·计算机工程与应用·

一种代理远程数据完整性审计协议

赵 洋[1]，王士雨[1]，吴松洋[2]，熊 虎[1]

(1. 电子科技大学计算机科学与工程学院 成都 611731；2. 公安部第三研究所 上海 徐汇区 201204)

【摘要】随着云计算技术的快速发展，越来越多的用户将个人数据存储到远端云服务器上。为确保用户的数据被正确地存储在云服务器上，远程数据的完整性检查受到了学术界和工业界的广泛关注。考虑到个人有限的计算资源和通信带宽，用户可以将远端数据的完整性审计任务委托给专业的代理。由于目前已有的代理远程数据完整性审计方案只能支持静态数据的存储，所以该文基于Merkle Hash树和双线性对技术，提出了一种能够支持动态操作的代理远端数据完整性审计方案。该方案不仅满足远端数据完整性审计协议所需的安全要求，而且支持针对远端数据执行插入、删除及追加等动态操作。安全性证明和性能分析，表明该方案是安全和高效的。

关 键 词 审计协议； 云计算； 动态操作； 完整性检查； 代理

中图分类号 TP309 文献标志码 A doi:10.3969/j.issn.1001-0548.2016.01.013

A Proxy Auditing Protocol for Data Storage in Cloud Computing

ZHAO Yang[1], WANG Shi-yu[1], WU Song-yang[2], and XIONG Hu[1]

(1. School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 611731;
2. The Third Research Institute of Ministry of Public Security Xuhui Shanghai 201204)

Abstract With the rapid development of cloud computing technology, more users will store their data on a remote cloud server. To ensure that the user's data is correctly stored on the cloud server, remote data integrity checking has attracted widespread attention by academia and industry. By considering limited computing resources and communication bandwidth on the client side, users can delegate the auditing task of remote data integrity to a professional proxy. Currently, the proxy auditing scheme of remote data integrity can only support storing static data. But in this paper, we propose a proxy auditing scheme of remote data integrity for dynamics, which is based on Merkle Hash Tree and bilinear pairings technology. The proposed scheme can not only meet the security requirements of auditing protocol for the remote data, but also support the dynamic operations, such as inserting, deleting and appending. Security proof and performance analysis show that the proposed scheme is safe and effective.

Key words auditing protocol; cloud computing; dynamic operating; integrity checking; proxy

随着计算机技术的快速发展，云计算[1]已经成为计算机技术的一种应用趋势。由于个人有限的计算资源和通信带宽等的限制，个人无法承担越来越大的计算任务，所以需要寻求一种新的方式将用户从繁重的计算任务中解脱出来。云计算能够将许多分散的计算机连接起来，形成一个巨大的分布式系统，大大增强了计算机的计算能力。用户将个人的数据存储在云服务器，不仅减轻了本地存储带来的存储负担，也使得数据可以被用户随时随地访问。

尽管云计算技术具有诸多优点，但随之而来的问题是如何确保数据的安全性[2]。用户将个人数据存储在云服务器上，但是云服务器为了自身的经济利益，可能故意删除用户的一部分数据；即使云服务器能够诚实地存储用户数据，也不可避免由于软硬件故障而造成的数据损坏问题。当上述问题发生时，云服务器可能会隐藏这些错误，使用户相信数据仍然被正确地存储在云服务器。为确保用户存储在云服务器上数据的正确性，需要定期地检查云服务器上数据的完整性。

目前，有一些方案[3-18]已经能够检测远端用户数据的完整性。考虑到用户的计算资源和通信带宽有限的情况，或是有些用户可能无法连接网络，用户

收稿日期：2014－10－14；修回日期：2015－05－31
基金项目：国家自然科学基金(61003230, 61370026)；中央高校基本科研业务费(ZYGX2013J073)
作者简介：赵洋(1973－)，男，博士，副教授，主要从事信息安全、移动互联网应用方面的研究.

