一种基于 H-MHT 的动态数据完整性检查方案*

李 莹[1,2], 张永胜[1,2], 马 洁[1,2]
(1. 山东师范大学 信息科学与工程学院, 济南 250014; 2. 山东省分布式计算机软件新技术重点实验室, 济南 250014)

摘 要: 云存储可以为用户提供高质量的、按需分配的数据存储服务,能够使用户以较小的代价来解决本地软/硬件资源不足、移动不便、存储设备毁坏或丢失等问题。但对于用户而言,云存储并不完全可信,他们往往担心存储在云端的数据的完整性问题。为此,提出一种基于 H-MHT 的动态数据完整性检查方案。在 RDPC(remote data possession checking)协议的基础上,引入 H-MHT 认证数据结构,将文件的使用频率作为参考因素添加到检查方案中,构建一棵类似哈夫曼树的二叉树,并通过动态调整树结构来支持数据的动态操作。模拟实验结果表明,该方案在检查的准确性和检查效率方面具有明显的优势,能够很好地支持云存储环境下动态数据的完整性检查。

关键词: 同态哈希函数; H-MHT; 动态 RDPC 协议; 数据完整性检查
中图分类号: TP309.2 **文献标志码**: A **文章编号**: 1001-3695(2015)12-3710-04
doi:10.3969/j.issn.1001-3695.2015.12.043

Dynamic data integrity checking scheme based on H-MHT

Li Ying[1,2], Zhang Yongsheng[1,2], Ma Jie[1,2]
(1. *School of Information Science & Engineering, Shandong Normal University, Jinan* 250014, *China*; 2. *Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology, Jinan* 250014, *China*)

Abstract: Cloud storage can provide users with high-quality and on-demand data storage services and free them from the burden of maintenance and management. However, the cloud servers are not fully trusted for users. Whether the data stored on cloud are complete or not has become a major concern of users. Therefore, how to verify the integrity of remote data efficiently has become a major problem to be solved. For this issue, this paper proposed a dynamic remote data integrity checking scheme based on H-MHT. In order to support dynamic operations better, the scheme adopted the H-MHT authentication data structure and took the frequency of file as a reference factor. Simulation experimental results show that the scheme has a distinct advantage in accuracy and efficiency and demonstrates the superiority of dynamic data integrity checking under the cloud storage environment.
Key words: homomorphic hash function; H-MHT; dynamic RDPC protocol; data integrity checking

0 引言

云存储是一个以数据存储和管理为核心的云计算系统,能够为用户提供动态可伸缩的数据存储服务,其最大的特点是存储即服务(storage as a service,SaaS),即用户可以通过服务的方式,按需计量随时随地地使用云存储中丰富的存储资源[1],并且能以较小的代价来解决本地软/硬件资源不足、移动不便、存储设备毁坏或丢失等问题。因此,越来越多的企业和个体开始使用云存储服务。然而使用云存储的服务意味着用户要将数据存储到他们掌控不到的存储设备中去,即数据处于用户的不可控之中,这使得数据的安全问题成为了云存储推广的重大障碍之一。从 Twinstrata 公司 2012 年的调查可以看出,只有 20% 的人愿意使用云存储服务来存储自己的私有数据,大约有 50% 的人愿意使用云存储服务来进行数据备份及灾难恢复等作业[2]。目前正在使用云存储服务的用户主要担心其数据的完整性问题,因此,要想使用户放心地使用云存储服务,云存储服务器需要向用户提供一系列证据来证明他们的数据被正确地持有。一些传统的检查数据是否安全的方法和技术,比如数据的完整性验证(如数字签名)需要用户从云端下载全部的数据后再与本地数据进行比较,显然,这种检查方式的效率比较低下。而在分布式的云存储环境下,由于大规模数据所导致的巨大通信代价,用户不可能将数据全部下载完成后再验证其完整性。因此,如何高效地对远程数据的完整性进行检查是一个亟待解决的问题。

