计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.181202371

一种面向云存储的数据动态验证方案

李树全　刘　磊　朱大勇　熊　超　李　锐
电子科技大学信息与软件工程学院　成都 610054

摘　要　云存储是一种新型的数据存储体系结构,云储存中数据的安全性、易管理性等也面临着新的挑战。由于用户在本地不再保留任何数据副本,无法确保云中数据的完整性,因此保护云端数据的完整性是云数据安全性研究的重点方向。数据完整性证明(Provable Data Integrity,PDI)被认为是解决这一问题的重要手段。文中提出了一种面向云存储环境的、基于格的数据完整性验证方案。本方案在已有研究的基础上,基于带权默克尔树(Ranked Merkle Hash Tree,RMHT),实现了云数据的动态验证。方案实现了数据粒度的签名,降低了用户方生成认证标签所需的消耗;引入 RMHT 对数据进行更改验证,支持数据动态更新;具有较强的隐私保护能力,在验证过程中对用户的原始数据进行盲化,使得第三方无法获取用户的真实数据信息,用户的数据隐私得到了有效的保护。此外,为了防止恶意第三方对云服务器发动拒绝服务攻击,方案中只有授权的第三方才能对用户数据进行完整性验证,这在保护云服务器安全的同时也保障了用户数据的隐私性。安全分析和性能分析表明,该方案不仅具有不可伪造性、隐私保护等特性,其签名计算量也优于同类算法。

关键词:云存储;公开验证;格密码;带权默克尔树;授权验证
中图法分类号　TP309

Protocol of Dynamic Provable Data Integrity for Cloud Storage

LI Shu-quan,LIU Lei,ZHU Da-yong,XIONG Chao and LI Rui
School of Information and Software Engineering,University of Electronic Science and Technology of China,Chengdu 610054,China

Abstract　Cloud storage is a novel data storage architecture. The security and manageability of data in cloud storage are also facing new challenges. Because users no longer store any copies of the data in their local memory,they cannot fully ensure whether the outsourced data are intact overall. How to protect the data integrity in the cloud has become a hot topic in academic research. The protocol of Provable Data Integrity(PDI) was considered to be the main method to solve this problem,this paper presented lattice-based provable data integrity for checking the integrity of the data in the cloud. The proposed scheme realizes the dynamic data verification by incorporating the idea of Ranked Merkle Hash Tree (RMHT) and lattice-based technology. The scheme realizes the fine-grained signature and reduces the computational cost required by the user to generate the authentication tag. The scheme introduces the RMHT to perform the modification verification of the data and supports the dynamic update of the data. It has strong privacy protection capability,blinds the user's original data during the verification process,and the third party cannot obtain user's real data information. Moreover,in order to prevent malicious third parties from launching denial-of-service attacks on cloud servers,only authorized third parties can verify the integrity of user data. Finally,security analysis and performance analysis show that the proposed scheme not only has characteristics of unforgeability and privacy protection,but also greatly reduces the computational cost of signature.

Keywords　Cloud storage,Public verification,Lattice-based cryptography,Ranked merkle hash tree (RMHT),Authorization checking

1　引言

近年来,互联网技术飞速发展,人们需要处理的数据量呈爆炸式增长,海量数据的存储和庞大的计算需求成为了普通用户的主要负担。云计算以其安全、可靠、扩展性强、性价比高等特点逐渐得到了用户的青睐。云数据服务是云计算的核心服务之一,越来越多的个人、公司和组织选择将他们的数据放在远程服务器上进行管理。然而,用户一旦选择将数据存储在云服务器中进行托管,便丧失了对数据的管理权,云端数据的安全性和可用性面临着极大的挑战[1-2]。云服务提供商为了盈利,可能会向用户隐瞒由于黑客攻击或自身拜占庭失效导致的数据丢失或损毁。用户需要防止不诚实的云服务提

到稿日期:2018-12-20　返修日期:2019-04-20　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
通信作者:李树全(shuquanli@uestc.edu.cn)

供商为了自身利益向用户隐瞒数据丢失或损坏,甚至为了节省存储空间删除访问频率低的文件的行为[3]。因此,设计一个能够确保用户数据真实、完整地存储在远程云服务器上的验证协议十分重要。

2007 年,Ateniese 等[4]首次给出了可证明数据持有(Provable Data Possession,PDP)协议的定义,该协议使用了同态验证标签(Homomorphic Verifiable Tags,HVTS)和随机验证机制,降低了验证所需的开销。在该协议的基础上,很多有效的数据完整性验证方案被提出,这些方案从高效验证[5]、动态操作[6]、隐私保护[7]等多个方面进行优化改进,但是其安全性是基于大整数分解[4]的困难性和 Diffie-Hellman 困难问题[7],无法抵御量子攻击[8]。

目前,基于格的困难问题还没有有效的破解方法,且被证明是可以抵御量子攻击的[10];此外,基于格的加密体制多使用小整数上的线性运算和模乘运算,与传统配对运算和指数运算相比效率有了很大的提升[10]。自 2008 年 Gentry 等[11]给出格上有效的签名方案后,基于格的签名体制便迅速发展,研究者从多个角度完善了基于格的签名体制[12-14]。

在云数据安全方面,通过利用格上的同态认证标签,很多基于格的数据完整性验证方案被提出。2013 年,Wang 等[15]构建了基于格的线性同态签名方案(Linear Homomorphic Signatures,LHS)。2014 年,Liu 等[16]基于 LHS 签名构造了一个支持公开验证的云存储验证方案,并在验证过程中使用随机向量来盲化原始数据,从而达到保护用户数据隐私的效果。但是,Zhang 等[17]指出这种方式会暴露原始数据的线性关系,易受到解线性方程的攻击。随后,Zhang 等[18]基于 GPV 签名构造了首个格上基于身份的云存储方案,并将隐私保护规约到格上困难问题。2016 年,Wang 等[20]改进了文献[16]中的方案,引入默克尔哈希树[19]对数据进行动态验证,构建了支持数据动态操作的验证协议。2017 年,Liu 等[21]给出了一个格上基于身份的线性同态签名,并以此设计了基于身份的远程数据验证方案。最近,Yan 等[22]又基于布隆过滤器实现了一个格上动态数据的验证方案。

