

支持受损数据定位与恢复的动态群用户可证明存储

姜涛¹, 徐航², 王良民³, 马建峰¹

(1. 西安电子科技大学 ISN 国家重点实验室, 陕西 西安 710071; 2. 西安电子科技大学网络与信息安全学院, 陕西 西安 710126;
3. 东南大学网络空间安全学院, 江苏 南京 211189)

摘 要: 云计算的外包存储模式导致数据拥有者的数据所有权和管理权分离, 进而改变了数据存储网络模型和安全模型。为了有效应对云服务器端的软硬件故障及潜在的不诚实服务提供商, 确保数据拥有者数据的可用性, 设计安全、高效的数据可用性、可恢复性审计方案对于解决用户担忧、保证云数据安全具有重要的理论和实践价值。然而, 现有研究多针对数据完整性或者可恢复性方案的安全性和效率进行设计, 没有考虑动态群用户下受损数据的快速定位和可靠恢复问题。因此, 针对动态群用户环境中受损数据定位与恢复问题, 设计了一个可公开验证的动态群组云用户存储证明方案。该方案在检测到数据受损时, 任何可信的第三方审计者能够通过挑战协议快速定位受损数据, 并在数据受损程度小于纠错能力门限情况下允许云平台对数据进行可靠恢复。该方案结合关联计算和累加计算, 有效减少了受损数据定位的计算次数; 通过纠删码与共享编码技术, 方案能够实现用户受损数据的有效恢复。同时, 方案支持用户的动态撤销, 确保了群用户共享数据在用户撤销后的完整性审计和可靠恢复。定义了方案的网络模型和威胁模型, 并在相应安全模型下证明了所设计方案的安全性。通过真实环境下的原型系统实现和模块化性能分析, 证明了所设计方案能够有效定位受损数据并在数据受损时对云端数据进行可靠恢复。同时, 与相关方案相比, 所设计方案在受损数据定位与恢复方面的计算开销较小。

关键词: 云存储; 数据共享; 损坏识别; 可恢复证明; 动态群用户

中图分类号: TP309

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2022071

Proof of storage with corruption identification and recovery for dynamic group users

JIANG Tao¹, XU Hang², WANG Liangmin³, MA Jianfeng¹

1. State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China

2. School of Cyber Engineering, Xidian University, Xi'an 710126, China

3. School of computer science and engineering, Southeast University, Nanjing 211189, China

收稿日期: 2022-04-11; 修回日期: 2022-07-22

通信作者: 王良民, laingmin@seu.edu.cn

基金项目: 中央高校基本科研业务费专项 (XJS211502)

Foundation Item: Fundamental Research Funds for the Central Universities (XJS211502)

引用格式: 姜涛, 徐航, 王良民, 等. 支持受损数据定位与恢复的动态群用户可证明存储[J]. 网络与信息安全学报, 2022, 8(5): 75-87.

Citation Format: JIANG T, XU H, WANG L M, et al. Proof of storage with corruption identification and recovery for dynamic group users[J]. Chinese Journal of Network and Information Security, 2022, 8(5): 75-87.

Abstract: The outsourced storage mode of cloud computing leads to the separation of data ownership and management rights of data owners, which changes the data storage network model and security model. To effectively deal with the software and hardware failures of the cloud server and the potential dishonest service provider and also ensure the availability of the data owners' data, the design of secure and efficient data availability and recoverability auditing scheme has both theoretical and practical importance in solving the concern of users and ensuring the security of cloud data. However, most of the existing studies were designed for the security and efficiency of data integrity or recoverability schemes, without considering the fast identification and reliable recovery of damaged data under dynamic group users. Thus, to quickly identify and recover damaged data, a publicly verifiable proof of storage scheme was proposed for dynamic group cloud users. The designed scheme enabled a trusted third-party auditor to efficiently identify the damaged files through a challenge-response protocol and allowed the cloud storage server to effectively recover them when the degree of data damage is less than an error correction ability threshold. The scheme combined association calculation and accumulation calculation, which effectively reduced the number of calculations for the identification of damaged data. By combining erasure coding and shared coding technology, the scheme achieved effective recovery of damaged data of dynamic group users. At the same time, the scheme also supported dynamic user revocation, which ensured the integrity audit and reliable recovery of the collective data after user revocation. The network model and threat model of the designed scheme were defined and the security of the scheme under the corresponding security model was proved. Through the prototype implementation of the scheme in the real environment and the modular performance analysis, it is proved that the proposed scheme can effectively identify the damaged data and reliably recover the cloud data when the data is damaged. Besides, compared with other schemes, it is also proved that the proposed scheme has less computational overhead in identifying and recovering damaged data.

Keywords: cloud storage, data sharing, corruption identification, proof of retrievability, dynamic group users

0 引言

随着云计算、大数据等数字技术的快速发展,云存储正在通过各种应用服务走进人们的生活。在云存储应用中,用户可以通过与一组用户共享数据来进一步降低存储成本。然而,这种外包存储模式致使用户失去对数据的物理控制权,存储在无法完全受信的云服务提供商^[1]中的用户数据将面临极大的安全风险。国内外相继爆发多起云存储安全事故,如 Amazon 曾因停电导致部分客户数据永久丢失^[2],某公司硬件故障/软件错误^[3]导致云数据被损毁^[4-5]等,云存储应用仍存在许多安全隐患,如何切实保障云数据的隐私安全性成为学术界亟待解决的研究热点之一。

学者相继提出了许多数据完整性/可恢复性审计方案,如数据可拥有性证明(PDP, provable data possession)^[6]、数据可恢复证明(POR, proof of retrievability)^[7]等。在此基础上设计了多种不同特性的数据完整性审计方案^[8-13]。其中,基于

代理的重签名审计方案^[8]结合可信代理重加密技术以保证多用户共享数据完整性审计中的用户安全撤销;基于共享群密钥的数据审计方案^[13]结合 Schnorr 签名与群组共享密钥以提高动态群用户对共享数据的完整性审计效率。上述方案虽然可以证明外包数据的完整性,但是无法快速识别审计数据中的损坏文件/块并在损坏后恢复。这是因为,当对云服务器上大规模数据的完整性进行审计时,单个受损文件/块可能导致审计失败。数据完整性审计失败之后,识别损坏文件/块带来了大量的计算和通信开销,因而会严重影响数据完整性审计方案的效率。为了解决以上挑战,学者提出了损坏文件识别方案^[14-17]。但是这些方案受限于单用户场景。事实上,在多用户云存储环境下,数据的审计标签私钥一般是不同的,因而在这种应用场景中对身份验证标签进行聚合以及对受损文件/块进行识别存在一定难度且现有方案的实施效率并不理想。目前关于如何实现动态群用户环境下受损数据的识别和恢复这一问题,学术界

尚缺乏行之有效的方案。虽然 POR 可以在一定程度上恢复受损的文件,但 POR 只适用于单个用户,不支持动态群用户环境。当群中某个用户撤离后,该用户将无法管理其原先存储在云服务器中的共享数据,这些数据将由管理员转交其他合法用户进行管理。当合法用户对该撤销用户的数据进行审计时,POR 方案无法恢复属于撤销用户的受损数据。总之,在 POR 中,每个用户的信息是相互独立的,属于撤销用户的数据不能被其他用户审计或者恢复。

针对上述问题,本文提出了一种动态群用户受损数据的快速定位和恢复方案。该方案能够有效地识别来自群用户的所有外包数据的可用性,而无须通过多个证明来检查。在本文方案中,用户只需要对原始文件和相应的标签执行一次处理/上传操作,而群用户的撤销不会影响到该操作;特别是该方案利用快速定位和恢复技术有效解决了数据受损的情况,本文贡献如下。

1) 针对云存储动态群用户数据完整性审计中无法同时支持受损文件的快速定位与可靠恢复问题,提出了一种动态群用户受损数据的定位与恢复方案。

2) 所设计方案结合关联计算、累加计算纠错码和共享编码等技术,在数据完整性设计中减少定位受损数据的计算次数的同时实现了动态群用户环境下受损数据的可靠恢复。

3) 利用双线性映射和数学矩阵的相关性质证明了所设计方案的安全性。同时,通过方案的原型系统实现,并对方案共享数据存储、受损数据定位和受损数据恢复等模块进行评估,证明了方案的可行性。通过与相关方案在整体性能和定位受损数据效率方面进行对比,证明了所设计方案的高效性。

