基于区块链的动态多云多副本数据完整性审计方法研究

王晨旭[1),2)] 孙一帆[1)] 柳博洋[1)] 师金宸[1)] 沈彦成[1)] 王 伟[1),2)]

[1)](西安交通大学软件学院 西安 710049)

[2)](西安交通大学智能网络与网络安全教育部重点实验室 西安 710049)

摘 要 随着云计算技术的快速发展和云存储服务的日益普及,众多用户选择将数据托管至云存储平台。然而,数据一旦外包,便存在失去控制权的风险,尤其是对数据安全性有较高要求的用户,需定期验证云上数据的完整性。传统的数据完整性审计方法依赖于第三方审计员来执行数据审查,不可避免地引入了第三方信任问题。为了增强数据的可靠性,部分用户会创建数据的多个副本,并将其分散存储于不同的服务器上。然而,现有的多云多副本数据完整性审计算法存在计算效率低下的问题。针对上述挑战,本文提出了一种基于区块链的动态多云多副本数据完整性审计方法。该方法通过在区块链上部署智能合约,自动执行数据审计并提供可信的审计结果,有效解决了第三方信任问题。通过优化审计算法中的标签生成机制,显著减少了群上乘法和幂运算的次数,并将验证过程中的累乘操作简化为累加操作,从而降低了算法的计算成本。此外,本研究还对所提模型进行了严格的安全性分析和大量的实验验证,并与其他数据完整性审计方法在不同参数设置下进行了比较,充分证明了所提出方法在安全性和有效性方面的优越性。

关键词 区块链;云存储;多云多副本存储;数据完整性审计

中图法分类号 TP309 **DOI号** 10.11897/SP.J.1016.2025.00497

Blockchain-Based Dynamic Multi-Cloud Multi-Replica Data Integrity Auditing

WANG Chen-Xu[1),2)] SUN Yi-Fan[1)] LIU Bo-Yang[1)] SHI Jin-Chen[1)]
SHEN Yan-Cheng[1)] WANG Wei[1),2)]

[1)](School of Software Engineering, Xi'an Jiaotong University, Xi'an 710049)

[2)](MOE Key Lab of Intelligent Network and Network Security, Xi'an Jiaotong University, Xi'an 710049)

Abstract With the rapid development of cloud computing and the widespread popularity of cloud storage, many users store their data on cloud storage servers. However, once the data is outsourced, there is a risk of losing control of the data. Users requiring high data security need to regularly check the integrity of the data on the cloud. Traditional data integrity auditing schemes usually rely on third-party auditors, leading to third-party trust issues. In addition, some users generate multiple replicas of their data and store them on different servers to enhance data reliability. However, existing multi-cloud multi-replica data integrity auditing algorithms are significantly deficient in terms of computational efficiency. To address the above problems, this

收稿日期:2024-05-24;在线发布日期:2024-12-10。本文的研究工作受到国家重点研发计划(2021YFF0900904)、国家自然科学基金项目(62272379, T2341003, U22B2019, U21A20463 , U22B2027, U23A20304)、陕西省自然科学基金(2021JM-018)、中央高校基本科研业务费(xzy012023068)、北京市基金-区块链(M23019)的资助。**王晨旭**(通信作者),博士,副教授,中国计算机学会(CCF)会员,主要研究领域为图大数据挖掘、网络与数据安全、区块链等,Email: cxwang@mail.xjtu.edu.cn. **孙一帆**,硕士,主要研究领域为网络安全、区块链等。**柳博洋**,硕士,主要研究领域为隐私计算、区块链等。**师金宸**,硕士,主要研究领域为隐私计算、区块链等。**沈彦成**,硕士,主要研究领域为数据治理与深度学习等。**王 伟**,博士,教授,中国计算机学会(CCF)会员,主要研究方向为Web安全、安卓平台安全、入侵检测。

paper proposes a dynamic multi-cloud multi-replica data integrity auditing scheme based on blockchain. By deploying smart contracts on the blockchain for data auditing, it can provide trustworthy results and effectively solves the trust problem. By optimizing the tag generation mechanism in the auditing algorithm, it significantly reduces the number of group multiplication operations and power operations in the tag generation process. It also transforms a large number of group multiplication operations into simpler addition operations in the verification process, significantly reducing the computational overhead. Besides, this paper carries out a rigorous security analysis and extensive experiments. The experimental results obtained by comparing with other data integrity auditing methods under different parameters verify the security and effectiveness of the proposed method.

Keywords blockchain; cloud storage; multi-clouds and multi-replica storage; data integrity auditing

1 引 言

随着信息技术的快速发展，全球数据总量呈现加速增长的趋势，预计到2025年将高达每日463TB[1]。企业和组织在日常运营中积累了海量数据，涵盖了财务、销售、人力资源、研发等各个方面，传统的数据存储方式需要昂贵的本地存储设备和专业的维护技能，给企业带来了显著的经济压力。云计算技术的兴起，尤其是云存储服务的普及，为数据存储提供了新的解决方案。云存储以其硬件冗余、可扩展性和跨平台访问等优势，为用户提供了灵活、经济的数据存储选择[2]。然而，随着数据存储的外包，用户对数据的控制权也随之减弱，存在云服务提供商未能透明地处理数据丢失或删除的可能。这些情况可能源于物理设备故障、黑客攻击或为节省存储空间而删除不活跃数据[3-4]。尽管这类事件的发生概率较低，但一旦发生，对用户造成的经济损失可能是巨大的。历史上的云服务故障案例，如2011年亚马逊云服务器崩溃导致用户部分数据永久丢失，2018年腾讯云故障导致前沿数控技术公司的数据丢失，2023年盛大云工程师因为误操作导致用户存储的数据丢失等，都凸显了云存储安全性的重要性。

为确保云存储数据的安全性，用户需定期进行数据完整性审计。目前，一种普遍的审计方法是通过可信的第三方审计员运用数据完整性审计算法，该方法通过随机抽样而非下载全部原始数据来进行审计。据理论推算，若假设1%的数据遭到篡改，随机选取360个数据块即可以95%的概率检测到篡改；选取460个数据块，检测概率可提升至99%[5]。然而，找到一个既受用户信任又不受云服务提供商影响的第三方审计员存在一定难度[6]，一旦审计员与云服务提供商勾结，提供虚假审计报告，将对用户造成重大损失。因此，亟须解决用户与云服务提供商间的信任问题。区块链技术以去中心化、可追溯和不可篡改等特性，为解决多个行业存在的信任问题提供了有效方案[7-8]。区块链技术可作为一个可信的第三方审计员，参与数据完整性审计，有效解决信任问题。众多学者提出了基于区块链的数据完整性审计模型。尽管现有方法主要将审计结果记录于区块链，为争议提供证据，但尚未充分发挥区块链智能合约的潜力。理论上，通过在区块链上部署智能合约，可完全取代第三方审计员，实现自动化审计过程。

为了降低数据丢失或损坏的风险，众多用户采取将关键数据制作多个副本并分散存储于不同云存储服务器的策略。然而，现有的多云多副本数据完整性审计算法存在计算效率低下的问题。具体而言，这些算法在初始化阶段需要为每一份数据副本生成相应的数据标签，并在验证阶段对所有云存储服务器提交的副本证明进行同步验证。这一过程带来了显著的计算开销，尤其是在处理GB级别的数据时，初始化阶段的耗时可能超过一小时。这种延迟严重降低了用户体验，并限制了这些算法在大规模云数据完整性审计场景中的应用潜力。为解决以上问题，本文提出了一种基于区块链的多云多副本数据完整性审计方法。本文主要贡献如下：

(1) 提出了一种基于区块链的多云多副本数据完整性审计方案，设计并部署了相应的智能合约，为用户和云存储服务器提供可信的审计结果，有效解决了传统第三方审计中存在的信任问题。

(2) 提出了一种改进的多云多副本数据完整性审计算法，通过优化标签的生成方式，显著减少了群

上乘法和幂运算的操作次数。同时，将数据完整性验证过程中的累乘操作简化为累加操作，大幅降低了计算开销。

（3）对所提出算法进行了严格的安全性分析，充分证明了所提方法在安全性和可靠性方面的优势。

（4）基于Hyperledger Fabric平台，设计并实现了相应的原型系统，通过与其他数据完整性审计方法在不同参数下的广泛对比实验，验证了所提方法的有效性和优越性。

2　相关工作

2.1　数据完整性审计

为了实现对云存储服务器上数据完整性的远程审计，学术界提出了多种解决方案。Ateniese等[9]在2007年提出了可证明数据拥有算法（Provable Data Possession，PDP），通过将原始数据分割成块并采用随机抽样策略来检查部分数据块的完整性，从而实现高概率的完整性审计。此后，一系列基于PDP的方法相继被提出，这些方法具备了支持数据更新[10-11]、无证书密码体制[12]、防密钥暴露[13]和数据共享[14]等特性。但是这些方法尚不支持多副本的完整性审计。

Curtmola等[15]在2008年首次提出了针对多个数据副本的完整性审计协议，但该方法需要对每个数据副本逐一验证，计算效率低下。为解决这一问题，Barsoum等[16]在2014年提出了一种动态多副本数据完整性审计方法，利用映射版本表同时验证多个副本的证明，并支持数据的动态更新和访问授权。Liu等[17]进一步提出了基于自顶向下多副本默克尔哈希树的动态云上大数据完整性审计协议，通过在默克尔树的子树中存储每个数据分块的所有副本哈希值，有效降低了动态更新的开销。Zhou等[18]针对云上电子病历数据，提出了一种改进的动态多副本审计方法，进一步降低了审计的计算和通信开销。

在这些工作的基础上，很多适用于多副本审计的方法被提出[19-21]，但这些方法并不适用于多服务器存储场景。为了应对这一挑战，Wang等[22]在2015年提出了一种基于身份ID的多服务器数据完整性审计方法，有效实现了多服务器场景下的数据审计，同时避免了复杂的证书管理开销。Zhou等[23]提出了一种无证书多云多副本数据完整性审计方法，通过改进标签生成方式，为同一数据块的所有副本生成标签，减少了审计过程中的计算开销。与Zhou等人的方法相比，Gou等[24]在2023年提出的方法不仅支持多服务器审计，还支持数据共享和用户撤销功能。

尽管这些方法在实现多云多副本数据完整性审计方面取得了一定进展，但它们在计算效率方面仍存在明显不足，尤其是在大规模数据完整性审计场景下的应用。因此，本研究旨在提出一种新的审计方案，以解决现有方法在计算开销方面的局限性。

如图1所示，传统的多云多副本数据完整性审计算法模型涉及四个主要角色：用户、可信第三方审计员、多个云存储服务器以及云存储服务器管理者。当用户需要进行数据完整性审计时，委托可信第三方审计员产生随机挑战，并将这些挑战发送到云存储服务器管理者，管理者随后将挑战分发给相应的云服务器。各个云存储服务器在收到挑战后，将基于其存储的数据生成相应的证明并回传给管理者。为了降低计算和通信成本，管理者会将所有收到的证明进行聚合，形成一个综合证明，再将其发送给审计员。审计员接着运用审计算法对聚合后的证明进行验证。在该框架中，审计员的角色可以由云服务器的管理者兼任，这样的设计旨在优化审计流程并提高效率。然而，这种模式高度依赖于对第三方审计员的信任，极可能引入潜在的安全风险和性能瓶颈。

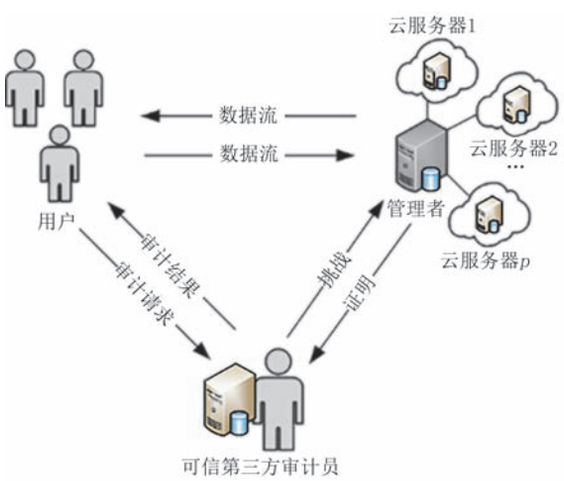

图1　传统的多云多副本数据完整性审计模型

2.2　基于区块链的数据完整性审计

中本聪在2008年提出比特币的概念[25]，标志着区块链的诞生。区块链作为一种分布式账本，记录了网络中所有实体的状态[26]。随着信息技术的快速发展，区块链技术以其去中心化的特性已广泛应用于保险[27]、医疗保健[28]、电子投票[29]和物联网[30]

等行业。区块链技术在数据完整性审计中的应用，有效地解决了第三方审计的信任问题。

众多学者提出了基于区块链的数据完整性审计方法。周家顺等[31]提出了一种支持多副本的区块链审计方法，适用于大数据环境下的非结构化数据审计。Miao等[32]提出了一种区块链辅助的多云存储中支持故障定位的多副本可证明数据拥有方法，通过在区块链上部署智能合约，并结合二分搜索技术来实现故障数据块的定位。Yang等[33]提出了一种基于区块链的多云多副本数据公开审计方法，利用区块链的不可预测性构建公平的挑战信息，以防止恶意审计员和云存储服务器之间的串通欺诈用户。不同于上述方法，Zhang等[34]提出的方法不仅实现了故障定位，还将用户、审计员以及云存储服务器之间的交互记录存储在区块链上，便于事后查证。Tian等[35]提出了一种具有去中心化信任管理能力的多副本数据共享审计方法，该方法为每个用户绑定一个存储在区块链上的信任值来表示用户的可信度，任何非法行为都将影响信任值的计算，并记录在区块链上。区块链的可追溯性、不可篡改性等优点可以代替用户执行审计。尽管已有的方法采用区块链技术，但区块链通常仅作为辅助验证的角色，并未完全取代可信第三方的角色，因此第三方审计的信任问题依然存在。

