基于改进跳表的云端数据完整性验证协议

王瑞锦，张凤荔，王馨云，陈学勤，罗 昊，秦圣智

(电子科技大学信息与软件工程学院 成都 610054)

【摘要】在云存储应用中，用户数据的完整性是用户最关心的问题之一，用户提交到云存储服务提供商处进行在线存储的文件面临着丢失以及被篡改的风险，因此用户需要通过使用某种技术手段对从云端取回的数据进行完整性的验证，以确定正在访问的数据是完整和正确的。考虑到在云存储系统的应用环境中用户计算资源受限的特点和云存储的安全需求，基于改进的跳表和短签名技术，该文提出一种能够对云端数据的动态操作提供良好支持的完整性验证协议。在跳表中引入可达范围计数以便高效地支持数据块在任意位置的插入或者删除操作，有效减少了执行动态操作时产生的计算开销。通过性能分析与实验比较，证明该协议是高效的。

关 键 词 云存储; 动态操作; 安全协议; 跳表; 完整性; 动态操作

中图分类号 TP309 **文献标志码** A doi:10.3969/j.issn.1001-0548.2018.01.013

A Cloud Data Integrity Verification Protocol Based on Improved Skip Lists

WANG Rui-jin, ZHANG Feng-li, WANG Xin-yun, CHENG Xue-qin, LUO Hao, and QING Sheng-zhi

(School of Information and Software Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract The integrity of user's data is one of the problems which is most concerned when users using cloud storage service to store their data and files. The data users commit to cloud storage servers may be lost and modified. Users need a technology to check the integrity and validity of their data and files when accessing them. By considering the limit of the calculation resource of users and the security request of cloud storage service, an integrity verification protocol based on improved skip lists and short signature is proposed, which can support dynamic operation. By combining accessible counting and skip lists, this protocol can support inserting and deleting data block at any position efficiently. The analysis and experiment show that the proposed protocol is security and efficient.

Key words cloud storage; dynamic operation; integrity; security protocol; skip lists

随着计算机技术和互联网的不断发展，在网络中传输的信息出现了爆炸式的增长，海量的数据量给存储能力和成本带来了巨大的压力[1]。在这种情况下云存储技术应运而生，云存储基于云计算，利用分布式计算和集群应用等技术，将大量设备整合起来，为用户提供数据存储和访问功能[2]。云存储具有成本低廉、不受地理位置限制的资源访问、易于管理和扩充等优点[3]，被视为IT企业的下一代数据存储模型[4]。

用户使用云存储服务将文件提交到云服务提供商(cloud service provider, CSP)进行在线存储，在减轻本地存储压力的同时，也失去了对数据的物理控制能力[5]，因此数据的安全问题[6-8]是一个不容忽视的问题。CSP可能是不诚实的[9]，会为了自身的利益威胁到用户的数据安全，例如当CSP的存储设备容量不足时，CSP可能会选择删除一些用户长时间没有访问过的数据或者将这些数据移动到廉价的离线存储介质中以节省开支[10]，这会造成用户数据的永久或暂时不可访问。另外在遭遇拜占庭失败时CSP可能会为了维护自己的信誉而试图向用户隐瞒数据丢失的情况[11]。

目前已经出现了多种针对云存储环境的数据完整性验证方案。一个实用的云存储完整性验证算法应该具有较小的网络开销，并且没有验证次数的限制，另外由于用户存在对云端的数据进行操作的需求，因此一个实用的完整性验证方案还要能支持动

收稿日期：2016 – 09 – 08；修回日期：2017 – 06 – 01

基金项目：国家自然科学基金(61472064)；中央高校基本科研基金(ZYGX2014J051, ZYGX2014J066)；四川省科技厅项目(2015JY0178, 2016ZC2575, 2014GZ0109, 2015KZ002, 2015JY0030)；四川省教育厅重点项目(17ZA0322)

作者简介：王瑞锦(1980 –)，男，博士，主要从事信息系统与内容安全、量子通信安全等方面的研究.

