第36卷第11期 东北大学学报(自然科学版) Vol. 36, No. 11
2015年11月 Journal of Northeastern University(Natural Science) Nov. 2015

doi: 10.3969/j.issn.1005-3026.2015.11.010

面向移动云计算的轻量级数据完整性验证方法

沈志东[1], 林 晨[1], 佟 强[2]
(1. 武汉大学 国际软件学院, 湖北 武汉 430079; 2. 东北大学 软件学院, 辽宁 沈阳 110819)

摘 要: 研究了面向移动云计算的数据完整性验证技术,依托BLS短签名算法和Merkle哈希树,提出了一种适合在移动云计算环境中部署的数据完整性验证方案. 该方案针对移动云计算环境中的移动设备计算能力较低和通信传输能力较弱的情况进行设计,能以相对较少的计算量和较低的数据通信量完成可信度较高的数据完整性验证. 该方案还具有支持验证外包、无需源文件块直接参与验证、验证中无状态信息保存、以及支持对云端数据的动态操作等特性,适合于移动云计算环境中面向数据的应用.

关 键 词: 移动云计算;数据完整性验证;轻量级数据;数据安全;云计算安全
中图分类号: TP 309 **文献标志码**: A **文章编号**: 1005-3026(2015)11-1562-05

A Method for Lightweight Verification on Data Integrity in Mobile Cloud Computing Environment

SHEN Zhi-dong[1], *LIN Chen*[1], *TONG Qiang*[2]
(1. International School of Software, Wuhan University, Wuhan 430079, China; 2. School of Software, Northeastern University, Shenyang 110819, China. Corresponding author: SHEN Zhi-dong, E-mail: shenzd@whu.edu.cn)

Abstract: The data integrity verification in mobile cloud computing environment was analyzed. A method for lightweight verification on data integrity was proposed, which was based on BLS short signature scheme and Merkle Hash tree data structure. Considering the resource constraints and "harsh" environment of mobile devices, the method is suitable for mobile computing environment and has many good features of proposed schemes. Through the method the verification process could be accomplished with high accuracy, light computing and low data transmission. In this method, outsourcing verification was supported, data integrity could be verified without using directly source file blocks, process of verification could be stateless, and dynamic data operation could be archived in cloud. It is suitable for the service on data in mobile cloud computing environment.

Key words: mobile cloud computing; data integrity verification; lightweight data; data security; cloud computing security

近年来,传统的云计算开始有向移动云计算发展的趋势[1-2],移动云计算中面向数据的应用服务要求云端拥有完备的数据保密机制,以免用户重要的私有数据丢失或者被盗[2-4]. 然而,云服务供应商的可信度是难以确定的,以云端数据完整性为例,一些遭受拜占庭失效(Byzantine failure)的云存储服务供应商可能会选择向用户隐藏数据错误,或者为节约成本故意删除用户不常用的数据[4-5]. 因此,将数据安全机制完全交由云端负责是不可取的,用户必须能够按照意愿对云端数据完整性进行验证. 文献[6]提出了一种基于RSA算法的数据完整性远程验证方案,对整个文件做RSA指数运算,服务器的运算处理开销较大. 文献[7]提出了一个基于标记(sentinal)的方案以验证云端数据是否完整,但此方案缺乏对动态数据的支持,并且随着标记码的减少需要重

收稿日期: 2014-03-30
基金项目: 国家自然科学基金资助项目(61003185); 辽宁省教育厅一般项目(L2013098).
作者简介: 沈志东(1975-),男,江苏淮安人,武汉大学副教授,博士.

新上传整个数据文件. 文献[8]中提出的使用同态可验证标签的方案可降低对通信带宽的要求，验证响应较快，但也缺乏对动态数据更新的有效支持. 考虑到移动云计算环境中的移动端计算和存储能力的局限性，将复杂的数据完整性验证处理放在移动端也是不合适的. 需提供一种解决方案，既能有效减轻移动端的计算压力和减少验证时的数据通信量，又能保障安全性. 本文以云文件存储服务为应用场景，提出一种面向移动云计算环境的数据完整性验证方案. 该方案支持移动端用户对云端数据进行完整性验证和数据更新操作，支持将验证外包给可信第三方处理，并具有无需文件块直接参与验证的特点，适合在移动云计算环境中部署.

