云存储完整性验证密码学技术研究进展

王玉珏[1,2], 伍前红[3]

[1] 桂林电子科技大学 计算机与信息安全学院 桂林 中国 541004
[2] 新加坡管理大学 信息系统学院 新加坡 178902
[3] 北京航空航天大学 电子信息工程学院 北京 中国 100191

摘要 云存储完整性验证技术允许用户将数据存储至云端服务器, 并为用户提供可验证的完整性保证。典型的云存储完整性验证方案由两个阶段组成: 一是数据处理阶段, 用户使用私钥处理数据、生成可验证的元数据存储于云服务器, 而本地只需保存与数据相关的一些参数, 如密钥和数据标签等; 二是数据完整性验证阶段, 验证者通过和云服务器交互执行一个挑战/证明协议, 能够以极高的概率判断出云端数据当前的完整性。到目前为止, 已经涌现了大量的相关密码学方案。本文对可证明安全的可公开验证的云存储完整性验证关键密码学技术研究进展进行简要回顾, 主要涵盖代理数据外包技术、代理完整性验证技术、基于身份的数据外包技术以及几种计算和通信效率优化技术等。

关键词 云计算; 远程数据存储; 数据完整性; 数据隐私; 数据持有证明; 元数据; 数字签名
中图法分类号 TP309.2 **DOI 号** 10.19363/j.cnki.cn10-1380/tn.2017.07.003

A Survey on Cryptographic Technologies for Data Integrity Checking in Clouds

WANG Yujue[1,2], WU Qianhong[3]

[1] School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China
[2] School of Information Systems, Singapore Management University, Singapore 178902, Singapore
[3] School of Electronic and Information Engineering, Beihang University, Beijing 100191, China

Abstract Data integrity checking schemes allow users to outsource data to a cloud server, with a guarantee that the integrity of outsourced data can be verified. A typical data integrity checking scheme consists of two phases. In the data processing phase, the data owner processes her data with a private key to generate verifiable metadata that will be stored at the cloud server, and only keeps at local some related parameters including keys and data labels. In the data integrity checking phase, the verifier interacts with the cloud server to jointly carry out a challenge/prove protocol, which can detect the current integrity state of outsourced data with overwhelming probability. To date, many data integrity checking schemes have been proposed. In this paper, we review the research progress of key cryptographic technologies of publicly verifiable data integrity checking with probable security, such as proxy data outsourcing, proxy data integrity checking, identity-based data outsourcing and several efficiency optimization technologies on computation and communication costs.

Key words cloud computing; remote data storage; data integrity; data privacy; provable data possession; metadata; digital signature

1 引言

随着云计算技术的发展和普及, 越来越多的个人及企业用户已借助远程云存储平台保存数据。用户可通过按需付费的方式使用云存储服务, 而不必在本地构建硬件资源, 因此, 这种远程数据存储方式能够为用户节省大量的成本开销, 并可提供便利的数据访问和分享等功能[1,2]。尤其随着移动通信技术近年的快速发展, 用户的移动设备能够方便、快捷地接入互联网, 但普通设备仍然受存储容量的限制而无法保存实时产生的大量数据(如照片、视频等), 在这种情况下, 云存储技术恰能弥补移动设备在存储能力方面的劣势。因此, 云存储服务具有广阔的市场应用前景。实际上, 国内外众多 IT 公司(如 Microsoft、Google、Amazon、

通讯作者: 伍前红, 博士, 教授, Email: qianhong.wu@buaa.edu.cn。
本课题得到国家自然科学基金(Nos. 61672083, 61370190, 61272501, 61402029, 61472429, 61202465, 61532021)资助。
收稿日期: 2017-02-10; 修改日期: 2017-05-08; 定稿日期: 2017-05-24

IBM、Yahoo、阿里巴巴、华为、中兴等)均已涉足云计算领域, 向用户提供云存储服务。

虽然云存储拥有上述优点, 但云端数据面临的安全问题仍是用户选用这种技术的主要顾虑, 也是影响云存储技术进一步发展的一个障碍[3-4]。显然, 一旦将数据发送至云端, 用户便失去了对它们的直接控制, 同时由于本地不再保存数据的副本, 云服务器端的任何不良行为均会对用户数据的完整性造成破坏, 进而可能对用户(数据拥有者和使用者)造成不可估量的损失。比如, 服务器的硬件错误可能造成用户数据的丢失, 此外, 为节省存储空间, 云服务器也可能删除用户极少访问的部分数据或转移至其他地方存储。因此, 需要构建一种有效的机制对云存储的数据进行监管, 使用户能方便地验证其外包数据的完整性, 从而可以约束云端服务器的不良行为。

针对上述云存储数据的完整性问题, 近年来已经涌现了大量相关的重要研究成果, 从不同的角度提出了许多密码学解决方案。文献[5]首次提出了数据持有证明(Provable Data Possession, PDP)的概念, 构建了一种模型来验证云存储数据的完整性, 使用户无需从云端下载完整的数据。该模型主要分为两个过程。一是数据存储过程, 用户(数据持有者)对数据进行处理, 为每个数据块生成一个可验证的元数据(metadata), 一起发送给云服务器进行保存, 而在本地只需保留一个数据标签和用户私钥, 其中, 数据标签一般包含数据名、数据块的数量和生成元数据的公开参数等信息。二是数据审计过程, 用户(验证者)向云端提交一个随机查询挑战, 据此云服务器利用存储的数据和元数据生成一个响应并返回给用户, 最后, 用户根据云服务器的响应来判定远程数据的完整性。近年涌现的其他相关云存储数据完整性验证方案都可看做是该模型的扩展。

本文将围绕可证明安全的可公开验证的云存储数据完整性验证密码学技术几个关键研究分支进行综述。首先简要概述其安全需求、模型定义、方案分类等, 然后根据功能需求和技术的不同, 针对几个重要研究分支的进展分别进行简要阐述, 包括代理数据外包技术、代理完整性验证技术、验证时的数据隐私保护技术、基于身份的数据外包技术以及批验证等效率优化技术, 最后对本文进行简单总结同时展望下一步还有待研究的相关问题。

2 系统模型及安全需求

2.1 系统模型

基本的云存储完整性验证系统包含三个实体, 即数据拥有者、云存储服务器和验证者, 其系统模型如图 1 所示。云存储服务器是非可信的实体, 具有强大的存储能力, 在向用户提供数据存储服务的同时也将完全控制用户的外包数据。存储在云端的远程数据可能遭受不经意的丢失(如由于硬件故障)或遭受恶意篡改(如云服务器为了节省存储资源而删除部分不经常访问的用户数据)。数据拥有者和验证者都是云用户, 前者处理数据、生成元数据并上传至云端, 不要求在本地保存数据副本, 后者负责验证外包数据的完整性。该系统也要求云存储服务器具有强大的计算能力, 即在完整性验证阶段, 针对验证者发送来的查询挑战, 云服务器应能快速生成一个响应。

图 1 云存储完整性验证系统模型

一个有效的云存储完整性验证方案至少需满足以下条件:

1) 高概率的完整性验证: 当云端数据部分丢失或被篡改时, 验证者应能在不取回完整外包数据的前提下以极高概率发现该数据的当前状态。

2) 无验证次数限制: 完整性验证协议不应该限制云端数据的完整性验证次数。验证过程应是无状态的, 即验证者和云存储服务器无需保存以前的完整性验证信息。

3) 计算和通信效率: 数据处理和完整性验证阶段的计算和通信效率应尽可能高, 以适用于计算能力较弱的设备。

4) 云端存储开销: 用户生成的元数据应尽可能小, 以减少云服务器的存储开销。

5) 非交互性: 在完整性验证阶段, 当云服务器接收到验证者的挑战后, 仅利用所存储的数据和元数据即可生成一个响应; 类似的, 验证者也无需和云服务器交互即可对响应进行验证。

2.2 系统框架及安全定义

一个基本的云存储完整性验证方案由以下五个高效的算法构成:

- $KeyGen(\lambda)\rightarrow(pk, sk)$: 输入一个系统安全参数 λ, 数据拥有者执行该密钥生成算法获得一对公钥 pk 和私钥 sk 。

- $Dataproc(M, sk, pk)\rightarrow(\tau, M^*)$: 输入一个数据 M 和一对公/私钥 (sk, pk), 数据拥有者执行该数

据处理算法生成一个数据标签 τ 和处理后的数据 M^*。其中, τ 包含一个唯一的数据名 *name* 和用于处理 M 的参数, M^* 由原数据 M 和一组元数据 σ_i 构成。数据拥有者在本地保存标签 τ, 并将 M^* 发送给云存储服务器。需说明的是, 在处理数据时, 云存储验证相关方案先对数据分块, 且大部分方案会对每个数据块继续分成数据段, 然后对各个数据块 m_i 生成一个可验证的元数据 σ_i。

