSP 为加密后数据块进行标签更新计算。之后 CSP 再从 ADCT 读出原数据文件的信息, 结合标签 σ_i' 组成云存储数据文件;

3. DO 生成一个修改信息集 M, 其中包含用户对数据进行更新的要求与用户身份信息, 发给至 CSP;

4. CSP 接受到信息后, 用新的数据块替换旧的数据块但不会改变不涉及更新的数据块。最后 CSP 将数据块版本信息 V_n 进行加 1 操作。

表 3 中对于 $N_0 = 3$ 处数据进行数据更新操作, 变化前 $N_0 = 3$ 处数据如表 2 所示, 在进行修改后原数据的版本号与标识符发生变化如表 3 所示。

表 3 数据更新操作示意表

Table 3 Data update operation diagram

ADCT			
N_0	B_n	V_n	*Time*
1	1	1	$F_{id} < t_1 >, F_{id} < t_2 >$
2	10	1	$F_{id} < t_3 >, F_{id} < t_4 >, F_{id} < t_5 >$
3	3	3	$F_{id} < t_7 >$

3.4.2 数据插入

数据插入操作是把新数据块插入到数据块 $f[i]$ 之后成为 $f[i+1]$,具体的操作步骤如下:

1. DO 在 ADCT 中搜索到数据块 i;

2. DO 对文件块 $f[i+1]$ 先进行加密操作, 然后将加密结果发送至雾节点 F_l 使用标签生成算法计算其标签值 σ_i, 然后将标签上传至 CSP 并删除本地存储;

3. DO 将插入的信息集 M 发送给 CSP 使其将新的数据块插入。

4. CSP 将所在的 ADCT 中 i 位置以后的数据块平移, 同时在 i 的位置后面插入一个空位置 $i+1$;

5. ADCT 中对应的版本号 V_n 进行加 1 操作表 4 对于表 2 所示原有的 ADCT 进行了数据插入操作, 新建了索引号为 4, 数据块号为 6 的数据, 操作后 ADCT 如上表所示。

表 4 数据插入操作示意表

Table 4 Schematic diagram of data insertion operation

ADCT			
N_0	B_n	V_n	*Time*
1	1	1	$F_{id} < t_1 >, F_{id} < t_2 >$
2	10	1	$F_{id} < t_3 >, F_{id} < t_4 >, F_{id} < t_5 >$
3	3	2	$F_{id} < t_6 >$
4	6	1	$F_{id} < t_7 >$

注释 1: 首次进行数据文件云存储也算作数据插入操作, DO 将原始文件加密并分块后插入到 ADCT。

3.4.3 数据删除

数据删除操作即从数据集中删除数据块 $f[i]$, 具体操作步骤如下:

1. DO 在 ADCT 中查找到逻辑索引号为 i 的数据块;

2. DO 将删除信息集 M 发送给 CSP, CSP 删除相应的数据块和标签集;

3. CSP 更改 ADCT 中受到影响的数据块序号;

4. CSP 将当前表中将 i 数据块后面的数据块向前移动一个位置;

表 5 对于表 2 所示原有的 ADCT 进行了数据删除操作, 将原有的索引号为 2, 数据块号为 10 的数据全部删除, 并对后续索引号进行了修改, 操作后 ADCT 如上表所示。

表 5 数据删除操作示意表

Table 5 Schematic diagram of data deletion operation

ADCT			
N_0	B_n	V_n	*Time*
1	1	1	$F_{id} < t_1 >, F_{id} < t_2 >$
2	3	2	$F_{id} < t_6 >$

4 安全性分析

方案安全性分析部分包含正确性、不可伪造性、以及审计者更换的安全性。

定理 1 (正确性) 若 CSP 是诚实的, 那么由 CSP 生成的审计证据可以通过雾节点的完整性检查。

证明. 正确性证明主要是验证 **ProofVerify** 阶段中等式(1)是否成立, 等式(1)左边为 CSP 在收到雾节

点发出的挑战后返回的证据, 等式(1)右边是雾节点根据 DO 提供的信息计算得到的标签证据。根据双线性对特性, 等式(1)的推导证明如下:

$$
\begin{aligned}
e(\sigma,g) &= e(\prod_{i\in Q}\sigma_i^{N_i}, g)\\
&= e(\prod_{i\in Q}((H(Fid_l \| i)\cdot \delta^{b_i})^{\gamma}\cdot T_k)^{N_i}, g)\\
&= e(\prod_{i\in Q}((H(Fid_l \| i)\cdot \delta^{b_i})^{\gamma N_i}\cdot Uid_k^{N_i}\cdot H(W\| t)^{\alpha N_i}), g)\\
&= e(\prod_{i\in Q}(H(Fid_l \| i)\cdot \delta^{b_i})^{\gamma N_i}\cdot \prod_{i\in Q}H(Uid)^{\alpha N_i}\\
&\quad \cdot \prod_{i\in Q}H(W\| t)^{\alpha N_i}, g)\\
&= e(\prod_{i\in Q}H(Fid_l \| i)^{N_i}\cdot \delta^{b_i N_i}, g^{\gamma})\cdot e(\prod_{i\in Q}H(Uid)^{N_i}, g^{\alpha})\cdot\\
&\quad e(\prod_{i\in Q}H(W\| t)^{N_i}, g^{\alpha})\\
&= e(\prod_{i\in Q}H(Fid_l \| i)^{N_i}\cdot \delta^{\mu}, pk)\cdot e(\prod_{i\in Q}H(Uid)^{N_i}, mpk)\cdot\\
&\quad e(\prod_{i\in Q}H(W\| t)^{N_i}, mpk)
\end{aligned}
$$

定理 2 (不可伪造性) 方案基于 CDH 和 DL 困难问题假设进行数学设计。若敌手 A 难以以不可忽略的概率伪造出能够通过雾节点完整性校验的审计证据, 那么本方案具有不可伪造性。

证明. 游戏在敌手 A 与挑战者 C 之间进行, 其中云数据是经过处理的数据块及其对应的块签名。

游戏 0: 首先 C 运行系统初始化算法、密钥生成算法, 生成系统参数和密钥, 并将系统参数发送给 A。C 运行标签生成算法并将块标签与文件签名并返回给 A。随后 C 向 A 发出随机挑战。最后 A 针对 C 发出的挑战返回一个审计证据 P 给 C,若 P 能够以不可忽略的概率通过 C 的验证, 那么 A 获胜, 意味着伪造成功。

游戏 1: 游戏 0 与游戏 1 的不同之处在于, C 将会对云数据进行动态更新从而使标签产生变化, 之后再向 A 发出挑战并接受 A 返回的审计证据。若 A 基于先前的知识计算出的审计证据 $P'=\{\mu',\sigma'\}$ 仍可通过 C 的验证, 意味着 A 获胜; 但基于更新后的数据产生的审计证据为 $P=\{\mu,\sigma\}$, 由于 $P=\{\mu,\sigma\}$ 与 $P'=\{\mu',\sigma'\}$ 不相同, 游戏将终止。

分析: C 在云数据进行动态更新之后会向 A 重新发出挑战, 假定 C 是诚实的, 那么将会返回有效证据 $P=\{\mu,\sigma\}$, 而 A 基于先前的知识企图伪造审计证据 $P'=\{\mu',\sigma'\}$ 通过 C 的验证。由于 $P=\{\mu,\sigma\}$ 被认为是有效证据, 因此可得:

$$
\begin{aligned}
e(\sigma,g) &= e(\prod_{i\in Q}H(Fid_l \| i)^{N_i}\cdot \delta^{\mu}, pk)\cdot\\
&\quad e(\prod_{i\in Q}H(Uid)^{N_i}, mpk)\cdot e(\prod_{i\in Q}H(W\| t)^{N_i}, mpk)
\end{aligned}
$$

如果 A 所伪造的证据能够通过 C 的验证, 首先假设 $\sigma\neq\sigma'$, 则:

$$
\begin{aligned}
e(\sigma',g) &= e(\prod_{i\in Q}H(Fid_l \| i)^{N_i}\cdot \delta^{\mu'}, pk)\cdot\\
&\quad e(\prod_{i\in Q}H(Uid)^{N_i}, mpk)\cdot e(\prod_{i\in Q}H(W\| t)^{N_i}, mpk)
\end{aligned}
$$

