计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2015.20140353
52(7):1672-1681,2015

一种基于公钥分割的多副本持有性证明方案

付 伟[1] 吴晓平[1] 叶 清[1] 肖 侬[2] 卢锡城[2]
[1](海军工程大学信息安全系 武汉 430033)
[2](国防科学技术大学计算机学院 长沙 410073)
(lukeyoyo@tom.com)

A Multiple Replica Possession Proving Scheme Based on Public Key Partition

Fu Wei[1], Wu Xiaoping[1], Ye Qing[1], Xiao Nong[2], and Lu Xicheng[2]
[1](*Department of Information Security, Naval University of Engineering, Wuhan* 430033)
[2](*School of Computer, National University of Defense Technology, Changsha* 410073)

Abstract In outsourcing cloud storage environment, users cannot completely trust storage service providers. It is a challenge problem to validate whether storage service providers are faithfully maintaining enough replicas complying its promise with users. Most of existing solutions have several disadvantages, such as low efficiency, high computation overload and the absence of supporting for dynamic data updating. A multiple replica cloud storage model with Collector is presented, and a novel multiple replica possession proving scheme, namely MRP-PKP (multiple replica possession proving scheme based on public key partition), is proposed based on public key partition. In preparing phrase, a public key is divided into several private shares and distributed to corresponding storage servers. In validating phrase, only after all storage servers show their possession evidences can the challenge be admitted as success. The scheme is designed to defeat collude adversaries, and can support dynamic data updating operations at block level easily. It is the first scheme to validate all replica's possessions with just one challenge. Both theoretical analysis and simulating experiment show that MRP-PKP scheme has higher secure guarantee, lower communication cost and computation overload than existing schemes.

Key words cloud storage; cloud security; multiple replica; possession proving; public key partition

摘 要 在数据外包的云存储环境中,如何验证存储服务方是否忠诚地按照客户需求保存足够数量的副本数据是一个挑战性问题.现有方案只能对各个副本逐一进行验证,存在验证效率低、计算开销大和对数据更新支持弱等缺点.提出一种带 Collector 的多副本云存储模型,在此基础上给出一种基于公钥分割的多副本持有性证明方案(multiple replica possession proving scheme based on public key partition, MRP-PKP).该方案将公钥分割为多个份额并分配给对应的副本存储节点;在验证时,能够一次性对所有副本的持有性进行高效验证.此外,该方案可有效防御同谋攻击,能够方便地支持数据块级更新操作.进一步理论分析和模拟实验表明:与传统方案相比,MRP-PKP 方案具有安全性高、通信开销低、运算代价小等优势.

关键词 云存储;云安全;多副本;持有性证明;公钥分割

中图法分类号 TP302.1

收稿日期:2014-04-22;**修回日期**:2014-10-24

基金项目:国家自然科学基金项目(61100042);后勤科技重大项目(AWS14R013);中国博士后科学基金项目(2013M532170);信息保障技术重点实验室开放基金项目(KJ-14-102)

云存储[1-2]是在云计算技术[3]基础上延伸和发展出来的一种新的存储方式,也是当前信息技术领域研究的热点和难点问题之一.云存储利用集群、网格或分布式文件系统等技术,将网络中大量不同类型的存储设备通过应用软件集成起来协同工作,共同完成复杂的数据管理功能.作为新兴的数据存储模式,云存储一经提出就立即取得了很多 IT 公司的支持和关注,包括国外的 Amazon S3[4]和 Dropbox,国内主要的云存储服务则有金山快盘、360 云盘、百度网盘和 115 网盘等.

典型的云存储与访问模式包括云存储服务端和用户端两大部分.云存储服务端由一个名字节点和数量众多的存储节点组成;用户端包括用户和共享用户,前者将自己的数据加密后以托管的形式上传到服务端中,在需要的时候下载并解密使用,后者在得到前者的授权后也可以从云端下载并使用该数据,如图 1 所示:

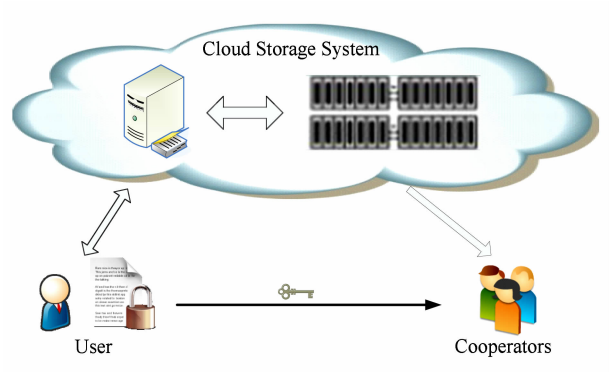


Fig. 1　Cloud storage and access pattern.

图 1　云存储与访问模式

副本技术是提高数据可用性,保证数据高效、快速访问的常用技术.但是云存储服务方可能不愿意或者不能同时维持足够数量的副本,可能的原因包括:

1) 与传统系统相比,云存储系统构成更复杂,系统规模更庞大,因此在任何时刻都可能因为各种故障造成某些副本不可用,使得系统维持的副本数量不足;

2) 出于节约电费成本的考虑,云存储系统通常是在满足一般需求的前提下尽可能地关闭一些节点,这将导致其上的数据副本不可用.

调查表明[5]:出于对公司声誉的影响,或者对经济因素的考虑,在出现上述情况时,云存储服务提供商一般不会主动公布真实情况.对于广大云用户而言,目前缺乏对多副本持有性验证的有效方法.

研究人员已经针对单一数据的持有性验证问题提出多种方案[6-12],然而由于"同谋(collude)攻击"的存在,这些方案并不能直接应用于多副本持有性验证.一些研究人员试图改进单一数据持有性证明方案以满足多副本持有性验证需求,但是这些方案普遍存在验证效率低、计算开销大和对数据更新支持弱等缺点.研究高效、支持数据更新的多副本持有性验证方案具有重要的理论意义和紧迫的应用需求.

