计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2015.20140610
52(8):1862-1872,2015

云存储中一种基于格的数据完整性验证方法

谭 霜 何 力 陈志坤 贾 焰
(国防科学技术大学计算机学院 长沙 410073)
(tanshuang@nudt.edu.cn)

A Method of Provable Data Integrity Based on Lattice in Cloud Storage

Tan Shuang, He Li, Chen Zhikun, and Jia Yan
(*College of Computer, National University of Defense Technology, Changsha* 410073)

Abstract Using the cloud storage technology, users can outsource their data to the cloud. Such outsourcing meets the requirements of saving hardware costs and simplifying data management, because they no longer store any copies of the data in their local memory, and users cannot fully ensure whether the outsourced data are intact overall. Further, considering the client's constrained computing power and the large size of the outsourced data, the client cannot take the extra time and effort to verify the data correctness in cloud environment. Therefore, ensuring the integrity of the outsourced data would lead to many security threats. In order to solve this problem, in this paper, we present lattice-based provable data integrity for checking the integrity of the data in the cloud. The proposed scheme not only detects any violations of client data in the cloud, but also has been proven to be safe in a random oracle. In particular, as opposed to schemes based on factoring or discrete log, the proposed scheme resists the cryptanalysis by quantum algorithms. Moreover, the proposed protocol has three other good attributes, namely support for data dynamics, computing on signed data, and multi-client verification. Finally, we present a comparison of the existing data integrity verification mechanism, as well as some open problems of lattice-based provable data integrity.

Key words cloud storage; lattice theory; public verification; provable data integrity; homomorphic

摘 要 随着云存储技术的发展,用户可以从远程云中按需获取高质量的应用和服务,而不用担心本地的数据管理存储.由于用户在本地不再保留任何数据副本,故无法确保云中数据的完整性.为了解决这一问题,提出了一种面向于云存储环境的、基于格的数据完有性验证机制,该机制能有效地识别云存储中侵犯用户数据完整性的违规行为,且在随机预言机模型下被证明是安全的.另外,设计的协议还具有其他3种好的属性,即支持数据块级的动态操作、支持签名数据上的同态计算及支持多用户验证.最后,给出了现有的多种完整性验证机制之间的对比,以及基于格的数据完整性验证方法存在的一些问题及发展方向.

关键词 云存储;格理论;公开验证;数据完整性验证;同态

中图法分类号 TP309

伴随着大量的应用在云中部署,用户的数据也被集中收集交给云来处理.对用户而言,将数据存储到云中可以带来很多好处,如减少了硬件的投入成本、减轻了本地存储负担、支持异地访问等.然而云

收稿日期:2014-07-07;**修回日期**:2014-10-24
基金项目:国家"八六三"高技术研究发展计划基金项目(2012AA01A401,2012AA01A402);国家自然科学基金项目(61472433)

存储带来便利的同时,也带来相应的挑战.高度集中的数据及复杂的计算环境使得用户数据遭受多个方面的威胁[1].首先,相对传统的数据中心而言,云具有更复杂的系统.数目繁多的组成部件,使云遭受攻击的可能性更大.随着系统复杂度上升,脆弱性也随之增加①.其次,多租户共享云计算资源,使数据遭受损坏的风险更大.在云计算环境下,租户与租户之间定制的资源通常是通过采用逻辑的方法隔离.恶意的攻击者可能伪装成租户从内部发起攻击,侵犯其他用户的数据[2].最后,云服务提供商可能蓄意隐藏用户数据已损坏的事实,或采用离线的方式存储用户很少访问的数据.考虑到大规模的外包数据和有限的计算能力,用户如何验证外包数据的完整性成为了云存储中一个关键问题.

为了解决这一问题,已有大量用于验证远程节点上数据完整性的方法被提出来[3-23],但这些方法所采用的安全模型大多是基于大数分解和离散对数等计算难问题,而该类问题已被证明无法抵御量子攻击[24-25].因此,本文主要针对验证机制的安全模型进行研究,利用格理论中的求解小整数解计算难问题来进行建模,提出一种基于格的数据完整性验证方法,该方法能有效地抵御量子攻击[24-25].

本文的创新点:1)提出了国际第1种基于格的数据完整性验证方法,该方法能在本地无副本的情况下验证云中外包数据的完整性,且在随机寓言模型下被证明是安全的;2)验证方法支持动态操作,允许用户对存储在云中的数据进行更新;3)验证方法能支持全同态签名,允许用户或第三方对云中的签名数据作运算,而新生的数据签名直接通过参与运算数据块的签名计算可得,降低了协议的通信开销和计算代价;4)支持批处理验证,验证者能同时对多个任务发起完整性验证,提高了验证者的验证效率.

1　相关工作

随着P2P网络、分布式计算、云计算等新型计算模式出现,越来越多的人选择将数据中心和应用部署到远程平台上,与之而来的问题是无法确保外包数据的完整性.Juels等人[26]最先提出一种基于岗哨的数据可恢复证明机制,该机制不仅能识别远程节点上侵犯数据完整性的行为,而且还能恢复一部分受损数据.但该机制不支持公开验证,且只能进行有限次验证.之后,Ateniese等人[4]提出另外一种用于验证不可信节点上数据完整性验证模型,他们称之为数据持有性证明(provable data possession,PDP),该模型支持公开验证,且不受验证次数的限制;但计算代价和通信开销较大,且安全模型是基于大数分解计算难问题.综合考虑引纠错能力和公开验证,Shacham等人[13]提出了另一类完整性验证模型——数据可恢复证明,该模型将BLS短签名机制和纠错编码结合起来,使验证机制不仅能验证远程节点上的数据文件是否损坏且能恢复一部分已损坏的数据,但该机制仅能支持静态的数据且协议的安全基础是基于离散对数计算难问题.针对云计算中存储应用,Wang等人提出了一系列满足不同应用场景的验证机制,包括:保护数据隐私的验证机制[15],支持识别损坏位置的验证机制[16]及支持动态操作的验证机制等[18-27].Zhu等人[19-21,28]对混合云存储中数据完整性验证问题进行了研究,考虑通过层次散列结构可以将数据文件准确地散布到多个云中存储;而完整性验证时,通过同态响应技术将来自多个云的证据汇集成一组值供验证者判断.其他类似保护数据隐私技术的研究还包括文献[12,29].

2　预备知识

2.1　格的相关知识

标记:$\mathbb{Z}_q$表示模q的整数环,若q为素数,则$\mathbb{Z}_q$是域.对于任意函数$f(n)$而言,若它的复杂度为$O(n^{-c})$,则称该函数是可忽略的,用$negl(n)$表示;若为$O(n^c)$,则称函数$f(n)$是多项式的函数,用$poly(n)$表示.一组事件若以$1-negl(n)$概率发生,则称它发生的概率是不可能忽略的.

另外,定义可数域D上2种分布$\boldsymbol{X},\boldsymbol{Y}$的统计距离为:$1/2\sum_{d\in D}|\boldsymbol{X}(d)-\boldsymbol{Y}(d)|$.如果2种分布的统计距离是可忽略的,则称这2种分布是统计接近的.

格[30]:令$\boldsymbol{B}=(\boldsymbol{b}_1,\cdots,\boldsymbol{b}_n)\subset\mathbb{R}^n$为一组线性无关的向量,则由$\boldsymbol{B}$生成的格$\boldsymbol{\Lambda}$为

$$\boldsymbol{\Lambda}=\{\boldsymbol{Bc}=\sum_{1\leqslant i\leqslant n}c_i\boldsymbol{b}_i,\boldsymbol{c}\in\mathbb{Z}^n\}.$$

基及基的长度:若格$\boldsymbol{\Lambda}$由$\boldsymbol{B}$生成,则称$\boldsymbol{B}$为格$\boldsymbol{\Lambda}$的基.定义基的长度$\|\boldsymbol{B}\|$为$\boldsymbol{B}$中最长向量的长度,即:$\|\boldsymbol{B}\|=\max_{1\leqslant i\leqslant n}\{\boldsymbol{b}_i\}$.通过施密特正交法得到$\boldsymbol{B}$的标准正交基为

① http://www.networkcomputing.com

$$\tilde{\boldsymbol{B}} = (\tilde{\boldsymbol{b}}_1, \cdots, \tilde{\boldsymbol{b}}_n).$$

给定格 $\boldsymbol{\Lambda}$ 和它的基 $\boldsymbol{B}$，存在一个多项式算法将任意一组满秩的向量集合变换为格中的另一组满足长度约束的基.