• $Chall(\tau, pk) \to C$: 输入一个数据标签 τ 和数据拥有者的公钥 pk, 验证者执行挑战生成算法输出一个查询挑战 C, 并发送给云存储服务器。

• $\mathrm{Pr}oofGen(C, M^*, pk) \to P$: 输入一个查询挑战 C, 一个云端的数据 M^* 和数据拥有者的公钥 pk, 云存储服务器执行证明生成算法输出一个响应 P, 并发送给验证者。

• $Verify(C, P, \tau, pk) \to 1/0$: 输入一对查询挑战 C 和响应 P, 一个数据标签 τ 和数据拥有者的公钥 pk, 验证者执行验证算法来判断云数据当前的完整性, 如保存完整则输出 1, 否则输出 0。

一个有效的云存储完整性验证方案需要满足两个条件, 即正确性和稳固性。正确性是指, 对于任意的安全参数 λ 和任意的一对公钥和私钥 $(pk, sk) \leftarrow KeyGen(\lambda)$, 对于任意数据 M 经数据处理算法生成 $(\tau, M^*) \leftarrow Dataproc(M, sk, pk)$, 其完整性都可被成功验证, 即 $Verify(C, P, \tau, pk) = 1$, 其中, $C \leftarrow Chall(\tau, pk)$ 以及 $P \leftarrow \mathrm{Pr}oofGen(C, M^*, pk)$。

稳固性由以下安全游戏进行定义[6-7]。

初始化: 挑战者 E 运行 $KeyGen(\lambda)$ 生成一对公钥 pk 和私钥 sk, 并将公钥发送给敌手 A。

询问阶段: 敌手 A 和挑战者 E 自适应地执行以下询问。

• 处理数据: 敌手 A 向挑战者 E 发送一个数据 M, 挑战者 E 运行 $Dataproc(M, sk, pk)$ 并将生成的 (τ, M^*) 返送给敌手 A。这里, 每个询问的数据都将产生一个唯一的数据名 *name*, 为保证其唯一性, 数据名由挑战者 E 选取并包含在数据标签 τ 中。

• 完整性验证: 对于任何已被上述询问处理后的数据 M^*, 挑战者 E 可与敌手 A 交互来验证其完整性, 即挑战者 E 担当验证者, 而敌手 A 担当证明者。具体地, 为验证 M^* 的完整性, 挑战者 E 向敌手 A 发送一个挑战 $C \leftarrow Chall(\tau, pk)$, 敌手 A 返回一个响应 P, 挑战者 E 运行 $Verify(C, P, \tau, pk)$ 并将验证结果告知敌手 A。

游戏结束: 敌手 A 最后输出关于某个已在处理数据阶段询问过的数据 $\hat{M}$ 的一个证明描述 $\Re$。假设数据 $\hat{M}$ 的标签为 $\hat{\tau}$。如果 $\Re$ 能够正确响应 ε 比例的完整性验证询问, 则称它是 ε 可容的。对于 ε 可容的证明描述 $\Re$, 如果存在一个高效的提取算法 $Ext(\cdot)$ 使得 $Ext(\hat{\tau}, \Re, pk, sk) = \hat{M}$, 则称该云存储完整性验证方案是 ε 稳固的。

2.3 方案分类

云存储完整性验证方案主要有两种分类方式。根据外包数据的验证方式可以分为私有验证方案和可公开验证的方案, 前者只允许持有数据拥有者私钥的用户(主要指数据拥有者自己)验证云端数据的完整性, 当验证者和数据拥有者不是同一用户时, 他们之间必然是相互信任的, 也可将他们看做同一用户; 而后者支持任何持有数据拥有者公钥的人员验证数据的完整性。可公开验证的方案适用于多用户环境下的云存储数据分享, 使得每个用户在获取云端数据的同时可验证其完整性。在设计具体方案时, 私有云存储完整性验证方案主要基于对称密码技术(如对称加密方案、伪随机函数等), 因此其数据处理和完整性验证效率较高; 由于可公开验证的方案主要基于公钥加密技术(如数字签名等), 因此计算效率相对较低。

另一种分类方式主要考虑数据拥有者是否能更新存储于云端的数据, 据此可分为静态和动态云存储完整性验证方案。数据更新主要包括数据块级别的修改、追加、插入、删除等操作。一个有效的动态方案应尽可能小的影响其他未被更新的数据块, 比如当修改某个数据块时, 理想的方案不应该对所有的数据块都重新计算元数据, 否则其计算量将相当于重新执行数据处理算法。显然, 数据更新时如果涉及到数据块数量的增加或减少, 数据标签将需重新生成。通常情况下是在静态方案之上施加额外的技术使其支持云端数据的更新操作, 比如, 文献[8]将 Merkle 哈希树(Merkle hash tree[9], MHT)与 Shacham 和 Waters[6]的静态方案相结合, 从而构造了一个动态云存储完整性验证方案。

3 典型的云存储完整性验证方案

3.1 原始 PDP 方案

在文献[5]中, Ateniese 等人设计了一个基于指数知识假设(Knowledge of Exponent Assumption, KEA-r)的 PDP 方案。在数据处理阶段, 其 *Dataproc* 算法对

每个数据块 m_i 生成一个元数据:

$$\sigma_i = \left(h(\varsigma \| i) \times g^{m_i}\right)^d \bmod N$$

其中, ς 可看做一个唯一的数据名, i 是当前数据块的编号, d 和 N 分别为 RSA 私钥和模数, g 是模 N 二次剩余集合 QR_N 的一个生成元, $h:\{0,1\}^* \to QR_N$ 为一个哈希函数。显然, 当给定 RSA 公钥 e 时, 此元数据 σ_i 是可公开验证的。注意到 σ_i 中的因子 $g^{m_i d}$ 支持聚合操作, 即多个不同的 m_i 对应的 $g^{m_i d}$ 可以聚合为 g^{md}, 其中 m 表示这些 m_i 聚合后的结果。元数据的这种定义方式将允许云存储服务器在完整性验证阶段生成一个部分聚合的响应, 从而节省验证者的计算开销。

在完整性验证阶段, 验证者发送一个挑战 $C=(c,k_1,k_2,g^z)$ 用以随机查询 c 个数据块, 这些数据块的序号由一个伪随机置换(Pseudo-random permutation, PRP) π 决定, 而所使用的随机系数由一个伪随机函数(Pseudo-random function, PRF) f 生成。这里, z 是一个随机数, 用以保证本次查询的随机性, 使得云服务器无法利用以前的查询挑战来生成满足本次完整性查询的响应。根据查询挑战 C, 云服务器针对所查询的数据块计算一个聚合数据:

$$\rho = H\left(\left(g^z\right)^{\sum_{j=1}^{c} v_j m_{i,j}} \bmod N\right)$$

和一个聚合标签:

$$\sigma = \prod_{j=1}^{c} \sigma_{i_j}^{v_j} \bmod N$$

其中, H 是一个共享的哈希函数, $i_j = \pi_{k_1}(j)$, $v_j = f_{k_2}(j)$。因此, 云服务器的响应为一个二元组 $P=(\rho,\sigma)$。为验证 P, *Verify* 算法也需要计算全部的 i_j 和 v_j, 然后判断下面的等式是否成立:

$$\rho = H\left(\left(\sigma^e \Big/ \sum_{j=1}^{c} h(\varsigma \| i_j)^{v_j}\right)^z \bmod N\right)$$

在文献[10]中, Ateniese, Kamara 和 Katz 提出了存储证明(Proofs of Storage, PoS)的概念, 并给出了一个基于同态识别协议的 PoS 通用构造。从功能上看, PoS 与 PDP 并无本质区别。

3.2 数据恢复证明

Juels 和 Kaliski[11]首次在云存储环境下提出了数据恢复证明(Proofs of Retrievability, POR), 以保证云存储数据的可恢复性。相比于 PDP 仅保证数据的完整性, POR 实现的数据可恢复性更加严格。为容忍一定比例的数据丢失或损坏, POR 采用纠错码技术对数据进行编码, 通过增加冗余来恢复数据。文献[11]选用的是 (n,k,d)-纠错码, 其中最小距离 d 是一个偶数, 意味着其纠错能力为 $d/2$。其数据处理算法包含以下步骤:

1) 编码和加密: 将数据 M 分成 b 块(b 是 k 的倍数), 对每 k 个数据块计算纠错编码。这里生成的冗余数据也进行分块, 使之与原数据块具有相同的长度。由此得到编码后的数据为 M', 共包含 bn/k 个数据块 m_i'。然后使用对称加密技术对每个数据块 m_i' 加密得到 m_i''。

2) 生成"卫士": 利用单向函数生成 η 个随机"卫士"(sentinel), "卫士"和数据块具有相同的长度, 因此可被视为伪数据块。

3) 置换: 利用随机置换算法对步骤 1 和 2 生成的(伪)数据块进行置换, 将置换后的整体数据发送给云存储服务器。

在执行完整性查询时, 验证者将验证随机"卫士"是否被完好保存, 而非直接对真实的数据块进行查询和验证, 从而判断云端数据的完整性(可恢复性)。由于共有 η 个随机"卫士", 假设每次完整性验证查询 κ 个"卫士", 则该方案只能允许有限次(η/κ 次)无交叉的完整性验证。

Shacham 和 Waters[6]首次提出了可严格证明的 POR 方案, 结合了编码技术和 PDP 技术。以下简单回顾基于计算 Diffie-Hellman 难题的 POR 方案, 然后探讨其相对于先前方案的优势。数据处理算法 *Dataproc* 先对 M 编码得到 M', 再对 M' 分块使得每个数据块包含 s 个数据段, 即 $m_i = (m_{i,1}, m_{i,2}, \cdots, m_{i,s})$。对每个数据块 m_i 计算元数据 σ_i 如下:

$$\sigma_i = \left(H(name \| i) \times \prod_{j=1}^{s} u_j^{m_{i,j}}\right)^{\alpha}$$

其中, α 是用户私钥, u_j $(1 \leqslant j \leqslant s)$是从双线性群 G 上随机选取的参数。和文献[5]类似, 此处元数据 σ_i 所包含的因子 $(\prod_{j=1}^{s} u_j^{m_{i,j}})^{\alpha}$ 也支持聚合操作, 因此在完整性验证阶段云存储服务器可生成部分聚合的响应。该算法还对数据名、数据块数量以及参数 u_j 进行签名, 得到数据标签 τ。

为验证该数据的完整性, 验证者提交一个查询挑战 $C=\{(i,v_i)\}$, 包含随机选取的数据块编号 i 和对应的系数 v_i。云服务器计算聚合的数据块

$\mu=(\mu_1,\mu_2,\cdots,\mu_s)$ 和元数据 σ 作为响应 P:

$$\mu_j=\sum_{(i,v_i)\in C} v_i m_{i,j}\ (1\leqslant j\leqslant s),\ \ \sigma=\prod_{(i,v_i)\in C}\sigma_i^{v_i}$$

验证响应 P 需要执行两次双线性对运算:

$$e(\sigma,g)=e\left(\prod_{(i,v_i)\in C} H(name\|i)^{v_i}\times\prod_{j=1}^{s}u_j^{\mu_j},\upsilon\right)$$

这里, υ 是和 α 对应的用户公钥。

上述方案中, Shacham 和 Waters[6]首次对数据进行双重分割, 这样每个数据段 $m_{i,j}$ 相当于前述方案的一个数据块。这种分割策略具有明显的优点, 通过对一组数据段生成一个元数据, 可减小处理后数据的大小, 从而降低云服务器得存储开销。基于类似的思路, 文献[6]也设计了基于 RSA 的可公开验证的 POR 方案, 和一个并基于消息认证码(Message authentication code, MAC)、PRF 和对称加密技术的私有验证 POR 方案, 并给出了严格的安全证明, 此处不再重复阐述。

4 代理数据处理和外包技术

Wang 等人[12]研究了一个多用户环境下的数据外包场景。比如在一个企业中, 雇员生成的数据只能由一个安全中介进行处理, 但中介不能知晓数据内容。为解决此问题, 文献[12]基于盲签名设计了一个 PDP 方案, 其数据处理过程由数据拥有者和安全中介共同完成。拥有者首先对数据进行预处理和盲化, 得到的数据块如下:

$$m_i'=\left(H(id_i)\times\prod_{j=1}^{s}u_j^{m_{i,j}}\right)\times g^r$$

其中, id_i 是数据块识别符, 相当于 $name\|i$, r 是用于盲化的随机元素, 其他符号同 3.2 节的方案。 安全中介对 m_i' 进行签名得到 $\sigma_i'=(m_i')^{\alpha}$, 由于 m_i' 是盲化后的数据, 此中介无法得到原数据内容 $m_{i,j}$。数据拥有者收到 σ_i' 后对其脱盲, 从而获得 $m_{i,j}$ 的元数据, 即 $\sigma_i=\sigma_i'\times y^{-r}$。完整性验证过程同 Shacham 和 Waters[6]的方案。

针对数据拥有者需委托他人处理数据的应用场景, 文献[13]设计了代理数据处理方案。比如在电子医疗系统[14,15]中, 患者(数据拥有者)可以授权医生(代理)处理其病历并上传至数据中心。Wang 等人[13]的方案解决了两个问题, 一是授权的安全问题, 即授权不能被代理伪造或滥用; 二是数据的综合审计, 即验证者不但可以验证数据的完整性, 而且能验证外包数据的来源等信息。

从技术角度说, 每个用户 ID_i 的私钥 $sk_i=(sk_{i,1},sk_{i,2})$ 由一个安全中心颁发, sk_i 可看做利用 Paterson 和 Schuldt 的方案[16]对 ID_i 的签名。为授权某个代理处理数据, 拥有者首先构造一个凭据 W, 可包含数据拥有者和代理的身份、授权的有效期和数据类型等信息, 然后对 W 签名生成授权 σ_w。在处理数据时, 授权的代理可利用其私钥 $sk_x=(sk_{x,1},sk_{x,2})$ 对数据进行分块, 对每个数据块 m_i 计算元数据如下:

$$\sigma_i=sk_{x,1}\times\left(H(W\|name\|i)\times\prod_{j=1}^{s}u_j^{m_{i,j}}\right)^{t_m}$$

其中, t_m 为一个和数据 M 相关的随机元素。在综合验证阶段, 除了数据的完整性, 验证者还可验证授权 σ_w 的有效性, 以及数据的来源信息是否和 W 保持一致。

文献[17]研究了一个类似场景, 使得一个授权的代理可以代表拥有者处理并外包数据。与文献[13]的技术相比, 代理不能利用自己的私钥 $sk_x=(sk_{x,1},sk_{x,2})$ 直接处理数据, 而需生成一个专门的代理数据处理密钥 k: 数据拥有者构造一个凭据 W, 并对其签名生成授权 $\sigma_w=(R_1,k_1)$, 代理进而计算 $k=k_1+sk_{x,2}H(W\|R_1)$。为处理数据, 代理需计算参数 $u=h(n+1\|ID_o)$, 而非从群 G 上随机选取, 这里 ID_o 表示数据拥有者的身份。然后, 代理为每个数据块 m_i 计算元数据如下:

$$\sigma_i=\left(h(N_i\|i)\times u^{m_i}\right)^k$$

其中, N_i 包含数据块的名称和相关属性。上述参数 u 的产生方式使得代理无需生成数据标签, 但同时使得方案不能支持数据块的增加和删除。在完整性验证阶段, 云存储服务器需保留所有的查询挑战, 使得验证者不能以同样的参数查询相同的数据块。

5 代理完整性验证技术

在可公开验证的云存储验证方案中, 完整性验证过程均要求验证者执行较繁重的模指数运算。此外, 为确保外包数据的完整性, 一般需要周期性的执行完整性验证过程。因此, 一种可行的方法是将周期性的完整性验证任务委托给一个可信第三方(Third party auditor, TPA)或代理, 从而减轻数据拥有者的计算负担。

5.1 代理验证技术

文献[18]首次引入 TPA 代表数据拥有者执行完

整性验证过程。一个重要的问题是, TPA 的功能应被限制在执行完整性验证, 因此需确保其不能获得或恢复云端的数据, 即完整性验证不能向 TPA 泄露数据内容。这样, 云存储服务器在收到查询挑战 C 后, 其计算的 μ (该文[18]假设每个数据块只有一个分段)不能直接和 σ 一起返回给 TPA 验证。根据 Wang 等人[18]的方案, 云存储器对 μ 进行随机化处理, 即选取一个随机数 $r \in Z_p^*$, 计算 $R = e(u,v)^r$, $\gamma = H(R)$ 和 $\mu' = r + \gamma\mu$, 其中 $e(u,\upsilon)$ 为公开密钥, 然后将响应 $P = (\sigma, R, \mu')$ 发送给 TPA。TPA 计算 γ 并验证下式是否成立:

$$R \times e(\sigma, g)^{\gamma} = e\left(\left(\prod_{(i,v_i)\in C} H(name\|i)^{v_i}\right)^{\gamma} \times u^{\mu'}, \upsilon\right)$$