对于云存储中数据可持有性检查问题,通常的做法是用户取回存储在云端的少量数据,通过某种知识证明协议或概率分析手段,以高置信概率判断存储在云端的数据是否完整[3]。Atenies 等人[4]利用同态标签提出了可证明数据持有(provable data possession,PDP)模型,但该模型不支持公开性验证和数据更新;随后他们改进了该模型,把原来的同态标签改为支持公开校验的标签,把 PDP 方案扩展到支持可公开校验方案[5]。同时,Juels 等人提出了可取回性证明 (proof of retrievability,

收稿日期: 2014-11-18; **修回日期**: 2015-01-08 **基金项目**: 山东省自然科学基金资助项目(ZR2011FM019);山东省研究生教育创新计划资助项目(SDYY11117)

作者简介:李莹(1990-),女,山东菏泽人,硕士研究生,主要研究方向为云计算安全、云存储环境下的数据安全(641367138@qq.com);张永胜(1962-),男,山东潍坊人,教授,硕导,主要研究方向为软件工程环境、Internet /Intranet 工程、网络信息安全;马洁(1990-),女,山东济南人,硕士研究生,主要研究方向为云计算安全、可信云计算安全.

POR)的概念,即云服务器提供一个证明使得用户相信他们的数据是可以取回的,并引入了“哨兵”和纠错编码机制,但该方案只允许有限次的验证。POR 和 PDP 方案提出以后,对于云存储数据完整性检查工作就侧重在这两个方面。在文献[6]中利用同态标签和范德蒙矩阵对文件进行加密处理,并加入 RS 编码,有效地解决了验证次数有限和错误定位的问题,但并不能很好地支持数据动态化。随后,一些人开始将研究重点集中在支持数据动态操作方面。例如,在文献[7,8]中提出了当文件发生动态变化时进行验证的方法,即采用 MHT(Merkel hash tree)的叶子节点来保存文件的相关信息,当文件变化时,对 MHT 的节点进行调整,从而保持文件发生变化后的同步信息。但该方法仅考虑了文件变化的情况,并没有考虑文件太多的实际因素(如文件的使用频率)。文献[9]提出了一个动态的 RDPC(remote data possession checking)协议,能够较好地支持数据动态操作,但容易受到不可信服务器的攻击;后来文献[10]针对文献[9]的不足改进了此协议,并进行了安全证明。

本文将文献[10]所提出的改进动态 RDPC(remote data procession checking)协议与 H-MHT(H-Merkel hash tree)认证数据结构相结合,将文件的使用频率考虑在内,当文件发生变化时,对 H-MHT 的节点进行调整以保持文件发生变化后的同步信息。经过相关理论知识和实验证明,该方案能够实现高概率、高效率的数据完整性检查。

1　相关知识

1.1　同态哈希函数

同态哈希函数包含密钥生成和哈希生成两个阶段。

a)密钥生成阶段

利用 KeyGen 算法[11],生成密钥 $k(p,q,t)$。其中:t 是在 Z_p 上的 q 阶 $1\times m$ 列向量,且 $t=\{t_0,t_1,\cdots,t_m\}$;p 和 q 是两个随机的大素数。

b)哈希生成阶段

文件 F 被分成 n 块,即 $F=F_1\parallel F_2\parallel\cdots\parallel F_n$,每个块又被分成 m 个基本块,即 $F_i=F_{i1}\parallel F_{i2}\parallel\cdots\parallel F_{im}$,文件 F 的哈希值 $HK(F_i)=(H_K(F_1),H_K(F_2),\cdots,H_K(F_n))$。由式(1)计算出 $H_K(F_i)$ 为

$$H_K(F_i)=\prod_{j=1}^{m}t_j^{F_{ij}}\bmod p \tag{1}$$

1.2　MHT 与 H-MHT

认证数据结构是一种将查询数据结构与相应密码学技术相结合,使其中的数据可认证化的一种结构[12]。MHT 是一种认证数据结构,能高效、可靠地证明一组数据是否完好无损。它是一棵二叉树,要求认证的数据的哈希值全部都存储在叶子节点,具有同态性和免碰撞性[13]。