本文针对云存储环境下的多副本持有性验证问题,创新地使用公钥分割思想,设计了一种符合 Map/Reduce 编程模型、支持数据动态更新的多副本存在性证明方案 MRP-PKP.该方案将公钥分割为多个份额并分配给对应的副本存储节点,通过数据分块和伪随机变换操作生成不同的副本数据块,能够一次验证所有副本的存在性,具有安全性高、通信开销低、运算代价小等优点.

1　相关工作

数据的持有性证明问题是云存储安全的挑战性问题之一[5].Ari 等人[6]和 Ateniese 等人[7]先后提出可恢复性证明(proofs of retrievability, POR)和数据持有性证明(provable data possession, PDP)技术,分别利用纠错编码和非对称密码体制给出了 2 种解决方案.此后 Ateniese 等人[8]进一步提出可扩展的高效 PDP(scalable and efficient PDP, SPDP)技术,Erway 等人[9]提出 DPDP(dynamic PDP)技术,Zhu 等人[10]提出协作式 PDP(cooperative PDP, CPDP)技术,试图解决 PDP 技术效率低、不支持动态更新等问题.Yun 等人[11]提出一种基于 Nonce 的 MAC(massage authentication code) Tree 方案,Wang 等人[12]提出一种基于同态签名和 RS(seed-solomon)纠错码的验证方法.这些方案只能处理单一数据的持有性验证问题.

然而对于多副本的存储系统而言,只要存在一个正确数据副本,云存储平台就可以在被挑战时临时生成多个副本来欺骗用户,这称为"同谋攻击".显然,上述方案均不能适用于多副本应用场景.Curtmola 等人[13]提出针对多副本持有性验证的 MR-PDP(multiple replica PDP)技术.该技术首先将数据加密,然后将加密数据与 n 个不同的随机掩码异或形成不同的数据文件,可有效防范同谋攻击.MR-PDP 对 n 个副本的证明开销小于 n 个不同文件的证明开销之和,且支持新副本生成.但是该方案需要逐一验证所有副本,效率较低;验证端和服务端的计算开销都比较大,而且不支持数据的动态更新.

Bowers 等人[14]在 POR 的基础上提出 HAIL (high-availability and integrity layer)方案,在多个存储服务提供者之间使用纠错编码产生冗余数据,然后检测岗哨验证数据是否被破坏. 这种方法具有计算开销小的优点,但由于每次挑战需要消耗掉一个岗哨,因此只能执行有限次的挑战. 此外,文件在更新时需要对所有数据重新编码和分块.

Hao 等人[15]基于双线性函数和同态认证标签技术提出一种需要第三方审计(third-party auditor, TPA)的多副本持有性证明方案. TPA 代表用户发起挑战并计算持有性证据,然后利用双线性函数的特性对所有副本的存在证据同时进行验证. 该方案需要一个可信的 TPA 服务器,然而这种服务器在现实应用中并不容易实现.

针对地理分布的混合云架构中节点之间带宽不对称的事实,He 等人[16]提出一种分布式多副本数据持有性检查(distributed multiple replicas data possession checking, DMRDPC)方案,通过在完全双向连通图中寻找最优 Spanning Tree 来定义多副本数据持有性证明的偏序,并获得更好的通信速度. 但是寻找最优 Spanning Tree 的开销将随着节点数目的增加而快速增长.

Barsoum 等人[17]基于 Merkle Tree 和图提出 2 种动态多副本数据持有性证明方案,防止云存储服务器不按照约定维持足够的副本数量. 该方案可同时支持数据修改、插入、删除与追加等更新操作,但是这些操作均需要对整个数据文件进行重新处理,带来的开销较大.

付艳艳等人[18]提出一种基于多轮"挑战-应答"的多副本文件完整性验证方案,可检测现有副本的完整性,且能发现故障副本的位置. 但该方案需要进行多轮交互,效率不高. 最重要的是,该方案无法有效抵御同谋攻击,存在很大的应用局限性.

与上述这些方案相比,本文提出的方案能够有效抵御同谋攻击,可以一次性对所有副本的持有性进行验证,提高了验证效率. 同时,方案所使用的模幂运算大大减少,可以方便地支持文件块级的数据动态更新.

2 模型与符号说明

2.1 符号说明

为了便于描述,首先给出本文所使用的符号定义:

S:由存储节点索引构成的集合,$S=\{1,2,\cdots,n\}$;

I:副本节点索引构成的集合,其规模记为 r;

E:包含 r 个秘密整数 e_i 的集合;

C:待挑战数据块索引构成的集合,其规模记为 k;

b:需要存储的数据,可被分为长度为 l 的 t 块,即 $b=\{b_j \mid j=1,2,\cdots,t\}$;

T:所有验证标签 T_j 组成的集合,即 $T=\{T_j \mid j=1,2,\cdots,t\}$;

M_i:分配给第 i 个存储节点的副本,记为 $M_i=\{m_{i,} \mid j=1,2,\cdots,t\}$;

η:服务端副本块的修改比例,$\eta\in(0,1)$;

ε:用户挑战的置信度,$\varepsilon\in(0,1)$;

ρ_i:第 i 个副本节点根据保存的副本数据生成的分证据,初始值为 1;所有分证据合成最终证据 ρ.

2.2 模型定义

结合云计算的 Map/Reduce 模型[19],本文提出一种基于 Collector 的多副本云存储模型,如图 2 所示:

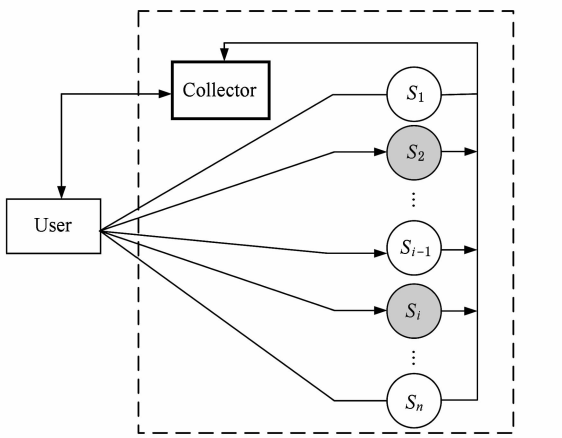


Fig. 2 A multi-replica cloud storage model with Collector.

