计算机科学
COMPUTER SCIENCE

抗密钥泄露的代理可证数据持有

安睿诚, 王化群

引用本文

安睿诚, 王化群. 抗密钥泄露的代理可证数据持有[J]. 计算机科学, 2024, 51(12): 310-316.

AN Ruicheng, WANG Huaqun. Proxy Provable Data Possession with Key-exposure Resilient[J]. Computer Science, 2024, 51(12): 310-316.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.231100085

抗密钥泄露的代理可证数据持有

安睿诚　王化群
南京邮电大学计算机学院　南京 210023
(1022041202@njupt.edu.cn)

摘　要　云存储近年来发展迅猛,越来越多的用户选择将他们的数据存储在云服务器中。为了检验云存储数据的完整性,研究者们提出了可证数据持有(Provable Data Possession,PDP)。用户在某些情况下无法访问互联网,例如在远洋轮渡上,或是参加某些涉密的项目时,因此必须将远程数据完整性检验委托给代理。然而在代理 PDP 中,一旦用户的私钥泄露,审计方案将无法进行。针对上述问题,所提方案将密钥隔离技术与代理 PDP 相结合,在系统模型中引入了物理上安全但计算受限的助手设备。助手设备在每个时间段生成更新信息并发送给用户,帮助用户计算当前时段的签名密钥。在此方案下,敌手无法在密钥未泄露的时间段伪造用户生成的认证器。安全性分析和性能分析表明,所提方案是安全高效的。

关键词:可证数据持有;抗密钥泄露;代理;云存储安全
中图分类号　TP309

Proxy Provable Data Possession with Key-exposure Resilient

AN Ruicheng and WANG Huaqun
School of Computer Science,Nanjing University of Posts and Telecommunications,Nanjing 210023,China

Abstract　More and more clients would like to store their data to public cloud server along with the rapid development of cloud storage. To check the integrity of remote data,researchers proposed provable data possession(PDP). In some cases,the client will be restricted to access the Internet,such as on the ocean-going vessel,participating in some classified projects. It has to delegate the remote data possession checking task to some proxy. However,in proxy PDP,once the client's private key is exposed,auditing schemes would inevitably become unable to work. To solve these problems,the proposed scheme combines key-insulated with proxy PDP,and introduces a physically-secure but computationally-limited helper into the system model. The helper generates an update message in each time period and then sends it to the client to help the client calculate the signing key for the current time peroid. In this scheme,adversaries cannot forge user-generated authenticators during the time period when the key is not leaked. Security analysis and performance analysis show that the proposed scheme is secure and efficient.

Keywords　Provable data possession,Key exposure resilient,Proxy,Cloud storage security

1　引言

云计算中,用户可以通过网络方便地使用服务提供商的计算资源。云存储是云计算中重要的服务,它允许数据所有者将数据从其本地存储系统移动到云中。越来越多的数据所有者选择在云中托管他们的数据,这样做可以使用户方便地对外包文件进行移动访问,并且从复杂的本地存储管理中解脱出来。然而,一些安全问题可能会阻碍用户使用云存储。其中,外包文件的完整性是一个重要问题。因为将文件存储到云服务器后,用户将失去对文件的物理控制,即使云服务提供商采取高度可靠的措施,也有可能发生数据丢失。有时,云服务提供商可能不诚实,他们会丢弃很少被访问的数据以节省存储空间。此外,云服务提供商可能选择隐藏数据丢失,并声称数据仍然正确地存储在云中[1]。因此,用户可能担心云服务提供商会更改或删除他们的重要文件。

为了解决这一问题,PDP[2]成为了学术界热烈探讨的话题。在 PDP 模型中,用户需要预先计算文件中每一个文件块对应的认证器,并将文件和认证器集合上传到云服务器中,用户只需保留私钥和少量的参数。为检查外包文件是否保持完整,用户或第三方审计(Third Party Auditor,TPA)向云服务器发起挑战。由于认证器具有同态属性,云服务器发回的证明只需包含文件块的线性组合和对应的认证器的线性组合,这可以极大地减少通信成本。如果文件的某些部分被更改或删除,云服务器将无法证明数据完整性。用户在某些情况下无法访问互联网,例如在远洋轮渡上,或是参加某些涉密的项目时,因此无法进行远程数据完整性检验。如果用户将任务

到稿日期:2023-11-14　返修日期:2024-04-26
基金项目:国家自然科学基金(62272238)
This work was supported by the National Natural Science Foundation of China(62272238).
通信作者:王化群(whq@njupt.edu.cn)

委托给 TPA，当云服务器无法通过验证时，TPA 无法通知用户采取进一步的行动。此外，用户不信任 TPA 评估的损失。因此，用户必须将远程数据完整性检验委托给代理。在用户授权后，代理将根据授权执行远程数据完整性检验。当云服务器无法通过检验时，代理可以联系云服务提供商评估损失并采取进一步的行动。

本文将密钥隔离技术与代理 PDP 相结合，在系统模型中引入了物理上安全但是计算受限的助手设备[3]。在我们的详细构造中，每个时间段 t 开始时，助手设备生成更新信息并发送给用户，用户使用收到的更新信息和自己前一时段的签名密钥来生成当前时段的签名密钥。即使敌手在某个时间段获得了用户的私钥以及签名密钥，其他时间段本方案依然是安全的。PDP 根据是否支持公开验证，可分类为私有验证和公开验证[4]。传统的私有 PDP 的验证者是上传文件的用户，本方案是一种特殊的私有 PDP，验证者可以是用户或者是得到授权的代理。

本文第 2 章介绍了可证数据持有的相关工作；第 3 章介绍了系统的模型、定义与准备工作；第 4 章对所提方案进行了详细的描述；第 5 章给出了安全性分析；第 6 章给出了性能分析与实验结果；最后总结全文并展望未来。

