基于MapReduce的云存储数据审计方法研究

金 瑜 严 冬
(武汉科技大学计算机科学与技术学院 武汉 430065)
(湖北省智能信息处理与实时工业系统重点实验室 武汉 430065)

摘 要 云存储是一种新兴的网络存储技术,它是云计算提供的一个重要服务。云存储因其快速、廉价和方便而广受云用户喜爱。然而,它也给云用户的外包数据带来了许多安全问题。其中一个重要问题就是如何确保半可信云服务器上数据的完整性。因此,云用户和云服务器亟需一个稳定、安全、可信的数据审计方法。随着大数据时代的到来,传统数据审计方案批量处理云环境下海量数据的效率不高;并且,随着移动客户端的流行,传统数据审计方案带给用户的在线负担太过繁重。因此,提出一种基于MapReduce编程框架的云数据审计方案,使用代理签名技术将用户对数据签名计算代理出去,并且并行化处理数据签名和批量审计过程。实验结果表明,所提方法明显提高了批量审计的效率,增强了云存储服务的可用性,并且减轻了用户的在线负担。

关键词 云存储,数据审计,MapReduce,代理签名
中图法分类号 TP393.08 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.02.031

Research on MapReduce-based Data Auditing Method for Cloud Storage

JIN Yu YAN Dong
(College of Computer Science & Technology, Wuhan University of Science and Technology, Wuhan 430065, China)
(Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan 430065, China)

Abstract Cloud storage is a new network storage technology. It is an important service provided by cloud computing. Cloud storage is very popular in cloud users for its characteristics of speediness, low price and convenience. However, it also brings many security problems towards user's outsourced data. One major problem is to ensure the integrity of data at semi-trusted cloud server. So cloud users and cloud servers are both in urgent need of a stable, safe and credible data auditing method. With the arrival of the era of big data, the efficiency of dealing with huge amounts of data in cloud by traditional data batch auditing methods is not high. What's more, with the popularity of mobile clients, the traditional auditing methods bring too much online burden to cloud user. Therefore, this paper proposed a data auditing method based on MapReduce programming framework for cloud storage. It uses the technology of proxy signature, which deals with data signing instead of cloud user. Moreover, it can also complete the work of data signing and batch auditing in parallel. The experiment results show that the proposed method clearly improves the efficiency of batch auditing, enhances the availability of cloud storage service and reduces the online burden towards cloud user.

Keywords Cloud storage, Data auditing, MapReduce, Proxy signature

1 引言

云计算[1]是一种以虚拟化技术为基础、以互联网为载体的按需付费模式,这种模式为用户提供分布式、可靠、高可用、低成本、强大计算能力以及无限存储容量的可能。云计算服务模式[2]大致分为基础设施即服务(IaaS)、平台即服务(PaaS)和软件即服务(SaaS)3种。云存储[3]是在云计算上延伸和发展出来的一个新的概念,是一种新兴的网络存储技术。它为用户提供相对廉价的、可扩展的、与位置无关的存储服务,减轻了用户对数据管理和维护的负担,越来越多的用户开始将数据存储在云服务器上。在享受云存储服务便利的同时,其安全问题也成为用户关心的重要问题。其中数据完整性是用户关心的安全问题[4]之一。由于用户的敏感数据存储在云服务器上,有可能会丢失或被恶意篡改,因此从用户的角度来说,希望在不需要在本地有数据备份的情况下找到一种有效且安全的方法来周期性地验证数据的完整性[5]。云数据审计方案应运而生。从验证者的角度出发,云数据审计[6]方案分为两种:私有审计(Private Auditability)和公众审计(Public Auditability)。私有审计仅允许用户自己对云服务器上的数据进行挑战验证。虽然这样能够完好地保证用户隐

到稿日期:2015-12-16 返修日期:2016-04-05 本文受国家自然科学基金:云计算环境下基于行为的动态信任模型研究(61303117)资助。
金 瑜(1973-),女,博士,副教授,主要研究方向为云计算、对等计算和信任模型;严 冬(1991-),男,硕士生,主要研究方向为云计算安全,E-mail:yandongwust@163.com。

私，并且单个数据验证任务效率较高，但是私有审计对用户计算和通讯上的负担较重，不适用于广大移动客户端，而且由此得到的结论不能让第三方信服。公众审计则可由用户或者任意第三方对服务器上数据进行挑战，即可以让用户将审计任务委托给一个独立且假设完全可信的第三方审计(TPA)[7]，这样做不仅能够减轻用户的在线负担，并且能够得到公开的关于数据是否完整的结论，具有证明作用，因此公众审计方案更加合理。然而，传统的云数据公众审计方案在处理批量审计中云服务器失败响应时效率不高；并且，在传统云数据审计方案中，繁重的签名计算对于日益兴起的移动客户端无法承受。

基于此，本文的主要工作有：

(1)在云数据公众审计方案中，引入代理签名方(Proxy Signer)，并且使用代理签名技术[8]将用户数据签名的过程代理到代理签名方，在保护用户数据隐私的情况下进一步减轻用户的在线负担，使云数据审计方案更适用于移动客户端。

(2)基于分布式系统 Hadoop 编程模型 MapReduce，设计并实现了两个并行化算法，分别将其应用于多用户分块数据代理签名过程和批量审计中云服务器失败响应处理过程，在保证用户隐私和安全性的同时，进一步提高了云数据审计方案的可用性和效率。