如果要检查数据的正确性，可以委托专业的代理服务器进行数据的完整性检查。虽然文献[14]提出了一种代理数据拥有协议，但是这种协议只支持静态数据的存储，不能实现用户的插入、删除和修改操作。

本文基于Merkle Hash树[13,19]和双线性对[13-14]技术，对文献[13-14]的方案进行扩展，提出了一种支持动态操作的代理远端数据完整性审计方案。获得用户授权的代理服务器，代替用户执行数据完整性检查的任务，这不仅大大减轻了用户的负担，而且还能增强用户对远端数据的隐私性保护。另外，本文的方案使用了一种有序的二叉树结构——Merkle Hash树，使得该方案能够很好地支持用户的动态操作。最后，本文对方案进行了安全性分析和性能分析，表明该方案是安全和高效的。

1　预备知识

在这一部分中，首先介绍系统模型；然后描述系统的安全模型，即介绍系统模型中存在的安全性问题。

1.1　系统模型

系统模型如图1所示，包含3个实体——用户、云服务器和代理服务器。

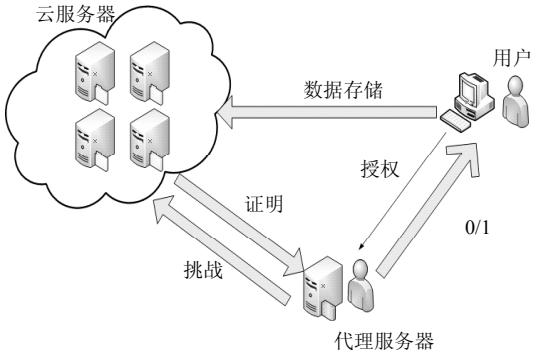


图1　系统模型

在系统模型中，用户将个人数据存储在云服务器上，并删除本地数据副本，这样可大大减轻用户的存储负担。用户将远端数据完整性检查的任务委托给代理服务器，当代理服务器拥有用户的授权并通过授权的验证时，就可以代替用户执行远端数据的完整性检查。然后，代理服务器向云服务器发起数据完整性检查的请求，云服务器生成一个证明并将其返回给代理服务器。最后，代理服务器对证明进行验证，并可将验证的结果告知用户。如果云服务器能够通过数据完整性的验证，就说明用户的数据被正确地存储在云服务器上；否则，就说明用户存储在云服务器上的数据被损坏。

1.2　安全模型

在云存储系统中，不仅要确保用户存储在云服务器上数据的正确性，还要支持用户更新数据的需求。由于云服务器是半可信的，且动态操作存在不安全的问题，所以系统可能会出现如下形式的攻击：

1) 重放攻击和伪造攻击。由于标签的形式为 $H(k,i)$，所以更新之后第 i 个数据块的标签仍然为 $H(k,i)$。如果云服务器没有将“数据块–签名”更新为 $(m_i^*,\sigma_{m_i^*})$，当代理服务器进行远端数据的完整性检查时，云服务器可以使用已经过期的 (m_i,σ_{m_i}) 和 $H(k,i)$ 计算证明以欺骗代理服务器。为避免插入数据块时需重新计算签名的问题，可以使用数据块 m_i 替换索引值 i。由于用户没有在本地存储个人数据，所以标签只能由云服务器生成，那么云服务器就可以使用任意组合的聚合“数据块–签名”以通过代理服务器的数据完整性检查。当云服务器使用这种方式应对代理服务器的数据完整性检查时，只需存储一个“数据块–签名”即可通过代理服务器的检查。为防止云服务器伪造标签，需要增加对标签的验证过程，以确保标签和数据块是相对应的。

2) 删除数据块的攻击。云服务器可能预先计算并存储聚合的“数据块–签名”，并将用户存储在云服务器上的“数据块–签名”删除。当代理服务器执行远端数据的完整性检查时，云服务器就可以将预先计算的“数据块–签名”发送给代理服务器，以欺骗代理服务器。