2 相关工作

2007 年，Ateniese 等[2]首先提出了“可证数据持有”，该方案允许用户在不检索全部文件的前提下验证存储在云上的数据的完整性。Juels 等[5]提出了“数据可恢复证明”(Proofs of Retrievability，POR)，该方案采用纠错码对数据进行编码，并在文件中嵌入若干“哨兵”。Shacham 等[6]提出了 POR 的一个紧凑版本，有效地实现了基于 BLS 短签名的公开审计。在一些公开审计方案中，云服务器需要将数据块的线性组合发送给第三方审计，这会将数据泄露给审计员。为了增强安全性，Wang 等[7]将 HLA 与随机掩蔽技术相结合，使审计人员无法从审计过程中推断出原始数据。Ateniese 等[8]首先研究了对数据的动态操作，之后研究者们提出了许多动态 PDP 方案，如 Wang 等[9]使用了梅克尔树来改进模型，Erway 等[10]使用了跳表。

然而上述方案都是基于 PKI 体系的，云服务器在存储用户上传的数据之前必须验证用户的证书。许多用户可能频繁地将数据上传到云服务器，这会导致巨大的计算成本。为了解决这些问题，研究者们提出了许多基于身份的 PDP 方案。Zhao 等[11]利用双线性对构造了一个基于身份的公共审计方案。Wang 等[12]通过对 Schnorr 签名进行变化提出了 ID-RD-PC 方案，并在文献[13]中提出了多云存储的 ID-DPDP 方案。Zhang 等[14]设计了一种新的 IBPA 方案，减少了在多用户场景下审计员的计算成本，然而 He 等[15]指出该方案是不安全的，并提出了两种具体的攻击。

在某些情况下，客户没有能力检查其数据所有权。Wang[16]首先提出了 PPDP 方案，客户可以将远程数据拥有的检查任务委托给代理。Wang 等[17]提出了一种基于身份的面向代理的云审计方案(ID-PUIC)，让代理帮助用户生成标签并上传数据。Wang 等[18]采用 Paterson 和 Schuldt 的基于身份的签名方案[19]作为构建模块，提出了 IBDO 方案。Yu 等[20]提出了一种基于身份的面向代理的公共审计，并支持动态操作，然而 Zhao 等[21]指出该方案很容易出现有关数据丢失和受到代理私钥恢复的恶意行为的攻击。

以上审计方案都是假设客户的私钥绝对安全，如果私钥泄露，大多数现有的审计方案将无法进行。Yu 等[22]使用基于二叉树结构的密钥更新技术来保护在密钥泄露之前的时间段中生成的认证器的安全性。Yu 等[23]支持将密钥更新安全地外包给一些授权方，客户端上的密钥更新负担将保持最小。Yu 等[24]提出了强抗密钥泄露审计方案，研究了当密钥泄露发生时，如何在密钥泄露时间段以外的任何时间段内保持云存储审计方案的安全性。Shen 等[25]设计了一种新的密钥更新技术，支持惰性更新，即用户仅在将文件上传到云时才更新其私钥，大大提高了密钥更新的效率和可行性。Zhang 等[26]提出基于格的抗密钥泄露的云审计方案，该方案具有前向安全性。Zhang 等[27]提出抗密钥泄露的支持密态数据去重的完整性审计方案。Nithya 等[28]提出基于身份的抗密钥泄露的审计方案，该方案支持分批审计。

3 定义与模型

3.1 系统模型

本文提出的系统由以下 4 类实体组成。

1)用户(Client)：有大量数据要上传到云服务器中。

2)云服务器(Public Cloud Server，PCS)：有大量的存储空间和计算资源来维护用户的数据。

3)代理(Proxy)：被用户授权检查用户存储在云服务器中的文件的完整性。当代理满足用户的授权 w 时，可以进行检查；否则，它不能执行该程序。

4)助手设备(Helper)：物理上安全但是计算受限的设备。其在不同时间段向用户提供更新信息来帮助用户计算他在当前时段的签名密钥。

系统模型如图 1 所示。

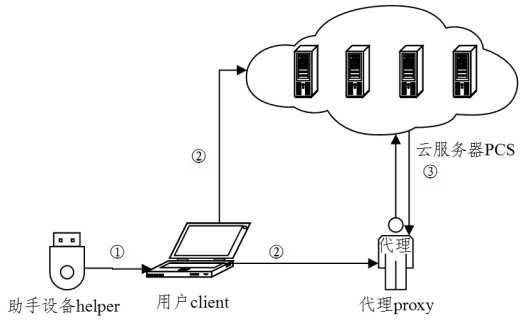

图 1 系统模型

Fig. 1 System model

在时段 t 开始时，助手设备首先给用户发送更新信息，接着用户根据更新信息和上一时段的签名密钥来生成当前时段的签名密钥。

在认证器生成阶段，用户输入文件块和签名密钥来生成认证器，并将文件和认证器集合上传到 PCS。此外，用户将授权 w 发送给代理。

在 *proof* 的生成和验证阶段，代理与 PCS 通过交互的

方式来检查用户远程数据的完整性。

3.2 符号定义

方案中涉及的符号与说明如表1所列。

表1 符号及描述

Table 1 Symbols and their descriptions

符号	说明
G_1, G_2	阶为 q 的乘法循环群
Z_q^*	$\{1,2,\cdots,q-1\}$
g, u	G_1 的生成元
H, H_1, H_2	3个抗碰撞哈希函数
(x, X)	用户私钥/公钥对
(y, Y)	PCS 私钥/公钥对
(z, Z)	代理私钥/公钥对
(hsk, hpk)	助手设备私钥/公钥对
δ_t	t 时刻助手设备的更新信息
sk_t	t 时刻用户的签名密钥
w	用户生成的授权证书
$F=\{m_1, m_2, \cdots, m_n\}$	文件 F 被分成 n 个数据块
T_i	文件块 m_i 对应的认证器
$SSig$	确保文件标识、时间和授权的完整性
(ssk, spk)	SSig 算法对应的私钥/公钥对
$chal$	代理生成的挑战
$P=\{t, R, T, \mu\}$	PCS 返回的完整性证明

