软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software [doi: 10.13328/j.cnki.jos.007166]

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

支持错误定位与数据恢复的多云关键词审计*

薛婧婷[1,2,3], 罗抒琴[2], 张文政[1], 李发根[3], 周 宇[1], 张晓均[2]

[1](保密通信全国重点实验室 (中国电子科技网络信息安全有限公司 中国电子科技集团公司第三十研究所), 四川 成都 610041)

[2](西南石油大学 计算机与软件学院, 四川 成都 610500)

[3](电子科技大学 计算机科学与工程学院, 四川 成都 611731)

通信作者: 薛婧婷, E-mail: jtxue@swpu.edu.cn

摘 要: 基于关键词的审计 (KA) 技术是保障云审计经济适用性的重要手段. 不同于概率性审计对外包数据进行随机抽样验证, KA 考虑多用户多属性数据的审计需求, 执行关键词检索和定向审计, 能有效降低审计开销. 然而, 现有的 KA 方案通常聚焦于目标数据的审计效率, 而很少关注审计失败后的错误定位及数据恢复等补救措施; 这无益于保障数据的可用性. 因此, 提出基于关键词的多云审计方案 (简称 KMCA), 结合智能合约技术实现定向审计、批量错位定位与数据恢复功能. 具体来说, 定向审计模块借鉴可搜索加密技术的索引结构, 定义关键词-文件数据映射关系, 并利用布隆过滤器的误报率特性来隐藏审计词频, 保护关键词隐私; 错误定位模块采用二分思想实现出错云服务器批量定位和受损数据细粒度定位; 数据恢复模块提出多云冗余存储与数据恢复策略, 避免单点故障, 提升存储容错率. 在随机预言机模型下, KMCA 是可证明安全的. 性能分析表明, KMCA 具备可行性.

关键词: 基于关键词的审计; 批量错误定位; 数据恢复; 智能合约; 多云存储

中图法分类号: TP309

中文引用格式: 薛婧婷, 罗抒琴, 张文政, 李发根, 周宇, 张晓均. 支持错误定位与数据恢复的多云关键词审计. 软件学报. http://www.jos.org.cn/1000-9825/7166.htm

英文引用格式: Xue JT, Luo SQ, Zhang WZ, Li FG, Zhou Y, Zhang XJ. Keyword-based Multi-cloud Auditing with Fault Localization and Data Recovery. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7166.htm

Keyword-based Multi-cloud Auditing with Fault Localization and Data Recovery

XUE Jing-Ting[1,2,3], LUO Shu-Qin[2], ZHANG Wen-Zheng[1], LI Fa-Gen[3], ZHOU Yu[1], ZHANG Xiao-Jun[2]

[1](National Key Laboratory of Security Communication (Institute of Southwestern Communication, China Electronics Technology Cyber Security Co. Ltd.), Chengdu 610041, China)

[2](School of Computer Science and Software Engineering, Southwest Petroleum University, Chengdu 610500, China)

[3](School of Computer Science & Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract: Keyword-based auditing (KA) technology is a crucial measure to achieve cost-effectiveness in cloud auditing applications. Different from probabilistic auditing, which verifies outsourced data by random sampling and verification, KA considers the auditing requirements of multi-user and multi-attribute data by performing keyword searches and targeted audits. KA can significantly reduce auditing costs. However, existing KA schemes usually focus only on auditing the efficiency of target data while paying little attention to remedial measures such as fault localization and data recovery after audit failures. This lack of attention to remediation measures does not guarantee data availability. Therefore, this study proposes a keyword-based multi-cloud auditing scheme (referred to as KMCA) that

* 基金项目: 国家自然科学基金 (61902327); 通信安全重点实验室科技基金 (61421030107012102); 四川省自然科学基金 (2023NSFSC1398, 2022YFG0172, 2022JDRC0061); 成都市重点研发项目 (2021-YF05-00965-SN)
收稿时间: 2023-06-14; 修改时间: 2023-11-13; 采用时间: 2024-02-06; jos 在线出版时间: 2024-08-21

leverages smart contracts to enable targeted auditing, batch fault localization, and data recovery. Specifically, the targeted auditing module defines the keyword-file mapping based on the searchable encryption index structure and employs Bloom filters' false-positive rate characteristic to hide keyword frequency and protect privacy. The fault localization module uses a binary search approach to locate error-prone cloud servers in batches and fine-grained localization of corrupted data. The data recovery module formulates multi-cloud redundant storage and data recovery strategies to avoid single-point failure and improve storage fault tolerance. Under the random oracle model, KMCA is provably secure. Performance analysis shows that KMCA is feasible.

Key words: keyword-based auditing; batch fault localization; data recovery; smart contract; multi-cloud storage

在多用户数据审计场景中, 如智慧政务[1]的数据完整性验证, 不同业务部门 (财务、人力资源等) 的数据具有不同属性, 部门用户对数据的关注度也不尽相同. 传统的概率性审计无差别挑战所有数据, 能以不可忽略的概率准确判断外包数据的完整性, 但对于用户而言并不具备经济效益. 为了解决这一问题, 基于关键词的审计 (KA)[2]被提出来应用于此类多用户多属性数据审计场景. KA 从原始数据中提取关键词, 建立关键词索引, 以便用户定向获取关注数据的可用状态. KA 将有限的审计资源分配到受关注度高的数据, 既能满足用户得审计需求, 又能最大化利用计算资源. 然而, KA 依赖索引表实施定向挑战与审计, 面临着关键词隐私泄露、索引表构建高开销等安全和性能问题. 如何确保外包数据定向审计的词频隐私和经济适用性成为新的技术挑战.

此外, KA 在审计验证通过时能够保障挑战数据的完整性, 但当验证不通过时则会暴露出缺乏追责机制和无法事后补救等问题. 在集中式云 (单云) 存储模式中, 事后追责相对容易, 仅凭借审计结果即可向云存储服务提供商索取赔偿. 然而, 该模式存在单点故障问题. 2022 年阿里云曾发生长达 12 h 的宕机事故, 2018–2019 年间阿里云、腾讯云发生 4–5 次宕机事故, 亚马逊云科技 AWS、谷歌云计算、微软 Azure 也多次发生宕机事故. 因此多云存储模式[3,4]逐渐取代单云存储. 在多云存储模式中, 尤其是考虑到阿里云、腾讯云等云服务商过去发生的宕机事故, 用户数据的分布式存储要求在出现问题时能够准确地定位具体的错误实体. 追责机制虽然能够为用户提供一定的保障, 但并不能避免数据损失. 因此, 采用如多副本存储等数据冗余存储策略[5−7]来支持受损数据的恢复变得至关重要, 尽管这些策略可能带来更大的存储开销. 综上所述, 如何实现外包存储数据审计出错后的错误定位和高效恢复是可信存储技术发展过程中不可忽视的难题.

现有的错误定位方案通常采用遍历的方式检查所有存储证明, 执行效率不高. 文献 [8] 采用二分查找算法查找出错数据块, 虽提高了定位效率但受二分查找算法本身的限制, 一次执行仅能定位一个出错信息. 此外, 考虑到审计结果和定位结果直接影响数据可用性, 审计方案在设计时应保证实施过程的可监管性与公正性, 以确保审计与定位结果完全可信. 幸运的是, 区块链技术支持公开可信、可追溯的执行. 以太坊智能合约具备计算能力, 可辅助定向审计、错误定位的实现.

在此基础上, 本文提出了一个多云存储策略下的关键词审计方案 KMCA, 结合智能合约技术实现外包数据的定向审计、批量错误定位与数据恢复功能. 主要贡献总结如下.

(1) 设计了多云存储策略下的关键词审计方案. 借鉴可搜索加密技术的索引表构建, 实现用户自定义关键词与文件的映射关系以支持快速检索和定向审计. 为了隐藏挑战-应答模型中关键词被挑战频率 (词频), 基于布隆过滤器的误报率特性实现挑战关键词的模糊匹配. 有效平衡了审计方案的经济适用性和词频隐私保护.

(2) 提出了多云冗余存储策略和数据恢复算法. 评估备份和冗余处理模式下的数据可恢复性, 提出多用户多属性数据的重要性层级恢复算法.

(3) 实现了两类智能合约, 即定向审计合约和错误定位合约. 审计合约对应 KMCA 中的审计模块. 错误定位合约支持两种错误定位——出错云存储服务器的定位和受损分片数据的定位. 错误定位算法采用二分查找逻辑, (相比二分查找) 重写了查找循环的跳出条件, 支持批量错误定位和细粒度错误定位.

(4) 证明了 KMCA 在随机预言机模型下是安全的. 安全性分析表明方案具备数据标签的不可伪造性、完整性证明信息的抗替换攻击、外包数据机密性以及词频隐私等安全属性. 性能评估从理论分析与仿真实验结果表明了

方案的可行性. KMCA 在审计和错误定位阶段的方案性能均优于同类其他方案.

本文第 1 节回顾现有审计方案的研究现状. 第 2 节介绍相关基础知识, 包括椭圆曲线群和智能合约. 第 3 节介绍本文的系统模型、威胁模型、安全定义与设计目标. 第 4 节详细描述方案设计. 第 5 节和第 6 节分别展示方案的安全性证明和性能评估. 第 7 节总结全文.

1 相关工作

2007 年, Ateniese 等人[9]首次提出了检查外包数据完整性的方法, 即数据持有性证明协议 (PDP). 其中数据拥有者无需下载远端数据就可远程完成完整性验证. 2009 年, Wang 等人[10]在 PDP 审计架构中引入了一个可信第三方审计者 (TPA), 形成了完整性审计基本框架. 其中 TPA 代理用户以挑战-应答模式执行审计任务. 随后, 大量完整性审计方案被提出来. 文献 [11−16] 研究了提供数据动态审计、批量审计、错误定位、数据去重等功能的完整性审计协议. 早期审计方案聚焦在单云存储模式, 而该模式的低容错率问题日益突出. 若单云服务器主观删除篡改外包数据, 或因客观的软/硬件故障导致外包数据丢失, 都会对数据拥有者造成不可逆的损失.

为了降低单云存储中单点失效故障造成的数据拥有者损失, Zhu 等人[3]在 2012 年提出了多云存储的构想. 通过引入一个管理者来负责分发审计请求并聚合审计证明, 以实现多云存储的责任协调. 随后大量多云存储协议被提出. 文献 [4,17] 从信息分发效率、租赁成本等角度进一步完善多云存储模型. 考虑到多云存储若不进行数据冗余, 数据损失仍不可挽回, Li 等人[7]在 2022 年提出了一个高效的多云多副本审计协议. 该协议提供了高存储容错率, 允许在某个云故障或某个副本受损时从其他云的副本中恢复受损数据. 但该方案未解决出错云服务器与对应错误数据的定位问题. 此外, 云管理员的引入给系统带来了额外的安全问题, 如云管理员出错、合谋等. 庞晓琼等人[18]基于默克尔哈希树 (MHT) 设计了一个定位标签, 通过重新计算定位标签来查找出错的云服务器. 2022 年, Miao 等人[8]使用二分查找算法定位出错数据, 将查找效率提升至 $O(\log_2 n)$. 然而受二分查找算法本身的限制, 该方案只能快速查找存在唯一错误的情况, 并不适用于审计出错时错误数量未知的场景. Zhang 等人[19]提出了基于 RSA 签名实现的支持错误定位功能的审计协议, 但未对定位算法做出优化. 考虑到实体诚信影响错误定位结果的可信度, Su 等人[20]提出了支持错误定位的自审计方案. 多个云服务器通过多轮交互各自计算出一个存储证明, 以云服务器相互监督的方式来完成错误定位. 文献 [8,19] 结合去中心化的区块链技术提出了公平错误定位的概念.

随着区块链技术的日益成熟, 大量研究在解决审计实体诚信问题时将区块链和智能合约用于抵抗恶意 TPA. 2019 年, 朱昱锦等人[21]研究了区块链技术与云服务的适配性, 从功能性和安全性角度阐释了区块链在云计算领域数据-网络-计算-存储方面的潜在应用模型, 并提出了区块链技术可推进数字政府构建的观点. Xue 等人[22]提出了一种基于身份的公共审计方案, 其中区块链的随机数被用于构建不可预测且易于验证的挑战消息, 从而防止恶意 TPA 伪造审计结果以欺骗用户. 2020 年, Xu 等人[23]、Fan 等人[24]使用智能合约实现去 TPA 的链上审计方案, 根本地解决了第三方实体的诚信问题. 不同地, 文献 [23] 关注链上审计承诺的构造, 基于 RSA 算法构造存储证明, 以指数和求逆运算替换双线性对运算. 文献 [24] 着重实现链上审计灵活性, 支持周期性与非周期性两种审计模式. Du 等人[25−27]致力于降低链上审计开销, 证明链上审计的可行性. 具体地, 文献 [25] 提出了一个同态线性验证器, 用以保护链上数据隐私. 文献 [26] 基于零知识证明构造了一个安全审计承诺, 以提高审计协议与区块链技术的兼容性. 文献 [27] 扩展了数据的动态更新概念. 2023 年, 李涛等人[28]基于博弈论部署了多个智能合约, 以抵抗参与者合谋攻击. 为提高合约的运行效率, 该方案仍引入了 TPA 作为审计任务的执行者.

上述审计协议均采用 PDP 概率性审计范式, 将数据拥有者视为单一实体, 对外包数据进行抽样审计. 然而, 对于多用户多属性数据应用场景, 无差别审计所有数据对于数据拥有者来说不具备经济效益. 考虑到审计的经济适用性, Gao 等人[2]提出了基于关键词的审计. 根据数据拥有者指定的审计陷门 (如关键词的摘要

值), 审计包含关键词的云数据. 由于数据拥有者选择的关键词单一地决定了挑战信息的内容, 这使得恶意云服务器可通过词频分析出数据拥有者的审计偏好, 进而删除不包含指定关键词的数据. 这违背了外包存储数据的完整性审计初衷. 为了保护词频隐私, 前期工作[29]基于布隆过滤器对搜索陷门进行模糊匹配, 保证审计目标数据的同时实现词频隐私保护. 该工作聚焦在多属性数据审计中的数据隐私安全, 而未详细讨论多云存储环境下索引表的高效建立方法. 综上所述, 多云多用户场景中支持错误定位和高效恢复的外包数据可信审计方法仍待研究.

2 基础知识

2.1 椭圆曲线群

$\mathbb{G}$ 表示有限域 GF(p) 上的一个椭圆曲线群, G 是 $\mathbb{G}$ 的生成元, 其中 p 是一个大素数. $\mathbb{G}$ 内的运算分为加法与点乘 (记为 $[\![\cdot]\!]$), 点乘是连续的加法运算. 基于椭圆曲线群 $\mathbb{G}$, 简介本文基于的安全性假设.