2　协议的构造

在这一部分中，首先介绍协议的基本方案，然后介绍协议的动态操作的算法。

2.1　基本方案

假设考虑的数据对象是集合 M，包含 n 个数据块，即 $M=\{m_1,m_2,\cdots,m_n\}$，其中 $m_i\in Z_p^*$ 并且 p 是一个大素数。基本方案由6个算法组成，下面介绍具体构造。

1) $\mathrm{SetUp}(1^\lambda)\to k$。用户选择一个随机数 $x\in Z_p^*$ 作为私钥，计算 $X=g^x$ 作为公钥，从而得到密钥对 (x,X)。同理，云服务器得到密钥对 (y,Y)，代理服务器得到密钥对 (z,Z)。用户、云服务器和代理服务器分别计算 $\hat{e}(Y,Z)^x$、$\hat{e}(X,Z)^y$ 和 $\hat{e}(X,Y)^z$，从而得到系统中三方共享的一个密钥 k。根据双线性对[13-14]的知识可知，显然有等式 $k=\hat{e}(g,g)^{xyz}=\hat{e}(Y,Z)^x=\hat{e}(X,Z)^y=\hat{e}(X,Y)^z$ 成立。用户还需要生成

一个授权 a，这个授权描述了代理服务器具有的执行远端数据完整性检查的权限。为确保上述授权的完整性，用户需要对其进行签名，记为 σ_a。远端数据完整性检查包含对授权的验证，当验证通过后才允许代理服务器执行后续的步骤。

2) $\text{TagGen}(M)\to(\Phi,\sigma_R)$。给定数据集合 $M=\{m_1,m_2,\cdots,m_n\}$，用户选择一个随机数 $u\in G_1^*$，用于对数据块进行签名，所以用户的公钥为 (X,u)，私钥为 x。用户计算数据块的标签和签名，具体步骤如下：

① 计算标签 $T_{m_i}=H(k,m_i)$，用于生成签名和构造MHT。② 计算数据块的签名 $\sigma_{m_i}=(T_{m_i}u^{m_i})^x$，并记签名的集合为 $\Phi=\{\sigma_{m_i}\}_{1\leqslant i\leqslant n}$。③ 对MHT的根 R 和 a 分别进行签名得到 $\sigma_R=(H(k,R))^x$ 和 $\sigma_a=(H(k,a))^x$。另外，数据集合 M 的标签用 $T_M=\text{name}\|n\|R\|\text{Sig}_x(\text{name}\|n\|R)$ 表示，以便记录数据对象的相关信息。

用户将 $\{M,T_M,\Phi,\sigma_R,a\}$ 发送给云服务器，并且删除本地个人数据的副本。云服务器收到用户发送的数据后，将其存储在云服务器上。另外，用户需要将 $\{a,\sigma_a\}$ 发送给代理服务器，以便用于执行远端数据的完整性检查。为确信云服务器诚实地应对代理服务器的数据完整性检查，用户和代理服务器需要存储 T_M，并且每一次更新操作之后都需要重新计算并存储 T_M。

3) $\text{SignVerfiy}(a,\sigma_a)\to\{1,0\}$。代理服务器收到用户发送的 (a,σ_a)，执行验证算法验证等式 $\hat{e}(\sigma_a,g)\overset{?}{=}\hat{e}(H(k,a),X)$ 是否成立。如果等式成立，代理服务器就接受 (a,σ_a)；否则，代理服务器将拒绝 (a,σ_a)，并要求用户发送新的 (a,σ_a)。

4) $\text{CheckTag}(\sigma_R)\to\{1,0\}$。当云服务器收到用户的存储请求之后，执行算法以验证标签的正确性。为了提高验证的效率，只需要验证MHT的正确性，验证过程如下：