2 相关工作

目前，针对云服务器上数据完整性验证问题，研究者提出了各种不同的云数据审计方案。在 2007 年，Ateniese 等人首次定义了公开可验证的数据持有证明方案，并且提出了 PDP 模型，实现了公开可验证数据的完整性。该作者在文献[9]中提出适用于静态数据的 S-PDP 和 E-PDP 两种方案，利用基于 RSA 的同态认证技术和随机取样方法解决了用户需要下载大规模数据带来的巨大通信代价问题，但是 PDP 方案并不支持动态审计和批量审计。同一时期，Juels 等人[10]提出可恢复证明方案(POR 模型)，利用点抽查和纠错码技术不仅能够验证服务器上的数据完整性，而且当数据遭到损坏时还可以以一定概率恢复数据。然而，POR 方案由于“哨兵”个数有限导致完整性验证次数有限，并且不支持任何动态数据操作。

为了有效地在审计方案中支持动态数据操作，2008 年，Ateniese 等人在 PDP 模型的基础上提出可扩展 PDP 方案(Scalable-PDP 模型)[11]，但是该方案仅支持部分动态操作；2009 年，Erway 等人在 Scalable-PDP 模型的基础上提出了动态 PDP 方案(Dynamic-PDP 模型)[12]，该方案支持完整的动态数据操作，但是无法提供用户数据的隐私保护。除了个体缺点之外，文献[11-12]有一个共同缺点，即均不支持批量审计，在云存储多用户环境下验证效率不高。2009 年，Wang 等人提出基于第三方审计(TPA)的云存储安全性方案[13]。当用户的计算能力负担不起持续的审计任务时，TPA 作为公开、可信的独立方为云用户和云服务器提供审计服务。同一年，Wang 等人还提出了一种支持数据动态操作的公开可验证方案[14]。该方案基于第三方审计 TPA，并且使用了 BLS 签名技术，但是此方案也仅支持部分数据的动态操作，并且不能保护数据的隐私。Wang 等人于 2010 年改进了之前方案的缺点，提出实现隐私保护的公共审计方案[15]。该方案利用带有随机掩码的同型验证器保证 TPA 不能直接或间接获得任何跟用户隐私数据有关的信息；同时，该方案还支持多用户批量审计。Wang 等人于 2011 年对之前的动态数据处理方案进行改进，使用 Merkle 哈希树作为存储模型来构造块标签认证，以支持高效的数据动态操作[16]。并且，该方案还结合双线性聚合签名的特性将完整性验证扩展到多用户环境中，实现更加高效的 TPA 多用户批量审计。然而，该方案在处理批量审计中的云服务器失败响应时效率不高。虽然 TPA 上的批量审计能够利用双线性聚集签名技术同时验证来自不同用户多个数据的完整性，如果聚集签名的验证通过，则说明所有用户的数据完整性都得到了保护。但是，一旦聚集签名的验证通不过，TPA 就必须验证单个数据签名，从而找出错误签名；并且，在整个审计过程中，由于存在批量审计，主要的计算任务其实是用户对分块数据的签名过程。也就是说，虽然 TPA 分担了用户验证数据完整性的负担，但是计算分块数据签名的任务依然很繁重，一些简单的移动客户端设备(如手机)完全无法承受，这将会影响云存储服务在移动客户端的普及。

3 本文工作

3.1 系统框架及流程

基于 MapReduce 的云数据审计方案框架如图 1 所示。

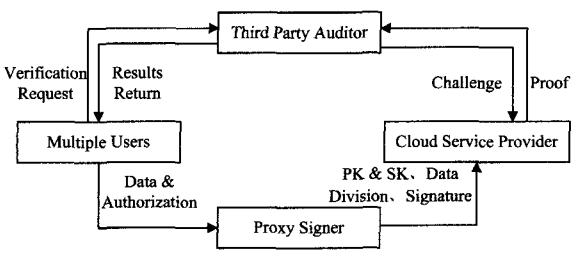

图 1 基于 MapReduce 的云数据审计方案框架

本文提出的基于 MapReduce 的云数据审计方案包含 4 个实体。1)多用户(Multiple User)：享受云服务器提供的云存储服务，有大量的文件需要存储和维护，本方案提供的数据审计可以为多个用户同时服务；2)云服务提供商(Cloud Service Provider)：半可信的云存储服务器，具有巨大的存储空间和计算资源，同时为多个云用户服务；3)第三方审计(Third Party Auditor)：拥有用户所不具有的专业知识和能力，站在用户的立场上评估云存储服务，对用户而言是完全可信的；4)代理签名方(Proxy Signer)：同时代替多个用户对其数据进行签名，减轻用户在线负担，同时提高云存储服务性能。

基于 MapReduce 的云数据审计方案流程如图 2 所示。

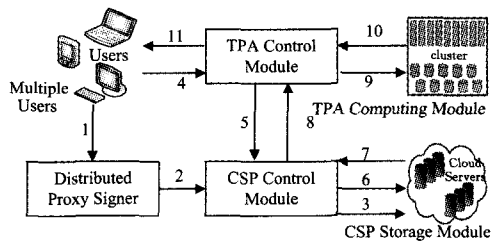

图 2 基于 MapReduce 的云数据审计方案流程

具体包括：

1）多个用户初始化公私钥，并且计算代理授权，然后用户将需要存储的数据和授权发送给代理签名方。

2）代理签名方一次性处理 K 个用户的授权任务，在收到授权时，首先验证授权的正确性，如果授权正确，则分别将 K 个用户的文件分为 n 块，并且利用 MapReduce 编程框架同时对 $K*n$ 个数据块进行签名，最后代理签名方将 K 个用户的数据及对应签名发送给 CSP 进行存储和维护。

3）CSP 收到代理签名方的数据及签名后，将它们存储在 CSP 存储模块中。

4）如果某个用户 $k(k\in[1,K])$ 需要验证其存储在云服务器上的数据的完整性，需发送请求及数据标签 T 至 TPA，要求返回数据完整性验证结果。

5）TPA 收到用户 k 的请求后，在 $[1,n]$ 中产生一个随机数据集 I，对于数据集 I 中的每个元素都对应产生一个审计随机值，TPA 将数据标签 T、数据集 I 及其随机值作为挑战 *chal* 发送给 CSP。