H-MHT 是在 MHT 的基础上结合哈夫曼树的性质提出的一种认证结构[14]。它与 MHT 相同的是:都是一棵二叉树,并且二叉树的叶子节点存储的是每个文件的哈希值;区别在于:H-MHT 的叶子节点比 MHT 的叶子节点多了一个权值(权值即文件的使用频率,用 W 来表示,W 值越大,使用频率越高),并且深度小的叶子节点的权值必须不小于比它深度大的叶子节点的权值。由于不同的文件使用频率不同,可以考虑将使用频率较高的文件放在离根节点相对较近的叶子节点上,将使用频率低的文件放在离根节点相对较远的叶子节点上,其构造过程与哈夫曼树的构造过程相似。图 1 为 MHT 与 H-MHT 的认证结构对比(左侧为 MHT,右侧为 H-MHT),其中 $H_K(X_i)$ 表示文件 X_i 的哈希值,W_i 表示文件 X_i 的权值。

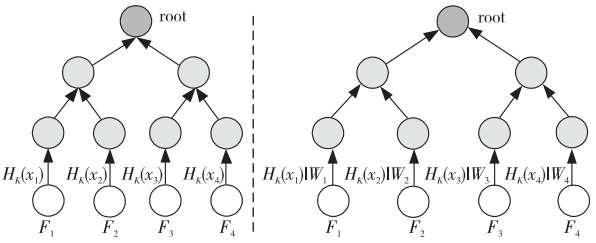

图 1　MHT 与 H-MHT 结构

2　云存储数据完整性检查方案

本文提出的方案基于文献[10]中提出的改进动态 RDPC 协议,采用了传统的两方模型:

a)云用户(CU)

云用户将自己的数据存储至云服务器,可以随时地对他们的数据进行添加、删除或修改操作;还可以随时向 CS 发送数据完整性检查请求。

b)云服务器(CS)

负责管理用户数据,为用户提供数据存储服务和计算服务;接受 CU 的挑战,并返回相应的证据。

首先 CU 在云平台注册,请求并生成签名公私钥对(sk,pk);再对文件 F 进行预处理:将文件 F 分成 n 块,即 $F=F_1\parallel F_2\parallel\cdots\parallel F_n$,又将每块分成 m 个基本块,即 $F_i=F_{i1}\parallel F_{i2}\parallel\cdots\parallel F_{im}$;然后在本地生成文件 F 的各项信息,包括文件名的标签信息、构造的 H-MHT 信息、每个数据块的标签信息及其聚合标签信息等,再把文件 F 的各项信息发送至 CS;最后 CU 与 CS 之间以挑战—应答的方式进行数据完整性检查。完整性检查过程包含以下五个阶段:

a)初始化阶段(setup)

主要生成各种密钥参数。CU 在云平台注册成功后,根据 KeyGen 算法生成一对私钥和公钥(sk,pk)作为签名机制 Sig(·)的私钥和公钥,并选择 Ken 作为对称加密 Enc(·)的密钥。其中用到以下三个密码学原语:

$$h(\cdot):\{0,1\}^*\rightarrow\{0,1\}^{\lambda q}$$
$$f(\cdot):Z_q^*\times Z_q^*\rightarrow Z_q^*$$
$$g(\cdot):Z_q^*\times\{1,\cdots,n\}\rightarrow\{1,\cdots,n\}$$

其中:$h(\cdot)$是同态哈希函数;$f(\cdot)$是一个伪随机函数;$g(\cdot)$是一个伪随机置换。

b)标签生成阶段(sigGen)

选择 $m+1$ 个随机值 $\alpha=\{\alpha_0,\alpha_1,\cdots,\alpha_m\}$,生成文件 F 的标签 $\tau=F_{id}\parallel n\parallel m\parallel\mathrm{Enc}_{\mathrm{ken}}(\alpha_0\parallel\alpha_1\parallel\cdots\parallel\alpha_m)$。再利用式(2)计算每个数据块 F_i 的标签 T_i,根据 $h(F_{id}\parallel F_i)$ 以及每个文件块赋予的初始权值 W_i 生成一棵 H-MHT 树(生成树的信息用 Struct 来表示),并利用自己的私钥 sk 对树的根节点 $h(R)$ 签名 $\sigma=\mathrm{Sig}_{\mathrm{sk}}(h(R))$。最后,CU 将$\{\tau,F,T=\{T_1,T_2,\cdots,T_n\},W=\{W_1,W_2,\cdots,W_n\},\mathrm{Struct},\sigma\}$发送给 CS。

$$T_i=\alpha_0^{h(F_{id}\parallel F_i)}\cdot\prod_{j=1}^{n}\alpha_j^{F_{ij}}\bmod p \tag{2}$$

c)挑战阶段(challenge)