引理 1[31]. 给定 n 维格 $\boldsymbol{\Lambda}$、格基 $\boldsymbol{B}$ 和一组满秩的格向量 $\boldsymbol{S}\subset\boldsymbol{\Lambda}$，存在一个多项式算法生成格 $\boldsymbol{\Lambda}$ 的另一组满足长度约束的格基 $\boldsymbol{T}$，即：

$$\|\tilde{\boldsymbol{T}}\| \leqslant \|\tilde{\boldsymbol{S}}\| \text{ 且 } \|\boldsymbol{T}\| \leqslant \|\boldsymbol{S}\|\sqrt{n}/2.$$

2.2 基于格的离散高斯度量

给定任意实数 $s>0$，定义 $\mathbb{R}^n$ 空间中以 $\boldsymbol{c}$ 为中心的高斯分布为

$$\forall \boldsymbol{x} \in \mathbb{R}^n, \rho_{s,\boldsymbol{c}} = \exp(-\pi\|\boldsymbol{x}-\boldsymbol{c}\|^2/s^2).$$

给定任意 $\boldsymbol{c}\in\mathbb{R}^n$，$s>0$ 及 n 维格 $\boldsymbol{\Lambda}$，定义 $\boldsymbol{\Lambda}$ 上的离散高斯分布为

$$\forall \boldsymbol{x} \in \boldsymbol{\Lambda}, D_{\boldsymbol{\Lambda},s,\boldsymbol{c}}(x) = \frac{\rho_{s,\boldsymbol{c}}(\boldsymbol{x})}{\rho_{s,\boldsymbol{c}}(\boldsymbol{\Lambda})},$$

其中，分母 $\rho_{s,\boldsymbol{c}}(\boldsymbol{\Lambda}) = \sum_{x\in\boldsymbol{\Lambda}}\rho_{s,\boldsymbol{c}}$.

平滑参数[24]：给定任意正有理数 $\varepsilon>0$，定义 n 维格 $\boldsymbol{\Lambda}$ 的平滑参数 $\eta_\varepsilon(\boldsymbol{\Lambda})$ 为满足关系 $\rho_{1/s}(\boldsymbol{\Lambda}^*/0)\leqslant\varepsilon$ 最小正有理数 s 的值.

平滑参数在基于格的密码学理论中有着重要的意义. 当 $\boldsymbol{\Lambda}$ 上的离散高斯分布中 s 的值满足 $s\geqslant\eta_\varepsilon(\boldsymbol{\Lambda})$ 时，格 $\boldsymbol{\Lambda}$ 中每个陪集内的元素数目在 $D_{\boldsymbol{\Lambda},s,\boldsymbol{c}}$ 分布下大致相同. 引理 2 给出了平滑参数的上界.

引理 2[31]. 对于任意格 $\boldsymbol{\Lambda}$ 和任意一组基 $\boldsymbol{B}$，存在一个可忽略的函数 $\varepsilon(n)$，使以下关系成立：

$$\eta_\varepsilon(\boldsymbol{\Lambda}) \leqslant \|\tilde{\boldsymbol{B}}\|\omega(\sqrt{\lg n}).$$

若按离散高斯分布输出样本点，则样本点的长度将以不可忽略的概率小于 $s\sqrt{n}$.

引理 3[24]. 给定格 $\boldsymbol{\Lambda}$、格基 $\boldsymbol{T}$，令 s 为满足关系 $s\geqslant\|\tilde{\boldsymbol{T}}\|\omega(\sqrt{\lg n})$ 的正有理数，则对于任意的 $\boldsymbol{c}\in\mathbb{R}^n$，

$$Pr[\|\boldsymbol{x}-\boldsymbol{c}\| \geqslant s\sqrt{n} : \boldsymbol{x} \xleftarrow{R} D_{\boldsymbol{\Lambda},s,\boldsymbol{c}}] \leqslant negl(n).$$

引理 4[31]. $\boldsymbol{\Lambda}_1$，$\boldsymbol{\Lambda}_2$ 为 2 个 n 维的格，满足关系 $\boldsymbol{\Lambda}_1+\boldsymbol{\Lambda}_2=\mathbb{Z}^n$. 则对于任意的 $\varepsilon\in(0,1/2)$，$s\geqslant\eta_\varepsilon(\boldsymbol{\Lambda}_1\cap\boldsymbol{\Lambda}_2)$ 及 $\boldsymbol{c}\in\mathbb{R}^n$，$D_{\boldsymbol{\Lambda}_1,\sigma,\boldsymbol{c}} \bmod \boldsymbol{\Lambda}_2$ 的分布与 $\boldsymbol{\Lambda}_1+\boldsymbol{\Lambda}_2/\boldsymbol{\Lambda}_2$ 上的均匀分布之间的统计距离最多为 2ε.

引理 5. 中国剩余定理. 给定 $\mathbb{Z}^n$ 中的 2 个 n 维的格，满足关系 $\boldsymbol{\Lambda}_1+\boldsymbol{\Lambda}_2=\mathbb{Z}^n$（称 $\boldsymbol{\Lambda}_1$，$\boldsymbol{\Lambda}_2$ 互素），则有环同态：

$$\frac{\mathbb{Z}^n}{\boldsymbol{\Lambda}_1\cap\boldsymbol{\Lambda}_2} \cong \frac{\mathbb{Z}^n}{\boldsymbol{\Lambda}_1} \oplus \frac{\mathbb{Z}^n}{\boldsymbol{\Lambda}_2}.$$

2.3 复杂度假定

通常，格密码学中只考虑整数格，即格向量的每个分量都是整数. 定义 q-整数格为

$$\boldsymbol{\Lambda}_q^{\perp}(\boldsymbol{A}) = \{\boldsymbol{e}\in\mathbb{Z}^n : \boldsymbol{A}\boldsymbol{e} = 0 \bmod q, \boldsymbol{A}\in\mathbb{Z}_q^{m\times n}\},$$

$$\boldsymbol{\Lambda}_q^{\boldsymbol{u}}(\boldsymbol{A}) = \{\boldsymbol{e}\in\mathbb{Z}^n : \boldsymbol{A}\boldsymbol{e} = \boldsymbol{u} \bmod q, \boldsymbol{A}\in\mathbb{Z}_q^{m\times n}\}.$$

$\boldsymbol{\Lambda}_q^{\boldsymbol{u}}$ 是 $\boldsymbol{\Lambda}_q^{\perp}$ 的陪集中一个向量，即

$$\boldsymbol{\Lambda}_q^{\boldsymbol{u}}(\boldsymbol{A}) = \boldsymbol{\Lambda}_q^{\perp}(\boldsymbol{A}) + \boldsymbol{t}.$$

基于格 $\boldsymbol{\Lambda}_q^{\perp}(\boldsymbol{A})$，$\boldsymbol{\Lambda}_q^{\boldsymbol{u}}(\boldsymbol{A})$ 可以构造出格上的 2 类计算难问题：小整数解问题（small integer solution, SIS）和非齐次小整数解问题（inhomogeneous small integer solution, ISIS）.

定义 1[32]. 小整数解问题. 给定整数 q、矩阵 $\boldsymbol{A}\in\mathbb{Z}_q^{m\times n}$ 及实数 β，找到一组非零解 $\boldsymbol{e}\in\mathbb{Z}^n$，使得齐次方程 $\boldsymbol{A}\boldsymbol{e}=0 \bmod q$ 成立，其中 $\boldsymbol{e}$ 满足条件 $\|\boldsymbol{e}\|_2\leqslant\beta$.

基于 SIS 问题，可以推出另一类计算难问题，即非齐次小整数解问题.

定义 2[32]. 非齐次小整数解问题. 给定整数 q，矩阵 $\boldsymbol{A}\in\mathbb{Z}_q^{m\times n}$，伴随子 $\boldsymbol{u}\in\mathbb{Z}_q^n$ 及实数 β，找到一组非零解 $\boldsymbol{e}\in\mathbb{Z}^m$，使齐次方程 $\boldsymbol{A}\boldsymbol{e}=\boldsymbol{u} \bmod q$ 成立，其中 $\boldsymbol{e}$ 满足条件 $\|\boldsymbol{e}\|_2\leqslant\beta$.

定理 1[32-33]. 对于任何多项式级别的 β，$m=poly(n)$，任何满足 $q\geqslant\beta\omega(\sqrt{n\lg n})$ 的素数，在一组确定的因子 $\gamma=\beta O(\sqrt{n})$、一般复杂度问题 $SIS_{q,m,\beta}$ 和 $ISIS_{q,m,\beta}$ 可以规约到格上最坏复杂度问题 SIVP.