本研究采用区块链智能合约来执行数据完整性审计，智能合约被设计为一个可信的第三方执行器，能够自动执行审计任务，并为用户及云存储服务供应商提供可信的审计结果。此设计有效解决了传统第三方审计服务中的信任问题。本文还对审计算法进行了优化。通过改进数据标签的生成过程，显著减少了群上乘法和幂运算的计算复杂度，并将验证阶段的累乘运算简化为累加运算，从而大幅度降低了算法的计算成本。这些创新为数据完整性审计提供了一种计算效率高、安全性强的解决方案，为云计算环境中的数据安全提供了新的保障措施。

3 相关知识

3.1 双线性映射

假设$<G,*>$是一个代数系统，若运算$*$满足：

（1）对于集合G中任意的元素a和b，如果$a*b$也属于集合G，则称集合G对运算$*$是封闭的；

（2）对于集合G中任意的元素a，b和c，有$a*(b*c)=(a*b)*c$，则称$*$在集合G上满足结合律；

（3）如果在集合G中存在一个元素e，使得对于集合G中的任意元素a，都满足$a*e=e*a=a$，则称e为代数系统$<G,*>$的单位元；

（4）对于集合G中的任意元素a，如果存在一个元素a^{-1}，使得$a*a^{-1}=a^{-1}*a=e$，那么称a^{-1}为元素a在集合G中的逆元。

若满足以上所有条件，称$<G,*>$是一个群，也可记为G。若G是一个有限集合则称为有限群，元素个数称为群的阶数。在群G中若存在元素g，使得对于任意的整数n，都有$g^n\in G$，那么群G就被称为循环群，g称为生成元。

双线性映射是一种具有特殊数学性质的映射关系，能够将两个不同群上的元素映射到另一个群上。假设G_1和G_T是两个阶数为p的乘法循环群，双线性映射$e:G_1\times G_1\rightarrow G_T$满足以下条件：

（1）双线性：对任意的$P,Q,S,R\in G_1$和$a,b\in Z_p$，有$e(P^a,Q^b)=e(P,Q)^{ab}$，$e(P\cdot R,Q)=e(P,Q)\cdot e(R,Q)$，$e(P,Q\cdot S)=e(P,Q)\cdot e(P,S)$。

（2）非退化性：对任意的$P,Q\in G_1$满足$e(P,Q)=e$（e为G_T中的单位元）。

（3）可计算性：对任意的$P,Q\in G_1$，存在一个高效的算法能够计算出映射$e(P,Q)$。

3.2 动态哈希表

用户在云服务器上存储的数据常需动态更新，而传统的数据完整性审计方法在处理数据动态更新时存在局限性。Tian等提出一种利用动态哈希表存储数据块相关信息的方法[10]。如图2所示，动态哈希表是一种二维数据结构，由文件元素和块元素两部分组成。每个文件元素包含了文件序号、文件唯一标识符，以及指向块元素的指针，并存储在一个类似于数组的数据结构中。每个块元素则包含了数据块版本号和时间戳信息，且这些块元素通过链表的形式相互连接。在动态哈希表中，每个用户的文件都通过链表进行组织，每个数据块都对应一个块元素，并存储在链表的节点中，而相应的文件元素则作为链表的头节点。该结构更新过程简单，只需对链表或数组执行相应的修改、插入或删除操作，即可实现数据的动态更新。

然而，已有动态方法在更新阶段不仅需要为更新数据块及其副本生成数据标签，还需要同时验证各个云服务器提交的更新证明，这一过程中涉及到大量的群上累乘和幂运算操作。一旦用户频繁或大

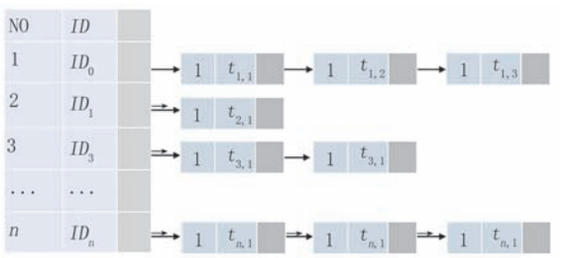

图2　动态哈希表

量更新数据时，将会带来显著的计算开销，影响数据更新速度和用户使用体验。为解决这一问题，本文提出了一种改进策略，通过优化标签的生成方式，减少了数据更新过程中的群上累乘和幂运算操作。这一改进有效降低了更新阶段的计算开销，提高了数据更新的速度。

3.3　区块链

区块链作为一种去中心化的分布式账本技术[25]，其核心特征在于网络中的每个节点都存储着链上的所有交易信息。这些信息被打包进区块中，并以链式结构相连，形成了所谓的区块链。该技术融合了数学、密码学和网络传输等学科的技术成果，构建了一个安全性、透明性、可追溯性和不可篡改性兼具的系统。

区块链的去中心化特性消除了对中央权威机构的依赖，每个节点都参与到数据的验证和存储中，确保了整个系统的透明性和公正性。密码学的应用保证了交易信息的安全性和用户身份的匿名性。通过复杂的加密算法，区块链确保了一旦交易信息被记录，就无法被篡改或撤销，从而提供了高度的数据完整性和可靠性。此外，区块链的链式结构设计使得每个交易记录都可以追溯到其历史源头，增强了系统的可审计性。

智能合约作为区块链核心技术之一，其概念最早由密码学家Nick Szabo在1994年提出。随着以太坊这一开源公共区块链平台的出现，智能合约的应用日益普及。以太坊平台具备图灵完备的以太坊虚拟机，支持开发者使用Solidity编程语言来编写智能合约。智能合约是一种嵌入特定条件、规则和操作的计算机程序代码，其核心特性包括：

（1）自动执行：当合约中的条件满足时就会自动触发执行相关操作，无需人工干预。这种自动执行机制赋予了智能合约独立性和可信任性。

（2）去中心化：智能合约的整个执行过程由区块链上的所有节点共同协作完成，不依赖于任何中心化的实体。

（3）不可篡改：一旦智能合约的执行数据被记录在区块链上，由于区块链的分布式账本特性，篡改这些数据需要控制超过网络上51%的算力，这在实际操作中极难实现，确保了智能合约的不可篡改性。

（4）透明性：区块链上的智能合约对所有网络参与者公开，任何人都可以查看合约的源代码、规则和执行结果，增强了透明度和信任度。

（5）可编程：智能合约允许用户根据特定需求定制功能和操作逻辑。不同的区块链平台支持不同的编程语言，例如以太坊支持Solidity，而Hyperledger Fabric支持Go和Java等，为用户提供了广泛的选择。

4　设计目标和威胁模型

4.1　设计目标

本文提出的方法专为多云多副本数据完整性验证设计，需要满足以下关键目标：

（1）标签的不可伪造性：确保用户为每个数据块生成的唯一数据标签无法被任何云存储服务器仿造或篡改；

（2）证明的不可伪造性：确保任何云存储服务器都无法制作虚假的数据完整性证明，以规避审计验证；

（3）审计结果的正确性：确保只有云存储服务器在审计过程中提供正确无误的证明才能通过数据完整性审计；

（4）数据存储的动态性：允许用户随时对其存储在各云存储服务器上的数据进行更新操作，包括数据的修改、插入和删除，且不会影响数据的安全性和完整性；

（5）多云多副本审计：能够对多个云存储服务器提交的多个数据副本的完整性证明进行同时审计，确保跨云平台的数据一致性和可靠性。

4.2　威胁模型

在本研究提出的方法中，潜在的安全威胁可归纳为以下几个方面：

（1）伪造攻击：云存储服务器可能企图伪造数据标签，以构建一份数据完整性证明；

（2）替换攻击：云存储服务器可能企图使用非挑战数据块中的数据来生成证明信息，以欺骗数据完整性审计；

（3）共谋攻击：不同的云存储服务器之间可能存在串通行为，通过共享同一份数据的证明来通过

数据完整性审计。

为应对这些潜在威胁，本研究提出了以下三个安全性假设，作为后续的安全性分析的基础：

假设1：所有云存储服务器都是潜在的敌手，它们可能会尝试执行各种攻击手段；

假设2：用户是诚实的，并且生成的数据标签真实有效；

假设3：网络中的通信是安全的，即所有传输的数据块和证明信息都是加密的，无法被未授权的第三方截获或篡改。

假设存在A_1、A_2和A_3三种类型的攻击。A_1试图伪造一个数据块的标签，并且能够使用任意值替换用户的公钥，但是不能访问主密钥。A_2也试图伪造一个数据块的标签，它可以访问主密钥，但是不能替换用户的公钥。A_3试图伪造证明通过验证。接下来，分别通过挑战者C和A_1、A_2以及A_3之间的博弈来定义本文的安全模型。

博弈1：该博弈发生在挑战者C和攻击者A_1之间。具体博弈过程如下：

初始化阶段：在初始化阶段，C产生参数$params$和主密钥msk，然后C将$params$发送给攻击者A_1并秘密保存msk。

质询阶段：在质询阶段，攻击者A_1可以发起一系列的质询，挑战者C会根据质询生成并返回相应的响应，具体如下：

(1)哈希质询：A_1向C发起一系列的哈希质询，C产生相应的哈希值作为响应。

(2)部分私钥质询：A_1自适应地向C发送一系列选中的元组$(ID', \{ID'_p\})$，其中ID'是用户身份字符串，$\{ID'_p\}$为用户为各个云存储服务器生成的标识字符串，C产生对应的部分私钥作为响应并发送给A_1。

(3)秘密值质询：A_1自适应地向C发送一系列选中的$(ID', \{ID'_p\})$，C产生对应的秘密值作为响应并发送给A_1。

(4)公钥质询：A_1自适应地向C发送一系列选中的$(ID', \{ID'_p\})$，C产生对应的公钥作为响应并发送给A_1。

(5)公钥值替换：A_1可以将用户$(ID', \{ID'_p\})$的公钥替换为任何值。

(6)副本质询：A_1自适应地选择一些数据块发送给C，C生成对应副本作为响应并返回给A_1。

(7)标签质询：A_1自适应地选择元组$(b', ID', \{ID'_p\})$发送给C，其中b'为数据块，C生成对应的数据标签并返回给A_1。

伪造阶段：A_1生成一个用户$(ID', \{ID'_p\})$的公钥pk'_{DO}以及数据块b'的标签σ'，如果满足以下所有条件，那么A_1将赢得博弈。

(1)A_1伪造的数据标签σ'对用户$(ID', \{ID'_p\})$和公钥pk'_{DO}是合法有效的；

(2)A_1没有查询过用户$(ID', \{ID'_p\})$的整个密钥；

(3)A_1没有查询过用户$(ID', \{ID'_p\})$的部分密钥，而是替换用户$(ID', \{ID'_p\})$的公钥；

(4)A_1没有查询过用户$(b', ID', \{ID'_p\})$的数据标签。

在博弈1中，攻击者伪装成用户与挑战者(密钥生成中心)进行交互，能够替换用户的公钥，但不能访问主密钥。攻击者试图伪造用户生成的数据标签。

博弈2：该博弈发生在挑战者C和攻击者A_2之间。具体定义如下：

在初始化阶段，C产生参数$params$和主密钥msk，并将$params$和msk都发送给攻击者A_2。

质询阶段：在质询阶段，攻击者A_2可以发起一系列的质询，挑战者C会根据质询返回相应的响应，具体如下：

(1)哈希质询：A_2向C发起一系列的哈希质询，C会产生相应的哈希值作为响应。

(2)秘密值质询：A_2自适应地向C发送一系列选中的$(ID', \{ID'_p\})$，C产生对应的秘密值作为响应并发送给A_2。

(3)公钥质询：A_2自适应地向C发送一系列选中的$(ID', \{ID'_p\})$，C产生对应的公钥作为响应并发送给A_2。

(4)副本质询：A_2自适应地选择一些数据块并发送C，C生成对应的副本作为响应并返回给A_2。

(5)标签质询：A_2自适应地选择元组$(b', ID', \{ID'_p\})$发送给C，C生成对应的数据标签并返回给A_2。

伪造阶段：A_2生成一个用户$(ID', \{ID'_p\})$的公钥pk'_{DO}以及数据块b'的标签σ'，且如果满足以下条件，那么A_2将赢得博弈。

(1)A_2伪造的数据标签σ'对用户$(ID', \{ID'_p\})$和公钥pk'_{DO}是合法有效的；

(2)A_2没有查询过用户$(ID', \{ID'_p\})$的秘密值；

(3)A_2没有查询过用户$(b', ID', \{ID'_p\})$的数据标签。

在博弈2中，攻击者伪装成密钥生成中心与挑战者(用户)交互，能够访问主密钥，但是不能替换用户的公钥。攻击者试图伪造用户生成的数据标签。

博弈3：该博弈发生在挑战者C和攻击者A_3之间，假设A_3是一个不诚实的云存储服务器，它试图通过伪造证明通过数据完整性审计。具体过程如下：

初始化阶段：C获取主密钥msk、参数$params$以及用户的私钥，然后将参数$params$发送给A_3，并秘密保存主密钥msk和用户的私钥。

副本质询：A_3自适应地发送四元组$(b_i, ID', \{ID'_p\}, n_b)$到$C$获取副本$\{b_{ij}\}$，其中$b_i$是数据块，$n_b$是副本个数。挑战者$C$生成$n_b$个副本作为响应发送给$A_3$。

