代理动态操作的云数据拥有性证明方案

赵 洋，任化强，熊 虎，陈 阳

(1. 电子科技大学信息与软件工程学院 成都 610054)

【摘要】传统支持动态操作的云存储系统中，对于云中动态文件的动态操作所产生的计算和通信开销是由用户完全承担的，这给用户带来了相应的计算和通信压力。为了解决该问题，构造了云环境下第三方代理动态操作的数据拥有性证明方案，方案中引入了一个功能强大的第三方审计者，用户不仅能委托其承担审计工作而且能够使其代理完成动态操作的任务，在任务处理的过程，系统能够保证用户的数据在第三方审计者中的隐私性。从安全和性能分析可以看出该方案能高效安全地完成审计任务。

关 键 词 代理动态操作； 云存储安全； Merkle哈希树； 第三方代理审计

中图分类号 TP309 **文献标志码** A doi:10.3969/j.issn.1001-0548.2016.05.015

Provable Cloud Data Possession Scheme via Third-Party Proxy Dynamic Operations

ZHAO Yang, REN Hua-qiang, XIONG Hu, and CHEN Yang

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

Abstract In the traditional cloud storage system which supports dynamic operations, the computing and communication cost derived from the dynamic operations to the file is undertaken by the user completely. It brings user the corresponding pressure on computation and communication. In order to solve this problem, we design a provable data possession scheme via third-party proxy dynamic operations in clouds. In this scheme we bring in a powerful third-party auditor, the users can delegate it to perform the audit and fulfill the dynamic operations. In this processing, the system can ensure the user's data privacy in the third party auditor. Finally, the security and performance analysis show that our scheme can complete the audit task efficiently and securely.

Key words agent dynamic operation; cloud storage security; Merkle Hash tree (MHT); the third-party audit

云存储系统能够给用户提供可靠的、自定义的、资源利用率最大化的服务。用户可以将自己的本地数据存储在云端，在不同地点通过网络访问云端数据。这样的数据存储方式有很多优点：节约了用户的本地存储空间与维护复杂度，用户可以更加灵活的访问自己的数据并获得更加强大的计算能力而不必担心硬件和软件等条件的限制。

云计算在给用户带来便利性的同时，由于用户并不真正存储数据，从而难以保证数据的完整性和可用性。首先，云存储服务器会不可避免地受到外界和内部的攻击，而使数据存在丢失与篡改等安全问题。其次，云服务提供商可能会为了获取更大利益，而刻意删除、更改用户数据。这些问题都会使数据的完整性和可用性难以保证，同时在传统的支持动态操作的云存储系统中，对于云中动态文件的动态操作所产生的计算和通信开销是由用户完全承担的，这给用户带来了相应的计算和通信压力。为了解决上述问题，本文构造了一种代理动态操作的云数据拥有性证明方案。该方案为了减轻用户的计算和通信负担，引入了一个功能强大的第三方审计者，用户不仅能够委托其承担审计工作还能使其代理完成动态操作的任务。在动态操作过程中用户只需承担计算插入与修改操作时的数据块签名和很小的通信消耗，动态操作的验证等工作则由第三方审计者代理完成。本文的方案是基于双线性映射和Merkle哈希树实现的，第三方审计者和用户无需下载云中数据即可完成验证。同时系统也保证了用户的数据在第三方审计者中的隐私性，使第三方审计

收稿日期：2015 – 04 – 07；修回日期：2015 – 08 – 24
基金项目：国家863项目(2015AA016007)；国家自然科学基金(61472064)
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.