态操作。文献[12]提出了两种针对云存储的完整性验证算法，然而这两种方法均不能对文件的动态操作提供支持；文献[13]提出了一种基于Diffie-Hellman体制的完整性验证方法，该方法对验证次数没有限制，但是不能支持数据块的插入操作并且计算开销较大；文献[10]提出了一种基于merkel hash tree(MHT)的完整性认证方案，在该方案中，CSP为产生一次完整性验证需要读取所有的数据块以得到Merkel哈希树，当用户数据较大或数据块较小时，会使得Merkel哈希树的深度过深而消耗较多的计算资源，影响CSP的服务质量；文献[14]提出了一种基于改进的Merkel哈希树和双线性对的完整性审计方案，在该方案中按照两个层级对用户数据进行划分，使Merkel哈希树中的每一个叶子结点对应多个数据块，从而降低Merkel哈希树的深度，提高构造效率，同时该方案也能够支持数据块的动态操作。然而在极端情况下，例如用户不断在文件的末尾追加数据块，则会出现文献[14]中提到的树不平衡的情况，从而使根结点到某个叶子节点的路径过长，效率将会在很大程度上降低。

为了解决文献[14]中提出的方案在上述情况下的效率问题，本文基于改进的跳表和短签名技术，提出一种能够支持动态操作的完整性验证协议。本文对跳表中的结点提出一种新的定义方式，该定义方式可以在逻辑上消除跳表中的环状结构，从而降低跳表中结点的依赖关系数量，在生成跳表、更新跳表和进行验证时降低计算开销；同时本文在跳表中引入可达范围计数，使得在发生数据块的插入和删除时，跳表中需要更新的结点数量更少，从而更有效地对动态操作提供支持。最后对该方案进行了理论分析和实验验证，证明该方案是高效的。

1　理　论

1.1　系统模型

针对云存储的完整性验证系统通常可以分为两方模型和三方模型，其中两方模型中包含用户(Client)和云存储服务提供商(CSP)，三方模型在两方模型的基础上引入了一个可信的第三方验证者(TPA)，TPA用于为用户承担验证任务，代替用户周期性的对存储在CSP中的数据和文件进行完整性验证。本文提出的方案基于两方模型，如图1所示。

图中用户指有着数据在线存储需求的云存储服务用户；云存储服务器是由云存储服务提供商管理的一组服务器，具有海量的存储能力和强大的计算能力，对外提供数据存储和访问功能。在本文提出的方案中，用户在对文件进行访问时，同时获取与之对应的辅助认证信息，使用辅助认证信息对正在访问的文件进行完整性验证。本文中对文件的分块处理与文献[14]类似，即按照固定的块大小对文件进行划分，产生认证数据的基本单位是数据块。

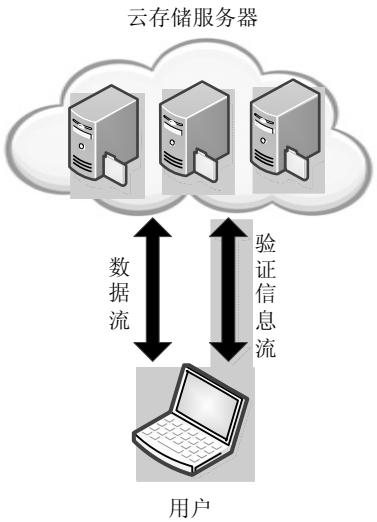

图1　两方模型

在本文提出的协议中，用户将数据提交到云端进行在线存储前，首先需要在本地对文件进行一些处理工作，包括文件的块划分、摘要生成、跳表生成、签名等，完成上述步骤后，用户将文件和辅助认证信息一并提交到云端进行保存，之后用户便删除本地的文件副本和辅助验证信息副本，以减轻本地的存储压力。用户在访问存储在云端的文件时，获取需要访问的部分以及该部分对应的辅助认证信息，使用辅助认证信息对该部分的文件内容进行验证，以便确定该部分文件内容是否正确和完整。