1 相关工作

近年来,学者相继提出了多种云存储数据完整性审计方案。例如,Ateniese 等在文献[6]中首次提出“可证明数据拥有”PDP 的概念,利用 RSA 方案和离散对数构造同态验证标签,在降低文件的检索和下载开销的同时成功保护了文件隐私。在此基础上,Sengupta 等^[9]提出了一种基于同态

哈希的动态云存储方案,该方案不仅支持公开验证数据的完整性,还可以高效地进行数据更新。Bernstein 等^[14]提出了一种基于椭圆曲线签名的审计方案,可以有效降低受损文件的定位计算成本。Shin 等^[15]提出了一种基于指数测试的搜索方案,该方案通过比较指数的验证值来识别受损文件/块的位置。但该方案有很大的局限性,即仅适用于单个文件受损场景,无法一次性定位多个受损文件。为了解决批量受损文件的定位问题,Matt 等^[16]提出了一种基于配对的签名方案,该方案可以在批量审计失败后定位受损文件,同时能够有效识别多个受损签名。Wang 等^[17]提出了一种批量审计方案以支持受损文件的快速查询。该方案可以有效抵抗“受损文件”攻击,在保证数据可用性的同时具备较高的查询效率。Zhou 等^[18]对数据纠错码技术进行了总结,介绍了多种用于数据恢复的编码方法。

数据共享是云存储应用的核心服务之一。通过数据共享服务,用户可以将各自的数据托管于云端并相互共享这些数据,这样可以大大减轻用户本地设备的存储负担。值得注意的是,当群组中某些成员用户被撤离出群组时,其所属数据由群组管理员指派给其他的合法用户进行管理。针对这一问题,近年来,学术界陆续出现一些群用户撤离场景下的数据完整性审计方案。例如,Wu 等^[19]提出了一种支持用户撤销的多关键字密文查询方案,该方案能够抵抗关键字选择攻击;Jiang 等^[11]提出了一种基于群签名的用户撤销共享数据完整性审计方案,但该方案效率较慢;Zhang 等^[13]提出了一种高效的支持用户撤销的数据完整性审计方案,该方案可以有效提高用户撤销的审计效率。然而,这些审计方案大多聚焦于数据动态性或者用户动态性的效率,并未考虑数据审计未通过的应对策略。如果云服务器未能通过数据完整性审计,则意味着存储在其上的用户数据已遭到破坏,此时如果能快速修复受损文件,将大大提升云服务器的存储性能。因此,设计一种能够快速定位和恢复受损文件的方案具有十分重要的意义。

2 模型与目标

本节主要介绍所设计的 CIRG (corruption

identification and recovery for dynamic group users) 方案的系统模型、威胁模型及设计目标。

2.1 系统模型

CIRG 系统模型如图 1 所示, 系统包含 5 个实体: 群用户、群管理员、云服务提供商 (CSP, cloud service provider)、私钥生成器 (PKG, private key generator) 和第三方审计者 (TPA, third party auditor)。其中, 群用户作为数据拥有者, 负责对外包给云服务器的文件进行编码并标记每个文件块以进行完整性审计。方案假设群用户是诚实的, 系统中群用户不会与任何非法人员 (包括被系统撤销的群用户) 共享私人信息; 群管理员负责管理用户的注册 (新用户加入群组时) 或撤销 (老用户退出群组时), 同时参与密钥的生成、群中数据编码算法公布; CSP 存储来自云用户的外包文件, 并为群用户提供数据共享服务。鉴于真实应用场景中 CSP 往往是不可信的, 方案设定系统中的 CSP 需要接受用户/TPA 的检查, 即对存储的数据进行完整性验证; PKG 是一个可信的实体, 它参与系统公共参数和密钥的生成; TPA 负责代表群用户对存储在云服务器上的数据进行完整性审计。当数据的完整性审计失败时 (即相应被检测文件遭受破坏), TPA 能够有效定位该受损文件。TPA 在数据上传阶段无法从用户的盲性数据中获得真实数据, 在审计阶段也无法从云服务器的数据拥有性证明中获得真实数据信息。

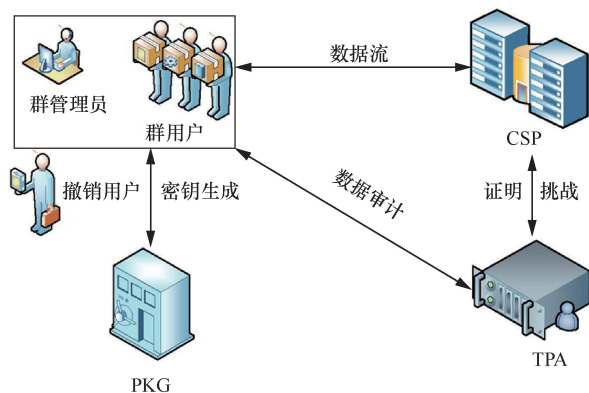


图 1 CIRG 系统模型
Figure 1 CIRG system model

2.2 威胁模型

本文设计的 CIRG 方案不仅支持动态群用户

下的数据完整性安全审计, 而且进一步考虑了数据完整性审计失败后的补救问题。也就是说, 一旦针对某数据的完整性审计失败, TPA 能够有效定位该受损文件的位置。在此之后, TPA 将受损文件的位置列表返回给用户 (或 CSP), 进而可以根据这些位置恢复相应的受损文件。CIRG 方案的威胁模型概括为以下几点。

1) 文件受损: 群用户外包的文件可能由于服务器硬件故障、服务器恶意操作、人为错误操作等因素而遭到损坏或删除。然而, 出于自身声誉及利益的考虑, 云服务商不会主动报告用户数据已被损坏。

2) 文件不可恢复性: 当 TPA 验证后判定 CSP 中存储的某文件已被损坏时, 如果没有特定的文件恢复机制, 系统中群用户将无法恢复并共享文件。

2.3 设计目标

为了在实现云存储共享数据的安全审计的同时支持针对受损文件的高效定位和恢复功能, 本文设计的 CIRG 方案应满足以下目标。

1) 安全目标: 主要考虑方案的正确性和可靠性。正确性意味着如果外包文件完好无损, CSP 总能通过数据完整性审计。当文件块受损总量低于特定阈值时, 任何合法群用户或 CSP 最终一定可以恢复这些受损数据块。可靠性意味着对于任一群用户的文件, 如果 CSP 中, 实际并没有存储该文件; 只存储该文件中较少部分的内容; 存储的相应文件有误, 那么从概率上可以认为 CSP 无法通过数据完整性审计。如果数据拥有性证明本身含有受损文件, TPA 无法定位被破坏文件位置的可能性可以忽略不计。

2) 效率目标: 主要考虑方案的计算、通信和存储效率等性能指标。本文设计的 CIRG 方案兼顾了文件恢复和公共审计功能, 前者是指当存储在云服务器中的数据由于物理磁盘损坏、故障等而受损或丢失时, 方案通过计算受损数据的有效块数可以确保用户能够有效地恢复相应受损文件; 后者意味着 TPA 可以对群用户存储在云服务器中的文件进行完整性审计, 这一过程仅需该文件的少量数据块协助计算即可实现。

3 预备知识和定义

本节主要介绍方案采用的密码工具、数据编解码函数和关联二分 (RDI, relational dichotomy) 计算的定义。

3.1 预备知识

双线性对：假设 $e: G_1 \times G_1 \rightarrow G_2$ 为一个双线性映射，其中 G_1 是一个 GDH (gap Diffie-Hellman) 群， G_2 是一个素数阶为 p 的循环乘法群。双线性对具有以下性质。

1) 双线性：对于所有的 $P, Q \in G_1$ 和 $a, b \in \mathbb{Z}_p$ ， $e(P^a, Q^b) = e(P, Q)^{ab}$ 。

2) 非退化性： $e(P, Q) \neq 1$ ，其中 P, Q 是 G_1 的生成元。

3) 可计算性：存在一个高效算法，该算法可以计算出映射函数 e 。

哈希函数：本文用到如下哈希函数。

1) $H_1: \{0, 1\}^* \times G_1 \rightarrow \mathbb{Z}_q^*$ 是以一个任意长度的字符串与群 G_1 中的一个字符映射到 $\mathbb{Z}_q^*$ 中的一个元素。