以上基于格的数据完整性方案均只实现了最基本的数据验证功能,没有考虑到用户方的计算压力,签名计算量较大。另一方面,这些方案大多只实现了静态数据的完整性校验,而在真实的云存储环境中,数据的动态操作更为普遍。虽然文献[20]给出了一个动态验证方案,但是该方案的签名量较大,且没有严格的正确性证明。

为了解决上述问题,本文提出了一种格上基于用户授权的云数据动态验证方案。本文的主要贡献有:1)扩展了文献[18]的签名方案,实现了数据粒度的签名,降低了用户方生成认证标签所需的消耗;2)引入 RMHT 对数据进行更改验证,支持数据动态更新,从而实现了对数据子块的更新操作;3)所提方案具有较强的隐私保护能力,验证过程中非授权第三方无法发起有效的挑战请求,能够在保护云服务器的同时在一定程度上保障用户数据的隐私性,且在生成的完整性证据中对数据聚合进行盲化操作,使得第三方无法获取用户数据信息。

2 背景知识

2.1 符号定义

本文中,$\mathbf{R}$ 表示实数,$\mathbf{Z}$ 表示整数,$\|\boldsymbol{T}\|$ 表示向量或矩阵的欧几里德范数,$\widetilde{\boldsymbol{T}}$ 表示格基的施密特正交化。

2.2 格

矩阵 $\boldsymbol{B}=(\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_m)\in\mathbf{R}^{m\times m}$ 包含 m 个线性无关的向量,则由 $\boldsymbol{B}$ 生成的格为 $\Lambda=L(\boldsymbol{B})=\{\sum_{i=1}^{m}x_i\boldsymbol{b}_i\mid x_i\in\mathbf{Z}\}$,$B$ 是格 Λ 的一组基。通常情况下,格密码学中只对整数格进行研究,即格向量的每个分量都是整数。

定义 1 q 是一个素数,对于矩阵 $\boldsymbol{A}\in\mathbf{Z}_q^{n\times m}$,$\boldsymbol{u}\in\mathbf{Z}_q^n$ 有如下定义:

$$\Lambda_q^{\perp}(\boldsymbol{A})=\{\boldsymbol{e}\in\mathbf{Z}_q^m,\boldsymbol{A}\boldsymbol{e}=\boldsymbol{0}(\bmod q)\}$$

$$\Lambda_q^{u}(\boldsymbol{A})=\{\boldsymbol{e}\in\mathbf{Z}_q^m,\boldsymbol{A}\boldsymbol{e}=\boldsymbol{u}(\bmod q)\}$$

引理 1[23] 素数 $q\geqslant 3$,$m=\lceil 6n\log q\rceil$,则存在概率多项式算法 TrapGen(1^n,1^m,q)输出矩阵 $\boldsymbol{A}\in\mathbf{Z}_q^{n\times m}$,$\boldsymbol{T}\in\mathbf{Z}_q^{m\times m}$,使得 $\boldsymbol{A}$ 在 $\mathbf{Z}_q^{n\times m}$ 上接近于均匀分布。$\boldsymbol{T}$ 是格 $\Lambda_q^{\perp}(\boldsymbol{A})$的一组基,以压倒性的概率满足 $\|\boldsymbol{T}\|\leqslant O(n\log q)$,$\|\widetilde{\boldsymbol{T}}\|\leqslant O(\sqrt{n\log q})$。

2.3 格上离散高斯分布

定义 2 对于任意实数 $\sigma>0$,$\boldsymbol{c}\in\mathbf{R}^m$,$\mathbf{R}^m$ 上以 $\boldsymbol{c}$ 为中心、σ 为参数的高斯函数为 $\rho_{\sigma,\boldsymbol{c}}=\exp(-\pi\|\boldsymbol{x}-\boldsymbol{c}\|^2/\sigma^2)$,格 Λ 上以 $\boldsymbol{c}$ 为中心、σ 为参数的离散高斯函数定义为 $D_{\Lambda,\sigma,\boldsymbol{c}}=\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})/\rho_{\sigma,\boldsymbol{c}}(\Lambda)$。

引理 2[11] 设矩阵 $\boldsymbol{A}\in\mathbf{Z}_q^{n\times m}$,$\boldsymbol{T}\in\mathbf{Z}_q^{m\times m}$ 是格 $\Lambda_q^{\perp}(\boldsymbol{A})$的一组基,高斯参数 $\sigma_1\geqslant\|\widetilde{\mathbf{T}}\|\cdot\omega\sqrt{\log m}$,对于 $\boldsymbol{c}\in\mathbf{R}^m$,$\boldsymbol{u}\in\mathbf{Z}_q^n$,存在一个概率多项式时间的抽样算法 SamplePre($\boldsymbol{A}$,$\boldsymbol{T}$,$\boldsymbol{u}$,σ_1),抽取一个向量 $\boldsymbol{e}$,使得其分布统计接近于 $D_{\Lambda_q^{u}(\boldsymbol{A}),\sigma_1,\boldsymbol{c}}$。

2.4 格上相关算法和困难问题

定义 3(非齐次小整数解问题(Inhomogeneous Small Interger Solution Problem,ISIS)) 给定随机矩阵 $\boldsymbol{A}\in\mathbf{Z}_q^{n\times m}$ 和一个实数 β,找到一个非零向量 $\boldsymbol{e}$ 使得 $\boldsymbol{A}\boldsymbol{e}=\boldsymbol{u}\bmod q$ 且 $\|\boldsymbol{e}\|\leqslant\beta$。

引理 3[18] 对于多项式有界的 $poly(n)$,任意素数 $q\geqslant\beta\omega(\sqrt{n\log n})$,当近似因子为 $\gamma=\beta\widetilde{O}\sqrt{n}$ 时,平均情况下问题 $ISIS_{q,m,\beta}$与最坏情况的格上最短独立向量问题(The Shortest Independent Vector Problem,SIVP)的难度相等。