① 存储用户的数据，并诚实地构造MHT以得到根 $\hat{R}$。② 判断等式 $\hat{e}(\sigma_R,g)\overset{?}{=}\hat{e}(H(k,\hat{R}),X)$ 是否成立。如果等式成立，云服务器就接受用户的存储请求；否则将拒绝用户的存储请求，并要求用户重新发送数据。

5) $\text{GenProof}(M,C,\Phi)\to P$。假设代理服务器选择的挑战集为 $C=\{(i,v_i)\}$，即代理服务器为执行远端数据完整性检查而发送的随机系数，其中 $i\in\{s_1,s_2,\cdots,s_c\}\subseteq\{1,2,\cdots,n\}$ 且假设 $s_1\leqslant s_2\leqslant\cdots\leqslant s_c$。在发送挑战之前，代理服务器需要将 T_M 发送给云服务器，表明将对数据集合 M 执行完整性检查。代理服务器将 C 和 (a,σ_a) 发送给云服务器，云服务器首先需要验证代理服务器授权的合法性。验证通过后才允许代理服务器执行数据的完整性检查，否则将拒绝代理服务器的请求。算法的执行过程如下：

① 判断等式 $\hat{e}(\sigma_a,g)\overset{?}{=}\hat{e}(H(k,a),X)$ 是否成立，如果成立则执行步骤2)，否则结束。② 计算 $m=\sum_{i=s_1}^{s_c}v_im_i$ 和 $\sigma=\prod_{i=s_1}^{s_c}\sigma_{m_i}^{v_i}$。③ 生成辅助信息集合 $\{\Delta\}_{s_1\leqslant i\leqslant s_c}$，该信息用于构造MHT。④ 输出证明 $P=\{m,\sigma,\{T_{m_i},\Delta_i\}_{s_1\leqslant i\leqslant s_c},\sigma_R\}$，并将 P 返回给代理服务器。

6) $\text{VerifyProof}(C,P)\to\{1,0\}$。当代理服务器收到证明 P 时，就执行下面的验证过程：

① 使用 $\{T_{m_i},\Delta_i\}_{s_1\leqslant i\leqslant s_c}$ 构造MHT，生成根 R。② 判断等式 $\hat{e}(\sigma_R,g)\overset{?}{=}\hat{e}(H(k,R),X)$ 是否成立；如果等式成立则执行步骤③，否则输出“0”并结束。③ 判断等式 $\hat{e}(\sigma,g)\overset{?}{=}\hat{e}(\prod_{i=s_1}^{s_c}T_{m_i}^{v_i}\cdot u^m,X)$ 是否成立；如果成立则输出“1”，否则输出“0”。

2.2 动态操作

对于远端数据，动态操作包括3种类型——插入(insert)、修改(modify)和删除(delete)。假设数据集合 M、签名 Φ 和签名 σ_R 已经生成，并存储到云服务器上，同时云服务器已经成功构造MHT。动态操作协议包含3个角色，将分4个步骤进行，下面详细介绍其过程。

1) 生成更新操作的消息。该步骤生成更新操作的消息，即执行算法 PrepareUpdate()，下面将分3种情况进行描述。

插入操作：假设用户想在第 i 个位置上插入一个新的数据块 m^*，那么需要将更新请求的消息 $M_u=(\text{Insert},i,m^*,\sigma_{m^*})$ 发送给云服务器，其中数据块的标签和签名分别为 $T_{m^*}=H(k,m^*)$ 和 $\sigma_{m^*}=(T_{m^*}u^{m^*})^x$，$i$ 表示数据块插入的位置。

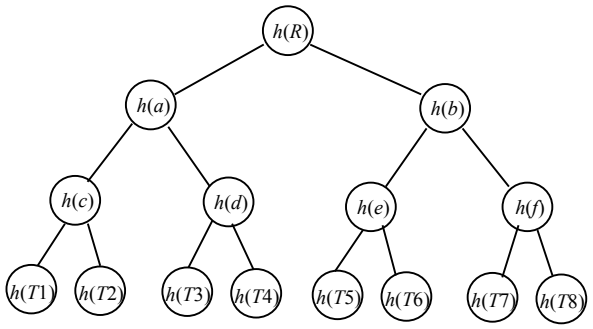
修改操作：修改操作的过程和插入操作类似，只需将更新消息修改成 $M_u=(\text{Modify},i,m^*,\sigma_{m^*})$，即将第 i 个位置的数据块修改为 m^*。