2) $H_2: \{0, 1\}^* \times G_1 \times \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ 是以两个任意长度的字符串与 G_1 中的一个字符映射到 $\mathbb{Z}_q^*$ 中的一个元素。

3) $h: \{0, 1\}^* \rightarrow G_1$ 是以一个任意长度的字符串映射到 G_1 中的一个元素。

编解码函数：本文用到的编解码函数定义如下。

1) 编码函数： $\text{Encod}(m_1, m_2, \dots, m_k) \rightarrow (\omega_1, \omega_2, \dots, \omega_n)$ 。该函数以初始数据 $F = (m_1, m_2, \dots, m_k)$ 作为输入，返回编码后的数据 $E = (\omega_1, \omega_2, \dots, \omega_n)$ ，其中 $m_i, \omega_j \in \mathbb{Z}_q^*$ ， $i \in [1, k]$ ， $j \in [1, n]$ ， $n = \text{red} + k$ ，red 为冗余量。

2) 解码函数： $\text{Decod}(\omega_1', \omega_2', \dots, \omega_k') \rightarrow (m_1, m_2, \dots, m_k)$ 。从受损数据中选取任意 k 个有效块 $E' = (\omega_1', \omega_2', \dots, \omega_k')$ 作为输入，返回初始数据 $F = (m_1, m_2, \dots, m_k)$ ，其中 $m_i, \omega_i' \in \mathbb{Z}_q^*$ ， $i \in [1, k]$ 。

RDI: RDI 将判断等式值是否相等 “ $e_L = e_R$ ” 转化为判断值是否为 0，即 “ $(e_L - e_R) = 0$ ”。

3.2 定义

本文目标是设计 CIRG 存储证明方案，该方

案能够有效识别受损文件/块并支持动态群用户恢复相应受损数据。CIRG 方案定义的算法如下。

1) Setup: 该算法为初始化算法，用于设置系统的公开参数。该算法由 PKG 运行，以安全参数 κ 作为输入，输出私钥 sk、公钥 pk 和部分公共参数 tuple。

2) KeyGen: 该算法用于生成数据标签的密钥。由群用户、群管理员和 PKG 交互运行并生成数据标签的相关密钥。

3) Encod: 该算法为编码算法，用于对群用户数据进行编码。它以初始数据 F 作为输入，输出编码后的数据 E 。

4) TagGen: 该算法用于生成数据标签。由群用户运行，以 E 作为输入，生成相应的数据标签。

5) Challenge: 该算法由 TPA 运行，用于生成审计挑战。其中，TPA 根据用户审计请求生成文件的挑战信息 chal 并将其发送给云服务器。

6) Prove: 该算法用于生成数据拥有性证明。具体而言，由云服务器运行该算法，以挑战信息 chal 作为输入，输出相应的文件存储证明并将其返回给 TPA。

7) Recover: 该算法首先对数据拥有性证明进行验证，如果验证通过，则说明云服务器完整存储了用户数据。一旦验证不通过，则说明用户存储的数据受损，这时 TPA 将快速定位该受损数据的位置，并将其返回给用户，最后由用户对受损数据进行恢复。

4 CIRG 方案具体构造

现有的云存储审计方案^[13,20-23]大多聚焦于用户数据完整性审计的效率提升问题。本文则主要侧重于解决群用户数据审计失败后针对相应受损数据的定位与恢复问题。此外，为了保障方案的整体完善性、效率 and 安全性，本文选用 Schnorr 签名^[13]生成用户数据的标签密钥，这样可以保证方案具备动态群用户的高效审计功能。本文方案沿用文献[13]的 ssk 和群身份 (UI, group identity) 作为文件标签的公私钥对。同时，方案采用基于 UI 的数字签名 τ 来确保文件标识符名称、用户撤销编号和验证值的完整性。其中，用户撤销数量 (RN, revocation number) 是支持用户动态操作的

关键。CIRG 方案中各算法的具体描述如下。

1) Setup 算法执行阶段: PKG 生成密钥和系统公开参数。

① PKG 选择主密钥 $\gamma \in Z_q^*$ 和部分私钥 $\beta \in Z_q^*$ 。PKG 计算两个公开参数 $Y_1 = g^\gamma$ 和 $Y_2 = g^\beta$ 。PKG 持有主私钥 γ 并将部分私钥 β 发送给群管理员以生成部分密钥。

② PKG 任意选取并发布 tuple 作为系统的公共参数, 其中, e 是由 PKG 任意选取的双线性映射函数, 并且 $e: G_1 \times G_1 \rightarrow G_2$ 。PKG 从 G_1 中随机选取 g 和 u , q 是群的素数阶, $\text{tuple} = \{G_1, G_2, e, q, g, u, Y_1, Y_2, H_1, H_2, h\}$ 。

2) KeyGen 算法执行阶段: 利用 Schnorr 签名协议生成数据标签密钥。如图 2 所示, 群用户和群管理员、PKG 交互生成数据标签密钥, 其中, 文件标识符标签密钥由用户进行计算。本文将所有合法用户的数据标签私钥关联起来, 群用户共用同一数据标签密钥。在群组中如果有群用户离开, 系统将及时更新群中成员的数据标签私钥。用户一旦撤离群组, 将无法再上传数据到云服务器上进行存储, 同时无法享用云服务器中存储的共享服务。此外, 该用户曾经存储在云服务器上的数据将由群管理员转交其他合法的群用户进行管理。值得注意的是, 该用户所属的数据标签在其撤离群后仍然可以被其他合法群用户使用, 即系统无须重新计算标签。这种可更新共享密钥的方式可以极大地节省计算和存储资源, 同时提高审计效率。密钥生成的具体细节如下。

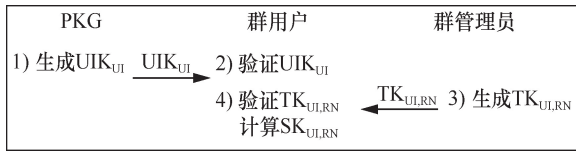


图 2 数据标签私钥生成

Figure 2 Data tag private key generation

① PKG 运行 $\{UI, r_{UI}\} \rightarrow \{UIK_{UI}\}$ 。当 PKG 接收到群的身份 UI 后, 任意选择随机数 $r_{UI} \in Z_q^*$ 并且计算 $R_{UI} = g^{r_{UI}}$ 和 $\sigma_{UI} = r_{UI} + \gamma H_1(UI, R_{UI}) \mod q$ 。然后 PKG 发送身份密钥 $UIK_{UI} = (R_{UI}, \sigma_{UI})$ 给群用户。

② 群用户运行 $\{UIK_{UI}\} \rightarrow \{0, 1\}$ 。在接收到身份密钥 UIK_{UI} 之后, 群用户验证 UIK_{UI} 的正确性。如果 $g^{\sigma_{UI}} = R_{UI} \cdot Y_1^{H_1(UI, R_{UI})}$, 那么群用户接受此身份密钥, 否则拒绝。

③ 群管理员运行 $\{RN, r_{RN}, UI\} \rightarrow \{TK_{UI,RN}\}$ 。初始时默认群组中没有用户离开, 群管理员设置用户撤销数为 $RN = 0$, 并将其发送给群用户和云服务器。之后, 群管理员选择随机数 $r_{RN} \in Z_q^*$, 并计算 $R_{RN} = g^{r_{RN}}$ 和 $\sigma_{RN} = r_{RN} + \beta H_2(UI, R_{RN}, RN) \mod q$ 。接下来, 群管理员发送部分密钥 $TK_{UI,RN} = (R_{RN}, \sigma_{RN})$ 给群用户, 其中, RN 初始设置为 0, 当某个用户离开群组时, 用户撤销数改变为 $RN = RN + 1$ 。与此同时, 部分密钥 $TK_{UI,RN}$ 随着用户离开群组后 RN 的增加而更新。最后, 群管理员选择一组线性无关向量 $vm_1, vm_2, \dots, vm_k$, 并将其在群组公开。

④ 群用户运行 $\{TK_{UI,RN}\} \rightarrow \{0, 1\}$ 。在接收到部分密钥 $TK_{UI,RN}$ 之后, 群用户验证其正确性。如果 $g^{\sigma_{RN}} = R_{RN} \cdot Y_2^{H_2(UI, R_{RN}, RN)}$, 那么群用户接受部分密钥 $TK_{UI,RN}$, 否则拒绝。