椭圆曲线离散对数问题 (elliptic curve discrete logarithm problem, ECDLP): 设 $\mathbb{G}$ 是椭圆曲线群, 已知 $P, Q \in \mathbb{G}$, $Q = [\![a]\!]P$, 在概率性多项式时间 (PPT) 内求解 a 的概率为:

$$\Pr[a|P, Q = [\![a]\!]P] \to \epsilon.$$

若使用任意 PPT 算法求解 a 的概率 ϵ 是忽略不计的, 则求解 ECDLP 是困难的.

Schnorr[30]提出一个高效安全的基于乘法循环群的数字签名方案. 文献 [31] 证明该方案在随机预言机模型下是安全的. 基于 Schnorr 签名算法与 ECDLP, 构造椭圆曲线上的 Schnorr 数字签名算法. 该算法将支撑 KMCA 中审计协议的设计.

椭圆曲线上的 Schnorr 数字签名 (Schnorr signature on elliptic curve) 方案是一个概率多项式时间算法组成的三元组 (*KeyGen*, *SigGen*, *SigVerify*), 满足下列条件:

(1) *KeyGen*. 签名者选取随机数 $x \in Z_q^*$ 作为签名密钥, 计算 $y = [\![x]\!]G$ 作为验证密钥.

(2) *SigGen*. 对于消息 m, 签名者选取随机数 $k \in Z_q^*$, 计算 $K = [\![k]\!]G$, $e = H(m, K)$, $s = k - x \cdot e \pmod p$, 生成签名 (e, s).

(3) *SigVerify*. 接收者接收到消息 m 与签名 (e, s) 后, 计算 $\bar{K} = [\![s]\!]G + [\![e]\!]y$, 验证等式 $e = H(m, \bar{K})$ 是否成立.

2.2 智能合约

智能合约是以太坊系统[32]中的链上可执行程序, 以确定性方式运行在以太坊虚拟机 (EVM) 上, 具有简单的计算功能. 具体地, 以太坊区块链上的零地址交易完成智能合约的部署, 并通过交易传递参数来触发智能合约的运行 (程序的执行), 伴随智能合约的状态跳转和合约运行结果的输出. 以太坊交易的结构如表 1 所示.

表 1 以太坊交易的结构

字段	含义	字段	含义
Nonce	序列号, 表示外部账户 (交易发起方)创建的交易数量	Recipient	交易收款方地址, 合约地址或者外部账户地址 (相当于比特币系统交易中的To字段)
Gas price	Gas单价, 表示交易发起方愿意支付的Gas最高价格	Data	数据域, 记录交易中的附加数据 (可完成智能合约调用的参数传递)
Gas limit	Gas消耗数量, 表示交易发起方愿意在本交易中消耗的最大Gas数量	r, v	与交易发起方公钥相关的参数 (相当于比特币交易中的From字段)
Value	交易转账金额 (可以为空)	s	交易发起方的签名 (相当于比特币交易中的Signature字段)

对于合约创建类交易, Recipient 字段固定为“0x0”, 标志着该交易的目的为部署智能合约, 而 Data 字段写入编译后的智能合约代码. 对于调用智能合约的交易, Recipient 字段为智能合约地址, 而 Data 字段为智能合约运行所需的相关参数, 完成传参功能. 通过读写不同交易的相应字段, 可在以太坊区块链上部署/调用智能合约实现本文核心功能 (定向审计、错误定位), 并为系统提供公开可追溯、不可伪造的运行结果.

3 准备工作

3.1 系统模型

以智慧政务为场景介绍 KMCA. 系统模型图如图 1 所示, 包含 3 类实体: 政府部门组 (department group, $\mathcal{DG}$), 联合服务器 (joint server, $\mathcal{JS}$), 分布式网络 (distributed network, $\mathcal{DN}$).

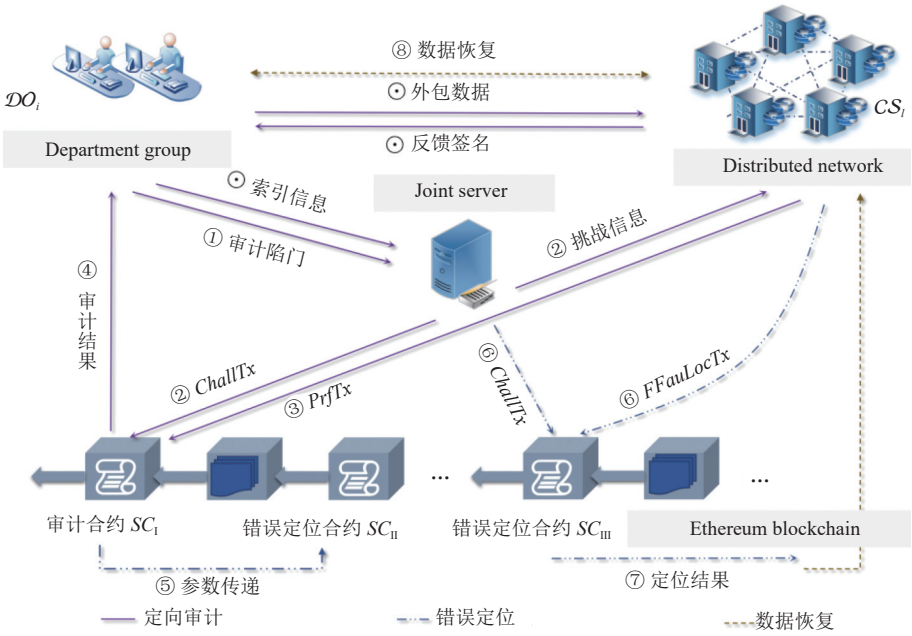


图 1 系统模型

- $\mathcal{DG}$ 由部门成员组成, 作为云存储模型中的数据拥有者 $\mathcal{DO}$, 管理持续增长的政务数据. $\mathcal{DO}$ 将政务数据外包存储到分布式存储网络中, 并制定完整性审计协议.
- $\mathcal{JS}$ 是一个由政府部门直接管理的可信私有服务器, 负责记录外包数据在分布式存储网络中的具体存储位置, 并根据审计协议负责挑战信息的生成. 此外, 为了提升审计结果可信度和受损数据的恢复效率, $\mathcal{JS}$ 部署智能合约来辅助外包数据的定向审计和审计出错后的错误定位.
- $\mathcal{DN}$ 是涉及多个独立云服务器 $\mathcal{CS}$ 的分布式存储网络. $\mathcal{CS}$ 出租空闲磁盘空间来存储 $\mathcal{DO}$ 的密态外包数据, 并根据审计协议反馈挑战块的存储证明, 触发智能合约完成审计程序的执行.

此外, KMCA 涉及一个以太坊区块链 (Ethereum blockchain, $\mathcal{EB}$). 各类实体与 $\mathcal{EB}$ 上的智能合约交互 (包括参数传递和结果读取) 以完成公开的存储证明验证与审计错误定位.

为了增强可读性, 以单个 $\mathcal{DO}$ 和 $\mathcal{CS}$ 为例, 基于图 1 简介 KMCA 的执行流程. 定向审计: $\mathcal{DO}$ 在本地处理原始政务数据 (如加密、冗余处理等), 并根据政务数据多云冗余存储策略将数据外包到 $\mathcal{DN}$ 的各个 $\mathcal{CS}$. 同时, 记录外包数据在 $\mathcal{CS}$ 上的存储位置, 并将位置信息发送给 $\mathcal{JS}$ (⊙). 准备工作完成后, 实体执行审计协议. $\mathcal{DO}$ 生成审计陷门并发送给 $\mathcal{JS}$ (①). $\mathcal{JS}$ 据此生成挑战信息并发送至相应的 $\mathcal{CS}$, 同时创建挑战交易 (②). $\mathcal{CS}$ 生成挑战信息对应的存储证明, 并创建证明交易 (③). 错误定位: 在验证挑战交易与证明交易的合法性后, 审计合约运行完成存储证明的验证 (④). 验证失败时, 审计合约调用错误定位合约运行 (⑤), 在接收定位所需交易后 (⑥) 定位出错证明所在的云服务器位置和具体数据分片位置 (⑦). 数据恢复: 在数据损坏甚至丢失的情况下, $\mathcal{DO}$ 与 $\mathcal{CS}$ 交互完成出错分片的数据恢复 (⑧).

3.2 符号定义

表 2 给出 KMCA 涉及的主要符号及其含义.

表 2 符号表

组成部分	符号	含义	符号	含义
方案主体	$\mathcal{DO}, \mathcal{JS}, \mathcal{CS}$	数据拥有者、联合服务器、云服务器	$Chall$	挑战信息
	$ID_{\mathcal{DO}}, ID_{\mathcal{JS}}, ID_{\mathcal{CS}}$	实体标识符	$\omega_k, \tilde{\omega}$	第 k 个关键词、 $Chall$ 涉及的关键词
	λ, pp	系统安全参数、系统公开参数	$\overrightarrow{\alpha_k} =< \alpha_{k1}, \alpha_{k2}, \ldots, \alpha_{kt} >^{\top}$	ω_k-F_i包含关系列向量
	$\mathbb{G}, p, G$	椭圆曲线群、大素数、 $\mathbb{G}$ 的生成元	$\vec{\beta}_i =< \beta_{i1}, \beta_{i2}, \ldots, \beta_{is} >$	F_i-CS_l存储关系行向量
	$[\![x]\!]G$	$\mathbb{G}$上的点乘运算	M_{ω_k}	ω_k的 F_i-CS_l 存储矩阵
	H_1, H_2, H_3	安全哈希函数	I	关键词多云存储索引表
	$sk_{\mathcal{DO}}, pk_{\mathcal{DO}}$	数据拥有者的签名密钥和验证密钥	$T, \hat{T}$	审计陷门、模糊审计陷门
	Enc, κ	对称加密算法、对称加密密钥	$\tilde{F}_l$	目标 CS_l 的拟审计分片集合
	$\pi, \varkappa$	伪随机函数、函数的保密参数	$\tilde{M}, \tilde{M}_c, \tilde{M}_r$	$\tilde{\omega}$的 F_i-CS_l 存储矩阵、 $\tilde{M}$ 的列向量、$\tilde{M}$ 的行向量
	$E = [e_{ii'}]_{m\times t}$	冗余编码矩阵	R_{CS}	智能合约接收证明的 CS 序号集合
	$D = [d_{ii'}]_{m\times m}$	冗余译码矩阵	$Prf = (P, \mu)$	存储证明
	$F^{\circ}, \overrightarrow{F^{\circ}}, f_i, \hat{m}_{ij}, \hat{c}_{ij}$	原始文件、原始分片向量、原始分片、原始数据块、密态数据块	$\mathcal{R} = [\![\gamma]\!]G, \gamma$	Prf 相关的盲化因子、盲化参数
	$\mathbb{F}, \vec{\mathbb{F}}, F_i, c_{ij}$	冗余分片集合、冗余分片向量、冗余分片、冗余密态数据块	$J, \{v_j\}_{j\in J}, \mathcal{T}$	随机挑战块序号、随机挑战块系数、目标云服务器随机系数
	K_{ij}, k_{ij}	标签验证标识、 K_{ij} 涉及的随机数	$\tilde{\Delta}_{CS}, \Delta_{CS}, ChalC_{\tilde{\omega}}, ChalC$	$\tilde{\omega}$的目标 CS 集合、目标 CS 集合、 $\tilde{\omega}$ 的目标 CS 向量、目标 CS 向量
	σ_{ij}	数据块标签	$Fault_{URS}, Fault_{Err}$	未响应型错误、非诚信存储型错误
	Φ, Φ_i	标签集合、 F_i 的标签	$Result = (Re, Fault)$	审计结果, 由审计状态 Re 与错误信息 $Fault$ 组成
	θ_{ij}	标签批量验证随机数	CS_{Err}, F_{Err}	出错云服务器、受损分片
智能合约	$SC_{\mathrm{I}}, SC_{\mathrm{II}}, SC_{\mathrm{III}}$	审计合约、定位出错云 CS_{Err} 的智能合约、定位受损分片 F_{Err} 的智能合约	$FFauLocTx$	CS_{Err}创建的定位交易(用于触发 SC_{III}, 完成 F_{Err} 的定位)
	$ChallTx$	$\mathcal{JS}$创建的挑战交易	$PrfTx$	CS创建的证明交易

3.3 威胁模型

在开放的对等网络环境中, $\mathcal{CS}$ 本身不是完全可信的, 其通信内容也可能被篡改. 好奇的 $\mathcal{CS}$ 可能尝试获取外包数据的内容, 以及审计阶段中挑战信息包含的隐私内容 (如关键词) 和其他 $\mathcal{CS}$ 的证明信息. 不诚实的 $\mathcal{CS}$ 可能为了隐瞒数据丢失或节省计算开销删除存储数据, 发起伪造攻击或替换攻击来欺骗验证者和用户. 在 KMCA 中, 不诚实的 $\mathcal{CS}$ 可能破坏关键词的审计词频隐私, 即尝试统计分析审计陷门, 将待审计数据缩小至一定范围, 进而删除其他文件以节省存储空间. 具体地, 伪造攻击指当外包数据及标签损坏甚至删除时, 不诚实的 $\mathcal{CS}$ 使用其他数据伪造结构合法的数据标签. 替换攻击指不诚实的 $\mathcal{CS}$ 试图用其他未受损的数据块与数据标签来生成合法的存储证明, 以通过完整性验证.

3.4 安全定义

本节设计挑战者 $\mathcal{C}$ 和敌手 $\mathcal{A}$ 之间的游戏, 展示 $\mathcal{A}$ 如何利用 $\mathcal{C}$ 的信息破解困难问题. KMCA 中数据拥有者 $\mathcal{DO}$ 扮演挑战者 $\mathcal{C}$, 云服务器 $\mathcal{CS}$ 扮演敌手 $\mathcal{A}$. 游戏如下.

游戏 1. $\mathcal{A}$ 试图伪造有效的数据标签.

(1) 初始化阶段. $\mathcal{C}$ 运行 $Setup$ 算法生成系统的公共参数 pp, 将 pp, $pk_{\mathcal{C}}$ 发送给 $\mathcal{A}$.

(2) 查询阶段. $\mathcal{A}$ 向 C 发起多项式时间查询. C 一一响应查询并做出诚实的回答.