算法 1[26] 设 $q>2$,对于矩阵 $\boldsymbol{A}\in\mathbf{Z}_q^{n\times m}$ 及其对应格 $\Lambda_q^{\perp}(\boldsymbol{A})$的一组优质基 $\boldsymbol{T}_A\in\mathbf{Z}_q^{m\times m}$,从分布 $D_{m\times m}$中选取低范数可逆矩阵 $\boldsymbol{R}\in\mathbf{Z}_q^{m\times m}$、高斯参数 $\sigma_2>\|\widetilde{\boldsymbol{T}}\|\sigma_R\sqrt{m}\omega(\log^{3/2}m)$,则存在一个多项式算法 NewBasisDel($\boldsymbol{A}$,$\boldsymbol{R}$,$\boldsymbol{T}_A$,$\sigma_2$),输出格 $\Lambda_q^{\perp}(\boldsymbol{A}\boldsymbol{R}^{-1})$的优质基 $\boldsymbol{T}_B$,$\boldsymbol{T}_B$ 的分布接近于随机算法 RandBasis($\boldsymbol{T}$,σ_2)的输出分布,$\boldsymbol{T}$ 为格 $\Lambda_q^{\perp}(\boldsymbol{A}\boldsymbol{R}^{-1})$的一组基,满足 $\|\widetilde{\boldsymbol{T}}\|\leqslant\sigma_2/\omega(\sqrt{\log m})$。

2.5 带权默克尔树

文献[24]给出的 RMHT 与普通 MHT[19]相似,只是对每个节点加了一个权值来标记其数据子块的数目。

如图 1 所示,每个叶子节点 $N=\{H,r_N\}$对应一个数据块

$\boldsymbol{u}_i$，其中 H 表示哈希值，r_N 表示其拥有的数据子块数。非叶子节点 N_P 由其左孩子节点 $N_L=\{H_L, r_L\}$ 和右孩子节点 $N_R=\{H_R, r_R\}$ 按照 $\{h(H_L \parallel H_R), (r_L+r_R)\}$ 的方式构造而成。Ω_i 为数据块 $\boldsymbol{u}_i$ 的辅助验证信息，包含目标叶子节点到根节点路径上所有节点的兄弟节点。通过 $\boldsymbol{u}_i$ 及其辅助验证信息 Ω_i，按照 RMHT 的构造方法可以很快求得根节点的值。图 1 中 $\boldsymbol{u}_2$ 的辅助信息 Ω_2 包含虚线标识的路径上所有节点(黑色圆圈)的兄弟节点信息。

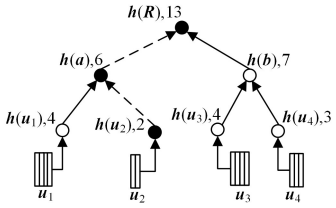

图 1 带权默克尔哈希树

Fig. 1 Ranked Merkle hash tree

3 格上基于用户授权的云数据动态验证方案

3.1 系统架构

如图 2 所示，本文的云存储系统由 3 个实体组成：用户(User)、云服务提供商(Cloud Server Provider，CSP)和第三方审计者(the Third Party Auditor，TPA)。本文算法的基本流程如下：1)密钥生成阶段，用户和 CSP 各自生成自身公私钥对，用户通过自己的密钥生成授权证据，然后通过秘密信道将其发送给 TPA；2)认证标签生成阶段，用户对原始文件分块后签名，然后通过分块数据哈希值构建 RMHT，使用自身密钥对根节点哈希值进行签名，最后将文件及签名发送给 CSP，并删除本地副本；3)公开验证阶段，TPA 生成随机挑战请求发送给 CSP，CSP 根据挑战请求生成完整性证据返回给 TPA，TPA 对完整性证据进行验证，并将验证结果发送给用户。

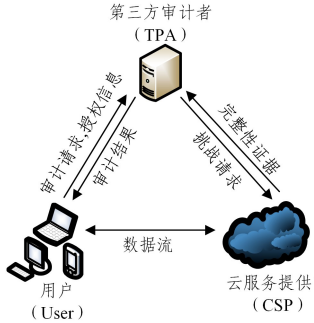

图 2 云存储系统模型

Fig. 2 System model of cloud storage

本文将 TPA 定义为半可信且好奇的第三方，其可能会通过接收的数据来分析用户的原始文件信息，但会正确地执行自己的工作，且其与 CSP 无合谋，因为如果 CSP 与 TPA 合谋，那么用户的授权证据将变得毫无意义，且用户永远不会知道云端文件的真实存储信息。

3.2 方案描述

本文提出的方案由 6 个多项式算法组成：KeyGen，VerDesig，TagGen，ChalGen，ProofGen，ProofCheck。设定 σ_1 和 σ_2 为满足算法 SamplePre() 和 NewBasisDel() 安全性的高斯参数。此外，该方案还需要 3 个安全哈希函数：$H_1:\{0,1\}^* \rightarrow \mathbf{Z}_q^{m\times m}$，$H_1$ 的输出在 $\boldsymbol{D}_{m\times m}$ 是均匀分布的 $H_2:\{0,1\}^* \rightarrow \mathbf{Z}_q^n$；$H_3:\mathbf{Z}_q^n \rightarrow \mathbf{Z}_q$。

KeyGen：给定安全参数 n 和 $params=\{q, m, \widetilde{L}\}$，其中 $q=poly(n)$，$m>2n\log q$，$\widetilde{L}=O(\sqrt{n\log q})$。用户执行引理 1 中的算法 TrapGen($1^n$, 1^m, q)，生成均匀随机矩阵 $\boldsymbol{B}\in\mathbf{Z}_q^{n\times m}$ 和格 $\Lambda_q^{\perp}(\boldsymbol{B})$ 的短基 $\boldsymbol{T}_{ID}\in\mathbf{Z}_q^{m\times m}$ 作为其公私钥对；同理，CSP 运行该算法生成其公私钥对 $(\boldsymbol{Q}, \boldsymbol{T}_{CSP})$。