挑战质询：C向A_3发送随机挑战$chall$并要求A_3生成证明。

伪造阶段：A_3根据挑战$chall$产生数据完整性证明$proof$。假如$proof$通过验证，那么A_3就赢得了博弈。

在博弈3中，恶意的云存储服务器与挑战者(用户和密钥生成中心)进行交互，试图伪造数据完整性证明。

4.3　方法模型

图3展示了本文所提的方法模型，涵盖了4个主要实体：用户、密钥生成中心、区块链以及云存储服务器集合。

用户：拥有大量数据，需存储于云存储服务器上。为确保数据安全，用户将数据制作多个副本，并分散存储于不同的云服务器上。

密钥生成中心：该中心负责生成用户及各个云存储服务器所需的部分密钥及相关参数，以支持数据安全性和审计过程。

云存储服务器集合：指代一组独立且无关联的云存储服务器。这些服务器具备强大的数据处理、管理和存储功能，为用户提供数据存储服务，并为此收取费用。同时，它们能够响应数据完整性挑战，并生成相应的证明。

区块链：区块链上部署了4个关键智能合约，包括元数据信息存储智能合约、挑战生成和分发智能合约、证明聚合和验证智能合约以及更新验证智能合约。这些智能合约使得区块链能够充当可信的第三方，代表用户执行数据完整性审计。

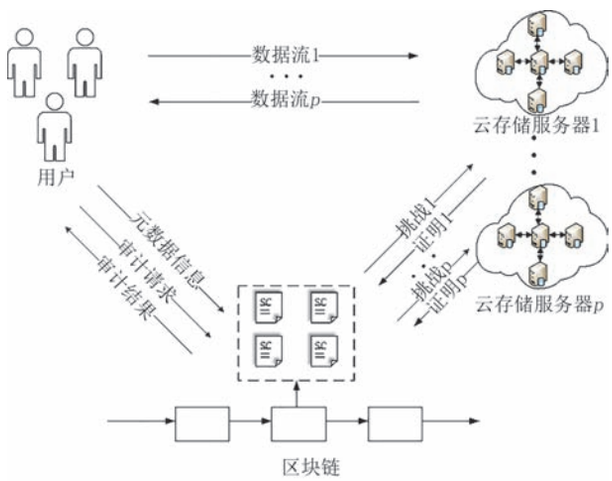

图3　方法模型

本文所提方法中的数据完整性审计包括三个阶段：初始化阶段、审计阶段和更新阶段。

初始化阶段

(1)由密钥生成中心生成审计所需要的参数；

(2)用户向密钥生成中心获取部分密钥并生成秘密参数以产生公私钥对，同时为各个云存储服务器获取独特的标识参数；

(3)用户将数据分块并为每个数据块生成独一无二的副本和标签；

(4)用户将公共参数以及文件相关的元数据信息上传到区块链，将文件和对应的数据标签集合存储到各个云存储服务器。

审计阶段

(1)当用户需要验证数据完整性时，会向区块链发送审计请求；

(2)区块链智能合约根据当前状态生成一系列挑战，并分发给各个云存储服务器；

(3)服务器在接收到挑战后，生成相应的证明并返回给区块链；

(4)区块链上的智能合约随后聚合这些证明并进行验证。

更新阶段

(1)用户产生更新请求并发送到区块链，同时生成更新信息发送到各个云存储服务器；

(2)每个服务器在接收到更新信息后，会生成更新证明并返回给区块链；

(3)更新验证智能合约聚合并验证这些更新证明，一旦验证通过，智能合约更新元数据信息并将结果返回给各个云存储服务器；

(4)收到验证通过消息的服务器会更新数据和相应的标签。

此方法模型通过明确的阶段划分和角色分配，确保了数据完整性审计的高效性和准确性，同时提供了数据动态更新的灵活性和安全性。

5 具体方案

5.1 初始化阶段

5.1.1 参数生成

假设G_1和G_T是两个阶数为素数p的乘法循环群，g是G_1的生成元，$e: G_1 \times G_1 \rightarrow G_T$是一个双线性映射。首先，密钥生成中心选取以下4个哈希函数：哈希函数$H(\cdot): \{0,1\}^* \rightarrow G_1$将字符串映射到群$G_1$上；哈希函数$h(\cdot): G_T \rightarrow Z_p$将群$G_T$上的元素映射到集合$Z_p$上；伪随机函数$\lambda_1: \{0,1\}^* \rightarrow \{1,2,\cdots,n\}$将字符串映射到$\{1,2,\cdots,n\}$中的一个整数；伪随机函数$\lambda_2: \{0,1\}^* \rightarrow Z_p$将任意字符串映射到集合$Z_p$上。然后，由密钥生成中心生成一个主密钥$msk \leftarrow Z_p$并计算出相应的主公钥$mpk = g^{msk}$，最终得到公开参数 $params = \{G_1, G_T, \mathrm{q}, \mathrm{e}, \mathrm{g}, \mathrm{H}(), \mathrm{h}() \lambda_1, \lambda_2, mpk\}$。在该阶段，用户会通过和密钥生成中心交互获取部分密钥和云存储服务器标识参数，这些部分密钥和标识参数将用于标签生成。标识参数只在用户端参与数据标签的生成，以此保证存储在每个服务器上的数据标签都不相同，从而防止不同云存储服务器之间相互串通。同时，用户通过一次通信从密钥生成中心获得所有标识参数，与逐个查询云服务器相比其效率更高，且简化了交互流程。标识参数的具体生成步骤如下：

(1)用户将自己的身份信息字符串$ID_{DO} \in \{0,1\}^*$和为各个云存储服务器生成的标识字符串$\{ID_p \in \{0,1\}^*\}_{1 \leqslant p \leqslant l}$发送到密钥生成中心，其中$l$表示服务器个数。密钥生成中心使用BLS签名算法生成部分密钥$ssk_{DO} = H(ID_{DO})^{msk}$和云存储服务器标识参数$\{ssk_{DO_p} = H(ID_p)^{msk}\}_{1 \leqslant p \leqslant l}$作为响应返回给用户。

(2)当用户收到密钥生成中心的响应之后，会使用主公钥mpk验证BLS签名$e(ssk_{DO}, g) = e(H(ID_{DO}), mpk)$，$\{e(ssk_{DO_p}, g) = e(H(ID_p), mpk)\}_{1 \leqslant p \leqslant l}$来验证部分私钥和云存储服务器标识参数。验证通过之后，用户选择一个随机元素$x \leftarrow Z_p$作为秘密参数，生成私钥$sk_{DO} = (x + ssk_{DO}) mod\ p$和公钥$pk_{DO} = g^{sk_{DO}}$。

(3)所有的云存储服务器也通过相同步骤将自己的身份信息字符串发送给密钥生成中心获取部分私钥并生成自己的公私钥对$\{sk_{CSS_p} = (x_p + ssk_{CSS_p}) mod\ p,\ pk_{CSS_p} = g^{sk_{CSS_p}}\}$。

5.1.2 副本生成

假设用户拥有的数据为F并想要将这些数据存储到l个云存储服务器上。首先，用户将自己的数据分割成n个数据块$F = \{b_i\}_{1 \leqslant i \leqslant n}$，数据分块大小$S$是一个可以调整的参数，如果最后一个数据块小于$S$则使用字符0进行填充。然后，用户为数据块生成$m$个副本$\{F_j = \{b_{ij}\}\}_{1 \leqslant j \leqslant m}$，每个副本都是在原始数据块的基础上加上额外的信息：$b_{ij} = b_i + h(name \| j \| ssk_{DO} \| ssk_{DO_p})$，其中$ssk_{DO_p}$是用户通过密钥生成中心生成的第$p$个云存储服务器的标识参数，表示这个数据分块将会存储到第p个云存储服务器上，$name$为该数据块的名称，j是副本块序号，ssk_{DO}是用户部分私钥。由于每个副本序号各不相同，所以用户为每个数据分块生成的副本也不相同，不仅能够有效防止云存储服务器欺瞒用户只保留一份数据副本而删除其他数据副本，也能防止不同云存储服务器之间串通共同使用一份数据副本生成完整性证明。同时，用户也可以轻松地从副本中还原原始数据：$b_i = b_{ij} - h(name \| j \| ssk_{DO} \| ssk_{DO_p})$。为了数据的安全，每一个云存储服务器至少存储一份数据副本。本文方法中数据副本的个数根据用户需求而定，用户为确保数据的安全会将数据生成多个副本并存储在云服务器上。为了保证数据的安全性和可用性，同时考虑到节省存储开销，可以参考Hadoop文件系统的设计，将数据副本设置为3。还可以使用纠删码技术，为数据分块生成校验单元，这样即使某些数据块丢失也能够恢复原始数据。最后，用户将数据分块进行分区$b_{ij} = \{m_{ijk}\}_{1 \leqslant k \leqslant s} \in Z_p$，每个分区的大小为160bits。

5.1.3 标签生成和数据存储

在生成副本之后，用户为每个副本块生成相应的标签。在审计阶段，通过验证云存储服务器基于标签和数据块内容生成的证明来验证数据的完整性。

首先用户随机选择集合Z_p上的S个元素$\{mk_k\}_{1 \leqslant k \leqslant s}$，并计算$\{MK_k = g^{mk_k}\}_{1 \leqslant k \leqslant s}$和$\{e(MK_k, pk_{DO})\}$。然

后，计算每个数据块的哈希值$h_{ij}=h(b_{ij}\|j\|sk_{DO}\|ssk_{DO_p})$，并通过公式(1)来生成副本块标签：

$$\sigma_{ij}=ssk_{DO_p}\cdot\left(BID_{ij}\cdot g^{\sum_{k=1}^{s}mk_k\cdot m_{ijk}}\right)^{sk_{DO}} \tag{1}$$

式中$BID_{ij}=g^{h_{ij}}$是数据块的唯一标识，mk_k是由用户生成的且秘密保存的随机数，ssk_{DO_p}是服务器标识参数，m_{ijk}是数据分块分区，sk_{DO}是用户私钥。

由于标签生成过程的计算主要由群G_1上的乘法运算和幂运算构成，导致了较高的计算开销。本文所提方案使用$BID_{ij}=g^{h_{ij}}$替代h_{ij}作为数据块的唯一标识参与标签的计算，能够有效减少标签生成过程中一次群G_1上的乘法操作和幂运算操作，从而减少计算开销。

用户使用h_{ij}生成一个副本索引表，在数据完整性审计阶段会通过该表快速找到抽样数据块的哈希值来完成数据完整性审计。副本索引表如图4所示，由动态哈希表改进而来。表中存放着四元组(h,FN,VN,SN)，其中h是哈希值，FN表示该数据块的副本号，VN是版本号，每次数据更新后版本号都会增加，SN为该数据块所在的服务器序号。副本索引表中有一个线性列表，每个列表节点存储一个数据块的首个副本生成的四元组。此外，每个节点还链接到一个链表，链表存储由该数据块的其他副本生成的四元组。

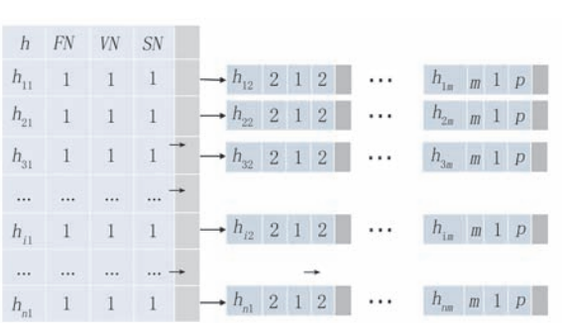

图4　副本索引表

最后，用户把数据相关的元数据信息$pp=\{params,\{MK_k\},\{e(MK_k,pk_{DO})\},\{H(ID_p)\},pk_{DO},\{Account_{CSSp}\},Account_{DO},\omega=\{\omega_p\},t=\{t_p\},n,m,\{(h,FN,VN,SN)\}\}$通过元数据信息存储智能合约公布到区块链，其中$Account_{CSSp}$表示第$p$个服务器的账户，$Account_{DO}$表示用户账户，$\omega=\{\omega_p\},t=\{t_p\}$代表每个服务器上存储的副本索引集合和个数，例如：假设第p个云存储服务器上存储数据的第1、3、5个数据副本，那么$\omega_p=\{1,3,5\}$，$t_p=3$。最后，用户把数据和对应的标签集合发送到相应的云存储服务器。各个云存储服务器在收到用户发送过来的信息之后会通过公式(2)进行一次审计以确保数据的完整性。

$$e(\prod_{j\in\omega_p}\prod_{i=1}^{n}\sigma_{ij},g)=e(H(ID_p)^{n\cdot t_p},pk)\cdot e(\prod_{j\in\omega_p}\prod_{i=1}^{n}BID_{ij}\cdot\prod_{k=1}^{s}MK_k^{\sum_{j\in\omega_p}\sum_{i=1}^{n}m_{ijk}},pk_{DO}) \tag{2}$$

式中σ_{ij}是数据标签，$BID_{ij}=g^{h_{ij}}$是数据块的唯一标识，m_{ijk}是数据分块分区，pk_{DO}是用户的公钥。

验证通过后云存储服务器存储数据并向用户发送消息。随后，用户删除数据及相应的标签。

5.2　审计阶段

5.2.1 挑战生成

当用户需要进行数据完整性审计时，将抽样数据块个数SN和签名sig_{DO}作为参数调用挑战生成和分发成智能合约。合约的伪代码如算法1所示，挑战生成和分发智能合约首先验证用户的数字签名，随后生成三个随机数集合$\{s_i=\lambda_1(\pi_1+i)\},\{v_i=\lambda_2(\pi_1+i)\},\{x_i=\lambda_2(\pi_2+i)\}$并存储在区块链上，其中$\pi_1$和$\pi_2$是区块链当前的状态信息，例如最后一个区块的哈希值、当前的时间等，本文采用当前时间作为状态信息，s_i是每个数据副本中的抽样数据块序号，v_i是为每个抽样数据块生成的随机数，x_i是为每个副本生成的随机数。最后，区块链将挑战$Chall=\{\pi_1,\pi_2,SN\}$分发到各个云存储服务器。