哈希查询: $\mathcal{A}$ 发送 $(ID_{\mathcal{A}}, ID_C, c_{ij}, j)$ 给 C, C 计算 $(ID_{\mathcal{A}}, ID_C, c_{ij}, j)$ 的哈希值并返回给 $\mathcal{A}$.

标签查询: $\mathcal{A}$ 发送 $(ID_{\mathcal{A}}, ID_C, c_{ij}, j)$ 给 C, C 计算 $(ID_{\mathcal{A}}, ID_C, c_{ij}, j)$ 的标签并返回给 $\mathcal{A}$.

(3) 伪造阶段. $\mathcal{A}$ 输出 $(ID_{\mathcal{A}^*}, ID_{C^*}, c_{ij}^*, j^*)$ 的标签 σ_{ij}^*, 若 σ_{ij}^* 能通过 C 的验证, 则认为 $\mathcal{A}$ 赢得了该场游戏.

定义 1. 抵抗标签伪造攻击. 在多项式时间内, $\mathcal{A}$ 伪造有效数据标签通过验证的概率是忽略不计的, 即 KMCA 遭受 $\mathcal{A}$ 发起的伪造攻击时是安全的.

游戏 2. $\mathcal{A}$ 试图偷窥关键词与文件的对应关系并分析文件的审计频率.

(1) 初始化阶段. C 运行 *Setup* 算法生成系统的公共参数 pp, 将 pp 发送给 $\mathcal{A}$.

(2) 查询阶段. $\mathcal{A}$ 向 C 发起多项式时间查询. C 一一响应查询并做出诚实的回答.

索引查询: $\mathcal{A}$ 发送 ω_k 给 C, C 计算 ω_k 的查找索引并返回给 $\mathcal{A}$.

陷门查询: $\mathcal{A}$ 发送 ω_k 的组合 W 给 C, C 计算 W 的模糊搜索结果并返回给 $\mathcal{A}$.

(3) 伪造阶段. $\mathcal{A}$ 输出的 W^* 的模糊搜索结果 $\overrightarrow{F^*}$, 若 $\overrightarrow{F^*}$ 能通过 C 的验证, 则认为 $\mathcal{A}$ 赢得了该场游戏.

定义 2. 关键词-文件隐私保护. KMCA 保护关键词-文件分片的关系隐私与对应文件分片的审计频率隐私. 即在多项式时间内, $\mathcal{A}$ 无法分析存储文件分片与关键词的存储关系, 也无法统计对应文件分片的审计频率.

定义 3. 抵抗存储证明替换攻击. KMCA 是抗替换攻击的, 即不诚实的 $\mathcal{CS}$ 基于未被挑战的数据生成的存储证明无法通过审计验证.

定义 4. 数据机密性. KMCA 提供外包数据机密性, 即好奇的 $\mathcal{CS}$ 无法根据密文数据和审计信息猜测出正确的明文内容.

3.5 设计目标

基于系统模型与威胁模型, 从功能、安全与效率方面定义 KMCA 的设计目标, 具体如下.

功能: (1) 在多用户多属性数据应用场景下, 设计更具经济效益的审计方案, 即根据用户指定关键词审计特定文件数据. (2) 保证数据可用性, 实现细粒度错位定位, 包括出错云服务器定位与受损数据定位. (3) 结合多云存储环境, 制定冗余存储策略以支持高效的数据恢复.

安全: (1) 保证方案可抵抗安全威胁, 具体包括标签抗伪造攻击、证明信息抗替换攻击、审计词频隐私保护和外包数据机密性. (2) 审计结果是公开可信的, 错误定位结果是准确且完备的.

效率: 各实体的本地计算控制在 ms 级.

4 方案设计

4.1 概 述

KMCA 包含 3 个模块: 定向审计模块、错误定位模块与数据恢复模块. 定向审计模块采用 KA 范式, 基于挑战-应答机制审计包含关键词的数据块. 为了保护审计词频隐私, KMCA 原创性地采用了布隆过滤器实现模糊匹配来隐藏挑战信息中的关键词. 当审计合约输出验证失败结果时, 定位合约被触发 (等待参数传入), 错误定位模块算法采用二分思想, 改进了二分查找算法的循环跳出条件, 实现高效定位多个出错云服务器和文件分片. 数据恢复模块首次结合应用场景提出了具体的数据恢复措施和多云冗余存储策略, $\mathcal{DO}$ 通过该模块及时恢复受损文件分片, 降低数据丢失的风险.

4.2 定向审计模块

如图 2 所示, 定向审计模块分为系统初始化阶段 (*Setup*)、准备阶段 (包括 *DataProcess* , *IndexGen* , *Store*)、审计阶段 (包括 *TrapGen* , *Challenge* , *PrfGen* , *PrfVerify*), 涉及政府部门组 $\mathcal{DG}$ 、联合服务器 $\mathcal{JS}$ 、分布式网络 $\mathcal{DN}$ 这 3 类实体.

系统初始化阶段: 负责初始化系统公开参数.

● *Setup* 算法: 由 $\mathcal{JS}$ 运行, 基于系统安全参数 λ, 生成系统公开参数 pp.

1) 选择参数. 基于有限域 GF(p) 确定椭圆曲线群 $\mathbb{G}$, G 是 $\mathbb{G}$ 的生成元; 选择安全哈希函数 $H_1, H_2, H_3: \{0,1\}^* \to Z_q^*$; 选择伪随机数生成函数 $\pi: Z_q^* \times Z_q^* \to Z_q^*$; 选择对称加密算法 $Enc(m, key)$.

2) 公开参数. 生成公开参数 $pp = (Z_q^*, p, \mathbb{G}, G, H_1, H_2, H_3, \pi, Enc)$.

准备阶段: 负责生成外包的数据包.

● *DataProcess* 算法: 由 $\mathcal{DO}$ 运行, 基于原始文件 F°, 生成冗余文件分片集合 $\mathbb{F}$、标签信息集合 Φ、$F_i - CS_l$ 存储向量集合 β.

1) 数据加密. 将原始文件 F° 划分为 m 个分片, 即 $F^\circ = \{f_i\}_{i\in[1,m]}$. 将每个文件分片划分成 n 个数据块, 即 $f_i = \{\hat{m}_{ij}\}_{j\in[1,n]}$. 以数据块为加密单位, 得到密态数据块 $\hat{c}_{ij} = Enc(\hat{m}_{ij}, \kappa)$, κ 为加密密钥.

2) 冗余处理. 将原始文件 F° 表示为向量 $\overrightarrow{F^\circ} = < f_1, f_2, \ldots, f_m >$. 首先, 生成冗余编码矩阵 $E = [e_{ii'}]_{m\times t}$. 具体地, 当 $0 \leqslant i \leqslant m$ 时, $e_{ii'} = \begin{cases} 1, i = i' \\ 0, i \neq i' \end{cases}$; 当 $m+1 \leqslant i \leqslant t$ 时, $e_{i(m+1)} = \pi(i, \varkappa)$, $e_{ii'} = e_{i(m+1)}^{i-m}$, 其中 $i \in [1, m]$, 随机数 $\varkappa$ 作为伪随机函数的保密参数. 然后, 计算冗余分片向量 $\vec{\mathbb{F}} = \overrightarrow{F^\circ} \cdot E$, 向量元素即为冗余分片 $F_i = \sum_{i'=1}^{m} f_{i'} \cdot e_{ii'}$, 其中 $i \in [1, t]$. 最后, 生成冗余分片集合 $\mathbb{F} = \{F_i\}_{i\in[1,t]}$, $F_i = \{c_{ij}\}_{j\in[1,n]}$, 其中当 $0 \leqslant i \leqslant m$ 时, $c_{ij} = \hat{c}_{ij}$; 当 $m+1 \leqslant i \leqslant t$ 时, c_{ij} 为 $\hat{c}_{ij}$ 的校验数据块.

3) 标签生成. 首先, 计算标签验证标识 $K_{ij} = [\![\pi(i\|j, \varkappa)]\!]G$. 然后, 随机选择 $sk_{\mathcal{DO}} \in Z_q^*$ 作为签名密钥, 计算 $pk_{\mathcal{DO}} = [\![sk_{\mathcal{DO}}]\!]G$ 作为签名验证密钥. 最后, 计算数据块标签:

$$\sigma_{ij} = k_{ij} + (c_{ij} + H_1(ID_{\mathcal{DO}} \| ID_{\mathcal{CS}_l} \| j)) sk_{\mathcal{DO}},$$

其中, $k_{ij} = \pi(i\|j, \varkappa)$, 并生成标签集合 $\Phi = \{\Phi_i\}_{i\in[1,t]} = \{(K_{ij}, \sigma_{ij})\}_{i\in[1,t], j\in[1,n]}$.

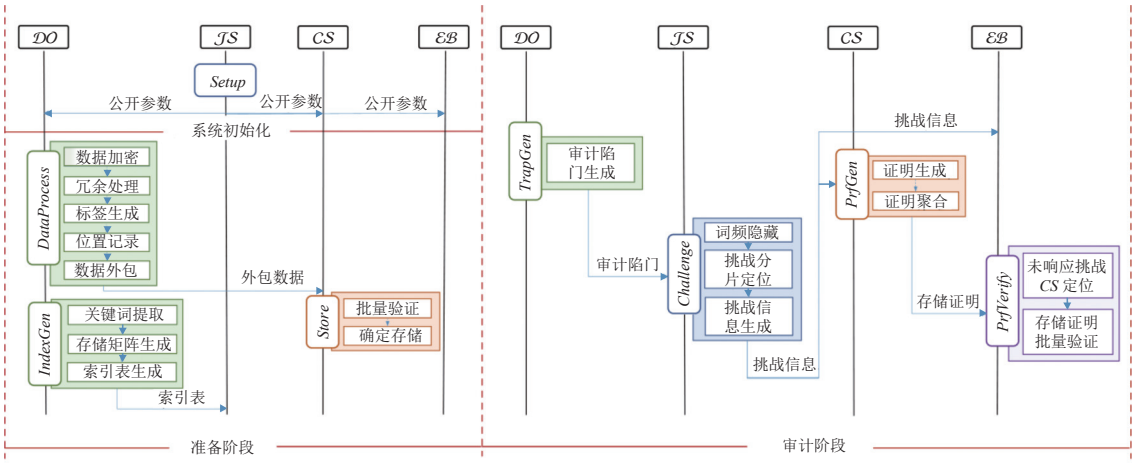


图 2 定向审计流程图

4) 位置记录. 根据多云冗余存储策略, 生成 $F_{i,\, i\in[1,t]}$-$\mathcal{CS}_{l,\, l\in[1,s]}$ 存储行向量 $\vec{\beta}_i = < \beta_{i1}, \beta_{i2}, \ldots, \beta_{is} >$. 具体地, 若 F_i 被外包存储到 $\mathcal{CS}_l$, 则 $\beta_{il} = 1$, 否则 $\beta_{il} = 0$.

5) 数据外包. 根据多云冗余存储策略, 将外包数据集合 $\{(F_i, \Phi_i)\}_{i\in[1,t]}$ 发送给对应的云服务器 $\mathcal{CS}_{l,\, l\in[1,s]}$.

● *IndexGen* 算法: 由 $\mathcal{DO}$ 运行, 基于原始文件 F°, 生成关键词多云存储索引表 I.

1) 关键词提取. 基于原始文件 F°, 提取 $\mathcal{K}$ 个自定义关键词 $\{\omega_k\}_{k\in[1,\mathcal{K}]}$ [33], 以摘要值 $\{H_2(\omega_k)\}_{k\in[1,\mathcal{K}]}$ 作为查找索引.

2) 存储矩阵生成. 根据分片 F_i 是否包含 ω_k, 生成 ω_k-$F_{i,\, i\in[1,t]}$ 关系列向量 $\overrightarrow{\alpha_k} = < \alpha_{k1}, \alpha_{k2}, \ldots, \alpha_{kt} >^\top$. 若分片 F_i 包含关键词 ω_k, 则 $\alpha_{ki} = 1$, 否则 $\alpha_{ki} = 0$. 根据多云冗余存储策略, 生成 ω_k 对应的 $s \times t$ 维 F_i-$\mathcal{CS}_l$ 存储矩阵 $M_{\omega_k} = (\overrightarrow{\alpha_k}, \overrightarrow{\alpha_k}, \ldots, \overrightarrow{\alpha_k})^\top \wedge (\overrightarrow{\beta_1}^\top, \overrightarrow{\beta_2}^\top, \ldots, \overrightarrow{\beta_t}^\top)$.

3) 索引表生成. 构造关键词多云存储索引表 $I = \{(H_2(\omega_k), M_{\omega_k})\}_{k=[1,\mathcal{K}]}$, 并发送给 $\mathcal{JS}$.

● *Store* 算法: 由云服务器 $\mathcal{CS}$ (以 $\mathcal{CS}_l$ 为例) 运行, 基于冗余分片集合 $\mathbb{F}$ 与标签集合 Φ, 输出存储状态 1/0.

1) 批量验证. 收到外包数据集合 $\{(F_i, \Phi_i)\}_{i\to\mathcal{CS}_l}$ 后. 首先计算 $h_{ij} = H_1(ID_{\mathcal{DO}}\|ID_{\mathcal{CS}_l}\|j)$, 然后选择随机数 $\theta_{ij} \in Z_q^*$ 来验证数据标签与数据块的一致性. 验证公式如下:

$$\left[\!\left[\sum\nolimits_{\beta_{il}=1}\sum\nolimits_{j=1}^{n}\theta_{ij}\sigma_{ij}\right]\!\right]G = \sum\nolimits_{\beta_{il}=1}\sum\nolimits_{j=1}^{n}\theta_{ij}K_{ij} + \left(\sum\nolimits_{\beta_{il}=1}\sum\nolimits_{j=1}^{n}\theta_{ij}(c_{ij}+h_{ij})\right)pk_{\mathcal{DO}}.$$

若验证成功, 向 $\mathcal{DO}$ 发送存储成功状态 1, 否则向 $\mathcal{DO}$ 发送存储失败状态 0.

审计阶段: 基于挑战-应答机制, 输出外包数据的完整性验证结果.

● *TrapGen* 算法: 由 $\mathcal{DO}$ 运行, 基于挑战关键词 $\tilde{\omega}$, 生成审计陷门 T.

1) 审计陷门生成. 基于挑战关键词集合 $\{\tilde{\omega}\}$, 生成审计陷门 $T = \{H_2(\tilde{\omega})\}$.

● *Challenge* 算法: 由 $\mathcal{JS}$ 运行, 基于审计陷门 T 生成挑战信息 $Chall$.

1) 词频隐藏. 使用布隆过滤器[29], 生成模糊审计陷门 $\hat{T} = \{BF(H_2(\tilde{\omega}))\}_{\tilde{\omega}\in T}$, $\hat{T} \supset T$.