6）－8）CSP 收到 *chal* 后，从 CSP 存储模块中找到 T 对应的数据及数据签名，并且利用本地存储的数据、数据签名、数据集 I 及其随机值计算证据 *proof*，将结果返还给 TPA。

9）－11）TPA 一次性处理 K 个 *proof*，利用双线性聚集签名技术，可以将 K 个 *proof* 里面包含的签名聚集为一个签名。如果该聚集签名验证通过，则给所有 K 个用户返回 TURE；如果聚集签名验证失败，则分别处理 K 个 *proof*，利用 MapReduce 编程框架并行化验证 K 个方程（见式(1)），得出 K 个不同的验证结果，并返回给用户。

3.2　基于 MapReduce 的云数据代理签名

3.2.1　云数据审计基本方案

Setup：用户选择随机元素 $x\leftarrow Z_p$，随机元素 $u\leftarrow G_1$，计算 $v\leftarrow g^x$ 和 $w\leftarrow u^x$，g 是乘法循环群 G_2 的生成元，双线性映射 $e:G_1\times G_2\rightarrow G_t$，其中 G_1，G_2，G_t 均为 p 阶乘法循环群。定义私钥 $sk=(x)$，公钥 $pk=(u,g,v,w)$。将文件分成 n 个块，$F=(m_1,m_2,\cdots,m_n)$，然后用户对 m_i 进行签名：$\sigma_i\leftarrow(H(m_i)\cdot u^{m_i})^x(i=1,2,\cdots,n)$，其中 $H(\cdot)$ 是一个将二进制字符串映射到 G_1 的哈希函数。签名集合表示为 $\Phi=\{\sigma_i\}_{1\leqslant i\leqslant n}$，然后把 $\{F,\Phi\}$ 发送到云服务器端并删除本地文件。

Audit：用户首先请求 TPA 验证数据的完整性，TPA 选择一个随机 c-element 集合 $I=\{s_1,\cdots,s_c\}$ 来表示 $[1,n]$，其中 $s_q=\pi_{k_{prp}}(q),1\leqslant q\leqslant c,k_{prp}$ 是 TPA 为每次审计而随机选择的，π 是一个伪随机置换函数。假设 $s_1\leqslant\cdots\leqslant s_c$，对于每个 $i\in I$，TPA 都会选择一个随机值 V_i，然后 TPA 会发送一个挑战给服务器：$chal=\{(i,V_i)\}_{i\in I}$。服务器收到 *chal* 之后，计算 $r=f_{k_{prf}}(chal)\leftarrow Z_p$，$k_{prf}$ 是服务器为每次审计而随机选择的，f 是一个伪随机函数。并且计算随机掩码 $R=(w)^r\in G_1$。样品数据块线性组合表示为：$\mu'=\sum_{i\in I}v_im_i$，为了保护用户数据隐私，计算 $\mu=\mu'+r\cdot h(R)\in Z_p$。同时服务器计算聚集签名：$\sigma=\prod_{i\in I}\sigma_i^{v_i}\in G_1$。然后服务器把 $proof=\{\mu,\sigma,R,H(m_i)_{i\in I}\}$ 发送到 TPA。TPA 收到 *proof* 之后，验证式(1)是否相等以判断数据完整性是否得到了保护。

$$e(\sigma\cdot R^{h(R)},g)\overset{?}{=}e(\prod_{i\subseteq I}H(m_i)^{v_i}\cdot u^{\mu},v)) \tag{1}$$

Batch Auditing：假设有 K 个用户都需要验证数据完整性，某用户 k 有文件 $F_k=(m_{k,i},\cdots,m_{k,n})$，其中 $k\in[1,K]$。对于用户 k，选择一个随机 $x_k\in Z_p$ 作为私钥，对应的公钥为 $(v_k,w_k,g_k,u_k)=(g^{x_k},u^{x_k},g_k,u_k)$。用户 k 的数据块 $m_{k,i}$ 的签名为 $\sigma_{k,i}\leftarrow[H(k\parallel m_i)\cdot(u_k)^{m_{k,i}}]^{x_k},i\in[1,n]$。在服务器收到挑战 *chal* 之后，随机为每个用户选择 r_k 并且计算：

$$\mu_k=\prod_{i\subseteq I}V_im_{k,i}+r_kh(R_k)\in Z_p$$

$$\sigma=\prod_{k=1}^{K}(\prod_{i=s_1}^{s_c}\sigma_{k,i}^{v_i}),R_k=(w_k)^{r_k}=(u_k^{x_k})^{r_k}\circ proof=\{\sigma,\{\mu_k\}_{1\leqslant k\leqslant K},\{R_k\}_{1\leqslant k\leqslant K},H(k\parallel m_i)_{i\in I}\}$$

作为证据返还给 TPA。TPA 收到证据之后，利用 *proof* 和公钥验证式(2)以一次性判断 K 个用户的数据完整性。

$$e(\sigma\cdot\prod_{k=1}^{K}R_k^{h(R_k)},g)\overset{?}{=}\prod_{k=1}^{K}e(\prod_{i=s_1}^{s_c}[H(k\parallel m_i)]^{V_i}\cdot(u_k)^{\mu_k},v_k) \tag{2}$$

3.2.2　云数据审计代理签名详细过程

本方案是基于 KPW 代理签名[17]的一种改进方案，原 KPW 代理签名方案提出了由证书来生成代理秘钥的概念，设计实现了一个带有证书的部分代理数字签名方案，但是不满足不可否认性和不可伪造性。本文结合云数据审计系统提出的代理签名具体包括 Setup，Authorization，Proxy Signature 和 Audit 4 个步骤。