若 $\mu=\mu'$, 则 $\sigma=\sigma'$, 与假设相悖, 意味着 A 未拥有正确数据。定义 $\triangle\mu=\mu'-\mu$, 由于 A 未拥有动态更新后的正确数据, 所以 $\triangle\mu$ 一定存在不为 0 的值。又因为存在伪造审计证据 $P'=\{\mu',\sigma'\}$, 可推导:

$$
\begin{aligned}
e(\sigma'/\sigma,g) &= e(\delta^{\triangle\mu}, g^{\alpha})\\
\delta &= (\sigma'\sigma^{-1}\times g^{-\triangle\mu})^{-\alpha\triangle\mu}
\end{aligned}
$$

由于 $\alpha\triangle\mu=0 \bmod q$ 的概率为 $1/q$, $\alpha\in Z_p^*$ 且不为 A 所知, q 是从 Z_p^* 中选择的一个大素数, 因此 C 终止游戏的概率为 $1/q$ 是可忽略不计的。

游戏 2: 游戏 1 与游戏 2 存在一点不同, C 将会保留 A 的查询结果并与 A 一起观察整个交互过程。若 A 基于先前的知识计算出的审计证据 $P'=\{\mu',\sigma'\}$ 通过了 C 的验证而赢得游戏且返回的证据 P' 中的 μ' 与有效证据中的 μ 不同, 则 C 将会终止游戏。

分析: 敌手 A 发送随机挑战给 C, C 在收到挑战后应基于动态更新后的数据生成有效审计证据 $P=\{\mu,\sigma\}$, 然而敌手 A 可能会在未拥有正确数据的情况下伪造审计证据 $P'=\{\mu',\sigma'\}$ 并企图通过 C 的验证。在先前的知识中已经证明了 $\sigma\neq\sigma'$ 的情况, 因此这里提供 $\mu\neq\mu'$ 的证明。定义 $\triangle\mu=\mu'-\mu$, 根据双线性对性质可得, $\delta^{\triangle\mu}=1$。

假定 G 是以大素数 q 为阶的乘法循环群,对于任意 2 个值 $h_1,h_2\in G$,能找到一个值 $\lambda\in Z_p^*$ 使得 $h_1=h_2^{\lambda}$ 成立, 则选定 $\alpha,\beta\in Z_p^*$, 可推导得: $\delta^{\triangle\mu}=h_1^{\alpha\triangle\mu}\times h_2^{\beta\triangle\mu}$。所以给定 h_1,h_2, 当 $\triangle\mu\neq 0$ 或 $\alpha\neq 0$ 时, 存在 DL 问题的一个解 $\lambda=-\beta/\alpha$ 使得 $h_1=h_2^{\lambda}$ 成立, 而 $\alpha\in Z_p^*$ 不被 A 所知道, 因此 $\alpha\triangle\mu=0 \bmod q$ 的概率为 $1/q$, 那么 C 终止游戏的概率也是可忽略的。这意味着如果云服务器能够通过审计验证, 就必须真正存储了用户数据, 本方案存在不可伪造性。

定理 3 (审计者更换安全性) 当旧雾节点退出审计服务后, 将会有新的雾节点代替执行审计服务, 方案可确保雾节点更换的安全性, 意味着旧雾节点无法发起审计挑战, 且无法获取到最新的审计内容。

证明. 在时刻 t' 旧雾节点 F_l 退出审计服务, 那么新的雾节点 F_l'' 将向 DO 发送 Fid_l'' 来获取新的授权信息 W'。新的雾节点 F_l'' 收到授权信息 W' 后向 KGC 提交时间密钥请求 req_t' 以获取根据当前时刻 t' 生成的新的时间密钥 T_k'。经过以上一系列交互后新的雾节点 F_l'' 得到更新后的时间密钥 T_k', 然后 KGC 进行更新因子 uf 的计算并发送给 CSP 进行块标签更新。随着旧雾节点的退出, 新雾节点的加入, 由所提方案 **UpdateKeyAndTag** 算法可知, 新雾节点将重新产生授权信息 W', 时间密钥 T_k' 与块标签 $\sigma_i^{(t')}$。因此, 公式(1)将变换为如下的形式:

$$e(\sigma_i^{(t')},g) = e(\prod_{i\in Q} H(Fid_l'')\|i)^{N_i}\cdot\delta'^{\mu},pk')\cdot e(\prod_{i\in Q} H(Uid)^{N_i},mpk)\cdot e(\prod_{i\in Q} H(W'\|t)^{N_i},mpk) \quad (2)$$