2) 审计分片定位. 首先, 基于查找索引 $H_2(\tilde{\omega}) \in \hat{T}$, 查找存储矩阵 $\tilde{M}$. 然后, 计算目标 $\mathcal{CS}$ 向量 $ChalC_{\tilde{\omega}} = \tilde{M}_{c1} \vee \ldots \vee \tilde{M}_{ci} \vee \ldots \vee \tilde{M}_{ct}$, 其中 $\tilde{M}_{ci}$ 表示 $\tilde{M}$ 的第 i 列向量. 遍历 $ChalC_{\tilde{\omega}}$ 中元素值为 1 的序号并记录在 $\tilde{\omega}$ 的目标 $\mathcal{CS}$ 集合 $\tilde{\Delta}_{\mathcal{CS}}$ 中. 汇总所有非重复 $\tilde{\omega}$ 的目标 $\mathcal{CS}$ 集合, 生成审计目标 $\mathcal{CS}$ 集合 $\Delta_{\mathcal{CS}}$. 最后, 计算 $\mathcal{CS}_l$ 的拟审计分片向量 $\bigcup_{\tilde{\omega}\in\hat{T}}\tilde{M}_{rl}$, 并记录向量中元素值为 1 的序号, 生成 $\mathcal{CS}_l$ 拟审计分片集合 $\widetilde{F}_l$.

3) 挑战信息生成. 随机选择 c 个随机数 $J = \{j_1, j_2, \ldots, j_c\} \subseteq \{1, 2, \ldots, n\}$ 作为挑战块序号集合, 生成随机挑战块系数集合 $\{v_j\}_{j\in J}$, 然后构建目标云服务器随机系数集合 $\mathcal{T} = \{\tau_i\}_{i\in\widetilde{F}_l}$, 最后生成挑战信息:

$$Chall = (\Delta_{\mathcal{CS}}, \{\widetilde{F}_l\}, J, \{v_j\}_{j\in J}, \mathcal{T}).$$

● *PrfGen* 算法: 由 $\mathcal{CS}_l$ 运行, 基于挑战信息 $Chall$、本地存储冗余分片和标签信息集合 $\{F_i, \Phi_i\}_{i\to\mathcal{CS}_l}$, 生成存储证明 $Prf_{\mathcal{CS}_l}$.

1) 证明生成. 随机选择 $\gamma_{\mathcal{CS}_l i} \in Z_q^*$, $i \in \widetilde{F}_l$, 计算盲化因子 $\mathcal{R}_{\mathcal{CS}_l i} = [\![\gamma_{\mathcal{CS}_l i}]\!]G$. 生成单个拟审计分片的存储证明, 具体如下:

$$\begin{cases} P_i = \sum_{j\in J} v_j\sigma_{ij} + \gamma_{\mathcal{CS}_l i}H_3(\mathcal{R}_{\mathcal{CS}_l i}) \\ \mu_i = \sum_{j\in J} v_j(c_{ij}+h_{ij})pk_{\mathcal{DO}} + H_3(\mathcal{R}_{\mathcal{CS}_l i})\mathcal{R}_{\mathcal{CS}_l i} \end{cases}$$

2) 证明聚合. 聚合拟审计分片的存储证明 $Prf_{\mathcal{CS}_l} = (P_{\mathcal{CS}_l}, \mu_{\mathcal{CS}_l})$, 具体如下:

$$\begin{cases} P_{\mathcal{CS}_l} = \sum_{i\in\widetilde{F}_l}\tau_i P_i \\ \mu_{\mathcal{CS}_l} = \sum_{i\in\widetilde{F}_l}\tau_i\mu_i \end{cases}$$

● *PrfVerify* 算法 (以代码形式记录在审计合约 SC_1 对应交易的 Data 字段中): 基于目标云服务器的存储证明集合 $\{Prf_{\mathcal{CS}_l}\}_{l\in\Delta_{\mathcal{CS}}}$ 与挑战信息 $Chall$, 输出审计结果 $Result$.

1) 未响应挑战的目标云服务器定位. 首先, 初始化向量 $R_{\mathcal{CS}} =< 0, 0, \ldots, 0 >$, $|R_{\mathcal{CS}}| = s$ 表示云服务器响应挑战并反馈证明信息的情况. 然后, 根据按审计协议反馈存储证明的 $\mathcal{CS}_l$ 情况, 赋值 $R_{\mathcal{CS}}[l] = 1$. 最后, 检验 $R_{\mathcal{CS}} \oplus ChalC = 0$ 是否成立, 其中 $ChalC = \bigcup_{\tilde{\omega}\in\hat{T}} ChalC_{\tilde{\omega}}$. 若成立, 则更新本次审计状态为 $Re^{(1)} = 1X$, 其中 $1X$ 的第 1 位 1 表示所有目标云服务器已响应审计并反馈存储证明, 第 2 位 X 表示存储证明的验证结果待定. 若等式不成立, 则将未响应的 $\mathcal{CS}_l$ (即 $R_{\mathcal{CS}}[l] \oplus ChalC[l] = 1$) 写入 $Fault_{URS}$, 更新 $Re^{(1)} = 0X$ 和 $\Delta_{\mathcal{CS}}$.

2) 批量验证目标云服务器的存储证明. 聚合各云服务器存储证明, 批量验证等式 $\left[\!\left[\sum_{l\in\Delta_{\mathcal{CS}}} P_{\mathcal{CS}_l}\right]\!\right]G = \sum_{l\in\Delta_{\mathcal{CS}}}\mu_{\mathcal{CS}_l} + |\Delta_{\mathcal{CS}}|\sum_{i\in\widetilde{F}_l}\tau_i\sum_{j\in J}v_jK_{ij}$ 是否成立. 等式不成立时, SC_1 状态跳转到 Fail, 并输出审计结果 $Re = \begin{cases} 00, Re^{(1)} = 0X \\ 10, Re^{(1)} = 1X \end{cases}$, 否则状态跳转到 Pass, 并输出审计结果 $Re = \begin{cases} 01, Re^{(1)} = 0X \\ 11, Re^{(1)} = 1X \end{cases}$, $Fault_{Err} = \varnothing$. $Re = 00$ 和 $Re = 10$ 是触发错误定位合约的根据.

4.3 错误定位模块

文献 [8] 提出使用二分查找算法实现高效率的错误定位. 然而二分查找一次只能处理有且仅有一个错误的情况. 在多云审计中, 错误数量与错误位置是未知的, 因此不能直接使用二分查找进行错误定位. KMCA 错误定位模块采用二分查找逻辑, (相比二分查找) 重写查找循环的跳出条件, 实现批量错误定位和细粒度错误定位.

4.3.1 错误定位算法

ImporSearch 实现两类错误定位: 定位出错的云服务器和定位受损的文件分片. 伪代码如算法 1 所示. 具体地, 每次并行检验左右两半集合的正确性. 若两边均验证不通过, 则优先查找左半集合并标记右半集合, 左半子集查找完毕则回退最近的标记查找右半子集, 直至所有错误被查出. 在代码实现部分, 标记的存放与回退基于栈结构实现, 栈为空是查找完毕的充分条件.

算法 1. *ImporSearch*(low, $high$, C).

1. 初始化算法变量: 迭代次数: $i=0$, 中值索引: $mid=\dfrac{low+high}{2}$;
2. 将 C 平均划分为左右两个子集: $LHS_i=C[low],\ldots,C[mid-1]$, $RHS_i=C[mid],\ldots,C[high]$;
3. 初始化结果集合: ψ;
4. **for (** LHS_i 非空或 RHS_i 非空**) do**
5. **if** ($|LHS_i|==1$) **then**
6. **if** ($LrLHS_i \neq RrLHS_i$) **then**
7. 将 $LHS_i[0]$ 写入 ψ;
8. **end**
9. **return** ψ;
10. **end**
11. **if** ($|RHS_i|==1$) **then**
12. **if** ($LrRHS_i \neq RrRHS_i$) **then**
13. 将 $RHS_i[0]$ 写入 ψ;
14. **end**
15. **return** ψ;
16. **end**
17. **if** ($LrLHS_i == RrLHS_i$) **then**
18. **if** ($LrRHS_i == RrRHS_i$) **then**
19. **return** ψ;
20. **else**
21. $\psi = ImporSearch(mid, high, RHS_i++)$;
22. **end**
23. **else if** ($LrRHS_i == RrRHS_i$) **then**
24. $\psi = ImporSearch(low, mid-1, LHS_i++)$;
25. **end**
26. **else**
27. $\psi = ImporSearch(mid, high, RHS_i++)$;
28. **end**
29. **end**
30. **return** ψ;

举例阐释 *ImporSearch* 的执行过程. 假设外包数据集合 $\{D_1, D_2, \ldots, D_9\}$ 未通过本次完整性验证, 且受损分片为 $\{D_2, D_6, D_9\}$. 其错误定位过程如图 3 所示. 具体地, ① 将集合 $\{D_1, D_2, \ldots, D_9\}$ 二分为 $\{D_1, D_2, D_3, D_4, D_5\}$ 和 $\{D_6, D_7, D_8, D_9\}$. 当左右两半集合验证均不通过时, 将右半集合 $\{6, \ldots, 9\}$ 入栈, 并继续在左半集合进行查找; ② 将集合 $\{D_1, D_2, D_3, D_4, D_5\}$ 二分为 $\{D_1, D_2, D_3\}$ 和 $\{D_4, D_5\}$. 当左半集合不通过且右半集合通过时, 继续查找左半集合; ③ 将集合 $\{D_1, D_2, D_3\}$ 二分为 $\{D_1, D_2\}$ 和 D_3. 右半集合只有分片 D_3, 则该部分子集查找完毕. 左半集合继续二分查找, 输出受损分片 D_2; ④ 回退至最新标记 (即栈顶元素 $\{6, \ldots, 9\}$) 验证标记集合. 将集合 $\{D_6, D_7, D_8, D_9\}$ 二分为 $\{D_6, D_7\}$ 和 $\{D_8, D_9\}$. 当左右两半集合验证均不通过, 将右半集合 $\{8, 9\}$ 入栈, 并继续在左半集合进行查找; ⑤ 将集合 $\{D_6, D_7\}$ 二分为 D_6 和 D_7. 输出受损分片 D_6, 并回退至最新标记 (即 $\{8, 9\}$ 出栈) 验证标记集合; ⑥ 将集合 $\{D_8, D_9\}$ 二分为 D_8 和 D_9. 输出受损分片 D_9. 至此, 定位出全部受损数据分片.

本例 *ImporSearch* 共进行 6 步查找. 相比顺序查找 (9 步) 和二分查找 (在错误数量已知的情况下查找 $\log_2 9 + \log_2 8 + \log_2 7 \approx 8$ 步), *ImporSearch* 算法是高效的.

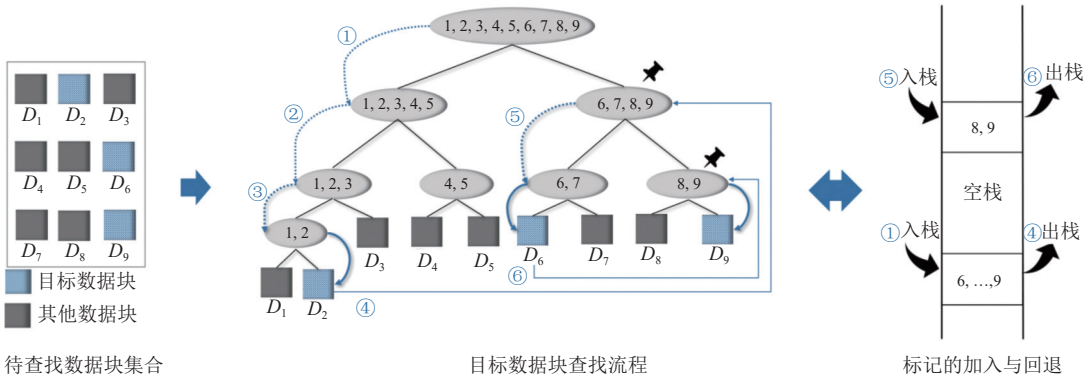


图 3 *ImporSearch* 算法的执行流程

4.3.2 错误定位合约

以出错云定位合约 SC_{II} 和受损分片定位合约 SC_{III} 表示 KMCA 实现的两类错误定位的智能合约实现.

SC_{II} 负责定位出错云服务器 CS_{Err}. 当审计合约 SC_{I} (SC_{I} 由 $\mathcal{JS}$ 创建的 *ChallTx* 交易与目标 CS 创建的 *PrfTx* 交易联合触发) 的执行状态为 Fail, $Re = 00/10$ 时调用 SC_{II}. SC_{II} 从 *ChallTx* 与 *PrfTx* 的数据域中读取合约代码运行所需参数 (即挑战信息与目标 CS 的聚合存储证明), 然后运行合约代码. 为了增强文章的可读性, 将证明验证公式的左侧 $\left[\left[\sum_{l\in\Delta_{CS}} P_{CS_l}\right]\right]G$ 记为 $Lr_{\Delta_{CS}}$, 公式右侧 $\sum_{l\in\Delta_{CS}} \mu_{CS_l} + |\Delta_{CS}| \sum_{i\in\tilde{F}_l} \tau_i \sum_{j\in J} v_j K_{ij}$ 记为 $Rr_{\Delta_{CS}}$. 第 i 次迭代查找过程中, 对于目标查找范围 SET_i, 智能合约通过验证 $Lr_{SET_i} = Rr_{SET_i}$ 确定该范围内是否有错误发生.

SC_{III} 负责定位受损分片 F_{Err}. 根据 SC_{II} 的运行结果, 出错的 CS_{Err} 创建 *FFauLocTx* 交易 (该交易包含独立的拟审计分片的存储证明) 触发 SC_{III}. SC_{III} 的运行结果即为某个 CS_{Err} 相关的所有受损文件分片集合 $\{F_{Err}\}$.

4.4 数据恢复模块

4.4.1 数据恢复算法

当外包数据集合的完整性审计失败, $\mathcal{DO}$ 读取错误定位合约输出的受损分片位置序号集合, 向分布式存储网络要求 CS 发送数据 (请求 m 个不重复的未受损分片), 以协助恢复受损分片数据. 基于多云存储协议, 数据恢复算法也委托新的云服务器执行.