Setup：用户选择随机元素 $x\leftarrow Z_p$，随机元素 $u\leftarrow G_1$，计算 $v\leftarrow g^x$，g 是乘法循环群 G_2 的生成元，定义私钥 $sk=(x)$，公钥 $pk=(u,g,v)$。

Authorization：对于较小的数据，用户可以自己签名，然后上传至云服务器。但是对于处理不了的大型数据，用户可以选择将签名工作授权给代理签名方，这时用户选择随机元素 $y\leftarrow Z_p$，并产生证书 m_w，证书记载了原签名者的身份、代理签名方的身份以及其他安全要求方面的信息。计算 $Q=g^y$，$P=h(m_w,Q)\cdot x+y$，其中 $h(\cdot)$ 是一个单向公开哈希函数。用户将 $\{F,(P,Q,m_w)\}$ 作为代理授权发送给代理签名方。

Proxy Signature：代理签名方收到授权信息之后。首先计算杂凑值 $e=h(m_w,Q)$，然后验证 $g^P=v^e\cdot Q$。若验证通过，则说明是正确的代理授权。代理签名方选择私钥 $z\leftarrow Z_p$，计算审计私钥 $PP=P+z\cdot e$，审计公钥 $vv=v^e\cdot g^{z\cdot e}\cdot Q$ 以及 $w=u^{PP}$，将文件分成 n 个块，即 $F=(m_1,m_2,\cdots,m_n)$。数据块 m_i 签名：$\sigma_i\leftarrow(H(m_i)\cdot u^{m_i})^{pp}(i=1,2,\cdots,n)$，签名集合表示为 $\Phi=\{\sigma_i\}_{1\leqslant i\leqslant n}$。代理签名方将 $\{F,\Phi\}$ 发送给 CSP 进行存储和维护。

Audit：当 CSP 收到来自 TPA 的挑战 *chal* 时，服务器提供证据 $proof=\{\mu,\sigma,R,H(m_i)_{i\in I}\}$ 给 TPA，其中 $\mu=\mu'+r\cdot h(R)$，其中 μ' 表示 *chal* 中样品数据块的线性组合：$\mu'=\sum_{i\in I}V_im_i$。R 表示随机掩码：$R=(w)^r=(u^{PP})^r$，其中 $r=f_{K_{prf}}(chal)$，K_{prf} 是 TPA 为每次审计任务而随机产生的。聚集签名 $\sigma=\prod_{i\in I}\sigma_i^{V_i}\in G_1$。而 TPA 收到 *proof* 之后，通过验证式(3)来判断数据完整性是否得到了保护。

$$e(\sigma\cdot R^{h(R)},g)\overset{?}{=}e(\prod_{i\in I}H(m_i)^{V_i}\cdot u^{\mu},vv) \tag{3}$$

式(3)的验证如下：

$$\text{左边}=e(\prod_{i\in I}\cdot[H(m_i)^{pp\cdot V_i}\cdot u^{pp\cdot m_i\cdot V_i}]\cdot u^{PP\cdot r\cdot h(R)},g)$$

$$=e(\prod_{i\in I}[H(m_i)^{V_i}\cdot u^{m_i\cdot V_i}]\cdot u^{r\cdot h(R)},g^{PP})$$
$$=e(\prod_{i\in I}H(m_i)^{v_i}\cdot u^{\mu},g^{PP})$$

且 $g^{PP}=g^{x\cdot e+y+z\cdot e}=v^{f}\cdot g^{y}\cdot g^{z\cdot e}=v^{f}\cdot g^{z\cdot e}\cdot Q=vv$。

所以，左边＝右边。通过以上数学推导，随机掩码 R 不会影响验证结果，且保证了数据隐私。签名中包含证书 m_w 和私钥 z，代理签名方不能获取用户私钥 x，未经授权不能进行签名；用户也不能获取代理私钥 z，也就不能伪造代理签名，所以本方案的代理签名具有不可伪造性和不可否认性，这保证了代理签名的安全性。

3.2.3 基于 MapReduce 的代理签名算法

本文提出的云数据审计方案的代理签名方是分布式的，可以同时处理来自多个用户的签名代理请求，以提高代理签名的效率。MapReduce[18]是一种编程模型，用于大规模数据集的并行运算，本文以 K 个用户为例，基于 MapReduce 编程框架定义 Map 和 Reduce 两个算法，并行化处理 K 个用户的签名任务，具体算法如下。

算法 1 Map()//for proxy signature

```
输入:SKFile_1,…,SKFile_K,UserFile_1,…,UserFile_K
输出:〈T_1,SK_1〉,〈T_1,F_1〉,〈T_1,u_1〉…
Begin
  While(有输入文件)
    If (fileTag==SKFile)
      //表示用户 k
      输出键值对〈T_k,PP_k〉;
    Endif
    Else if (fileTag==UserFile)
      value=FileContent. split();
      If (value. hasMoreTokens())
      //输出键值对〈T_k,F_k〉或〈T_k,u_k〉:
        context. write(T_k,value. nextToken());
      Endif
    Endif
  Endwhile
End
```