⑤ 群用户运行 $\{UIK_{UI}, TK_{UI,RN}\} \rightarrow \{SK_{UI,RN}\}$ 。群用户基于身份密钥 $UIK_{UI} = (R_{UI}, \sigma_{UI})$ 和部分密钥 $TK_{UI,RN} = (R_{RN}, \sigma_{RN})$ (设置初始 $RN = 0$) 计算私钥。也就是说, 群用户计算 $\sigma = (\sigma_{UI} + \sigma_{RN}) \mod q$, 得到数据标签私钥 $SK_{UI,RN}$ 为 $(RN_{UI}, RN_{RN}, \sigma)$ 。

综上所述, 系统利用 Schnorr 签名协议与群组共享密钥生成数据标签密钥, 其中 σ 用于生成云存储数据块的签名。为了保证文件本身的完整性与正确性, 本文对文件标识符、RN 以及相关参数进行签名。该签名是基于 UI 的数字签名 τ , 选取 UI 和 ssk 作为公私钥对, 私钥 ssk 用于生成数字签名 τ , 公钥 UI 用来验证签名 τ 的正确性。

3) Encod 算法执行阶段: 如图 3 所示, 利用 RS 纠错码对用户存储的共享数据 $F = (m_1, m_2, \dots, m_k)$ 进行编码。首先, 用户将群管理员设置的并在群组公开的一组线性无关向量

$vm_1, vm_2, \dots, vm_k$ 组成数学矩阵 A (如 $A = \begin{bmatrix} EI \\ V \end{bmatrix}$),

其中 $\mathbf{EI}$ 为单位矩阵, $\mathbf{V}$ 为 Vandermonde 矩阵)。

$$\mathbf{A} = \begin{bmatrix} vm_{1,1} & vm_{1,2} & \cdots & vm_{1,k} \\ vm_{k,1} & vm_{k,2} & \cdots & vm_{k,k} \\ vm_{k+1,1} & vm_{k+1,2} & \cdots & vm_{k+1,k} \\ \vdots & \vdots & \ddots & \vdots \\ vm_{n,1} & vm_{n,2} & \cdots & vm_{n,k} \end{bmatrix}$$

然后对文件 $\mathbf{F}$ 进行编码。

$$\mathbf{E}^T = \mathbf{A} \cdot \mathbf{F}^T = \mathbf{A} \cdot \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_k \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_k \\ \vdots \\ \omega_n \end{bmatrix}$$

其中 $\omega_i = \sum_{t=1}^{t=k} m_t vm_{i,t}$, $i = 1, 2, \dots, n$, $n = \text{red} + k$, red

是 RS 纠删码的冗余量。通过计算得到编码后的文件 $\mathbf{E}$ 。 $\mathbf{A}$ 是 k 个长度为 n 且线性无关的向量组成的矩阵。 vm_i 属于有限域 $\text{GF}(2^w)$, w 必须足够大使得 $n \leq 2^w + 1$ 。

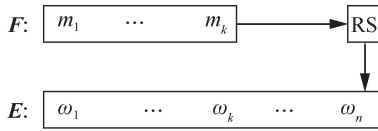


图3 数据编码
Figure 3 Data encoding

4) TagGen 算法执行阶段: 根据数据编码阶段得到的文件 $\mathbf{E}$ 对其进行签名。以密钥 σ 和文件 $\mathbf{E}$ 作为输入, 得到数据块的标签。

$$t_{ij} = \left(h(\text{FID} \parallel i_j \parallel \text{RN}) \cdot u^{\omega_{ij}} \right)^{\sigma} \quad (1)$$

其中, $i \in [1, N]$, $j \in [1, n]$, FID 是文件标识符, i 是第 i 个用户, j 是数据块的索引, u 是选取的任意值且 $u \in G_1$, RN 是用户撤销的数量, h 为密码学的哈希函数。随后, 计算文件的标签: $\text{Ftag} = \text{FID} \parallel \text{RN} \parallel R_{\text{UI}} \parallel R_{\text{RN}} \parallel \tau$, 通过文件标签的私钥 ssk 计算 $\tau = \text{SSig}_{\text{ssk}}(\text{FID} \parallel \text{RN} \parallel R_{\text{UI}} \parallel R_{\text{RN}})$ 。

5) Challenge 算法执行阶段: 用户授权由 TPA 对其存储在云服务器上的数据进行完整性审计。

在该算法中, TPA 为了验证用户数据的完整性, 发起审计挑战。本文假设 TPA 一次性接收多个用户的审计请求, 并且可以为多个用户发起审计挑战。具体而言, TPA 生成的审计挑战如下。

① 任意选取一个共有 $N \cdot s$ 个元素的 $I = \{1, \dots, 1_s, \dots, N_1, \dots, N_s\}$, $s \in [1, n]$, N 表示审计的用户数量。

② 选取 I 个对应的随机值 $v_{i_s} \in Z_q^*$, 其中 $i_s \in I$, $i \in [1, N]$ 。

③ TPA 向云服务器发送审计挑战 $\text{chal} = \{C_1, C_2, \dots, C_N\}$, 其中 $C_i = \{i_j, v_{i_j}\}$, $i \in [1, N]$ 。

6) Prove 算法执行阶段: CSP 根据 TPA 发起的挑战 chal 生成数据拥有性证明, 以此表明正确存储了用户完整的数据。之后, CSP 将该数据拥有性证明发送给 TPA 进行验证。数据拥有性证明计算如下。

① 计算 $T = \{T_1, T_2, \dots, T_N\}$, $\mu = \{\mu_1, \mu_2, \dots, \mu_N\}$

其中 $T_i = \prod_{j=1}^s t_{ij}^{v_{ij}}$, $\mu_i = \sum_{j=1}^s v_{ij} \cdot \omega_{ij}$, $i \in [1, N]$ 。

② 设置数据拥有性证明 P 和对应的用户文件标签 Ftag , 其中 $P = (T, \mu)$ 。

③ 将 P 和文件标签 Ftag 一起返回给 TPA。

7) Recover 算法执行阶段: TPA 通过数据拥有性证明验证数据的完整性。这时可能出现两种情况: 数据的完整性验证通过/不通过。完整性验证不通过, 表明用户存储在 CSP 的文件受损, 意味着用户的数据不可用。针对这种情况, 本文设计的 CIRG 方案可以快速定位该受损文件并将其恢复, 具体内容如下。

① 数据完整性验证。TPA 首先通过 UI 检查 $\tau = \text{SSig}_{\text{ssk}}(\text{FID} \parallel \text{RN} \parallel R_{\text{UI}} \parallel R_{\text{RN}})$ 来判断文件标签的正确性。UI 和 ssk 是文件标签的一对公私钥, 方案通过 UI 验证 τ 中的 FID , RN , R_{UI} , R_{RN} 与文件标签 Ftag 中的 FID , RN , R_{UI} , R_{RN} 是否一致。如图 4 所示, 如果一致, TPA 将接收 CSP 返回的数据拥有性证明 P 并验证其正确性。由于数据块标签密钥随着 RN 的改变而更新, 当文件标签的 RN 相等时, 文件数据块的密钥是相同的。因此, 当文件的 RN 相等时, 可以批量审计这些文件的完整性, 审计公式如下。

$$\prod_{i=1}^N e(T_i, g) = \prod_{i=1}^N e\left(\prod_{j=1}^s h(\text{FID} \| i_j \| \text{RN})^{v_{ij}} \cdot u^{\mu_i}, v\right) \quad (2)$$

其中, $v = R_{\text{UI}} \cdot R_{\text{RN}} \cdot Y_1^{H_1(\text{UI}, R_{\text{UI}})} \cdot Y_2^{H_2(\text{UI}, R_{\text{RN}}, \text{RN})}$ 。系统需要判断等式是否成立, 如果等式成立, 则返回 1, 表示用户的数据是完整存储在 CSP 中; 否则返回 0, 表示 CSP 中至少存在一个受损数据, 这将导致用户文件不可用。