CU 随机选取两个参数 $k_1,k_2\in Z_p^*$ 和挑战块数 $c\in Z_n$,向

CS 发起挑战 chal = (c,k_1,k_2)。

d)证据生成阶段(proofGen)

CS 接收挑战 chal = (c,k_1,k_2)后,首先计算 $r_i = g_{k_1}(i)$, $\lambda_i = f_{k_2}(i)$,其中,$1 \leqslant i \leqslant c$;接着计算 $T = \prod_{i=r_1}^{r_c} T_i^{\lambda_i} \bmod p$, $\beta_j = \sum_{i=r_1}^{r_c} \lambda_i F_{ij} \bmod q$,其中,$1 \leqslant j \leqslant m$;最后返回 CU 一个证据 $P = \{\tau,\beta = \{\beta_1,\cdots,\beta_m\},T,\{h(F_{id} \parallel F_i),M_i\},\sigma\}$,其中,$\{M_i\}$是从叶子节点 $h(F_{id} \parallel F_i)$到根节点 R 这条路径上的所有节点的兄弟节点的哈希值,并且 $r_1 \leqslant i \leqslant r_c$。

e)验证阶段(proofVerify)

CU 接收证据 P 后,首先根据$\{h(F_{id} \parallel F_i),M_i\}$重新计算根节点的哈希值 R',并判断式(3)是否成立。若不成立,则输出 false,拒绝接收证据;否则,计算 $r_i = g_{k_1}(i)$和 $\lambda_i = f_{k_2}(i)$,其中,$1 \leqslant i \leqslant c$。接着利用密钥 k_{en}解密出 $\alpha_0 \parallel \alpha_1 \parallel \cdots \parallel \alpha_m = \mathrm{Dec}_{k_{en}}(\mathrm{Enc}_{k_{en}}(\alpha_0 \parallel \alpha_1 \parallel \cdots \parallel \alpha_m))$。最后判断等式(4)是否成立。若成立,则输出 true,验证成功;否则,输出 false,验证失败。

$$\mathrm{Ver}_{pk}(\sigma) = h(R') \tag{3}$$

$$T = \alpha_0^{\sum_{i=r_1}^{r_c} \lambda_i h(F_{id} \parallel F_i)} \cdot \prod_{j=1}^{m} \alpha_j^{\beta_j} (\bmod p) \tag{4}$$

3 动态数据操作

用户使用云存储服务就可以随时对他们的数据进行各种操作,主要包括插入、删除和修改。当用户对文件进行操作(即对 H-MHT 树的叶子节点进行操作)时,改变的是它们的使用频率(即权值),这就需要根据叶子节点的权值对 H-MHT 的结构进行调整。原则上是在保证效率的情况下,又尽量使树的特性比较接近哈夫曼树的特性。假设文件 F 和其相应的标签等信息$\{\tau,T,\mathrm{Struct},\sigma\}$已经存储在云服务器上。用户先在本地计算操作后的数据的各项信息;然后与操作指令一块发送到云服务器,云服务器根据发送过来的信息调整 H-MHT,并重新计算根节点的哈希值;最后执行步骤 d)和 e),从而实现动态数据的完整性检查。

3.1 数据的插入

数据块的插入操作是指在文件中的某些特定位置上插入新的数据块。假设 CU 想在第 i 个数据块的后面插入一个新的数据块。调整方法是:在 F_i 节点下方插入两个子节点,让这两个节点分别代表节点 F_i 和新插入的 F_i^* 节点,而原来的节点的权值则删除,哈希值变为两个子节点的哈希值的哈希,同时将节点的所有父节点都重新计算哈希值,如图 2 所示。