1　系统组成简介

本文研究的移动云计算数据完整性验证方案(mobile cloud computing data integrity，MCCDI)的原型系统由以下几部分组成：①数据拥有者(data owner，DO)，将数据提交给云服务供应商存储的用户；②云服务供应商(cloud service provider，CSP)，提供云存储服务的机构；③可信第三方(third－party auditor，TPA)，提供数据外包验证服务的机构；④存储服务供应商(storage service provider，SSP)，实际的数据存储服务提供者；系统的基本组成部分如图 1 所示.

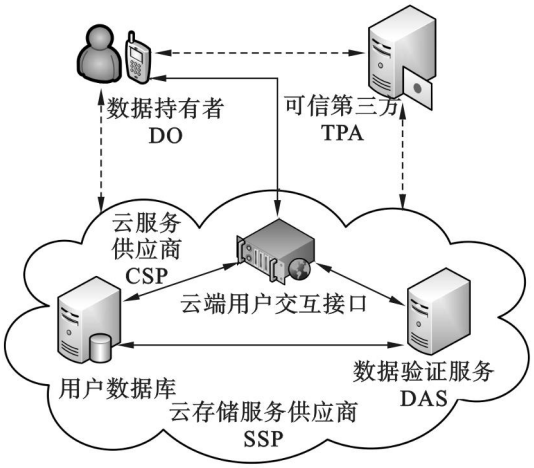

图 1　MCCDI 方案的系统组成
Fig. 1　System architecture of MCCDI scheme

MCCDI 方案中的 DO 是安装在用户移动端软件，可与 CSP 交互，向 CSP 发送请求，并接受从 CSP 返回的数据. CSP 由云端用户交互模块、数据验证服务(DAS)和用户数据库这 3 部分组成. 其中用户交互模块是 DAS 和 DO 之间通信的桥梁；DAS 负责执行具体的验证算法逻辑；用户数据库负责存储用户信息. TPA 是可信的第三方验证机构，用户可把数据完整性验证外包给 TPA，以减轻自身的计算和存储负担. SSP 需要提供具有高可靠性、低错误率和高响应速率的存储服务. 本文的算法描述中将把 CSP 和 SSP 看作同一个机构.

2　数学模型

2.1　算法的理论依据

本文研究的 MCCDI 方案以基于椭圆曲线双线性对的 BLS(Boneh－Lynn－Shacham)短签名方案[9]为理论依据. BLS 短签名方案基于椭圆曲线上双线性对映射：$e:G_1\times G_2\to G_T$，其中群 G_1 和 G_2 是两个 Gap Diffie－Hellman (GDH)群，G_T 是一个具有素数阶 p 的乘法群. 映射 e 满足下面几条性质：

1) 可计算性：计算双线性映射 e 存在有效算法.

2) 双线性：对于任意 $h_1,h_2\in G,a,b\in Z_p$(其中 Z_p 是小于 p 的非负整数的集合)，映射 e 满足：

$$e(h_1^a,h_2^b)=e(h_1^b,h_2^a)=e(h_1,h_2)^{ab}. \quad (1)$$

3) 非退化性：$e(h_1,h_2)\neq 1$，其中 g 是群 G 的一个生成元.

采用 BLS 方案时，假设用户 A 想对一条消息签名，他按照下面的方法生成自己的公钥和私钥：私钥 x 是集合 Z_r 的任意一元素，与之对应的公钥是 g^x. 为了对消息签名，A 必须将消息映射成群 G 上的某一元素 h，并生成消息签名 h^x. 用户 B 想要验证这条消息，则检查 $e(h,g^x)=e(\sigma,g)$ 是否成立，若满足则消息确认，反之，否认.