图 2 基于 Collector 的多副本云存储模型

该模型主要包括客户端和云存储服务端 2 个部分;云存储端由一个 Collector 节点和数量为 n 的存储节点 $S_1,S_2,\cdots,S_n$ 组成;所有存储节点的标示符构成集合 $S=\{1,2,\cdots,n\}$. 用户的数据被保存在多个存储节点上,这些节点的标示符构成另外一个副本集合 I,如图 2 中灰色节点所示. 用户和其他授权用户可以通过 Collector 查询副本集合信息,然后选择从某个副本节点获得数据. 其工作过程主要包括 3 个阶段:

1) 副本存储阶段

用户向云存储服务端提出副本数量要求,例如为数据 b 保存 r 个副本. 然后用户需要对 b 进行预

处理生成副本及对应的数据标签,最后将这些副本、标签及一个公钥分割后产生的私有份额发送给对应的存储节点.

2) 多副本挑战阶段

在数据的有效生存期内,用户可以随时向云存储系统提出挑战;Collector 则据此在各个副本节点上根据保存的数据块内容和公钥私有份额计算得到分证据,并发送给 Collector 汇总生成存在性证据.

3) 证据验证阶段

用户接收来自 Collector 的证据并进行分析,最后根据比对结果判断所有副本是否真实地存储于服务端.

2.3 敌手模型

为了节约存储空间,在不被用户发现的前提下,Collector 和存储节点有可能通过"同谋攻击"的方式联合起来欺骗用户,即:云存储端将尽力在保存少于 r 个副本的前提下也能产生一个证据 ρ,从而成功地通过用户的验证. 因此本文将要面对的是一种信任受限前提下的"同谋"敌手 $\mathcal{A}$. 他们在大多数时间会按照约定为用户提供基本的多副本存储服务,但是他们也将利用 Collector 和存储节点收集和互相交换用户的一切信息. 若 $\mathcal{A}$ 发现删除某些副本并且依然能够以高于 ε 的概率成功通过挑战时,他们将秘密删除这些副本以节约存储空间;反之,如果 $\mathcal{A}$ 认为它的欺骗行为将以不低于 ε 的概率被用户发现,它将选择忠实履行自己的约定,在足够数量的服务器上为用户保留对应数量的副本.

3 MRP-PKP 方案介绍

本文提出一种基于公钥分割的多副本持有性证明(multiple replica possession proving scheme based-on public key partition, MRP-PKP)方案. 该方案将用户的公钥以份额的形式进行分割,并将特定的份额分配给特定的副本节点,且特定的份额只能作用于特定的副本内容形成持有性分证据. 该方案可保证:如果某些副本被删除或者修改,同谋敌手 $\mathcal{A}$ 将不能形成正确的持有性证据,使得这种行为有较大概率被客户发现.

3.1 相关函数介绍

MRP-PKP 方案涉及到 3 个函数,分别描述如下:

1) 秘密份额生成函数 $PKPartition(\pi,r)\rightarrow E$

令 π 为系统参数集合,该函数秘密选取大安全素数 $p=2p'+1$,$q=2q'+1$,并记 $N=p\times q$,则欧拉函数 $\varphi(N)=(p-1)(q-1)=4p'q'$. 然后选择 RSA 体制下的公私钥对 (sk,pk),即 $sk\times pk=1 \bmod \varphi(N)$. 最后将公钥 pk 分割为 r 份,方法是:选取 r 个秘密数 e_i,使得 $\sum_{i=1}^{r} e_i \equiv pk \bmod \varphi(N)$,返回集合 $E=\{e_i \mid 1\leqslant i\leqslant r\}$.

2) 伪随机函数 $\psi_\kappa(i,j)\rightarrow\{0,1\}^{128}$

令 κ 为用户保管的密钥(长度为 $|\kappa|$),i,j 为整数,则:$\psi_\kappa(i,j)=MD5(MD5(i\parallel j)\parallel MD5(\kappa))$.

3) 伪随机挑选函数 $\zeta_k(A,m)\rightarrow B$

其中 $A=\{a_1,a_2,\cdots,a_n\}$,记 $\zeta_k(A,m)=\{b_1,b_2,\cdots,b_m\}$,则 B 中元素 b_j 生成规则为 $b_j=(i\times\psi_\kappa(i,j)) \bmod n$,其中 $i=1$,n 为 A 中元素个数. 如果 b_j 已经在 B 中,则将 i 增加 1 后继续计算新的 b_j,直至产生新的元素.

3.2 方案详细描述

1) 副本存储阶段

主要由用户端完成,各个步骤的详细描述如下:

① 系统初始化,用户针对数据 b 选择密钥 κ.

② 副本存储请求及应答,用户向 Collector 发起需要 r 个副本的存储请求.

③ 公钥分割与份额分配,Collector 根据存储请求查询各存储节点状态,从中挑选出 r 个节点形成副本集合 I 并返回给用户;然后用户调用 $PKPartition(\cdot)$ 函数选择公私钥对 (sk,pk),最后为 I 中的每个节点 S_i 分配并发送一个私有份额 e_i.

④ 数据分块处理,将数据 b 分为长度为 l 的 t 个数据分块. 接下来每个分块将被转换为 $[0,2^l-1]$ 范围之内的一个大数,并以整数的方式参与计算.

⑤ 多副本的生成与分发,对于副本集合中的第 i 个节点 S_i,设对应副本 M_i 的第 j 块数据块为 $m_{i,j}$,其生成过程为

$$m_{i,j}=b_j\times\psi_\kappa(i,j), \tag{1}$$

可见每个副本分块的内容是互不相同的,对应副本为

$$M_i=\{m_{i,1},m_{i,2},\cdots,m_{i,r}\}, \tag{2}$$

最后生成的 M_i 将发往对应副本节点 S_i 存储.