VerDesig：为了防止恶意的攻击者频繁地发动验证请求，从而导致云服务器瘫痪，本方案中只有被用户授权的 TPA 才能发出有效的完整性验证请求。对于身份信息为 ID_{TPA} 的授权 TPA，用户计算 $\boldsymbol{P}=\boldsymbol{B}\boldsymbol{R}_{ID_{TPA}}^{-1}=\boldsymbol{B}H_1(ID_{TPA})^{-1}$；然后调用格基委派算法 NewBasisDel($\boldsymbol{B}, \boldsymbol{R}_{ID_{TPA}}, \boldsymbol{T}_{ID}, \sigma_2$)输出 $\boldsymbol{T}_{TPA}$，$\boldsymbol{T}_{TPA}$ 为格 $\Lambda_q^{\perp}(\boldsymbol{P})$ 的格基；最后将 $(\boldsymbol{P}, \boldsymbol{T}_{TPA})$ 通过安全信道发送给对应的授权 TPA。

TagGen：对于标签为 id 的文件 F，首先对数据进行分块。将文件 F 分割为 l 个数据块 $F=\{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_l\}$，每个数据块 $\boldsymbol{u}_i$ 再划分为 r 个数据子块，不足时进行补零。最终，文件 F 被划分为 $l\times r$ 个数据子块，使得每个数据子块 $\boldsymbol{u}_{i,j}\in\mathbf{Z}_q^m$。然后，按照如下步骤生成认证标签。

1)利用文件标识符 id 和分割文件数据块索引 i 计算 $\boldsymbol{h}_i=H_2(id \parallel i)$。随机选取 r 个元素 $a_j\in\mathbf{Z}_q$，计算 $\boldsymbol{U}_i=\boldsymbol{h}_i+\boldsymbol{Q}\sum_{j=1}^{r}a_j\boldsymbol{u}_{i,j}$。这是为了防止 CSP 进行压缩存储，由于验证时需要对数据子块单独聚合，因此必须保证 CSP 真实地存储每个数据子块。调用 SamplePre($\boldsymbol{B}, \boldsymbol{T}_{ID}, \boldsymbol{U}_i, \sigma_1$)产生一个向量 $\boldsymbol{e}_i\in\mathbf{Z}_q^m$，其满足 $\boldsymbol{B}\boldsymbol{e}_i=\boldsymbol{U}_i \bmod q$，且 $\parallel \boldsymbol{e}_i \parallel \leqslant\sigma_1\sqrt{m}$，记 $\boldsymbol{\Phi}=\{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_l\}$ 为认证标签集合。

2)构造 RMHT，使用安全哈希函数 H_2 计算数据块 $\boldsymbol{u}_i$ 的哈希值作为叶子节点，然后按照前文中 RMHT 的构造方法逐层构造非叶子节点，最终生成根节点 $R=\{H_R, r_R\}$。运行算法 SamplePre($\boldsymbol{B}, \boldsymbol{T}_{ID}, H_R, \sigma_1$)，产生对根节点哈希值 H_R 的签名 $Sig(H_R)$。

3)用户将 $\{F, \boldsymbol{\Phi}, id, Sig(H_R)\}$ 发送给 CSP，并删除本地副本，保留 RMHT 中各个节点的权值以及数据块的索引信息，用于后续更新操作的位置查找。

ChalGen：收到用户的完整性验证请求后，根据安全要求，TPA 从文件 F 的 l 个分块索引集合中随机挑选含有 c 个索引的子集作为抽样数据块的下标集合，记为 I。文献[25]的研究结果表明，当数据分块数较多时，只需要随机取回 300～460 个数据块进行验证，便可以达到 95%的损坏识别率。为每一个 $i\in I$ 随机选择一个元素 $v_i\in\mathbf{Z}_q$，记 $\boldsymbol{\Theta}=\{i, v_i\}_{i\in I}$。然后随机挑选向量 $\boldsymbol{\theta}\in\mathbf{Z}_q^n$，运行 SamplePre($\boldsymbol{P}, \boldsymbol{T}_{TPA}, \boldsymbol{\theta}, \sigma_1$)生成对 $\boldsymbol{\theta}$ 的签名 $sig\in\mathbf{Z}_q^m$。将二者组合，生成挑战请求 $Chal=\{\boldsymbol{\Theta}, (\boldsymbol{\theta}, sig)\}$ 发送给 CSP。

ProofGen：接收到 TPA 的挑战请求后，CSP 首先对 TPA 的授权信息进行验证。计算 $\boldsymbol{P}=\boldsymbol{B}H_1(ID_{TPA})^{-1}$，验证 $\boldsymbol{P}\cdot sig=\theta$ 及 $\parallel sig \parallel \leqslant\sigma_1\sqrt{m}$ 是否均成立，若成立则表示提出验证请求的为授权 TPA，然后按照如下步骤生成完整性证据 $proof$。

1) 选取一个随机向量 $\boldsymbol{v}\in\boldsymbol{Z}_q^n$,运行 SamplePre($\boldsymbol{Q},\boldsymbol{T}_{CSP},\boldsymbol{v},\sigma_1$),生成对 $\boldsymbol{v}$ 的签名 $\boldsymbol{\xi}\in\boldsymbol{Z}_q^m$。

2) 计算 $\boldsymbol{U}_{com,j}=\sum_{i\in I}v_i\boldsymbol{u}_{i,j}$,利用 $\boldsymbol{v}$ 对其进行盲化,计算 $\boldsymbol{U}'_{com,j}=\boldsymbol{U}_{com,j}+\boldsymbol{\xi}H_3(\boldsymbol{v})$,记盲化后的数据块聚合集合为 $\boldsymbol{U}'_{com}=\{\boldsymbol{U}'_{com,j}\}_{j\in[r]}$,计算签名聚合 $\boldsymbol{e}_{com}=\sum_{i\in I}\boldsymbol{v}_i\boldsymbol{e}_i$。

3)计算抽样数据块哈希值 $h(\boldsymbol{u}_i)=H_2(\boldsymbol{u}_i)$及其辅助验证信息 Ω_i。

4)CSP 将完整性验证证据 $proof=\{\boldsymbol{U}'_{com},\boldsymbol{e}_{com},\{h(\boldsymbol{u}_i),\Omega_i\}_{i\in I},Sig(H_R),\boldsymbol{v}\}$发送给 TPA。

ProofCheck:TPA 按照如下步骤对完整性证据 $proof$ 的正确性进行验证。

1)验证 RMHT 的正确性。根据$\{h(\boldsymbol{u}_i),\Omega_i\}_{i\in I}$求得 RMHT 的根节点哈希值 H_R',验证 $\boldsymbol{B}\cdot Sig(H_R)=H_R'$及 $\|Sig(H_R)\|\leqslant\sigma_1\sqrt{m}$是否均成立。