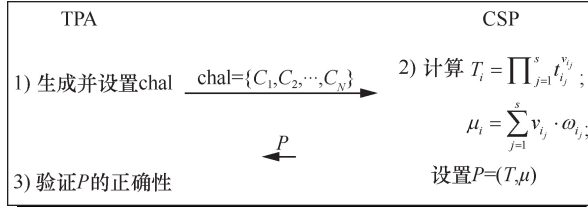


图 4 数据完整性审计
Figure 4 Data integrity audit

为了保证用户数据的完整性和可用性, 必须恢复受损数据。在此之前首先面临的是如何在审计的所有数据中找到受损数据的位置。本文采用 RDI 法查询受损数据的位置, 并通过 RS 纠删码恢复受损数据。

② 定位受损数据。根据 RDI 累加计算来定位受损数据的位置。以数据拥有性证明作为输入, TPA 定位所有的受损数据并输出相应的列表。通过 RDI 将数据完整性审计结果 “ $e_L = e_R$ ” 转化为 $\text{RDI}(\cdot) \rightarrow (e_L - e_R) = 0$ 。也就是说, RDI 将审计结果由判断等式相等转换成判断值是否为 0: $\text{INF}_{\text{RDI}} = e_L - e_R$ 。具体而言, 将上述参数代入计算如下。

$$\text{INF}_{\text{RDI}} = e(T_i, g) - e(\theta_i, v) \quad (3)$$

其中, $\theta_i = \prod_{j=1}^s h(\text{FID} \| i_j \| \text{RN})^{v_{ij}} \cdot u^{\mu_i}$, v 与式(2)

的值相同。根据式(3), 通过 RDI 可以对数据完整性审计公式进行转换, 但这只是对单个受损文件的定位。为了支持多个文件的受损数据定位, 需要进一步将数据完整性的审计结果 e_L 和 e_R 通过移动位置的方式进行转换。之后通过累加的方式将多个用户文件 (即批量文件) 的审计结果关联起来。

如图 5 所示, 二叉搜索树中父节点的验证值等于左右子节点的验证值之和。文件审计集合

$X_{[1,N]}$ 表示 N 个用户的文件。该审计公式根据值为 0 来确定受损文件的位置。值为 0 的判断方法可以建立父节点与左右子节点之间的相关性, 这种方法借助其中审计结果的中间值来减少定位受损数据的计算次数。对于 N 个用户的文件, RDI 批量审计的具体计算公式为

$$\text{INF}_{\text{RDI}}^{[1,N]} = \sum_{i=1}^N e(T_i, g) - \sum_{i=1}^N e(\theta_i, v) \quad (4)$$

其中, θ_i 和 v 与式(3)所表示的计算值相同。

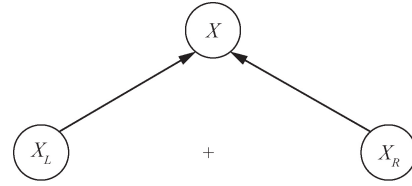


图 5 关联二分计算
Figure 5 RDI computing

通过 RDI 将左右子树的值关联起来, 由左/右子树的值推导出另一个子树的值。其中, 通过计算父节点和左子节点验证值, 能够直接确定右子节点的值, 具体计算公式为

$$\text{INF}_{\text{RDI}}^{[N/2+1,N]} = \text{INF}_{\text{RDI}}^{[1,N]} - \text{INF}_{\text{RDI}}^{[1,N/2]} \quad (5)$$

综上所述, 利用 RDI 和数学累加计算不仅可以定位受损数据的位置, 同时减少了计算次数并提高了查询效率。具体算法如下: 以文件审计结果作为输入, 通过关联二分法的循环计算, 输出一个 N 位字符串 $(b_1, b_2, \dots, b_N)$ 。其中以 b_i 的值表示第 i 个文件是否损坏, 并且 b_i 的值为 0 时表示文件已损坏。当第 i 个文件有效时, $b_i = 1$ 。定位步骤如下。

批验证: 计算 $\text{INF}_{\text{RDI}}^{[1,N]}$, 如果 $\text{INF}_{\text{RDI}}^{[1,N]} = 0$, 输出 N 位字符串 $(1, 1, \dots, 1)$ 并且退出。

根值计算: 设置 $X_{[1,N]}$ 是根节点或者左子节点, 计算 $\text{INF}_{\text{RDI}}^{[1,N]}$ 。

关联二分: 如果 $X_{[1,N]}$ 节点是无效的, 则 $\text{INF}_{\text{RDI}}^{[1,N]} \neq 0$, 并且对 $\text{INF}_{\text{RDI}}^{[1,N]}$ 进行受损文件查询。

RDI 关联查询: 如果 $\text{INF}_{\text{RDI}}^{[1,N/2]} = 0$, 则 $\text{INF}_{\text{RDI}}^{[N/2+1,N]} = \text{INF}_{\text{RDI}}^{[1,N]}$; 如果 $\text{INF}_{\text{RDI}}^{[1,N/2]} = \text{INF}_{\text{RDI}}^{[1,N]}$, 那么 $\text{INF}_{\text{RDI}}^{[N/2+1,N]} = 0$; 否则计算 $\text{INF}_{\text{RDI}}^{[N/2+1,N]} = \text{INF}_{\text{RDI}}^{[1,N]} - \text{INF}_{\text{RDI}}^{[1,N/2]}$ 。

循环输出：循环对左子树和右子树运行相同的算法，最后定位所有受损文件并输出 $(b_1, b_2, \dots, b_N)$ 。

TPA 将受损数据的位置列表返回给用户，以备用户进行受损数据的有效恢复。

③ 受损数据恢复。本文设计的方案可以恢复群用户的受损数据。 E_1 和 F_1 分别为第一个用户的受损数据和初始数据。方案首先从 E_1 中选取 k 个有效的数据块 $E_1' = (\omega_{1,1}', \omega_{1,2}', \dots, \omega_{1,k}')$ 。随后，根据数据编码时所用的线性无关向量组成的矩阵 A ，得到 E_1' 对应的线性无关向量组成的子矩阵 B ，如图 6 所示。最后通过计算 $B^{-1} \times (E_1')^T = F_1^T$ 恢复初始数据，其中 B^{-1} 是对应子矩阵 B 的逆。具体而言，恢复初始数据 F_1 计算步骤如下。

$$A = \begin{bmatrix} vm_{1,1} & vm_{1,2} & \cdots & vm_{1,k} \\ \vdots & \vdots & \ddots & \vdots \\ vm_{k,1} & vm_{k,2} & \cdots & vm_{k,k} \\ vm_{k+1,1} & vm_{k+1,2} & \cdots & vm_{k+1,k} \\ \vdots & \vdots & \ddots & \vdots \\ vm_{n,1} & vm_{n,2} & \cdots & vm_{n,k} \end{bmatrix} \begin{matrix} \leftarrow \omega_{1,1} \\ \leftarrow \omega_{k,1} \\ \leftarrow \omega_{k+1,1} \\ \leftarrow \vdots \\ \leftarrow \omega_{n,1} \end{matrix}$$

图 6 矩阵与编码数据
Figure 6 Matrix and encoded data

选取恢复向量：首先对从 CSP 下载的受损数据中的有效数据块进行编号，之后借助编码阶段的矩阵，将有效编号与矩阵所对应的向量摘取出来并组成一个 $k \times k$ 的子矩阵 $B = (b_{ij})_{1 \leq i, j \leq k}$ 。

$$B = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,k} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{k,1} & b_{k,2} & \cdots & b_{k,k} \end{bmatrix}_{k \times k}$$

矩阵逆计算：对 $k \times k$ 的子矩阵 B 求逆。子矩阵 B 的逆为 $B^{-1} = (b'_{ij})_{1 \leq i, j \leq k}$ 。

$$B^{-1} = \begin{bmatrix} b'_{1,1} & b'_{1,2} & \cdots & b'_{1,k} \\ b'_{2,1} & b'_{2,2} & \cdots & b'_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ b'_{k,1} & b'_{k,2} & \cdots & b'_{k,k} \end{bmatrix}_{k \times k}$$