1.2　跳表

跳表(skip lists)是一种能够快速进行查找、删除和插入的数据结构，跳表的结构如图2所示。

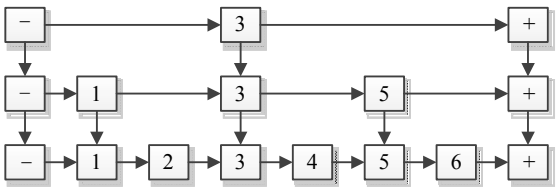

图2　跳表的结构

跳表包含若干层，其中每一层均由链表构建，跳表中的每一个结点均包含一个指向右边结点的指针和一个指向下方结点的指针，跳表中最左边一列和最右边一列的结点被称为哨兵结点。跳表中的底层链表有序地存储所有元素，对于某一层 S_i 和 S_i 的

上一层 S_{i+1}，出现在 S_i 中的非哨兵结点将以某个概率出现在上一层中，即以一定概率在 S_{i+1} 层中生成对应结点，并且哨兵结点所在列的层数不小于任何非哨兵结点所在列。

在跳表中进行查找，从左上角的结点开始，用p表示指向当前结点的指针，p只能向右或者向下移动，查找某个结点的算法描述如下：

```
found=false;
while(p != null)
begin
    if(p==target)
    begin
        found=true;
        break;
    end
    else if(p.right!=guard && p.right < target)
        p=p.right;
    else
        p=p.down;
end
```

查找过程结束时，p指向目标结点或者小于目标结点的最大结点。

在跳表中插入一个结点时，例如需要插入的结点编号为x，则首先执行查找x的操作，得到一个指向小于x的最大结点的指针，此时再将x结点插入到p所指向的结点的后面，并概率性地产生x结点的上层结点。

执行删除操作时，首先执行查找操作找到小于目标结点的最大结点所在位置，之后删除目标结点以及目标结点所在列的所有结点，该过程与在链表中删除一个结点类似。

1.3 可交换哈希函数

标准的哈希函数只有一个输入参数，可交换哈希函数是一种包含多个输入参数的哈希函数，并且当输入参数均保持不变时，任意交换各个参数的输入顺序，函数的输出是一致的。包含2个输入参数的可交换哈希函数满足下述性质：

$$\mathrm{EH}(a,b)=\mathrm{EH}(b,a)$$

式中，EH是可交换哈希函数，本文使用的可交换哈希函数包含3个输入参数。

1.4 安全模型

在云存储系统中，由于云存储服务器是半可信的，因此用户存储在云端的文件面临着损坏、篡改甚至被云存储服务提供商有意删除等风险，而云存储服务提供可能会为了维护其自身利益而向用户隐瞒并且试图在文件的完整性被破坏后通过某种方法使得用户不能发现文件的完整性已经被破坏，所以该系统中可能会出现如下形式的攻击：

1) 伪造数据块的攻击。用户在访问某个存储在云服务器中的数据块时，如果云服务器发现该数据块存在丢失、部分丢失或被篡改的情况，云服务器可能会通过某种技术手段伪造一个数据块提供给用户，从而使得用户不能发现完整性已被破坏。

2) 伪造辅助认证信息的攻击。用户在访问某个存储在云服务器中的数据块时，如果该数据块或对应的辅助认证信息存在丢失、部分丢失或者被篡改的情况，云服务器可能会伪造一对对应的数据块和辅助认证信息提供给用户，从而使得用户不能发现完整性已被破坏。

2 协议的构造

2.1 改进的跳表

本文对跳表的改进主要有3点：

定义 1 定义跳表中的子结点。对于跳表中的结点i，定义其左边的结点j和下面的结点k为结点i的子结点，如果结点j是所在列的顶层结点，则称j是i的实子结点，否则称j是i的虚子结点。

定义 2 跳表中的底层非哨兵结点存储2个哈希值，分别称为contentHash和nodeHash。跳表中的其他结点只存储nodeHash，其中contentHash直接通过使用哈希函数对数据块计算摘要得到。

定义 3 跳表中的结点不再存储对应的数据块的索引，而是存储以该结点为祖先结点的底层结点数量。

根据定义1可知本文所使用的跳表的根结点为右边哨兵结点的顶层结点，该跳表的结构如图3所示，其中虚线框代表的是虚子结点，虚线箭头为父结点指向虚子结点的指针，方框内“+”和“−”为哨兵结点。

