doi:10.11959/j.issn.1000-436x.2015175

基于确定/概率性文件拥有证明的机密数据安全去重方案

陈越[1,2]，李超零[3]，兰巨龙[1,4]，金开春[1]，王仲辉[3]

(1. 解放军信息工程大学 网络空间安全学院，河南 郑州 450001；2. 国家数学工程与先进计算重点实验室，河南 郑州 450001；
3. 解放军 78179 部队，四川 都江堰 611800；4. 国家数字交换系统工程技术研究中心，河南 郑州 450002)

摘 要：为解决云存储系统中机密数据去重面临的密文重复性检测与拥有性证明、针对数据机密性的攻击等难题，提出了基于 Merkle 散列树的 MHT-Dedup 方案和基于同态 MAC 的 hMAC-Dedup 方案。两者均通过对密文文件的拥有证明进行跨用户文件级重复性检测，并通过检查数据块明文的摘要进行本地数据块级重复性检测，避免了跨用户文件级客户端重复性检测中 hash-as-a-proof 方法存在的安全缺陷。MHT-Dedup 方案通过数据块密文的标签生成的验证二叉树提供确定性的文件拥有证明，具有较低的计算和传输开销，而 hMAC-Dedup 方案则通过对抽样数据块密文和其标签进行同态 MAC 运算提供概率性的文件拥有证明，具有较低的额外存储开销。分析与比较表明，本方案在同时支持两级客户端机密数据安全去重和抵抗对数据块的暴力搜索攻击方面具有明显优势。

关键词：云存储；机密数据去重；数据拥有证明；Merkle 散列树；同态 MAC

中图分类号：TP309.7 **文献标识码**：A

Secure sensitive data deduplication schemes based on deterministic/probabilistic proof of file ownership

CHEN Yue[1,2], LI Chao-ling[3], LAN Ju-long[1,4], JIN Kai-chun[1], WANG Zhong-hui[3]

(1. Faculty of Cyberspace Security, PLA Information Engineering University, Zhengzhou 450001, China;
2. State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China;
3. PLA 78179 Unit, Dujiangyan 611800, China;
4. National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450002, China)

Abstract: To solve the difficult problems of sensitive data deduplication in cloud storage, such as detection and PoW (proofs of ownership) of the duplicated ciphertext, the attacks aiming at data sensitivity, etc, a Merkle hash tree based scheme called MHT-Dedup and a homomorphic MAC based scheme called hMAC-Dedup were proposed. Both schemes provided PoW of the ciphertext file to find duplicated files on cross-user file level and check the hash of block plaintext to find duplicated blocks on local block-level, which avoided the security flaws of the hash-as-a-proof method in the cross-user file-level client-side duplication detection. MHT-Dedup provided the deterministic PoW of file with an authenticating binary tree generated from the tags of encrypted blocks, which had lower computing and transferring cost, and hMAC-Dedup provided the probabilistic PoW of file by verifying some sampled blocks and their homomorphic MAC tags, which had lower additional storage cost. Analyses and comparisons show that proposed schemes are preferable in many aspects such as supporting secure two-level client-side sensitive data deduplication and resisting to brute force attack to blocks.

Key words: cloud storage; sensitive data deduplication; proofs of ownership; Merkle hash tree; homomorphic MAC

1 引言

数据去重可以减小云存储的数据传输和存储量，它使服务器对相同文件只存储一次，而不论有多少用户拥有和申请存储该文件。据统计，对于不同应用所产生的不同类型的文件，数据的去重比可以在 1:10 到 1:500 之间，从而节约磁盘空间和带宽 90%以上[1]。实际应用中，所减小的磁盘

收稿日期：2014-09-18；修回日期：2014-11-25
基金项目:国家重点基础研究发展计划(“973”计划)基金资助项目(2012CB315901)
Foundation Item: The National Basic Research Program of China(973 Program)(2012CB315901)

空间和带宽的量与文件类型和去重技术的具体实现相关。

然而，用户将其数据交由不完全可信的云服务提供商（CSP, cloud service provider）管理的同时，对自己数据的安全性总是心存疑虑。尽管 Amazon（S3）、Apple（iCloud）、Dropbox 等主流的 CSP 都在其隐私策略声明中表示其在保有访问用户文件的权力情况下保护用户的隐私[2~6]，但由于利益驱动、误操作、系统漏洞等原因造成的数据泄露事件却时有发生[7,8]。解决云存储数据机密性威胁较好的办法是在客户端对数据加密，并使密钥的生成和保存独立于云存储服务的客户端软件。

1.1 机密数据去重面临的问题

数据去重可以有效提高云存储系统的性能，而机密数据去重却面临着诸多的技术难题[9~11]，包括如下问题。

1) 机密数据重复性检测

当对数据密文去重时，需要解决的首要问题是如何判断多个密文是否来自相同的明文，即机密数据重复性检测问题。另外，机密数据重复性检测方案应提供完善的数据保密性功能，防止来自 CSP 内部（如云存储管理员）和外部的攻击所造成的数据泄露。