者无法获取用户的数据。

1 相关工作

随着云计算的快速发展和基于对云端数据安全性的担忧，国内外研究者设计了多种不同的解决方案用于实现云数据完整性审计[1-13]。这些方案中有使用哈希和签名技术对全部数据审计的方案和使用概率统计检测的方案，前者需要用户在本地拥有数据备份，这些方案的效率比较低。使用同态标签技术的方案典型的有文献[1]的PDP(provable data possession)方案，其通过使用挑战–响应协议和概率检测，实现对半可信的云服务器中的数据进行完整性验证，概率性检测使其效率大大提高，在不获取全部数据时即可高概率地进行完整性审计。PDP方案适合于静态文件的验证，当用户的数据进行插入、删除和修改等操作后，并不能提供正确的验证结果。为了满足数据动态审计的要求，文献[3]扩展了PDP方案，提出了一个动态版的数据拥有性证明方案(dynamic provable data possession, DPDP)，但是该方案仍然不能支持完整的动态操作，难以抵御欺骗攻击。文献[12]基于等级的认证跳表和RSA树的结构，首次构造了一个完整的DPDP方案，但是该方案可能导致数据块泄露。文献[5]提出了一个带隐私保护的审计方案，支持外部审计者审计用户在云服务器上的数据，但是方案的分块策略带来了大量的额外存储开销。文献[7]提出了一个综合性的方案，使用Merkle哈希树(MHT)来支持完整的数据操作并支持公共验证，但在动态操作的过程中需要用户全程参与，给用户带来了计算负担，同时也存在庞大的额外存储开销的问题。文献[6]解决了数据分块带来大量额外存储开销的问题，但其用来代理用户执行动态操作的授权应用端(AA)拥有用户的私钥，这样会导致方案存在单点失效与私钥泄露的风险。

基于以上的相关工作，本文提出一个综合性的方案。该方案通过对私钥分配的设计让第三方审计者代理用户进行动态操作，避免了文献[6]中AA带来的安全隐患，减少了用户的计算和通信消耗，使用户在移动端也能高效方便地对文件进行动态操作。

2 系统安全模型与预备知识

2.1 安全定义

定义 1 在审计的过程中如果云服务提供者发送正确的证据，第三方审计者审计时会通过验证。

定义 2 云服务提供者不能够伪造出数据块的签名通过第三方审计者的验证。

定义 3 云服务提供者不能够伪造出动态操作后的证据通过第三方审计者对动态操作的验证。

定义 4 第三方审计者根据已有数据无法获取用户的私密数据。

2.2 系统模型和安全模型

代理动态操作的云数据拥有性证明方案有3个实体，分别是用户端(cloud user, CU)、云服务提供商(cloud server provider, CSP)和第三方审计平台(third party auditor, TPA)，系统整体架构如图1所示。

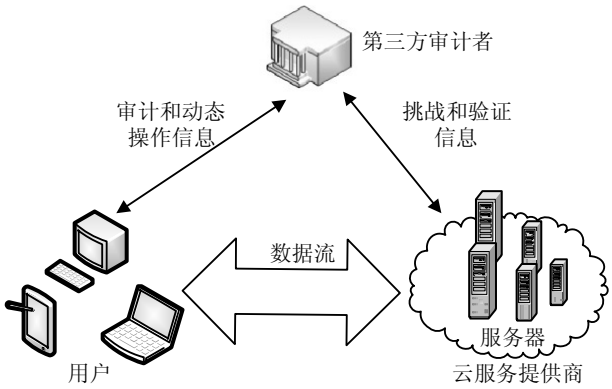

图1 支持第三方审计和动态操作的云存储系统架构

首先CU将拥有的本地数据传送给CSP，当CSP收到用户元数据和标签等信息后进行保存。在完整性审计时，CU将完整性验证任务交于第三方验证者代理执行，这样CU在没有网络情况下也可以对云端的数据进行检测。CU与TPA签订好代理协议后将代理信息发送给CSP，CSP确认TPA代理审计的合法性后，TPA将会代理用户周期的执行完整性审计和动态操作审计的任务。在进行完整性审计时TPA将对CSP进行挑战，CSP根据挑战信息生成相应的证据传送给TPA进行验证。当CU对云端数据进行增删改操作时，CU首先把操作信息发送给CSP，CSP收到更新信息后对用户云端的数据进行相应的动态操作，同时CSP将产生的证据发送给TPA进行验证完成动态操作。

2.3 双线性映射

双线映射：一个双线性映射是这样一个映射 e：$G\times G\to G_T$，其中 G、G_T 是给定素数 p 阶的循环乘法群，g 是 G 的生成元。对任意 $h\in G$，函数 e 拥有以下性质[14]：