2.2　关键方法

本文的 MCCDI 方案依托基于 Merkle 哈希树(Merkle Hash tree，简称哈希树)[10]的文件完整性检验方法，将文件块标签的哈希值按从左至右的顺序(文件块的自然顺序)依次对应哈希树的叶子节点，再按顺序两两级联哈希，最终计算出根节点的哈希值. 如图 2 所示，A 节点的值 h_a 等于 $h(h(h(x_1)\parallel h(x_2))\parallel h(h(x_3)\parallel h(x_4)))$. 对于一个拥有 n 个块(n 为 2 的整数次幂)的文件，需要进行 $2n-1$ 次哈希计算(包括最初对每个文件块标签的哈希)才能算出根节点的值. 本方案中，哈希树的价值体现在对文件块完整性的验证上. 验证任意一文件块 m_i，只需知道该文件块的块标签所对应的叶子节点到哈希树根节点路径上的所有兄弟节点的值就可以了.

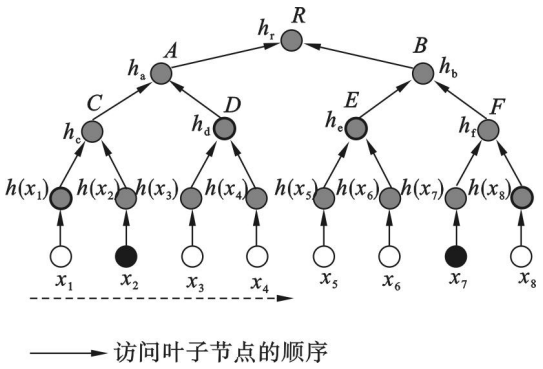

图 2 用于文件完整性验证的哈希树结构

Fig. 2 The architecture of hash tree used for verification of file data integrity

如图 2 所示,想要验证块标签 x_2 所对应的文件块是否完整,则需要知道块标签 x_2 对应的叶子节点 $h(x_2)$ 的值(计算 x_2 的哈希即可),以及 $h(x_2)$ 到根节点 Root 路径上的所有兄弟节点的值,即 $h(x_1)$,h_d 和 h_b 的值. 本文将这些兄弟节点称为 AAI(auxiliary authentication information)节点. 验证拥有 n 个块的文件中任意一块,只需要 lb$n$ 次哈希值计算即可. 这个值同时也是哈希树的层数(记叶子节点层高为 0,依次往上递增). 基于哈希树的文件完整性检验方法只需对少量文件块的检验就能以极高的概率检验出文件完整性[10]. 哈希树对于动态数据有很好的支持,只需进行一些二叉树操作和更新即可.

MCCDI 方案中的文件块标签不采用位置索引,而是直接以 $H(m_i)$ 表示(H 为哈希映射函数,m_i 是加密后的源文件块). 在验证阶段,DO(TPA)并不需要自己计算 $H(m_i)$,$H(m_i)$ 应由 CSP 计算并返回给 DO(TPA). MCCDI 方案可以避免如文献[7]中用户更新操作造成的受影响文件块需重新签名上传的情况,降低了开销,适合于移动云计算环境.

2.3 关键算法定义

KeyGen(l)→(pk,sk):密钥生成算法. 该算法由 DO 执行,输入秘密值 l,输出公钥 pk 和私钥 sk.

SigGen(sk,F)→(Φ,$\mathrm{sig}_{sk}(H(R))$):签名产生算法. 该算法由 DO 执行,输入 sk 和 F($F=\{m_i\}$,$1\leqslant i\leqslant n$,m_i 为经分块并用对称密钥加密后的源文件),生成文件块签名 $\Phi=\{\sigma_i\}$,$1\leqslant i\leqslant n$. 之后对 m_i 用 H 函数处理得到文件块标签,即计算 $H(m_i)$,$1\leqslant i\leqslant n$,再用密码学哈希函数(SHA-1)计算出 $h(H(m_i))$,$1\leqslant i\leqslant n$,并构造哈希树,最后产生根节点 R 的签名 $\mathrm{sig}_{sk}(R)$.