恢复计算：根据公式 $B^{-1} \times (E_1')^T = F_1^T$ 计算并恢复初始数据 F_1 。

$$F_1^T = \begin{bmatrix} b'_{1,1} & b'_{1,2} & \cdots & b'_{1,k} \\ b'_{2,1} & b'_{2,2} & \cdots & b'_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ b'_{k,1} & b'_{k,2} & \cdots & b'_{k,k} \end{bmatrix} \times \begin{bmatrix} \omega_{1,1}' \\ \omega_{1,2}' \\ \vdots \\ \omega_{1,k}' \end{bmatrix} = \begin{bmatrix} m_{1,1} \\ m_{1,2} \\ \vdots \\ m_{1,k} \end{bmatrix}$$

当存在多个受损数据时，综合利用上述方法可批量恢复受损数据。即使受损数据原本属于某个已撤群用户，由于本文方案的群用户均采用群管理员公开的编码算法对数据进行编码，被系统授权接管该数据的群合法用户仍然可以通过编码算法恢复曾属于撤群用户的共享数据。

5 安全性分析

(1) 正确性

定理 1 如果群用户、群管理员、CSP 和 TPA 都正确执行相应计算，那么 CIRG 方案的批量审计是正确的。

证明 为简便起见，假设需要对 A、B、C 这 3 个用户的数据发起完整性审计，每个用户存储的文件个数为 1，同时设置这 3 个用户的 RN 相等，相应审计公式如下。

$$\begin{aligned} \text{left} &= \prod_{i=1}^3 e(T_i, g) = \\ &= \prod_{i=1}^3 e\left(\prod_{j=1}^s (h(\text{FID} \| i_j \| \text{RN}) \cdot u^{\omega_{ij}})^{\sigma \cdot v_{ij}}, g\right) = \\ &= \prod_{i=1}^3 e\left(\prod_{j=1}^s h(\text{FID} \| i_j \| \text{RN})^{v_{ij}} \cdot u^{\mu_i}, g^{\sigma}\right) = \\ &= \prod_{i=1}^3 e\left(\prod_{j=1}^s h(\text{FID} \| i_j \| \text{RN})^{v_{ij}} \cdot u^{\mu_i}, g^{\sigma_{\text{UI}} + \sigma_{\text{RN}}}\right) = \\ &= \prod_{i=1}^3 e\left(\prod_{j=1}^s h(\text{FID} \| i_j \| \text{RN})^{v_{ij}} \cdot u^{\mu_i}, v\right) \\ \text{right} &= \prod_{i=1}^3 e\left(\prod_{j=1}^s h(\text{FID} \| i_j \| \text{RN})^{v_{ij}} \cdot u^{\mu_i}, v\right) \end{aligned}$$

其中， $g^{\sigma_{\text{UI}}} = R_{\text{UI}} \cdot Y_1^{H_1(\text{UI}, R_{\text{UI}})}$ ， $g^{\sigma_{\text{RN}}} = R_{\text{RN}} \cdot Y_2^{H_2(\text{UI}, R_{\text{RN}}, \text{RN})}$ ， $v = R_{\text{UI}} \cdot R_{\text{RN}} \cdot Y_1^{H_1(\text{UI}, R_{\text{UI}})} \cdot Y_2^{H_2(\text{UI}, R_{\text{RN}}, \text{RN})}$ 。综合以上验证结果可知 $\text{left} = \text{right}$ ，也就是说，CIRG 方案对于批量文件的完整性审计具有正确性。如果在证明一开始假设只对一个用户（如 A）发起审计，最终结果依然为 $\text{left} = \text{right}$ ，那么 CIRG 方案对于单一文件的完整性审计同样是正确的。

(2) 受损数据的可恢复性

定理 2 当群用户存储在 CSP 的数据受损时, CIRG 方案能够有效恢复该受损数据。

证明 首先, 令 dd 代表某个受损文件的受损块数, red 代表数据编码后的冗余量。当且仅当 $dd \leq red$ 时, 受损数据能够被恢复。根据 k 个未损坏数据块 $E' = (\omega'_1, \omega'_2, \dots, \omega'_k)$, 计算出 A 矩阵的子矩阵 B 。随后, 根据公式 $B^{-1} \times (E')^T = F^T$, 即可恢复受损数据。其中, E' 是从文件中得到的 k 个有效数据块, B 是 E' 对应的线性无关向量所组成的矩阵, $(vm_{i,1}, vm_{i,2}, \dots, vm_{i,k}) \rightleftharpoons \omega_i, i \in [1, n]$ 。当 $dd > red$ 时, 有效数据 $E'_{k' \times 1}$ 的块数小于 k , 进一步计算得到行数小于 k 的子矩阵 $B_{k' \times k}$ 。显而易见, $B_{k' \times k}$ 的列数大于 $E'_{k' \times 1}$ 的行数 ($k' < k$), 这有悖于矩阵运算法则。也就是说, 受损数据将无法恢复。

(3) 密钥更新特性

定理 3 当群组中某个用户撤群时, 如果群用户、群管理员都正确执行相应计算, 群组中数据标签私钥将会被正确更新。

证明 在群组中某个用户撤群之后, 群组中用户撤销数将增加为 $RN = RN + 1$ 。群组中根据 Schnorr 签名协议生成的群组共享密钥 $SK_{UI, RN}$ 将会更新。首先, 群管理员任意选取 $r_{RN} \in Z_q^*$, 并更新计算 $R_{RN} = g^{r_{RN}}$ 和 $\sigma_{RN} = r_{RN} + \beta H_2(UI, R_{RN}, RN) \bmod q$, 将更新后的 $TK_{UI, RN} = (R_{RN}, \sigma_{RN})$ 发送给合法群用户, 之后由合法群用户验证 $TK_{UI, RN}$ 的正确性。如果 $g^{\sigma_{RN}} = R_{RN} \cdot Y_2^{H_2(UI, R_{RN}, RN)}$, 那么群用户接受更新后的 $TK_{UI, RN}$, 否则群用户拒绝该值。最后, 群用户根据 $UIK_{UI} = (R_{UI}, \sigma_{UI})$ 和更新后的 $TK_{UI, RN}$ 计算 $\sigma = (\sigma_{UI} + \sigma_{RN}) \bmod q$, 得到更新后的数据标签私钥 $SK_{UI, RN}$ 为 $(RN_{UI}, RN_{RN}, \sigma)$ 。

6 实验分析

6.1 实验环境

本文在 Windows10 操作系统中部署实现了 CIRG 方案原型。系统测试使用的计算机具体配置参数为: 处理器为 8 核 2.90 GHz Intel Core i7-10700, 内存 16 GB。实验使用 Java 语言, 通过 SHA-256 和开源的密码学算法库 JPBC 实现相应签名。采用 JAVA 内置随机数生成器生成实验测试中所用到的

随机数。调用 JAVA 自带的 SECURITY 类, 通过 RSA 非对称加密算法生成文件标签。RSA 算法的密钥长度为 1024 bit。根据性能设置, 仿真将每种实验情况运行 50 次, 以得到平均值。

6.2 系统功能测试

(1) 数据存储模块测试

如图 7 所示, 本文测试了密钥生成模块的时间开销。生成身份密钥 GenUIK 阶段和生成部分密钥 GenTK 阶段花费的时间较短, 分别约为 5 ms 和 4 ms。验证身份密钥 VerifyUIK 阶段和验证部分密钥 VerifyTK 阶段花费的时间相对较长, 分别约为 46 ms 和 48 ms。计算私钥 ComputeSK 阶段的时间约为 37 μ s。这 3 个阶段的总时间 (Sum) 约为 104 ms。

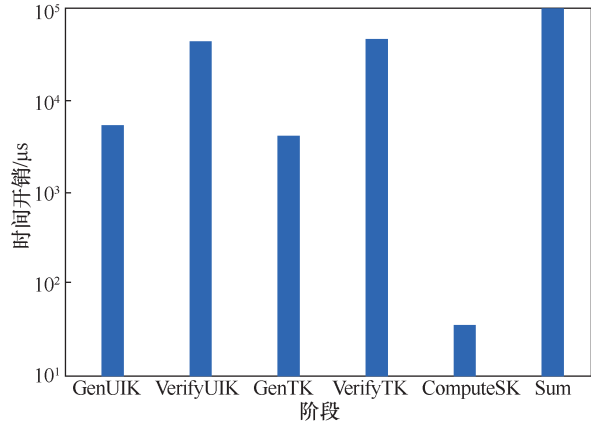


图 7 密钥生成模块的时间开销
Figure 7 Time cost of key generation module

如图 8 所示, 本文测试了密钥更新与密钥未更新的时间开销。当群组中没有用户离开时, 即密钥未更新时 (SAC-SK) 密钥生成的时间开销为 104 ms。当群组中某一用户撤群时, 密钥更新 (UDE-SK) 的时间开销为 156.02 ms。