从而判断云端数据当前的完整性。在上述完整性验证过程中, 数据拥有者和 TPA 之间并无严格、可证明的授权机制。

Wang[19]研究了可授权的代理完整性验证, 数据拥有者向某代理颁发一对凭据 W 和授权 σ_w, 从而允许该代理验证云存储数据的完整性。假设 (x, X), (y, Y), (z, Z) 分别为数据拥有者、云服务器和代理的密钥对。数据处理算法计算 $t = H_1(e(Y,Z)^x \| W)$, 并对每个数据块 m_i 计算元数据为 $\sigma_i = (H_2(t\|i) \times u^{m_i})^x$。在完整性验证阶段, 云存储服务器生成响应 P 的方式和前文其他方案类似, 根据 $e(Y,Z)^x$ 中 x, y, z 的可交换性, 代理利用其私钥 z 可计算出 $t = H_1(e(X,Y)^z \| W)$, 进而验证如下等式是否成立:

$$e(\sigma, g) = e\left(\prod_{(i,v_i)\in C} H_2(t\|i)^{v_i} \times u^{\mu}, X\right)$$

上述完整性验证为一个简化版本, 原文[19]使用了 PRP 选取查询的数据块, 并使用 PRF 生成查询系数 v_i, 该方式降低了通信量, 但增加了 TPA 和云服务器的计算量。

文献[20]设计的代理完整性验证方案无需使用一对凭据和授权, 而是由数据拥有者向云存储服务器颁发一个代理密钥 $dk_{o\to x}$。数据拥有者的密钥组包含私钥 sk_o, 公钥 pk_o 和密钥令牌 $kt_o = H_2(pk_o)^{sk_o}$, 类似地, 代理也持有密钥组(sk_x, pk_x, kt_x)。为生成代理密钥 $dk_{o\to x}$, 代理先将其密钥令牌 kt_x 发送给数据拥有者, 后者计算 $dk_{o\to x} = kt_x^{1/sk_o}$ 并发送给云服务器。在处理数据时, 生成的元数据与 Shacham 和 Waters[6]的方案相同, 但此处每个数据块只包含一个分段。

该方案完整性验证阶段的计算和通信复杂度都高于 Wang 的方案[19]。代理选取一个查询挑战 $C = (C_1, C_2, C_3)$, 其中 C_1 包含为每个数据块选取的随机系数 v_i, $C_2 = u^r$, $C_3 = H_3(C_1)^r$, 这里 r 是 Z_p^* 上的一个随机元素。云存储服务器据此计算一个响应 $P = (\rho, P_1, P_2, P_3, P_4)$, 其中:

$$t \in_R Z_p^*, \rho = e\left(\prod_{i=1}^{n} \sigma_i^{v_i}, dk_{o\to x}\right)^t, \quad P_1 = C_2^{\sum_{i=1}^{n} v_i m_i},$$

$$P_2 = C_3^{\sum_{i=1}^{n} v_i m_i}, \quad P_3 = H_2(pk_x)^t, \quad P_4 = g^t$$

收到响应 P 之后, 代理执行 Pr*oofGen* 算法:

$$\rho^s = e\left(\left(\prod_{i=1}^{n} H_1(name\|i)^{v_i}\right)^r \times P_1, P_3\right)^{sk_x},$$

$$e(P_1, H_3(C_1)) = e(u, P_2),$$

$$e(P_3, g) = e(H_2(pk_x), P_4)。$$

显然, 该验证过程要求代理计算($n+3$)次模指数和 5 次双线性对运算。

Chen 等人[21]利用同态消息认证码(Homomorphic message authentication code, HomMac)设计了标准模型下的(代理)云存储完整性验证方案。HomMac 技术最初由 Agrawal 和 Boneh[22]提出并用于构造安全网络编码, 以保护网络中中继数据的完整性, 近年来被用于设计静态的[21]、动态的[23]以及多用户环境下[24]的云存储完整性验证方案。根据 Chen 等人[21]的私有验证方案, 假设 $f: Z \times \mathrm{K}_{prf} \to Z_p$ 是一个 PRF, $\kappa: \mathrm{K}_{prg} \to Z_p^{s+n}$ 是一个伪随机数生成器(Pseudo-random generator, PRG), 令 k_1 和 k_2 分别为 f 和 κ 的密钥, k_1 和 k_2 共同构成了数据拥有者的私钥。对于第 i 个数据块 $m_i = (m_{i,1}, m_{i,2}, \cdots, m_{i,s})$, 数据拥有者调用 HomMac 方案的签名算法计算其元数据, 即

$$\sigma_i = \sum_{j=1}^{s} m_{i,j}\delta_j + b_i \bmod p,$$

其中, $\delta = (\delta_1, \cdots, \delta_{s+n}) \leftarrow \kappa(k_2)$, $b_i \leftarrow f(i, k_1)$。由于 σ_i 支持同态操作, 在代理完整性验证阶段, 云存储服务器可对聚合后的数据块和 σ 进行随机的线性操作, 使得验证者能够执行 HomMac 方案的 Verify 算法验证响应 P 的正确性, 但无法获知所审计的数据内容。Zhang 等人[25]也基于 HomMac 构造了一个私有验证的云存储完整性验证方案, 并基于域扩展技术增强了其安全性。假设 $f: (I \times Z) \times \mathrm{K}_{prf} \to Z_p$ 和 $\kappa: \mathrm{K}_{prg} \to Z_p^{s+n}$ 分别为 PRF、PRG, k_1 和 k_2 分别为它

们的密钥。数据处理算法生成的 σ_i 和上述文献[21]中的元数据相似, 但 $b_i \leftarrow f\left((id,i),k_1\right)$, 其中 id 表示数据标识符(包含数据长度等信息)。

Armknecht 等人[26]研究了如何将私有验证的 POR 方案[6]外包给一个指定的验证者。该方案要求数据拥有者和验证者各自生成一份元数据。数据拥有者对数据分块并为每个数据块计算元数据如下:

$$\sigma_i = f_{k_o}(i) + \sum_{j=1}^{s} u_j m_{i,j}$$

这里, f 为一个 PRF, k_o 和 u_j 是数据拥有者的私钥。当处理后的数据 M^* 存至云端后, 验证者从云端取回 M^*, 并使用其私钥(k_x 和 u'_j)以类似的方式计算元数据 σ'_i。验证者将处理后的数据 $\hat{M}$ 发给云服务器, 并采用零知识证明技术向数据拥有者证明 σ'_i 计算过程的正确性。在完整性验证阶段, 代理生成两个查询挑战 C_o 和 C_x, 云服务器因而需计算两个响应 P_o 和 P_x, 并签名。代理仅可验证(C_x, P_x), 而(C_o, P_o)则转发给数据拥有者验证。需指出的是, 代理仅将 P_o 转发给数据拥有者, 而 C_o 需由拥有者重构出来。重构过程建立在数据拥有者和验证者双方默认的一个协议之上, 使得 C_o 中的元素可从比特币中最新的数据块提取出来。

5.2 验证时的数据隐私保护技术

在 TPA 或代理执行云数据完整性验证时, 如何保护数据的隐私也是一个关键问题。文献[27]考虑了外包数据的不可区分性, 即 TPA 无法分辨一个数据是否被存储至云服务器。Fan 等人[27]首先对文献[8]和[18]中的方案做了分析, 发现这些方案都无法实现外包数据的不可区分性, 但文献[18]的方案可确保 TPA 无法恢复云存储的数据。Fan 等人进一步利用 Groth 和 Sahai 的凭据不可区分的知识证明技术[28]改进了 Wang 等人[8]的方案, 从而增强了云存储数据的隐私性。

Yu 等人[29]发现文献[30]提出的方案可向 TPA 泄露外包数据, 进而提出一个基于随机预言模型的改进方案, 实现了数据的零知识隐私(zero-knowledge privacy)。在改进方案中, 数据拥有者按如下方式对数据块 m_i 计算元数据:

$$\sigma_i = g^{m_i} h^{H_1(m_i \| name)} \bmod N$$

其中, N 是 RSA 模数, g 和 h 是 QR_N 的生成元。和文献[30]相比, 这里的 σ_i 增加了后半部分 $h^{H_1(m_i \| name)}$, 因此增加了计算复杂度。在完整性验证阶段, 当收到查询挑战 C 之后, 云服务器按如下方式计算响应 $P=(P_1,P_2,P_3)$:

$$\gamma_1, \gamma_2 \in Z_N,$$

$$P_1 = H_2(g^{\gamma_1} h^{\gamma_2} \| C),$$

$$P_2 = \gamma_1 - P_1 \times \left(\sum_{(i,v_i)\in C} v_i m_i \right),$$

$$P_3 = \gamma_2 - P_1 \times \left(\sum_{(i,v_i)\in C} v_i H_1(m_i \| name) \right)。$$