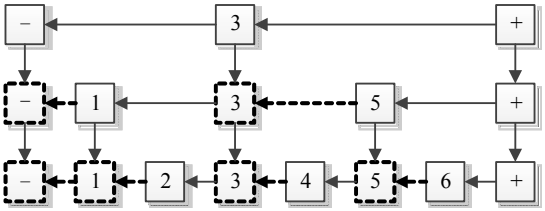

图3 改进的跳表结构

在该跳表中，底层结点中的contentHash用于存储数据块的摘要值。假设当前结点是i，i的左子结点是j，i的下子结点是k，计算i结点的nodeHash时，根

据下面所定义的依赖关系进行。

1) 若k==null，即i是底层结点。

①如果i是哨兵结点，并且j是虚子结点：

$$i.\text{nodeHash} = \text{EH}(\varphi,\varphi,\varphi)$$

②如果i是哨兵结点，并且j是实子结点：

$$i.\text{nodeHash} = \text{EH}(j.\text{nodeHash},\varphi,\varphi)$$

③如果i不是哨兵结点，并且j是虚子结点：

$$i.\text{nodeHash} = \text{EH}(\varphi,i.\text{contentHash},\varphi)$$

④如果i不是哨兵结点，并且j是实子结点：

$$i.\text{nodeHash} = \text{EH}(j.\text{nodeHash},i.\text{contentHash},\varphi)$$

2) 若k!=null，即i是非底层结点。

①如果j是虚子结点：

$$i.\text{nodeHash} = \text{EH}(\varphi,\varphi,k.\text{nodeHash})$$

②如果j是实子结点：

$$i.\text{nodeHash} = \text{EH}(j.\text{nodeHash},\varphi,k.\text{nodeHash})$$

式中，φ是一个特定的字符串。根据上述依赖关系可知，在本文所使用的跳表中，信息的流向所构成的结构为树状结构，并且该树状结构的根结点为跳表的右上方结点。

根据定义3，本文使用的跳表中不记录数据块在文件中的索引，而使用可达范围计数来确定跳表结点与数据块的对应关系，因此在某个位置新加入一个结点后，不需要对该结点之后的每一个结点进行修改，只需要修改从新加入结点到根结点这一条路径上所经过的结点的可达范围计数，该路径的平均长度为log(n)，因此能够更高效的对动态操作提供支持。使用可达范围计数标识数据块与结点对应关系的跳表结构如图4所示。其中椭圆型框图代表数据块，其中的数字代表数据块索引；跳表中各个结点中的数字代表可达范围计数。对于结点i以及其左子结点j和下子结点k，如果j是虚子结点，则i的计数等于k的计数；如果j是实子结点，则i的计数等于j与k的计数之和。

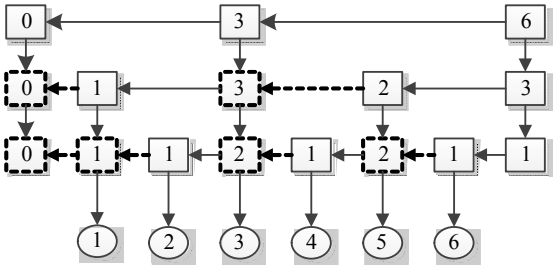

图4　使用可达范围计数的跳表结构

在图4所示的跳表中进行查找某个数据块对应的结点时，指针p开始时指向根结点，查找算法如下：

```
if(p.reachableCount<target)
    return null;
while(p.down != null)
begin
    if(p.left.isReal==false)//是虚子结点
        p=p.down;
    else if(p.left.reachableCount>=target)
        p=p.left;
    else
    begin
        target=target-p.left.reachableCount;
        p=p.down;
end
end
if(p.right==null)//是右边的哨兵结点
    p=p.left;
steps=p.reachableCount-target;
for(1:steps)
    p=p.left;
return p;
```

在执行查找操作时，可以使用栈来记录过程中经过的结点路径，使用该路径可以得到目标结点对应的辅助认证信息。在该跳表中执行数据块的插入和删除操作时，首先执行查找操作，其余部分与基础跳表的操作方法类似，在此不再赘述。