⑥ 数据标签生成阶段,用户端需要计算每个数据块 b_j 对应的块标签 T_j 并发送给 Collector 保存. 实际上经过 r 次循环后:

$$T_j=b_j\times\prod_{i\in I}\psi_\kappa(i,j)^{sk\times e_i} \bmod N. \tag{3}$$

用户端操作步骤伪代码如图 3 所示：

```
Step1. Choose a secret key κ;
Step2. Send request {r} to Collector;
Step3. Receive I from Collector, PKPartition(π,r)=E;
       For 1≤i≤r
         Send e_i to the i-th Storage Server;
Step4. b={b_1,b_2,…,b_t};
Step5. For 1≤i≤r, generate each replica M_i
         For 1≤j≤t, generate each block of M_i
           m_{i,j}=b_j×ψ_κ(i,j);
         Send M_i to S_i;
Step6. For 1≤j≤t, generate block tag T_j
         For 1≤i≤r
           T_j=T_j×ψ_κ(i,j)^{sk×e_i} mod N;
         T_j=b_j×T_j;
         Send T_j to Collector.
```

Fig. 3 The pseudo-code of user side at replica storage phrase.

图 3 副本存储阶段用户端伪码描述

2) 多副本持有性挑战阶段

这个阶段需要由用户端、Collector 和 Storage Server 这 3 方共同完成. 其中，用户端根据敌手模型参数 ε 和 η 计算得到需要挑战的数据块数 c（详见 5.1 节），然后获得待挑战数据块索引集合 C 并发送给 Collector. 各个 Storage Server 依据 C 计算分证据 ρ_i：

$$\rho_i = \prod_{k\in C} m_{i,k}^{e_{i,k}} \bmod N, \tag{4}$$

然后发送给 Collector. 主要工作由 Collector 完成，包括 4 个步骤：

① 接收分证据. Collector 从每个 S_i 接收分证据 ρ_i.

② 生成最终证据. Collector 将所有分证据累乘后得到最终证据 ρ：

$$\rho = \prod_{i\in I} \rho_i \bmod N. \tag{5}$$

③ 生成验证标签. 依据挑战数据块对应的数据标签生成验证标签 T：

$$T = \prod_{k\in C} T_k \bmod N. \tag{6}$$

④ 生成应答信息. Collector 将最终证据 (ρ, T) 作为应答消息返回给用户.

Collector 的流程伪代码如图 4 所示.

3) 证据验证阶段

首先用户从 Collector 端接收最终证据 (ρ, T)，然后使用自己的私钥计算 ρ^{sk}，并与 T 进行比较. 如果两者相同，则返回 TRUE 表示挑战成功；否则，挑战失败. 主要过程如图 5 伪代码所示.

```
For each i∈I, do
Step1. Receive ρ_i from S_i;
Step2. ρ=ρ×ρ_i mod N;
Step3. For each k∈C, generate validating tag T
         T=T×T_k mod N;
Step4. Send (ρ,T) to User.
```

Fig. 4 The pseudo-code of storage server side at replica challenging phrase.

图 4 多副本持有性挑战阶段各存储节点执行的伪码描述

```
Step1: Receive (ρ, T) from Collector;
Step2: If ρ^{sk}=T return TRUE;
       Else return FALSE.
```

Fig. 5 The pseudo-code of user side at evidence validating phrase.

图 5 用户端证据验证阶段伪代码描述

3.3 访问与更新机制

1) 数据访问与共享机制

当用户提出数据 b 的访问请求时，Collector 依据当前系统状态从副本集合 I 中选择访问第 x 个副本节点，于是 S_x 返回 $M_x=\{m_{x,1}, m_{x,2}, \cdots, m_{x,r}\}$. 接着用户针对每一个数据分块分别计算 $b_j = m_{x,j}/\psi_\kappa(x,j)$，即可得到 b.

当其他共享用户希望访问该数据时，必须得到该用户的授权，方法是通过某种秘密途径获得密钥 κ 即可.

2) 数据动态更新机制

① 数据块修改. 假设第 y 块数据发生了变化，则需要重新计算并更新所有的副本第 y 个数据块的内容（其他数据块不变）. 对 I 中第 i 个节点而言，第 y 个数据块内容为 $m'_{i,y}=b_y\times\psi_\kappa(i,y)$.

② 数据块删除. 假设第 y 块数据被删除，则只需要对 I 中所有副本节点 i 上删除对应的块 $m_{i,y}$，挑战和验证时的操作无需作任何修改.

③ 数据块插入. 当需要在原始数据中插入新的数据块 b_y 时，只需要对节点 i 生成新的副本数据块 $m_{i,y}=b_y\times\psi_\kappa(i,y)$，同时计算和上传对应的数据标签（其他操作不变）：

$$T_y = b_y \times \prod_{i\in I} \psi_\kappa(i,y)^{sk\times e_i} \bmod N.$$

4 PKP 方案分析

4.1 方案正确性证明

定理 1. 在 MRP-PKP 方案中，如果任何副本均未经修改，则 $\rho^{sk}=T$ 成立.

证明. 首先分析用户端接收到的最终证据 ρ:

$$\rho=\prod_{i\in I}\rho_i \bmod N=\prod_{i\in I}\prod_{k\in C}m_{i,k}^{e_i} \bmod N=$$
$$\prod_{i\in I}\prod_{k\in C}(b_k\psi_\kappa(i,k))^{e_i} \bmod N=$$
$$\prod_{i\in I}\prod_{k\in C}(b_k)^{e_i}\prod_{i\in I}\prod_{k\in C}(\psi_\kappa(i,k))^{e_i} \bmod N=$$
$$(\prod_{k\in C}b_k)^{\sum_{i\in I}e_i}\prod_{i\in I}\prod_{k\in C}(\psi_\kappa(i,k))^{e_i} \bmod N=$$
$$(\prod_{k\in C}b_k)^{pk}\prod_{i\in I}\prod_{k\in C}(\psi_\kappa(i,k))^{e_i} \bmod N.$$

因此:

$$\rho^{sk}=[(\prod_{k\in C}b_k)^{pk}]^{sk}[\prod_{i\in I}\prod_{k\in C}(\psi_\kappa(i,k))^{e_i} \bmod N]^{sk}=$$
$$(\prod_{k\in C}b_k)^{pk\times sk}\prod_{i\in I}\prod_{k\in C}(\psi_\kappa(i,k))^{sk\times e_i} \bmod N=$$
$$(\prod_{k\in C}b_k)\prod_{i\in I}\prod_{k\in C}(\psi_\kappa(i,k))^{sk\times e_i} \bmod N.$$

另一方面,用户端计算的数据块标签乘积 T 为

$$T=\prod_{k\in C}T_k \bmod N=$$
$$\prod_{k\in C}(b_k\prod_{i\in I}\psi_\kappa(i,k)^{sk\times e_i}) \bmod N=$$
$$(\prod_{k\in C}b_k)(\prod_{k\in C}\prod_{i\in I}\psi_\kappa(i,k)^{sk\times e_i}) \bmod N.$$

可见,T 与 ρ^{sk} 完全相等. 证毕.