可以看出, 响应 P 不但和挑战 C 完全绑定, 而且也使用 γ_1 和 γ_2 做了随机化处理。收到响应 P 后, TPA 只需验证下式是否成立:

$$P_1 = H_2\left(\left(g^{P_2} h^{P_3} \prod_{(i,v_i)\in C} \sigma_i^{v_i} \bmod N \right) \| C \right)$$

上述完整性验证为简化版本, 原文[29]使用了 PRP 选取查询的数据块, 并使用 PRF 生成查询系数 v_i。

Cui, Mu 和 Au[31]研究了可抵抗相关密钥攻击[32]的云存储完整性验证方案。在相关密钥攻击下, 敌手能获知存储在硬件中的用户私钥, 从而可计算出一个相关私钥。在完整性验证阶段, 云存储服务器可利用该相关私钥使验证者相信云端的数据仍被完整保存, 但实际上可能已被修改或删除。在该攻击模型下, 文献[31]发现现有方案(如[6,8,18])均是不安全的, 进而对 Shacham 和 Waters[6]的方案进行改进, 通过将数据拥有者的公钥放入哈希函数中, 设计了一个能抵抗相关密钥攻击的云存储完整性验证方案。比如, 此时的元数据应按如下方式生成, 即公钥 g^{α} 被嵌入至每个元数据当中:

$$\sigma_i = \left(H(name \| i \| g^{\alpha}) \times \prod_{j=1}^{s} u_j^{m_{i,j}} \right)^{\alpha}$$

完整性验证等式只需做类似的变化即可。

Yu 等人[33]研究了和文献[29]相同的问题, 即如何确保完整性验证过程不会向 TPA 泄露数据内容, 实现数据的零知识隐私特性。该文设计了身份环境下的云存储完整性验证方案, 每个用户 ID_i 的私钥 $sk_i = H_1(ID_i)^{\alpha}$ 均由一个可信的密钥分发中心(key distribution center, KGC)生成, 这里 α 为 KGC 的主私钥。在数据处理算法中, 数据拥有者 ID_o 为每个数据块 m_i 计算如下的元数据:

$$\sigma_i = sk_o{}^{m_i} \times H_2(name \| i)^{\eta}$$

其中, $\eta \in Z_p^*$ 是针对数据 M 选取的随机元素。ID_o 还

需计算 $r = g^{\eta}$ 并放入 M^*。

为实现完整性验证阶段数据的零知识隐私, TPA 和云存储服务器的交互需建立在一个零知识证明系统之上。因此, 挑战生成算法 *Chall* 不但选取 $\{(i, v_i)\}$, 而且需计算 $\rho \in Z_p^*$, $c_1 = g^{\rho}$, $c_2 = e(H_1(ID_o), g^{\alpha})^{\rho}$, 以及关于($\rho$, c_1, c_2)的一个零知识证明 *pf*。云存储服务器据此计算一个响应, 包含 r 和:

$$m' = H_3\left(e\left(\prod_{(i,v_i)\in C}\sigma_i^{v_i}, c_1\right) \times c_2^{-\sum_{(i,v_i)\in C} v_i m_i}\right)$$

最后 TPA 只需验证下面的等式即可判断云端数据当前的完整性:

$$m' = H_3\left(e\left(\prod_{(i,v_i)\in C} H_2(name\|i)^{v_i}, r^{\rho}\right)\right)$$

Zhu 等人[34]设计了一个交互性的云存储完整性验证方案, 利用零知识证明系统来确保验证者无法获得云端的数据内容。和本文综述的其他方案相比, 该文的完整性验证过程有两点不同。一是完整性验证过程由云服务器触发, 而非由验证者首先生成一个随机的查询挑战 C。二是完整性验证包含三次信息传递, 而其余方案只需进行两次信息传输。根据该文算法生成的元数据具有如下形式:

$$\sigma_i = H(\xi\|\chi_i)^{\alpha} \times g^{\beta \times \sum_{j=1}^{s} \tau_j m_{i,j}}$$

其中, $\xi = H(\sum_{j=1}^{s}\tau_j \|name)$, α, β 为数据拥有者的私钥(对应的公钥为 $h_1 = h^{\alpha}, h_2 = h^{\beta}$), $\tau_j \in Z_p$ 是为数据 M 选取的随机元素(对应的公开参数为 $u_j = g^{\tau_j}$), χ_i 是为 M 构建的索引表中的元素, 可用于支持动态操作。

为发起完整性验证过程, 云存储服务器首先计算一个承诺 $T = (h_1', \pi) = (h_1^{\gamma}, e(\prod_{j=1}^{s} u_j^{\lambda_j}, h_2))$, 其中 $\gamma, \lambda_j \in Z_p^*$。如果收到的 T 是正确的, 则验证者生成一个随机查询挑战(如前文的方案)。云服务器据此计算一个响应 $P = (\sigma, \mu_1, \cdots, \mu_s)$:

$$\sigma = \left(\prod_{(i,v_i)\in C}\sigma_i^{v_i}\right)^{\gamma}, \quad \mu_j = \lambda_j + \gamma \sum_{(i,v_i)\in C} v_i m_{i,j}$$

最后验证者只需验证下面的等式, 即可判断数据的完整性:

$$\pi \times e(\sigma, h) = e\left(\prod_{(i,v_i)\in C} H(\xi\|\chi_i)^{v_i}, h_1'\right) \times e\left(\prod_{j=1}^{s} u_j^{\mu_j}, h_2\right)$$

6 基于身份的数据外包技术

在公钥环境下, 每个用户需要持有一份由可信中介颁发的公钥证书, 这增加了系统管理、维护和验证的负担。在此背景下, 许多基于身份的云存储完整性验证方案已被提出(比如, 前面章节所介绍的方案[13,17,33]), 通过使用用户的身份作为其公钥, 可消除管理公钥证书的负担。

在 Wang 等人[35]的方案中, KGC 为每个用户 ID_i 计算一个私钥 $sk_i = (sk_{i,1}, sk_{i,2})$, 其中

$$sk_{i,1} = g^{r}, \quad sk_{i,2} = r + \alpha H_1(ID_i \|sk_{i,1}) \bmod p$$

ID_i 收到密钥后, 可通过 $g^{sk_{i,2}} = sk_{i,1} \times y^{H_1(ID_i\|sk_{i,1})}$ 验证其正确性, 这里 α 和 $y = g^{\alpha}$ 分别为 KGC 的私钥和公钥。在处理数据时, 数据拥有者 ID_o 对每个数据块 m_i 计算元数据为 $\sigma_i = \left(H_2(name\|i) \times u^{m_i}\right)^{sk_{o,2}}$, 这里原方案[35]中未包含数据名 *name*。在完整性验证阶段, 当收到云存储服务器的响应 $P = (\mu, \sigma)$ 后, 验证者只需验证下面的等式, 即可判断数据的完整性:

$$e(\sigma, g) = e\left(\prod_{(i,v_i)\in C} H_2(name\|i)^{v_i} u^{\mu}, sk_{o,1} y^{H_1(ID_o\|sk_{o,1})}\right)$$

上述验证过程为简化版本, 原文[35]也利用 PRP 选取查询的数据块, 并使用 PRF 生成查询系数 v_i。可以看出, 上面的验证等式需要使用数据拥有者 ID_o 的一个私钥 $sk_{o,1}$, 因此严格来说该方案不支持公开的云存储完整性验证。

Zhang 和 Dong[36]也设计了一个基于身份的云存储验证方案, 并给出了紧的(tight)的安全证明, 其中, 云端的存储开销和验证者的计算开销都小于文献[35]的方案, 但处理数据的复杂度高于后者。在该方案中, KGC 为每个用户 ID_i 计算一个私钥 $sk_i = (sk_{i,1}, sk_{i,2})$, 其中

$$sk_{i,1} = H_1(ID_i \|0)^{\alpha}, \quad sk_{i,2} = H_1(ID_i \|1)^{\alpha}$$

在处理数据时, 数据拥有者 ID_o 对每个数据块 m_i 选取 $r_i \in Z_p^*$, 计算元数据 $\sigma_i = (\sigma_{i,1}, \sigma_{i,2})$, 其中:

$$\sigma_{i,1} = g^{r_i}, \quad \sigma_{i,2} = h^{r_i m_i} sk_{o,1}^{H_2(m_i\|ID_o\|\sigma_{i,1}\|0)} sk_{o,2}^{H_2(name\|i\|ID_o\|1)}$$