3 系统模型及安全模型

3.1 系统模型

云存储体系结构如图 1 所示，由用户（client）、云服务提供商（cloud server provider, CSP）及第三方审计（the third party auditor, TPA）组成. 用户拥有大量的数据需要保存到云中；云存储服务提供商

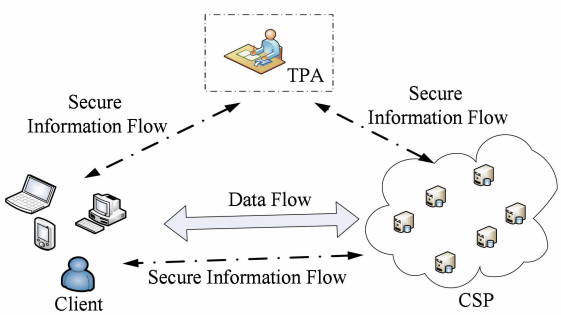


Fig. 1 The architecture in cloud storage.

图 1 云存储体系结构

为用户提供数据存储或计算服务,具有较强的计算能力和存储空间;第三方审计代替用户对其存储在云中的数据执行完整性审计任务,拥有丰富的审计经验.

用户将数据存储到云中,可以减轻本地存储负担和维护开销.但由于可能在本地并没有保存任何数据副本,采用云存储的最关键问题是如何确保数据在整个数据生命周期内都是正确的.本文假定 TPA 是无偏歧的、可信的实体,而 CSP 是半可信的实体.

3.2　安全模型

为了持续确保云中数据的完整性,验证者须周期性地对存储在云中的数据发起完整性验证.根据已有的研究[13,15,18],定义基于格的数据完整性验证方法的安全模型为:在验证过程中,不存在一个多项式对手能让验证者以不可忽略的概率接受他生成的证据.

本文对基于格的数据完整性验证方法的安全模型形式化描述如下:敌手(adversary)扮演着证明者(prover)的角色,他能访问用户存储在云中的数据信息$\{F,\Phi\}$,用符号 $\mathcal{A}$ 表示;挑战者(challenge)扮演者验证者(verifer)的角色,拥有用户数据文件的辅助验证信息,用符号 $\mathcal{B}$ 表示.游戏基于以下假定:若敌手 $\mathcal{A}$ 能在多项式时间范围内以不可忽略的概率产生一组捏造的证据 $\mathcal{P}^*$ 且该证据通过了验证,则挑战者 $\mathcal{B}$ 在敌手 $\mathcal{A}$ 的帮助下可以解决格中的 SIS 计算难问题.

定义 3. 给定安全参数 n,对于基于格数据完整性机制 $\mathcal{S}$ 而言,如果不存在一组概率多项式敌手 $\mathcal{A}$ 以不可忽略的概率赢得以下游戏,则认为 $\mathcal{S}$ 是安全的.

初始化($Setup$):挑战者调用算法 $KeyGen(\cdot)$ 获得公私密钥对(sk,pk),并将发送给敌手 $\mathcal{A}$.

查询($Queries$):敌手 $\mathcal{A}$ 指定数据文件 F 及数据分块集合$\{m_1,\cdots,m_k\}$.对于每个数据块 m_i,挑战者随机选择 $\tau_i\in\{0,1\}^*$ 作为数据块的唯一标识,调用 $\sigma_i\leftarrow Sign(sk,\tau_i,m_i)$,组成验证元信息 $\Phi=\{\sigma_1,\cdots,\sigma_k\}$并将其发送给敌手 $\mathcal{A}$.

输出($Output$):对于指定的文件 F 和挑战请求 $chal$,$\mathcal{A}$ 输出证据 $\mathcal{P}^*$.如果返回的证据 $\mathcal{P}^*$ 通过了验证等式 $VerifyProof(pk,\tau,chal,\mathcal{P}^*)=1$,则认为敌手 $\mathcal{A}$ 获胜.

敌手获胜的优势(advantage)定义为他获胜的概率.

4　协议实现

4.1　基本原理

根据文献[13,15,34]可知,数据完整性验证协议通常分成初始化和审计 2 个阶段:1)在初始化阶段,用户首先对文件进行分块,$F=\{m_1,\cdots,m_k\}$;然后,利用有效率的同态签名机制为每个分块生成块签名 σ_i,组成验证元数据集合 $\Phi=\{\sigma_1,\cdots,\sigma_k\}$;之后,将数据文件 F 及 Φ 一起存储到云中,本地仅需保存少许私钥信息.另外,为了辅助 TPA 代替用户完成完整性检测,用户需要将公钥及数据文件信息(主要是文件名与数据分块数目)发送给 TPA.2)在审计阶段,TPA 首先向云存储服务器发送挑战请求 $chal$;之后,云服务器根据请求 $chal$ 计算相应的证据;最后,TPA 根据证据判断存储在云中的文件是否是完整的.

具体而言,协议引入 2 个格 $\boldsymbol{\Lambda}_1,\boldsymbol{\Lambda}_2$,满足关系:$\boldsymbol{\Lambda}_1+\boldsymbol{\Lambda}_2=\mathbb{Z}^n$.定义格上的 2 种同态映射:

$$\varphi_1:\mathbb{Z}^n\rightarrow\mathbb{Z}^n/\boldsymbol{\Lambda}_2\cong\mathbb{Z}_p^n,\boldsymbol{x}\mapsto\boldsymbol{x}\bmod\boldsymbol{\Lambda}_1;$$

$$\varphi_2:\mathbb{Z}^n\rightarrow\mathbb{Z}^n/\boldsymbol{\Lambda}_2\cong\mathbb{Z}_q^m,\boldsymbol{x}\mapsto\boldsymbol{x}\bmod\boldsymbol{\Lambda}_2.$$

其中,p 和 q 为 2 个不同的素数.在初始化阶段,用户需要为文件 $F=\{m_1,\cdots,m_k\}$生成验证元数据集合 $\Phi=\{\sigma_1,\cdots,\sigma_k\}$.对于任意分块 m_i,包含有数据值和数据块索引 $\tau\|i$,其中数据块值 m_i 确定了格 $\boldsymbol{\Lambda}_1$ 的一个陪集(实例化 φ_1 为 $\boldsymbol{x}\bmod p$,将数据块值映射到格 $\boldsymbol{\Lambda}_1$ 上),而数据块标签 $\tau\|i$ 确定格 $\boldsymbol{\Lambda}_2$ 的一个陪集(实例化 φ_2 为 $\boldsymbol{Ax}\bmod q$,将数据块标识的散列值映射到格 $\boldsymbol{\Lambda}_2$).通过中国剩余定理,可以确定由 m_i 与 $\tau\|i$ 共同确定的格 $\boldsymbol{\Lambda}_1\cap\boldsymbol{\Lambda}_2$ 中的一个陪集 $\mathcal{C}$.最后,利用格 $\boldsymbol{\Lambda}_1\cap\boldsymbol{\Lambda}_2$ 中的一组短基 $\boldsymbol{T}$ 计算陪集 $\mathcal{C}$ 中一组较短向量作为数据块 m_i 的签名 σ_i[32-33].因此,每个 σ_i 满足以下关系:$\sigma_i\bmod\boldsymbol{\Lambda}_1=m_i$ 且 $\sigma_i\bmod\boldsymbol{\Lambda}_2=H(\tau\|i)$.

验证时,TPA 首先解析审计辅助信息$\{\tau,k,H,\boldsymbol{\Lambda}_1,\boldsymbol{\Lambda}_2\}$.然后,从分块索引$[1,k]$中,随机抽取 c 个块索引作为挑战索引 $I=\{i_1,\cdots,i_c\}$.挑战索引 I 指出了云中文件 F 中待验证的数据块位置.对于每个块索引 $i\in I$,随机选取 $v_i\in(-p/2,p/2]$作为挑战权重.之后,将块索引和挑战权重两者结合起来组成挑战请求 $chal=\{i,v_i\}_{i\in I}$.最后,TPA 将挑战请求 $chal$ 发送给云服务器.云服务器接到挑战请求 $chal$ 后,调用算法 $GenProof(\cdot)$计算完整性证据 $\mathcal{P}$.证据 $\mathcal{P}$ 由$\{\mu,\sigma\}$组成:1)计算数据块值的线性组合值 μ,

防止云服务器对存储内容的篡改,这里$\mu=\sum_{i\in I}v_i m_i\in\mathbb{Z}_p^n$;2)计算数据块标签的线性组合值$\sigma$,防止云服务器对验证元数据的篡改,这里$\sigma=\sum_{i\in I}v_i\sigma_i\in\mathbb{Z}^n$. 云服务器将证据$P=\{\mu,\sigma\}$发送给 TPA.