Gen Proof(F,Φ,chal)→(P):验证信息产生算法. 该算法由 CSP 执行,输入 F,Φ,挑战值 chal(此三者由 DO 上传所得),产生验证信息 P.

Verify Proof(pk,chal,P)→{TRUE,FALSE}:验证信息算法. 该算法由 DO 或 TPA 执行,输入 DO 的公钥 pk,挑战值 chal 和从 CSP 返回来的验证信息 P,输出是否通过验证.

ExecUpdate(F,Φ,update)→{F',Φ',P_{update}}:更新执行算法. 该算法由 CSP 执行,输入欲更新的原文件块、原文件块签名和从 DO 收到的 update 请求. 输出为更新之后的文件块 F',更新后的文件块签名 Φ' 和更新操作的验证信息 P_{update}. 该算法包括批量更新操作.

ExecInsert(F,Φ,insert)→{F',Φ',P_{insert}}:插入执行算法. 该算法由 CSP 执行,输入欲插入位置的原文件块、原文件块签名和从 DO 收到的 insert 请求. 输出为更新之后的文件块 F',更新后的文件块签名 Φ' 和插入操作的验证信息 P_{insert}.

ExecDelete(delete)→{P_{delete}}:删除执行算法. 该算法由 CSP 执行,输入从 DO 收到的 delete 请求(删除文件不需要上传文件块和块签名). 输出删除操作的验证信息 P_{delete}.

VerifyUpdate(pk,update,P_{update})→{(TRUE,$\mathrm{sig}_{sk}(R')$),FALSE}:更新、插入、删除操作的验证算法. 该算法由 DO 或 TPA 执行,输入 DO 公钥 pk,update(insert,delete)请求和从 CSP 返回的 P_{update}(P_{insert},P_{delete})验证信息,3 种操作的验证过程类似.

2.4 算法执行步骤

本方案中数据完整性的算法执行步骤主要分为 3 个部分:准备阶段、完整性验证以及数据更新,每个部分的关键算法执行步骤描述如下:

2.4.1 准备阶段

1)执行算法 KeyGen(l):DO 选择一个公钥密码体系密钥对(spk,ssk),选择一随机数 $\alpha\in Z_p$,计算 $v=g^{\alpha}$. 然后令 sk=(α,ssk),pk=(v,spk).

2)执行算法 SigGen(sk,F):DO 将源文件分块并用对称密钥加密生成 $F=\{m_i\}$,$1\leqslant i\leqslant n$. 之后 DO 随机选择 $u\in G$,令

$$t=h(\mathrm{name}\,\|\,n\,\|\,u)\,\|\,\mathrm{Sig}_{ssk}(h(\mathrm{name}\,\|\,n\,\|\,u)) \quad (2)$$

其中:name 代表文件名;n 代表文件的块数. 将 t 作为文件 F 的标签. 之后计算文件块签名 $\sigma_i=(H(m_i)\cdot u^{m_i})^{\alpha}$(将文件块 m_i 看作 $\in Z_p$ 的大整数),令 $\Phi=\{\sigma_i\}$,$1\leqslant i\leqslant n$. 随后根据 $h(H(m_i))$,$1\leqslant i\leqslant n$ 构造一棵哈希树. 构造哈希树时需要注意,由于哈希树初构建时是一棵满二叉树(后面

可能变成完全二叉树),其叶子节点数目必须是 2 的整数次幂,那么构建时往往需要填充叶子节点使之满足条件. 建树完成后,记其根节点为 R,用私钥 α 对 R 签名:$\mathrm{sig}_{\mathrm{sk}}(R)=(R)^{\alpha}$. DO 构造$\{F,t,\Phi,\mathrm{sig}_{\mathrm{sk}}(R)\}$,将其上传给 CSP,并删除本地的$\{F,\Phi,\mathrm{sig}_{\mathrm{sk}}(R)\}$.