对于用户 $k(k\in[1,K])$ 的数据块 $m_{k,i}$，都有对应签名：$\sigma_{k,i}\leftarrow[H(k\parallel m_i)\cdot(u_k)^{m_{k,i}}]^{PP_k}$。如算法 1 所示，代理签名 Map 算法输入为 $2*K$ 个文件，其中 K 个 SKFile 存放代理签名方为每个用户计算的代理签名私钥 PP_k，另外 K 个 UserFile 存放代理签名所需的用户数据 F_k 和签名公钥 u_k。MapReduce 框架会将输入文件按行分解(默认是按行，可以修改源码改变)依次传给多个 Map 函数，框架会根据配置设置多个进程或 PC 来处理 Map 函数。代理签名 Map 的输出为 $3*K$ 个键值对：$\langle T_k,PP_k\rangle$，$\langle T_k,F_k\rangle$，$\langle T_k,u_k\rangle$，T_k 表示用户数据标签。在 Map 函数执行完成后，MapReduce 编程框架的 Combine 函数会将 $3*K$ 个键值对中 key 相同的 value 联接到一个新的 value。也就是说，Combine 之后，仅剩 K 个键值对〈key，values〉，对应着 K 个用户数据。Map 和 Combine 所做工作就是从不同文件中收集每个用户数据签名所需参数，然后归并到一个键值对中，作为 Reduce 函数的输入。

算法 2 Reduce()//for proxy signature

```
输入:K 个键值对〈key,values〉
输出:n*K 个签名
Begin
  For (int i=0;i<K;i++)
    将用户数据进行分块 F_k=(m_{k,i},…,m_{k,n});
    For (int i=0;i<n;i++)
      签名算法:sig_{k,i}=Signature(m_{k,i},u_k,pp_k);
      输出签名〈T_k,sig_{k,i}〉;
    Endfor
  Endfor
End
```

如算法 2 所示，K 个键值对作为 Reduce 函数的输入，每个键值对对应每个用户数据的签名参数，将数据 F_k 分块之后，对其分块数据逐一签名，结果返回 $n*K$ 个签名。整个代理签名过程都是在 Hadoop 分布式系统中完成的，即使有某个计算节点故障，它的计算任务也会由其他节点代替，不会造成太大损失，这增强了云存储服务的可用性。而且，代理签名方进一步减少了用户的计算负担，使得整个云数据审计服务更适合移动客户端，用户可以更便捷地享受云存储服务。

3.3 基于 MapReduce 的云数据批量审计

本节主要介绍基于 MapReduce 的云存储方案批量审计工作及其算法。Wang 在文献[16]中提到，批量审计用双线性聚集签名一次性验证多个用户的签名，如果验证成功则说明所有用户的数据的完整性得到了保护；如果验证失败，可以采用一种二分查找的方法找出聚集签名中出问题的签名。简单来说就是将聚集签名分为两半，逐一递归验证。但是如果聚集签名中出问题的签名存在两个或者多个，这种二分查找的方法效率还是不高，还是得由 TPA 对所有用户签名进行逐一验证。由此可见，聚集签名失败响应的处理工作较大，而且集中式 TPA 存在单点失效的风险。所以本文将 TPA 分布式化，并用 MapReduce 编程框架实现批量审计算法失败响应处理过程，不仅提高了云数据审计服务的可用性，而且在一定程度上提高了 TPA 的审计效率。

算法 3 Map()//for invalid response of batch auditing

```
输入:PKFile_1,…,PKFile_K,proofFile_1,…,proofFile_K
输出:〈T_1,u_1〉,〈T_1,g_1〉,〈T_1,vv_1〉,〈T_1,proof〉…
Begin
  While(有输入文件)
    If (fileTag==PKFile)
      value=FileContent. split();
      If (value. hasMoreTokens())
      //输出〈T_k,g_k〉,〈T_k,u_k〉,〈T_k,vv_k〉
        context. write(T_k,value. nextToken());
      Endif
    Else if (fileTag==proofFile)
      value=FileContent. split();
      If (value. hasMoreTokens())
      //输出〈T_k,σ_k〉,〈T_k,μ_k〉,〈T_k,H(m_i)_k〉,〈T_k,R_k〉
        context. write(T_k,value. nextToken());
      Endif
    Endif
  Endwhile
End
```

如算法 3 所示，以 K 个用户为例，审计验证式(1)中的验

证参数包括：验证公钥 u,g,vv 和证据 $\mu,\sigma,R,H(m_i)$，公钥和证据分别来自文件 PKFile 和文件 proofFile，所以 Map 函数的输入为 $2*K$ 个文件。Map 函数的输出为 $7*K$ 个键值对，经过 Combine 函数，所有键相同的键值对都被归并到一个键值对〈key，values〉中，之后就只剩 K 个键值对，其对应着 K 个用户文件，每个键值对的 key 对应某个用户数据的标签 T，而 values 对应着这个用户数据完整性验证所需参数。

算法 4　Reduce()//for invalid response of batch auditing

输入：K 个键值对〈key，values〉

输出：K 个结果〈T_1，TRUE/FALSE〉…

```
Begin
  For ( int i=0;i<K;i++ )
    从〈key,values〉的 values 中取出验证公钥 u,g,vv 和 CSP 提供的
    证据 μ,σ,R,H(mi);
    验证式(1)并且返回 TRUE 或 FALSE;
    输出验证结果:〈Tk,TRUE/FALSE〉;
  Endfor
End
```

如算法 4 所示，Reduce 函数每次处理一个输入的键值对〈key，values〉，首先取出 values 中的公钥和验证参数，通过验证式(1)来判断该键值对所对应用户数据的完整性是否得到了保护，然后将验证结果返回给 key 对应的用户。每个 Reduce 函数处理一个键值对，Master 节点会分配多个 Slave 节点来处理 Reduce 过程，即每个 Slave 节点完成一次单个文件验证过程，然后将所有验证结果返回给 Master 节点，最后 Master 节点将收集的结果返回给用户。分布式环境下处理批量审计算法在失败响应过程的效率提升较高，并且如果出现节点失效，Master 节点会将该节点负责的任务分配到其他节点。由此可见，基于 MapReduce 的云存储批量审计提高了服务可用性和审计效率。

4　性能分析

本文实现了基于 MapReduce 的云数据审计方案原型系统，其中 CSP 存储模块使用分布式 NoSQL 数据库 Cassandra[19]，而其代理签名模块和 TPA 计算模块使用分布式系统 Hadoop 实现。