收到证据$P=\{\mu,\sigma\}$后,TPA 首先计算挑战集合中块索引的标识,$\alpha_{i\in I}=H(\tau\|i)$;然后,根据下式可以判断存储在云中的数据是否是完整的.

$$\begin{cases}\sigma \bmod \boldsymbol{\Lambda}_1=\sum_{i\in I}v_i m_i=\mu,\\ \sigma \bmod \boldsymbol{\Lambda}_2=\sum_{i\in I}v_i H(\tau\|i)=\sum_{i\in I}v_i\alpha_i.\end{cases}$$

4.2 具体实现

定义基于格的门限函数为$f_{\boldsymbol{A}}(\boldsymbol{e})=\boldsymbol{A}\boldsymbol{e} \bmod q$,其中$\boldsymbol{A}$定义了$\boldsymbol{\Lambda}_q^{\perp}(\boldsymbol{A})$,$\boldsymbol{e}\in\mathbb{Z}^n$为$\boldsymbol{\Lambda}_q^u(\boldsymbol{A})$中的点向量,$q$为$poly(n)$素数. 当$\boldsymbol{e}$服从$D_{\mathbb{Z}^m,s}$分布时,门限函数$f_{\boldsymbol{A}}(\cdot)$输出$\mathbb{Z}_q^m$上均匀分布的值,通常用$\boldsymbol{u}$表示. 给定一个$\boldsymbol{u}\in\mathbb{Z}_q^m$,一定存在一个相对应的值$\boldsymbol{e}\in\mathbb{Z}^m$,满足等式$\boldsymbol{u}=\boldsymbol{A}\boldsymbol{e} \bmod q$[32]. 另外,令$\boldsymbol{t}$为等式$\boldsymbol{u}=\boldsymbol{A}\boldsymbol{x} \bmod q$的任意一组解,则对于同样满足等式$\boldsymbol{u}=\boldsymbol{A}\boldsymbol{e} \bmod q$的$\boldsymbol{e}$的条件分布服从$\boldsymbol{t}+D_{\boldsymbol{\Lambda}^{\perp},s,-\boldsymbol{t}}$. 令$\boldsymbol{T}$为格$\boldsymbol{\Lambda}^{\perp}(\boldsymbol{A})$的一组优质基,作为门限函数$f_{\boldsymbol{A}}(\boldsymbol{e})$的门限知识. 在门限知识$\boldsymbol{T}$帮助下,给定$\boldsymbol{u}\in\mathbb{Z}_q^m$,可以生成一组小的整数解$\boldsymbol{e}$,使得等式$\boldsymbol{u}=\boldsymbol{A}\boldsymbol{e} \bmod q$成立. 反之,等价于求解 ISIS 计算难问题. 此方法也称为原象抽样算法.

定理 2[35]. 对于任何素数$q=poly(n)$和$m\geqslant 5n\lg q$,存在一组多项式时间算法$TrapGen(q,m,n)$产生满足以下要求的$\boldsymbol{A}\in\mathbb{Z}_q^{m\times n}$和$\boldsymbol{S}\subset\boldsymbol{\Lambda}^{\perp}(\boldsymbol{A})$:

1) $\boldsymbol{A}$统计接近于$\mathbb{Z}_q^{m\times n}$上的均匀分布;

2) $\|\boldsymbol{S}\|\leqslant O(m\lg q)$;

3) $\|\tilde{\boldsymbol{S}}\|\leqslant O(\sqrt{m\lg q})$.

利用定理 2 可以生成门限函数所需要的参数$\boldsymbol{A}$和$\boldsymbol{S}$. 通过引理 1 将$\boldsymbol{S}$转化为格$\boldsymbol{\Lambda}^{\perp}(\boldsymbol{A})$的一组优质基$\boldsymbol{T}$.

令参数$s\geqslant\eta_{\varepsilon}(\boldsymbol{\Lambda}^{\perp}(\boldsymbol{A}))$,定理 3 给出了门限算法$SamplePre(\cdot)$的具体实现方法. 该算法以格$\boldsymbol{\Lambda}^{\perp}(\boldsymbol{A})$、格的一组优质基$\boldsymbol{T}$及任意向量$\boldsymbol{u}$作为输入,从多个原象$f_{\boldsymbol{u}}^{-1}(\boldsymbol{e})$中按高斯分布$D_{\boldsymbol{\Lambda}^{\perp},s,-\boldsymbol{e}}$输出一组小整数样本点$\boldsymbol{e}\in\mathbb{Z}^m$. 在没有门限信息$\boldsymbol{T}$的情况下,攻击者获取$\boldsymbol{e}$的概率是可忽略的(样本点的长度小于$s\sqrt{n}$).

定理 3. 通过以下步骤构造出一个具有门限功能的原象抽样算法$SamplePre(\cdot)$,该算法通过高斯离散抽样算法$sampleD(\cdot)$从众多原象$f_{\boldsymbol{A}}^{-1}(\cdot)$中按高斯分布随机抽取长度满足$\|\boldsymbol{e}\|\leqslant s\sqrt{n}$的样本点$\boldsymbol{e}$:

1) 计算等式$\boldsymbol{A}\boldsymbol{t}=\boldsymbol{u} \bmod q$任意一个解向量$\boldsymbol{t}\in\mathbb{Z}^n$;

2) 利用算法$sampleD(\boldsymbol{T},s,-\boldsymbol{t})$按随机分布抽取一个样本点$\boldsymbol{v}\sim\boldsymbol{D}_{\boldsymbol{\Lambda}^{\perp},s,-\boldsymbol{t}}$,并输出$\boldsymbol{e}=\boldsymbol{t}+\boldsymbol{v}$.

根据引理 3 可知,当抽取的样本点服从分布$\boldsymbol{v}\sim D_{\boldsymbol{\Lambda}^{\perp}},s,-\boldsymbol{t}$时,$\boldsymbol{e}$的长度满足关系:$\|\boldsymbol{e}\|=\|\boldsymbol{v}-(-\boldsymbol{t})\|=\|\boldsymbol{v}+\boldsymbol{t}\|\leqslant s\sqrt{n}$. 附录 A 和附录 B 给出了基于格的离散高斯抽样算法$sampleD(\cdot)$的具体实现.

定义 4. 基于格的数据完整性验证机制由 4 个多项式时间内算法$KeyGen(\cdot)$,$Sign(\cdot)$,$GenProof(\cdot)$及$VerifyProof(\cdot)$组成,具体实现如下:

算法 1. $KeyGen(1^n)\rightarrow(sk,pk)$. 算法由用户执行,以安全参数作为输入,生成公私密钥对(sk,pk)步骤如下:

1) 选取 2 个多项式素数p,q,满足$q\geqslant(nkp)^2$. 定义$n=\lfloor m/6\lg q\rfloor$;

2) 调用算法$TrapGen(q,m,n)$生成矩阵$\boldsymbol{A}\in\mathbb{Z}_q^{m\times n}$和$\boldsymbol{\Lambda}_q^{\perp}(\boldsymbol{A})$的一组短的基$\boldsymbol{T}_q$;

3) 定义 3 个格$\boldsymbol{\Lambda}$,$\boldsymbol{\Lambda}_1$及$\boldsymbol{\Lambda}_2$,其中$\boldsymbol{\Lambda}=\boldsymbol{\Lambda}_1\cap\boldsymbol{\Lambda}_2$,$\boldsymbol{\Lambda}_1=p\mathbb{Z}^n$,$\boldsymbol{\Lambda}_2=\boldsymbol{\Lambda}_q^{\perp}(\boldsymbol{A})$. 计算$\boldsymbol{T}=p\boldsymbol{T}_q$,则$\boldsymbol{T}$为格$\boldsymbol{\Lambda}$中的一组优质基($\boldsymbol{\Lambda}=\boldsymbol{\Lambda}_1\cap\boldsymbol{\Lambda}_2=p\boldsymbol{\Lambda}_2$);

4) 令$s=p\sqrt{n\lg q}\lg n$;

5) 定义散列函数$H:\{0,1\}^*\rightarrow\mathbb{Z}_q^m$.

协议所需要的公钥和私钥为:$pk=(\boldsymbol{\Lambda},\boldsymbol{\Lambda}_1,\boldsymbol{\Lambda}_2,s,H)$,$sk=\boldsymbol{T}$.

算法 2. $Sign(sk,\tau,F)\rightarrow\Phi$. 算法由用户执行,以密钥$sk$、文件唯一标识$\tau$及数据文件$F$作为输入,采用以下方式生成验证元数据集合$\Phi$:

1) 将F分成k个块,$F=\{m_1,\cdots,m_k\}$.

2) 对每个数据块m_i,分别作以下计算:

① 计算块索引$\alpha_i=H(\tau\|i)\in\mathbb{Z}_q^m$.