2.4.2　完整性验证阶段

1)构造挑战信息 chal:DO 可在本地验证数据完整性,亦可将验证外包给 TPA 实现,以此降低 DO 的计算和存储负担. 下面将以 TPA 代理验证叙述. TPA 在向 CSP 发送验证挑战 chal 之前,先要求 CSP 将文件标签 t 发送过来,通过 DO 的公钥 pk 验证文件标签 t(用 spk 对 t 包含的签名验证即可). 若验证通过则说明文件标签完好,反之直接返回 FALSE. 文件标签验证通过后 TPA 着手构造 chal. TPA 随机选择有 c 个元素的集合$[1,n]$的子集 $I=\{s_1,\cdots,s_c\}$,假定 $s_1\leqslant\cdots\leqslant s_c$. 对于每个 $i\in I$,TPA 选择一个随机数 $v_i\in Z_p$,构造 chal$\{(i,v_i)\}_{s_1\leqslant i\leqslant s_c}$,将其发送给 CSP. 关于 chal 的意义:有 c 个元素的集合$\{i\}$即表示想要挑战的 c 个文件块的位置索引,随机数集合$\{v_i\}$供 CSP 生成验证信息.

2)执行算法 Gen Proof(F,Φ,chal):在收到 chal$\{(i,v_i)\}_{s_1\leqslant i\leqslant s_c}$后,CSP 计算下面两个值:

$$\sigma=\prod_{i=s_1}^{s_c}\sigma_i^{v_i}\in G,\mu=\sum_{i=s_1}^{s_c}v_im_i\in Z_p. \quad (3)$$

其中:m_i 是文件的第 i 个文件块(同样,看做$\in Z_p$的大整数);σ_i 是第 i 个文件块对应的签名. 除此之外,CSP 还需提供$\{\Omega_i\}_{s_1\leqslant i\leqslant s_c}$,其为 2.2 节所述的 AAI 节点,供 TPA 重构哈希树的根节点 R. 之后 CSP 返回 $P=\{\mu,\sigma,\{H(m_i),\Omega_i\}_{s_1\leqslant i\leqslant s_c},\mathrm{sig}_{\mathrm{sk}}(R)\}$给 TPA 供其验证.

3)执行算法 Verify Proof$(\mathrm{pk},\mathrm{chal},P)$:在收到 CSP 返回的验证信息 P 后,TPA 利用$\{H(m_i),\Omega_i\}_{s_1\leqslant i\leqslant s_c}$重构哈希树根节点 R. 并进行下面两个验证:

验证 1:$e(\mathrm{sig}_{\mathrm{sk}}(R),g)=e(R,g^{\alpha})$.

证明　$e(\mathrm{sig}_{\mathrm{sk}}(R),g)=e((R)^{\alpha},g)=e(R,g^{\alpha})$.

若该验证失败,则返回 FALSE. 若验证通过则说明在 CSP 给定根节点签名 $\mathrm{sig}_{\mathrm{sk}}(R)$ 的前提下,根节点 R 与签名 $\mathrm{sig}_{\mathrm{sk}}(R)$是匹配的,还可以说明文件块标签$\{H(m_i)\}_{s_1\leqslant i\leqslant s_c}$是完好的(因为根据它可以重构出匹配签名的根节点). 继续进行下一个验证.

验证 2:$e(\sigma,g)=e(\prod_{i=s_1}^{s_c}H(m_i)^{v_i}\cdot u^{\mu},v)$,

证明

$$e(\sigma,g)=e(\prod_{i=s_1}^{s_c}\sigma_i^{v_i},g)=e(\prod_{i=s_1}^{s_c}(H(m_i)\cdot u^{m_i})^{\alpha v_i},g)=e(\prod_{i=s_1}^{s_c}(H(m_i)^{v_i}\cdot u^{m_iv_i})^{\alpha},g)=$$

$$e(\prod_{i=s_1}^{s_c}H(m_i)^{v_i}\cdot u^{\sum_{i=s_1}^{s_c}m_iv_i},g^{\alpha})=e(\prod_{i=s_1}^{s_c}H(m_i)^{v_i}\cdot u^{\mu},v).$$

若该验证失败说明文件块损坏,返回 FALSE. 反之,说明文件块完整,验证通过. 一旦通过了以上两个验证,则算法返回 TRUE,验证成功. 文件数据完整性验证的关键步骤如图 3 所示.