删除操作：对于删除操作而言，只需将更新消息修改成 $M_u=(\text{Delete},i)$，即将第 i 个位置的数据块删除。

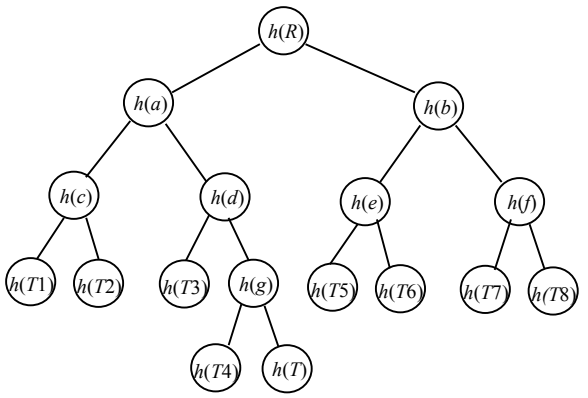
最后，当更新请求的消息 M_u 生成之后，用户就将其发送给云服务器，以便执行更新操作。

2) 执行更新操作。当云服务器收到更新请求的消息后，将会执行算法ExecUpdate()，下面详细介绍其过程。

① 云服务器计算标签$T_{m^*}=H(k,m^*)$。② 插入数据块时，云服务器计算$h(T_{m^*})$作为MHT的叶子结点，并将其插入到相应的位置，如图2所示；同时云服务器需要存储数据块m^*和签名σ_{m^*}。修改数据块时，云服务器将MHT的第i个叶子结点替换为$h(T_{m^*})$，同时将数据块m_i和其签名σ_{m_i}分别替换为m^*和σ_{m^*}。删除数据块时，云服务器将MHT的第i个叶子结点删除，同时删除数据块m_i和其签名σ_{m_i}。③ 云服务器更新MHT，得到新的根R'，最终生成证明$P_u=(\Delta_i,T_{m_i},\sigma_R,R')$，并将其返回给用户。



a. 执行插入操作之前的MHT



b. 执行插入操作之后的MHT

图2　MHT的结构

3) 验证更新操作。当用户收到云服务器发送的证明P_u后，将执行算法VerifyUpdate()，验证云服务器是否正确地执行更新操作，过程如下：

① 用户使用$\{\Delta_i,T_{m_i}\}$构造MHT的根R，并判断等式$\hat{e}(\sigma_R,g)\stackrel{?}{=}\hat{e}(H(k,R),X)$是否成立；如果等式不成立，则输出“0”并结束，否则执行步骤②。② 用户根据3种不同的动态操作，分别更新MHT并生成新的根R''，然后比较R''和R'是否相等；如果不相等，则输出“0”并结束，否则执行步骤③。③ 用户对R'进行签名得到$\sigma_{R'}=(H(k,R'))^x$，并将其发送给云服务器，最后系统输出“1”。

4) 执行数据的完整性检查。当上述3个过程都正确执行以后，代理服务器就可以向云服务器发起挑战，执行远端数据的完整性检查。如果云服务器通过完整性检查，用户就可以删除本地数据；否则，需要执行多次远端数据的完整性检查，以确信云服务器上的个人数据没有被损坏。

3　安全分析

这一部分将分析方案的安全性，解决方案中存在的安全性问题。

3.1　伪造攻击和重放攻击

本文方案将会移除标签中的数据块索引值i，即将$H(k,i)$替换为$H(k,m_i)$，这样在对数据进行更新时，$H(k,m_i)$会发生改变，MHT也会随之发生改变。$H(k,i)$可以被系统中任意一个合法的实体生成，而本文方案中的$H(k,m_i)$却只能由云服务器生成，所以需要验证标签是否被恶意的云服务器伪造。