4.2　方案安全性说明

1) 数据保密能力

在 MRP-PKP 方案中,云存储服务端仅掌握公钥 pk、各副本服务器份额集合 E 和所有的副本数据块 $m_{i,j}$. 原始数据 b_j 转换为大整数,与伪随机数 $\psi_\kappa(i,j)$ 相乘后得到 $m_{i,j}$,实际上是对数据进行了加密. 在不能获悉 κ 的情况下,云存储服务端无法逆向求得原始数据 b_j,因此可获得一定的保密性. 但这种加密比较容易破解,存在安全隐患. 在保密性要求较高的情况下,方案可以进行如下改进:在数据分块后可以对所有分块使用某种密码算法 E 进行加密处理,然后将密文 $E(b_j)$ 视为新的数据分块即可提供足够的保密性.

在需要对共享用户提供数据共享时,用户需要提供密钥 κ. 但此密钥只对该文件有效,共享用户并不能据此解密其他文件,保证了其他文件的保密性.

与其他方案相比,MRP-PKP 方案的另一个突出优势在于:该方案只允许用户本人验证副本的持有性;任何第三方因为没有 sk,因此不能验证副本的持有情况.

2) 防伪造能力

在 MRP-PKP 方案中,云存储端掌握公钥份额集合 E 和所有的副本数据块 $m_{i,j}=b_j\times\psi_\kappa(i,j)$. 在不能获悉 κ 的情况下,云存储端无法逆向分解求得原始数据 b_j,更不可能重构某个副本的数据块. 可见本方案的副本只能在用户的控制下产生,包括存储服务提供商在内的任何人都无法生成新的副本或者恢复原有副本内容,因此具有较强的防伪造能力.

3) 抗同谋攻击能力

为了欺骗用户,敌手 $\mathcal{A}$ 可以预先计算并保存所有可能的证据 ρ,在用户端发起挑战并提供挑战索引集合 C 时出示相应证据. 但敌手 $\mathcal{A}$ 无法事先知道用户将挑战哪些数据分块,而所有可能证据个数为:$C_t^1+C_t^2+\cdots+C_t^t$,无论从计算还是从存储角度来看,这显然都是不可行的.

下面重点考虑在第 i 个副本节点内容被篡改甚至整个副本丢失的情况下,敌手 $\mathcal{A}$ 是否能够构造虚假证据 ρ' 和验证标签 T',使得 $\rho'^{sk}=T'$,从而通过通过用户的挑战. ①$\mathcal{A}$ 不能得到 sk. 假设云存储服务端 Collector 和各个存储节点之间可以充分交流所有信息,$\mathcal{A}$ 只能从 N 和所有公钥份额 e_i 通过等式 $\sum_{i=1}^{r}e_i\equiv pk \bmod \varphi(N)$ 尝试得到 pk. 显然,通过 pk 无法得到对应的 sk. ②$\mathcal{A}$ 不能获得原始数据块内容 b_j. 由于 $\mathcal{A}$ 不能获悉密钥 κ,因此不能通过保存的副本分块 $m_{i,j}=b_j\times\psi_\kappa(i,j)$ 计算获得 b_j. 由于以上 2 个原因,$\mathcal{A}$ 不能伪造丢失的分证据 $\rho_i'=\prod_{k\in C}m_{i,k}^{e_i} \bmod N=\prod_{k\in C}(b_j\times\psi_\kappa(i,j))^{e_i} \bmod N$,更不能伪造 ρ' 和 T'. 因此,该方案具有较好的抗同谋攻击能力.

4.3　方案效率分析

从计算量来看,MRP-PKP 方案在用户端的工作主要集中在副本数据块计算的式(1)、数据标签计算的式(3),分别需要 $r\times j$ 次循环,每次循环需要计算不超过 l 位的 2 个整数乘法;而各存储节点的工作是按照式(4)计算分证据 ρ_i,这需要 c 次模 N 乘法;Collector 需要按照式(5)计算最终证据 ρ,这需要 r 次模 N 乘法. 由于一次挑战即可完成对所有副本持有性的验证,因此平均每个副本持有性验证的计算开销仅为 j 次 l 位的整数乘法和不超过 2 次模 N 乘法. 从通信量来看,同样由于一次挑战验证所有副本持有性,MRP-PKP 方案具有明显的优势. 可见,与现有方案[14,16]相比,本方案减少了模幂运算的数量和网络通信的次数,具有较大的计算优势和一定的通信优势.

此外,本方案还可以进一步改进. 将秘密份额生成函数中的欧拉函数 $\varphi(N)=4p'q'$ 改为 Carmichael 函数 $\lambda(N)=2p'q'$,可进一步减小计算量并加快

公钥份额集合 E 的生成. 可以证明:改进后的方案仍然能保证验证的正确性,并具有与原方案完全相同的安全性. 这里限于篇幅不再赘述.

5 实验与分析

5.1 挑战数据块数

在挑战阶段,用户选定某个可接受的置信度$\varepsilon\in(0,1)$和可接受的修改比例 η 可以求得需要挑战的数据块数 c,这里说明其求解过程.