4.1　系统环境搭建

基于 MapReduce 的云数据审计系统提供支持多用户的云存储服务。本文所涉及的实体都是分布式的，其中用户端是多用户同时请求审计服务；代理签名方是用分布式系统 Hadoop 完成多用户的代理签名工作；CSP 端是云存储服务器，本系统采用分布式数据库 Cassandra 来存储用户数据及其签名；TPA 端由于主要涉及计算过程，本系统搭建分布式集群 Hadoop 来完成其审计工作。具体系统环境如下。

(1)硬件环境

8 台 PC 机；CPU：3.00GHz；RAM：2048MB。

(2)软件环境

操作系统：32 位 Windows 8.1；编程语言：Java；编程工具：Eclipse，jdk1.8.0_25，并装有 Hadoop 集群管理插件；Hadoop 版本：hadoop-0.20.2；Cassandra 版本：cassandra-1.2.11；Cygwin：模拟 Linux 环境来启动 Hadoop 集群；Java Pairing-Based Cryptography (JPBC)：jpbc-2.0.0，采用密码学 API 来完成签名和审计算法。

本系统使用 3 台 PC 机来搭建 Cassandra 云存储服务器，并将其用于存储和维护多用户的数据及其签名，并且在数据审计阶段提供证据。分布式数据库 Cassandra 没有主从节点之分，各个节点之间地位平等，可以良好地处理单点失效问题，其默认数据备份数为 3 份，以便在某一节点数据丢失时进行数据恢复，而数据一致性问题由集群底层本身负责。本系统使用另外 5 台 PC 机来搭建 Hadoop 分布式系统，以完成代理签名和批量审计失败响应处理过程。1 台 PC 作为 Master 节点：它是 MapReduce 过程的主要控制节点，负责分发任务给 Slave 节点并收集它们的处理结果；4 台 PC 机均作为 Slave 节点：其是处理 MapReduce 过程的主要计算节点，并将计算结果返回给 Master 节点。本文根据该原型系统做了 5 个实验以测试其性能，分别测试了代理签名算法的性能、多用户代理签名的性能和批量审计失败响应处理的性能。

4.2　代理签名算法性能分析

为了测试基于 MapReduce 的云数据审计方案代理签名算法的性能，实验 1—实验 3 采集了 250k，500k，1M，2M，5M，10M，15M，20M，25M，30M 共 10 个不同大小的数据集。实验 1 的结果如图 3 所示，其示出了在单用户的情况下，分块数 $n=10$，基于不同大小数据，代理签名方案与 Wang 方案[16]关于用户开销进行对比；实验 2 的结果如图 4 所示，其示出了在单用户的情况下，实验数据大小为 10M，分块数 n 分别为 10，20，…，100，代理签名方案与 Wang 方案[16]的用户开销对比。

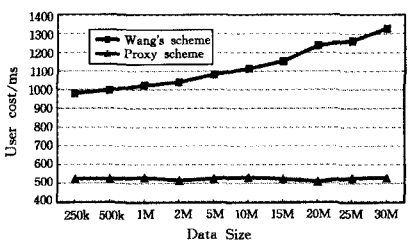

图 3　$n=10$ 时不同大小数据用户开销

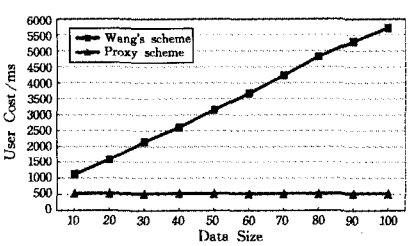

图 4　data=10M 时不同分块数用户开销

由实验 1 和实验 2 可以看出：1)Wang 的方案中用户开销随着数据大小的增长而增长缓慢，随着分块数的增长而迅速增长，这是因为虽然数据大小在增长，但是签名次数没变，而分块数越多，签名次数也越多，签名时间也就越长；2)签名开销占用户开销的绝大部分，而且会随着数据大小和分块数的增长而增长，因此为了进一步减少用户的在线负担，引入代理签名方是非常有必要的；3)Wang 的方案中用户开销会随着数据大小的增大而增大，也会随着分块数的增大而增大，而本文所提出代理签名的方案的用户开销几乎是常量，因为用户

只需要生成公私钥和产生授权，其他工作由代理签名方完成，因此本文提出方案确实进一步减少了审计系统中用户的在线负担，提高了云存储服务的可用性。

4.3 多用户代理签名性能分析

由于本文引入了代理签名方，而代理签名方面向多用户，需要同时处理多个签名代理请求，计算量在云环境下会更大，所以本文用分布式系统 Hadoop 的 MapReduce 框架来实现代理签名过程。

为了进一步测试代理签名方的性能，实验 3 从多用户角度出发，测试了在用户数 $K=10$、文件分块数 $n=10$ 且不同数据大小的情况下，代理签名方案（分布式系统 Hadoop 的 Slave 节点数 $N=2,3,4$）与 Wang 方案[16]初始化时间的对比（Wang 方案初始化时间为用户初始化时间，本文所提及方案的初始化时间为用户初始化时间加上代理方初始化时间总和）。

实验 3 的结果如图 5 所示，基于 MapReduce 的代理签名方案初始化时间（包括 $N=2,3,4$）明显小于 Wang 方案的初始化时间，并且随着 Slave 节点个数 N 的增大，初始化时间越来越小，而 Wang 的方案随着数据大小增长，初始化时间会呈线性增长。因此本文提出的基于 MapReduce 代理签名方案在效率上要优于 Wang 的方案。