2.2　基本方案

假设用户需要提交到云服务器进行保存的文件是M，M包含n个数据块，即$M=\{m_1,m_2,\cdots,m_n\}$，基本方案由3个步骤组成。

1) $\text{SetUp}(1^\lambda)\rightarrow(H,\text{EH},\text{sk},\text{pk})$。用户选择随机数$x\in Z_p^*$，得到用户私钥$\text{sk}=(x,g)$，其中$g$是群$G$的生成元，$G$是基于椭圆曲线构造的乘法循环群，$G$的阶数为$p$。用户根据私钥计算公钥$\text{pk}=(y,g)$，其中$y=g^x$，根据双线性对的性质[15]，有等式$e(y,g)=e(g,g)^x=e(g,y)$成立。同时用户还需要选择在协议中使用的哈希函数和可交换哈希函数；

2) $\text{GenSL}(M,H,\text{EH},\text{sk})\rightarrow(\text{SL},\sigma)$。用户对文件$M$进行分块处理，得到数据块集合$M=\{m_1,m_2,\cdots,m_n\}$，并产生跳表SL，SL的底层结点数量为$n+2$，包含2个哨兵结点和$n$个非哨兵结点。之后用户使用哈希函数$H$计算数据块集合中每一个元素的摘要指，并将摘要值按顺序写入跳表底层非哨兵结点的contentHash中。最后用户使用可交换哈希函数EH计算SL中每一个结点的nodeHash值，

并使用私钥sk对根节点r的nodeHash值进行签名，得到签名值σ。完成上述步骤后，用户将(SL,σ)和文件M提交到云存储服务器进行在线存储；

3) $\mathrm{Verify}(m_i,\eta_i,\sigma,H,\mathrm{EH},\mathrm{pk})\rightarrow$ true/false。当用户访问数据块集合中的某个数据块m_i时，云存储服务器一并向用户返回数据块m_i、辅助验证信息η_i和根结点的签名σ。辅助验证信息是一个集合，包含从m_i所对应的跳表结点到根节点所在路径上的所需nodeHash，根据跳表的性质，该集合的大小平均为$\log(n)$，因此根据2.1节中定义的依赖关系，可以使用m_i和辅助验证信息η_i计算出根节点r'。用户计算出r'后，使用数字签名σ验证r'的合法性，如果验证通过则系统返回true，否则返回false。

2.3 动态操作

在本方案中，动态操作包括3种类型：数据块的修改(update)、插入(insert)和删除(delete)。假设认证结构SL和签名σ已经生成并提交到云存储服务器中进行保存，下面介绍动态操作过程。

1) 用户产生操作消息。用户执行算法GenOPMsg()，下面对不同的操作类型进行描述。

修改操作：假设用户对索引为x的数据块进行修改，得到新数据块m'_x，则用户产生消息(update,x)，并将消息发送至云存储服务器。

插入操作：假设用户需要在文件中的x位置新插入一个数据块m_x，则用户使用特定概率计算新插入数据块在跳表中对应的结点所在列的层数L，并将消息(insert,x,L)发送至云存储服务器。

删除操作：假设用户需要删除文件中的第x个数据块，则用户产生消息(delete, x)并发送至云存储服务器。

2) 更新SL与签名。云存储服务器在接收到用户的动态操作请求后，根据消息内容返回所需的辅助认证信息，以便用户对SL进行更新和对新的SL签名，下面详细介绍其过程。

修改操作：云存储服务器返回原数据块m_x所对应的辅助认证信息η_x。用户接收到η_x后，使用m'_x计算摘要值并将其作为第x个跳表结点的contentHash，再根据辅助认证信息η_x计算得到根节点的nodeHash，并对其签名。

插入操作：云存储服务器在SL中的第x个位置新插入一列结点，该列的高度为L。此时新插入结点的contentHash和nodeHash均为空值，然后执行查找算法find(x)并得到辅助认证信息η_x，云存储服务器将η_x返回给用户，用户使用m_x计算摘要值并将其作为第x个跳表结点的contentHash，再根据辅助认证信息η_x计算得到根节点的nodeHash，并对其签名。

删除操作：云存储服务器将目标结点x所在列删除，目标结点后面的结点将占据第x个位置。此时云存储服务器执行算法find(x)得到第x个节点的辅助认证信息η_x，并将η_x与删除第x个数据块后、在文件中的索引变更为x的数据块m_x返回给用户。用户根据m_x和η_x计算得到SL根结点的nodeHash并对其签名。