这里 g 和 h 都是群 G 的生成元。

在完整性验证阶段, 云存储服务器计算一个响应 $P = (P_1, P_2, P_3)$, 其中, $P_1 = \prod_{(i,v_i)\in C}\sigma_{i,1}^{v_i m_i}$,

$P_2=\prod_{(i,v_i)\in C}\sigma_{i,2}^{v_i}$，$P_3=\sum_{(i,v_i)\in C}v_iH_2(m_i\|ID_o\|\sigma_{i,1}\|0)$。这里，$v_i$并非直接包含在查询挑战$C$中，而是由$C$中的$\rho$计算而来，即$v_i=\rho^i \bmod p$。最后，验证者只需验证下面的等式，即可判断数据的完整性:

$$e(P_2,g)=e(h,P_1)e(H_1(ID_o\|0)^{P_3}H_1(ID_o\|1)^{\delta},y)$$

其中，$\delta=\sum_{(i,v_i)\in C}v_iH_2(name\|i\|ID_o\|1)$，$y=g^{\alpha}$为KGC的公钥。

文献[37-38]基于拟同态签名方案[38](pre-homomorphic signature)设计了面向群组的云存储完整性验证通用构造。每个群成员从可信的群管理员处获得一个与其身份相关的私钥。该方案实现了外包数据的所有权隐藏(ownership-hidden)，即使在完整性验证阶段，云服务器也无法获知数据拥有者的身份信息。作者利用Boneh–Boyen短签名[40]设计了一个方案，其中每个用户ID_i的私钥为$sk_i=g^{(\alpha+H_1(ID_i))^{-1}}$。为处理数据$M$，$ID_o$从$Z_p^*$上选取$s+1$个随机元素$r_0,\cdots,r_s$，计算公开参数$u_j=g^{r_j}$(0≤$j$≤$s$)并保存在数据标签中，并为每个数据块$m_i$计算元数据为:

$$\sigma_i=sk_o^{r_0H_2(name\|i)+\sum_{j=1}^{s}r_jm_{i,j}}$$

当数据M处理完毕之后，ID_o从本地删除r_j(0≤j≤s)。在完整性验证阶段，根据云存储服务器的响应$P=(\mu,\sigma)$，验证者通过验证下面的等式即可判断数据的完整性:

$$e\left(\sigma,y\times g^{H_1(ID_o)}\right)=e\left(u_0^{\prod_{(i,v_i)\in C}v_iH_2(name\|i)}\times\prod_{j=1}^{s}u_j^{\mu_j},g\right)$$

7 效率优化技术

在可公开验证的云存储方案中，处理数据和完整性验证阶段均要求用户执行大量的模指数运算。众所周知，在公钥密码方案中，相对于加法和乘法运算，模指数运行需消耗更多的计算资源。因此，借助于普通的计算能力较弱的电子设备(如手机等)，用户可能无法高效的处理或验证规模较大的数据。另一方面，前面章节回顾的方案在完整性验证阶段要求验证者和云存储服务器承担线性的通信开销，因此，有必要将此开销降至常量级从而提升该阶段的通信效率。许多现有文献专注于解决这些问题，本节对部分相关效率优化技术进行简要回顾。

7.1 在线、离线技术

文献[7]研究了如何提高数据处理阶段的计算效率，提出了在线/离线数据持有证明模型，其中，数据处理算法被分为离线和在线两个阶段执行。离线阶段可看做一个预计算过程，此时用户无需知晓将被处理的数据，该过程承担了数据处理算法中几乎全部的繁杂运算。对于给定的数据，利用离线阶段的预计算结果，在线阶段只需执行轻量级的运算(如模加、模乘等)，从而提升用户处理数据的实时效率。针对满足元数据可聚合性和可公开扩展性的云存储验证方案，该文[7]构造了一个半通用的转换方案，并基于文献[6,41]给出了两个具体的在线/离线方案。作为构造模块，文中定义了向量变色龙哈希函数，可以对一输入的向量计算哈希值，同时具有变色龙属性。该函数可看做变色龙哈希函数[42]的扩展，已被文献[39]用于构造同态签名方案，但文献[39]未给出安全证明。

根据文献[7]的在线/离线数据持有证明方案，离线阶段需计算一组元数据令牌:

$$\theta_i=\left(H(name\|i)\times u_1^{m_i'}\times g^{\beta_i'}\right)^{\alpha}$$

其中，m_i'和β_i'均是从Z_p'上随机选取的元素。利用这些元数据令牌和门限私钥r_j，在线阶段生成元数据$\sigma_i=(\theta_i,\beta_i)$只涉及模加和模乘运算:

$$\beta_i=m_i'r_1+\beta_i'-\textstyle\sum_{j=1}^{s}m_{i,j}r_j \bmod p$$

相比Shacham和Waters[6]的方案，这种元数据生成方式显然极大地提高了数据处理效率，同时完整性审计阶段仅增加了有限的计算开销。云存储服务器在生成响应P时，聚合的元数据σ需包含两部分，即:

$$\theta=\prod_{(i,v_i)\in C}\theta_i^{v_i}\ 和\ \beta=\sum_{(i,v_i)\in C}v_i\beta_i$$

相应地，验证者通过验证下面的等式，从而判断数据的完整性:

$$e(\theta,g)=e\left(\prod_{(i,v_i)\in C}H(name\|i)^{v_i}\cdot\textstyle\prod_{j=1}^{s}u_j^{\mu_j}\cdot g^{\beta},\upsilon\right)$$

可以看出，上述离线阶段要求预先选定数据名$name$，但此时用户并不知道所需处理的数据内容。该文[7]注意到此情形和基于身份的在线/离线加密方案类似，即加密者在离线阶段并不知晓接收者的身份信息。因此，文献[7]探讨了如何利用Lai等人[43]的技术来改进上述方案，使其允许用户在在线阶段选取数据名。

7.2 批审计技术

Wang 等人[18]探讨了如何对多个外包数据的完整性实现批验证。相对于对每个数据单独验证, 批验证方式允许验证者(或 TPA)和云存储服务器通过一轮交互即可实现对多个数据的完整性验证, 因此可以节省验证者(TPA)的计算和通信开销。假设共有 n 个用户, 每个用户已使用如下方式处理其数据 $M_\ell (1 \leqslant \ell \leqslant n)$, 即生成元数据:

$$\sigma_{\ell,i} = \left(H(name_\ell \| i) \times u_\ell^{m_{\ell,i}} \right)^{\alpha_\ell}$$

在完整性验证阶段, 云存储服务器执行以下步骤生成响应 $P = (\{\mu_\ell, \sigma_\ell : 1 \leqslant \ell \leqslant n\}, \Re)$:

• 对每个数据 M_ℓ 选择随机数据 $r_\ell \in Z_p^*$, 计算 $R_\ell = e(u_\ell, \upsilon_\ell)^{r_\ell}$;

• 计算 $\Re = \prod_{\ell=1}^{n} R_\ell$;

• 对每个数据 M_ℓ, 计算 $\gamma_\ell = H(\Re \| \upsilon_\ell)$, $\mu_\ell = \gamma_\ell \sum_{(i,v_i)\in C} v_i m_{\ell,i} + r_\ell$ 和 $\sigma_\ell = \prod_{(i,v_i)\in C} \sigma_{\ell,i}^{v_i}$。

当收到响应后, TPA 通过验证下面的等式即可判断数据的完整性:

$$\Re e(\prod_{\ell=1}^{n} \sigma_\ell^{\gamma_\ell}, g) = \prod_{\ell=1}^{n} e\left(\left(\prod_{(i,v_i)\in C} H(name_\ell \| i)^{v_i} \right)^{\gamma_\ell} u_\ell^{\mu_\ell}, \upsilon_\ell \right)$$