3.3 方案框架

本文系统由以下6个算法组成。

1)系统建立(Setup)算法:输入安全参数 k,生成系统中实体的私钥/公钥对,包括用户的私钥/公钥对(x, X)、PCS 的私钥/公钥对(y, Y)和代理的私钥/公钥对(z, Z)。此外,用户生成授权证书 w,其中包含代理需要满足的条件。

2)初始化签名密钥(InitialSigningKey)算法:该算法由用户在初始阶段执行,输入代理和云服务器的公钥以及用户的私钥,生成用户的初始签名密钥 sk_0 以及助手设备的私钥/公钥对(hsk, hpk)。

3)更新签名密钥(UpdateSigningKey)算法:助手设备在每个时间段 t 开始时,向用户发送更新信息 δ_t。用户根据 δ_t 和自己前一阶段的签名密钥 sk_{t-1} 计算当前时段的签名密钥 sk_t。

4)生成认证器(AuthGen)算法:该算法由用户执行。输入文件块 m_i 和签名密钥 sk_t,生成文件块对应的认证器 T_i,然后将文件和认证器集合上传到云服务器,并将授权书 w 以及对 w 的签名发送给代理。

5)生成证明(ProofGen)算法:代理发起挑战,提出$(t, chal)$,云服务器将挑战作为输入,生成证明 $proof$。$proof$ 用来证实存储文件的完整性。

6)验证证明(ProofVerify)算法:该算法由代理运行。它将公钥 X, Y, hpk,私钥 z,时间段 t,挑战 $chal$ 和证明 $proof$ 作为输入,如果验证通过,返回"true",否则返回"false"。

3.4 安全定义

定义1(抗密钥泄露性) 敌手A和挑战者C之间的游戏如下:

1)系统建立:初始设定时间 $t=0$,挑战者C执行Setup方法,获得用户的私钥/公钥对(x, X)、云服务器的私钥/公钥对(y, Y),以及代理的私钥/公钥对(z, Z),并将系统公开参数以及 z 发送给A。

2)询问阶段:C执行InitialSigningKey和UpdateSigningKey,生成时间段 t 的签名密钥 sk_t。

(1)认证器询问:A自适应地选择一系列块$(m_1, \cdots, m_n)$,并将它们提交给C,以在时间 t 查询对应的认证器。挑战者C计算并返回当前时间段与$(m_1, \cdots, m_n)$相关的认证器。

(2)密钥询问:A可以选择在时间段 t 查询客户的私钥,C将客户的私钥 x 以及时间段 t 的签名密钥 sk_t 发送给A。

在每个时间段结束时,A可以选择停留在该阶段或进入挑战阶段。

3)挑战:C选择一个时间段 t^*,在这个时间段内 A 没有进行过密钥询问。C向A发送挑战。

$$chal=\{i, v_i\}_{i\in I}(I=\{s_1, s_2, \cdots, s_c\}, 1\leqslant s_l\leqslant n, 1\leqslant l\leqslant c, 1\leqslant c\leqslant n)$$

A需要提供文件 F 在挑战 $chal$ 下对文件块$\{m_{s_1}, \cdots, m_{s_c}\}$在时间段 t^* 的远程数据完整性证明。

4)伪造:A 将 θ 作为对挑战 chal 的回应。如果回应 θ 可以通过C的验证,则敌手A赢得游戏。

上述定义指出,如果一些被挑战的块已经被修改或者删除,那么恶意云无法生成有效的远程数据完整性证明。此外,一个实用的方案还需要使用户相信,其所有的外包数据有很大的概率保持完整。以下定义指出了安全要求。

定义2 云存储审计方案是(ρ, δ)可检测的。如果被PCS破坏的块的数量占整个文件块数量的比例为 ρ,那么损坏块被检测到的概率至少为 δ。

3.5 准备工作

1)双线性对映射:有两个乘法循环群 G_1, G_2,它们具有相同的素数阶 q。双线性对映射 $e: G_1\times G_1\rightarrow G_2$ 需要满足以下性质。

(1)双线性:对于 $\forall g_1, g_2, g_3\in G_1$ 和 $\forall a, b\in Z_q^*$,$e(g_1, g_2g_3)=e(g_2g_3, g_1)=e(g_2, g_1)e(g_3, g_1)e(g_1^a, g_2^b)=e(g_1, g_2)^{ab}$。

(2)非退化性:$e(g_1, g_2)\neq 1$,在这里 g_1, g_2 是 G_1 的生成元。

(3)可计算性:对于 $g_1, g_2\in G_1$,有一个有效的算法用来计算 $e(g_1, g_2)$。

2)CDH问题:给出(g, g^a, g^b),这里 g 是阶为q的乘法群 G_1 的生成元,$a, b\in Z_q^*$,计算 g^{ab} 是困难的。

3)BDH问题:给出(g, g^a, g^b, g^c),这里 g 是阶为q的乘法群 G_1 的生成元,$a, b, c\in Z_q^*$,计算 $e(g, g)^{abc}$ 是困难的。