假设数据总共分为 t 块,其中 $X=\eta\times t$ 块(如果不为整数则取整)被篡改,则从中挑选出 c 块挑战且发现篡改的概率为

$$P_X = P\{X\geqslant 1\} = 1-P\{X=0\} = 1-\mathrm{C}_{t(1-\eta)}^{c}/\mathrm{C}_{t}^{c} = 1-\frac{(t-\eta\times t)(t-1-\eta\times t)\cdots(t-c+1-\eta\times t)}{t(t-1)\cdots(t-c+1)},$$

而$\frac{t-\eta\times t}{t}<\frac{t-1-\eta\times t}{t-1}<\cdots<\frac{t-c+1-\eta\times t}{t-c+1}$,因此得到:$P_X>1-(1-\eta)^c$. 为了使 P_X 能满足用户的概率要求,即 $P_X\geqslant\varepsilon$,令其下界 $1-(1-\eta)^c=\varepsilon$,从而求得:$c=\frac{\ln(1-\varepsilon)}{\ln(1-\eta)}$. 显然,如果$\frac{\ln(1-\varepsilon)}{\ln(1-\eta)}>t$,则需要挑战所有 t 块数据块,因此挑战块数的计算公式为:$c=\min\left(\left|\frac{\ln(1-\varepsilon)}{\ln(1-\eta)}\right|,t\right)$.

图 6 显示了在数据块修改比例 η 分别为 1%,2%,5%,10%的情况下对应的挑战块数 c 随置信度 ε 的变化情况(假设总块数 $t>500$):

从图 6 中可以看到,在 η 相同的前提下,c 随着 ε 的提高而迅速增加;而在 ε 相同的前提下,η 越大,则需要挑战的块数越少,即服务端的修改越容易被用户发现. 假设用户需要以置信度 $\varepsilon=99\%$的概率确定副本是否被修改,如果文件被修改 1%,则约需要挑战 458 块即可发现此次修改;若 $\eta=5\%$,则只需要挑战 90 块;当修改比例达到 $\eta=10\%$时,则只需要挑战 44 块.

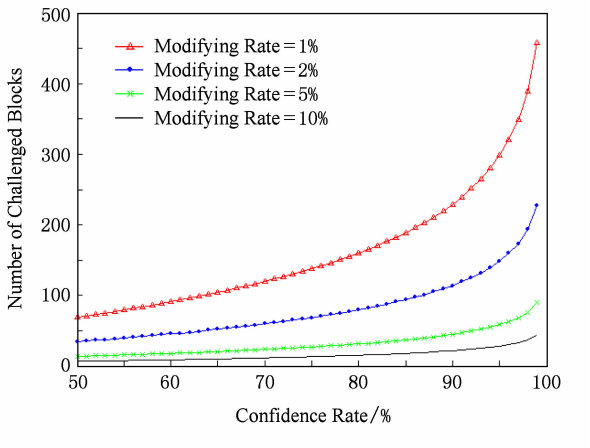


Fig. 6 The block numbers change with different modifying proportion.

图 6 不同修改比例下挑战块数随置信度变化情况图

可见,当文件块数较大时,需挑战块数与文件总块数并无直接关系;文件越大、分块越多,则在相同置信度下需要挑战的数据块数占所有块数的比例将越小. 在云存储环境下往往保存海量的大文件,因此与传统方式相比,MRP-PKP 方案的效率优势非常明显.

5.2 实验环境搭建

为了检验方案的正确性与验证效率,搭建了一个基于 Hadoop[19] 的云存储实验系统. 该系统由 1 个 Collector(由 NameNode 充当)和 15 个存储节点(由 DataNode 充当)组成,用户通过一个用户终端

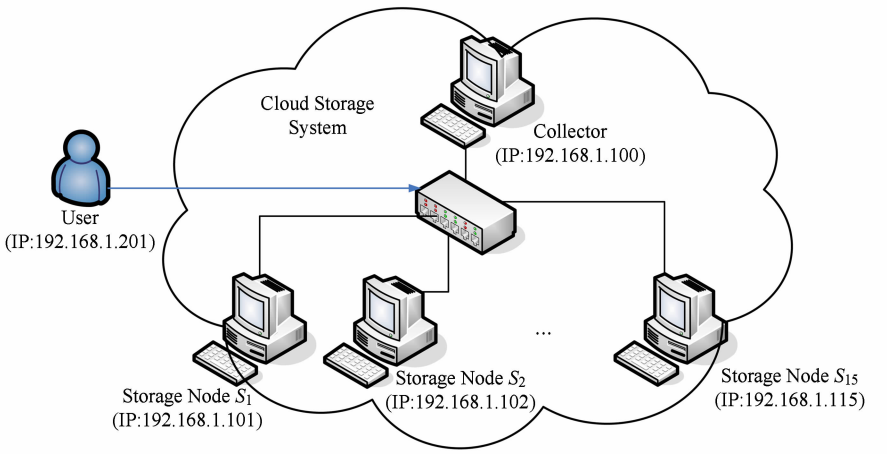


Fig. 7 The topology of the experimental system.

图 7 实验系统拓扑结构

访问该系统.其网络拓扑结构如图 7 所示.

云存储服务端节点(包括 1 台 Collector 和 15 台 Storage Server)采用 IBM 机架式服务器;客户端采用普通 PC 机,使用 Java 编写了用户端程序;所有节点连接在一台网络交换机上构成一个百兆局域网,如表 1 所示:

Table 1　The Parameters of Both Server and Client

表 1　服务端和客户端节点配置表

Device Type	IBM Server	PC
OS Type	Ubuntu10.0.2	Windows XP
CPU	8	2
Memory/GB	4(DDR3)	1
Hard Disk/GB	146×2(SAS)	160
Network Adapter/Mbps	100	100

5.3　实验结果与分析

在上述实验系统中编程实现了 MRP-PKP 方案,并同时模拟实现了 MR-PDP[13] 方案.用户端密钥和模数 N 的位数均为 512 b;所有实验执行 10 次,取 10 次执行结果的平均值为有效数据.

1) 存储开销

从生成副本分块的大小来看,MR-PDP 方案在原数据块上与一个随机掩码相加,基本不增加数据容量;而 MRP-PKP 方案则是在原数据块上与一个随机掩码相乘,每个副本块在原来基础上增加不到 16 B 的容量.在使用较大数据分块时,这种增加是可以容忍的.例如当数据分块取 512 B 时,实际上只增加了约 3%的容量.当采用更大分块时,这种增量将更低.