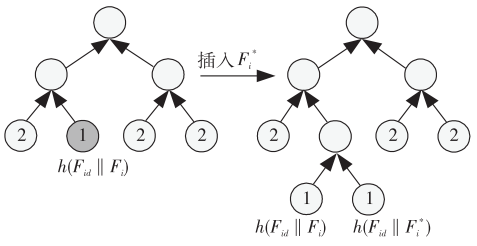

图 2 数据块的插入

3.2 数据的删除

数据块的删除操作是指删除某些特定位置上的数据块。假设 CU 想删除第 i 个数据块。调整方法是:需要找到 F_i 节点,用它的父节点的另一棵子树(根是灰色节点的子树)来代替它的父节点,然后再计算灰色节点的所有父节点重新计算哈希值,如图 3 所示。

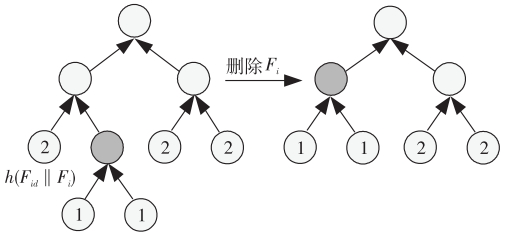

图 3 数据块的删除

3.3 数据的修改

数据块的修改操作是指用新的数据块来代替指定的数据块,而修改后叶子节点的权值要增加 1。假设 CU 想把第 i 个数据块 F_i 修改成 F_i^*。调整方法是:先将节点的权值增加 1,与深度比它小的叶子节点的权值相比较。若发现灰色节点的权值比它小,则与灰色节点交换并重新计算这两个节点的父节点的哈希值,如图 4 所示。

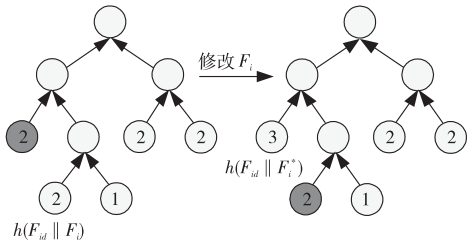

图 4 数据块的修改

4 性能分析

4.1 检测概率分析

从概率上来分析本方案的检测准确性。假设用户每次挑战 n 块当中的 c 块数据,并且存在一个恶意的云服务器窜改了任意 k 块数据。设随机变量 X 表示检测到数据有损坏的块数,则 $X = \{0,1,2,\cdots,k\}$。可以得出:$P\{X \geqslant 1\} = 1 - P\{X = 0\} = 1 - \frac{n-k}{n} \times \frac{n-k-1}{n-1} \times \cdots \times \frac{n-k-c+1}{n-c+1}$。

令 c 为自变量 x,则 $f(x) = \frac{n-k-x}{n-x}$,其中 $x \in [0,c-1]$,不难证明 $f(x)$在$[0,c-1]$上是单调递减的,故 $f(x) \geqslant f(x+1)$,可以推出:$f(0) \geqslant f(1) \geqslant \cdots \geqslant f(c-1)$,即$[f(c-1)]^c \leqslant f(0) \cdot f(1) \cdots f(c-1) \leqslant [f(0)]^c$,即 $(\frac{n-k-c+1}{n-c+1})^c \leqslant p\{X = 0\} \leqslant (\frac{n-k}{n})^c$。所以,$1 - (\frac{n-k}{n})^c \leqslant p\{X \geqslant 1\} \leqslant 1 - (\frac{n-k-c+1}{n-c+1})^c$,那么,完成一次检查能检测出有错的概率至少是 $1 - (\frac{n-k}{n})^c$。

令 c 为自变量 x,设 $p(x) = 1 - (\frac{n-k}{n})^x$,其中 $x \geqslant 0$。当数据块损坏的块数 k 分别等于 $0.005n$、$0.01n$、$0.02n$、$0.04n$、$0.08n$ 时,$p(x)$分别等于 $1 - 0.995^x$、$1 - 0.99^x$、$1 - 0.98^x$、$1 - 0.96^x$、$1 - 0.92^x$。由此利用 MATLAB 绘制出随机挑战的数据块数 x 与成功检测出数据损坏的概率 $p(x)$之间的变化关系,如图 5 所示。