4 具体方案

设 $e: G_1\times G_1\rightarrow G_2$ 是一个双线性映射,其中,G_1, G_2 是两个 q 阶的乘法群。g 和 u 是群 G_1 的两个生成元。定义3个抗碰撞哈希函数:

$H: G_2\rightarrow G_1$

$H_1: \{0,1\}^*\rightarrow G_1$

$H_2: \{0,1\}^*\times G_1\rightarrow G_1$

通常在先前的云存储审计方案中[7,9,28],存在用于确保文件标识 $name$ 完整性的数字签名 $SSig$。在本文中,我们也使用相同的数字签名 $SSig$ 来确保文件标识 $name$、时间 t、授权

w 的完整性。我们假设(spk,ssk)是用户对应于签名 $SSig$ 的公私钥对,用户持有 ssk 并且将 spk 公开。

1)Setup

输入安全参数 k,用户随机选择私钥 $x\in Z_q^*$,计算 $X=g^x$ 作为公钥。PCS 随机选择私钥 $y\in Z_q^*$,计算 $Y=g^y$ 作为公钥。代理随机选择私钥 $z\in Z_q^*$,计算 $Z=g^z$ 作为公钥。用户生成授权 w,其中包含代理需要满足的条件。公开参数

$params=\{G_1,G_2,e,q,g,u,X,Y,Z,H,H_1,H_2\}$

2)InitialSigningKey

(1)用户为助手设备随机选择私钥 $hsk\in Z_q^*$,计算 $hpk=g^{hsk}$ 作为助手设备公钥。

(2)用户计算自己在初始时刻的签名密钥为 $sk_0=H(e(Y,Z)^x)^x H_1(0)^{hsk}$。

(3)用户将 hsk 存储在助手设备中,公开 hpk,并在计算完 sk_0 后将本地的 hsk 删除。

3)UpdateSigningKey

(1)在时段 t 开始时,助手设备将更新信息 $\delta_t=(H_1(t)H_1(t-1)^{-1})^{hsk}$ 发送给用户。

(2)收到更新信息后,用户计算 $sk_t=sk_{t-1}\delta_t$ 并删除本地的 sk_{t-1},δ_t。签名密钥 sk_t 用来生成认证器。

4)AuthGen

在时段 t,用户使用轻量级对称加密方案加密文件,然后将文件划分为 n 个块,即 $F=\{m_1,m_2,\cdots,m_n\}$,$m_i\in Z_q^*$。之后用户做如下操作:

(1)选择随机数 $r\in Z_q^*$,计算 $R=g^r$,公开 R。

(2)以如下方式计算认证器 T_i,其中 $name$ 是文件 F 的名称。

$T_i=H_2(t\parallel i\parallel name\parallel w,R)^r u^{r\cdot m_i}sk_t$

(3)将标签 $tag=name\parallel t\parallel SSig_{ssk}(name\parallel t)$和认证器集合 $\phi=\{t,w,R,T_1,\cdots,T_n\}$以及文件 F 一起上传到 PCS。

(4)将$(w,SSig_{ssk}(w))$发送给代理。

5)ProofGen

(1)代理首先验证 $SSig_{ssk}(name\parallel t)$是否为 spk 的有效签名。如果检查通过,代理选择 c 个元素 $I=\{s_1,s_2,\cdots,s_c\}$($1\leqslant s_l\leqslant n$)作为要检查块的索引。

(2)对于每一个 $i\in I$,代理选择随机数 $v_i\in Z_q^*$ 作为参数,然后将挑战 $chal=\{i,v_i\}_{i\in I}$发送给 PCS。

(3)PCS 在接收到挑战 $chal=\{i,v_i\}_{i\in I}$后,首先检查代理是否满足授权 w,如果满足,计算被挑战的文件块的线性组合 $\mu=\sum_{i\in I}v_i m_i$,以及对应认证器的线性组合 $T=\prod_{i\in I}T_i^{v_i}$。

(4)PCS 将证明 $P=\{t,R,T,\mu\}$发回给代理。

6)ProofVerify

当代理接收到证明 P 时,验证以下等式是否成立:

$$e(g,T)=e(R,\prod_{i\in I}H_2(t\parallel i\parallel name\parallel w,R)^{v_i}u^{\mu})\cdot e(X,H(e(X,Y)^z)^{\sum_{i\in I}v_i})e(hpk,H_1(t)^{\sum_{i\in I}v_i})$$

如果成立,返回“true”,否则返回“false”。

5 安全性分析

定理 1(正确性) 对于一个随机的挑战 $chal=\{i,v_i\}_{i\in I}$ 和一个有效的证明 $P=\{t,R,T,\mu\}$,ProofVerify 算法总是返回 true。

证明:

因为 $e(Y,Z)^x=e(g,g)^{xyz}=e(X,Y)^z$,所以如下等式成立:

$$\begin{aligned}e(g,T)&=e(g,\prod_{i\in I}T_i^{v_i})\\&=e(g,\prod_{i\in I}(H_2(t\parallel i\parallel name\parallel w,R)^r u^{r\cdot m_i}sk_t)^{v_i})\\&=e(g,\prod_{i\in I}(H_2(t\parallel i\parallel name\parallel w,R)^r u^{r\cdot m_i})^{v_i})\cdot e(g,\prod_{i\in I}(H(e(Y,Z)^x)^x H_1(t)^{hsk})^{v_i})\\&=e(R,\prod_{i\in I}(H_2(t\parallel i\parallel name\parallel w,R)u^{m_i})^{v_i})\cdot e(g,\prod_{i\in I}(H(e(X,Y)^z)^x)^{v_i})e(g,\prod_{i\in I}(H_1(t)^{hsk})^{v_i})\\&=e(R,\prod_{i\in I}H_2(t\parallel i\parallel name\parallel w,R)^{v_i}u^{\mu})\cdot e(X,H(e(X,Y)^z)^{\sum_{i\in I}v_i})e(hpk,H_1(t)^{\sum_{i\in I}v_i})\end{aligned}$$