3) 提交更改。如果用户执行的是修改操作，则向云存储服务器提交(m'_x,σ')；如果执行的是插入操作，则向云存储服务器提交(m_x,σ')；如果执行的删除操作，则向云存储服务器提交(σ')。

2.4 安全性分析

本协议的攻击类型主要有数据块伪造攻击和辅助认证信息伪造攻击两种，下面分别对其进行简要的安全性分析。

1) 数据块伪造攻击。假设在用户对数据块m_x进行完整性验证时，云存储服务器伪造了一个数据块m'_x并且$m'_x\neq m_x$，云存储服务器向用户返回(m'_x,η_x,σ)，假设该三元组能够通过验证，则等价于云存储服务器有能力找到$m_1\neq m_2$，使得$\mathrm{Hash}(m_1)=\mathrm{Hash}(m_2)$，但根据哈希函数的单向性和抗碰撞性，这个问题是一个困难问题，因此上述三元组不能以不可忽略的概率通过用户的验证，本协议能够抵抗数据块伪造攻击。

2) 辅助认证信息伪造攻击。假设云存储能够用不合法的数据块m'_x代替合法数据块m_x计算出非法的SL根节点nodeHash并对其签名，并且该非法签名和非法的SL根结点nodeHash能够通过用户的公钥验证，则等价于云存储服务器能够在没有私钥的情况下产生合法签名，根据数字签名的定义，可知该问题是一个困难问题，因此云存储服务器无法以不可忽略的概率产生合法签名，本协议能够抵抗辅助认证信息伪造攻击。

3 性能分析

将本文算法与另外3种针对云存储环境的数据完整性验证方案进行性能对比，之后将本文的方案与文献[14]的方案进行模拟分析实验。本文所提出的协议与文献[14]中所提出的方案在初始化阶段均需生成辅助认证信息，而产生辅助认证信息的时间复

杂度均为$O(n)$，且产生的认证结构中所包含的节点数量的期望值均为文件块数量的2倍，因此该部分不做性能测试。

设n代表用户需要存储在云服务器中的文件包含的数据块数量，本文与PDP方案、文献[10]和文献[14]方案的性能分析对比结果如表1所示。

表1　性能分析比较

项目	方案			
	PDP[16]	文献[10]	文献[14]	本文方案
是否支持动态操作	否	否	是	是
用户存储开销	$O(1)$	$O(1)$	$O(1)$	$O(1)$
云服务器存储开销	$O(n)$	$O(n)$	$O(n)$	$O(n)$
云服务器计算开销	$O(1)$	$O(\log n)$	$O(1)$	$O(1)$
验证时的计算开销	$O(1)$	$O(\log n)$	$O(\log n)$	O($\log n$)

为支持动态操作，需要使用能够支持动态操作的认证性数据结构，本文方案使用的是前面介绍的跳表。在验证时，从根结点到目标结点的平均路径长度是log$n$，因此验证时的计算复杂度是$O(\log n)$，如表1所示。

下面将本文的方案与文献[14]中的方案进行模拟对比实验。实现环境的配置为：主频为3.2 GHz的双核CPU台式计算机，8 GB内存，64位Windows 7操作系统，编码所用的开发语言为Java，使用的密码学工具库为jPBC v2.0.0。

对于日志文件，在文件的尾部追加内容是一种常态操作。本文测试的场景为连续向文件尾部追加数据块，测试开始前用大小为1 GB的文件产生认证结构，文件块的大小设置为8 KB，通过在文件的尾部追加不同数量的文件块以造成认证结构出现不同程度的不平衡。测量在不同程度的不平衡状态下，为每一个追加的文件块进行结构调整所消耗的平均时间，测试结果如图5所示。

从理论分析而言，本文所提出的协议在执行尾部追加操作时，为每一个追加的文件块进行结构调整的平均时间复杂度为$O(\log n)$。从上述测试结果可知，在向文件尾部进行连续追加文件块的场景中，本文提出的协议性能优于文献[14]中的方案。