假设受损分片是 F_{m_0}, 请求的未受损分片是 $\{F_{m_1}, F_{m_2}, \ldots, F_{m_m}\}$. 计算冗余译码矩阵 $D = [d_{ii'}]_{m\times m}$, 其中, 当 $m_{i'} \leqslant m$, $d_{ii'} = \begin{cases} 1, & i = i' \\ 0, & i \neq i' \end{cases}$; 当 $m_{i'} > m$, $d_{(m+1)i'} = \pi(i', \varkappa)$, $d_{ii'} = d_{(m+1)i'}^{i-m}$, $i = m+1, m+2, \ldots, t$, $i' = m_{i'}$. 已知冗余译码矩阵与原始

分片和请求的未受损分片存在如下数学关系 $D\cdot<f_1, f_2, \ldots, f_m>^{\top}=<F_{m_1}, F_{m_2}, \ldots, F_{m_m}>^{\top}$, 计算受损文件分片 $F_{m_0}=E_{r(m_0)}\cdot D^{-1}\cdot<F_{m_1}, F_{m_2}, \ldots, F_{m_m}>^{\top}$, 其中 $E_{r(m_0)}$ 是冗余编码矩阵 E 的第 m_0 行向量.

4.4.2 多云冗余存储策略

考虑到多属性数据 (如智慧政务场景的文件) 可用性和安全性需求以及 KA 的经济实用性目标, 本节介绍一个涉及文件安全等级、文件尺寸、文件所含关键词数目等属性的多云冗余存储策略, 具体参数为数据冗余度与预存储云服务器数目. 符号说明: $Fun^{+}(\cdot)$ 表示正相关函数, $Fun^{-}(\cdot)$ 表示负相关函数, θ 为影响因子, 其取值根据不同应用场景的需求变化. 该策略包括分片切割和冗余存储两个部分.

分片切割: 根据文件的尺寸与所含关键词数目决定原始分片数目. 考虑到存储效率, 文件尺寸越大, 分片数量越多; 考虑到关键词分布对审计分布的影响, 所含关键词数目越多, 分片数量越多. 即:

$$Split(FSize, KNum)\rightarrow m,\ m=\theta_1 Fun^{+}(FSize)+\theta_2 Fun^{+}(KNum).$$

冗余存储: 根据文件安全等级与文件大小决定文件的冗余度与预存储云服务器数目. 考虑到存储安全性, 文件安全等级越高, 冗余度越大, 涉及的云服务器越多; 考虑到分布式存储效率, 文件尺寸越大, 冗余度越低. 即:

$$RStore(FSLevel, FSize)\rightarrow(\eta, CSNum),\ \eta=\theta_3 Fun^{+}(FSLevel)+\theta_4 Fun^{-}(FSize),\ CSNum=Fun^{+}(\eta),$$

其中, 文件安全等级的影响占比远大于文件尺寸, 即 $\theta_3\gg\theta_4$.

基于该策略, 方案采取的 RSC (m, n) 可扩展为 RSC (m, n, η), 其中, η 是冗余度, $n=(1+\eta)m$. 该策略可参考文献 [34] 对最优冗余度的讨论.

5 安全性分析

5.1 正确性证明

为了公式的简洁性, 本节中记 $h_{ij}=H_1(ID_{\mathcal{DO}}\|ID_{CS}\|j)$. 正确性证明如下.

标签验证等式: $[\![\sigma_{ij}]\!]G=K_{ij}+(c_{ij}+h_{ij})pk_{\mathcal{DO}}$.

$$\begin{aligned}[\![\sigma_{ij}]\!]G&=[\![k_{ij}+(c_{ij}+h_{ij})\,sk_{\mathcal{DO}}]\!]G\\&=k_{ij}G+(c_{ij}+h_{ij})\,sk_{\mathcal{DO}}G\\&=K_{ij}+(c_{ij}+h_{ij})\,pk_{\mathcal{DO}}.\end{aligned}$$

标签批量验证等式: $\left[\!\left[\sum_{\beta_{il}=1}\sum_{j=1}^{n}\theta_{ij}\sigma_{ij}\right]\!\right]G=\sum_{\beta_{il}=1}\sum_{j=1}^{n}\theta_{ij}K_{ij}+\left(\sum_{\beta_{il}=1}\sum_{j=1}^{n}\theta_{ij}(c_{ij}+h_{ij})\right)pk_{\mathcal{DO}}$.

$$\begin{aligned}\left[\!\left[\sum_{\beta_{il}=1}\sum_{j=1}^{n}\theta_{ij}\sigma_{ij}\right]\!\right]G&=\left[\!\left[\sum_{\beta_{il}=1}\sum_{j=1}^{n}\theta_{ij}(k_{ij}+(c_{ij}+h_{ij})sk_{\mathcal{DO}})\right]\!\right]G\\&=\sum_{\beta_{il}=1}\sum_{j=1}^{n}\theta_{ij}[\![(k_{ij}+(c_{ij}+h_{ij})sk_{\mathcal{DO}})]\!]G\\&=\sum_{\beta_{il}=1}\sum_{j=1}^{n}\theta_{ij}([\![k_{ij}]\!]G+[\![(c_{ij}+h_{ij})sk_{\mathcal{DO}}]\!]G)\\&=\sum_{\beta_{il}=1}\sum_{j=1}^{n}\theta_{ij}(K_{ij}+(c_{ij}+h_{ij})pk_{\mathcal{DO}})\\&=\sum_{\beta_{il}=1}\sum_{j=1}^{n}\theta_{ij}K_{ij}+\left(\sum_{\beta_{il}=1}\sum_{j=1}^{n}\theta_{ij}(c_{ij}+h_{ij})\right)pk_{\mathcal{DO}}.\end{aligned}$$

单个文件分片存储证明验证等式: $[\![P_i]\!]G=\mu_i+\sum_{j\in J}v_jK_{ij}$.

$$
\begin{aligned}
[\![P_i]\!]G &= \left[\!\!\left[\sum_{j\in J} v_j\sigma_{ij} + \gamma_{CS_{li}} H_3(\mathcal{R}_{CS_{li}})\right]\!\!\right] G \\
&= \sum_{j\in J} [\![v_j(k_{ij} + (c_{ij}+h_{ij})sk_{\mathcal{DO}}) + \gamma_{CS_{li}} H_3(\mathcal{R}_{CS_{li}})]\!] G \\
&= \sum_{j\in J} v_j [\![k_{ij}]\!] G + \sum_{j\in J} v_j [\![(c_{ij}+h_{ij})sk_{\mathcal{DO}} + \gamma_{CS_{li}} H_3(\mathcal{R}_{CS_{li}})]\!] G \\
&= \sum_{j\in J} v_j K_{ij} + \sum_{j\in J} v_j [\![(c_{ij}+h_{ij})sk_{\mathcal{DO}}]\!] G + H_3(\mathcal{R}_{CS_{li}}) [\![\gamma_{CS_{li}}]\!] G \\
&= \left(\sum_{j\in J} v_j(c_{ij}+h_{ij})pk_{\mathcal{DO}} + H_3(\mathcal{R}_{CS_{li}})\mathcal{R}_{CS_{li}}\right) + \sum_{j\in J} v_j K_{ij} \\
&= \mu_i + \sum\nolimits_{j\in J} v_j K_{ij}.
\end{aligned}
$$

聚合文件分片的存储证明验证等式: $[\![P_{CS_l}]\!]G = \mu_{CS_l} + \sum_{i\in\tilde{F}_l}\tau_i \sum_{j\in J} v_j K_{ij}$.

$$
\begin{aligned}
[\![P_{CS_l}]\!]G &= \left[\!\!\left[\sum\nolimits_{i\in\tilde{F}_l}\tau_i\left(\sum\nolimits_{j\in J} v_j\sigma_{ij} + \gamma_{CS_{li}} H_3(\mathcal{R}_{CS_{li}})\right)\right]\!\!\right] G \\
&= \sum\nolimits_{i\in\tilde{F}_l}\tau_i \left[\!\!\left[\sum\nolimits_{j\in J} v_j(k_{ij} + (c_{ij}+h_{ij})sk_{\mathcal{DO}}) + \gamma_{CS_{li}} H_3(\mathcal{R}_{CS_{li}})\right]\!\!\right] G \\
&= \sum\nolimits_{i\in\tilde{F}_l}\tau_i \sum\nolimits_{j\in J} v_j [\![k_{ij}]\!] G + \sum\nolimits_{i\in\tilde{F}_l}\tau_i \left[\!\!\left[\sum\nolimits_{j\in J} v_j(c_{ij}+h_{ij})sk_{\mathcal{DO}} + \gamma_{CS_{li}} H_3(\mathcal{R}_{CS_{li}})\right]\!\!\right] G \\
&= \sum\nolimits_{i\in\tilde{F}_l}\tau_i \sum\nolimits_{j\in J} v_j K_{ij} + \sum\nolimits_{i\in\tilde{F}_l}\tau_i\mu_i \\
&= \mu_{CS_l} + \sum\nolimits_{i\in\tilde{F}_l}\tau_i \sum\nolimits_{j\in J} v_j K_{ij}.
\end{aligned}
$$

多个 CS_l 的聚合存储证明批量验证等式: $\left[\!\!\left[\sum_{l\in\Delta_{CS}} P_{CS_l}\right]\!\!\right] G = \sum_{l\in\Delta_{CS}}\mu_{CS_l} + |\Delta_{CS}| \sum_{i\in\tilde{F}_l}\tau_i \sum_{j\in J} v_j K_{ij}$.

$$
\begin{aligned}
\left[\!\!\left[\sum\nolimits_{l\in\Delta_{CS}} P_{CS_l}\right]\!\!\right] G &= \sum\nolimits_{l\in\Delta_{CS}}\sum\nolimits_{i\in\tilde{F}_l}\tau_i \left[\!\!\left[\left(\sum\nolimits_{j\in J} v_j\sigma_{ij} + \gamma_{CS_{li}} H_3(\mathcal{R}_{CS_{li}})\right)\right]\!\!\right] G \\
&= \sum\nolimits_{l\in\Delta_{CS}}\sum\nolimits_{i\in\tilde{F}_l}\tau_i \left[\!\!\left[\sum\nolimits_{j\in J} v_j(k_{ij} + (c_{ij}+h_{ij})sk_{\mathcal{DO}}) + \gamma_{CS_{li}} H_3(\mathcal{R}_{CS_{li}})\right]\!\!\right] G \\
&= \sum\nolimits_{l\in\Delta_{CS}}\left(\sum\nolimits_{i\in\tilde{F}_l}\tau_i \sum\nolimits_{j\in J} v_j [\![k_{ij}]\!] G + \sum\nolimits_{i\in\tilde{F}_l}\tau_i \left[\!\!\left[\sum\nolimits_{j\in J} v_j(c_{ij}+h_{ij})sk_{\mathcal{DO}} + \gamma_{CS_{li}} H_3(\mathcal{R}_{CS_{li}})\right]\!\!\right] G\right) \\
&= \sum\nolimits_{l\in\Delta_{CS}}\sum\nolimits_{i\in\tilde{F}_l}\tau_i \left[\!\!\left[\sum\nolimits_{j\in J} v_j(c_{ij}+h_{ij})sk_{\mathcal{DO}} + \gamma_{CS_{li}} H_3(\mathcal{R}_{CS_{li}})\right]\!\!\right] G + |\Delta_{CS}| \sum\nolimits_{i\in\tilde{F}_l}\tau_i \sum\nolimits_{j\in J} v_j K_{ij} \\
&= \sum\nolimits_{l\in\Delta_{CS}}\mu_{CS_l} + |\Delta_{CS}| \sum\nolimits_{i\in\tilde{F}_l}\tau_i \sum\nolimits_{j\in J} v_j K_{ij}.
\end{aligned}
$$

5.2 安全性证明

本节给出 KMCA 的安全性证明, 包括标签的不可伪造性、存储证明的不可替换性、关键词隐私不可推测性以及外包数据的机密性.

定理 1. 在多项式时间内, 敌手 $\mathcal{A}$ 利用公开信息与多轮查询无法伪造有效数据标签通过验证, 即 KMCA 可抵抗标签伪造攻击.

证明: 将敌手 $\mathcal{A}$ 的攻击分为 3 类: TypeA 为 $\mathcal{A}$ 未掌握 σ_{ij} 的构造方式, 其余两类为 $\mathcal{A}$ 掌握 σ_{ij} 的构造方式. 已知标签 $\sigma_{ij} = k_{ij} + (c_{ij} + H_1(ID_{\mathcal{DO}}\|ID_{CS}\|j))sk_C$, 挑战者 C 通过等式 $[\![\sigma_{ij}]\!]G = K_{ij} + (c_{ij} + H_1(ID_C\|ID_{\mathcal{A}}\|j))pk_C$ 验证 σ_{ij} 的合法性, 其中 $K_{ij} = [\![\pi(i\|j,\varkappa)]\!]G$.

TypeA : $\mathcal{A}$ 尝试通过标签验证, 即找到两个元素 x , y 满足 $[\![x]\!]G = K_{ij} + [\![y]\!]\,pk_c$. 过程如下.

初始化阶段: C 运行 $Setup$ 算法生成系统的公共参数 pp , 并将 pp, pk_C 发送给 $\mathcal{A}$.

查询阶段: $\mathcal{A}$ 询问有效标签的标签验证标识, C 响应 σ_{ij} 的验证标识 K_{ij} .

伪造阶段: $\mathcal{A}$ 选择 x 计算 $[\![x]\!]G - K_{ij}$ 寻找多项式算法 $PPA_1([\![x]\!]G - K_{ij}, pk_C) \to y$. 基于 ECDLP 安全假设, 算法 PPA_1 存在的概率可忽略不计.

TypeB : $\mathcal{A}$ 尝试伪造一个合法的标签通过标签验证, 即通过轮询 C, 得到两组有效的 σ_{ij}, c_{ij} 与 $H_1(ID_C \| ID_{\mathcal{A}} \| j)$, 伪造符合 σ_{ij} 构造结构的 $\sigma^*{}_{ij}$ 通过标签验证. 过程如下.

初始化阶段: C 运行 *Setup* 算法生成系统的公共参数 pp, 并将 pp 发送给 $\mathcal{A}$.

查询阶段: (1) $\mathcal{A}$ 请求有效密态数据与相应数据标签, C 响应 $c_{i_1 j_1}, \sigma_{i_1 j_1}$ 与 $c_{i_2 j_2}, \sigma_{i_2 j_2}$; (2) $\mathcal{A}$ 发送 $ID_{\mathcal{A}}$ 与 $c_{i_1 j_1}, c_{i_2 j_2}$ 给 C, C 响应对应的 $H_1(ID_C \| ID_{\mathcal{A}} \| j_1), H_1(ID_C \| ID_{\mathcal{A}} \| j_2)$.