算法1. 挑战生成和分发智能合约

输入：抽样数据块个数SN和签名sig_{DO}

输出：挑战信息$chall$

```
1.  //智能合约首先验证用户的签名
2.  if (verify(sig_DO)==true) then
3.  //生成挑战信息
4.  S={s_i=λ1(π+i)}_{1≤i≤SN,1≤s_i≤n}
5.  V={v_i=λ2(π1+i)}_{1≤i≤SN}
6.  X={x_i=λ2(π2+i)}_{1≤i≤m}
7.      //在区块链上存储挑战信息
8.      store{S, V, X}
9.      return chall={S, V, X}
10. //签名验证失败则返回空的挑战信息
11. else
12.     return null
```

5.2.2 证明生成

当云存储服务器收到挑战之后，首先生成随机数集合$\{r_{pk}\leftarrow Z_p\}_{1\leqslant k\leqslant s}$并根据区块链上存储的元数

据信息计算$\{R_{pk}=e\left(MK_k,pk_{DO}\right)^{r_{pk}}\}_{1\leqslant k\leqslant s}$。紧接着，使用每个副本和相应的标签生成副本证明信息$\theta_{pj}=\prod_{i=1}^{SN}\sigma_{s_i j}^{v_i}$和$\{\delta'_{pjk}=\sum_{i=1}^{SN}v_i\cdot m_{ps_i jk}\}$，其中$m_{ps_i jk}$是第$p$个云服务器上的存储的第$j$个副本中的抽样数据块的第$k$个分区，$\sigma_{s_i j}$是抽样数据块对应的标签。最后，云存储服务器聚合所有副本的证明信息以得到最终证明$proof=\{\{R_{pk},\delta_{pk}\},\theta_p\}$，其中$\theta_p=\prod_{j\in\omega_p}\theta_{pj}^{x_j}$，$\delta_{pk}=r_{pk}+\sum_{j\in\omega_p}x_j\cdot\delta'_{pjk}$，并将最终证明提交到证明聚合和验证智能合约。

5.2.3 证明聚合和验证

如算法2所示，证明聚合和验证智能合约在收到证明之后，首先将所有证明信息进行聚合以生成证明聚合$proof\,Agg=\{\{\delta_k=\sum_{p\in l}\delta_{pk}\},\theta=\prod_{p\in l}\theta_p,\{R_k=\prod_{p\in l}R_{pk}\}\}$。然后在区块链上存储的副本索引表中取出挑战数据块的哈希值。最后使用公式(3)验证数据的完整性：

$$R_1\cdot R_2\cdots R_s\cdot\mathrm{e}(\theta,\mathrm{g})=e\left(\prod_{p=1}^{l}H(ID_{DO_p})^{\sum_{j\in\omega_p}x_j\cdot\sum_{i=1}^{SN}v_i},mpk\right)\cdot e\left(BID\cdot\prod_{k=1}^{s}MK_k{}^{\delta_k},PK_{DO}\right)\tag{3}$$

式中$BID=\prod_{i=1}^{SN}\prod_{j=1}^{m}BID_{s_i j}{}^{x_j\cdot v_i}=g^{\sum_{i=1}^{SN}\sum_{j=1}^{m}h_{s_i j}\cdot x_j\cdot v_i}$。

在公式(3)的计算过程中，将群G_1上的累乘运算转化为了累加运算，大大减少了计算开销。公式相等则说明验证通过。最后智能合约将验证结果发送给用户。若数据完整性证明未能通过聚合与验证智能合约的验证，说明存储在云存储服务器上的数据是不完整的，存在被篡改或者丢失的风险，智能合约会通知用户。用户会找到出现问题的数据块并尽力恢复数据，随后与云存储服务器协商赔偿。

算法2. 证明聚合和验证智能合约

输入：l个云存储服务器的数据完整性证明$\left\{\left\{R_{pk},\delta_{pk}\right\},\theta_p\right\}_{1\leqslant p\leqslant l}$

输出：证明验证结果true或者false

1. //智能合约验证用户和所有云存储服务器的签名
2. if ($verify(sig_{DO},\left\{sig_{CSP_p}\right\}_{1\leqslant p\leqslant l})==true$) then
3. //签名验证之后聚合所有证明
4. $\left\{\delta_k=\sum_{p\in l}\delta_{pk}\right\}$,
5. $\theta=\prod_{p\in l}\theta_p$,
6. $\left\{R_k=\prod_{p\in l}R_{pk},\right\}$
7. //使用公式(3)验证证明
8. if($equation(3)==ture$) then
9. //返回验证结果
10. return true
11. return false

5.3 更新阶段

在该阶段，用户可以更新存储在各个云存储服务器上的数据，包括修改、插入和删除三种操作。由于用户会将数据的多个副本存储在不同的云存储服务器上，为了保证数据的一致性，在更新阶段会同时更新存储在各个云存储服务器上的数据。

5.3.1 数据修改

当用户想要将数据块b_i修改为b'_i时，首先会生成数据副本$\{b'_{ij}\}_{1\leqslant j\leqslant m}$和相应的数据标签$\{\sigma'_{ij}=ssk_{DO_p}\cdot(BID'_{ij}\cdot g^{\sum_{k=1}^{s}mk_k\cdot m'_{ijk}})^{sk_{DO}}\}$，其中$BID'_{ij}=g^{h'_{ij}}$，$h'_{ij}=h(b'_{ij}\|j\|ssk_{DO}\|ssk_{DO_p})$。然后，生成修改请求$modRequest=\{SIG_{DO},mod,i\}$并发送到区块链，生成更新信息$\{modInfo_p=\{SIG_{DO},mod,i,\{b'_{ij}\}_{j\in\omega_p},\{\sigma'_{ij}\}_{j\in\omega_p}\}\}_{1\leqslant p\leqslant l}$并发送到各个云存储服务器，其中，$mod$表示该操作是修改操作。

当区块链收到修改请求之后，挑战生成和分发智能合约会验证用户的签名，验证通过后会生成挑战$modChall=\{v,i\}$发送到各个云存储服务器。当云存储服务器收到挑战和修改信息之后，首先生成随机数集合$\{r_{pk}\leftarrow Z_p\}_{1\leqslant k\leqslant s}$并计算$\{R_{pk}=e\left(MK_k,pk_{DO}\right)^{r_{pk}}\}_{1\leqslant k\leqslant s}$，然后计算$\theta_{pi}=\prod_{j\in\omega_p}\sigma_{ij}^{v}$和$\{\delta_{pik}=r_{pk}+\sum_{j\in\omega_p}v\cdot m_{ijk}\}$得到修改证明$modProof_p=\{\theta_{pi},\{\delta_{pik},R_{pk}\}\}$并发送到更新验证智能合约。更新验证智能合约如算法3所示。

算法3. 更新验证智能合约

输入：各个云存储服务器提供的数据完整性证明$modProof_p=\left\{\theta_{pi},\left\{\delta_{pik},R_{pk}\right\}\right\}_{1\leqslant p\leqslant l}$

输出：更新验证结果true或者false

1. //智能合约验证用户和云存储服务器的签名
2. if ($verify(sig_{DO},\left\{sig_{CSP_p}\right\}_{1\leqslant p\leqslant l})==true$) then
3. //聚合所有的证明信息
4. $proof=\left\{\left\{\delta_{ik},R_k\right\}_{1\leqslant k\leqslant s},\theta_i\right\}$
5. //利用公式(4)验证证明
6. if($equation(4)==ture$) then
7. //返回验证结果
8. return true

return false

随后智能合约聚合所有服务器的证明$proof=$

$\{\{\delta_{ik}=\sum_{p\in l}\delta_{pik}, R_k=\prod_{p\in l}R_{pk}\}, \theta_i=\prod_{p\in l}\theta_{pi}\}$并使用公式(4)进行验证：

$$R_1\cdot R_2\cdots R_s\cdot \mathrm{e}(\theta, \mathrm{g})=$$
$$e(\prod_{p=1}^{l}H(ID_{DO_p})^{t_p\cdot v}, mpk)\cdot$$
$$e((\prod_{j=1}^{m}\mathrm{BID}_{ij}^{\ v})\cdot\prod_{k=1}^{s}MK_k^{\delta_{ik}}, pk_{DO}) \quad (4)$$

式中θ_i和$\{\delta_{ik}, R_k\}_{1\leqslant k\leqslant s}$是更新证明，$BID_{ij}=g^{h_{ij}}$是数据块的唯一标识，$pk_{DO}$是用户公钥。

若验证通过，智能合约将会通知用户和各个云存储服务器。云存储服务器随后更新数据与标签并向区块链发送消息。最后，更新验证智能合约会更新数据分块个数和副本索引表。若验证失败，智能合约向用户和各个服务器发送验证失败的消息。各个云存服务器随后终止此次修改并删除修改信息。

5.3.2 数据插入

当用户想要插入数据块b_i，首先会生成数据副本$\{b_{ij}\}_{1\leqslant j\leqslant m}$和相应的数据标签$\{\sigma_{ij}=ssk_{DO_p}\cdot(BID_{ij}\cdot g^{\sum_{k=1}^{s}mk_k m_{ijk}})^{sk_{DO}}\}$，其中$BID_{ij}=g^{h_{ij}}$，$h_{ij}=h(b_{ij}\|j\|ssk_{DO}\|ssk_{DO_p})$。紧接着，用户生成插入请求$insRequest=\{SIG_{DO}, ins, i\}$并发送到区块链，生成插入信息$\{insInfo_p=\{SIG_{DO}, ins, i, \{b_{ij}\}_{j\in\omega_p}, \{\sigma_{ij}\}_{j\in\omega_p}\}\}_{1\leqslant p\leqslant l}$发送到各个云存储服务器，其中，$ins$表示该操作是插入操作。当区块链收到插入请求之后，挑战生成和分发智能合约会验证用户的签名，验证通过后生成挑战$insChall=\{v, i\}$并发送到各个云存储服务器。当云存储服务器收到挑战和插入信息之后，首先生成随机数集合$\{r_{pk}\leftarrow Z_p\}_{1\leqslant k\leqslant s}$并计算$\{R_{pk}=e(MK_k, pk_{DO})^{r_{pk}}\}_{1\leqslant k\leqslant s}$，然后计算$\theta_{pi}=\prod_{j\in\omega_p}\sigma_{ij}^{v}$和$\{\delta_{pik}=r_{pk}+\sum_{j\in\omega_p}v\cdot m_{ijk}\}$得到插入证明$insProof_p=\{\theta_{pi}, \{\delta_{pik}, R_{pk}\}\}$发送到更新验证智能合约。随后智能合约聚合各个云存储服务器的证明$proof=\{\{\delta_{ik}=\sum_{p\in l}\delta_{pik}, R_k=\prod_{p\in l}R_{pk}\}, \theta_i=\prod_{p\in l}\theta_{pi}\}$并使用公式(4)进行验证。验证通过，智能合约将会通知用户和各个云存储服务器。云存储服务器随后更新数据与标签并向区块链发送消息。最后，智能合约修改元数据信息和副本索引表。若验证失败，智能合约向用户和各个服务器发送验证失败的消息。各个云存服务器随后终止此次插入并删除插入信息。

5.3.3 数据删除

当用户想要删除数据时，向区块链发送的数据删除请求为$delRequest=\{SIG_{DO}, del, i\}$，向各个云存储服务器发送的数据删除信息为$\{delInfo_p=\{SIG_{DO}, del, i, \}_{1\leqslant p\leqslant l}$。当区块链收到删除请求后，挑战生成和分发智能合约会验证用户的签名，验证通过之后会生成挑战$modChall=\{i\}$发送到各个云存储服务器。各个云存储服务器收到挑战和删除信息后，删除相应的数据块和标签，并将删除成功的信息返回到区块链。区块链上的更新验证智能合约会修改元数据信息并更新副本索引表。

6　理论分析

6.1　设计目标分析

(1)标签的不可伪造性：如果攻击者A_1赢得博弈1和攻击者A_2赢得博弈2的概率可以忽略不计，那么所提出方法的标签具有不可伪造性。

证明：如果攻击者A_1可以赢得博弈，那么存在一个提取器B能够在给定的(g, G, g^a, g^b)的条件下计算出g^{ab}。因此，提取器B能够解决计算性Diffie-Hellman问题(Computational Diffie-Hellman Problem)，提取器B与攻击者A_1的交互过程如下：

初始化阶段：提取器B生成生参数$params$和主密钥msk。

H_1哈希质询：A_1使用选中的$(ID', \{ID'_p\})$执行H_1哈希质询。提取器B为H_1哈希质询保存着一个列表$L_{H_1}=\{(ID, \{ID_p\}, h_1, \{h_{1p}\}, D, \{D_p\}, T)\}$。如果$L_{H_1}$中保存着$(ID', \{ID'_p\})$的质询信息，那么就将$D', \{D'_p\}$作为响应返回给$A_1$。否则，提取器$B$选择随机元素$h'_1\in Z_p, \{h'_{1p}\in Z_p\}$，然后随机地掷出一个硬币$T'\in\{0, 1\}$。假设硬币为1的面朝上的概率为$\gamma$，那么0朝上的概率为$1-\gamma$。当$T'=0$时，$B$计算$D'=g^{h'_1}, \{D'_p=g^{h'_{1p}}\}$。当$T'=1$时，$B$计算$D'=(g^b)^{h'_1}, \{D'_p=(g^b)^{h'_{1p}}\}$。最后$B$将$D', \{D'_p\}$作为响应返回给$A_1$并将$(ID', \{ID'_p\}, h'_1, \{h'_{1p}\}, D', \{D'_p\}, T')$添加到$L_{H_1}$。