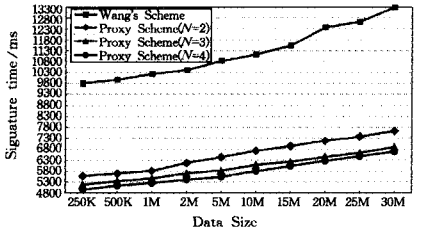

图 5 分布式代理签名初始化开销

4.4 批量审计失败响应处理性能分析

为了测试基于 MapReduce 的批量审计失败响应处理的性能，实验 4 在节点数 $N=1,2,3,4$ 情况下测量了 100 个失败响应审计时间，并与 Wang 的方案[16]进行了对比。

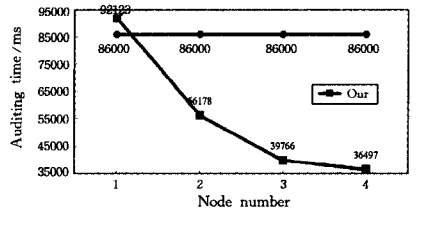

图 6 不同节点数量审计时间

实验 4 的结果如图 6 所示，可以得出以下结论：1）在 $N=1$ 的情况下，本方案的审计时间多于 Wang 方案，因为本文提出的方案的审计时间包括 Hadoop 的 MapReduce 编程框架的初始化时间和 Master 与 Slave 节点之间信息交互时间，但是这个时间基本为固定值，不会随着数据大小或分块数呈线性增长；2）节点数 $N=2,3,4$ 时，审计时间明显小于 Wang 的方案，这说明分布式审计方案的效率明显高于单机审计方案。

实验 5 测量的是在节点个数 $N=4$，失败响应为 10，…，50 的情况下，本文提出的方案与 Wang 的方案的审计时间对比。实验 5 的结果如图 7 所示，两种方案的审计时间都会随着失败响应个数的增加而增加，但是本文基于 MapReduce 的批量审计方案的审计时间增长较慢，而 Wang 的方案的审计时间呈线性增长。也就是说，本文方案的优势会随着失败响应个数的增多而越来越大，这说明本文方案比 Wang 的方案在效率上有所提升，并且更适用于拥有海量数据的云环境。

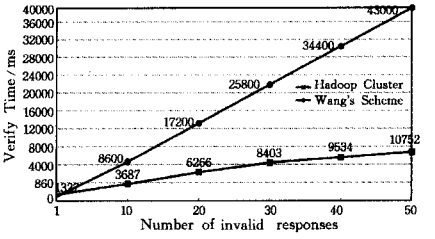

图 7 不同失败响应数量审计时间

结束语 由于原有审计方案中用户计算开销不适用于移动客户端，本文提出了一种基于 MapReduce 的云存储数据审计方法，并且在原有审计方案的基础上引入了代理签名方，在保证安全性的同时，代替用户完成数据签名工作，将用户开销减少到近乎为常量。并且还考虑到云存储服务的计算效率和服务性能，本文基于 MapReduce 编程框架，设计了两个算法，分别实现了多用户代理签名过程和批量审计失败响应处理过程。根据实验性能分析，本文所提方案确实减少了用户的在线负担，并且两个 MapReduce 算法提高了云储存服务效率和可用性。

对于本文所提出方案，其实还有许多后续工作要做，例如，若代理签名方不是完全可信的，同时 TPA 审计结果为数据完整性遭到破坏，则有可能是云服务器篡改了用户数据，也有可能是代理签名没有如实地完成签名工作，代理签名方和云存储服务器的问责该如何解决？

参 考 文 献

[1] LIU P. Cloud Computing (Second edition)[M]. Beijing: Beijing Electronic Industry Press, 2011.

[2] MELL P, GRANCE T. The NIST definition of cloud computing [J]. Communications of the ACM, 2011, 53(6): 50.

[3] HASAN R, YURCIK W, MYAGMAR S. The evolution of storage service providers: techniques and challenges to outsourcing storage[J]. ACM Workshop on Storage Security & Survivability ACM, 2005: 1-8.

[4] MORSY M A, GRUNDY J, MÜLLER I. An Analysis of the Cloud Computing Security Problem[C]//Proceedings of APSEC 2010 Cloud Workshop. Sydney, Australia, 2010.

[5] GIANI A, BITAR E, GARCIA M, et al. Smart Grid Data Integrity Attacks[J]. IEEE Transactions on Smart Grid, 2013, 4(3): 1244-1253.

[6] OUALHA N, LENEUTRE J, ROUDIER Y. Verifying remote data integrity in peer-to-peer data storage: A comprehensive survey of protocols[J]. Peerto-Peer Networking and Applications, 2012, 5(3): 231-243.

[7] WANG C, REN K, LOU W, et al. Toward publicly auditable secure cloud data storage services[J]. Network, IEEE, 2010, 24 (4): 19-24.

[8] MAMBO M, USUDA K, OKAMOTO E. Proxy signature: delegation Of the power to sign messages[J]. Ieice Transactions on

Fundamentals of Electronics Communications & Computer Sciences, 1996, 79(9): 1338-1354.

[9] ATENIESE G, BURNS R C, 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.

[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. ACM, 2007: 584-597.

[11] 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. ACM, 2008: 1-10.

[12] ERWAY C. Dynamic provable data possession[C] // Proceedings of the 16th ACM Conference on Computer and Communications Security. ACM, 2009: 213-222.

[13] WANG C, WANG Q, REN K. Ensuring data storage security in cloud computing[C] // 17th International Workshop on Quality of Service, 2009. IWQoS. 2009.

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

[15] WANG C, WANG Q, REN K, et al. Privacy-preserving public auditing for data storage security in cloud computing[C] // 2010 Proceedings IEEE INFOCOM. 2010: 1-9.

[16] 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, 2011, 22(5): 847-859.

[17] KIM S, PARK S, WON D. Proxy signatures, Revisited [J]. Information and Communications Security, 1997: 223-232.