这里, 和单独验证 n 个数据相比, 上述验证等式左边只包含一次双线性对运算, 因此减轻了 TPA 的计算负担。

此外, 文献[36-37,41]也使用了类似的批处理技术来优化完整性验证效率, 这里不再详细阐述。

7.3 外包计算技术

Wang 等人[44]利用外包计算技术改进了可公开验证的云存储方案, 使得数据拥有者和验证者均可将模指数运算委托给一个计算服务器。从技术角度说, 该文在单计算服务器模型下设计了一个通用的模指数外包算法, 用户可同时外包多个连乘可变指数、可变底数模指数运算(variable-exponent variable-base exponentiations), 并可确保运算参数的隐私以及验证服务器计算结果的正确性。实际上, 文献[45-47]设计的可验证模指数外包算法也可用于改进云存储完整性验证方案的计算效率, 但文献[45-46]的算法需借助于两个非合谋的(non-colluding)计算服务器, 文献[47]的算法虽只利用一个计算服务器, 但无法确保外包参数的隐私性。作为特例, 下面简要介绍如何利用文献[44]的技术外包单个模指数运算 u^a。

• 用户预计算一定数量的数对 $(\alpha_i, \mu_i = g^{\alpha_i})$。

• 用户随机选取四个数对 (α_1, μ_1), …, (α_4, μ_4), 两个随机数 $b \in Z_p^*$ 和 $\chi > 2^\lambda$, 这里 λ 为安全参数。计算如下四个数据:

$c = a - b\chi \bmod p$,

$w = u / \mu_1$,

$h = u / \mu_3$,

$\theta = (\alpha_1 b - \alpha_2)\chi + (\alpha_3 c - \alpha_4) \bmod p$。

这里, χ 将使外包用户能够对计算服务器返回的结果进行验证。

• 用户随机选取三个数对 (α_5, μ_5), (α_6, μ_6), (α_7, μ_7), 然后以随机顺序向计算服务器 U 提交以下询问:

$U((\alpha_7 - \theta)/\alpha_6, \mu_6) \to A$;

$U(\theta/\alpha_5, \mu_5) \to B$;

$U(b, w) \to C$;

$U(c, h) \to D$。

• 用户验证 $A \times B = \mu_7$ 是否成立。若是, 则计算 $u^a = (\mu_2 \times C)^\chi \times B \times \mu_4 \times D$。

在上述特例中, 用户在最终计算 u^a 时还需要计算一个小指数的幂运算, 即 χ 的规模一般远小于 a。从计算效率来说, 文献[43]更加适用于外包连乘模指数运算, 如 $\prod_{j=1}^{s} u_j^{a_j}$, 此时最后一步的幂运算将由 s 个 $u_j^{a_j}$ 共同分担。

7.4 通信效率优化技术

为验证云端数据的完整性, 验证者生成的查询挑战 $C = \{(i, v_i)\}$ 一般需指明被查询的数据块编号 i, 并为每个数据块指定一个随机系数 v_i。云服务器据此生成的响应为 $P = (\mu = (\mu_1, \cdots, \mu_s), \sigma)$。因此, 完整性验证阶段的通信开销与 $|C|$ 和 s ($s \geqslant 1$)线性相关。近年的研究文献已提出两种技术以优化上述通信效率, 可降至常量级。

第一种技术利用伪随机置换选取查询的数据块, 并使用伪随机函数生成查询系数, 这样可减小查询挑战 C 的大小, 其应用可参见文献[17,19,29,35]。假设 ϕ 和 φ 分别为安全的 PRF 和 PRP, 均为系统公开参数。在生成查询挑战 C 时, 验证者或 TPA 选取所需查询的数据块数量 c ($c \leqslant n$), ϕ 和 φ 的密钥 k_1 和 k_2,

即 $C=(c,k_1,k_2)$。这样, 云存储服务器和验证者在生成/验证响应 P 时, 均可计算被查询的数据块编号 $i_j=\phi_{k_1}(j)$, 以及数据块 i_j 对应的系数 $v_j=\varphi_{k_2}(j)$, 其中, $1\leqslant j\leqslant c$。

第二种方式是采用多项式承诺技术[47]减少响应 P 的大小, 其应用可参见文献[7,37-38,41,44,49]。云存储服务器将聚合的数据块 $\mu=(\mu_1,\cdots,\mu_s)$ 看做多项式的系数, 利用多项式承诺技术计算一个对应的承诺值, 当做响应的一部分返回给验证者, 从而降低通信量。在具体执行时, 验证者选取一个随机值 r 包含在 C 中。云服务器根据 μ 构造多项式 $f(x)=\sum_{j=1}^{s}\mu_j x^{j+1}$, 计算 $\omega=f(r) \bmod p$, 然后利用多项式除法计算 $\psi=(f(x)-\omega)/(x-r)$。显然, ψ 的阶为 s。假设 ψ 的系数为 ψ_j, 云服务器计算 $\rho=\prod_{j=0}^{s}u_j^{\psi_j}$。因此, 云服务器的响应 P 只需包含 ω 和 ρ, 而无需向验证者返回聚合的数据块 μ。验证者可直接利用文献[48]的多项式承诺技术验证 P 的有效性, 从而判断云端数据的完整性。

容易看出, 若一个云存储完整性验证方案同时使用上述两种技术, 则 C 和 P 的大小都将为常量级, 即该云存储方案完整性验证阶段仅需常量级的通信开销。

8 结语

云存储技术可极大地降低用户本地的存储成本, 并易于在多用户环境下实现数据共享。但云端数据的完整性是一个不可忽视的安全问题。在此背景下, 许多基于密码学技术的云存储数据完整性验证方案被提出, 为用户数据提供了一种可验证的完整性保护机制, 从而可以有效地检测和约束云存储服务器的行为。本文从功能和技术的角度出发, 对现有的典型可公开验证方案进行了简要回顾。

虽然云存储完整性验证已经取得了大量的研究成果, 但随着云计算技术更普遍的使用以及 IT 技术的发展, 必定会出现更多的相关安全问题亟待研究。比如, 量子技术的进步已使我们看到了量子计算机商用的曙光, 这将急切需要能够抵抗量子攻击的云存储完整性验证方案, 同时具有丰富的系统功能(如本文中回顾的方案功能等)。另外, 在确保外包数据完整性的同时, 如何让云服务器基于所存储的数据向用户提供可验证的各类计算服务, 也是值得深入研究的课题。

参考文献

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “Above the clouds: A berkeley view of cloud computing”. Technical report, TR UCB/EECS-2009-28, Electrical Engineering and Computer Sciences, University of California at Berkeley, 2009.

[2] D. Feng, M. Zhang, Y. Zhang, and Z. Xu, “Study on Cloud Computing Security,” *Journal of Software*, vol. 22, no. 1, pp. 71-83 (in Chinese), 2011.
(冯登国, 张敏, 张妍, 徐震, “云计算安全研究”, *软件学报*, 2011, 22(1):71−83。)

[3] N. Yu, Z. Hao, J. Xu, W. Zhang, and C. Zhang, “Review of Cloud Computing Security,” *Chinese Journal of Electronics*, vol. 41, no. 2, pp. 371-381 (in Chinese), 2013.
(俞能海, 郝卓, 徐甲甲, 张卫明, 张驰, “云安全研究进展综述”, *电子学报*, 2013, 41(2):371-381。)

[4] D. Song, E. Shi, I. Fischer, and U. Shankar, “Cloud Data Protection for the Masses,” *IEEE Computer*, vol. 45, no. 1, pp. 39-45, 2012.

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable data possession at untrusted stores,” in Proc. of the 14th ACM Conf. *on Computer and Communications Security (CCS’07),* pp. 598–609, 2007.

[6] H. Shacham and B. Waters, “Compact proofs of retrievability,” *Journal of Cryptology,* vol. 26, no. 3, pp. 442–483, 2013.

[7] Y. Wang, Q. Wu, B. Qin, S. Tang, and W. Susilo, “Online/Offline Provable Data Possession,” IEEE Trans. *Information Forensics and Security*, vol. 12, no. 5, pp. 1182-1194, 2017.

[8] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, “Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing,” IEEE Trans. *Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847- 859, 2011.

[9] R. C. Merkle, “Protocols for Public Key Cryptosystems,” *in Proc. IEEE Symp. Security and Privacy,* pp. 122–134, 1980.

[10] G. Ateniese, S. Kamara, and J. Katz, “Proofs of Storage from Homomorphic Identification Protocols,” in Proc. the 15th Int’l Conf. *on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '09),* pp. 319-333, 2009.

[11] A. Juels and B. S. Kaliski Jr., “PORs: Proofs of Retrievability for Large Files,” in Proc. of the 14th ACM Conf. *on Computer and Communications Security (CCS’07),* pp. 584–597, 2007.

[12] B. Wang, S. S. M. Chow, M. Li, and H. Li, “Storing Shared Data on the Cloud via Security-Mediator,” in Proc. of IEEE 33rd Int’l Conf. *on Distributed Computing Systems (ICDCS’13),* pp. 124-133, 2013.