通过公式(2)发现, 由于旧雾节点所持参数已过期, 因此无法在持有旧参数的情况下验证出审计证据的正确性; 另外, 旧雾节点很难通过猜测的方式伪造出可通过正确性验证的新参数。以下给出简要的分析: 为伪造第一项 $e(\sigma_i^{(t')},g)$, 需要 $1/q$ 的概率, 第二项 $e(\prod_{i\in Q} H(Fid_l'')\|i)^{N_i}\cdot\delta'^{\mu},pk')$, 第三项 ..... 均是 $1/q$, 即成功伪造出可通过正确性验证的新参数, 至少需要 $1/q^4$ 的概率, q 是一个大素数, 因此旧雾节点至少需要 $1/q^4$ 的概率才能成功伪造出新参数以通过公式(2)的正确性校验, 成功的概率是可忽略的。

等式(2)的成立, 意味着 CSP 正确执行了块标签的更新, 那么新的块标签可以在相同数据块上进行签名; 而新的签名密钥能够产生有效的审计证据并通过新的雾节点 F_l'' 的验证时, 云数据的完整性可被保证。由于新的授权签名仅被 DO 和新的雾节点 F_l'' 所持有, 而旧的雾节点 F_l 所持有的签名密钥已经无法通过 CSP 的身份验证, 因此方案可确保审计者更换的安全性。

5 性能分析

为保证对比实验的相关性与公平性, 性能分析部分选取同样把雾节点作为审计者进行审计服务的文献[13], 并从理论分析和实验评估两个方面进行了比较。表 6 给出本节所使用的符合及其含义。

表 6 实验操作符号及含义

Table 6 Experimental operation symbols and meanings

符号	含义
P	一次双线性配对运算
Exp	一次幂指数运算
Mul	一次点乘运算
$\|G\|$	群 G,G_1 中的元素的大小
$\|q\|$	有限域 Z_q^* 中的元素的大小
$Hash$	一次哈希运算
c	被挑战的数据块个数
$\|t\|$	系统时间 t

5.1 理论分析

5.1.1 计算开销

表 7 给出了本方案与文献[13]在不同阶段的计算开销的比较。本方案中, 在标签生成阶段, 对于拥有 c 个数据块的文件需要执行两次乘运算、一次模幂运算和一次哈希运算, 因此标签生成阶段的代价为 $2cMul+cExp+cHash$。文献[13]在标签生成阶段的计算开销为 $2P+cMul+4cExp+cHash$。在证据生成阶段, 本方案中雾节点随机选取 c 个数据块生成完整性挑战 $chal=\{W,Q\}$ 发送到 CSP, 由 CSP 计算 $\mu=\sum_{i\in Q} N_i b_i, \sigma=\prod_{i\in Q}\sigma_i^{N_i}$, 可知产生审计证据需要 $(2c-1)Mul+cExp$。文献[13]在证据生成阶段则需要为每个数据块进行一次乘运算和一次模幂运算, 因此代价为 $cMul+cExp$。在证据验证阶段雾节点收到 CSP 返回后的审计证据进行验证, 为验证等式(1)是否成立, 雾节点在该阶段需耗费的计算开销为 $5P+(c+1)Mul+(c+3)Exp+3Hash$。文献[13]也需要根据审计证据进行验证, 在证据验证阶段的代价为

表 7 计算开销比较

Table 7 Comparison of computational expenses

阶段	本方案	文献[13]
标签生成	$2cMul+cExp+cHash$	$2P+cMul+4cExp+cHash$
证据生成	$(2c-1)Mul+cExp$	$cMul+cExp$
证据验证	$5P+(c+1)Mul+(c+3)Exp+3Hash$	$3P+cMul+(c+1)Exp+2Hash$
标签与密钥更新	$2P+Mul$	–

$3P+cMul+(c+1)Exp+2Hash$。在标签与密钥更新阶段, 雾节点需要花费 2 次配对运算验证时间密钥的正确性, 还需要通过更新因子进行一次乘运算重新生成标签与密钥, 因此需要花费 $2P+Mul$。文献[13]方案中并没有涉及标签与密钥更新的计算, 因此该阶段没有计算开销。

5.1.2 通信开销

表 8 给出了本方案与文献[13]在不同阶段的通信开销的比较。本方案的通信开销包括标签生成开销、证据验证开销与标签与密钥更新开销。标签生成阶段的通信开销本方案为 $(c+1)|G|$, 文献[13]为 $c(|G|+|q|)$。文献[13]中证据验证的通信开销为 $2|G|+|q|$, 本方案中证据验证的通信开销为 $3|G|+|q|$。在标签与密钥生成阶段, 文献[13]由于使用智能合约代替第三方审计者, 不涉及标签与密钥的更新, 因此通信开销为 0, 本方案所需开销为 $2|G|+|t|$。