在验证数据块的正确性之前，需要对标签的正确性进行验证。如果标签通过验证，则说明云服务器没有伪造标签，可继续进一步的数据完整性验证；否则，说明云服务器伪造了标签，将结束验证的过程。因为用户将数据存储在云服务器或是执行更新操作后，都对MHT的根进行重新签名，所以云服务器无法对MHT的根进行伪造。根据抗碰撞Hash函数的单向性可知，云服务器伪造的标签不能以一个不可忽略的概率通过代理服务器的验证。同理，如果云服务器使用过期版本的数据块替换最新版本的数据块，将不能通过代理服务器对标签的验证过程。

3.2　数据块的删除

为防止云服务器使用任意组合的聚合“数据块–签名”欺骗代理服务器，需要对数据块对应的标签进行验证。假设云服务器已经通过上述的验证过程，当代理服务器随机地选择s_c个数据块以执行完整性检查时，那么聚合的数据块数量就存在$C_n^1+C_n^2+\cdots+C_n^n=2^n-1$种组合。同时代理服务器向云服务器发送了$s_c$个随机系数，所以云服务器需要存储的数据块数量将远远大于n。通过上述分析可知，如果云服务器想要通过预先计算聚合数据块的方式通过代理服务器的验证是不现实的。

4　性能分析

在这一部分中，首先对执行远端数据完整性检

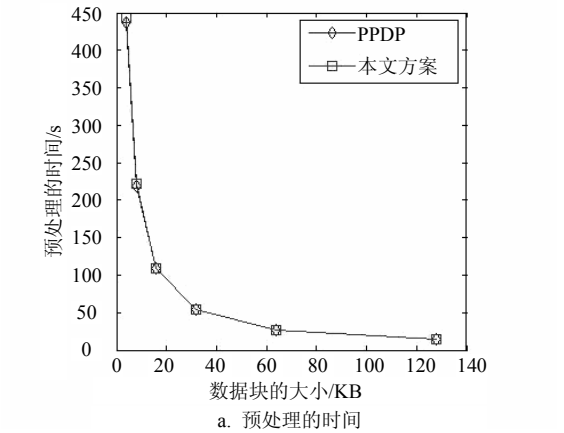
查的5种方案进行性能比较，然后对本文和文献[14]的方案进行模拟分析。

假设 n 表示用户总的数据块数量，b 表示损坏的数据块数量，c 表示代理服务器向云服务器发起远端数据完整性检查时选择的数据块数量，那么可以计算出5种方案的检查概率都相等，如表1所示。由于要支持动态操作，所以需要引入能够支持动态操作的数据结构，本文使用的是MHT。如果修改某个数据块，那么只会影响验证路径上结点的hash值，MHT的这种优点将大大降低更新操作的复杂度。