2)验证抽样数据块聚合与签名聚合的对应关系,计算 $\boldsymbol{h}_i=H_2(id\|i)$,然后验证

$$\boldsymbol{B}\boldsymbol{e}_{com}=\sum_{i\in I}v_i\boldsymbol{h}_i+\boldsymbol{Q}\sum_{j=1}^{r}a_j\boldsymbol{U}'_{com,j}-\sum_{j=1}^{r}a_j\boldsymbol{v}H_3(\boldsymbol{v}) \tag{1}$$

以及 $\|\boldsymbol{e}_{com}\|\leqslant c\sigma_1\sqrt{m}$。

以上验证过程循序渐进,若有一个不通过则可以认为数据块不完整,无须进行后面的验证,返回 false;若全部通过验证,则表示完整性证据验证通过,返回 true。

3.3 动态操作

3.3.1 动态操作的定义

对于云存储用户,数据的基本动态操作包括修改、插入、删除 3 种。文献[24]将数据块层面的更新操作定义为 $Type=\{PM,M,D,J,SP\}$。其中,PM 表示局部更新操作,对一个数据块内部的数据子块进行插入、删除、修改操作;J,D,M 分别表示数据整块的插入、删除、修改操作;SP 表示数据块分割操作。文献[24]证明了任意一个动态更新操作都可以通过这 5 个基本操作组合来实现。图 3 以一个基本的跨数据块的插入操作为例,展示了一个更新操作是如何由多个子操作实现的。图 3 中的最大子块数 r 设定为 5,当用户想将 6 个子块插入到 $\boldsymbol{u}_2$中时,子块数超过 r,需要首先通过 SP 操作将 $\boldsymbol{u}_2$分割为 $\boldsymbol{u}_2'$和 $\boldsymbol{u}^*$,再分别对其进行 PM 操作来完成整个更新操作。

定义更新请求 $Update=\{Type,i,o,\boldsymbol{u}_{new}\}$。其中,$i$ 表示目标数据块 $\boldsymbol{u}_i$ 的索引;o 表示更新位置在文件中的偏移量;$\boldsymbol{u}_{new}$表示要更新的数据,当执行删除操作时,$\boldsymbol{u}_{new}$为空。在叙述动态操作时,为了简单起见,只对 RMHT 节点的 Hash 值进行操作。事实上,节点的权值也需要进行更改。

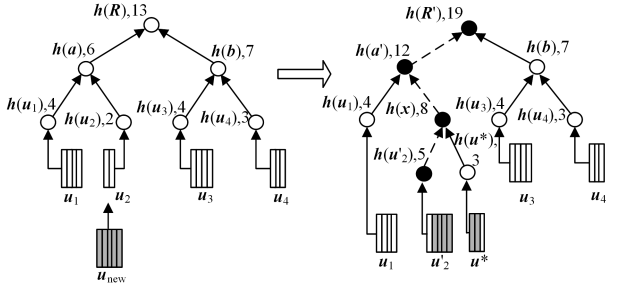

图 3 插入操作示例

Fig. 3 Example of insert operation

3.3.2 局部更新操作

局部更新操作 PM 可以对一个数据块内的连续数据子块进行插入、修改、删除,该操作不会波及其他数据块。其主要流程是将受影响的数据块下载下来,对其进行正确性验证后,生成并更新签名,PM 操作中将更新类型 $Type$ 修改为具体的数据更新操作。PM 操作有以下 4 个步骤:

1)用户想要在指定位置 o 对文件进行更新操作,首先调用文献[24]给出的 $FindBlock()$算法,该算法通过本地保留的 RMHT 权值快速查找文件偏移位置 o 所在数据块的索引 i。记操作类型 $operation=\{$insert,delete,modfiy$\}$表示对局部数据子块的插入、删除、修改,然后生成更新请求 $Update=\{operation,i,o,\boldsymbol{u}_{new}\}$发送给 CSP。

2)CSP 执行更新算法 UpdateExe($Update$),根据 $Update$ 信息确认更新操作后,生成验证信息 $P_{Update}=\{h(\boldsymbol{u}_i),\Omega_i,H_R'\}$,其中 H_R'为将 $\boldsymbol{u}_i$根据 $Update$ 信息更新为 $\boldsymbol{u}_i'$后,通过$\{\boldsymbol{u}_i',\Omega_i\}$计算出的新的根节点哈希值。最后,将 P_{Update}及原始数据 $\boldsymbol{u}_i$发送给用户。

3)用户执行更新验证算法 UpdateVerify(P_{Update})。首先对原始数据的正确性进行验证,根据 P_{Update}中的$\{h(\boldsymbol{u}_i),\Omega_i\}$计算原始根哈希值 H_R',验证 $\boldsymbol{B}\cdot Sig(H_R)=H_R^*$ 和 $\|Sig(H_R)\|\leqslant\sigma_1\sqrt{m}$是否均成立,不成立则返回 false,否则执行后续验证。后续验证中,用户根据 $\boldsymbol{u}_i$和 $\boldsymbol{u}_{new}$得到 $\boldsymbol{u}_i'$,计算 $h(\boldsymbol{u}_i')$,结合 P_{Update}中的$\{\Omega_i\}$计算新的根哈希值 H_{new},并验证其与 H_R'是否相等,以判断 CSP 是否正确地完成了数据更新操作。以上验证全部通过后,用户使用自身密钥运行抽样算法,产生新的 RMHT 根节点的签名 $Sig(H_{new})$及更新后数据块 $\boldsymbol{u}_i'$的签名 $\boldsymbol{e}_i'$,并将二者发送给 CSP。

4)CSP 将对应的数据块签名 $\boldsymbol{e}_i$更新为 $\boldsymbol{e}_i'$,并将根节点签名替换为 $Sig(H_{new})$,整个更新操作完成。