4.2 效率分析

将文件 F 划分成 n 块,即 $F = F_1 \parallel F_2 \parallel \cdots \parallel F_n$,每个文件块使用的频率分别为 f_1, f_2, …, f_n,分别构造一棵 MHT 和 H-MHT。MHT 的叶子节点深度为$\lfloor \log_2^n \rfloor$,将 H-MHT 的叶子节点

深度分别设置为 $d_1,d_2,\cdots,d_n$。假设更新一个叶子节点所需时间为 t,则平均每次更新 MHT 和 H-MHT 所需时间分别为 $\lfloor \log_2^n \rfloor t$ 和 $(\frac{f_1d_1+f_2d_2+\cdots+f_nd_n}{n})t$。不难看出,当文件的更新频率方差越大,$\frac{f_1d_1+f_2d_2+\cdots+f_nd_n}{n}$ 值就越小,H-MHT 性能就越好。

在 Windows 7 环境下,安装 MyEclipse 10.0 以及 MySQL 数据库,利用 Java 编程。取 16 个文件块,权值 $W=\{32,37,38,40,63,68,52,55,57,27,31,43,45,21,25,27\}$ 进行模拟实验。假设更新次数为 100 次,记录每次更新 MHT 和 H-MHT 所需的平均时间,利用 MATLAB 绘制出关系图像,如图 6 所示。

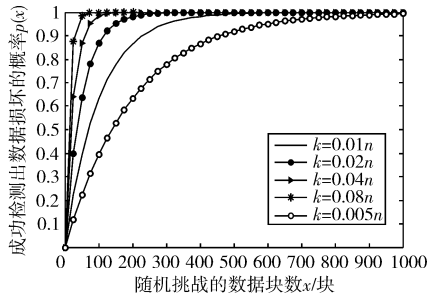

图 5　检测准确性概率分析

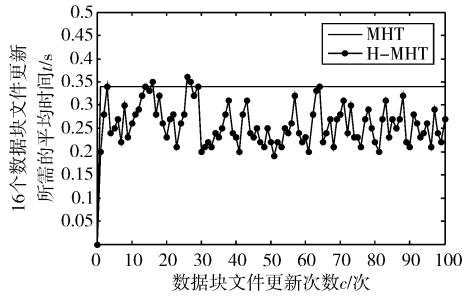

图 6　H-MHT 与 MHT 性能对比

由图 5 可以看出,在随机挑战块数相同的情况下,数据损坏的块数越多,成功检测出数据损坏的概率越大,这表明本方案对数据损坏的敏感性比较好,从而在一定程度上反映出检测的准确性较高。由图 6 可以看出,随着数据块文件更新次数的增多,更新 H-MHT 所需的平均时间比更新 MHT 所需的平均时间少,这表明采用 H-MHT 认证数据结构具有更高的检测效率。因此,本方案在性能方面具有明显的优势。

5　结束语

目前,越来越多的企业和个人倾向于将他们的数据外包到远程的云服务器中,这给云存储的普及带来了可观的发展前景,而数据的不可控性以及云服务器的不完全可信使得云存储环境下数据的安全问题日益突出。针对数据的完整性检查问题,本文提出了一种基于 H-MHT 的动态数据完整性检查方案,并详细地介绍了完整性检查方案的过程和动态化数据操作的过程。该方案把文件的使用频率考虑在内,构建一棵类似哈夫曼树的二叉树,使得路径长度最短,从而节省了节点查找时间,并且通过动态调整 H-MHT 的结构来实现数据的动态操作。性能分析表明该方案能够实现高概率、高效率的完整性检查,因此,本文提出的方案具有明显的优越性。

参考文献:

[1] 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.

[2] 胡德敏,余星. 一种基于同态标签的动态云存储数据完整性验证方法[J]. 计算机应用研究,2014,31(5):1362-1365,1395.

[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.