1) 可计算性：存在一个有效的算法计算 e；
2) 双线性：$e(h^a,h^b)=e(h,h)^{ab}$，$a,b\in\mathbb{Z}_P$；
3) 可交换性: $e(h^a,h^b)=e(h^b,h^a)$，$a,b\in\mathbb{Z}_P$；
4) 非退化性：$e(g,g)\neq 1$。

2.4 Merkle哈希树

Merkle哈希树(MHT)是一种树型的数据结构，该结构的每一个节点都是哈希值，其中叶子节点是原始数据块的哈希值，每一个内部节点是其孩子节点数据联接后计算得到的哈希值。Merkle哈希树可以高效安全地证明数据集是否被损坏与修改[15]，同时也可以支持数据动态操作验证。本文方案的Merkle哈希树结构如图2所示，叶子节点由二级分区的文件块的哈希构成。在应用中还要用哈希树的辅助信息的概念，如图2所示当知道叶子节点 L 时需要辅助信息 $\Omega_2=\{L_1,B\}$ 获取哈希树的根 R，Ω_i 即相应节点的辅助信息。

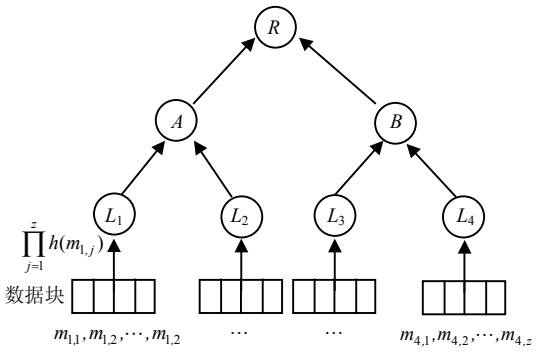

图2 Merkle哈希树结构示例

3 方案提出

3.1 符号引入

$F-(m_{1,1},m_{1,2},\cdots,m_{1,z};m_{2,1},m_{2,2},\cdots,m_{2,z};\cdots;m_{n,1},m_{n,2},\cdots,m_{n,z})$：$F$ 是通过文件分区技术划分的二级分区的文件数据块组成的集合。

$h(\cdot)-\{0,1\}^*\to G$：密码学中的哈希函数，G 是一个循环乘法群。

$H(\cdot)-\{0,1\}^*\to G$：密码学中的哈希函数，G 是一个循环乘法群。

$f_{\text{key}}(\cdot)-\{0,1\}^*\times\text{key}\to\mathbb{Z}_p$：伪随机函数(PRF)。

$\pi_{\text{key}}(\cdot)-\{0,1\}^{\log_2(n)}\times\text{key}\to\{0,1\}^{\log_2(n)}$：伪随机排列(PRP)。

3.2 方案的构造

方案中用户将完整性审计和动态操作工作委托TPA完成。为了支持这些性能，用户生成了两个私钥，一个由用户拥有用于对文件块进行签名，完成完整性审计功能；另一个由用户和TPA共同拥有用于对MHT的根进行签名，完成动态操作。

改进动态操作方案的云存储系统由3个阶段组成：初始化、审计和动态操作。

首先定义一个双线性映射，G、G_T 是给定素数 p 阶的循环乘法群，e：$G\times G\to G_T$，令 g 为 G 的生成元。

1) 初始化阶段。

用户运行 KeyGen(1^k) 函数产生系统中的私钥和公钥：首先选择两个随机数 $\text{csk}\leftarrow\mathbb{Z}_P$，$\text{tsk}\leftarrow\mathbb{Z}_P$ 作为私钥，然后选择一个随机数 $u\leftarrow G$，计算 $v=u^{\text{tsk}}$，$w=g^{\text{csk}}$ 作为公钥。所以系统的私钥集合 $\text{sk}=(\text{csk},\text{tsk})$，公钥集合 $\text{pk}=(g,v,u,w)$，其中 tsk 作为TPA的私钥由用户保存后发送给TPA，csk 用于对数据块的签名。拥有了私钥和公钥后用户运行 ParaGen(csk, tsk, F) 函数计算文件块和Merkle哈希树根的签名。文件块的签名为：

$$s_i=(\prod_{j=1}^{z}H(m_{i,j})\cdot g^{m_{i,j}})^{\text{csk}}\in G,S=\{s_i\}_{i\in(1,2,\cdots,n)}$$