3.3.3 其他动态操作

数据整块的操作 J,D,M 中关于默克尔树结构调整的部分,文献[20]已经给出了详细的介绍。对数据整块的更新验证无须使用原始数据文件,比 PM 操作简洁得多。对于 J,M 操作,用户将需要更新的数据块及其签名和 $Update$ 信息一起发送给 CSP,然后对 CSP 返回的更新操作信息的正确性进行验证即可。D 操作则更为简洁,用户无须生成更新数据块签名,更新操作也只是简单的默克尔树结构的调整。

SP 操作和 PM 操作的过程类似,不同的是 SP 操作在将 $\boldsymbol{u}_i$更新为 $\boldsymbol{u}_i'$后,还需要产生一个数据块 $\boldsymbol{u}^*$插在树中 $\boldsymbol{u}_i'$的位置之后。验证过程中,需要使用$\{h(\boldsymbol{u}_i'),h(\boldsymbol{u}^*),\Omega_i\}$来代替$\{h(\boldsymbol{u}_i'),\Omega_i\}$生成新的根节点哈希值,这一点从图 3 中可以明显看出。

4 方案分析

4.1 正确性分析

首先,对验证过程的 RMHT 的正确性进行验证分析。由 RMHT 的构造方式可知,只需要对根节点的正确性进行验证,便可以判定整棵树的节点是否被修改。对于根节点签名 $Sig(H_R)$,若 $\boldsymbol{B}\cdot Sig(H_R)=H_R^*$,$\|Sig(H_R)\|\leqslant\sigma_1\sqrt{m}$不

成立，则表示 CSP 提供的数据哈希值和辅助验证信息至少有一方不正确，可以认为原数据块被损毁。但是，因为 CSP 可能会预先计算原始数据的哈希值并保留下来，所以即使通过了该验证，还需要验证抽样数据块聚合与签名聚合之间是否满足等式(1)来确保原始数据的正确性。

若 TPA 与 CSP 均诚实地完成了自己的工作，则本文给出的验证方案是正确的，推导过程如下：

$$\begin{aligned}\boldsymbol{Be}_{com}&=\boldsymbol{B}\sum_{i\in I}v_i\boldsymbol{e}_i=\sum_{i\in I}v_i\boldsymbol{Be}_i\\&=\sum_{i\in I}v_i(\boldsymbol{h}_i+\boldsymbol{Q}\sum_{j=1}^{r}a_j\boldsymbol{u}_{i,j})\\&=\sum_{i\in I}v_i\boldsymbol{h}_i+\boldsymbol{Q}\sum_{j=1}^{r}a_j\sum_{i\in I}v_i\boldsymbol{u}_{i,j}\\&=\sum_{i\in I}v_i\boldsymbol{h}_i+\boldsymbol{Q}\sum_{j=1}^{r}a_j(\boldsymbol{U}'_{com,j}-\boldsymbol{\xi}H_3(\boldsymbol{v}))\\&=\sum_{i\in I}v_i\boldsymbol{h}_i+\boldsymbol{Q}\sum_{j=1}^{r}a_j\boldsymbol{U}'_{com,j}-\sum_{j=1}^{r}a_j\boldsymbol{v}H_3(\boldsymbol{v})\end{aligned}$$

原式得证，因此对于 $\|\boldsymbol{e}_{com}\|$，有 $\|\boldsymbol{e}_i\|\leqslant\sigma_1\sqrt{m}$，则有 $\|\boldsymbol{e}_{com}\|=\|\sum_{i\in I}v_i\boldsymbol{e}_i\|\leqslant c\sigma_1\sqrt{m}$。

4.2 安全性分析

4.2.1 不可伪造性

CSP 作为半可信实体，为了自身利益，可能会向用户隐瞒数据丢失的情况，仍然向用户声明数据正确地存储在服务器中；恶意云服务器甚至会篡改用户的数据，并伪造完整性证据，企图通过 TPA 的验证。文献[20]认为通过验证默克尔哈希树的根节点，可以判定 CSP 是否篡改用户数据。正如本文在正确性分析中指出的，恶意 CSP 可能会预先计算原始数据的哈希值并保留下来，所以上述方法只能在一定程度上说明数据的完整性。本文将数据的不可伪造性规约到格上的困难问题，如定理 1 所示，从而使得所提方案可以抵御恶意云服务器的伪造攻击。

定理 1 如果 ISIS 问题是困难的，则本方案中恶意的云服务器生成的无效的完整性证据是无法通过 TPA 的完整性校验的。

证明：假设一个恶意云服务器以不可忽略的概率将用户的数据子块 $\boldsymbol{u}_{k,s}$ 篡改为 $\boldsymbol{u}^*_{k,s}$，将 $\boldsymbol{u}_{k,s}$ 所在数据块的签名 $\boldsymbol{e}_k$ 修改为 $\boldsymbol{e}^*_k$，生成了伪造的完整性证据，那么它将进行如下操作：选取一个随机向量 $\boldsymbol{v}^*\in\boldsymbol{Z}_q^n$，运行 SamplePre($\boldsymbol{Q},\boldsymbol{T}_{CSP},\boldsymbol{v}^*,\sigma_1$)生成对 $\boldsymbol{v}^*$ 的签名 $\boldsymbol{\xi}^*\in\boldsymbol{Z}_q^m$。因此，恶意的 CSP 可以计算伪造的数据聚合 $\boldsymbol{U}^*_{com}=\{\boldsymbol{U}^*_{com,1},\cdots,\boldsymbol{U}^*_{com,s},\cdots,\boldsymbol{U}^*_{com,r}\}$ 及伪造签名聚合 $\boldsymbol{e}^*_{com}=\sum_{i\in I,i\neq k}v_i\boldsymbol{e}_i+v_k\boldsymbol{e}^*_k$。对于伪造的数据聚合 $\boldsymbol{U}^*_{com,j}$，有：

$$\boldsymbol{U}^*_{com,j}=\sum_{i\in I,i\neq k}v_i\boldsymbol{u}_{i,j}+v_k\boldsymbol{u}_{k,j}+\boldsymbol{\xi}^*H_3(\boldsymbol{v}^*)\tag{2}$$