② 计算任意解向量$\boldsymbol{t}_i\in\mathbb{Z}^n$,

$$\begin{cases}\boldsymbol{t}_i \bmod p=m_i;\\ \boldsymbol{A}\boldsymbol{t}_i \bmod q=\alpha_i.\end{cases}$$

③ 计算块签名

$$\sigma_i\leftarrow SamplePre(\boldsymbol{\Lambda},\boldsymbol{T},s,\boldsymbol{t}_i)\in\boldsymbol{\Lambda}+\boldsymbol{t}_i.$$

3) 生成验证元数据集合$\Phi=\{\sigma_1,\cdots,\sigma_k\}$,将数据文件$F$和$\Phi$上传至云端,删除本地的$\{F,\Phi\}$. 最后,用户将$\{\tau,n\}$发送给 TPA 保存.

算法 3. $GenProof(F,\Phi,chal)\rightarrow\mathcal{P}$. 算法由云服务器执行. 云服务器解析 TPA 发送过来的挑战

请求 $chal=\{i,v_i\}_{i\in I}$,再利用自身存储的数据$\{F,\Phi\}$计算完整性证据 $\mathcal{P}=\{\mu,\sigma\}$. 证据 μ 和 σ 的计算如下:

$$\mu=\sum_{i\in I}v_i m_i \bmod p\in\mathbb{Z}_p^n,$$

$$\sigma=\sum_{i\in I}v_i\sigma_i\in\mathbb{Z}^n.$$

最后,云服务器将证据$=\{\mu,\sigma\}$发送给 TPA.

算法 4. $VerifyProof(pk,\tau,chal,\mathcal{P})\rightarrow$"1","0". 算法由 TPA 执行,分为 2 步进行:

1) TPA 从块索引$[1,k]$中随机抽取 c 个块索引值 $I=\{s_1,\cdots,s_c\}$作为待检测的数据块. 对每个索引值 $i\in I$,随机生成一个与之相对应地的权重值 $v_i\in(-p/2,p/2]$,组成挑战请求 $chal=\{i,v_i\}_{i\in I}$,并将其发送给云存储服务器.

2) 收到云返回的证据 $\mathcal{P}=\{\mu,\sigma\}$之后,TPA 解析得到$\{\mu,\sigma\}$. 作以下计算判断云中存储的数据是否是完整的:

① $\|\sigma\|\leqslant c\frac{p}{2}s\sqrt{n}$;

② $\sigma \bmod p=\mu$;

③ 对于 $i\in I$,计算 $\alpha_i=H(\tau\|i)\in\mathbb{Z}_q^m$,判断 $\boldsymbol{A}\sigma \bmod q=\sum_{i\in I}v_i\alpha_i$ 是否成立.

如果上述条件都满足,则表明存储在云中的数据是完整的,输出"1";否则,文件已损坏,输出"0".

4.3　算法安全分析

本节将对 4.2 节所提出算法的正确性和安全性进行分析.

4.3.1　正确性

令 τ 为文件标识,$chal=\{i,v_i\}_{i\in I}$为挑战请求,返回的证据为 $\mathcal{P}=\{\mu,\sigma\}$. 对于算法 4 中步骤 2 中的 3 个验证条件分别进行验证.

1) 对于条件①,因平滑参数满足条件 $\eta_\varepsilon(\boldsymbol{\Lambda})\leqslant\|\tilde{\boldsymbol{T}}\|\omega(\sqrt{\lg n})$,所以 $s\geqslant\|\tilde{\boldsymbol{T}}\|\omega(\sqrt{\lg n})\geqslant\eta_\varepsilon(\boldsymbol{\Lambda})$. 根据定理 3 可知,数据分块的签名长度满足 $\|\sigma_i\|\leqslant s\sqrt{n}$. 而对于证据 $\mathcal{P}$ 中 σ 而言,根据三角不等式,满足以下关系:

$$\|\sigma\|=\|\sum_{s_1\leqslant i\leqslant s_c}v_i\sigma_i\|\leqslant c\frac{p}{2}\max_{s_1\leqslant i\leqslant s_c}\|\sigma_i\|\leqslant c\frac{p}{2}s\sqrt{n}.$$

2) 对于条件②,因为 $\sigma_i\in\boldsymbol{\Lambda}+\boldsymbol{t}_i$,可以推出 $\sigma=\sum_{i\in I}v_i\sigma_i=\sum_{i\in I}v_i(\boldsymbol{\Lambda}+\boldsymbol{t}_i)=\boldsymbol{\Lambda}+\sum_{i\in I}v_i\boldsymbol{t}_i$, 可知成立:

$$\sigma/p=\sum_{i\in I}c_i\boldsymbol{t}_i/p=\sum_{i\in I}c_i m_i=\mu.$$

3) 同理,对于条件③,由于 $\boldsymbol{\Lambda}\subset\boldsymbol{\Lambda}_2$,可知成立:

$$\boldsymbol{A}\sigma=\boldsymbol{A}(\boldsymbol{\Lambda}+\sum_{i\in I}c_i\boldsymbol{t}_i)=\boldsymbol{A}\cdot(\sum_{i\in I}c_i\boldsymbol{t}_i)=\sum_{i\in I}v_i\alpha_i(\bmod q).$$

4.3.2　不可伪造性

对完整性验证机制而言,若敌手无法根据数据捏造出完整性证据,则表明用户存储在云中的数据是安全的.

定理 4. 对于 $\beta=cp^2n\lg n\sqrt{\lg q}$,如果求解 $SIS_{q,n,\beta}$问题是不可行的,那么基于格的数据完整性验证协议在随机语言模型下是不可伪造的.

证明. 令 $\mathcal{A}$ 为 3.2 节中多项式时间敌手. 若存在这样的一个敌手 $\mathcal{A}$,挑战者 $\mathcal{B}$ 则可以构造出一个模拟器 $\mathcal{S}$,该模拟器能解决格 $\boldsymbol{\Lambda}_2$ 中的 $SIS_{q,n,\beta}$问题. $\mathcal{S}$ 由初始化($Setup$)、散列查询 $\mathcal{H}$ 及 $Sign$ 算法组成.

初始化 $Setup$:假定存在 3 个格 $\boldsymbol{\Lambda},\boldsymbol{\Lambda}_1,\boldsymbol{\Lambda}_2$,满足以下关系 $\boldsymbol{\Lambda}=\boldsymbol{\Lambda}_1\cap\boldsymbol{\Lambda}_2$,且 $\boldsymbol{\Lambda}_1+\boldsymbol{\Lambda}_2=\mathbb{Z}^n$,其中 $\boldsymbol{\Lambda}_1$ 为 $\mathcal{B}$ 选取满足协议要求的格,$\boldsymbol{\Lambda}_2$ 为挑战格. 令平滑参数满足条件 $s\geqslant\eta_\varepsilon(\boldsymbol{\Lambda})$,$\boldsymbol{T}_1$ 为 $\boldsymbol{\Lambda}_1$ 的基,且满足 $\|\tilde{\boldsymbol{T}}_1\|\leqslant O(\sqrt{m\lg q})$. 其他参数与真实系统类似.

散列查询 $\mathcal{H}$:任何人可以通过查询预言机 $\mathcal{H}$. 给定文件标识 τ 及数据块索引 i,如果 $\tau\|i$ 已经查询过,则返回 $H(\tau\|i)$;否则,按从分布 $D_{\mathbf{Z}^n,s}$中随机选取样本点作为数据块标签,即 $\sigma_i\leftarrow D_{\mathbf{Z}^n,s}$,并设置散列值 $H(\tau\|i)=\sigma_i \bmod \boldsymbol{\Lambda}_2$.

签名($Sign$):当敌手 $\mathcal{A}$ 查询文件 $F=\{m_i\}_{1\leqslant i\leqslant k}$ 的数据块签名 $\Phi=\{\sigma_i\}_{1\leqslant i\leqslant k}$时,作以下计算:

1) 随机选择 $\tau\in_R\{0,1\}^n$. 对于 $i\in[1,k]$,如果 $H(\tau\|i)$已经查询过 $\mathcal{H}$,则模拟器承认失败.

2) 对于每个 $i\in[1,k]$,选择 $\boldsymbol{t}_i\in m_i \bmod p$,并计算数据块的签名:

$$\sigma_i\leftarrow SamplePre(\boldsymbol{\Lambda}_1,\boldsymbol{T}_1,s,\boldsymbol{t}_i).$$

3) 对于每个 $i\in[1,k]$,定义:

$$H(\tau\|i)=\sigma_i \bmod \boldsymbol{\Lambda}_2.$$

4) 返回验证元数据集合 $\Phi=\{\sigma_1,\cdots,\sigma_k\}$给 $\mathcal{A}$.