定理 2(抗密钥泄露性) 如果在 G_1 上 CDH 问题是困难的,那么我们提出的方案具有抗密钥泄露性。我们将证明,如果敌手可以使挑战者在游戏中中止,那么我们可以构建一个模拟器,能够以不可忽略的概率解决 CDH 问题。

证明:

首先,向挑战者 C 给出 CDH 挑战$(g,P_1=g^a,P_2=g^b)$。挑战者 C 将通过运行敌手 A 的子程序来计算 g^{ab}。挑战者 C 选择(spk,ssk)作为公钥和私钥以生成针对文件名和时间段的签名。假设敌手 A 进行 q_k 次密钥查询和 q_s 次认证器查询。

1)系统建立(Setup):挑战者 C 执行 Setup 方法,获得用户的私钥/公钥对(x,X)、云服务器的私钥/公钥对(y,Y)和代理的私钥/公钥对(z,Z)。w 为客户对代理的授权证书。设置 $P_1=hpk$,随机选择 $\alpha\in Z_q^*$,设置 $u=g^{\alpha}$。最终,C 向 A 发送(g,u,w,X,Y,Z,spk,z)。

2)询问(Query):H,H_1,H_2 被视为由挑战者 C 控制和存储的 3 个随机预言机。挑战者需要回答 A 对这些随机预言机的询问。

①H_1 预言机询问:挑战者 C 维护 H_1-$table$ 用来回答 A 对 H_1 预言机的询问。首先,C 将表初始化为空。当 A 输入$\langle t\rangle$查询 H_1 时,C 做如下操作。

①对于输入$\langle t\rangle$,如果 H_1-$table$ 包含元组(t,c,λ,h_1),则 C 回复 h_1。

②如果不包含,那么 C 掷硬币,$c\in\{0,1\}$,产生 0 的概率为$\frac{q_k+q_s}{q_k+q_s+1}$,产生 1 的概率为$\frac{1}{q_k+q_s+1}$。当 $c=0$ 时,C 选择 $\lambda\in Z_q^*$,计算 $h_1=g^{\lambda}$,并将$(t,0,\lambda,h_1=g^{\lambda})$加入 H_1-$table$ 中。否则当 $c=1$ 时,C 选择 $\lambda\in Z_q^*$,计算 $h_1=P_2^{\lambda}$,并将$(t,1,\lambda,h_1=P_2^{\lambda})$加入 H_1-$table$ 中。最终,C 回复 h_1。

(2)H_2 预言机询问:挑战者 C 维护 H_2-$table$ 用来回答 A 对 H_2 预言机的查询。首先,C 将表初始化为空。当 A 输入$\langle t\parallel i\parallel name\parallel w,R\rangle$查询 H_2 时,C 做如下操作。

①对于输入$\langle t\parallel i\parallel name\parallel w,R\rangle$,如果 H_2-$table$ 包含元组$(t\parallel i\parallel name\parallel w,R,\varphi,h_2)$,C 回复 h_2。

②否则,C 选择 $\varphi\in Z_q^*$,计算 $h_2=g^{\varphi}$,并将$(t\parallel i\parallel name\parallel w,R,\varphi,h_2=g^{\varphi})$加入 H_2-$table$ 中,那么 C 回复 h_2。

(3)H 预言机询问:挑战者 C 维护 *H-table* 用来回答 A 对 H 预言机的询问。当 A 输入$\langle PK_1, PK_2, sk\rangle$查询 H 时,C 做如下操作:

①对于输入$\langle PK_1, PK_2, sk\rangle$,如果 *H-table* 包含元组$(PK_1, PK_2, sk, h)$,那么 C 回复 h。

②否则,C 选择 $\eta \in Z_q^*$,并将$(PK_1, PK_2, sk, h=\eta)$加入 *H-table* 中,C 回复 h。

(4)密钥询问:当 A 输入$\langle t\rangle$查询密钥时,C 从 H_1-*table* 中取出(t, c, λ, h_1),从 *H-table* 中取出(X, Y, z, h)。(这里假设 A 已经进行过这些查询了)

①如果 $c=1$,那么 C 中止(将此事件表示为E_1)

②否则,C 计算 $sk_t = h^x P_1^{\lambda}$

其中:

$sk_t = h^x H_1(t)^{hsk} = h^x g^{\lambda \cdot hsk} = h^x P_1^{\lambda}$

最终 C 回复:$(sk_c = x, sk_t = h^x P_1^{\lambda})$。

(5)认证器询问:当 A 输入$\langle t, m_i, i\rangle$查询认证器时,C 做如下操作:

①首先,C 从 $H_1-table$ 中恢复出(t, c, λ, h_1)

②如果 $c=1$,C 中止(将此事件表示为E_2)

③否则,C 选择 $r \in Z_q^*$,计算 $R=g^r$,C 还需选择 $T_i \in G_1$。

④C 定义哈希值:

$H_2(t \| i \| name \| w, R) = (T_i h^{-x} P_1^{-\lambda})^{r^{-1}} / u^{m_i}$

注意:

$$\begin{aligned} &H_2(t \| i \| name \| w, R)^r u^{r \cdot m_i} sk_t \\ &= H_2(t \| i \| name \| w, R)^r u^{r \cdot m_i} h^x P_1^{\lambda} \\ &= ((T_i h^{-x} P_1^{-\lambda})^{r^{-1}} / u^{m_i})^r u^{r \cdot m_i} h^x P_1^{\lambda} \\ &= T_i \end{aligned}$$