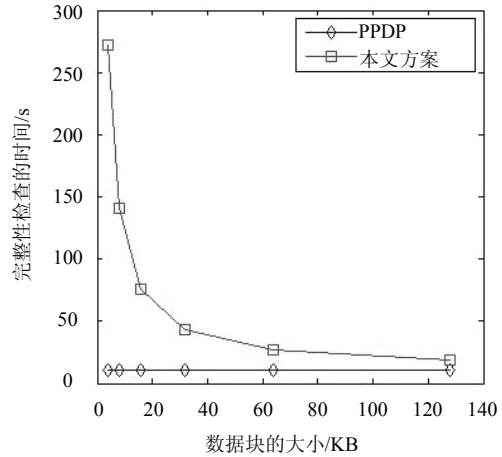
表1 执行远端数据完整性检查的方案性能比较

	PDP[3]	DPDP[5]	文献[13]	PPDP[14]	本文的方案
是否支持动态操作	×	√	√	×	√
是否支持代理	×	×	×	√	√
检测概率	$1-\left(1-\frac{b}{n}\right)^c$	$1-\left(1-\frac{b}{n}\right)^c$	$1-\left(1-\frac{b}{n}\right)^c$	$1-\left(1-\frac{b}{n}\right)^c$	$1-\left(1-\frac{b}{n}\right)^c$
用户的存储复杂度	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
云服务器端的存储复杂度	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
云服务器端的计算复杂度	$O(1)$	$O(\log n)$	$O(\log n)$	$O(1)$	$O(\log n)$
验证的计算复杂度	$O(1)$	$O(\log n)$	$O(\log n)$	$O(1)$	$O(\log n)$

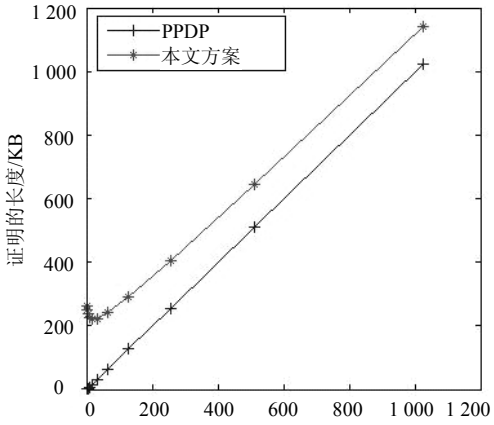
下面将本文的方案和文献[14]的方案进行模拟分析比较，其中程序编码主要基于版本0.5.11的PBC库，并且使用BLS[20-21]作为签名方案。模拟环境的配置如下：一台装有Ubuntu 13.10操作系统的计算机，主频为1.9 GHz的Intel Core 2 Duo CPU和2 GB RAM，文件系统为ext4。需要说明的是，两种方案需要在相同的环境下进行多次模拟计算，所以本文进行了10次测试。



a. 预处理的时间



b. 执行完整性检查的时间



c. 证明的长度

图3 性能分析

为了实现远端数据的完整性检查，需要对数据进行预处理，其中最主要的步骤是对数据块进行签名，预处理的时间如图3a所示。从图中可以看出，当数据块长度取64 KB或是128 KB时，预处理的时间将变得很小并且减小不再明显。

当代理服务器向云服务器发起远端数据完整性检查后，云服务器需要生成证明并将其返回给代理服务器，然后代理服务器判断证明的正确性。由于本文的方案需要支持动态操作，所以执行完整性检查的时间将会增大，如图3b所示。两种方案会出现这种差别，主要原因是本文的方案为抵御重放攻击和伪造攻击，需要对MHT的构造进行验证。

云服务器生成证明后，需要将证明返回给代理服务器以便验证远端数据的正确性，证明的长度曲线如图3c所示。考虑到预处理的时间和执行完整性检查的时间，数据块的长度不能选取的太小。从图中可看出，数据块长度可选取为64 KB或是128 KB，

这样在证明的长度基本不变的情况下，执行完整性检查的时间将会变得更加合理。

通过上述分析，表明用户将数据存储在云服务器上，并将远端数据的完整性检查委托给代理服务器，可以大大减轻用户的负担。本文中为能够支持用户的动态操作而使用了MHT，虽然增加了计算和通信的开销，但是如果适当地选取数据块的长度，可使方案变得非常高效。

5 结 束 语

为确保用户的数据被正确地存储在云服务器上，用户需要做远端数据的完整性检查。但是当用户没有能力执行完整性检查任务时，就可以将该任务委托给代理服务器。本文提出的支持动态操作的代理远程数据审计协议，代理服务器能够代替用户执行远端数据的完整性检查。在本文的方案中，为支持动态操作而引入MHT，不但能很好支持用户的动态操作，而且可以大大减轻云服务器检验数据标签的签名的负担。最后，通过安全性分析和性能分析，说明本文的方案是安全和高效的。但是，本文没有给出安全性证明，所以未来需要进一步证明方案的安全性。

参 考 文 献

[1] MELL P, GRANCE T. The NIST definition of cloud computing[J]. National Institute of Standards and Technology, 2009, 53(6): 50-57.