输出($Output$):$\mathcal{B}$ 从数据块索引集合 $i\in[1,k]$ 中随机选取 c 个块索引组成挑战块的索引集合 $I=\{s_1,\cdots,s_c\}_{s_i\in[1,n]}$. 对 $i\in I$,$\mathcal{B}$ 随机选取与之相对应的权重值 $v_i\in_R(-p/2,p/2]$组成挑战请求 $chal=\{i,v_i\}_{i\in I}$,并将其发给 $\mathcal{A}$. $\mathcal{A}$ 根据 $chal$ 输出一组捏造的证据集合 P^*,这里 $P^*=\{\mu^*,\sigma^*\}$. $\mathcal{B}$ 可以查询 $\mathcal{H}$ 获取 $\tau\|i$ 的散列值. 同样地,$\mathcal{B}$ 可以调用 $Sign$ 获取数据块集合 $\Phi=\{\sigma_1,\cdots,\sigma_k\}$. 在本地计算完整性证据 $\mathcal{P}:\sigma=$

$\sum_i v_i\sigma_i, \mu = \sum_i v_i m_i$，则 $\mathcal{B}$ 输出 $\sigma^* - \sigma$ 及 $\mu^* - \mu$.

在整个模拟的过程，$\mathcal{S}$ 与真实的系统是无法区分的. 在模拟器执行 $Sign$ 的步骤 1 时，对于随机的选择文件标识 τ，其放弃的概率可以忽略不计. 另外，在 $s \geqslant \eta_\varepsilon(\boldsymbol{\Lambda})$ 条件下，根据引理 4 散列查询 $\mathcal{H}$ 的输出结果和算法 2 步骤 2 的散列值在 $\boldsymbol{\Lambda}_2$ 上都是随机均匀分布的. 而对于数据块签名 σ_i，在真实的系统中，先利用中国剩余定理求出满足特定要求的 t 值，然后计算 $\sigma_i \leftarrow D_{\boldsymbol{\Lambda}_1 \cap \boldsymbol{\Lambda}_2 + t, s}$. 对于模拟器 $\mathcal{S}$ 而言，先计算数据块签名 $\sigma_i \leftarrow D_{\boldsymbol{\Lambda}_1 + t, s}$，然后再设定散列值 $H(\tau \| i) = \sigma_i \bmod \boldsymbol{\Lambda}_2 = \alpha_i$. 因此，与真实的系统相比，生成数据分块签名的过程不一样，但两者结果的分布是一致的.

假定敌手 $\mathcal{A}$ 返回一组伪造的证据 $\mathcal{P}^*$ 给挑战者 $\mathcal{B}$，$\mathcal{B}$ 解析 $\mathcal{P}^*$ 获得 σ^* 及 μ^*. 此外，$\mathcal{B}$ 可以利用本地存储的数据块签名集合 Φ 计算证据 $\mathcal{P} = \{\mu, \sigma\}$. 对于签名聚集值 $\| \sigma^* \|$ 和 $\| \sigma \|$，两者的长度都小于 $c\frac{p}{2}s\sqrt{n}$. 因此，根据三角不等式原则，$0 \leqslant \| \sigma^* - \sigma \| \leqslant cs\sqrt{n} \leqslant \beta$. 另外，由于 $\mu \neq \mu^*$，根据算法 4 步骤 2 中②可知，$\sigma^* \neq \sigma$，即 $\sigma^* - \sigma \neq 0$. 最后，根据算法 4 步骤 2 中③可知，存在以下关系：$\boldsymbol{A}\sigma^* \bmod q = \sum_{i \in I}\alpha_i \bmod q = \boldsymbol{A}\sigma \bmod q$. 因此，$\mathcal{B}$ 可以得到一组齐次线性等式：$\boldsymbol{A}(\sigma^* - \sigma) \bmod q = 0$，即 $\mathcal{B}$ 解决了格 $\boldsymbol{\Lambda}_2$ 中 SIS 问题，与定理 1 矛盾.

由此可以推断，基于格的数据完整性验证协议是一种安全的协议. 证毕.

5 其他属性

5.1 支持动态操作

为了满足多种应用需求，用户可能频繁地对存储在云中的数据进行更新[17,23]. 因此，支持数据块级的动态操作成为了数据完整性验证机制的重要属性. 本节将对之前提出的基于格数据完整性验证机制作一定的修改，使其支持数据块级的动态操作，包括数据修改($\mathcal{M}$)、数据插入($\mathcal{I}$)、数据删除($\mathcal{D}$)及数据追加($\mathcal{X}$).

在文献[17]中，协议在生成数据块签名 σ_i 过程中考虑用数据块值 m_i 代替数据块索引 i，并引入 Merkle 散列树确保数据块在位置上的正确性. 本文采用类似的技术让基于格的数据完整性验证机制支持动态操作. 初始化阶段，计算数据块的块索引信息时需替换为 $\alpha_i = H(\tau \| m_i) \in \mathbb{Z}_q^m$；同时将块索引信息 α_i 作为 MHT 散列树的叶节点，迭代计算出根散列值 R_{MHT}，并将其作为验证元数据存储在 TPA 上. 完整性验证时，云服务器除了生成 $\{\mu, \sigma\}$ 外，还需将 $chal$ 指定数据块的散列值及辅助认证信息 $\{H(m_i), \Omega_i\}_{i \in I}$ 发送给 TPA. TPA 首先根据 $\{H(m_i), \Omega_i\}_{i \in I}$ 计算根散列值 R'_{MHT}，判断 $R_{\mathrm{MHT}} \stackrel{?}{=} R'_{\mathrm{MHT}}$ 是否相等，相等则表示数据块块在位置上是正确的，否则返回数据文件已损坏的信息. 最后，调用算法 4 继续验证数据块值是否是正确的. 图 2 给出了 2 种最常用的数据块更新操作的更新协议.

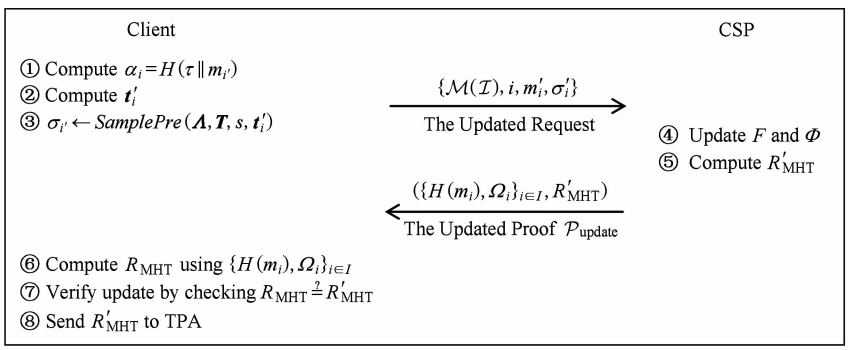


Fig. 2 The protocol for provable data update(Modification and Insertion).

图 2 数据更新协议(修改和插入操作)

5.2 支持同态操作

用户可能委托云或其他用户对其存储在云中的签名数据作计算，比如说计算数据集的均值、方差或者根据已有的数据计算新的数据信息等[31]. 在之前的签名机制[4,13,17]中，由于新生成数据的签名只能由拥有私钥的用户生成，导致每次对数据集作操作都需要用户将新生成的数据取回到客户端后再重新计算签名，给用户带来了额外的通信开销和计算代价. 本文设计的协议能有效地缓解这一问题，通过支持签名数据的同态计算，使新生成数据的签名可以

由参与操作的数据的签名直接作计算得来.如图3所示,假定用户需要对文件 F 中2个数据块 m_i,m_j 作运算 $\odot$,若采用之前的签名机制[4,17],新生成的数据块 $m(m=m_i\odot m_j)$ 的签名需由用户取回数据块后重新在客户端计算签名,而本文的协议可以直接对参与操作的数据块签名作计算,图3中数据块 m 的签名为 σ 直接通过 $\sigma_1\odot\sigma_2$ 可得.通过这样一种同态计算方式,可以减少用户和云之间的通信开销和客户端的计算代价.

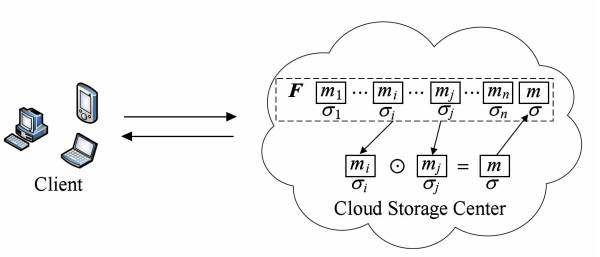


Fig. 3　Homomorphic calculation in the cloud.