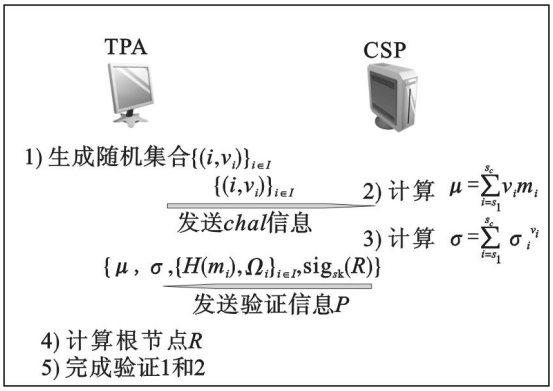

图 3　文件数据的完整性验证步骤

Fig. 3　Steps of file data integrity verification

2.4.3　数据更新阶段

数据更新操作关键步骤和数据流如图 4 所示.

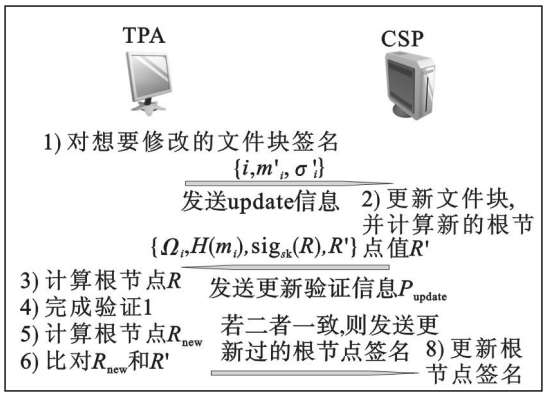

图 4　文件数据的更新操作步骤

Fig. 4　Steps of updating for file data

1)构造更新信息. 假设 DO 想将第 i 块数据 m_i 更新为 m'_i. DO 为 m'_i 产生签名 $\sigma_i=(H(m'_i)\cdot u^{m'_i})^{\alpha}$. 然后构造更新信息 update $=(i,m'_i,\sigma'_i)$发送给 CSP.

2)执行算法 ExecUpdate(F,Φ,update). CSP 替换块 m'_i 和块签名 σ'_i,更新哈希树(此时 CSP

可缓存某些哈希树节点以提高构造效率),生成新的根节点 R'. 随后 CSP 产生更新验证信息 $P_{\text{update}}=(\Omega_i, H(m_i), \text{sig}_{\text{sk}}(R), R')$,发送给 TPA. TPA 利用 $\{\Omega_i, H(m_i)\}$ 重构根节点 R,并验证 $e(\text{sig}_{\text{sk}}(R), g)=e(R, g^{\alpha})$,即 2.4.2 节所述验证 1. 若验证通过,DO 继续检查 CSP 是否真的完成了更新数据操作. DO 利用 $\{\Omega_i, H(m'_i)\}$ 构造更新后的根节点值(若在 DO 本地验证,DO 可根据 m'_i 直接算出 $H(m'_i)$;若为 TPA 代理验证,则由 DO 算出 $H(m'_i)$ 发送给 TPA),并将该值同 R' 比对,若一致,则说明 CSP 完成了数据更新工作,已算出新的根节点值. 这时由 DO 用自己的私钥 sk 对新的根节点 R' 签名得到 $\text{sig}_{\text{sk}}(R')$,将其发送给 CSP 更新.