伪造阶段: $\mathcal{A}$ 计算 $\sigma^*_{ij} = \sigma_{i_1 j_1} - \sigma_{i_2 j_2}$ 发送给 C, C 验证 $\left[\!\left[\sigma^*_{ij}\right]\!\right] G = K^*_{ij} + (c^*_{ij} + H^*_1(ID_C \| ID_{\mathcal{A}} \| j)) pk_C$.

若 $\mathcal{A}$ 伪造成功, 则 $\left[\!\left[\sigma^*_{ij}\right]\!\right] G = [\![k_{i_1 j_1} - k_{i_2 j_2}]\!] G + ((c_{i_1 j_1} - c_{i_2 j_2}) + (H_1(ID_C \| ID_{\mathcal{A}} \| j_1) - H_1(ID_C \| ID_{\mathcal{A}} \| j_2))) pk_C$ 成立, 则 $c^*_{ij} + H^*_1(ID_C \| ID_{\mathcal{A}} \| j) = (c_{i_1 j_1} - c_{i_2 j_2}) + (H_1(ID_C \| ID_{\mathcal{A}} \| j_1) - H_1(ID_C \| ID_{\mathcal{A}} \| j_2))$ 也成立.

然而, $K^*_{ij} = [\![\pi(i^* \| j^*), \varkappa]\!] G = [\![\pi(i_1 - i_2 \| j_1 - j_2), \varkappa]\!] G$, $k_{i_1 j_1} - k_{i_2 j_2} = \pi(i_1 \| j_1, \varkappa) - \pi(i_2 \| j_2, \varkappa)$. 函数 π 不具有线性特征, $[\![\pi(i_1 - i_2 \| j_1 - j_2), \varkappa]\!] G \neq \pi(i_1 \| j_1, \varkappa) - \pi(i_2 \| j_2, \varkappa)$, 故 $K^*_{ij} = [\![k_{i_1 j_1} - k_{i_2 j_2}]\!] G$ 不成立, C 验证 σ^*_{ij} 不通过.

TypeC : $\mathcal{A}$ 尝试伪造一个合法的标签, 即通过不断轮询 C, 得到多组有效的 $\sigma_{ij}, c_{ij}, H_1(ID_C \| ID_{\mathcal{A}} \| j)$ 与 K_{ij}, 通过试探 K_{ij} 之间的数学关系, 伪造有效标签. 过程如下.

初始化阶段: C 运行 *Setup* 算法生成系统的公共参数 pp, 并将发送给 $\mathcal{A}$.

查询阶段: (1) $\mathcal{A}$ 请求多轮标签验证标识, C 响应 $K_{i_1 j_1}$, …, $K_{i_n j_n}$; (2) $\mathcal{A}$ 请求 $K_{i_1 j_1}$, …, $K_{i_n j_n}$ 对应的有效密态数据与相应数据标签, C 响应 $c_{i_1 j_1}, \ldots, c_{i_n j_n}$ 与 $\sigma_{i_1 j_1}, \ldots, \sigma_{i_n j_n}$; (3) $\mathcal{A}$ 发送 $ID_{\mathcal{A}}$ 与 $c_{i_1 j_1}, \ldots, c_{i_n j_n}$ 给 C, C 响应 $H_1(ID_C \| ID_{\mathcal{A}} \| j_1)$, …, $H_1(ID_C \| ID_{\mathcal{A}} \| j_n)$.

伪造阶段: C 构造多项式算法 $PPA_2(K_{i_1 j_1}, \ldots, K_{i_2 j_2}) \to (K_{i_x j_x}, K_{i_y j_y}, K_{i_z j_z} = K_{i_x j_x} + K_{i_y j_y})$, 计算 $\sigma^*_{i_z j_z} = \sigma_{i_x j_x} + \sigma_{i_y j_y}$ 发送给 C, C 验证 $\left[\!\left[\sigma^*_{i_z j_z}\right]\!\right] G = K_{i_z j_z} + (c_{i_z j_z} + H^*_1(ID_C \| ID_{\mathcal{A}} \| j_z)) pk_C$.

设 $\mathcal{A}$ 成功构造 PPA_2 的概率为 a_1, 伪随机函数内部元素不发生碰撞的概率为 a_2, 则赢得 TypeC 游戏的优势为 $Adv : a_1 - a_2 \geqslant 0$. 基于伪随机函数安全性假设, $\Pr[Adv : a_1 - a_2 \geqslant 0] \leqslant negl(k)$, 因此 $\mathcal{A}$ 无法赢得 TypeC 游戏.

综上所述, $\mathcal{A}$ 既不能伪造合法的标签, 也不能伪造有效的信息通过标签验证, 故 KMCA 能抵抗伪造攻击.

定理 2. 多项式时间内, 敌手 $\mathcal{A}$ 无法通过分析已存在的 $W \to \vec{F}$ 推测出具体的 ω_k 对应的 F_i, 也无法统计 F_i 的实际审计频率, 即 KMCA 保护关键词-文件隐私.

证明: $\mathcal{A}$ 尝试根据审计陷门窥探关键词信息, 尝试推测审计陷门与文件分片的对应关系. 过程如下.

初始化阶段: C 运行 *Setup* 算法生成系统的公共参数 pp, 将 pp 发送给 $\mathcal{A}$.

查询阶段: (1) $\mathcal{A}$ 发送不同的 ω_1, ω_2, ..., ω_n 给 C, C 响应对应的查找索引 $H_2(\omega_1)$, $H_2(\omega_2)$, …, $H_2(\omega_n)$; (2) $\mathcal{A}$ 构造一个审计陷门 $T = \{H_2(\omega_{k_1})\}$ 发送给 C, C 响应对应的文件分片组合 $\vec{F} = \{F_{i_1}, F_{i_2}, F_{i_3}\}$; (3) $\mathcal{A}$ 构造另一个审计陷门 $T' = \{H_2(\omega_{k_1}), H_2(\omega_{k_2})\}$ 发送给 C, C 响应对应的文件分片组合 $\overrightarrow{F'} = \{F_{i_1}, F_{i_2}, F_{i_3}, F_{i_4}, F_{i_5}\}$.

伪造阶段: (1) $\mathcal{A}$ 构造多项式算法 $PPA_3(H_2(\omega_k)) \to \omega_k$, 尝试获取陷门包含的关键词; (2) $\mathcal{A}$ 通过 T 与 $\vec{F}$, 推测出 $F_{i_1}, F_{i_2}, F_{i_3}$ 包含 $H_2(\omega_{k_1})$, 根据 PPA_3 解密 ω_{k_1}, 猜测 $F_{i_1}, F_{i_2}, F_{i_3}$ 文件类型; 通过不同的 T 与对应 $\vec{F}$, 进一步确定各文件类型; (3) $\mathcal{A}$ 通过 T' 与对应 $\overrightarrow{F'}$, 窥探文件间的交叉关系; (4) $\mathcal{A}$ 构造陷门 $\overrightarrow{T^*}$, 伪造文件分片组合 $\overrightarrow{F^*}$, 将 $\overrightarrow{T^*}$ 发送给 C, 若 C 返回的文件分片组合与 $\overrightarrow{F^*}$ 一致, 则 $\mathcal{A}$ 伪造 $W \to \vec{F}$ 关系成功, 成功偷窥关键词-文件隐私.

假设 $\mathcal{A}$ 构造多项式算法 PPA_3 的概率为 b_{11}, 哈希函数元素发生碰撞的概率为 b_{12}, $\mathcal{A}$ 伪造阶段 (1) 的优势为 $Adv_1 : b_{11} - b_{12} \geqslant 0$; $\mathcal{A}$ 通过陷门锁定文件分片的概率为 b_{21}, 布隆过滤器误报率为 b_{22}, $\mathcal{A}$ 伪造阶段 (2)(3) 的优势为 $Adv_2 : b_{21} - b_{22} \geqslant 0$. 根据哈希函数的抗碰撞性, 可知 $\mathcal{A}$ 成功偷窥关键词隐私的概率 $\Pr[Adv_1] \leqslant negl(k)$. 由于 $b_{21} = C^{|\hat{T}|}_{|\vec{F}|} / A^{|\hat{T}|}_{|\vec{F}|}$, $|\hat{T}| = (1 + b_{22})|T|$, $|\overrightarrow{\hat{F}}| > |\vec{F}|$, 有 $b_{22} \neq 0$ 时, $b_{21} \to 0$, 故 $\mathcal{A}$ 成功关键词-文件隐私的概率 $\Pr[Adv_2] \leqslant negl(k)$.

综上所述, $\mathcal{A}$ 成功伪造 $W \to \vec{F}$ 关系的概率可忽略不计, 即 $\mathcal{A}$ 无法通过分析已存在的 $W \to \vec{F}$ 推测出具体的 ω_k 对应的 F_i, 也无法统计 F_i 的实际审计频率, KMCA 具备关键词隐私不可推测性.

定理 3. 多项式时间内, 敌手 $\mathcal{A}$ 无法利用挑战信息与有效的存储证明, 使用未被挑战的数据块替换被挑战数据块, 从而生成可通过完整性验证的存储证明, 即 KMCA 可抵抗存储证明的替换攻击.

证明: 已知数据块存储证明生成如下:

$$P=\sum_{i\in\bar{F}_l}\tau_i\left(\sum_{j\in J}v_j\sigma_{ij}+\gamma_{CS_li}H_3(\mathcal{R}_{CS_li})\right),$$

$$\mu=\sum_{i\in\bar{F}_l}\tau_i\left(\sum_{j\in J}v_j(c_{ij}+h_{ij})pk_C+H_3(\mathcal{R}_{CS_li})\mathcal{R}_{CS_li}\right),$$

其中, $h_{ij}=H_1(ID_C\|ID_{\mathcal{A}}\|j)$. 存储证明的验证 $[\![P]\!]G=\mu+\sum_{i\in\bar{F}_l}\sum_{j\in J}\tau_iv_jK_{ij}$.

假设被挑战的数据块中, $\mathcal{A}$ 缺失 $\sigma_{i_1j_1}$, $c_{i_1j_1}$, 并尝试使用 σ'_{ij}, c'_{ij} 替换 $\sigma_{i_1j_1}$, $c_{i_1j_1}$ 来生成合法存储证明, 具体如下:

$$P'=\sum_{i\in\bar{F}_l\setminus i_1}\tau_i\left(\sum_{j\in J\setminus j_1}v_j\sigma_{ij}\right)+\sum_{i\in\bar{F}_l}\tau_i(\gamma_{CS_li}H_3(\mathcal{R}_{CS_li}))+\tau_{i_1}v_{j_1}\sigma'_{ij},$$

$$\mu'=\sum_{i\in\bar{F}_l\setminus i_1}\tau_i\left(\sum_{j\in J\setminus j_1}v_j(c_{ij}+h_{ij})\right)pk_C+\sum_{i\in\bar{F}_l}\tau_iH_3(\mathcal{R}_{CS_li})\mathcal{R}_{CS_li}+(\tau_{i_1}v_{j_1}c'_{ij})pk_C.$$

随后 C 验证 $[\![P']\!]G=\mu'+\sum_{i\in\bar{F}_l}\sum_{j\in J}\tau_iv_jK_{ij}$. 因为每个 σ_{ij} 对应的 k_{ij} 是不同的, 即:

$$\sum_{i\in\bar{F}_l}\sum_{j\in J}\tau_iv_jK_{ij}\neq\sum_{i\in\bar{F}_l\setminus i_1}\sum_{j\in J\setminus j_1}\tau_iv_jK_{ij}+\tau_{i_1}v_{j_1}K'_{ij},$$

则伪造的 P', μ' 不能通过完整性验证. 综上所述, KMCA 能抵抗替换攻击.

定理 4. 在多云存储中, 敌手 $\mathcal{A}$ 不能通过公开信息与存储证明推导出其他 CS 存储的数据, 敌手 $\mathcal{A}$ 也不能根据密态数据推导出明文数据.

证明: 根据 $\mathcal{A}$ 获取具体信息的程度, 将 $\mathcal{A}$ 的攻击行为分为两类.

Case1. $\mathcal{A}$ 根据公开信息与存储证明推导出其他 CS 存储的数据块. 假设每轮挑战信息 $|J|=k$, 则 CS_l 每轮单个文件分片证明为 $P_i=\sum_{j=j_1}^{j_k}v_j\sigma_{ij}+\gamma_{CS_li}H_3(\mathcal{R}_{CS_li})$, $\mu_i=\sum_{j=j_1}^{j_k}v_j(c_{ij}+h_{ij})pk_{CS_l}+H_3(\mathcal{R}_{CS_li})\mathcal{R}_{CS_li}$. 求解 μ_i 内容的难度等同于求解 ECDLP, 故 $\mathcal{A}$ 试图求解 P_i. 通过 P_i 构造公式可知, P_i 含 $k+1$ 个的未知数 $\{\sigma_{ij}\}_{j\in[1,k]}$, γ_{CS_li}.

设 $\mathcal{A}$ 收集 k 轮 P_1 及对应的 $\{J_i\}_{i\in[1,k]}$, $\{v_{ij}\}_{i\in[1,k],\,j\in J_i}$, 尝试求解 σ_{11}, σ_{12}, …, σ_{1k}, 令 $r_1=\gamma_{CS_l1}H_3(\mathcal{R}_{CS_l1})$, 即求解 $k+1$ 元线性方程组. 求解矩阵 $A_{k\times(k+1)}\cdot B_{(k+1)\times1}=C_{k\times1}$ 的过程如下:

$$\begin{bmatrix}v_{11}&v_{12}&\cdots&v_{1k}&1\\v_{21}&v_{22}&\cdots&v_{2k}&1\\\vdots&\vdots&\ddots&\vdots&\vdots\\v_{k1}&v_{k2}&\cdots&v_{kk}&1\end{bmatrix}\begin{bmatrix}\sigma_{11}\\\sigma_{12}\\\vdots\\\sigma_{1k}\\r_1\end{bmatrix}=\begin{bmatrix}P_1^{(1)}\\P_1^{(2)}\\\vdots\\P_1^{(k)}\end{bmatrix}.$$

已知 $R(A)\leqslant k<k+1$, 即 A 非满秩矩阵, 根据线性方程组的求解条件可知该方程组具有无限解. 故 $\mathcal{A}$ 无法解密出其他 CS 存储的数据块.

Case2. $\mathcal{A}$ 根据密态数据推导出明文数据.

由于 $c_{ij}=Enc(m_{ij},\kappa)$, 加密方式为对称加密, κ 是 C 私有的密钥. 在未获取到 κ 的情况下, $\mathcal{A}$ 解密出明文数据的概率忽略不计的[35].