式中，S 为 F 的 n 个文件块的签名集合；z 为每个文件块二级分区的数量。用户根据Merkle哈希树的构造方法生成树根 R，其中Merkle哈希树的叶子节点是由每个一级文件块的哈希值构成。Merkle哈希树根 R 的值为 $h(R)$，用户用TPA的私钥 tsk 对Merkle哈希树的根进行签名 $\text{sig}_{\text{tsk}}(h(R))=(h(R))^{\text{tsk}}\in G$，最后用户将 $\{F,S,\text{sig}_{\text{tsk}}(h(R))\}$ 发送给CSP并删除本地数据。

2) 审计阶段。

TPA：在审计阶段首先由TPA运行 GenChal() 函数，从[1,n]中选择一组随机序列：$I=\{c_1,c_2,\cdots,c_q\}$，$1<q<n$，$c_q=\pi_{\text{key}}(q)$，key 为每次审计时随机选取，然后再为每个 $c\in I$ 选择一个随机参数：$\rho_c\leftarrow\mathbb{Z}_P,\rho_c=f_{\text{key}}(c)$，$\text{chal}=\{(c,\rho_c)\}_{c_1<c<c_q}$，chal 为挑战信息，$c$ 为要挑战的文件块号。TPA将挑战信息 chal 发送给CSP进行挑战。

CSP：CSP收到TPA发送的挑战信息chal，运行函数 GenProof(F,S, chal) 生成证据。首先CSP执行 $f_{\text{key}}(\text{chal})$ 选择一个随机数 $\gamma\leftarrow\mathbb{Z}_P$，其中 key 为每次审计时随机选取，然后令 $k=w^{\gamma}=(g^{\text{csk}})^{\gamma}\in G$，$\pi'_j=\sum_{c\in I}\rho_c\cdot m_{c,j}$，$\pi'=\sum_{j=1}^{z}\pi'_j$，$\pi=\pi'+\gamma h(k)\in\mathbb{Z}_P$，$\delta=\prod_{c\in I}s_c^{\rho_c}\in G$。CSP生成证据：

$$\text{proof}=\{\pi,\delta,k,(\prod_{j=1}^{z}h(m_{c,j}),\Omega_c)_{c\in I},\text{sig}_{\text{tsk}}(h(R))\}$$

CSP将证据 proof 发送给TPA进行验证。

TPA：TPA接收到 proof 后执行 VerifyProof{pk, chal, proof} 进行验证。首先用

$\{\prod_{j=1}^{z} h(m_{c,j}), \Omega_c\}_{c\in I}$ 产生MHT的根 $h(R)$，然后验证 $e(\mathrm{sig}_{\mathrm{tsk}}(h(R)),u)?=e(h(R),v)$ 是否正确，如果错误输出 false，如果正确再验证证据：

$$e(\delta(k^{h(k)}),g)?=e(\prod_{c\in I}\prod_{j=1}^{z} H(m_{c,j})^{\rho_c}\cdot g^{\pi},w)$$