3 方案的安全性讨论

MCCDI 方案的安全性建立在无有效算法能够解决椭圆曲线上的离散对数问题及 GDH(Gap Diffie – Hellman)群的特性上. 该方案中,如果 CSP 不把陈旧信息删除,反而在执行 Gen Proof 操作时将"陈旧"的 $\{H(m_i), \Omega_i\}_{s_1\leqslant i\leqslant s_c}$ 和 $\text{sig}_{\text{sk}}(R)$ 值返回给 DO(TPA)的话,在完整性验证阶段 Verify Proof 的验证 1 中比较重构出的根节点 R 与存放在 TPA 处的 R' 是否一致,就可判定 CSP 是否返回了陈旧的验证信息. 该方案中的 DO 在 SigGen 阶段生成文件块签名时并没有直接套用 BLS 方案计算 $\sigma_i=H(m_i)^{\alpha}$ 而是将文件块 m_i 带入计算 $(\sigma_i=(H(m_i)\cdot u^{m_i})^{\alpha})$ 是基于以下考量:CSP 可能已经把用户的文件数据删除,但是却保留了文件块标签 $H(m_i)$(块标签比文件块小得多). 由于 DO(TPA)在验证过程中不接触任何文件块 $\{m_i\}_{s_1\leqslant i\leqslant s_c}$(而是块标签),对于 DO(TPA)来说 CSP 所做出的举动它们一无所知. 因为块标签完好,验证操作仍然可以顺利进行,但实际上文件块可能已经被 CSP 删除了. 所以,在生成文件块签名时必须要将文件块代入计算,这样一来 CSP 在 Gen Proof 过程中也必须将文件块 $\{m_i\}_{s_1\leqslant i\leqslant s_c}$ 带入计算 $\left(\mu=\sum_{i=s_1}^{s_c} v_i m_i\in Z_p^*\right)$,CSP 也就不能随便删除用户的文件数据了. 另外,由于 MCCDI 支持外包验证,而 DO 并不对 TPA 返回的验证结果做任何处理而是直接采纳,即便 TPA 与 CSP"共谋",用户的文件数据和私钥也不会暴露给 TPA. 因此,在 VerifyProof 阶段的两个验证和一个充分可信赖的 TPA 的保证下,MCCDI 方案是安全的.

4 结 论

1)支持验证过程外包,可减轻移动端的计算和存储负担.

2)验证过程无状态信息保存,CSP 只需根据 DO(TPA)发来的挑战请求 chal 构造相应的验证数据返回给 DO(TPA)即可.

3)验证无需源文件块参与,DO(TPA)端在整个验证流程中操作的是经映射函数 H 处理过的文件块,可保证文件信息不暴露给 TPA.

参考文献:

[1] Mell P, Grance T. The NIST cloud computing definition [M]. New York: NIST Special Publication, 2011.

[2] Yang J, Wang H, Wang J, et al. Provable data possession of resource constrained mobile devices in cloud computing[J]. *Journal of Networks*, 2011, 6(7): 1033 – 1040.

[3] 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.)

[4] Khan A N, Mat-Kiah M L, Khan S U, et al. Towards secure mobile cloud computing: a survey[J]. *Future Generation Computer Systems*, 2013, 29(5): 1278 – 1299.

[5] Wang Q, Wang C, Li J, et al. Enabling public verifiability and data dynamics for storage security in cloud computing [C]//Proceedings of the 14th European Conference on Research in Computer Security. Heidelberg: Springer, 2009: 355 – 370.

[6] Filho D L G, Baretto P S L M. Demonstrating data possession and uncheatable data transfer[EB/OL]. (2006 – 01 – 15)[2014-03-20], http://eprint.iacr.org/2006/150

[7] Juels A, Burton J, Kaliski S. Pors: proofs of retrievability for large files[C]//Proceedings of the 14th ACM Conference on Computer and Communications Security. Alexandria, 2007: 584 – 597.

[8] Ateniese G. Provable data possession at untrusted stores [C]//Proceedings of the 14th ACM Conference on Computer and Communications Security. Alexandria, 2007: 598 – 609.

[9] Boneh D, Lynn B, Shacham H. Short signatures from the weil pairing[C]//Proceedings of ASIACRYPT'01. London: Springer, 2001: 514 – 532.

[10] Merkle R. Secrecy, authentication, and public key systems [D]. Stanford: UMI Research, 1982.