表 8 通信开销比较

Table 8 Comparison of communication costs

阶段	本方案	文献[13]
标签生成	$(c+1)\|G\|$	$c(\|G\|+\|q\|)$
证据验证	$3\|G\|+\|q\|$	$2\|G\|+\|q\|$
标签与密钥更新	$2\|G\|+\|t\|$	–

5.2 实验评估

实验基于 Java Pairing-Based Cryptography (JPBC)函数库编写, 实验环境为 2.80 GHz Intel(R) Core(TM) i7-7700 处理器和 16.0 GB 内存, 选择 A 型双线性对, 2 个乘法循环群分别为 160 阶和 256 基域, 即 $|q|=160$ 与 $|G|=512$。假设 $|t|=64$。设置外包文件的大小为 20M, 划分为 1,000,000 个数据块。实验中各阶段结果均为进行 20 轮取平均值。

在实验部分, 本方案将与文献[13]进行一系列的比较, 数据块的数量设置为从 0 递增至 1,000, 以 100 为递增单位。在标签生成阶段, 本方案与文献[13]标签生成的计算开销都随着数据块数 c 呈线性增长, 但如图 2 所示, 本方案的计算开销要低于文献[13]。在证据生成阶段, 如图 3 所示, 本方案与文献[13]的计算开销几乎一样, 这与之前计算开销的理论分析一致。在证据验证阶段, 随着被挑战数据块数量的增加, 两个方案的计算开销均呈现线性增加的趋势, 但由于文献[13]中证据验证阶段雾节点和用户端都会调用智能合约进行验证, 因此产生的额外开销大大增加了验证时间, 如图 4 所示, 本方案的计算开销低于文献[13]。