[4] Atenies G, Burns R, Curtmola R, *et al*. Provable data possession at untrusted stores[C]//Proc of the 14th ACM Conference on Computer and Communications Security. New York: ACM Press, 2007:598-609.

[5] Atenies G, Kamara S, Katz J. Proofs of storage from homomorphic identification protocols[C]// Proc of the 15th International Conference on the Theory and Application of Cryptology and Information Security. 2009: 319-333.

[6] WANG Cong, WANG Qian, REN Kui, *et al*. Ensuring data storage security in cloud computing [C]//Proc of the 17th International Workshop Quality of Service. [S.l.]: IEEE Press, 2009:1-9.

[7] Yang Kan, Jia Xiaohua. An efficient and secure dynamic auditing protocol for data storage in cloud computing [J]. IEEE Trans on Parallel and Distributed Systems, 2013, 24(9): 1717-1726.

[8] Wang Qian, Wang Cong, Li Jin, *et al*. Enabling public verifiability and data dynamics for storage security in cloud computing [J]. IEEE Trans on Parallel and Distributed Systems, 2011, 22(5):355-370.

[9] Chen Lanxiang, Zhou Shuming, Huang Xinyi, *et al*. Data dynamics for remote data possession checking in cloud storage[J]. Computers and Electrical Engineering, 2013, 39(7): 2413-2424.

[10] Yu Yong, Ni Jianbin, Au M H, *et al*. Improved security of a dynamic remote data possession checking protocol for cloud storage [J]. Expert Systems with Applications, 2014, 41(17): 7789-7796.

[11] 曹夕,许力,陈兰香. 云存储系统中数据完整性验证协议[J]. 计算机应用,2013,32(1):8-12.

[12] 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.

[13] 周锐,王晓明. 基于同态哈希函数的云数据完整性验证算法[J]. 计算机工程,2014,40(6):64-69.

[14] 李刚锐. 云存储数据验证算法的相关研究[D]. 杭州:浙江大学, 2013.

(上接第 3705 页)

[6] Zhang Weiming, Ma Kede, Yu Nenghai. Reversibility improved data hiding in encrypted images[J]. Signal Processing, 2014,94(1):118-127.

[7] 肖迪,杜社,郑洪英. 基于差值域直方图平移的密文可逆水印算法[J]. 计算机应用研究, 2014,31(12):3668-3672.

[8] 陈宁,马会杰. 基于 Contourlet 和 SVD 的鲁棒双水印算法[J]. 计算机应用研究,2012,29(7):2700-2702.

[9] 曾文权,熊祥光,余爱民. 基于奇异值分解的彩色图像水印算法[J]. 计算机应用研究,2013,30(10):3114-3116,3120.

[10] 方旺盛,张蓉. 基于 DWT-SVD 域的彩色图像自适应水印算法[J]. 计算机应用研究,2012,29(11):4323-4326.

[11] 蔡宜嘉,牛玉刚,苏庆堂. 基于 DWT-SVD 和 Fibonacci 变换的彩色图像盲水印算法[J]. 计算机应用研究,2012,29(8):3025-3028.

[12] 赵丽红,王永军,王佳禾. 双正交提升小波和奇异值分解的彩色水印算法研究[J]. 计算机应用研究,2014,31(2):568-570,575.

[13] 朱光,张军亮. 基于 SVD 和小波包分解的自适应鲁棒水印算法[J]. 计算机应用研究,2013,30(4):1230-1233.

[14] 王郁雨,杨晓梅,胡学妹. 基于奇异值分解的压缩感知核磁共振图像重构算法[J]. 计算机应用研究,2013,30(4):1247-1249, 1252.

[15] 刘大瑾. 联合非下采样 Contourlet 变换与奇异值分解的多水印算法[J]. 计算机应用研究,2013,30(12):3850-3853.

[16] Image database [EB/OL]. (2014). http://sipi.usc.edu/database/.

[17] Xiao Di, Liao Xiaofeng, Wei Pengcheng. Analysis and improvement of a chaos based image encryption algorithm[J]. Chaos Solitons Fractals, 2009,40(5):2191-2199.