当挑战信息中的数据块和签名在CSP中存放正确时验证通过。

3) 动态操作阶段。

方案中的动态操作是基于Merkle哈希树实现的，动态操作后的验证工作由TPA代理实现。动态操作流程如图3所示。

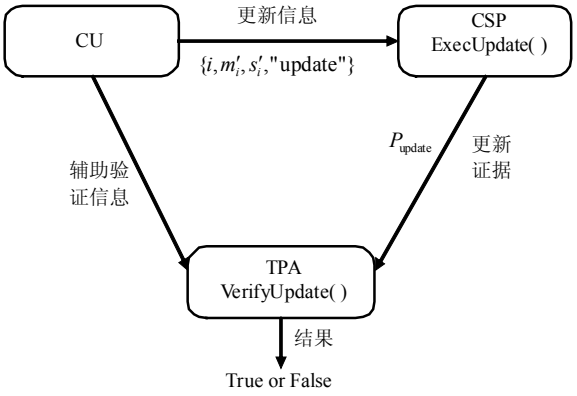

图3　动态操作执行流程

插入操作：当用户要在原有文件中插入一个新文件块时，用户将插入位置i、文件块m'_j、文件块签名s'_i和操作符“insert”发送给CSP进行动态操作。同时用户将要插入的信息i、“insert”和$\prod_{j=1}^{z} h(m'_{i,j})$发送给TPA进行验证时使用，然后CSP根据用户发送来的操作符判断用户要进行的操作。CSP执行 ExecUpdate$(i,m'_i,s'_i,\text{insert})$ 函数对文件进行更新。首先CSP将m'_i插入到原文件的第i文件块处生成新的F'，并保存文件块签名s'_i，生成新的签名集合S'，然后CSP根据Merkle哈希树的构造生成新的MHT根R'。CSP将更新后的证据 $P_{\mathrm{update}}=\{\Omega_i,\prod_{j=1}^{z} h(m_{i,j})$，$\mathrm{sig}_{\mathrm{tsk}}(h(R)),R'\}$ 发送给TPA，由TPA执行VerifyUpdate函数验证更新是否成功。验证过程首先计算出MHT旧的根R，然后验证 $e(\mathrm{sig}_{\mathrm{tsk}}(h(R)),u)?=e(h(R),v)$，如果不相等输出false，验证正确则继续进一步验证。进一步验证时，TPA根据Ω_i和$\prod_{j=1}^{z} h(m'_{i,j})$生成新的根$R''$，如果$R''$和$R'$不相等输出false，相等则TPA对新的根$R'$进行签名 $\mathrm{sig}_{\mathrm{tsk}}(h(R)')=(h(R)')^{\mathrm{tsk}}$ 发送给CSP完成动态操作，然后输出 true。

修改操作和删除操作的步骤同插入操作，在这里不再叙述。

4　安全性分析

本文方案分3个部分进行安全性分析，以证明方案的可靠性。

1) 审计过程的安全性。

定理 1　在审计的过程中如果云服务提供者发送正确的证据，第三方审计者审计时通过验证。

证明：如果CSP发送的证据正确则：

$$e(\delta(k^{h(k)}),g)?=e(\prod_{c\in I}\prod_{j=1}^{z} H(m_{c,j})^{\rho_c}\cdot g^{\pi},w)$$

其推导过程如下：

$$e(\delta(k^{h(k)}),g)=e(\prod_{c\in I} s_c^{\rho_c}\cdot((g^{\mathrm{csk}})^{\gamma})^{h(k)},g)=$$

$$e(\prod_{c\in I}((\prod_{j=1}^{z} H(m_{c,j})\cdot g^{m_{c,j}})^{\mathrm{csk}})^{\rho_c}\cdot((g^{csk})^{\gamma})^{h(k)},g)=$$

$$e(\prod_{c\in I}(\prod_{j=1}^{z} H(m_{c,j})\cdot g^{m_{c,j}})^{\rho_c}\cdot g^{\gamma\cdot h(k)},g)^{\mathrm{csk}}=$$

$$e(\prod_{c\in I}(\prod_{j=1}^{z} H(m_{c,j}))^{\rho_c}\cdot g^{\rho_c m_{c,j}+\gamma\cdot h(k)},w)=$$

$$e(\prod_{c\in I}\prod_{j=1}^{z} H(m_{c,j})^{\rho_c}\cdot g^{\pi},w)$$

定理 2　在审计的过程中云服务提供者不能够伪造出数据块的签名通过TPA的验证。

证明：CSP在已知用户的元数据、数据块签名和公共参数w和u的情况下计算伪造一个新的数据块签名，是一个Diffie-Hellman难题，Diffie-Hellman难题可以描述为：在知道g、g^a、g^b的情况下计算g^{ab}是困难的，因为很难求出a、b的值。

2) 动态操作的安全性。

定理 3　云服务提供者不能够伪造出动态操作后的证据通过第三方审计者对动态操作的验证。

证明：如果要伪造证据通过动态操作的验证，必须伪造旧的MHT的根R的签名：$\mathrm{sig}_{\mathrm{tsk}}(h(R))$和其MHT的辅助函数，这需要云服务提供者获得tsk，根据DHP(Diffie-Hellman problem)，CSP在知道公共参数v和g的情况下，计算出密钥tsk是困难的。