2) 通信开销

在副本存储阶段,MR-PDP 与 MRP-PKP 方案都需要将副本发送给云存储服务端,如式(1)可知,MRP-PKP 方案中副本的容量略微增加.此外,用户需要给每个副本节点发送一个公钥的私有份额,约为 128 b.在挑战和验证阶段,为了验证 r 个副本的存在,MRP-PKP 方案需要从服务端发送一次 ρ 和 T 到用户端,只需要传输 2 个 512 b 的消息;而 MR-PDP 方案则需要逐一验证每个副本的存在,每次都需要传输 2 个 512 b 的消息.因此 MRP-PKP 方案的通信开销仅为 MR-PDP 方案的 $1/r$.

3) 计算开销

① 副本生成开销测试

首先测试在不同数据块大小下采用两种方案对不同大小的数据文件生成副本的开销.不失一般性,这里设定每个数据块大小为 512 B,副本数为 5,测试结果如表 2 所示:

Table 2　The Replica Generating Time Cost of Both Schemes

表 2　不同算法副本生成开销比较　ms

Block Size	Algorithm	
	MR-PDP	MRP-PKP
1 KB	0.014 3	0.039 0
10 KB	0.125 7	0.355 9
100 KB	7.273 6	5.021 3
1 MB	20.402 8	50.314 7
10 MB	251.350 0	556.949 3
100 MB	2 654.169 2	4 683.853 1

可见,由于 MR-PDP 方案是使用数据块对应的大整数与一个随机掩码相加;而 MRP-PKP 方案是使用数据块对应的大整数与一个随机掩码相乘,因此对于相同大小的数据文件,MR-PDP 方案比 MRP-PKP 方案略快,但总体仍处于同一个数量级.在使用其他数据块大小时,情况基本与表 2 类似.

② 数据标签生成开销测试

分别测试在不同数据块大小下采用 2 种方案对不同大小的数据文件生成数据标签的时间开销,如表 3、表 4 所示:

Table 3　The Data Tag Generating Time Cost of MR-PDP

表 3　MR-PDP 方案数据标签生成开销　ms

File Size	Block Size				
	128 B	256 B	512 B	1 KB	2 KB
1 KB	0.016 0	0.015 1	0.095 93	0.089 6	0.089 7
10 KB	1.261	1.097	1.058	0.953	0.916
100 KB	13.538	10.638	9.473	8.670	8.347
1 MB	118.496	98.396	89.834	83.230	77.558
10 MB	1 192.46	967.864	867.832	815.30	726.26
100 MB	12 469.5	9 823.70	9 045.73	8 526.1	7 838.8

Table 4　The Data Tag Generating Time Cost of MRP-PKP

表 4 MRP-PKP 方案数据标签生成开销　ms

File Size	Block Size				
	128 B	256 B	512 B	1 KB	2 KB
1 KB	0.014 2	0.014 7	0.003 8	0.006 1	0.003 3
10 KB	0.267 7	0.131 8	0.086 1	0.024 9	0.010 4
100 KB	1.544 6	0.673 2	0.410 9	0.306 3	0.168 9
1 MB	15.029	7.689 6	6.376 8	2.075	1.009 2
10 MB	103.764	71.931	35.494	28.723	17.039
100 MB	116.403	725.85	462.875	322.26	268.92

可见,对于同一种方案而言,随着分块大小增加,需要划分的数据块更少,于是需要生成的数据标签数目更少,所以数据标签生成开销随之减小.对于相同分块大小,随着文件大小的增加,数据标签生成开销也相应增加.测试数据表明,这种增加大致呈线性趋势.

对比表 3 和表 4 则可以发现,在相同分块大小和数据文件大小的前提下,MRP-PKP 方案比 MR-PDP 方案所需时间开销更少.这是因为 MR-PDP 方案中标签计算公式为[13]

$$T_j = (h(W_i) \times g^m)^d \bmod N. \quad (7)$$

实验结果表明:MRP-PKP 方案使用的式(3)相比式(7),其运算效率约快 10 倍.

③ 挑战与验证开销测试

不失一般性,取置信度 $\varepsilon=95\%$,副本数 $r=5$,数据块修改比例为 $\eta=5\%$.根据第 5.1 节挑战块数的计算公式,在这种条件下每次需要挑战 min(59, t)个数据块.分别测试在不同数据块大小下,采用 2 种方案对不同大小的数据文件进行测试,分别记录证据生成与完成验证所需的总时间开销.挑战测试结果如表 5、表 6 所示:

Table 5 The Challenging Time Cost of MR-PDP

表 5 MR-PDP 方案副本挑战开销 ms

File Size	Block Size				
	128 B	256 B	512 B	1 KB	2 KB
1 KB	534.86	327.91	321.82	263.42	155.02
10 KB	3 112.1	2 740.0	1 657.4	1 073.5	821.40
100 KB	2 548.2	2 386.3	2 371.3	2 437.4	2 415.7
1 MB	2 837.6	2 664.7	2 874.6	2 707.7	3 732.5
10 MB	3 545.1	2 848.9	3 147.6	2 582.8	3 974.6
100 MB	3 530.8	3 374.5	2 750.9	2 979.4	3 483.3

Table 6 The Challenging Time Cost of MRP-PKP

表 6 MRP-PKP 方案副本挑战开销 ms

File Size	Block Size				
	128 B	256 B	512 B	1K B	2 KB
1 KB	404.022	298.263	200.183	87.655	88.903
10 KB	459.782	475.134	333.700	367.068	245.27
100 KB	558.302	512.141	692.967	608.169	723.08
1 MB	580.611	529.574	634.053	594.850	639.99
10 MB	647.127	594.850	587.886	633.236	673.93
100 MB	670.185	575.854	614.542	608.678	702.28

可见,对于同一种方案而言,挑战开销随着分块大小增加而减少,特别是在文件大小较小的时候尤为明显,这是因为需要挑战的数据块更少.而当数据块大于 60 后,需要挑战的块数稳定在 59 块,因此挑战开销基本趋于稳定:MR-PDP 方案约在 2 500～3 500 ms 之间;而 MRP-PKP 方案约在 500～700 ms 之间.