综上所述, KMCA 提供外包数据机密性.

6 性能分析

6.1 性能评估

本节评估 KMCA 的性能, 并与代表性多云审计方案 IBPDP[7], MCPDP[8], DSAS[20]进行对比. 在计算开销评估中, BP 表示双线性对运算, $MUL_{Z_q^*}$ 与 $MUL_{\mathbb{G}}$ 分别表示 Z_q^* 群上的普通乘法运算与椭圆曲线群上的乘法运算, $Add_{Z_q^*}$ 与 $Add_{\mathbb{G}}$ 分别表示 Z_q^* 群上的普通乘法运算与椭圆曲线群上的加法运算, EXP 表示 Z_q^* 群上的指数运算. 逻辑运算(与/或) 和哈希运算在评估中忽略不计.

6.1.1 通信开销

在系统初始化阶段, 评估 *Setup* 与 *DataProcess* 算法的通信开销. 由于各方案的算法设计不尽相同, *Setup* 算法对应其他方案中的 *KeyGen* , 评估公私钥传输的通信开销; *DataProcess* 算法对应其他方案中的 *TagGen* 与 *SigGen* , 评估传输文件数据块、数据块标签及对应签名的通信开销.

在审计阶段, 评估 *Challenge* , *PrfGen* 算法的通信开销. 在挑战信息生成时, 对比 IBPDP 与 MCPDP 仅选择随机挑战种子与挑战数, KMCA 生成挑战随机序号与系数; 对比 DSAS , KMCA 增加了被挑战分片的挑战系数. 在证明生成阶段, KMCA 共有 $|\Delta_{CS}|$ 个 CS 运行 *PrfGen* 算法, 通信总开销为 $2|\Delta_{CS}||Z_q^*|+|\Delta_{CS}||\mathbb{G}|$, 而 IBPDP , MCPDP 中所有 CS . 需运行 *PrfGen* 算法, DSAS 中每个 CS 都需要运行, 且每个 CS 需要与 k 个其余 CS 进行两轮交互以得到最终审计证明. 通信开销对比如表 3 所示, 在安全级别相同的情况下, $|Z_q^*|$ 与 $|G^*|$ 的尺寸远大于 $|\mathbb{G}|$. 故 KMCA 的通信开销优于对比方案.

表 3 通信开销对比

方案	系统初始化阶段		审计阶段	
	Setup	*DataProcess*	*Challenge*	*PrfGen*
IBPDP	$\|G^*\|$	$n(\|G^*\|+v\|Z_q^*\|)$	$2\|Z_q^*\|$	$(s+3)\|G^*\|+(s+t)\|Z_q^*\|$
MCPDP	$2\|G^*\|+3\|Z_q^*\|$	$n(\|G^*\|+v\|Z_q^*\|)+\|G^*\|$	$3\|Z_q^*\|+\log_2 n$	$(2+2s)\|Z_q^*\|+2\|G^*\|$
DSAS	$2\|G^*\|+\|Z_q^*\|$	$n\|G^*\|+(s+n)\|Z_q^*\|$	$2c\|Z_q^*\|+c$	$4k\cdot s(\|G^*\|+\|Z_q^*\|)$
Ours	$\|Z_q^*\|+\|\mathbb{G}\|$	$n(2\|Z_q^*\|+\|\mathbb{G}\|)$	$\|\Delta_{CS}\|+\|\widetilde{F}_l\|(1+\|Z_q^*\|)+c(1+\|Z_q^*\|)$	$2\|\Delta_{CS}\|\|Z_q^*\|+\|\Delta_{CS}\|\|\mathbb{G}\|$

6.1.2 计算开销

评估 KMCA 在标签生成 (*TagGen*)、证明生成 (*PrfGen*)、证明验证 (*PrfVerify*) 各步骤与代表性方案的计算开销对比. 如表 4 所示, KMCA 的标签生成仅需一次 $MUL_{Z_q^*}$ 与 $ADD_{Z_q^*}$, 而其余方案涉及多次 $MUL_{Z_q^*}$ 与 $ADD_{Z_q^*}$, 并引入 *EXP* 运算. 在证明生成时, KMCA 增添了 $Add_{\mathbb{G}}$, $MUL_{\mathbb{G}}$, 而 IBPDP , MCPDP 涉及 *EXP* 运算. 在 160 位安全级别中, *EXP* 计算开销大于 $Add_{\mathbb{G}}$ 与 $MUL_{\mathbb{G}}$. 证明验证时, IBPDP , MCPDP , DSAS 引入计算开销远高于其他运算操作的 *BP* , 而 KMCA 仍只涉及 $ADD_{Z_q^*}$, $Add_{\mathbb{G}}$, $MUL_{\mathbb{G}}$, 并且 KMCA 每种运算执行次数低于对比方案. 因此, 理论上看, KMCA 在标签生成 (*TagGen*)、证明生成 (*PrfGen*)、证明验证 (*PrfVerify*) 各步骤的计算开销优于对比方案.

表 4 初始化阶段的计算开销对比

方案	*TagGen*	*PrfGen*	*PrfVerify*
IBPDP	$3MUL_{Z_q^*}+EXP+a\cdot ADD_{Z_q^*}$	$s\cdot c\cdot b(2MUL_{Z_q^*}+EXP+ADD_{Z_q^*})$	$s\cdot c\cdot bMUL_{Z_q^*}+(1+s\cdot c\cdot b)EXP$ $+3BP+cADD_{Z_q^*}$
MCPDP	$3MUL_{Z_q^*}+EXP+a\cdot ADD_{Z_q^*}$	$(2c+b+s)MUL_{Z_q^*}+(c+b)EXP$ $+(c+b+s)ADD_{Z_q^*}$	$c(1+ab)MUL_{Z_q^*}+(a\cdot b\cdot c)EXP$ $+3BP+(c+b)ADD_{Z_q^*}$
DSAS	$MUL_{Z_q^*}+2EXP$	$k(MUL_{Z_q^*}+ADD_{Z_q^*})$	$[k(c+1)+2]MUL_{Z_q^*}+$ $[k(c+1)+1]EXP+2BP$
Ours	$MUL_{Z_q^*}+ADD_{Z_q^*}$	$(2c+b+1)MUL_{Z_q^*}+(2c+b)MUL_{\mathbb{G}}$ $+(2c+b+1)ADD_{Z_q^*}+(b+2)ADD_{\mathbb{G}}$	$MUL_{\mathbb{G}}+sADD_{Z_q^*}+sADD_{\mathbb{G}}$

6.2 实验仿真

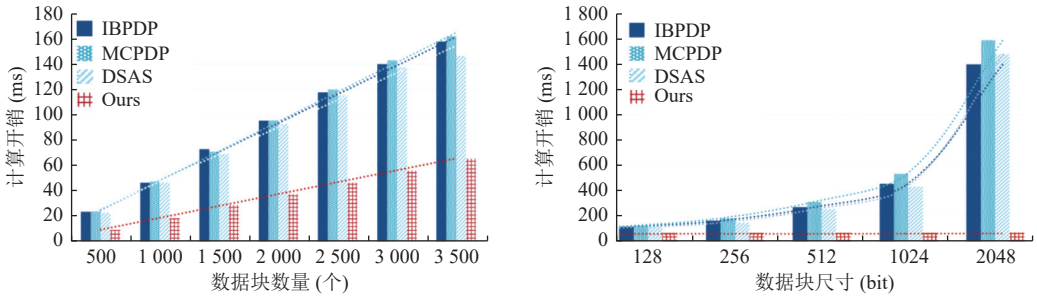
基于 Miracl 库完成本地密码学相关运算, 基于 elliptic-curve-solidity-master 库完成链上密码学相关测试, 基于 Jerasure 库完成数据恢复性能测试. 实验基于 C++, Solidity 编程语言实现, 运行环境为 Linux version 4.15.0-99-generic, Ubuntu 20.04 with 8-GB RAM, Remix IDE. 由于仿真实验在于突出本工作较对比多云审计方案的可行性,

故实验省略了数据在多云环境传输过程中的通信开销, 所有实验在单机环境下运行. Z_q^*, $\mathbb{G}$ 中元素尺寸设置为 160 位, 方案涉及的随机数和私钥为 160 位. 根据文献 [1] 的描述, 当 3 500 个数据块中存在 1% 的受损数据块时, 选取挑战数 $c = 460$ 可使其检测受损数据块的成功率达到 99%; 选取挑战数 $c = 300$ 可使其检测受损数据块的成功率达到 95%. 在本实验中, 评估 $c = 460$ 和 $c = 300$ 情况下的审计开销.

6.2.1 计算开销

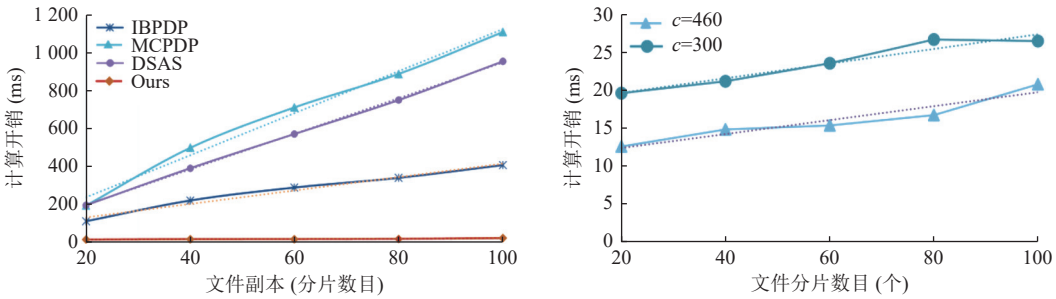
本节设置两组对比实验, 评估标签生成、证明生成、证明验证对应的计算开销. 在标签生成性能评估中, 实验 1 设定数据块尺寸为 256 位, 数据块数量范围为 500–3 500 块. 如图 4(a) 所示, 随着数据块数量增加, 方案 IBPDP, MCPDP, DSAS, KMCA 的标签生成时间呈线性增加, 但 KMCA 计算开销明显低于其他对比方案. 实验 2 设定数据块数目为 3 500, 数据块尺寸范围 128–2 048 位. 如图 4(b) 所示, 随着数据块尺寸的增加, 方案 IBPDP, MCPDP, DSAS 的标签生成时间呈线性变化, KMCA 趋于稳定, 且计算开销低于对比方案.

在证明生成性能评估中, 设定挑战块数 $c = 300$, 数据块尺寸为 256 位, 比较 4 个方案的目标文件片数或副本数从 20 增至 100 的证明生成时间. 如图 5(a) 所示, KMCA 证明生成时间远低于其余方案, 并且受文件片数的影响较小. 然后比较 KMCA 在挑战块数为 300 与 460 时的证明生成时间. 如图 5(b) 所示, 在完整性验证准确率分别为 95% 与 99% 的情况下, KMCA 证明生成开销稳定在 30 ms 内.



(a) 数据块尺寸: 256 位 (b) 数据块数量: 3 500 块

图 4 标签生成时间对比



(a) 证明生成的时间对比 (b) *KMCA* 不同挑战块的证明生成时间对比

图 5 证明生成时间对比

在证明验证性能评估时, 设定挑战块数为 300, 文件片数/副本数为 40. 比较随着 CS 的增多, 证明验证的时间开销变化. 后文图 6 展示了方案 IBPDP, MCPDP, DSAS 的证明验证开销随着 CS 增多而线性增加, 而 KMCA 在证明验证过程受 CS 数量影响较小.

6.2.2 定位效率

本节评估 KMCA 的错误定位效率, 比较了顺序查找、二分查找与 *ImporSearch* 算法的查找效率. 设数据块总数为 n, 共存在 k 个错误. 通过对比计算复杂度来看, 顺序查找算法需要计算 n 次验证方程, 二分查找需要计算

$\log_2 n+\log_2(n-1)+\log_2(n-k+1)$ 次, *ImporSearch* 算法需要计算 $\log_2 n+\log_2\frac{n}{2}+\ldots+\log_2\frac{n}{2^{k-1}}$ 次. n 越大, 二分查找与 *ImporSearch* 算法的效率优势就越明显; k 越大, *ImporSearch* 算法的效率优势越明显; k 接近 $n/2$ 时, 顺序查找效率最高.

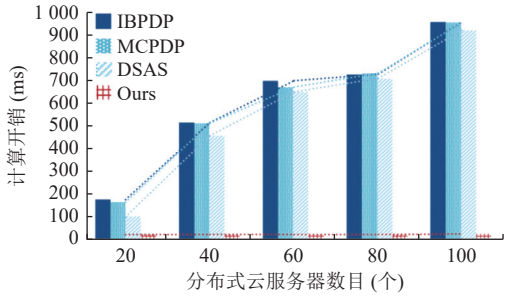
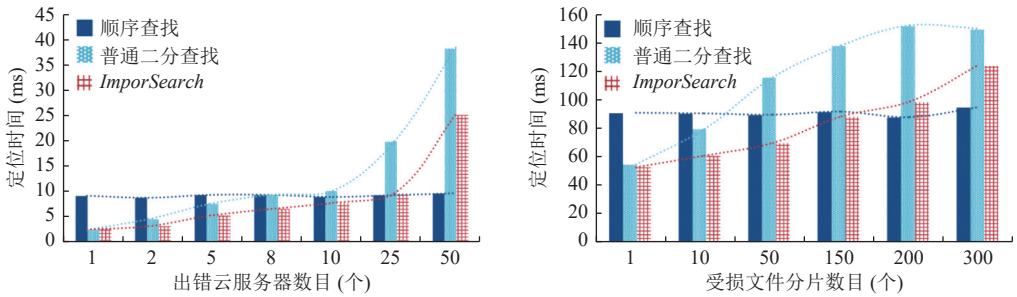


图 6 证明验证的时间对比

在出错 *CS* 定位实验中, 设置 *CS* 总数为 50, 出错 *CS* 的个数分别为 1, 2, 5, 8, 10, 25, 50; 在受损分片定位时, 设定分片总数为 300, 受损分片个数分别是 1, 10, 50, 150, 200, 300. 实验结果如图 7(a)、图 7(b) 所示, 在仅存在一个错误时, 二分查找和 *ImporSearch* 算法效率相当, 并远高于顺序查找. 随着错误个数增多, *ImporSearch* 的查找效率逐渐高于二分查找. 当错误数量超过总数的 1/2 时, *ImporSearch* 开始低于顺序查找. 鉴于实际情况下错误出现概率不超过 5%, 故 *ImporSearch* 具备高效性和可行性.