最终，恶意 CSP 生成了伪造的完整性验证证据。如果该伪造的证据通过了 TPA 的验证，则其应满足等式(1)，有：

$$\boldsymbol{Be}^*_{com}=\sum_{i\in I}v_i\boldsymbol{h}_i+\boldsymbol{Q}\sum_{j=1}^{r}a_j\boldsymbol{U}^*_{com,j}-\sum_{j=1}^{r}a_j\boldsymbol{v}^*H_3(\boldsymbol{v}^*)$$

同时，对于未篡改的签名聚合 $\sum_{i\in I,i\neq k}v_i\boldsymbol{e}_i$，存在对应的未被篡改的数据聚合 $\boldsymbol{U}'_{com,j}=\sum_{i\in I,i\neq k}v_i\boldsymbol{u}_{i,j}+\boldsymbol{\xi}'H_3(\boldsymbol{v}')$，可以产生有效的验证证据，其中 $\boldsymbol{v}'\in\boldsymbol{Z}_q^n$ 为另一个随机向量。运行 SamplePre($\boldsymbol{Q},\boldsymbol{T}_{CSP},\boldsymbol{v}',\sigma_1$)生成签名 $\boldsymbol{\xi}'$，同样满足等式(1)，可得 $\boldsymbol{B}\sum_{i\in I,i\neq k}v_i\boldsymbol{e}_i=\sum_{i\in I,i\neq k}v_i\boldsymbol{h}_i+\boldsymbol{Q}\sum_{j=1}^{r}a_j\boldsymbol{U}'_{com,j}-\sum_{j=1}^{r}a_j\boldsymbol{v}'H_3(\boldsymbol{v}')$ 且 $\boldsymbol{U}'_{com,j}=\boldsymbol{U}^*_{com,j}-v_k\boldsymbol{u}_{k,j}-\boldsymbol{\xi}^*H_3(\boldsymbol{v}^*)+\boldsymbol{\xi}'H_3(\boldsymbol{v}')$，则 $\boldsymbol{Be}^*_{com}=\boldsymbol{B}\sum_{i\in I,i\neq k}v_i\boldsymbol{e}_i+\boldsymbol{B}v_k\boldsymbol{e}^*_k=\sum_{i\in I,i\neq k}v_i\boldsymbol{h}_i+\boldsymbol{Q}\sum_{j=1}^{r}a_j\boldsymbol{U}'_{com,j}-\sum_{j=1}^{r}a_j\boldsymbol{v}'H_3(\boldsymbol{v}')+\boldsymbol{B}v_k\boldsymbol{e}^*_k$。

由此可得：$\boldsymbol{Be}^*_{com}=\sum_{i\in I,i\neq k}v_i\boldsymbol{h}_i+\boldsymbol{Q}\sum_{j=1}^{r}a_j(\boldsymbol{U}^*_{com,j}-v_k\boldsymbol{u}_{k,j})-\sum_{j=1}^{r}a_j\boldsymbol{v}^*H_3(\boldsymbol{v}^*)+\boldsymbol{B}v_k\boldsymbol{e}^*_k$。

结合式(2)，可得 $\boldsymbol{Be}^*_k=\boldsymbol{h}_k+\boldsymbol{Q}\sum_{j=1}^{r}a_j\boldsymbol{u}_{k,j}$。

令 $\boldsymbol{U}^*_k=\boldsymbol{h}_k+\boldsymbol{Q}\sum_{j=1}^{r}a_j\boldsymbol{u}_{k,j}$，可知 $\boldsymbol{U}^*_k$ 是 $\boldsymbol{Z}_q^n$ 上的一个非零向量，所以在不知道用户密钥 $\boldsymbol{T}_{ID}$ 的情况下，恶意 CSP 篡改用户数据及对应签名并通过 TPA 的验证，相当于找到了 $\boldsymbol{U}^*_k$ 的一个签名 $\boldsymbol{e}^*_k$，满足 $\boldsymbol{Be}^*_k=\boldsymbol{U}^*_k$，即找到了 ISIS 实例的一个解。可知，如果恶意 CSP 以不可忽略的概率篡改用户数据及签名并通过 TPA 的验证，那么我们可以构造一个多项式算法以同样的概率解决 ISIS 问题，这与 ISIS 问题的难度是相违背的，所以本文方案可以抵御恶意云服务器的伪造攻击。

4.2.2 隐私保护

前文提到文献[16,20]使用随机向量来保护数据隐私的方法会暴露数据的线性关系，易受到第三方验证者的解线性方程攻击；而本文方案中，TPA 无法从完整性证据中获取任何关于原始数据的线性组合信息。对于盲化后的数据子块聚合 $\boldsymbol{U}'_{com,j}=\boldsymbol{U}_{com,j}+\boldsymbol{\xi}H_3(\boldsymbol{v})$，TPA 要想根据 $\boldsymbol{U}'_{com,j}$ 来获取原始数据聚合 $\boldsymbol{U}_{com,j}$，从而得到原始数据的线性组合信息，则需要在不知道 CSP 的密钥 $\boldsymbol{T}_{CSP}$ 的情况下解得 $\boldsymbol{\xi}$。基于 GPV 签名[11]的安全性，好奇的 TPA 无法通过完整性证据来获得关于原始数据的线性信息，所以无法通过求解线性方程来获取原始数据[17]。

另一方面，在完整性验证的过程中，只有授权的 TPA 才能发起有效的挑战请求，这在一定程度上也保护了用户的数据隐私。

4.3 算法性能分析

表 1 对比了几种基于格的云数据完整性验证方案的性能。认证标签的生成计算量主要集中在抽样算法的运行上，其余都是一些杂凑运算和格上的线性运算。本文采用文献[16]的方法，通过对比原象抽样算法 SamplePre()的运行次数来衡量计算量。表 1 中，psf 表示运行一次抽样算法，$l\times r$ 代表数据子块的总数量，l 代表数据块的个数，r 代表每个数据块中数据子块的个数。

表 1 不同方案的效率及功能的比较

Table 1 Comparison of efficiency and function for different schemes

方案	隐私保护	动态操作	细粒度划分	签名计算量
本文方案	是	是	否	$l\cdot psf$
文献[17]	是	否	是	$lr\cdot psf$
文献[20]	是	否	否	$lr\cdot psf$