对比表 5 和表 6 则可以发现,MRP-PKP 方案验证所有副本存在所需时间开销远比 MR-PDP 方案验证副本开销所需时间开销少.当挑战块数较多、甚至稳定为 59 块时,MR-PDP 方案所需时间约为 MRP-PKP 方案的 $r=5$ 倍.进一步实验证明了更大的副本数 r 将导致两者之间更大的差距.

从整个测试的结果来看,与 MRP-PKP 方案相比,MRP-PKP 方案通过增加一部分存储开销(副本块容量增加约 3%和 128 b 公钥份额),换取了通信开销和验证计算开销上 r 倍的提高.

6 结 论

在云存储环境中,数据不再置于其拥有者的直接控制之下,而是被云存储服务提供商所托管;如何验证服务方是否忠诚地按照客户需求存储足够数量的副本数据是一个挑战性问题.通过分析发现:现有方案大多存在验证效率低、通信开销大和对数据更新支持弱等不足.为此,提出一种带 Collector 的多副本云存储模型,并利用公钥分割给出一种新的多副本持有性证明方案 MRP-PKP.该方案的突出优势在于可以一次验证所有副本的持有性.虽然该方案牺牲了少量的存储开销,但是节约了通信开销并获得了验证效率的大幅提升.此外,该方案能够防御同谋攻击,可方便地支持更新、插入、删除等数据块级的更新操作,并可以直接应用于 Map/Reduce 模式以提高效率,具有较好的理论价值和应用前景.

参 考 文 献

[1] Fu Yingxun, Luo Shengmei, Shu Jiwu. Survey of secure cloud storage system and key technologies [J]. Journal of Computer Research and Development, 2013, 50(1): 136-145 (in Chinese)
(傅颖勋, 罗圣美, 舒继武. 安全云存储系统与关键技术综述[J]. 计算机研究与发展, 2013, 50(1): 136-145)

[2] Mahajan P, Setty S, Lee S, et al. Depot: Cloud storage with minimal trust [C] //Proc of OSDI'10. Berkley, CA: USENIX Association, 2010: 307-322

[3] Chen Kang, Zheng Weimin. Cloud computing: System instances and current research [J]. Journal of Software, 2009, 20(5): 1337-1348 (in Chinese)
(陈康,郑纬民. 云计算:系统实例与研究现状[J]. 软件学报, 2009, 20(5): 1337-1348)

[4] Amazon Web Service, Inc. Amazon simple storage service (Amazon S3) [EB/OL]. [2014-04-22]. https://s3.amazonaws.com/

[5] Kamara S, Lauter K. Cryptographic cloud storage [G] // LNCS 6054: Financial Cryptography and Data Security. Berlin: Springer, 2010: 136-149

[6] Ari J, Burton K. PORs: Proofs of retrievability for large files [C] //Proc of the 14th ACM CCS. New York: ACM, 2007: 584-597

[7] 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): 12-34

[8] Ateniese G, Pietro D, Manceici V. Scalable and efficient provable data procession [C] //Proc of SecureComm'08. New York: ACM, 2008: 1-10

[9] Erway C, Kupcu A, Papamanthou C, et al. Dynamic provable data procession [C] //Proc of the 16th ACM Conf on Computer and Communications Security. New York: ACM, 2009: 213-222

[10] 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

[11] Yun A, Shi C, Kim Y. On protecting integrity and confidentiality of cryptographic file system for outsourced storage [C] //Proc of ACM Workshop on Cloud Computing Security. New York: ACM, 2009: 67-76

[12] 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 Conf on Research in Computer Security. Berlin: Springer, 2009: 355-370

[13] Curtmola R, Khan O, Ateniese G. MR-PDP: Multiple replica provable data possession [C] //Proc of ICDCS 2008. Piscataway, NJ: IEEE, 2008: 411-420

[14] Bowers D, Juel A, Oprea A. HAIL: A high-availability and integrity layer for cloud storage [C] //Proc of CCS 2009. New York: ACM, 2009: 187-198

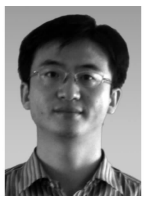
[15] Hao Zhuo, Yu Nenghai. A mulitple-replica remote data possession checking protocol with public verifiability [C] // Proc of the 2nd Int Symp on Data, Privacy and E-Commerce. Piscataway, NJ: IEEE, 2010: 84-89

[16] He Jing, Zhang Yanchun, Huang Guangyan, et al. Distributed data possession check for securing multiple replicas in geographically-dispersed clouds [J]. Journal of Computer and System Sciences, 2010, 78: 1345-1358

[17] Barsoum A, Hasanm A. On verifying dynamic multiple data copies over cloud servers [EB/OL]. (2011-08-15) [2014-04-22]. IACR Cryptology ePrint Archive. https://eprint.org/2011/441.pdf

[18] Fu Yanyan, Zhang Min, Chen Kaiqu, et al. Proofs of data possession of multiple copies [J]. Journal of Computer Research and Development, 2014, 51(7): 1410-1416 (in Chinese)
(付艳艳, 张敏, 陈开渠, 等. 面向云存储的多副本文件完整性验证方案[J]. 计算机研究与发展, 2014, 51(7): 1410-1416)

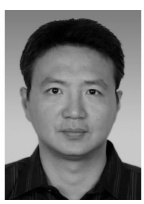
[19] Tan Jian, Meng Xiaoqiao, Zhang Li. Performance analysis of coupling scheduler for MapReduce/Hadoop [C] //Proc of INFOCOM'12. New York: IEEE Communication Society, 2012: 2586-2590



Fu Wei, born in 1978. PhD. Member of China Computer Federation. His main research interests include cloud computing & cloud security, distributed computing and information security.

Wu Xiaoping, born in 1961. PhD, professor and PhD supervisor. His research interests include information security and applicable mathematics.

Ye Qing, born in 1978. PhD and associate professor. Member of China Computer Federation. His main research interests include network security and information security.



Xiao Nong, born in 1969. PhD, professor and PhD supervisor. Senior member of China Computer Federation. His research interests include high-performance computing and massive storage.

Lu Xicheng, born in 1946. Professor and PhD supervisor. Academician of Chinese Academy of Engineering. His research interests include computer network, parallel and distributed computing.