最终,C 回复(t, R, T_i)。

3)挑战(Challenge):挑战者 C 选择一个时间段 t^*,在这个时间段内 A 没有进行过密钥询问。C 向 A 发送挑战 $chal=\{i, v_i\}_{i \in I}$($I=\{s_1, s_2, \cdots, s_c\}$,$1 \leqslant s_l \leqslant n$,$1 \leqslant l \leqslant c$,$1 \leqslant c \leqslant n$)和时间段 t^*。C 需要 A 提供文件 F 在挑战 chal 下对文件块$\{m_{s_1}, \cdots, m_{s_c}\}$在时间段 t^* 的远程数据完整性证明 P。

4)伪造(Forgery):最终,敌手 A 输出证明 $P=\{t^*, R, T, \mu\}$。C 从 H_1-*table* 中取出$(t^*, c^*, \lambda^*, h_1^*)$,从 *H-table* 中取出$(X, Y, z, h)$。

(1)如果 $c^*=0$,C 中止(将此事件表示为E_3)。

(2)否则,如果 R 不同于认证器集合中的 R,那么 C 从 H_2-*table* 取出$(t^* \| i \| name \| w, R, \varphi_i^*, h_2 = g^{\varphi_i^*})$。在这种情况下,$h_1^* = P_2^{\lambda^*}$。如果这个伪造是有效的,那么:

$$\begin{aligned} e(g, T) &= e(R, \prod_{i \in I} H_2(t \| i \| name \| w, R)^{v_i} u^{\mu}) \cdot e(X, \\ &\quad H(e^{(X}, Y)^z)^{\sum v_i}) e(hpk, H_1(t)^{\sum_{i \in I} v_i}) \\ &= e(R, \prod_{i \in I} g^{\varphi_i^* v_i} \cdot g^{a\mu}) e(X, h^{\sum_{i \in I} v_i}) e(P_1, P_2^{\lambda^* \sum_{i \in I} v_i}) \\ &= e(R, g^{a\mu + \sum_{i \in I} \varphi_i^* v_i}) e(X, h^{\sum_{i \in I} v_i}) e(P_1, P_2^{\lambda^* \sum_{i \in I} v_i}) \end{aligned}$$

所以:

$$e(P_1, P_2^{\lambda^* \sum_{i \in I} v_i}) = e(g, T) e(g, R^{-(a\mu + \sum_{i \in I} \varphi_i^* v_i)} \cdot e(g, h^{-x \sum_{i \in I} v_i})$$

①如果 $\sum_{i \in I} v_i = 0$,C 中止(将此事件表示为E_4)。

②否则 C 可以计算出 g^{ab}:

$$g^{ab} = (T \cdot R^{a\mu + \sum_{i \in I} \varphi_i^* v_i} \cdot h^{-x \sum_{i \in I} v_i})^{(\lambda^* \sum_{i \in I} v_i)^{-1}}$$

我们分析了 C 在上述情形中不中止的可能性。根据以上的描述,C 不中止的概率为:

$$\begin{aligned} &Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4] \\ &\geqslant \left(\frac{q_k + q_s}{q_k + q_s + 1}\right)^{q_k} \left(\frac{q_k + q_s}{q_k + q_s + 1}\right)^{q_s} \frac{1}{q_k + q_s + 1} \frac{q-1}{q} \\ &= \left(\frac{q_k + q_s}{q_k + q_s + 1}\right)^{q_k + q_s} \frac{1}{q_k + q_s + 1} \frac{q-1}{q} \\ &\geqslant \frac{q-1}{e \cdot q \cdot (q_k + q_s + 1)} \end{aligned}$$

这里 q_k 是敌手 A 进行密钥查询的次数,q_s 是进行认证器查询的次数。如果敌手成功,但是在他的完整性证明 P 中的 R 和认证器列表中存储的 R 不同,那么 C 能够以不可忽略的概率解决 G_1 上的 CDH 问题。

定理 3(可检测性) 本文提出的审计方案是$\left(\frac{m}{n}, 1-\left(\frac{n-m}{n}\right)^c\right)$可检测的。如果云存储的文件具有 n 个块,其中包含 m 个坏块(被删除或修改),TPA 挑战的区块数量是 c。

证明:假设云存储的文件共 n 个块,其中包含 m 个坏块(被删除或修改),TPA 挑战的区块数量是 c。因此,当且仅当 TPA 挑战的块中至少包含一个坏块时,我们的审计方案才能生效。使用离散随机变量 X 来表示 TPA 挑战的块中包含坏块的数量,使用 P_x 表示 TPA 挑战的块中至少包含一个坏块的概率:

$$\begin{aligned} P_x &= P\{X \geqslant 1\} \\ &= 1 - P\{X=0\} \\ &= 1 - \frac{n-m}{n} \frac{n-1-m}{n-1} \times \cdots \times \frac{n-c+1-m}{n-c+1} \end{aligned}$$

可以得到 $P_x \geqslant 1-\left(\frac{n-m}{n}\right)^c$。因此,我们提出的审计方案是$\left(\frac{m}{n}, 1-\left(\frac{n-m}{n}\right)^c\right)$可检测的。