由表 1 可以看出，与文献[17,20]方案相比，本方案不仅功能更加完善，且基于数据粒度的签名方法使得原象抽样算法的运行次数减少为对比方案的 $1/r$，从而使签名的计算量与数量减少为对比方案的 $1/r$，极大地削减了用户方的计算量。虽然本方案进行验证时，计算量和通信消耗有所增加，但大数据文件的分块数较多，与削减的签名计算量相比，这些消耗的增加是可以接受的。此外，用户还可以根据实际情况对

r 进行设定,在方案的签名计算量和各方的通信消耗量之间达到一个较好的平衡。

结束语 本文提出了一个格上基于用户授权的动态数据完整性验证方案,该方案支持公开验证、用户隐私保护和动态操作。基于数据粒度的签名方法减少了签名所需的计算量和签名的数量,并且用户可以根据自己的需求设定子块的数量,因此该方法有良好的扩展性。在效率方面,基于格的密码体制使得本方案的计算大多是格上的线性计算,与传统的双线性对方案[5-7]相比,在计算效率方面有明显提高[10]。在安全性方面,本方案可以规约到格上的小整数解问题,因此能够抵抗量子攻击性,有广阔的应用前景。

参 考 文 献

[1] WEI L,ZHU H,CAO Z,et al. SecCloud:Bridging Secure Storage and Computation in Cloud[C]// IEEE International Conference on Distributed Computing Systems Workshops. IEEE,2010.

[2] FENG D G,ZHANG M,ZHANG Y. et al. Study on Cloud Computing Security[J]. Journal of Software,2011,22(1):71-83.

[3] THANGAVEL M,VARALAKSHMI P,SINDHUJA R,et al. A survey on provable data possession in cloud storage[C]// Eighth International Conference on Advanced Computing. IEEE,2017.

[4] ATENIESE G,BURNS R,CURTMOLA R,et al. Provable data possession at untrusted stores [C]// Proceedings of the 14th ACM Conference on Computer and Communications Security. 2007:598-609.

[5] SHEN J,SHEN J,CHEN X,et al. An Efficient Public Auditing Protocol With Novel Dynamic Structure for Cloud Data[J]. IEEE Transactions on Information Forensics & Security,2017,12(10):2402-2415.

[6] JIN Y,CAI C,HE H,et al. BTDA:Dynamic Cloud Data Updating Audit Scheme Based on Semi-trusted Third Party[J]. Computer Science,2018,45(3):144-150.

[7] RAZAQUE A,RIZVI S S. Privacy preserving model:a new scheme for auditing cloud stakeholders[J]. Journal of Cloud Computing,2017,6(1):7.

[8] SHOR SHOR P W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer[J]. SIAM Review,1999,41(2):303-332.

[9] MICCIANCIO D,REGEV O. Worst-Case to Average-Case Reductions Based on Gaussian Measures[C]// 45th Annual IEEE Symposium on Foundations of Computer Science. IEEE,2004:372-381.

[10] BUCHMANN J,LINDNER R,SCHNEIDER M. Post-quantum cryptography:lattice signatures[J]. Computing,2009,85(1/2):105-125.

[11] GENTRY C,PEIKERT C,VAIKUNTANATHAN V. Trapdoors for hard lattices and new cryptographic constructions [C]// Proceedings of the 40th Annual ACM Symposium on Theory of Computing. Victoria:ACM,2008.

[12] PEIKERT C. Bonsai trees (or,arboriculture in lattice based cryptog2raphy)[J]. Manuscript,2009(1/2):147-191.

[13] LING S,NGUYEN K,ROUX-LANGLOIS A,et al. A lattice-based group signature scheme with verifier-local revocation[J]. Theoretical Computer Science,2018,730:1-20.

[14] GAO W,CHEN L,HU Y,et al. Lattice-based deniable ring signatures[J]. International Journal of Information Security,2019,18(3):355-370.

[15] WANG F H,HU Y P,WANG B C. Lattice-based linearly homomorphic signature scheme over binary field[J]. Science China (Information Sciences),2013,56(11):1-9.

[16] LIU H,CAO W. Public Proof of Cloud Storage from Lattice Assumption[J]. Chinese Journal of Electronics,2014,23(1):186-190.

[17] ZHANG X,XU C,ZHANG Y,et al. Insecurity of a Public Proof of Cloud Storage from Lattice Assumption[J]. Chinese Journal of Electronics,2017,26(1):88-92.

[18] ZHANG X,XU C. Efficient Identity-Based Public Auditing Scheme for Cloud Storage from Lattice Assumption[C]// IEEE,17th International Conference on Computational Science and Engineering(CSE 2014). 2014:1819-1826.

[19] MERKLE R C. A Digital Signature Based on a Conventional Encryption Function[M]// Advances in Cryptology-CRYPTO '87. Berlin:Springer,1988:369-378.

[20] WANG Y X,YANG Q,CHENG W,et al. Application of lattice-based linearly homomo-rphic signatures in cloud[J]. CHINA SCIEN-PAPER,2016,11(20):2381-2386.

[21] LIU Z,LIAO Y,YANG X,et al. Identity-Based Remote Data Integrity Checking of Cloud Storage From Lattices[C]// International Conference on Big Data Computing & Communications. 2017:128-135.

[22] YAN Y,WU L,GAO G,et al. A dynamic integrity verification scheme of cloud storage data based on lattice and Bloom filter [J]. Journal of Information Security & Applications,2018,39(C):10-18.

[23] ALWEN J,PEIKERT C. Generating Shorter Bases for Hard Random Lattices [J]. Theory of Computing Systems,2011,48(3):535-553.

[24] LIU C,CHEN J,YANG L T,et al. Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-Grained Updates[J]. IEEE Transactions on Parallel & Distributed Systems,2014,25(9):2234-2244.

[25] SOOKHAK M,YU R,ZOMAYA A. Auditing Big Data Storage in Cloud Computing Using Divide and Conquer Tables[J]. IEEE Transactions on Parallel and Distributed Systems,2017,29(5):999-1012.

[26] AGRAWAL S,BONEH D,BOYEN X. Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE. [C]// Annual Cryptology Conference. Berlin:Springer,2010:98-115.

LI Shu-quan, born in 1971, postgraduate, vice professor, master supervisor, is member of China Computer Federation (CCF). His main research interests include information security and trusted Computing.