图3　云中同态计算

5.3　支持批处理

批处理技术能让TPA对来自云服务提供商中的多个用户同时进行数据完整性验证,提高数据验证效率[17].假定TPA需验证云中来自 K 个不同用户的数据文件 F_k,$1\leqslant k\leqslant K$.1)每个用户调用算法1生成密钥对 $\{sk_k,pk_k\}$;2)对每个文件 F_k 进行分块,表示为 $F_k=\{m_{k1},\cdots,m_{kn}\}$;3)调用算法2为每个数据文件生成验证元数据集合,$\Phi_k=\{\sigma_{k1},\cdots,\sigma_{kn}\}$,$1\leqslant k\leqslant K$;4)用户将数据文件 F_k 和验证元数据集合 Φ_k 一起存储到云中,并删除本地的 F_k 和 Φ_k;5)将辅助验证信息 $\{\tau_k,n\}$ 发送给TPA.

挑战请求阶段,假定每个数据文件采用相同的分块数 n.TPA首先从 $[1,n]$ 中随机选取 c 个块索引作为待检测数据块的块索引 I,并随机生成 c 个随机数,组成挑战请求 $chal=\{i,v_i\}_{i\in I}$,并将挑战请求 $chal$ 发送给云服务器.

云服务器接收到挑战请求后,根据 $\{i,v_i\}_{i\in I}$ 为每个用户的数据文件生成完整性证据 $\{\mu_k,\sigma_k\}$;然后,云服务器将证据返回 K 个 $\{\mu_k,\sigma_k\}_{1\leqslant k\leqslant K}$ 给TPA.TPA接收到挑战请求后,利用式(1)计算 K 个用户的聚集证据 μ,σ.

$$\begin{aligned}\mu&=\sum_{1\leqslant k\leqslant K}\mu_k,\\ \sigma&=\sum_{1\leqslant k\leqslant K}\sigma_k.\end{aligned}\tag{1}$$

然后,验证利用式(2)同时对 K 个文件进行完整性验证:

$$\begin{cases}\sigma \bmod p=\mu,\\ [\boldsymbol{A}_1\cdots\boldsymbol{A}_K][\sigma_1\cdots\sigma_K]^{\mathrm{T}} \bmod q=\sum\limits_{1\leqslant k\leqslant K}\sum\limits_{i\in I}v_i\alpha_{ki}.\end{cases}\tag{2}$$

若满足式(2),则表明 K 个文件都是完整的;否则,至少存在一个文件是损坏的.

6　总结与开放性问题

本文提出了第3种应用于云存储环境的、基于格的数据完整性验证机制,该机制能有效地检测存储在云中损坏数据的行为,并在随机寓言机模型下证明是安全的.另外,本文设计的协议还具有其他3种好的属性:支持数据块级的动态操作、支持签名数据的同态操作及支持批处理验证.表1给出了现有10种经典的数据完整性验证机制的各种性能对比.

Table 1　Performance Comparison of Different Provable Data Integirty Protocols

表1　10种数据完整性验证机制对比

Scheme	Ref.[4]	Ref.[11]	Ref.[12]	Ref.[13]	Ref.[23]	Ref.[28]	Ref.[34]	Our's
Type of Guarantee	Pro./Det.	Det.	Det.	Pro.	Pro.	Pro.	Pro.	Pro.
Public Verification	Yes	Yes	No	Yes	Yes	Yes	Yes/No	Yes
Dynamics Operation	No	Yes	No	No	Yes	Yes	Yes/No	Yes
Homomorphic Signature	No	No	No	No	No	No	No	Yes
Batch Verification	No	No	No	No	No	No	Yes	Yes
Communication	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(\lg n)$	$O(s)$	$O(c\lg n)/O(1)$	$O(c\lg n)$
Server Block Access	$O(c)$	$O(n)$	$O(n)$	$O(c)$	$O(\lg n)$	$O(\lg n)$	$O(c\lg n)/O(c)$	$O(\lg n)$
Server Computation	$O(c)$	$O(n)$	$O(n)$	$O(cs)$	$O(c\lg n)$	$O(cs)$	$O(c\lg n)/O(c)$	$O(c\lg n)$
TPA Computation	$O(c)$	$O(n)$	$O(n)$	$O(cs)$	$O(c\lg n)$	$O(cs)$	$O(c\lg n)/O(c)$	$O(c\lg n)$
Client Storage	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$

Notes: Pro. is the probabilistic verification, Det. is the deterministic verification, n is the number of all blocks, c is the number of sampled blocks, and s is the number of sectors in a block.

开放性问题:在这个领域仍然有许多开放性的问题需要解决.本文并没有考虑采用 TPA 代替用户进行完整性审计是否会泄漏用户的数据隐私问题,根据文献[34]可知,若采用线性组合的方式生成证据,对于恶意的第三方可以有效地获取用户的数据文件和元数据验证集合.因此,基于格的数据完整性验证机制在确保数据完整性的同时能有效地保护用户数据隐私成为了一个还待解决的问题.

致 谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是国防科学技术大学计算机学院软件所 681 教研室的老师和同学们表示感谢!

参 考 文 献

[1] Ristenpart T, Tromer E, Stefan S, et al. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds [C] //Proc of the 16th ACM Conf on Computer and Communications Security. New York: ACM, 2009: 199-212

[2] Ren K, Wang C, Wang Q. Security challenges for the public cloud [J]. IEEE Internet Computing, 2012, 16(1): 69-73

[3] Ateniese G, Burns R, Curtmola R, et al. Remote data checking using provable data possession [J]. ACM Trans on Information and System Security, 2011, 14(1): 1-34

[4] Ateniese G, Burns R, Curtmola R, et al. Provable data possession at untrusted stores [C] //Proc of the 14th ACM Conf on Computer and Communications Security. New York: ACM, 2008: 598-609

[5] Ateniese G, Pietro R D, Mancini L V, et al. Scalable and efficient provable data possession [C] //Proc of the 4th Int Conf on Security and Privacy in Communication Networks. New York: ACM, 2008: 9:1-9:10

[6] Barsoum A F, Hasan M A. Integrity verification of multiple data copies over untrusted cloud servers [C] //Proc of the 12th IEEE/ACM Int Symp on Cluster, Cloud and Grid Computing. Los Alamitos, CA: IEEE Computer Society, 2012: 829-834

[7] Barsoum A F, Hasan M A. On verifying dynamic multiple data copies over cloud servers [J]. Cryptology ePrint Archive, 2011, 447: 1-30

[8] Curtmola R, Khan O, Ateniese G. MR-PDP: Multiple-replica provable data possession [C] // Proc of the 28th Int Conf on Distributed Computing Systems. Piscataway, NJ: IEEE, 2008: 411-420

[9] Deswarte Y, Quisquater J J, Saïdane A. Remote integrity checking [C] // Proc of the 6th Working Conf on Integrity and Internal Control in Information Systems. Berlin: Springer, 2004: 1-11

[10] Filho D G, Barreto P. Demonstrating data possession and uncheatable data transfer [J]. Cryptology ePrint Archive, 2006, 150: 1-19

[11] Hao Zhuo, Zhong Sheng, Yu Nenghai. A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability [J]. IEEE Trans on Knowledge and Data Engineering, 2011, 23(9): 1432-1437

[12] Sebé F. Efficient remote data possession checking in critical information infrastructures [J]. IEEE Trans on Knowledge and Data Engineering, 2008, 20(8): 1034-1038

[13] Shacham H, Waters B. Compact proofs of retrievability [C] //Proc of the 14th Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2008: 90-107

[14] Wang Boyang, Li Baochun, Li Hui. Oruta: Privacy-preserving public auditing for shared data in the cloud [C] // Proc of the 5th Int Conf on Cloud Computing. Piscataway, NJ: IEEE, 2012: 295-302

[15] Wang C, Wang Q, Ren K, et al. Privacy-preserving public auditing for data storage security in cloud computing [C] // Proc of the 29th Conf on Computer Communications. Piscataway, NJ: IEEE, 2010: 1-9

[16] Wang Cong, Wang Qian, Ren Kui, et al. Toward secure and dependable storage services in cloud computing [J]. IEEE Trans on Services Computing, 2012, 5(2): 220-232

[17] Wang Q, Wang C, Li J, et al. Enabling public verifiability and data dynamics for storage security in cloud computing [C] //Proc of the 14th European Symp on Research in Computer Security. Berlin: Springer, 2009: 355-370

[18] Wang Q, Wang C, Li J, et al. Enabling public auditability and data dynamics for storage security in cloud computing [J]. IEEE Trans on Parallel and Distributed Systems, 2011, 22(5): 847-859