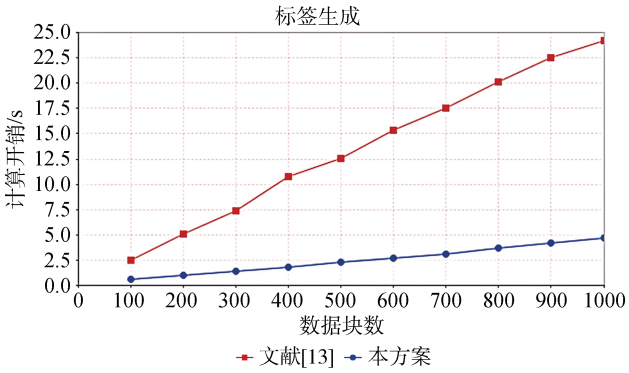

图 2 标签生成阶段计算开销

Figure 2 Calculation overhead of tag generation

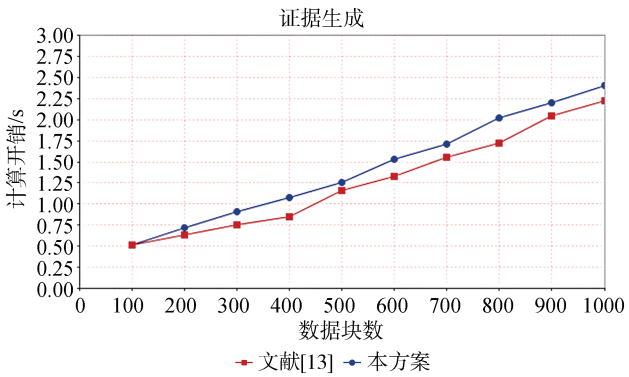

图 3 证据生成阶段计算开销

Figure 3 Calculation overhead of proof generation

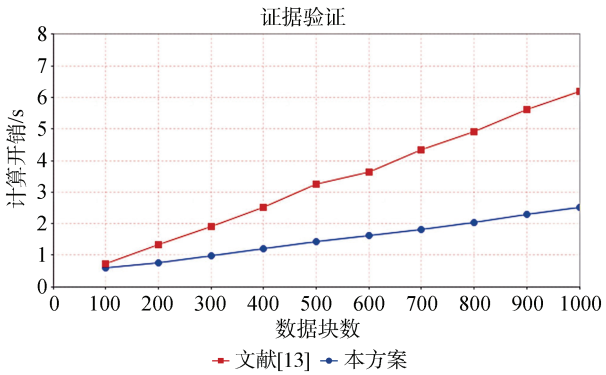

图 4 证据验证阶段计算开销

Figure 4 Calculation cost of proof verification

总体而言, 对比文献[13], 本方案在只证据生成阶段的计算开销略高于文献[13], 虽然本方案中在审计者更换时, 标签与密钥更新阶段会产生额外的计算开销。但相较于文献[13]在标签生成阶段与证据验证阶段因调用智能合约而产生的计算开销, 本方案的效率更高。并且本方案在标签与密钥生成阶段的开销只需要进行配对运算与数据块数量并无关联, 而且标签与密钥的更新所带来的开销由 CSP 承担,

用户不需要消耗任何成本。因此针对审计者更换所带来的一系列问题时, 本方案是高效可行的, 且更加适用于设备受限但使用云存储的用户。

6 结论

针对云存储审计中的审计者更换问题, 本文提出一个支持数据动态更新与审计者更换的云存储审计方案。本文由云服务器承担复杂的计算操作而减少用户端的成本, 同时将时间戳与用户身份信息与雾节点设备信息联系到一起对标签与密钥进行更新以实现访问控制, 确保在雾节点充当审计者进行更换时, 原审计者退出服务时, 新的审计者可以继续行使审计服务, 避免已经退出服务的审计者可能带来的威胁。并且利用雾节点设备的特性减少在审计者更换时所带来的复杂计算等一系列问题。安全分析表明本方案满足正确性、不可伪造性、审计者更换。理论分析与实验评估进一步证实了本方案在密钥更新与标签更新阶段引入的开销是少量且高效的。

参考文献

[1] Ezhil Arasi V, Indra Gandhi K, Kulothungan K. Auditable Attribute-Based Data Access Control Using Blockchain in Cloud Storage[J]. *The Journal of Supercomputing*, 2022, 78(8): 10772-10798.

[2] Tian J F, Wang H N, Wang M. Data Integrity Auditing for Secure Cloud Storage Using User Behavior Prediction[J]. *Computers & Security*, 2021, 105: 102245.

[3] Wang C Y, Wang D, Duan Y H, et al. Secure and Lightweight User Authentication Scheme for Cloud-Assisted Internet of Things[J]. *IEEE Transactions on Information Forensics and Security*, 2023, 18: 2961-2976.

[4] Song Z S, Ma H, Zhang R, et al. Everything under Control: Secure Data Sharing Mechanism for Cloud-Edge Computing[J]. *IEEE Transactions on Information Forensics and Security*, 2023, 18: 2234-2249.

[5] Chang J Y, Xu M Z, Xue R. Public Auditing Protocol with Dynamic Update and Privacy-Preserving Properties in Fog-to-Cloud-Based IoT Applications[J]. *Peer-to-Peer Networking and Applications*, 2022, 15(4): 2021-2036.

[6] Lavanya P, Rani K S, Chaitanya B. Efficient Auditing Scheme for Secure Data Storage in Fog[C]. *Advances in Computational Intelligence and Informatics*, 2024: 335-342.

[7] Al Muhtadi J, Alamri R A, Khan F A, et al. Subjective Logic-Based Trust Model for Fog Computing[J]. *Computer Communications*, 2021, 178: 221-233.

[8] Ateniese G, Burns R, Curtmola R, et al. Provable Data Possession at Untrusted Stores[C]. *The 14th ACM Conference on Computer and Communications Security*, 2007: 598-609.

[9] Shacham H, Waters B. Compact Proofs of Retrievability[M]. Advances in Cryptology - ASIACRYPT 2008. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008: 90-107.

[10] Li J W, Li J, Xie D Q, et al. Secure Auditing and Deduplicating Data in Cloud[J]. *IEEE Transactions on Computers*, 2016, 65(8): 2386-2396.

[11] Zhou L, Chen Z Z, Fu A M, et al. Cloud Secure Auditing Scheme Supporting Key Update and Auditor Replacement[J]. *Journal of Computer Research and Development*, 2022, 59(10): 2247-2260.
(周磊, 陈珍珠, 付安民, 等. 支持密钥更新与审计者更换的云安全审计方案[J]. *计算机研究与发展*, 2022, 59(10): 2247-2260.)