3) 用户的隐私安全。

定理 4　第三方审计者根据已有数据无法获取到用户的私密数据。

证明：第三方审计者拥有挑战证据和公共参数，

根据密码学中哈希函数 $h(\cdot)$ 的单向性，TPA在MHT根的签名 $\mathrm{sig}_{\mathrm{tsk}}(h(R))$，辅助信息 Ω 和叶子节点中提取不到文件块的信息，同时在 π 中由于随机数 γ 的存在，无法获得 π' 中的 $m_{c,j}$ 信息。

5 性能分析

本文介绍的代理动态操作云数据拥有性证明方案，相对于同样使用MHT的动态操作方案[7]，用户减少了更新证据 P_{update} 和MHT根签名 $\mathrm{sig}_{\mathrm{tsk}}(h(R)')$ 的通信消耗，同时用户也无需对 P_{update} 和MHT的旧根进行验证，减少了 $e(\mathrm{sig}_{\mathrm{tsk}}(h(R)),u)?=e(h(R),v)$ 和生成新MHT根 R'' 与验证的计算消耗。本文方案中这些动态操作步骤全部由TPA进行代理执行，在进行删除操作时用户没有计算消耗，只需发送操作序号和删除标识即可完成整个操作，减少了用户的计算和通信压力，使用户在移动端也能高效方便地对文件进行动态操作。下面介绍本文方案和相关方案其他性能方面的对比。

表1给出了相关方案和本文方案的比较，可以看出本文方案与同样拥有代理动态操作的文献[6]方案相比，本文方案解决了其单点失效的安全问题。同时由于本文方案采用MHT实现动态操作，在文件块较多的情况下比文献[6]中的索引哈希表性能更稳定高效，MHT动态操作复杂度为 $O(\log(n))$，索引哈希表插入和删除的复杂度为 $O(n)$。表2为动态操作中MHT与索引哈希表的性能对比，试验中，数据块个数 n 为1 000 000，操作项数为200。从表2中可以看出当 n 很大时本文的动态操作性能相对于文献[6]是高效的。

表1 相关方案与本文方案的比较

属性	相关方案							
	文献[2]	文献[3]	文献[4]	文献[5]	文献[6]	文献[7]	文献[12]	本文方案
私有审计	√	√	√	√	√	√	√	√
第三方审计	×	×	×	√	√	√	√	√
完整的动态操作	×	×	×	√	√	√	√	√
代理动态操作	×	×	×	×	√	×	×	√
安全问题		欺骗攻击[6]	数据块泄露[5]		单点失效		数据块泄露[6]	

表2 动态操作中MHT与索引哈希表的性能对比

方法	操作	消耗时间/ms
MHT	插入	24
	更新	18
	删除	21
索引哈希表[6]	插入	62
	更新	2
	删除	62

在表2的对比中可以看出MHT动态操作的总体性能要优于索引哈希表，索引哈希表是采用Java中ArrayList结构实现的，相当于数组的实现需要连续的存储空间，所以其更新的效率很高。由于MHT的树结构，当数据块很多时其性能也很稳定高效，总体优于索引哈希表。

表3 本文方案与文献[5]审计的计算时间对比

方案	审计的二级文件块数	审计消耗的时间/ms
本文方案	300	157
	200	141
文献[5]	300	158
	200	141

在文献[5]中文件分块相当于本文的二级文件块，为了减少签名带来的额外存储空间本文方案采用了文献[6]中的文件分块技术，只存储一级文件块的签名，从表3可以看出其性能仍然和文献[5]中方案一样高效。