本文对数据上传时不同数量文件在密钥生成、数据编码、标签生成的时间开销进行了测试。实验测试的每个文件大小为 1 MB, 分为 160 块, 其中冗余编码块为 32 块, 实验测试文件的数量从 2 增加到 128 个。如图 9 所示, 数据编码的时间开销约为密钥生成的 2.1 倍, 标签生成的时间开销约为数据编码的 764.7 倍。

(2) 完整性审计模块测试

图 10 为相应数据完整性审计的时间开销测

试结果。具体而言,设置每个文件大小为1 MB,每个文件块的大小为20 B,测试的文件数量从2增加到128个。当批量审计的文件数量为128时,挑战阶段、证明阶段和完整性验证阶段的时间成本分别为35.6 ms、91.9 ms、111.0 ms。

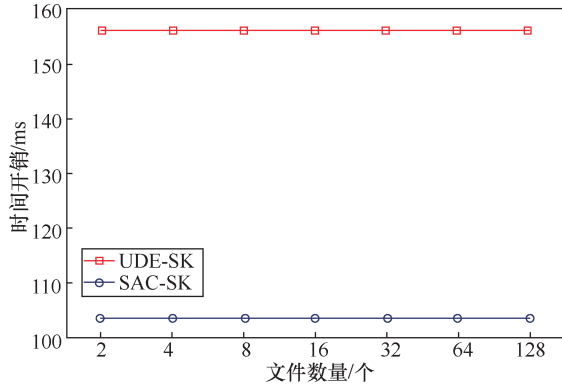


图8 密钥更新时的时间开销
Figure 8 Time cost when key update

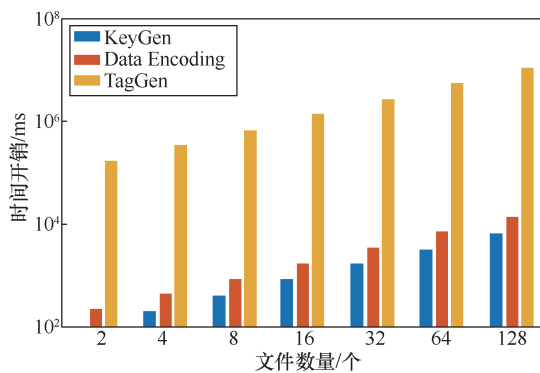


图9 数据上传时的时间开销
Figure 9 Time cost when data upload

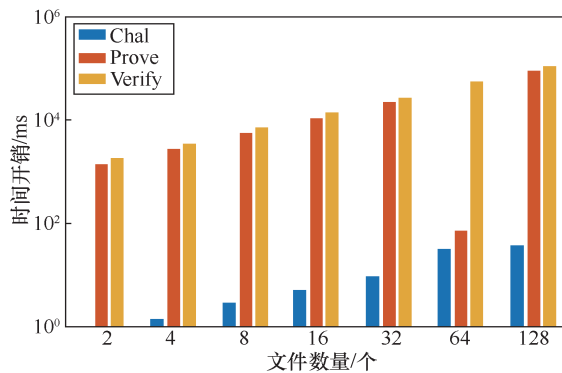


图10 完整性审计时的时间开销
Figure 10 Time cost when integrity audit

(3) 受损数据定位模块测试

本文分别对数据完整性审计与受损数据定位

这两种情况进行了对比测试。首先,测试受损文件数量不同时完整性审计和定位的时间开销,具体选取128个用户的文件进行测试,每个文件大小为1 MB,编码后大小为1.25 MB。文件编码之后的总块数为160,冗余块为32块。同时,设置不同数量的受损文件(数量分别从16增加到64个),并且每个文件的受损块数设置为32块。如图11所示,受损文件定位的时间开销约为完整性审计时间开销的3.95倍。

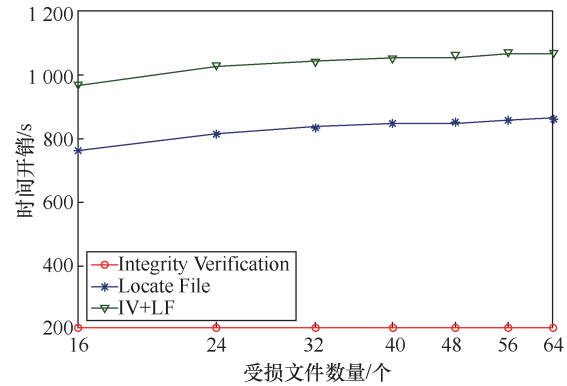


图11 受损文件数量不同的时间开销
Figure 11 Time cost when the number of damaged files varies

此外,本文还测试了审计文件数量不同时完整性审计和定位的时间开销。首先,设置受损文件的数量为4个,每个文件的大小是1 MB,测试的文件数量从8增加到128个。如图12所示,受损文件定位的时间开销约为完整性审计的2.64倍。

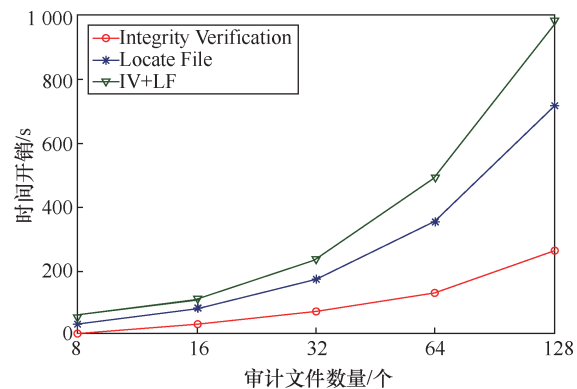


图12 审计文件数量不同的时间开销
Figure 12 Time cost when the number of audit files varies

(4) 受损数据恢复模块测试

通过选取不同文件数量,本文分别测试了数据编码和恢复阶段的时间开销。该测试将文件划分为128块,并且设置文件的冗余量大小为1/4,

即每个文件包含 32 块冗余, 编码之后的文件块数为 160, 设置文件的损坏率为 1/4。如图 13 所示, 数据编码和恢复的时间开销随着文件大小的增加而增大, 并且两者之间的差距较小。当文件 ≤ 16 MB 时, 数据恢复的时间开销小于数据编码时间。当 $16 \text{ MB} < \text{文件大小} \leq 32 \text{ MB}$ 时, 恢复时间开销大于编码时间开销。

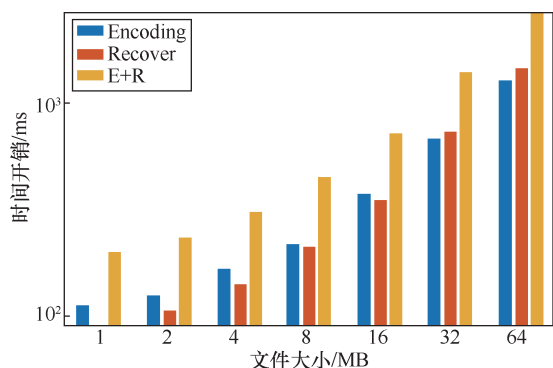


图 13 编码和恢复阶段的时间开销

Figure 13 Time cost when encode and recover process

(5) 对比测试

本文分别对方案的整体性能、受损数据定位这两种情况进行了对比测试。首先, 测试本文 CIRG 方案和文献[13]方案整体性能的时间开销, 具体选取 128 个用户的文件进行测试。同时, 设置不同数量的受损文件(数量分别从 16 增加到 64 个)。CIRG 方案测试的功能包括数据编码、数据完整性审计、受损数据定位与恢复。文献[13]方案测试的功能包括数据完整性审计、受损数据定位(二分法)。如图 14 所示, 与文献[13]方案相比, 本文的 CIRG 方案不仅在功能上更完善, 并且能够以较少的时间开销完成受损数据的快速定位与恢复。