测试向文件中随机插入文件块，测试前先用1 GB大小的文件产生认证结构，文件块的大小设置为8 KB，测试在文件中随机插入不同数量的文件块后，平均插入每一个文件块时，调整认证结构所消耗的时间，测试结果如图6所示。测试结果显示本文方案与文献[14]中的方案性能相近。

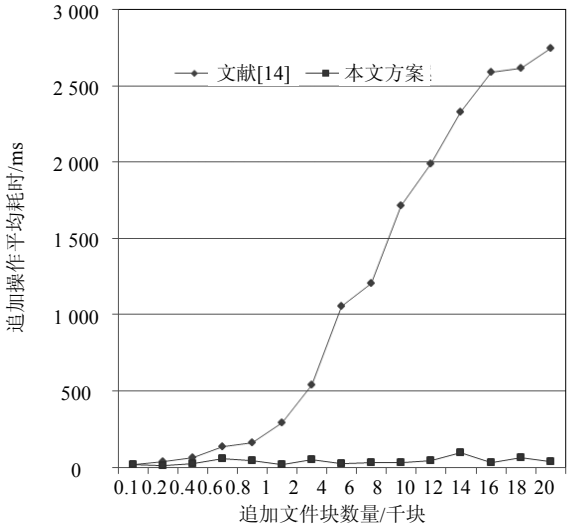

图5　连续在末尾追加数据块

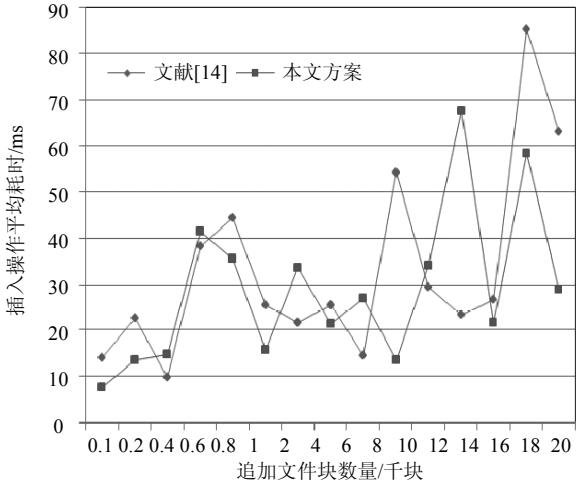

图6　在随机位置插入数据块

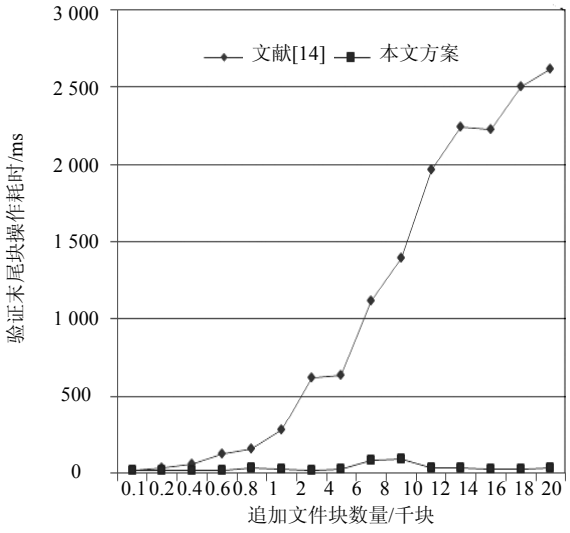

图7　对尾部数据块进行完整性验证

测试不平衡状态下验证的时间消耗，首先在云端存储大小为1 GB的文件块和对应的认证结构，文件块的大小设置为8 KB，通过在原始文件尾部追加不同数量的文件块以造成认证结构出现不同程度的不平衡后，对最后一个文件块进行完整性验证以观察在不同程度的不平衡下，验证所需要的时间，测试结果如图7所示。从测试结果可知，在不平衡状态下，本文所提出的方案在验证时的性能优于文献[14]中的方案。

4 结 束 语

云存储中的数据安全问题是不可回避的问题，而数据的完整性验证是保护数据安全的重要手段之一，因此面向云存储和量子存储的完整性验证方法和协议成为了近几年的研究热点之一[15-16]。本文基于改进的跳表提出一种适用于两方模型的云存储数据完整性验证协议，理论分析和模拟实验证明该协议具有较高的效率，但该方案没有考虑验证任务的代理问题，今后的工作将是对验证任务的代理问题进行研究。

参 考 文 献

[1] 耿纪昭. 云存储中数据完整性验证机制的研究与实现[D]. 成都: 电子科技大学, 2013.
GENG Ji-zhao. Research and implementation of data integrity verification mechanism cloud storag[D]. Chengdu: University of electronic science and technology of china, 2013.