部分密钥质询：A_1使用选中的$(ID', \{ID'_p\})$执行部分密钥质询。B为部分密钥质询保存着$L_{ssk}=\{(ID, \{ID_p\}, ssk_{ID}, \{ssk_{ID_p}\}, x_{ID})\}$。首先，$B$检查$L_{H_1}$中是否存在$(ID', \{ID'_p\}, h'_1, \{h'_{1p}\}, D', \{D'_p\}, T')$，如果不存在，$B$则会使用$(ID', \{ID'_p\})$执行$H_1$哈希质询。

如果L_{ssk}中包含$(ID', \{ID'_p\}, *, *, *)$，$B$检查$ssk_{ID'}=\perp, \{ssk_{ID'_p}=\perp\}$。如果$ssk_{ID'}=\perp, \{ssk_{ID'_p}=\perp\}$，$B$从$L_{H_1}$中取出$(ID', \{ID'_p\}, h'_1, \{h'_{1p}\}, D',$

$\{D'_p\}, T')$。当$T'=1$,B中止质询。当$T'=0$,B计算$ssk_{ID'}=D'^{a}=g^{ah'_1}$,$\{ssk_{ID'_p}=D'^{a}_p=g^{ah'_{1p}}\}$并更新$ssk_{ID'}$,$\{ssk_{ID'_p}\}$。如果$ssk_{ID'}\neq\perp$,$\{ssk_{ID'_p}\neq\perp\}$,$B$取出$ssk_{ID'}$,$\{ssk_{ID'_p}\}$并返回给$A_1$。

如果L_{ssk}不包含$(ID', \{ID'_p\}, *, *, *)$,$B$从$L_{H_1}$中取出$(ID', \{ID'_p\}, h'_1, \{h'_{1p}\}, D', \{D'_p\}, T')$。当$T'=1$,$B$中止质询。当$T'=0$。$B$计算$ssk_{ID'}=D'^{a}=g^{ah'_1}$,$\{ssk_{ID'_p}=D'^{a}_p=g^{ah'_{1p}}\}$并返回给$A_1$,最后$B$在列表$L_{ssk}$中添加$(ID', \{ID'_p\}, ssk_{ID'}, \{ssk_{ID'_p}\}, \perp)$。

秘密值质询:A_1使用选中的$(ID', \{ID'_p\})$执行秘密值质询。首先,B检查L_{H_1}中是否存在$(ID', \{ID'_p\}, h'_1, \{h'_{1p}\}, D', \{D'_p\}, T')$,如果不存在,$B$使用$(ID', \{ID'_p\})$执行$H_1$哈希质询。然后,$B$继续检查$L_{ssk}$中是否包含$(ID', \{ID'_p\}, *, *, *)$。

如果L_{ssk}中包含$(ID', \{ID'_p\}, *, *, *)$,$B$检查$x'_{ID}=\perp$。如果$x'_{ID}=\perp$,$B$生成$x'_{ID}\leftarrow Z_p$并更新$x'_{ID}$。如果$T'=1$,$B$终止质询。最后,$B$取出$x'_{ID}$并返回给$A_1$。

如果L_{ssk}中不包含$(ID', \{ID'_p\}, *, *, *)$,$B$生成$x'_{ID}\leftarrow Z_p$并在$L_{ssk}$中添加$(ID', \{ID'_p\}, \perp, \perp, x'_{ID})$。

H_2哈希质询:A_1使用选中的$(ID', \{ID'_p\})$执行H_2哈希质询。B为H_2哈希质询保存着$L_{H_2}=\{(ID, \{ID_p\}, h_2, sk_{ID}, pk_{ID})\}$。首先,$B$检查$(ID', \{ID'_p\}, h'_1, \{h'_{1p}\}, D', \{D'_p\}, T')$是否存在于$L_{H_1}$中,如果不存在$B$使用$(ID', \{ID'_p\})$执行$H_1$哈希质询。然后,$B$继续检查$L_{ssk}$中是否包含$(ID', \{ID'_p\}, *, *, x'_{ID})$,若不包含,$B$使用$(ID', \{ID'_p\})$执行秘密值质询。最后$B$检查$L_{H_2}$是否包含$(ID', \{ID'_p\}, *, *, *)$。

如果包含,B计算$sk_{ID'}=x_{ID'}+h'_2 \bmod q$和$pk_{ID'}=g^{sk_{ID'}}$并更新$sk_{ID'}$和$pk_{ID'}$,然后将$sk_{ID'}$返回给$A_1$。

如果不包含,B选择$h'_2\leftarrow Z_p$,并计算$sk_{ID'}=x_{ID'}+h'_2 \bmod q$和$pk_{ID'}=g^{sk_{ID'}}$,然后$B$增加$(ID', \{ID'_p\}, h'_2, sk_{ID'}, pk_{ID'})$到在$L_{H_2}$并将$sk_{ID'}$返回给$A_1$。

公钥查询:A_1使用选中的$(ID', \{ID'_p\})$执行公钥查询。首先,B检查L_{ssk}中是否包含$(ID', \{ID'_p\}, *, *, x'_{ID})$,如果不包含,$B$使用$(ID', \{ID'_p\})$执行秘密值的质询。然后,$B$会检查$L_{H_2}$中是否包含$(ID', \{ID'_p\}, h'_2, sk_{ID'}, pk_{ID'})$。

如果存在,B取出$pk_{ID'}$并返回给A_1。

如果不存在,B随机选择$h'_2\in Z_p$,并计算$sk_{ID'}=x_{ID'}+h'_2 \bmod q$和$pk_{ID'}=g^{sk_{ID'}}$。然后$B$将$pk_{ID'}$返回给$A_1$并在$L_{H_2}$中增加$(ID', \{ID'_p\}, h'_2, sk_{ID'}, pk_{ID'})$。

公钥替换:A_1使用选中的$(ID', \{ID'_p\}, pk^*_{ID'})$执行公钥替换。首先检查$L_{H_2}$中是否存在$(ID', \{ID'_p\}, h'_2, sk_{ID'}, pk_{ID'})$。

如果L_{H_2}中存在,B使用$(ID', \{ID'_p\}, h'_2, \perp, pk^*_{ID'})$替换$(ID', \{ID'_p\}, h'_2, sk_{ID'}, pk_{ID'})$。

如果L_{H_2}中不存在,则向L_{H_2}中添加$(ID', \{ID'_p\}, \perp, \perp, pk^*_{ID'})$

H_3哈希质询:A_1使用三元组$(ID', \{ID'_p\}, b'_i)$执行H_3哈希值质询。提取器B保存着$L_{H_3}=\{(b_i, \{h_{3ij}\}_{1\leqslant j\leqslant m}, \{b_{ij}\}_{1\leqslant j\leqslant m})\}$。如果$L_{H_3}$中存在$(b'_i, *, *)$,$B$就会取出$\{h_{3ij}\}_{1\leqslant j\leqslant m}$并返回给$A_1$。否则,$B$随机选择$m$个元素$\{h'_{3ij}\leftarrow Z_p\}$并在$L_{H_3}$中增加$(b'_i, \{h'_{3ij}\}, \perp)$。最后将$\{h'_{3ij}\}_{1\leqslant j\leqslant m}$返回给$A_1$。

副本质询:A_1使用选中的原始数据块$(ID', \{ID'_p\}, b'_i)$执行副本质询。如果L_{H_1}中的$T'=0$,B结束质询。否则,B取出L_{H_3}中的$\{h'_{3ij}\}$并计算$\{b'_{ij}=b'_i+h'_{3ij}\}$。最后,$B$将$\{b'_{ij}\}$返回给$A_1$。

标签查询:A_1使用选中的原始数据块$(ID', \{ID'_p\}, h'_{ij}, b'_{ij})$执行标签质询。如果$L_{H_1}$中的$T'=1$,$B$结束质询。否则$B$从$L_{ssk}$取出$ssk_{ID}$,$\{ssk_{ID_p}\}$,从$L_{H_2}$中取出$sk_{ID'}$,从$L_{H_3}$中取出$\{b_{ij}\}_{1\leqslant i\leqslant n}$和$\{h'_{ij}\}$。最后生成标签并返回给$A_1$。

伪造阶段:A_1输出一个元组$(ID^*, \{ID^*_p\}, b^*_{ij}, h^*_{3ij}, \sigma^*_{ij}, PK_{ID^*})$,其中$b^*_{ij}$是数据块,$\sigma^*_{ij}$是相应的标签。

分析:当A_1赢得博弈,那么可以通过公式(5)的验证:

$$e(\sigma^*_{ij}, g)=e\left(H(ID^*_p), mpk\right)\cdot e(g^{h^*_{3ij}}, \prod_{k=1}^{s} MK_k{}^{b_{ijk}}, PK_{ID^*}) \tag{5}$$

根据博弈,B从L_{H_1}中取出元组$(ID^*, \{ID^*_p\}, h_1{}^*, D^*, T^*)$,如果$T^*=1$,$B$结束质询。否则,$B$从$L_{H1}$取出$h^*_1$并计算$H(ID^*_p)=g^{bh^*_1}$,同时从$L_{H_3}$中取出$h'_{3ij}$。所以$e(\sigma^*_{ij}, g)=e(g^{bh^*_{1p}}, g^a)\cdot e(g^{h'_{3ij}}, \prod_{k=1}^{s} MK_k{}^{b_{ijk}}, PK_{ID^*})$成立并有

$$g^{ab}=\left(\frac{\sigma^*_{ij}}{\prod_{k=1}^{s} MK_k{}^{b_{ijk}}\cdot PK_{ID^*}{}^{h'_{3ij}}}\right)^{\frac{1}{h^*_{1p}}} \tag{6}$$

假设A_1在时间t内以概率ϵ赢得博弈,并最多经过q_1次H_1哈希质询、q_{ssk}次部分密钥质询、q_{sv}次秘密值质询、q_2次H_2哈希质询、q_{pk}次公钥查询、q_{pkr}次

公钥替换、q_3次H_3哈希质询、q_c次副本质询、q_t次标签查询。那么B在博弈中不被打断的概率为$(1-\varsigma)^{q_{ssk}+q_c+q_t}$。因此$A_1$计算出$g^{ab}$的概率为$\epsilon' \geqslant \epsilon \cdot \varsigma \left(1-\varsigma\right)^{q_{ssk}+q_c+q_t} \geqslant \epsilon/((q_{ssk}+q_c+q_t)\cdot 2e)$，需要的时间为$t' \leqslant t+O(q_1+q_{ssk}+q_{sv}+q_2+q_{pk}+q_{pkr}+q_3+q_c+q_t)$。

类似的，可以得出A_2计算出g^{ab}的概率为$\epsilon' \geqslant \epsilon \cdot \varsigma \left(1-\varsigma\right)^{q_{sv}+q_c+q_t} \geqslant \epsilon/((q_{sv}+q_c+q_t)\cdot 2e)$，需要的时间为$t' \leqslant t+O(q_1+q_{sv}+q_2+q_{pk}+q_3+q_c+q_t)$。

综上所述，A_1和A_2能解决CDH问题，这与事实相悖。因此，所提出方法的标签具有不可伪造性。

(2)证明的不可伪造性：如果A_3获胜的概率不计，那么本文提出的多云多副本方法对不诚实的云存储服务器是安全的。即证明信息不可伪造。

当各个云存储服务器生成的证明$\{\{\delta_k, R_k\}, \theta\}$正确时，公式(3)一定成立。当云存储服务器伪造的证明$\{\{\delta'_k, R_k\}, \theta'\}$试图通过验证，那么验证公式为

$$R_1 \cdot R_2 \cdots R_k \cdot \mathrm{e}\left(\theta', \mathrm{g}\right) = e\left(\prod_{p=1}^{l} H(ID_{DO_p})^{\sum_{j\in\omega_p} x_j \cdot \sum_{i=1}^{SN} v_i}, mpk\right) \cdot e\left(BID \cdot \prod_{k=1}^{s} MK_k^{\delta'_k}, PK_{DO}\right) \tag{7}$$

由于标签具有不可篡改性，所以$\theta' = \theta$。将公式(3)和公式(7)相除得到公式(8)：

$$\prod_{k=1}^{s} MK_k^{\delta'_k} = g^{\sum_{k=1}^{s} mk_k \cdot \delta'_k} = \prod_{k=1}^{s} MK_k^{\delta_k} = g^{\sum_{k=1}^{s} mk_k \cdot \delta_k} \tag{8}$$

即$\sum_{k=1}^{s} mk_k(\delta'_k - \delta_k) = 0$。因为证明是伪造的，所以假设有$\Delta$个$\delta'_k \neq \delta_k$。在初始化阶段，用户会生成随机数集合$\{mk_k\}_{1\leqslant k\leqslant s}$并将其保持私密，因此$\sum_{k=1}^{s} mk_k(\delta'_k - \delta_k) = 0$的概率小于$\frac{q^{\Delta-1}}{q^s} \leqslant \frac{q^{\Delta-1}}{q^{\Delta}} = \frac{1}{q}$，这可以忽略不计。