(a) 不同云服务器数量下的错误定位效率 (b) 不同文件分片总数下的错误定位效率

图 7 在云服务器出错率/文件分片受损率不同时的定位效率对比

6.2.3 数据恢复性能

本节基于 Jerasure 库评估 KMCA 的数据恢复性能. 实验对比了不同参数配置下 RS 纠删码的数据冗余处理效率与数据恢复效率. 充分考虑实际场景中的 *CS* 故障率与 *CS* 存储开销, 以最大限度保障数据可用性[34]. 设冗余存储策略中冗余度 $\eta=0.5$, 故将策略 (m,n,η) 简化为 (m,n). 实验使用一个 1 GB 的数据文件, 基于 GF(2^8) 测试冗余策略 (m,n) 分别为 (8, 12), (12, 18), (16, 24), (20, 30), (24, 36), (30, 45) 时的冗余处理效率与数据恢复效率.

如图 8(a) 所示, 设置文件分片受损率为 10% 时, 冗余处理效率与数据恢复效率随着分片数目增多呈近似线性下降. 整体来看, 冗余处理效率与数据恢复效率都在 100 MB/s 以上, 满足实际应用的可行性要求. 此外, 对比实验通过调整文件分片受损率观察数据恢复效率的变化. 设置文件分片受损率分别为 33.33% 与 16.67%, 即受损文件分片数为容许最大受损数及其一半时 (对应参数 $\eta=0.5$, $\frac{m\eta}{m(1+\eta)}=1/3\approx33.33\%$, $\frac{0.5m\eta}{m(1+\eta)}=1/6\approx16.67\%$). 如图 8(b) 所示, RS 纠删码的数据恢复效率随文件分片受损率增加而下降. 值得注意的是, 即使在受损率较大的情况下, KMCA 的数据恢复速率仍然是可接受的. 综上, 多云冗余存储策略 (如基于 RS 纠删码技术) 可满足分布式存储的性能需求.

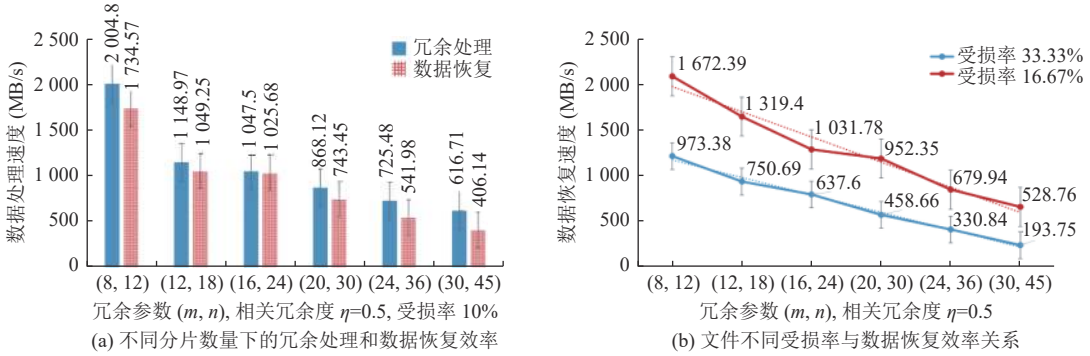


图 8 不同分片和受损率情况下数据恢复效率对比

7 结 论

本文研究了多云存储环境下的多用户数据外包的完整性问题, 并结合智能合约技术实现了支持多重错误定位与数据恢复的多云关键词公开审计方案 KMCA. 关键词多云存储索引表的设计不仅使得索引工作简单高效, 并且方便判断云服务器的拒绝服务攻击. 通过改进二分查找算法, KMCA 可一次性定位多个出错云服务器与受损数据, 并恢复定位出的受损数据. 通过在区块链链上部署审计合约和错误定位合约, 审计结果与定位结果具备公开可信属性. 基于椭圆曲线群上的 Schnorr 签名思想设计的审计方案实现了无双线性对运算, 使方案兼顾安全性与高性能, 既可通过随机预言机模型测试, 也具有优秀的性能开销. 在后续工作中, 考虑在保证安全性的前提下调整数据粒度, 包括外包数据存储粒度和受损数据定位粒度, 以便方案扩展到更多应用场景.

References:

[1] Wang YK. The arrival of digital government and new trends in the development of intelligent government affairs-5G era government information technology foresight. People's Tribune, 2019(11): 33–35. (in Chinese with English abstract). [doi: 10.3969/j.issn.1004-3381.2019.11.011]

[2] Gao X, Yu J, Chang Y, Wang HQ, Fan JX. Checking only when it is necessary: Enabling integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data. IEEE Trans. on Dependable and Secure Computing, 2022, 19(6): 3774–3789. [doi: 10.1109/TDSC.2021.3106780]

[3] Zhu Y, Hu HX, Ahn GJ, Yu MY. Cooperative provable data possession for integrity verification in multicloud storage. IEEE Trans. on Parallel and Distributed Systems, 2012, 23(12): 2231–2244. [doi: 10.1109/TPDS.2012.66]

[4] Wang HQ. Identity-based distributed provable data possession in multicloud storage. IEEE Trans. on Services Computing, 2015, 8(2): 328–340. [doi: 10.1109/TSC.2014.1]

[5] Tian H, Nan FL, Jiang H, Chang CC, Ning JT, Huang YF. Public auditing for shared cloud data with efficient and secure group management. Information Sciences, 2019, 472: 107–125. [doi: 10.1016/j.ins.2018.09.009]

[6] Behrouzi-Far A, Soljanin E. Data replication for reducing computing time in distributed systems with stragglers. In: Proc. of the 2019 IEEE Int'l Conf. on Big Data. Los Angeles: IEEE, 2019. 5986–5988. [doi: 10.1109/BigData47090.2019.9006012]

[7] Li JG, Yan H, Zhang YC. Efficient identity-based provable multi-copy data possession in multi-cloud storage. IEEE Trans. on Cloud Computing, 2022, 10(1): 356–365. [doi: 10.1109/TCC.2019.2929045]

[8] Miao Y, Huang Q, Xiao MY, Susilo W. Blockchain assisted multi-copy provable data possession with faults localization in multi-cloud storage. IEEE Trans. on Information Forensics and Security, 2022, 17: 3663–3676. [doi: 10.1109/TIFS.2022.3211642]

[9] 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: Association for Computing Machinery, 2007. 598–609. [doi: 10.1145/1315245.1315318]

[10] Wang Q, Wang C, Li J, Ren K, Lou WJ. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Proc. of the 14th European Symp. on Research in Computer Security. Saint-Malo: Springer, 2009. 355–370. [doi: 10.1007/978-3-642-04444-1_22]

[11] 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]

[12] Shen J, Shen J, Chen XF, Huang XY, Susilo W. An efficient public auditing protocol with novel dynamic structure for cloud data. IEEE Trans. on Information Forensics and Security, 2017, 12(10): 2402–2415. [doi: 10.1109/TIFS.2017.2705620]

[13] Zhu K, Ren YJ, Zhu QF. A provable data possession protocol in cloud storage systems with fault tolerance. In: Proc. of the 2021 IEEE Conf. on Dependable and Secure Computing. Aizuwakamatsu: IEEE, 2021. 1–6. [doi: 10.1109/DSC49826.2021.9346241]

[14] Ding Y, Li YP, Yang WJ, Zhang K. Edge data integrity verification scheme supporting data dynamics and batch auditing. Journal of Systems Architecture, 2022, 128: 102560. [doi: 10.1016/j.sysarc.2022.102560]

[15] Tian XX, Liu TS, Niu XY, Zhou AY. Lightweight dynamic integrity auditing scheme for cloud data of ubiquitous power internet of things. Chinese Journal of Computers, 2020, 43(12): 2298–2314 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2020.02298]

[16] Widodo RNS, Lim H, Atiquzzaman M. A new content-defined chunking algorithm for data deduplication in cloud storage. Future Generation Computer Systems, 2017, 71: 145–156. [doi: 10.1016/j.future.2017.02.013]

[17] Singh Y, Kandah F, Zhang WY. A secured cost-effective multi-cloud storage in cloud computing. In: Proc. of the 2011 IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS). Shanghai: IEEE, 2011. 619–624. [doi: 10.1109/INFCOMW.2011.5928887]

[18] Pang XQ, Wang TQ, Chen WJ, Ren MQ. Batch provable data possession scheme with error locating. Ruan Jian Xue Bao/Journal of Software, 2019, 30(2): 362–380 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5423.htm [doi: 10.13328/j.cnki.jos.005423]

[19] Zhang C, Xu Y, Hu YP, Wu JJ, Ren J, Zhang YX. A blockchain-based multi-cloud storage data auditing scheme to locate faults. IEEE Trans. on Cloud Computing, 2022, 10(4): 2252–2263. [doi: 10.1109/TCC.2021.3057771]

[20] 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]

[21] Zhu YJ, Yao JG, Guan HB. Blockchain as a service: Next generation of cloud services. Ruan Jian Xue Bao/Journal of Software, 2020, 31(1): 1–19. (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5891.htm [doi: 10.13328/j.cnki.jos.005891]

[22] Xue JT, Xu CX, Zhao JN, Ma JF. Identity-based public auditing for cloud storage systems against malicious auditors via blockchain. Science China Information Sciences, 2019, 62(3): 32104. [doi: 10.1007/s11432-018-9462-0]

[23] Xu Y, Ren J, Zhang Y, Zhang C, Shen B, Zhang YX. Blockchain empowered arbitrable data auditing scheme for network storage as a service. IEEE Trans. on Services Computing, 2020, 13(2): 289–300. [doi: 10.1109/TSC.2019.2953033]

[24] Fan K, Bao ZJ, Liu MX, Vasilakos AV, Shi WB. Dredas: Decentralized, reliable and efficient remote outsourced data auditing scheme with blockchain smart contract for industrial IoT. Future Generation Computer Systems, 2020, 110: 665–674. [doi: 10.1016/j.future.2019.10.014]

[25] Du YF, Duan HY, Zhou AX, Wang C, Au MH, Wang Q. Towards privacy-assured and lightweight on-chain auditing of decentralized storage. In: Proc. of the 40th IEEE Int'l Conf. on Distributed Computing Systems (ICDCS). Singapore: IEEE, 2020. 201–211. [doi: 10.1109/ICDCS47774.2020.00023]

[26] Du YF, Duan HY, Zhou AX, Wang C, Au MH, Wang Q. Enabling secure and efficient decentralized storage auditing with blockchain. IEEE Trans. on Dependable and Secure Computing, 2022, 19(5): 3038–3054. [doi: 10.1109/TDSC.2021.3081826]

[27] 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]

[28] Li T, Yang AJ, Weng J, Guo ZF. Public auditing scheme for industrial Internet data based on smart contracts. Ruan Jian Xue Bao/Journal of Software, 2023, 34(3): 1491–1511 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6716.htm [doi: 10.13328/j.cnki.jos.006716]

[29] Xue JT, Luo SQ, Deng QF, Shi LJ, Zhang XJ, Wang HX. KA: Keyword-based auditing with frequency hiding and retrieval reliability for smart government. Journal of Systems Architecture, 2023, 138: 102856. [doi: 10.1016/j.sysarc.2023.102856]

[30] Schnorr CP. Efficient signature generation by smart cards. Journal of Cryptology, 1991, 4(3): 161–174. [doi: 10.1007/BF00196725]

[31] Peng C, Luo M, Wang HQ, Khan MK, He DB. An efficient privacy-preserving aggregation scheme for multidimensional data in IoT. IEEE Internet of Things Journal, 2022, 9(1): 589–600. [doi: 10.1109/JIOT.2021.3083136]

[32] Gavin W. Ethereum: A secure decentralised generalised transaction ledger. 2014. https://www.win.tue.nl/~mholende/seminar/references/ethereum_yellowpaper.pdf

[33] Yu JD, Lu P, Zhu YM, Xue GT, Li ML. Toward secure multikeyword top-k retrieval over encrypted cloud data. IEEE Trans. on Dependable and Secure Computing, 2013, 10(4): 239–250. [doi: 10.1109/TDSC.2013.9]

[34] Lin WK, Chiu DM, Lee YB. Erasure code replication revisited. In: Proc. of the 4th Int'l Conf. on Peer-to-peer Computing. Zurich: IEEE Computer Society, 2004. 90–97. [doi: 10.1109/PTP.2004.1334935]

[35] Takami G, Sugawara T, Sakiyama K, Li Y. Mixture-based 5-round physical attack against AES: Attack proposal and noise evaluation. IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, 2022, 105.A(3): 289–299. [doi: 10.1587/transfun.2021CIP0016]

附中文参考文献:

[1] 汪玉凯. 数字政府的到来与智慧政务发展新趋势——5G 时代政务信息化前瞻. 人民论坛, 2019(11): 33–35. [doi: 10.3969/j.issn.1004-3381.2019.11.011]

[11] 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]

[15] 田秀霞, 刘天顺, 牛晓宇, 周傲英. 面向泛在电力物联网云端数据的轻型动态完整性审计方案. 计算机学报, 2020, 43(12): 2298–2314. [doi: 10.11897/SP.J.1016.2020.02298]

[18] 庞晓琼, 王田琪, 陈文俊, 任孟琦. 一个支持错误定位的批处理数据拥有性证明方案. 软件学报, 2019, 30(2): 362–380. http://www.jos.org.cn/1000-9825/5423.htm [doi: 10.13328/j.cnki.jos.005423]

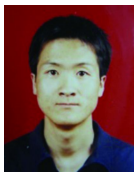
[21] 朱昱锦, 姚建国, 管海兵. 区块链即服务: 下一个云服务前沿. 软件学报, 2020, 31(1): 1–19. http://www.jos.org.cn/1000-9825/5891.htm [doi: 10.13328/j.cnki.jos.005891]

[28] 李涛, 杨安家, 翁健, 郭梓繁. 基于智能合约的工业互联网数据公开审计方案. 软件学报, 2023, 34(3): 1491–1511. http://www.jos.org.cn/1000-9825/6716.htm [doi: 10.13328/j.cnki.jos.006716]

薛婧婷(1990－), 女, 博士, 讲师, 主要研究领域为应用密码学, 云数据安全审计, 区块链技术.

李发根(1979－), 男, 博士, 教授, 博士生导师, 主要研究领域为密码学, 信息安全.

罗抒琴(1999－), 女, 硕士生, 主要研究领域为分布式云存储与审计, 区块链应用.



周宇(1980－), 男, 博士, 研究员, 主要研究领域为序列设计, 编码理论, 密码协议.

张文政(1966－), 男, 研究员, 主要研究领域为密码技术, 网络与信息安全.

张晓均(1985－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为应用密码学, 云计算安全, 智能电网安全隐私.