[13] Y. Wang, Q. Wu, B. Qin, W. Shi, R. H. Deng, and J. Hu, "Identity-Based Data Outsourcing with Comprehensive Auditing in Clouds," IEEE Trans. *Information Forensics and Security,* vol. 12, no. 4, pp. 940-952, 2017.

[14] J. Sun and Y. Fang, "Cross-Domain Data Sharing in Distributed Electronic Health Record Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 21, no. 6, pp. 754–764, 2010.

[15] W. Liu, J. Liu, Q. Wu, W. Susilo, H. Deng, and B. Qin, "SAKE: scalable authenticated key exchange for mobile e-health networks," *Security and Communication Networks,* vol. 9, no. 15, pp. 2754-2765, 2016.

[16] K. G. Paterson and J. C. N. Schuldt, "Efficient Identity-Based Signatures Secure in the Standard Model," in Proc. of the 11th Australasian Conf. *on Information Security and Privacy* (ACISP'06), vol. 4058, pp. 207–222, 2006.

[17] H. Wang, D. He, and S. Tang, "Identity-Based Proxy-Oriented Data Uploading and Remote Data Integrity Checking in Public Cloud," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 6, pp. 1165-1176, 2016.

[18] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," *IEEE Trans. Computers,* vol. 62, no. 2, pp. 362–275, 2013.

[19] H. Wang, "Proxy Provable Data Possession in Public Clouds," *IEEE Trans. Services Computing*, vol. 6, no. 4, pp. 551–559, 2013.

[20] S. T. Shen and W. G. Tzeng, "Delegable provable data possession for remote data in the clouds," in Proc. of the 13th Int'l Conf. *on Information and Communications Security (ICICS'11),* vol. 7043, pp. 93–111, 2011.

[21] F. Chen, T. Xiang, Y. Yang, and S. S. M. Chow, "Secure Cloud Storage Meets with Secure Network Coding," *IEEE Trans. Computer,* vol. 65, no. 6, pp. 1936-1948, 2016.

[22] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-Based Integrity for Network Coding," in Proc. of Int'l Conf. Appl. *Cryptography Netw. Security*, vol. 5536, pp. 292–305, 2009.

[23] B. Sengupta and S. Ruj, "Publicly Verifiable Secure Cloud Storage for Dynamic Data Using Secure Network Coding," *in Proc. of the 11th ACM on Asia Conf. on Computer and Communications Security,* pp. 107-118, 2016.

[24] B. Wang, B. Li, and H. Li, "Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud," *in Proc. of the 10th Int'l Conf. Appl. Cryptography Netw. Security,* vol. 7341, pp. 507–525, 2012.

[25] R. Zhang, H. Ma, Y. Lu, and Y. Li, "Provably secure cloud storage for mobile networks with less computation and smaller overhead," *Sci China Inf Sci,* vol. 60, no. 12, pp. 122104, 2017.

[26] F. Armknecht, J. M. Bohli, G. O. Karame, Z. Liu, and C. A. Reuter, "Outsourced proofs of retrievability," *in Proc. of the 2014 ACM SIGSAC Conf. on Computer and Communications Security (CCS'14),* pp. 831–843, 2014.

[27] X. Fan, G. Yang, Y. Mu, and Y. Yu, "On indistinguishability in remote data integrity checking," *The Computer Journal*, vol. 58, no. 4, pp. 823–830, 2015.

[28] J. Groth and A. Sahai, "Efficient Non-Interactive Proof Systems for Bilinear Groups," *in Proc. Advances in Cryptology-EUROCRYPT 2008,* vol. 4965, pp. 415–432, 2008.

[29] Y. Yu, M. H. Au, Y. Mu, S. Tang, J. Ren, W. Susilo, and L. Dong, "Enhanced privacy of a remote data integrity-checking protocol for secure cloud storage," *International Journal of Information Security,* vol. 14, no. 4, pp. 307–318, 2015.

[30] Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," *IEEE Trans. Knowl. Data Eng*., vol. 23, no. 9, pp. 1432–1437, 2011.

[31] H. Cui, Y. Mu, and M. H. Au, "Proof of retrievability with public verifiability resilient against related-key attacks," *IET Information Security,* vol. 9, no. 1, pp. 43–49, 2015.

[32] M. Bellare and T. Kohno, "A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications," in Proc. *Advances in Cryptology-EUROCRYPT 2003,* vol. 2656, pp. 491-506, 2003.

[33] Y. Yu, M. H. A. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Information Forensics and Security,* vol. 12, no. 4, pp. 767-778, 2017.

[34] Y. Zhu, H. Wang, Z. Hu, G. Ahn, and H. Hu, "Zero-knowledge proofs of retrievability," *Sci. China Inf. Sci.,* vol. 54, no. 8, pp. 1608–1617, 2011.

[35] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Identity-based remote data possession checking in public louds," *IET Inf. Secur.,* vol. 8, no. 2, pp. 114-121, 2014.

[36] J. Zhang and Q. Dong, "Efficient id-based public auditing for the outsourced data in cloud storage," *Information Sciences,* vol. 343-344, pp. 1–14, 2016.

[37] Y. Wang, Q. Wu, B. Qin, X. Chen, X. Huang, and J. Lou, "Ownership-hidden group-oriented proofs of storage from pre-homomorphic signatures," Peer-to-Peer Networking and Applications, doi: 10.1007/s12083-016-0530-8, https://link.springer.com/article/10.1007/s12083-016-0530-8

[38] Y. Wang, Q. Wu, B. Qin, X. Chen, X. Huang, and Y. Zhou, "Group-oriented proofs of storage," *in Proc. of the 10th ACM Symposium on Information, Computer and Communications Security (AsiaCCS'15),* pp. 73–84, 2015.

[39] D. M. Freeman, "Improved security for linearly homomorphic signatures: a generic framework," *in Proc. Public key cryptogra-*

phy–PKC 2012, vol. 7293, pp. 697-714, 2012.

[40] D. Boneh and X. Boyen, "Short signatures without random oracles and the SDH assumption in bilinear groups," *J. Cryptol.*, vol. 21, no. 2, pp. 149–177, 2008.

[41] J. Yuan and S. Yu, "PCPOR: Public and constant-cost proofs of retrievability in cloud," *Journal of Computer Security,* vol. 23, no. 3, pp.403–425, 2015.

[42] H. Krawczyk and T. Rabin, "Chameleon signatures," *in Symposium on Network and Distributed Systems Security (NDSS' 00),* pp. 143–154, 2000.

[43] J. Lai, Y. Mu, F. Guo, and W. Susilo, "Improved identity-based online/offline encryption," *in Proc. of the 20th Australasian Conf. on Information Security and Privacy (ACISP'15),* vol. 9144, pp. 160–173, 2015.

[44] Y. Wang, Q. Wu, D. S. Wong, B. Qin, S. S. M. Chow, Z. Liu, and X. Tan, "Securely outsourcing exponentiations with single untrusted program for cloud storage," *in Proc. Computer Security–ESORICS 2014,* vol. 8712, pp. 326–343, 2014.

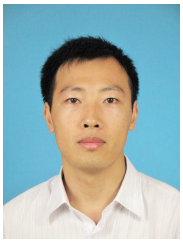
[45] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," *in Proc. Theory of Cryptography Conf. (TCC'05),* vol. 3378, pp. 264-282, 2005.

[46] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New Algorithms for Secure Outsourcing of Modular Exponentiations," *IEEE Trans. Parallel and Distributed Systems,* vol. 25, no. 9, pp. 2386-2396, 2014.

[47] M. Dijk, D. Clarke, B. Gassend, G. Suh, and S. Devadas, "Speeding up exponentiation using an untrusted computational resource," *Designs, Codes and Cryptography*, vol. 39, no. 2, pp. 253-273, 2006.

[48] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," *in: Proc. Advances in cryptology–ASIACRYPT 2010,* vol. 6477, pp. 177-194, 2010.

[49] J. Xu and E. C. Chang, "Towards efficient proofs of retrievability," in Proc. of the 7th ACM Symposium on Information, *Computer and Communications Security (AsiaCCS'12),* pp. 79–80, 2012.



王玉珏 于 2015 年在武汉大学和香港城市大学(联合培养)获得博士学位。现任新加坡管理大学信息系统学院 Research Fellow。主要研究领域为应用密码学。Email: yjwang@smu.edu.sg

伍前红 2005 年获西安电子科技大学密码学博士学位。现为北京航空航天大学教授、博士生导师。研究领域包括公钥密码学, 云计算安全与大数据隐私, 密码货币与区块链。Email: qianhong.wu@buaa.edu.cn