(3)审计结果的正确性：云存储服务器提供的正确的数据完整性证明能够通过数据完整性审计。根据公式(3)，有

$$\begin{aligned}
&R_1 \cdot R_2 \cdots R_s \cdot \mathrm{e}\left(\theta, \mathrm{g}\right) = \\
&\prod_{p=1}^{l}\prod_{j=1}^{s} e\left(MK_j, pk_{DO}\right)^{r_{pj}} \cdot e\left(\prod_{i=1}^{SN}\prod_{j=1}^{m} \theta_{s_i j}^{x_j \cdot v_i}, g\right) = \\
&\prod_{p=1}^{l}\prod_{j=1}^{s} e\left(MK_j, pk_{DO}\right)^{r_{pj}} \cdot \\
&e\left(\prod_{p=1}^{l}\prod_{j\in\omega_p}\prod_{i=1}^{SN} (ssk_{DO_p}^{x_j \cdot v_i} \cdot (BID_{s_i j} \cdot g^{\sum_{k=1}^{s} mk_k \cdot m_{bjs_i k}})^{sk_{DO} \cdot x_j \cdot v_i}), g\right) = \\
&\prod_{p=1}^{l}\prod_{j=1}^{s} e\left(MK_j, pk_{DO}\right)^{r_{pj}} \cdot e\left(\prod_{p=1}^{l} H(ID_{DO_p})^{\sum_{j\in\omega_p} x_j \cdot \sum_{i=1}^{SN} v_i}, mpk\right) \cdot \\
&e\left(\prod_{i=1}^{SN}\prod_{j=1}^{m} BID_{s_i j}^{x_j \cdot v_i} \cdot \prod_{k=1}^{s} MK_k^{\sum_{j=1}^{m}\sum_{i=1}^{SN} x_j \cdot v_i \cdot m_{s_i jk}}, PK_{DO}\right) = \\
&e\left(\prod_{p=1}^{l} H(ID_{DO_p})^{\sum_{j\in\omega_p} x_j \cdot \sum_{i=1}^{SN} v_i}, mpk\right) \cdot \\
&e\left(\prod_{i=1}^{SN}\prod_{j=1}^{m} BID_{s_i j}^{x_j \cdot v_i} \cdot \prod_{k=1}^{s} MK_k^{\sum_{p=1}^{l} r_{pk} + \sum_{j=1}^{m}\sum_{i=1}^{SN} x_j \cdot v_i \cdot m_{s_i jk}}, PK_{DO}\right) = \\
&e\left(\prod_{p=1}^{l} H\left(ID_{DO_p}\right)^{\sum_{j\in\omega_p} x_j \cdot \sum_{i=1}^{SN} v_i}, mpk\right) \cdot e\left(BID \cdot \prod_{k=1}^{s} MK_k^{\delta_k}, PK_{DO}\right)。
\end{aligned}$$

(4)数据存储的动态性：本文所提出数据完整性审计方案允许数据的动态更新，通过在标签中嵌入数据块的相关属性而非位置索引，并在区块链上维护一个副本索引表来存储这些属性信息，实现了数据的动态管理。同时，还通过优化算法减少了更新阶段的计算复杂度，不仅提高了更新速度还优化了用户的使用体验。

(5)多云多副本审计：本文所提方案实现了多云多副本数据完整性审计的功能。能够对多个云存储服务器生成的各个数据副本的完整性证明进行同时验证，确保了分散在不同云环境中的数据副本的完整性和一致性。强化了数据的可靠性，并为用户在多云环境中的数据存储提供了强有力的安全保障。

6.2　安全性分析

(1)抗伪造攻击：由6.1节可知，本文方法的数据完整性证明具有抗伪造性，所以本文的方法能够防止不诚实的云存储服务器使用伪造的证明通过验证。

(2)抗替换攻击：假设不诚信的第p'个云存储服务器使用其存储的第j'个数据副本的第$s_{i'}$个数据块代替挑战中的第s_i个数据块生成数据完整性证明，则有

$$left = \prod_{p=1}^{l}\prod_{j=1}^{s} e\left(MK_j, pk_{DO}\right)^{r_{pj}} \cdot e(\theta, g) = \prod_{p=1}^{l}\prod_{j=1}^{s} e\left(MK_j, pk_{DO}\right)^{r_{pj}} \cdot$$

$$e(\prod_{p=p'+1}^{l}\prod_{p=1}^{p=p'-1}\prod_{j\in\omega_p}\prod_{i=1}^{SN}\theta_{pjs_i}^{x_j\cdot v_i}\cdot\prod_{j\in\omega_{p'}\setminus j'}\prod_{i=1}^{SN}\theta_{p'js_i}^{x_j\cdot v_i}\cdot$$
$$\prod_{i=i'+1}^{SN}\prod_{i=1}^{i=i'-1}\theta_{p'j's_i}^{x_{j'}\cdot v_i}\cdot\theta_{p'j's_{i'}}^{x_{j'}\cdot v_{i'}},g)=\prod_{p=1}^{l}\prod_{j=1}^{s}e\left(MK_j,pk_{DO}\right)^{r_{pj}}\cdot$$
$$e(\prod_{p=p'+1}^{l}\prod_{p=1}^{p=p'-1}\prod_{j\in\omega_p}\prod_{i=1}^{SN}ssk_{DO_p}\cdot(BID_{pjs_i}\cdot g^{\sum_{k=1}^{s}mk_k\cdot m_{pjs_ik}})^{sk_{DO}\cdot x_j\cdot v_i}\cdot$$
$$\prod_{j\in\omega_{p'}\setminus j'}\prod_{i=1}^{SN}ssk_{DO_{p'}}\cdot\left(BID_{p'js_i}\cdot g^{\sum_{k=1}^{s}mk_k\cdot m_{p'js_ik}}\right)^{sk_{DO}\cdot x_j\cdot v_i})\cdot$$
$$\prod_{i=i'+1}^{SN}\prod_{i=1}^{i=i'-1}ssk_{DO_p}\cdot\left(BID_{p'j's_i}\cdot g^{\sum_{k=1}^{s}mk_k\cdot m_{p'j's_ik}}\right)^{sk_{DO}\cdot x_{j'}\cdot v_i}\cdot ssk_{DO_p}\cdot(BID_{p'j's_{i'}}\cdot g^{\sum_{k=1}^{s}mk_k\cdot m_{p'j's_{i'}k}})^{sk_{DO}\cdot x_{j'}\cdot v_{i'}},g)=$$
$$\prod_{p=1}^{l}\prod_{j=1}^{s}e\left(MK_j,pk_{DO}\right)^{r_{pj}}\cdot e(\prod_{p=1}^{l}H(DO_p)^{\sum_{j\in\omega_p}x_j\cdot\sum_{i=1}^{SN}v_i},g^{sk_{DO}})$$
$$e(\prod_{p=p'+1}^{l}\prod_{p=1}^{p=p'-1}\prod_{j\in\omega_p}\prod_{i=1}^{SN}BID_{pjs_i}\cdot\prod_{k=1}^{s}MK_k^{\sum_{p=p'+1}^{l}\sum_{p=1}^{p=p'-1}\sum_{j\in\omega_p}\sum_{i=1}^{SN}x_j\cdot v_i\cdot m_{pjs_ik}}\cdot$$
$$\prod_{j\in\omega_{p'}\setminus j'}\prod_{i=1}^{SN}BID_{p'js_i}\cdot\prod_{k=1}^{s}MK_k^{\sum_{j\in\omega_{p'}\setminus j'}\sum_{i=1}^{SN}x_j\cdot v_i\cdot m_{p'js_ik}}\cdot$$
$$\prod_{i=i'+1}^{SN}\prod_{i=1}^{i=i'-1}BID_{p'j's_i}\cdot\prod_{k=1}^{s}MK_k^{\sum_{i=i'+1}^{SN}\sum_{i=1}^{i=i'-1}x_{j'}\cdot v_i\cdot m_{p'j's_ik}}\cdot$$
$$BID_{p'j's_{i'}}\cdot\prod_{k=1}^{s}MK_k^{x_{j'}\cdot v_{i'}\cdot m_{p'j's_{i'}k}},PK_{DO})=e(\prod_{p=1}^{l}H(DO_p)^{\sum_{j\in\omega_p}x_j\cdot\sum_{i=1}^{SN}v_i},mpk)\cdot$$
$$e(\prod_{p=p'+1}^{l}\prod_{p=1}^{p=p'-1}\prod_{j\in\omega_p}\prod_{i=1}^{SN}BID_{pjs_i}\cdot\prod_{j\in\omega_{p'}\setminus j'}\prod_{i=1}^{SN}BID_{p'js_i}\cdot$$
$$\prod_{i=i'+1}^{SN}\prod_{i=1}^{i=i'-1}BID_{p'j's_i}\cdot BID_{p'j's_{i'}},PK_{DO})\cdot$$
$$e(\prod_{k=1}^{s}MK_k^{\sum_{p=1}^{l}r_{pk}+\sum_{p=p'+1}^{l}\sum_{p=1}^{p=p'-1}\sum_{j\in\omega_p}\sum_{i=1}^{SN}x_j\cdot v_i\cdot m_{pjs_ik}}\cdot\prod_{k=1}^{s}MK_k^{\sum_{j\in\omega_{p'}\setminus j'}\sum_{i=1}^{SN}x_j\cdot v_i\cdot m_{p'js_ik}}\cdot$$
$$\prod_{k=1}^{s}MK_k^{\sum_{i=i'+1}^{SN}\sum_{i=1}^{i=i'-1}x_{j'}\cdot v_i\cdot m_{p'j's_ik}}\cdot\prod_{k=1}^{s}MK_k^{x_{j'}\cdot v_{i'}\cdot m_{p'j's_{i'}k}},PK_{DO})$$
$$right=e(\prod_{p=1}^{l}H\left(DO_p\right)^{\sum_{j\in\omega_p}x_j\cdot\sum_{i=1}^{SN}v_i},mpk)\cdot e(\prod_{p=1}^{l}\prod_{j\in\omega_p}\prod_{i=1}^{SN}BID_{pjs_i}\cdot$$
$$\prod_{k=1}^{s}MK_k^{r_{pk}+\sum_{j\in\omega_p}\sum_{i=1}^{SN}x_j\cdot v_i\cdot m_{pjs_ik}},PK_{DO})=$$
$$e(\prod_{p=1}^{l}H\left(DO_p\right)^{\sum_{j\in\omega_p}x_j\cdot\sum_{i=1}^{SN}v_i},mpk)\cdot e(\prod_{p=1}^{l}\prod_{j\in\omega_p}\prod_{i=1}^{SN}BID_{pjs_i}\cdot$$
$$\prod_{k=1}^{s}MK_k^{\sum_{p=1}^{l}r_{pk}+\sum_{p=p'+1}^{l}\sum_{p=1}^{p=p'-1}\sum_{j\in\omega_p}\sum_{i=1}^{SN}x_j\cdot v_i\cdot m_{pjs_ik}}\cdot\prod_{k=1}^{s}MK_k^{\sum_{j\in\omega_{p'}\setminus j'}\sum_{i=1}^{SN}x_j\cdot v_i\cdot m_{p'js_ik}}\cdot$$
$$\prod_{k=1}^{s}MK_k^{\sum_{i=i'+1}^{SN}\sum_{i=1}^{i=i'-1}x_{j'}\cdot v_i\cdot m_{p'j's_ik}}\cdot\prod_{k=1}^{s}MK_k^{x_{j'}\cdot v_{i'}\cdot m_{p'j's_{i'}k}},PK_{DO})。$$

由上可知，当且仅当$BID_{pjs_i}=g^{h_{pjs_i}}=BID_{p'js_{i'}}=g^{h_{p'js_{i'}}}$，即$h_{pjs_i}=h_{p'js_{i'}}$时，$left=right$。因为哈希函数具有抗碰撞性，所以$h_{pjs_i}=h_{p'js_{i'}}$的概率可忽略不计。因此任何不诚信的云存储服务器使用其他数据块生成的数据完整性证明都无法通过验证。

(3)抗共谋攻击：在本文方法中，初始化阶段生成的每个数据副本$\{F_j=\{b_{ij}\}\}_{1\leqslant j\leqslant m}$，$b_{ij}=b_i+h(name\|j\|ssk_{DO}\|ssk_{DO_p})$都是不同的，所以每份数据的完整性证明信息都不相同。由6.1所知，只有正确的数据完整性证明才能通过验证，因此使用其他数据生成的证明也将无法通过验证。

6.3 性能分析

本小节分析了方法的计算和通信性能，并对比了Li等[20]和Miao等[32]的方法。Li等的方法实现了多副本审计，能够同时验证多个副本的证明，Miao等的方法不仅实现了多云多副本审计还结合了区块链技术实现了错误定位。本文使用T_{pair}、T_{exp}、T_{mul}、$T_{mul_{Z_q}}$分别表示双线性映射、群G_1上的幂运算操作、群G_1上的乘法操作以及集合Z_p上的乘法操作的时间复杂度。设置数据分块个数为n，数据块分区个数为s，数据副本个数为m，抽样数据块个数为c，以及云存储服务器个数为l。

6.3.1 计算开销

在初始化阶段，根据本文方法的标签生成方式公式(1)可以得到标签生成的计算复杂度为：$O(mn(T_{exp}+T_{mul}))$。而在Li等和Miao等的方法中，标签生成方式为公式(9)：

$$\sigma_{ij}=ssk_{DO}\cdot(H\left(m_{ij}\right)\cdot g^{\sum_{k=1}^{s}mk_k\cdot m_{ijk}})^{sk_{DO}} \tag{9}$$