6 性能分析与实验结果

首先,给出所提方案的计算和通信开销。接着,实现了方案的原型并评估其时间成本。最后,比较了所提方案与其他代理可证数据持有方案。

1)计算开销:假设上传的文件被分为 n 块,挑战 $chal=\{i, v_i\}_{i \in I}$。根据不同的阶段,给出计算开销。在群 G_1 上,双线性对运算、指数运算和乘法运算贡献了大多数的计算成本。和这些运算相比,其他运算速度较快,例如哈希运算、Z_q^* 上的运算和 G_2 上的运算。因此,我们只考虑 G_1 上的双线性对运、指数运算和乘法运算。对于用户,计算开销主要来自 AuthGen 阶段。在 AuthGen 阶段,用户在群 G_1 上执行 $2n$ 次指数运算和 $2n$ 次乘法运算。在 *proof* 的生成和验证阶段,代理生成挑战 *chal*,PCS 对其响应。PCS 在群 G_1 上执行 c 次指数运算和 $c-1$ 次乘法运算,然后将响应 *proof* 发送给代理。为了检查 *proof* 的有效性,代理在群 G_1 上执行 $c+3$ 次指数运算,c 次乘法运算和 5 次双线性对的运算。为了显示所提方案的实际开销,我们进行了模拟,并和文献[24]中的方案进行

了对比。该方案也实现了抗密钥泄露性，区别在于它是支持公开验证的 PDP，而本文方案是私有 PDP。

使用 java 编程语言，JPBC 库、Commons-IO 库和 Fastjson 库，给出在 windows10 环境下的效率分析结果。实验配置为 Intel(R) Core(TM) i5-5257U CPU @ 2.70 GHz，8.00 GB RAM。在实验中，我们选择 A 型双线性对，其中群的阶数为 160 bits。在 AuthGen 阶段，用户创建 200 个认证器的耗时为 9.839 s，创建 1 000 个认证器的耗时为 46.794 s。图 2 显示了用户在 AuthGen 阶段的时间消耗。x 轴表示创建标签的数量，y 轴表示创建对应数量认证器的耗时。从图 2 中可以看出，在这两个方案中用户生成认证器的时间消耗几乎以线性的速度随着认证器数量的增加而增加，本文方案与文献[24]中的方案在 AuthGen 阶段的时间消耗几乎相同。接下来的实验设定文件的分块数量为 1 000，在证明 *proof* 的生成和验证阶段，当被挑战的文件块数量为 200 时，代理生成 *chal* 的时间消耗为 0.006 s，PCS 生成 *proof* 的时间消耗为 2.321 s，代理验证 *proof* 的时间消耗为 7.287 s。当被挑战的文件块数量为 500 时，代理生成 *chal* 的时间消耗为 0.011 s，PCS 生成 *proof* 的时间消耗为 5.682 s，代理验证 *proof* 的时间消耗为 18.042 s。

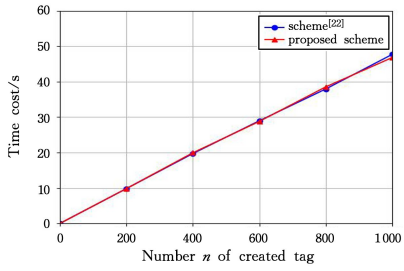

图 2 用户生成认证器的时间消耗

Fig. 2 Client's time cost in AuthGen

图 3 给出了 PCS 和代理在挑战与证明阶段的时间消耗，x 轴表示挑战的文件块数量 c，y 轴表示时间消耗。从图中可以看出，生成 *chal* 阶段花费的时间最少，生成 *proof* 阶段花费的时间稍多，ProofVerify 阶段花费的时间最多。因此我们只与文献[24]中的方案对比 ProofVerify 阶段的时间消耗。通过对比可以发现，本文方案与文献[24]中的方案在 ProofVerify 阶段的时间消耗几乎相同。

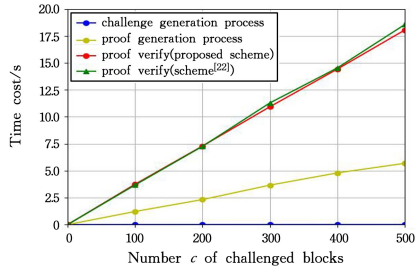

图 3 PCS 和代理在挑战与证明阶段的时间消耗

Fig. 3 PCS and Proxy's time cost in challenge & proof phases

2)通信开销：在处理完文件后，文件和认证器集合被一次性上传到 PCS 中。因此，我们只考虑方案中传输挑战 *chal* 和证明 *proof* 的通信开销。挑战 $chal=\{i,v_i\}_{i\in I}$ 由 c 对元素组成，因此传输挑战的通信开销为 $c\cdot(|I|+|q|)$，其中 $|I|$ 为文件块下标的长度，$|q|$ 为 Z_q^* 中一个元素的长度。证明 $proof=\{t,R,T,\mu\}$，因此传输证明的通信开销为 $2|G_1|+|q|$（t 的大小可忽略不计），其中 $|G_1|$ 为 G_1 上一个元素的大小。

3)比较：为了证明所提方案的优越性，将所提方案与另外两篇代理 PDP，即 Wang 的方案[16]和 Wang 的方案[17]进行了对比。由于大多数计算成本来自于群 G_1 上的双线性对运算、指数运算和乘法运算，因此我们在表 2 中给出了简单的比较。表 2 中，将双线性对运算、指数运算和乘法运算的时间开销分别表示为 C_p，C_e，C_m，可以看出，所提方案在各个阶段的计算成本几乎相同。此外，所提方案可以实现抗密钥泄露。

表 2 计算开销和安全属性的比较

Table 2 Computing overhead vs. security property

方案	生成认证器	生成 *Proof*	验证 *Proof*	抗密钥泄露
Wang 等[16]	$1C_p+2C_e+1C_m$	$cC_e+(c-1)C_m$	$3C_p+(c+1)C_e+cC_m$	No
Wang 等[17]	$2C_e+1C_m$	$cC_e+(c-1)C_m$	$2C_p+(c+1)C_e+cC_m$	No
Ours	$2C_e+2C_m$	$cC_e+(c-1)C_m$	$5C_p+(c+3)C_e+cC_m$	Yes

结束语 本文提出了抗密钥泄露的代理可证数据持有。在所提方案中，攻击者无法在密钥未泄露的时间段伪造用户生成的认证器。安全性分析表明，本文方案具有抗密钥泄露性、可检测性。之后的工作可以考虑研究更高效的代理 PDP。

参 考 文 献

[1] YANG K, JIA X. Data storage auditing service in cloud computing: challenges, methods and opportunities[J]. World Wide Web, 2012, 15: 409-428.