[18] http://wiki.apache.org/hadoop/#MapReduce.

[19] GUO Peng. Cassandra in Action. China Machine Press.

(上接第 194 页)

通过计算可得 VHF 算法的 S 盒的最大线性概率是 $2^{-2.83}$，通过程序计算得到 VHF 算法 14 轮活动 S 盒的个数 LS，如表 3 所列。由此可得 VHF 的 7 轮最大线性概率为 $LCP_{\max}^{7r} \leqslant 2^{48\times(-2.83)} = 2^{-135.84} \leqslant 2^{-128}$。当迭代轮数大于 7 时，找不到一个有效的线性特征进行分析，所以完整轮数的 VHF 算法可以抵抗线性分析。

4.2 不可能差分分析

不可能差分分析[8]对 VHF 算法来说是一种非常有效的攻击手段。J. Kim 等[9]发明了一种矩算法 μ-method 用来对分组密码的结构进行不可能差分分析，该方法能够找到不同的不可能差分路径。采用此方法对 VHF 算法进行不可能差分分析得到的最大轮数为 6，找到了 8 条不可差分路径。

$$(0,0,0,\alpha,0,\alpha,\alpha,\alpha) \xrightarrow{6r} \not\rightarrow (0,0,0,0,\alpha,\alpha,\alpha,\alpha), p=1$$

$$(0,0,\alpha,0,\alpha,\alpha,\alpha,0) \xrightarrow{6r} \not\rightarrow (0,0,0,0,\alpha,\alpha,\alpha,\alpha), p=1$$

$$(0,\alpha,0,\alpha,\alpha,\alpha,0,0) \xrightarrow{6r} \not\rightarrow (0,0,0,\alpha,\alpha,\alpha,\alpha,0), p=1$$

$$(0,\alpha,\alpha,\alpha,0,0,0,\alpha) \xrightarrow{6r} \not\rightarrow (0,\alpha,\alpha,\alpha,\alpha,0,0,0), p=1$$

$$(\alpha,0,0,0,\alpha,0,\alpha,\alpha) \xrightarrow{6r} \not\rightarrow (\alpha,\alpha,0,0,0,0,\alpha,\alpha), p=1$$

$$(\alpha,0,\alpha,\alpha,\alpha,0,0,0) \xrightarrow{6r} \not\rightarrow (0,0,\alpha,\alpha,0,0,0,0), p=1$$

$$(\alpha,\alpha,0,0,0,\alpha,0,\alpha) \xrightarrow{6r} \not\rightarrow (\alpha,\alpha,\alpha,0,0,0,0,\alpha), p=1$$

$$(\alpha,\alpha,\alpha,0,0,0,\alpha,0) \xrightarrow{6r} \not\rightarrow (\alpha,\alpha,\alpha,\alpha,0,0,0,0), p=1$$

其中，$\alpha \in GF(2^8)$，表示非零差分。由此可知，不可能差分分析对 VHF 算法攻击无效。

结束语　本文提出了一种基于双伪随机变换和 Feistel 结构的轻量级分组密码算法 VHF，并将其应用于无线通信中低成本嵌入式移动终端。通过产生加密变换表 S[256]进行行伪随机变换和对角线伪随机变换，实现混淆和扩散。通过软件效率测试与 MIBS, CLEFIA, PRESENT 等轻量级分组密码算法进行比较得出，VHF 算法的软件实现性能优异；通过硬件实现与其他轻量级分组密码算法进行比较，得到的 1632 个 GE 数高于国际标准算法 CLEFIA 得到的 5979 个 GE 数；总体来说，VHF 算法的软件效率和硬件实现都高于同为面向 8 位平台的国际标准 CLEFIA 算法。对 VHF 算法采用差分、线性分析和不可能差分分析方法进行安全分析，验证了 VHF 算法的安全性。

参 考 文 献

[1] DAEMEN J, RIJMEN V. The design of Rijndael: AES-the advanced encryption standard [M]. Berlin Heidelberg: Springer, 2002: 10-18.

[2] SHIRAI T, SHIBUTANI K, AKISHITA T, et al. The 128-bit Blockcipher CLEFIA, 2007[C] // Proceedings of the 14th International Conference on Fast Software Encryption. Berlin Heidelberg: Springer, 2007: 181-195.

[3] BOGDANOV A, KNUDSEN L R, LEADER G, et al. PRESENT: An ultra-lightweight block cipher[J]. Lecture Notes in Computer Science, 2007: 450-466.

[4] WU W L, FENG D G, ZHANG W T. Design and analysis of Block cipher [M]. Beijing: TsingHua University Press, 2009: 5-20. (in Chinese)
吴文玲，冯登国，张文涛. 分组密码的设计与分析[M]. 北京：清华大学出版社，2009：5-20.

[5] IZADI M, SADGHIYAN B, SADEGHIAN S S, et al. MIBS: a new lightweight block cipher[C] // International Conference on Cryptology and Network Security Berlin Heidelberg: Springer, 2009: 334-348.

[6] BIHAM E, SHAMIR A. Differential cryptanalysis of the data encryption standard [M]. New York: Springer-Verlag, 1993.

[7] MATSUI M. Linear Cryptanalysis Method for DES Cipher, 2007 [M]. Berlin Heidelberg: Springer, 1994: 386-397.

[8] BIHAM E, BIRYUKOV A, SHAMIR A. Cryptanalysis of Skipjack Reduced to 31 Rounds using Impossible Differentials, 1999 [M]. Berlin Heidelberg: Springer, 1999: 12-23.

[9] KIM J, HONG S, SUNG J, et al. Impossible Differential Cryptanalysis for Block Cipher Structures[J]. Lecture Notes in Computer Science, 2003, 2904: 82-96.