其时间复杂度为$O(2mn(T_{exp}+T_{mul}))$，明显地高于本文的方法。在审计阶段的证明生成过程中，

本文方法的时间复杂度为$O((c+t_p)(T_{exp}+T_{mul_{Z_q}}+T_{mul})+sT_{pair})$，Li等人的方法时间复杂度为$O(cT_{exp}+(c+t_p)T_{mul}+cT_{mul_{Z_q}})$，而Miao等的方法实际复杂度为$O((c+t_p)(T_{exp}+T_{mul_{Z_q}}+T_{mul}))$。可以看出，本文方法在证明生成的计算复杂度略高于另外两种方法，这是因为本文方法实现了用户数据的隐私保护，增加了一定的计算开销，但该特性在区块链环境中尤为重要。在审计阶段中的证明验证过程中，本文方法使用公式(4)来验证数据的完整性，由于公式(4)中$BID=\prod_{p=1}^{l}\prod_{j\in\omega_p}\prod_{i=1}^{SN}BID_{s_i,j}^{x_j v_i}=g^{\sum_p^l\sum_{j\in\omega_p}\sum_{i=1}^{Sn}h_{s_i,j}}$，可以减少群上乘法操作的次数，所以证明验证的时间复杂度为$O(3T_{pair}+(l+\mathrm{s})T_{exp}+(2\mathrm{s}+l)T_{mul}+2cmT_{mul_{Z_q}})$，而Li等和Miao等的方法分别为$O(3T_{pair}+(\mathrm{c}+\mathrm{s}+2)T_{exp}+(\mathrm{s}+\mathrm{c})T_{mul})$和$O(3T_{pair}+(\mathrm{cm}+\mathrm{s}+1)(T_{exp}+T_{mul})+2\mathrm{cm}T_{mul_{Z_q}})$。

6.3.2 通信开销

通讯开销主要来自数据完整性审计过程中区块链和云平台之间传输的挑战和证明信息。在挑战生成过程中，Li等[20]的方法会产生两个Z_p上的随机数作为伪随机函数的种子生成数据完整性挑战，并将这两个随机数和挑战数据块个数一起发送给云存储服务器，所以通信成本为$2|Z_p|+\log_2 n$，而Miao等[32]的方法则会使用三个伪随机函数的种子，因此通信开销为$3|Z_p|+\log_2 n$，其中n为挑战数据块个数。本文提出的方法会将区块链当前状态信息作为伪随机函数的种子，因此通信成本为$3\log_2 n$。在证明生成的过程中，Li等[20]和Miao等[32]的方法中的证明是一样的，通信成本均为$|G_1|+s|Z_p|$，但是本文方法使用了额外参数实现了隐私保护，所以通信成本为$|G_1|+2s|Z_p|$，略高于Li等和Miao等的方法。

7　实验评估

本节通过实验验证了所提方法的有效性，实验中将本文方法与Li等[20]和Miao等[32]的方法进行了比较，详细分析了这三种方法在不同阶段的计算成本。此外，为了进一步证明本方法的实用性，本文利用Hyperledger Fabric联盟链测试网络搭建了一个原型系统。该测试网络依托Docker技术构建，包括一个背书节点和一条通道，通道内涵盖两个组织，每个组织各有一个对等节点。通过在该测试网络上部署审计算法的智能合约，实现了代替用户自动执行数据审计流程的功能。

7.1　实验设置

本文使用Java PairingBased Cryptography(JPBC)库来模拟三种方法的所有过程。三种方法都是基于BLS短签名并使用A型曲线进行实验。Z_p和G_1的大小设置为$|Z_p|=|G_1|=160$ bits。所有的实验都是在一台拥有16 GB内存和Intel(R) Core(TM) i5-9500处理器@ 3.00 GHz的Windows计算机上进行的，每个实验结果取10次的平均值。

在初始化阶段主要进行了四个实验。第一个实验是关于计算开销随数据分块大小的变化情况(参数设置为：$n=3000, m=3, s=$ 10 KB, 30 KB, 50 KB, 70 KB, 90 KB, 110 KB, $p=3$)。第二个实验是计算开销随副本个数的变化情况(参数设置为：$n=3000, m=1,2,3,4,5,6, s=$ 40 KB, $p=3$)。第三个实验是关于数据块个数的实验(参数设置为：$n=1000, 2000, 3000, 4000, 5000,$ 6000, $m=3, s=40$ KB, $p=3$)。最后一个实验是计算开销随云存储服务器个数的变化情况(参数设置为：$n=3000, m=3, s=40$ KB, $p=1,2,3,4,5,6$)。审计阶段分为证明产生和证明验证，在证明产生和验证过程中本文也进行了四个实验。第一个实验评估计算开销与数据块大小的关系(参数设置为：$m=3, s=$ 10 KB, 30 KB, 50 KB, 70 KB, 90 KB, 110 KB, $p=3$, $c=460$)。第二个实验是计算开销随副本个数的变化情况(参数设置为：$m=1,2,3,4,5,6, s=40$ KB, $p=3, c=460$)。第三个评估计算开销与抽样数据块个数的关系(参数设置为：$m=3, s=40$ KB, $p=3, c=100, 200, 300,$ 400, 500, 600)。最后一个实验关于云存储服务器个数的实验(参数设置为：$m=3, s=40$ KB, $p=1,2,3,4,5,6, c=460$)。最后，本文在更新阶段进行了更新证明产生和验证的四个实验，分别评估计算开销与数据块大小(参数设置为：$m=3, s=$ 10 KB, 30 KB, 50 KB, 70 KB, 90 KB, 110 KB, $p=3, u=30$)、副本个数(参数设置为：$m=1,2,3,4,5,6, s=40$ KB, $p=3, u=30$)、更新数据块个数(参数设置为：$m=3, s=40$ KB, $p=3, u=10, 20, 30, 40, 50, 60$)和云存储服务器个数(参数设置为：$m=3, s=40$ KB, $p=1,2,3,4,5,6, u=30$)

的关系。

7.2 初始化阶段的开销

初始化阶段的计算开销主要来自于标签生成。图5(a)展示了计算开销随数据块大小的变化情况。结果表明,本文所提方法的计算开销远远低于其他两种方法,这一优势源于对标签生成机制的优化,减少了一次群上乘法操作和一次群上幂运算操作,实现计算效率的显著提升。此外,三种方法的计算时间基本维持不变,原因在于数据块个数的增加只会导致分区个数的增加,但是分区只会参与计算复杂度较低的有限域Z_p上的乘法和加法操作,对整体的计算开销影响不大。

图5(b)、5(c)和5(d)分别展示了计算开销与副本个数、数据块个数和云存储服务器个数之间的关系。结果表明,三种方法的计算开销都随着参数的增大而增加,但相比之下,本文方法依然具有显著的优势。特别是在处理大规模数据集时,本方法的优越性尤为突出。这些实验结果与初始化阶段计算开销的理论分析相吻合,并验证了本研究方法在减少计算成本方面的有效性。

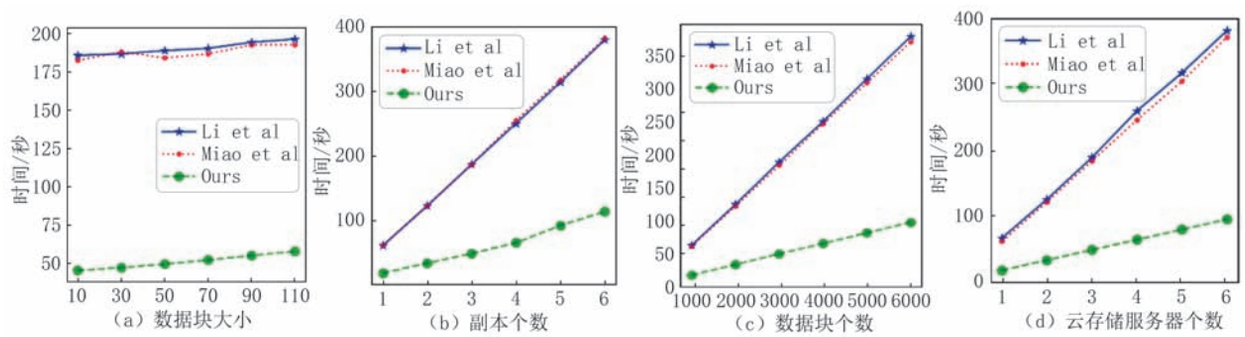

图5 初始化阶段计算开销与数据块大小(KB)、副本个数、数据块个数和云存储服务器个数之间的关系

7.3 审计阶段的开销

在审计阶段,本文比较了三种方法在证明生成和证明验证过程中的计算开销。

图6(a)展示了三种方法的证明生成所需的计算时间开销与数据块大小的关系。结果表明,三种方法证明生成的计算开销都随着数据块的增大而增加,但是本文方法的计算开销略大于其他两种方法。其原因在于本文方法实现了隐私保护,额外计算了一组参数$\{R_{pk}=e\left(MK_k,pk_{DO}\right)^{r_{pk}}\}_{1\leqslant k\leqslant s}$并用于生成证明$\{\delta_{pk}=r_{pk}+\sum_{j\in\omega_p}x_j\cdot\delta'_{pjk}\}$。随着数据分块大小的增加,分区的数量会相应增多,从而导致需要计算的参数数量也相应增加。图6(b)、6(c)和6(d)分别展示了三种方法的计算开销随副本个数、抽样数据块个数和云存储服务器个数的变化情况。结果表明,三种方法的计算开销非常接近且均随着对应参数的增加而逐渐上升。这是由于数据块的大小被固定设置为40KB,因此分区数量也保持恒定,使得隐私保护的计算开销保持稳定。然而,随着参数值的增加,证明生成过程中其他部分的计算开销开始显著增加,导致其在整体计算开销中所占的比例逐渐增加。所以本文方法的计算开销随着相应参数的增大而增加,与理论分析一致。

图7(a)显示了证明验证所需的计算开销与数据块大小的关系。可以看到,三种方法的计算开销都随着数据块大小的增大而增加,但是本文方

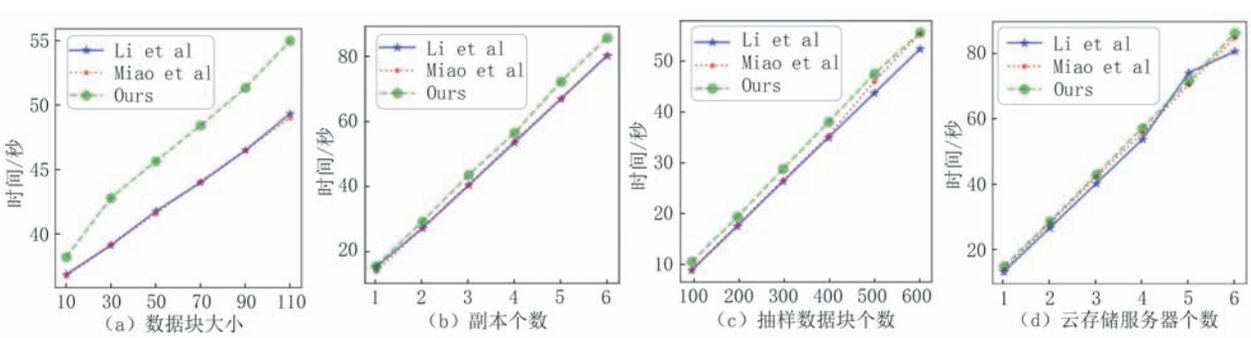

图6 审计阶段证明生成的计算开销与数据块大小(KB)、副本个数、抽样数据块个数和云存储服务器个数的关系

法明显优于其他两种方法。在验证过程中本文方法将群G_1上的累乘操作转化为累加操作，进而减少了计算开销。但是随着数据块大小的增加，公式(3)中的$\prod_{k=1}^{s} MK_k^{\delta_k}$的计算复杂度也会随之增加，所以整体计算开销会随着数据块大小的增大而增加。图7(b)、7(c)和7(d)分别展示了证明验证的计算时间随副本个数、抽样数据块个数和云存储服务器个数的变化情况。可以观察到，对比方法的计算时间都随着数据块大小的增加而增加，但本文方法基本不变。这是因为副本个数、抽样数据块个数和云存储服务器个数的增加只会导致公式(3)中$\sum_{p}^{l}\sum_{j\in\omega_p}\sum_{i=1}^{Sn} h_{s_i,j}$计算成本的增加，而加法操作的时间复杂度对整体计算开销影响不大，因此，本文方法验证过程中的计算开销基本维持不变。以上实验结果与理论分析相符，验证了本文方法能有效减少审计阶段的计算开销。

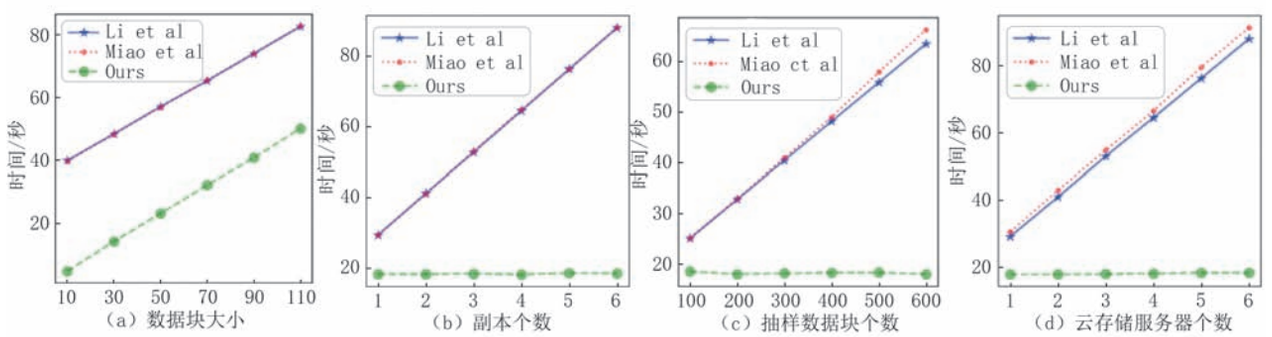

图7　审计阶段证明验证的计算开销与数据块大小(KB)、副本个数、抽样数据块个数和云存储服务器个数的关系

7.4　更新阶段的开销

对于更新阶段，实验比较了更新证明产生和验证的计算开销。由于Li等人的方法不支持数据动态更新，仅与Miao等人的方法进行比较。

图8(a)展示了更新证明产生时间与数据块大小之间的关系。从图中可以观察到，随着数据块大小的增加，本文方法的更新证明产生时间呈现出迅速增长的趋势，且大于Miao等人的方法。图8(b)、8(c)和8(d)分别展示了更新证明产生时间与副本个数、更新数据块个数和云存储服务器个数的关系，结果表明，本文方法计算开销高于Miao等人的方法。其原因在于本文实现了隐私保护。考虑到云存储服务器强大的计算能力，这些计算开销的增加对实际应用并不会构成显著影响。