通过选取不同数量的受损文件, 测试本文方案采用 RDI 累加法和文献[13]方案采用二分法定位受损数据的时间开销, 具体选取 128 个用户的文件进行测试, 每个文件大小为 1 MB, 文件编码后大小为 1.25 MB。文件编码后的总块数为 160, 冗余块数为 32。同时, 设置受损文件的数量分别从 16 增加到 64 个, 每个文件的受损块数为 32。如图 15 所示, 相较于文献[13]方案, 本文方案定位受损文件所用的时间开销明显较少。

综上, 实验分析和比较证明了 CIRG 方案在动态群用户场景下的可行性。同时, 通过测试不

同审计文件数量、不同受损文件数量等相关属性的时间开销可知, 方案的开销主要来自数据完整性验证和受损数据定位环节。由方案对比测试可知, CIRG 方案不但支持动态群用户数据受损的定位与恢复需求, 而且在数据完整性审计失败后定位与恢复受损数据比文献[13]需要更少的计算时间开销。

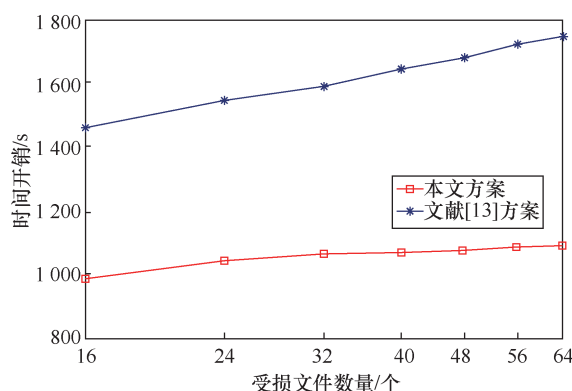


图 14 方案对比测试

Figure 14 Scheme comparison test

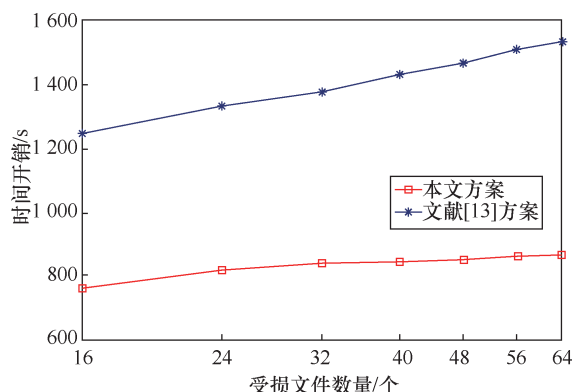


图 15 定位对比测试

Figure 15 Location contrast test

7 结束语

本文提出了一种公开的、可验证的存储证明方案 CIRG, 该方案支持在动态群用户环境中快速定位和恢复受损数据。方案不仅允许群用户高效审计存储在云服务器上共享数据的完整性, 更重要的是, 当审计未通过时(数据受损), 用户可以高效定位并恢复相应受损数据。相关安全性分析证明, 该方案在威胁模型下是安全的。本文具体实现了 CIRG 原型系统并进行了功能测试, 相关实验结果表明系统在动态群用户受损数据的定位与恢复中是有效的。

参考文献:

- [1] ZAFAR F, KHAN A, MALIK S U R, et al. A survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends[J]. Comput Secur, 2017, 65: 29-49.
- [2] GUMASTE P. Amazon aws outage[EB].
- [3] PAN Y, HU N. Research on dependability of cloud computing systems[C]//2014 10th International Conference on Reliability, Maintainability and Safety (ICRMS). 2014: 435-439.
- [4] SCHROEDER B, GIBSON G A. A large-scale study of failures in high-performance computing systems[J]. IEEE Trans Dependable Secur Comput, 2010, 7(4): 337-351.
- [5] WU D, XIA Y, SUN X S, et al. Masking failures from application performance in data center networks with shareable backup[C]//Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. 2018: 176-190.
- [6] ATENIESE G, BURNS R C, CURTMOLA R, et al. Provable data possession at untrusted stores[C]//Proceedings of the 2007 ACM Conference on Computer and Communications Security (CCS 2007). 2007: 598-609.
- [7] JUELS A, KALISKI B S. Pors: proofs of retrievability for large files[C]//Proceedings of the 2007 ACM Conference on Computer and Communications Security (CCS 2007). 2007: 584-597.
- [8] GUO H, ZHANG Z, XU J, et al. Accountable proxy re-encryption for secure data sharing[J]. IEEE Trans Dependable Secur Comput, 2021, 18(1): 145-159.
- [9] SENGUPTA B, RUJ S. Efficient proofs of retrievability with public verifiability for dynamic cloud storage[J]. IEEE Trans Cloud Comput, 2020, 8(1): 138-151.
- [10] SHEN J, ZHOU T, HE D, et al. Block design-based key agreement for group data sharing in cloud computing[J]. IEEE Trans Dependable Secur Comput, 2019, 16(6): 996-1010.
- [11] JIANG T, CHEN X, MA J. Public integrity auditing for shared dynamic cloud data with group user revocation[J]. IEEE Trans Computers, 2016, 65(8): 2363-2373.
- [12] LIU X, MA J, XIONG J, et al. Personal health records integrity verification using attribute based proxy signature in cloud computing[C]//Internet and Distributed Computing Systems - 6th International Conference (IDCS 2013). 2013: 238-251.
- [13] ZHANG Y, YU J, HAO R, et al. Enabling efficient user revocation in identity-based cloud storage auditing for shared big data[J]. IEEE Trans Dependable Secur Comput, 2020, 17(3): 608-619.
- [14] BERNSTEIN D J, DOUMEN J, LANGE T, et al. Faster batch forgery identification[C]//Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology. 2012: 454-473.
- [15] SHIN S, KIM S, KWON T. Identification of corrupted cloud storage in batch auditing for multi-cloud environments[C]//Information and Communication Technology-Third IFIP TC 5/8 International Conference (ICT), and 9th IFIP WG 8.9 Working Conference (CONFENIS 2015). 2015: 221-225.
- [16] MATT B J. Identification of multiple invalid signatures in pairing-based batched signatures[C]//Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, 2009: 337-356.
- [17] 王惠峰, 李战怀, 张晓, 等. 云存储中支持失效文件快速查询的批量审计方法[J]. 计算机学报, 2017, 40(10): 2338-2351.
WANG H F, LI Z H, ZHANG X, et al. Batch auditing supporting fast searching invalid files in cloud storage[J]. Chinese Journal of Computers, 2017, 40(10): 2338-2351.
- [18] ZHOU T, TIAN C. Fast erasure coding for data storage: a comprehensive study of the acceleration techniques[C]//17th USENIX Conference on File and Storage Technologies (FAST 2019), 2019: 317-329.
- [19] 伍祈应, 马建峰, 李辉, 等. 支持用户撤销的多关键字密文查询方案[J]. 通信学报, 2017, 38(8): 183-193.
WU Q Y, MA J F, LI H, et al. Multi-keyword search over encrypted data with user revocation[J]. Journal on Communications, 2017, 38(8): 183-193.
- [20] WANG C, CHOW S S M, WANG Q, et al. Privacy-preserving public auditing for secure cloud storage[J]. IEEE Trans Computers, 2013, 62(2): 362-375.
- [21] YU J, REN K, WANG C, et al. Enabling cloud storage auditing with key-exposure resistance[J]. IEEE Trans Inf Forensics Secur, 2015, 10(6): 1167-1179.
- [22] HUO H, JIANG T, TAN S, et al. Efficient public integrity auditing with secure deduplication in cloud computing[J]. Int J Embed Syst, 2019, 11(6): 764-777.
- [23] LI Y, YU Y, MIN G, et al. Fuzzy identity-based data integrity auditing for reliable cloud storage systems[J]. IEEE Trans Dependable Secur Comput, 2019, 16(1): 72-83.

[作者简介]



姜涛 (1984-), 男, 山东威海人, 博士, 西安电子科技大学副教授, 主要研究方向为密码学与安全协议、大数据存储安全和智能安全。



徐航 (1995-), 男, 陕西西安人, 西安电子科技大学硕士生, 主要研究方向为云存储数据安全。



王良民 (1977-), 男, 安徽潜山人, 博士, 东南大学教授、博士生导师, 主要研究方向为密码学与安全协议、物联网安全、大数据安全。



马建峰 (1963-), 男, 陕西西安人, 博士, 西安电子科技大学教授、博士生导师, 主要研究方向为计算机系统安全、移动与无线安全、系统可生存性和可信计算。