[12] Yang X D, Wang X X, Li X X, et al. Decentralized Integrity Auditing Scheme for Cloud Data Based on Blockchain and Edge Computing[J]. *Journal of Electronics & Information Technology*, 2023, 45(10): 3759-3766.
(杨小东, 王秀秀, 李茜茜, 等. 基于区块链和雾计算的去中心化云端数据完整性审计方案[J]. *电子与信息学报*, 2023, 45(10): 3759-3766.)

[13] Qin B L, Luo Y S, Zheng F L, et al. Research of Key Technologies of Big Data Security Based on Fog Computing[C]. *2019 IEEE 3rd Advanced Information Management*, *Communicates*, *Electronic and Automation Control Conference*, 2019: 1295-1303.

[14] Xue J T, Xu C X, Zhao J N, et al. Identity-Based Public Auditing for Cloud Storage Systems Against Malicious Auditors *via* Blockchain[J]. *Science China Information Sciences*, 2019, 62(3): 32104.

[15] Zhang Y, Xu C X, Lin X D, et al. Blockchain-Based Public Integrity Verification for Cloud Storage Against Procrastinating Auditors[J]. *IEEE Transactions on Cloud Computing*, 2021, 9(3): 923-937.

[16] Huang P, Fan K, Yang H Z, et al. A Collaborative Auditing Blockchain for Trustworthy Data Integrity in Cloud Storage System[J]. *IEEE Access*, 2020, 8: 94780-94794.

[17] Fan K, Bao Z J, Liu M X, et al. Dredas: Decentralized, Reliable and Efficient Remote Outsourced Data Auditing Scheme with Blockchain Smart Contract for Industrial IoT[J]. *Future Generation Computer Systems*, 2020, 110: 665-674.

[18] Shu J G, Zou X, Jia X H, et al. Blockchain-Based Decentralized Public Auditing for Cloud Storage[J]. *IEEE Transactions on Cloud Computing*, 2022, 10(4): 2366-2380.

[19] Yang A J, Xu J, Weng J, et al. Lightweight and Privacy-Preserving Delegatable Proofs of Storage with Data Dynamics in Cloud Storage[J]. *IEEE Transactions on Cloud Computing*, 2021, 9(1): 212-225.

[20] Wang H, Qin H, Zhao M H, et al. Blockchain-Based Fair Payment Smart Contract for Public Cloud Storage Auditing[J]. *Information Sciences*, 2020, 519: 348-362.

[21] 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.

[22] Erway C C, Küpçü A, Papamanthou C, et al. Dynamic Provable Data Possession[J]. *ACM Transactions on Information and System Security*, 2015, 17(4): 1-29.

[23] Wang C, Chow S S M, Wang Q, et al. Privacy-Preserving Public Auditing for Secure Cloud Storage[J]. *IEEE Transactions on Computers*, 2013, 62(2): 362-375.

[24] Li S, Han J G, Tong D Y, et al. Redactable Signature-Based Public

Auditing Scheme with Sensitive Data Sharing for Cloud Storage[J]. *IEEE Systems Journal*, 2022, 16(3): 3613-3624.

[25] Hou G P, Ma J F, Liang C, et al. Efficient Audit Protocol Supporting Virtual Nodes in Cloud Storage[J]. *Transactions on Emerging Telecommunications Technologies*, 2021, 32(5): e3911.

[26] Singh A P, Pasupuleti S K. Optimized Public Auditing and Data Dynamics for Data Storage Security in Cloud Computing[J]. *Procedia Computer Science*, 2016, 93: 751-759.

杨帆 于 2022 年在南昌大学信息安全专业获得学士学位。现在北京印刷学院网络空间安全专业攻读硕士学位。研究领域为云计算、区块链。Email: yangf0219@163.com

袁艺林 于 2022 年在北京工业大学计算机科学与技术专业获得工学博士学位。现在北京印刷学院信息工程学院讲师。研究领域为云计算、云安全与区块链。Email: yuanyilin@bigc.edu.cn

李子臣 于 1999 年获北京邮电大学密码学博士学位。现任北京印刷学院信息工程学院教授。研究领域包括信息安全、数字水印、数字签名、密码学。Email: lizc2020@163.com

张邓凡 于 2021 年在西安电子科技大学信息与计算科学专业获得学士学位。现在北京印刷学院攻读硕士学位。研究领域为云安全。Email: 15291437403@163.com