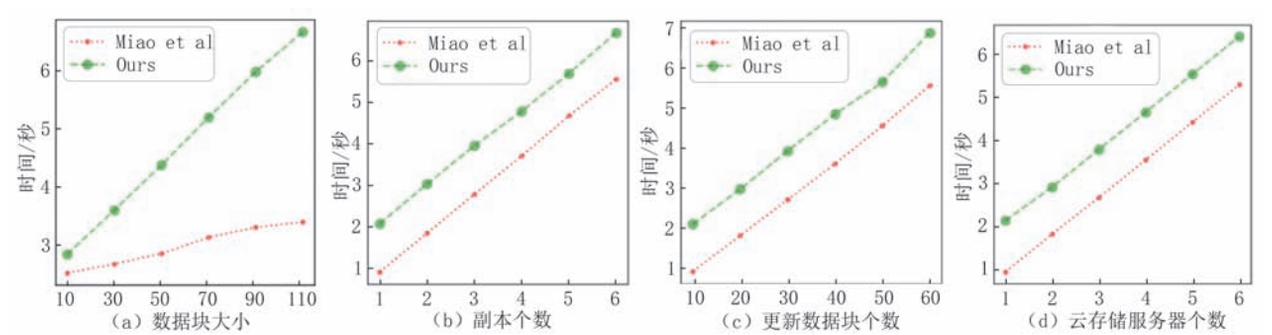

图8　更新阶段更新证明产生的计算开销与数据块大小(KB)、副本个数、更新数据块个数和云存储服务器个数的关系

图9(a)展示更新证明验证时间与数据分块大小的关系。结果表明，两种方法的计算开销随着数据分块的增大不断增加，但是本文方法略优于Miao等人的方法。图9(b)、9(c)和9(d)分别展示了更新证明验证的计算开销与副本个数、更新数据块个数和云存储服务器个数的关系。分析可知，Miao等人的方法计算开销远大于本文方法，这是因为群上幂运算的时间复杂度远远大于乘法运算且分区个数远大于副本个数，所以验证过程中的主要计算开销来源于公式中对$\prod_{k=1}^{s} MK_k^{\delta_{ik}}$的计算。随着云存储服务器个数的增加，公式中和的个数会随之增多，但是分区个数保持不变，因此验证的计算开销基本保持不变。图9(d)所示的实验结果中，本文方法的计算时间出现微小波动，这主要是由于实验过程中可能

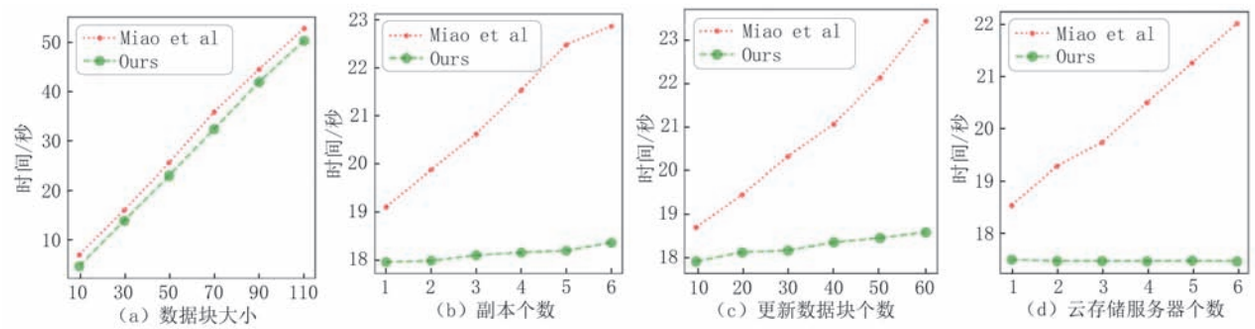

图9 更新阶段更新证明验证的计算开销与数据块大小(KB)、副本个数、更新数据块个数和云存储服务器个数的关系

存在的误差所致,并不影响总体趋势的正确性。

8 结 论

为了解决传统数据完整性审计模型中的第三方信任问题和多云多副本审计算法的高计算开销问题,本文提出了一种基于区块链的动态多云多副本数据完整性审计方法,通过在区块链上部署智能合约作为第三方审计员为用户和云存储服务器提供可信的审计结果,通过改进审计算法,显著降低了计算开销。此外,本文进行了全面的安全性分析和大量的实验,证明了该方法的安全性和有效性。本文提出方法同样适用于单云单副本的应用场景,在该场景下,云存储服务器将数据完整性证明提交到证明聚合和验证智能合约,智能合约就能验证数据的完整性。

为了实现数据的动态性,本文将副本索引表存储在区块链上,这样虽然大大降低了审计过程中的通信开销,但同时也带来了用户数据部分信息暴露的潜在风险,可能对数据安全构成威胁。因此,未来研究将继续探索能够存储在云服务器端的数据结构,以增强数据的安全性,并进一步优化审计过程。

参 考 文 献

[1] Uleta, How much data is created every day? [27 staggering stats]. Available: https://seedscientific.com/how-much-data-is-created-every-day/

[2] Sehgal N K, Bhatt P C P, Acken J M. Future trends in cloud computing. Cloud computing with security and scalability. Concepts and practices. Cham: Springer, 2022: 289-317

[3] Cheng L, Liu F, Yao D. Enterprise data breach: Causes, challenges, prevention, and future directions. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2017, 7(5): e1211

[4] Ali M, Khan S U, Vasilakos A V. Security in cloud computing: Opportunities and challenges. Information Sciences, 2015, 305: 357-383

[5] Wang Q, Wang C, Ren K, et al. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Transactions on Parallel and Distributed Systems, 2010, 22(5): 847-859

[6] Wang H, Zhang Y. On the knowledge soundness of a cooperative provable data possession scheme in multicloud storage. IEEE Transactions on Parallel and Distributed Systems, 2013, 25(1): 264-267

[7] Guo H, Yu X. A survey on blockchain technology and its security. Blockchain: Research and Applications, 2022, 3(2): 100067

[8] Zheng Z, Xie S, Dai H, et al. An overview of blockchain technology: Architecture, consensus, and future trends// Proceedings of the 2017 IEEE International Congress on Big Data. Honolulu, Hawaii, USA, 2017: 557-564

[9] Ateniese G, Burns R, Curtmola R, et al. Provable data possession at untrusted stores//Proceedings of the 14th ACM Conference on Computer and Communications Security. New York, USA, 2007: 598-609

[10] Tian H, Chen Y, Chang C C, et al. Dynamic-hash-table based public auditing for secure cloud storage. IEEE Transactions on Services Computing, 2015, 10(5): 701-714

[11] Rao L, Zhang H, Tu T. Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated merkle hash tree. IEEE Transactions on Services Computing, 2017, 13(3): 451-463

[12] WANG Ziyuan, DU Ruizhong. Certificateless public key cryptography based provable data possession scheme in edge environment. Journal on Communications, 2022,43(7):62-72 (王子园,杜瑞忠. 边缘环境下基于无证书公钥密码的数据完整性审计方案. 通信学报,2022,43(7):62-72)

[13] Yu J, Ren K, Wang C, et al. Enabling cloud storage auditing with key-exposure resistance. IEEE Transactions on Information Forensics and Security, 2015, 10(6): 1167-1179

[14] Yan Y X, Wu L, Xu W Y, et al. Integrity audit of shared cloud data with identity tracking. Security and Communication Networks, 2019, 2019

[15] Curtmola R, Khan O, Burns R, et al. MR-PDP: Multiple-replica provable data possession//Proceedings of the 28th International Conference on Distributed Computing Systems. Beijing, China, 2008: 411-420

[16] Barsoum A F, Hasan M A. Provable multicopy dynamic

data possession in cloud computing systems. IEEE Transactions on Information Forensics and Security, 2014, 10(3): 485-497

[17] Liu C, Ranjan R, Yang C, et al. MuR-DPA: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud. IEEE Transactions on Computers, 2014, 64(9): 2609-2622

[18] Zhou L, Fu A, Mu Y, et al. Multicopy provable data possession scheme supporting data dynamics for cloud-based electronic medical record system. Information Sciences, 2021, 545: 254-276

[19] Chang J, Shao B, Ji Y, et al. Efficient identity-based provable multi-copy data possession in multi-cloud storage, revisited. IEEE Communications Letters, 2020, 24(12): 2723-2727

[20] Li J, Yan H, Zhang Y. Efficient identity-based provable multi-copy data possession in multi-cloud storage. IEEE Transactions on Cloud Computing, 2019, 10(1): 356-365

[21] Gudeme J R, Pasupuleti S K, Kandukuri R. Certificateless multi-replica public integrity auditing scheme for dynamic shared data in cloud storage. Computers & Security, 2021, 103: 102176

[22] Wang H. Identity-based distributed provable data possession in multicloud storage. IEEE Transactions on Services Computing, 2014, 8(2): 328-340

[23] Zhou L, Fu A, Yang G, et al. Efficient certificateless multi-copy integrity auditing scheme supporting data dynamics. IEEE Transactions on Dependable and Secure Computing, 2020, 19(2): 1118-1132

[24] Guo Z, Zhang K, Wei L, et al. RDIMM: Revocable and dynamic identity-based multi-copy data auditing for multi-cloud storage. Journal of Systems Architecture, 2023: 102913

[25] Nakamoto S, Bitcoin A. A peer-to-peer electronic cash system. Bitcoin.-URL: https://bitcoin. org/bitcoin. pdf, 2008, 4(2): 15

[26] Yang R, Yu F R, Si P, et al. Integrated blockchain and edge computing systems: A survey, some research issues and challenges. IEEE Communications Surveys and Tutorials, 2019, 21(2): 1508-1532

[27] Gatteschi V, Lamberti F, Demartini C, et al. Blockchain and smart contracts for insurance: Is the technology mature enough? Future Internet, 2018, 10(2): 20

[28] Yue X, Wang H, Jin D, et al. Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control. Journal of Medical Systems, 2016, 40: 1-8

[29] McCorry P, Shahandashti S F, Hao F. A smart contract for boardroom voting with maximum voter privacy//Proceedings of the 21st International Conference Financial Cryptography and Data Security. Sliema, Malta, 2017: 357-375

[30] Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things. Ieee Access, 2016, 4: 2292-2303

[31] ZHOU Jiashun, Na WANG, Xuehui DU. Multi-party efficient audit mechanism for data integrity based on blockchain. Chinese Journal of Network and Information Security, 2021, 7(6): 113-125
(周家顺，王娜，杜学绘. 基于区块链的数据完整性多方高效审计机制. 网络与信息安全学报，2021, 7(6): 113-125)

[32] Miao Y, Huang Q, Xiao M, et al. Blockchain assisted multi-copy provable data possession with faults localization in multi-cloud storage. IEEE Transactions on Information Forensics and Security, 2022, 17: 3663-3676

[33] Yang X, Pei X, Wang M, et al. Multi-replica and multi-cloud data public audit scheme based on blockchain. IEEE Access, 2020, 8: 144809-144822

[34] Zhang C, Xu Y, Hu Y, et al. A blockchain-based multi-cloud storage data auditing scheme to locate faults. IEEE Transactions on Cloud Computing, 2021, 10(4): 2252-2263

[35] Tian Y, Tan H, Shen J, et al. Efficient identity-based multi-copy data sharing auditing scheme with decentralized trust management. Information Sciences, 2023, 644: 119255

WANG Chen-Xu, Ph. D., associate professor. His research interests include graph data mining, network and data security, and blockchain.

SUN Yi-Fan, M. S. His research focuses on network security and blockchain technology.

LIU Bo-Yang, M. S. candidate. His research interests include federated learning and blockchain technology.

SHI Jin-Chen, M. S. candidate. His research interests include privacy-preserving computation and blockchain technology.

SHEN Yan-Cheng, M. S. candidate. His research interests include data governance and deep learning.

WANG Wei, Ph. D., professor. His recent research interests focus on Web security, Android platform security and intrusion detection.

Background

The topic studied in this paper is multi-cloud multi-replica data integrity auditing in the field of cloud storage security. Multi-cloud multi-replica

data integrity auditing has long been studied because of its important applications in various fields such as data storage, data backup and recovery, and data sharing. At present, researchers focus on reliability and computational efficiency of auditing scheme with development of cloud storage. In this paper, we propose a dynamic multi-cloud multi- replica data integrity auditing scheme based on blockchain. We deploy smart contracts on the blockchain to perform data auditing, which can provide fully trusted auditing results and solve the third-party trust problem. We also optimize the tag generation mechanism in the auditing algorithm, which not only reduces the number of multiplication and power operations in tag generation process. It also transforms a large number of group multiplication operations into simpler addition operations in the verification process, thus reducing the computational overhead. In addition, we have conducted a rigorous security analysis and extensive experiments. The experimental results obtained by comparing with other data integrity auditing methods under different parameters verify the security and effectiveness of the method. The research presented in this paper is supported in part by National Key Research and Development Program of China (2021YFF0900904), National Natural Science Foundation of China (No. 62272379, No. T2341003, No. U22B2019, No. U21A20463, No. U22B2027, No. U23A20304), Natural Science Basic Research Plan in Shaanxi Province (2021JM-018), the Fundamental Research Funds for the Central Universities (xzy012023068), and the Beijing Municipal Fund-Blockchain(M23019).