[2] 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.

[3] DODIS Y, KATZ J, XU S, et al. Key-insulated public key cryptosystems[C]// Advances in Cryptology—EUROCRYPT 2002: International Conference on the Theory and Applications of Cryptographic Techniques Amsterdam, The Netherlands, April 28— May 2, 2002 Proceedings 21. Springer Berlin Heidelberg, 2002: 65-82.

[4] YUAN Y, ZHU H L, CHEN Y L, et al. Survey of data integrity verification technology based on provable data possession[J]. Computer Engineering and Applications, 2019, 55(18): 1-7, 52.

[5] JUELS A, KALISKI JR B S. PORs: Proofs of retrievability for large files[C]// Proceedings of the 14th ACM Conference on Computer and Communications Security. 2007: 584-597.

[6] SHACHAM H, WATERS B. Compact proofs of retrievability[J]. Journal of Cryptology, 2013, 26(3): 442-483.

[7] WANG C, CHOW S S M, WANG Q, et al. Privacy-preserving public auditing for secure cloud storage[J]. IEEE Transactions on Computers, 2011, 62(2): 362-375.
[8] ATENIESE G, DI PIETRO R, MANCINI L V, et al. Scalable and efficient provable data possession[C]// Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks. 2008: 1-10.
[9] WANG Q, WANG C, REN K, et al. Enabling public auditability and data dynamics for storage security in cloud computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2010, 22(5): 847-859.
[10] ERWAY C C, KÜPÇÜ A, PAPAMANTHOU C, et al. Dynamic provable data possession[J]. ACM Transactions on Information and System Security(TISSEC), 2015, 17(4): 1-29.
[11] ZHAO J, XU C, LI F, et al. Identity-based public verification with privacy-preserving for data storage security in cloud computing[J]. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2013, 96(12): 2709-2716.
[12] WANG H, WU Q, QIN B, et al. Identity-based remote data possession checking in public clouds[J]. IET Information Security, 2014, 8(2): 114-121.
[13] WANG H. Identity-based distributed provable data possession in multicloud storage[J]. IEEE Transactions on Services Computing, 2014, 8(2): 328-340.
[14] ZHANG J, DONG Q. Efficient ID-based public auditing for the outsourced data in cloud storage[J]. Information Sciences, 2016, 343: 1-14.
[15] HE D, WANG H, ZHANG J, et al. Insecurity of an identity-based public auditing protocol for the outsourced data in cloud storage[J]. Information Sciences, 2017, 375: 48-53.
[16] WANG H. Proxy provable data possession in public clouds[J]. IEEE Transactions on Services Computing, 2012, 6(4): 551-559.
[17] WANG H, HE D, TANG S. Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(6): 1165-1176.
[18] WANG Y, WU Q, QIN B, et al. Identity-based data outsourcing with comprehensive auditing in clouds[J]. IEEE Transactions on Information Forensics and Security, 2016, 12(4): 940-952.
[19] PATERSON K G, SCHULDT J C N. Efficient identity-based signatures secure in the standard model[C]// Australasian Conference on Information Security and Privacy. Berlin, Heidelberg: Springer, 2006: 207-222.
[20] YU H, CAI Y, KONG S, et al. Efficient and Secure Identity-Based Public Auditing for Dynamic Outsourced Data with Proxy [J]. KSII Transactions on Internet & Information Systems, 2017, 11(10): 5039-5061.
[21] ZHAO J, XU C, CHEN K. Detailed analysis and improvement of an efficient and secure identity-based public auditing for dynamic outsourced data with proxy[J]. Journal of Information Security and Applications, 2019, 47: 39-49.
[22] YU J, REN K, WANG C, et al. Enabling cloud storage auditing with key-exposure resistance[J]. IEEE Transactions on Information Forensics and Security, 2015, 10(6): 1167-1179.
[23] YU J, REN K, WANG C. Enabling cloud storage auditing with verifiable outsourcing of key updates[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(6): 1362-1375.
[24] YU J, WANG H. Strong key-exposure resilient auditing for secure cloud storage[J]. IEEE Transactions on Information Forensics and Security, 2017, 12(8): 1931-1940.
[25] SHEN W, YU J, YANG M, et al. Efficient identity-based data integrity auditing with key-exposure resistance for cloud storage [J]. IEEE Transactions on Dependable and Secure Computing, 2022, 20(6): 4593-4606.
[26] ZHANG X, WANG H, XU C. Identity-based key - exposure resilient cloud storage public auditing scheme from lattices[J]. Information Sciences, 2019, 472: 223-234.
[27] ZHANG X S, LI C, LIU Z H. Key-exposure resilient integrity auditing scheme with encrypted data deduplication[J]. Journal on Communications, 2019, 40(4): 95-106.
[28] NITHYA S M V, UTHARIARAJ V R. Identity-based public auditing scheme for cloud storage with strong key-exposure resilience[J]. Security and Communication Networks, 2020, 2020: 1-13.

AN Ruicheng, born in 1997, master. His main research interests include applied cryptography and information security.

WANG Huaqun, born in 1974, Ph. D, professor. His main research interests include applied cryptography, blockchain, and cloud computing security.

(责任编辑:何杨)