[2] 苏兰. 面向云计算的数据完整性检验方法研究与实现[D]. 成都: 电子科技大学, 2013.
SU Lan. Research and implementation of data integrity verification method for cloud computing[D]. Chengdu: University of electronic science and technology of china, 2013.

[3] 邓晓鹏, 马自堂, 高敏霞. 一种基于双线性对的云数据完整性验证算法[J]. 计算机应用研究, 2013, 30(7): 2124-2127.
DENG Xiao-peng, MA Zi-tang, GAO Min-xia. Integrety verifying algorithm for cloud data based on bilinear pairings[J]. Application research of computers, 2013, 30(7): 2124-2127.

[4] WANG Cong, WANG Qian, REN Kui. Ensuring data storage security in Cloud Computing[C]//17th International Workshop on Charleston: Quality of Service (IWQoS). Charleston: IEEE, 2009.

[5] WANG Cong, WANG Qian, REN Kui. Privacy-preserving public auditing for data storage security in cloud computing [C]//2010 Proceedings IEEE INFOCOM. San Diego: IEEE, 2010.

[6] 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.
FU Ying-xun, LUO Sheng-mei, SHU Ji-wu. Survey of cloud storage system and key technologies[J]. Journal of Computer Research and Development, 2013, 50(1): 136-145.

[7] 李晖, 孙文海, 李凤华. 公共云存储服务数据安全及隐私保护技术综述[J]. 计算机研究与发展, 2014, 51(7): 1397-1409.
Li Hui, Shun Wen-hai, Li Feng-hua. Secure and privacy preserving data storage service in public cloud[J]. Journal of Computer Research and Development, 2014, 51(7): 1397-1409.

[8] BEHL A, BEHL K. An analysis of cloud computing security issues[C]//2012 World Congress on Trivandrum: Information and Communication Technologies (WICT). Trivandrum: IEEE, 2012.

[9] YANG Kan, JIA Xiao-hua. An efficient and secure dynamic auditing protocol for data storage in cloud computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2013, 24(9): 1717-1726.

[10] WANG Cong, WANG Qian, REN Kui. Enabling public auditability and data dynamics for storage security in cloud computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(5): 847-859.

[11] ATENIESE G, BURNS R, CURTMOLA R, et al. Provable data possession at untrusted stores[C]//ACM CCS'07. Alexandria: ACM, 2007.

[12] DESWARTE Y, QUISQUATER J J, SAIDANE A. Remote integrity checking[C]//Proc of Conference on Integrity and Internal Control in information Systems(IICIS'03). Boston: Springer, 2003.

[13] SEBE F, MARTNEZ-BALLESTE A, DESWARTE Y. Efficient remote data possession checking in critical information infrastructures[J]. IEEE Transactions on Knowledge and Data Engineering, 2008, 20(8): 1034-1038.

[14] 秦志光, 王士雨, 赵洋. 云存储服务的动态数据完整性审计方案[J]. 计算机研究与发展, 2015, 52(10): 2192-2199.
QIN Zhi-guang, WANG Shi-yu, ZHAO Yang. An auditing protocol for data storage in cloud computing with data dynamics[J]. Journal of Computer Research and Ddevelopment, 2015, 52(10): 2192-2199.

[15] LI Dong-fen, WANG Rui-jin, ZHANG Feng-li, et al. A noise immunity controlled quantum teleportation protocol [J]. Quantum Information Processing, 2016, 15(11): 4819-4837.

[16] LI Dong-fen, WANG Rui-jin, ZHANG Feng-Li, et al. Quantum information splitting of arbitrary two-qubit state by using four-qubit cluster state and Bell-state[J]. Quantum Information Processing, 2015, 14(3): 1103-1116.

编 辑 叶 芳