[19] Zhu Yan, Hu Hongxin, Ahn G J, et al. Collaborative integrity verification in hybrid clouds [C] //Proc of the 7th Int Conf on Collaborative Computing: Networking, Applications and Worksharing. Piscataway, NJ: IEEE, 2011: 191-200

[20] Zhu Yan, Hu Zexing, Wang Huaixi, et al. A collaborative framework for privacy protection in online social networks [C] //Proc of the 6th Int Conf on Collaborative Computing: Networking, Applications and Worksharing. Piscataway, NJ: IEEE, 2010: 1-10

[21] Zhu Yan, Wang Huaixi, Hu Zexing, et al. Efficient provable data possession for hybrid clouds [C] //Proc of the 17th ACM Conf on Computer and Communications Security. New York: ACM, 2010: 756-758

[22] Zeng K. Publicly verifiable remote data integrity [C] //Proc of the 10th Int Conf on Information and Communications Security. Berlin: Springer, 2008: 419-434

[23] Erway C, Kupcu A, Papamanthou C. Dynamic provable data possession [C] //Proc of the 16th ACM Conf on Computer and Communications Security. Berlin: Springer, 2009: 213-222

[24] Micciancio D, Regev O. Worst-case to average-case reductions based on Gaussian measures [J]. SIAM Journal on Computing, 2007, 37(1): 267-302

[25] Regev O. New lattice-based cryptographic constructions [J]. Journal of the ACM, 2004, 51(6): 899-942

[26] Juels A, Kaliski B S. Pors: Proofs of retrievability for large files [C] //Proc of the 14th ACM Conf on Computer and Communications Security. Berlin: Springer, 2007: 584-597

[27] Wang Qian, Ren Kui, Yu Shucheng, et al. Dependable and secure sensor data storage with dynamic integrity assurance [J]. ACM Trans on Sensor Networks, 2011, 8(1): 1-12

[28] Zhu Yan, Hu Hongxin, Ahn G J, et al. Cooperative provable data possession for integrity verification in multicloud storage [J]. IEEE Trans on Parallel and Distributed Systems, 2012, 23(12): 2231-2244

[29] Shah M A, Swaminathan R, Baker M. Privacy-preserving audit and extraction of digital contents [J]. Cryptology ePrint Archive, 2008, 147: 1-121

[30] Micciancio D, Goldwasser S. Complexity of Lattice Problems: A Cryptographic Perspective [M]. Berlin: Springer, 2002

[31] Boneh D, Freeman D M. Homomorphic signatures for polynomial functions [C] //Proc of the 30th Annual Int Conf on Theory and Applications of Cryptographic Techniques: Advances in Cryptology. Berlin: Springer, 2011: 149-168

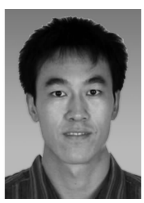
[32] Gentry C, Peikert C, Vaikuntanathan V. Trapdoors for hard lattices and new cryptographic constructions [C] //Proc of the 40th Annual ACM Symp on Theory of Computing. New York: ACM, 2008: 197-206

[33] Gentry C, Peikert C, Vaikuntanathan V. How to use a short basis: Trapdoors for hard lattices and new cryptographic constructions [J]. Cryptology ePrint Archive, 2009, 133: 1-30

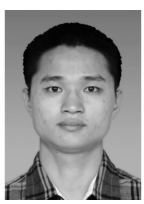
[34] Wang C, Wang Q, Ren K, et al. Privacy-preserving public auditing for data storage security in cloud computing [J]. IEEE Trans on Computers, 2013, 62(2): 362-375

[35] Ajtai M. Generating hard instances of lattice problems (extended abstract) [C] //Proc of the 28 Annual ACM Symp on Theory of computing. New York: ACM, 1996: 99-108

Tan Shuang, born in 1984. PhD. His research interests include cloud security and network security.



He Li, born in 1984. PhD. His research interests include cloud computing and data warehouse.



Chen Zhikun, born in 1984. PhD. His research interests include cloud computing and data warehouse.

Jia Yan, born in 1961. Professor at the College of Computer, National University of Defense Technology, Changsha, China. His research interests include data warehouse and cloud security.

附录 A. Babai 算法求解 CVP 计算难问题.

令 $\boldsymbol{\Lambda}\subset\mathbb{R}^n$ 为格,$\boldsymbol{v}_1,\cdots,\boldsymbol{v}_n$ 为 $\boldsymbol{\Lambda}$ 的基,且令 $\boldsymbol{w}\in\mathbb{R}^n$ 为任意向量. 如果基中的向量两两足够正交,则以下算法能解决 CVP 问题.

算法 A1. Babai 算法.

输入:$\boldsymbol{w}\in\mathbb{R}^n$ 为目标向量,$\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n$ 为格 $\boldsymbol{\Lambda}$ 的一组基;

输出:$\boldsymbol{v}\in\boldsymbol{\Lambda}$.

1) $\boldsymbol{w}=t_1\boldsymbol{v}_1+t_2\boldsymbol{v}_2+\cdots+t_n\boldsymbol{v}_n$;

2) for $i\leftarrow n,\cdots,1$ do

$a_i=\lfloor t_i\rceil$;

end for

3) $\boldsymbol{v}=a_1\boldsymbol{v}_1+\cdots+a_n\boldsymbol{v}_n$.

附录 B. 基于格的离散高斯抽样算法.

利用格 $\boldsymbol{\Lambda}$ 的基 $\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n$ 按照高斯分布从格 $\boldsymbol{\Lambda}$ 中随机抽取样本点.

1) 从$\mathbb{Z}$中按高斯分布随机抽取整数样本点.

算法 $SampleZ$ 从 1 维格$\mathbb{Z}$中按 $D_{\mathbf{Z},s,c}$ 分布随机抽取整数样本点.

算法 B1. $SampleZ(s,c)$.

输入:s 为平滑因子,c 为分布中心位置,n 为安全参数,t 为 n 的多项式函数;

输出:$x\in\mathbb{Z}$.

1) $x\leftarrow\mathbb{Z}\cap[c-s\times t,c+s\times t]$;

2) $\rho_s(x-c)$输出 x.

引理 B1. 对于 $0\leqslant\varepsilon\leqslant\exp(-\pi)$, $s\geqslant\eta_{\varepsilon}(\mathbb{Z})$ 及 $t\geqslant 0$, 有以下关系:

$$Pr_{x\leftarrow D_{\mathbf{Z},s,c}}[|x-c|\geqslant t\times s]\leqslant 2\exp^{-\pi t^2}\times\frac{1+\varepsilon}{1-\varepsilon}.$$

当 $\varepsilon\in(0,1/2)$, $t\geqslant\omega(\sqrt{\lg n})$ 时, $|x-c|\geqslant t\times s$ 的概率可以忽略不计.

2) 用格 $\boldsymbol{\Lambda}$ 的基 $\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n$ 按高斯分布抽取接近任意点 c 的样本点 $\boldsymbol{v}\in\boldsymbol{\Lambda}$.

算法 $SampleD(\boldsymbol{\Lambda},\boldsymbol{T},s,c)$, 以格 $\boldsymbol{\Lambda}$、格的基 $\boldsymbol{T}$、s 及目标向量 c 作为输入, 输出格中最接近 $\boldsymbol{c}$ 的样本点 $\boldsymbol{v}$. 算法 B2 与算法 A1 唯一的区别是, 样本点 $\boldsymbol{v}$ 用格基表示时, 系数是根据它与目标点 c 距离按概率选取的.

算法 B2. $SampleD(\boldsymbol{\Lambda},\boldsymbol{T},s,c)$.

输入: $\boldsymbol{\Lambda}$ 为任意格子、$\boldsymbol{T}$ 为门限、s 为平滑因子、c 为分布的中心位置;

输出: $\boldsymbol{v}\in\boldsymbol{\Lambda}$.

1) $c_n\leftarrow c$;

2) for $i\leftarrow n,\cdots,1$ do

$c_i'=\langle c_i,\tilde{\boldsymbol{b}}_i\rangle/\langle\tilde{\boldsymbol{b}}_i,\tilde{\boldsymbol{b}}_i\rangle$, $s_i'=s/\|\tilde{\boldsymbol{b}}_i\|$;

$z_i\leftarrow Sample\ \mathbb{Z}(s_i',c_i')$;

$c_i'\leftarrow c_i-z_i\boldsymbol{b}_i$, $\boldsymbol{v}_{i'}\leftarrow\boldsymbol{v}_i+z_i\boldsymbol{b}_i$;

end for

3) $\boldsymbol{v}\leftarrow\boldsymbol{v}_0$