[2] 冯登国, 张敏, 张妍, 等. 云计算安全研究[J]. 软件学报, 2011, 22(1): 71-83.
FENG Deng-guo, ZHANG Min, ZHANG Yan, et al. Study on cloud computing security[J]. Journal of Software, 2011, 22(1): 71-83.

[3] GIUSEPPE A, RANDAL B, REZA C. Provable data possession at untrusted stores[C]//Proceedings of the 14th ACM Conference on Computer and Communications Security. [S.l.]: ACM, 2007: 598-609.

[4] ATENIESE G, DI P 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 Networks. [S.l.]: ACM, 2008.

[5] ERWAY C, KÜPÇÜ A, PAPAMANTHOU C, et al. Dynamic provable data possession[C]//Proceedings of the 16th ACM Conference on Computer and Communications Security. [S.l.]: ACM, 2009: 213-222.

[6] WANG C, WANG Q, REN K, et al. Privacy-preserving public auditing for data storage security in cloud computing[C]//Proceedings of the 28th IEEE INFOCOM. [S.l.]: IEEE, 2010: 1-9.

[7] WANG Q, WANG C, LI J, et al. Enabling public verifiability and data dynamics for storage security in cloud computing[M]//Computer Security. Berlin Heidelberg: Springer, 2009: 355-370.

[8] LIU C, CHEN J, YANG L, et al. Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates[J]. IEEE Transactions on Parallel and Distributed Systems, 2013, 25(9): 2234-2244.

[9] ETEMAD M, KÜPÇÜ A. Transparent, distributed, and replicated dynamic provable data possession[M]//Applied Cryptography and Network Security. Berlin Heidelberg: Springer, 2013: 1-18.

[10] 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. [S.l.]: ACM, 2007: 584-597.

[11] SHACHAM H, WATERS B. Compact proofs of retrievability[C]//Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security. Berlin Heidelberg: Springer, 2008: 90-107.

[12] YANG K, JIA X. An efficient and secure dynamic auditing protocol for data storage in cloud computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2012, 24(9): 1717-1726.

[13] 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.

[14] WANG H. Proxy provable data possession in public clouds[J]. IEEE Transactions on Services Computing, 2012, 6(4): 551-559

[15] WANG H. Identity-based distributed provable data possession in multi-cloud storage[J]. IEEE Transactions on Services Computing, 2014, 8(2): 328-340.

[16] REN Y, XU J, WANG J, et al. Designated-verifier provable data possession in public cloud storage[J]. International Journal of Security and Its Applications, 2013, 7(6): 11-20.

[17] LIU C, YANG C, ZHANG X, et al. External integrity verification for outsourced big data in cloud and IoT: a big picture[J]. Future Generation Computer Systems, 2015, 49: 58-67.

[18] ATENIESE G, BURNS R, CURTMOLA R, et al. Remote data checking using provable data possession[J]. ACM Transactions on Information and System Security (TISSEC), 2011, 14(1): 1165-1182.

[19] GOLLE P, JARECKI S, MIRONOV I. Cryptographic primitives enforcing communication and storage complexity[M]//Financial Cryptography. Berlin Heidelberg: Springer, 2003: 120-135.

[20] BONEH D, LYNN B, SHACHAM H. Short signatures from the Weil pairing[J]. Journal of Cryptology, 2004, 17(4): 297-319.

[21] BONEH D, GENTRY C, LYNN B, et al. Aggregate and verifiably encrypted signatures from bilinear maps[M]// Advances in Cryptology—EUROCRYPT 2003. Berlin Heidelberg: Springer, 2003: 416-432.

编 辑 漆 蓉