图4为二级文件块的划分块数与计算签名，GenProof和VerifyProof运行时间的变化图，其中一级文件块数为20，块大小为160字节。

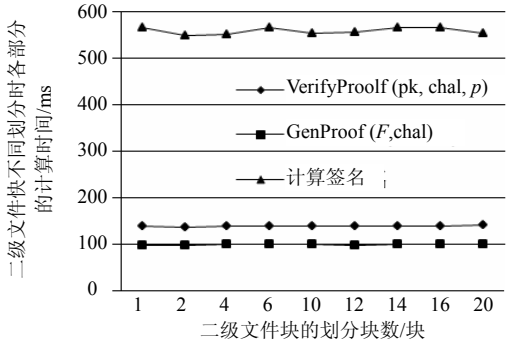

图4 计算签名、GenProof和VerifyProof运行时间的变化图

从图4中也可以看出，随着二级文件块的块数增加， GenProof 和 VerifyProof 部分所用的时间并没有明显的变化，计算签名所用的时间变化稍微大一点。这是由于在计算签名时增加了计算二级文件块签名的聚合，但在GenProof 阶段减少了文件块签名的聚合。从实验中看出二级文件块签名的聚合带来

的影响几乎可以忽略不计。但本文方案在与文献[5]方案计算性能相差很小的情况下减少了文件块标签带来的额外存储，同时本文实现了TPA代理动态操作，减少了用户的计算量和通信量，以此可以看出本文方案是一个更高效的方案。

本文实验是在Intel Core 2，2.20 GHz的处理器，3.00 GB RAM，32位的Windows 7操作系统下基于Java语言下完成的，所使用的密码学库是Java Pairing-Based Cryptography Library (JPBC)版本为2.0.0。

6 结 束 语

本文构造了一个代理动态操作的云数据拥有性证明方案，该方案通过对私钥分配的设计让第三方审计者代理用户进行动态操作，避免了文献[6]中AA带来的安全隐患，减少了用户的计算和通信消耗，使用户在移动端也能对文件进行动态操作。在性能分析中可以看出，本文方案使用户能够高效方便地对云中数据进行动态操作和完整性审计。

参 考 文 献

[1] 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. [S.l.]: ACM, 2007: 598-609.

[2] 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.

[3] 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. [S.l.]: ACM, 2008: 9.

[4] SHACHAM H, WATERS B. Compact proofs of retrievability[M]//PIEPRZYK J. Advances in Cryptology-ASIACRYPT 2008. Berlin Heidelberg: Springer, 2008: 90-107.

[5] WANG C, WANG Q, REN K, et al. Privacy-preserving public auditing for data storage security in cloud computing[C]// INFOCOM, 2010 Proceedings IEEE. [S.l.]: IEEE, 2010: 1-9.[6]ZHU Y, WANG H, HU Z, et al. Dynamic audit services for integrity verification of outsourced storages in clouds[C]//Proceedings of the 2011 ACM Symposium on Applied Computing. [S.l.]: ACM, 2011: 1550-1557.

[7] WANG Qian, WANG Cong, LI Jin, et al. Enabling public verifiability and data dynamics for storage security in cloud computing[J]. European Symposium on Computer Security-Esorics, 2009, 22(5): 355-370.

[8] BOWERS K D, JUELS A, OPREA A. Proofs of retrievability: Theory and implementation[C]//Proceedings of the 2009 ACM Workshop on Cloud Computing Security. [S.l.]: ACM, 2009: 43-54.

[9] CHANG E C, XU J. Remote integrity check with dishonest storage server[J]. European Symposium on Research in Computer Security: Computer Security, 2008, 5283: 223-237.

[10] SHAH M A, SWAMINATHAN R, BAKER M. Privacy-preserving audit and extraction of digital contents[R]. [S.l.]: Hewlett-Packard Development Company, 2008.

[11] SCHWARZ T S J, MILLER E L. Store, forget, and check: Using algebraic signatures to check remotely administered storage[C]//Conference on Distributed Computing Systems, 2006. [S.l.]: IEEE, 2006: 12.

[12] 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.

[13] BOWERS K D, JUELS A, OPREA A. HAIL: a high-availability and integrity layer for cloud storage[C]// Proceedings of the 16th ACM Computer and Communications Security. [S.l.]: ACM, 2009: 187-198.

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

[15] MERKLE R C. Protocols for public key cryptosystems [C]//2012 IEEE Symposium on Security and Privacy. [S.l.]: IEEE Computer Society, 1980: 122-122.

编 辑 漆 蓉