2) hash-as-a-proof 方法存在的安全风险

目前，典型的存储系统通常采用文件的摘要作为用户拥有文件的证明，该方法被称为 hash-as-a-proof。它存在以下安全风险[10,12]：第一，恶意用户可以利用存储系统的 hash-as-a-proof 特性将其转变为内容分发网络，达到滥用存储系统的目的；第二，当在客户端对文件加密，并且采用数据明文的摘要以 hash-as-a-proof 执行客户端去重时；攻击者可以对存储系统进行目标冲突攻击（target collision attack），它使用户从云中获得的数据不是所存储的原数据，而是攻击者上传的污染数据（poisoned file）；第三，如果攻击者获得了文件的摘要，他就能轻易地从云中下载到用户的文件。

3) 对数据块的暴力搜索攻击

若一个具有较高最小熵的文件所包含的某些分块的最小熵很小，则攻击者可以通过暴力搜索攻击获得这些分块。

导致该类攻击的原因在于：第一，数据去重机理不可避免地为攻击者提供了对未知信息进行暴力搜索攻击的可能；第二，执行数据块级去重时，分块操作将同时分离各数据块的最小熵，从而使攻击者可以在更小的搜索空间中对各分块进行暴力搜索攻击。一个具有较高最小熵的文件所包含的某些分块的最小熵却很小这种情况是较常见的，因而跨用户、数据块级客户端去重并不适用于机密数据。

针对上述问题，本文提出了 2 种分别基于 MHT 的 MHT-Dedup 方案和基于同态 MAC[13]（homomorphic MAC）的 hMAC-Dedup 方案。它们通过验证文件本身而不是其摘要来进行重复性检测，从而避免了 hash-as-a-proof 带来的安全风险；实现了对数据块的加密保护，并同时实现了跨用户文件级客户端去重和本地数据块级客户端去重，可以避免对数据块的暴力搜索攻击。2 种方案分别实现了确定型和概率型文件重复性检测，并具有不同的性能特征，从而可适用于不同的应用环境和满足不同的应用需求。

1.2 相关工作

文献[14]综述了相同和相似数据检测、冗余复制技术等数据去重基础技术。文献[9]第一次提出了 Farsite 分布式文件系统中通过收敛加密（convergent encryption）解决密文数据去重的方法；但 Farsite 只能执行文件级去重，对存储空间利用率的提高能力有限。同样，文献[15~17]也基于收敛加密来解决机密数据的去重问题。文献[15]的核心思想是将文件分块后对数据块进行收敛加密以实现文件级去重；文献[16]提出了一种通过收敛加密将主机上的文件或整个目录备份到本地服务器的方案；文献[17]将文件在客户端分块和收敛加密后发送到服务器执行服务器端去重，增加了传输带宽的开销。上述这些基于收敛加密的方案都无法避免泄露用户隐私的可能。为此，文献[18,19]提出利用代理重加密方案，并采用 hash-as-a-proof 实现了跨用户、数据块级客户端去重。文献[10]指出了客户端去重中采用 hash-as-a-proof 的安全问题，提出了数据拥有证明（PoW, proof of ownership）的概念，并提出了 3 种 PoW 方案。第一种方案中，先由服务器对文件进行纠错码编码，再对编码后的文件运用 MHT（Merkle hash tree）计算出一个树根节点，执行 PoW 检查时要求用户生成一个同样的 MHT 以进行重复性检测；第二种方案利用两两独立的散列（PIH, pairwise independent hash）H_k 代替纠错码编码，对文件的摘要 $H_k(F)$ 运用 MHT 进行重复性检测；第

三种方案利用一种高效的散列 H_k' 计算文件摘要 $H_k'(F)$，再对 $H_k'(F)$ 运用 MHT 进行重复性检测。这 3 种方案都没有考虑对机密数据的加密保护，并且第二和第三种方案对文件摘要运用 MHT 方法证明文件的存在性，其实质也是 hash-as-a-proof。文献[11]提出了 2 种分别称为 WEAK-CSD 和 STRONG-CSD 的方案，其中 WEAK-CSD 以密文文件的摘要作为标签进行重复性检测，而 STRONG-CSD 是对文献[10]的第二种方案的改进，它对密文文件的摘要运用 MHT 方法进行重复性检测，采用了一种随机预言模型下可证明两两独立的散列函数计算文件摘要，从而对文件的类型具有更广的适用性和更高的安全性。文献[20]提出了一种采用纠删码和 Merkle 树的方案，但它未对机密数据进行加密保护，只针对文件级的去重，且在一次 PoW 中需要用户和服务器间进行多次交互证明。文献[21]分析了跨用户客户端去重存在的 3 类旁道问题：一是攻击者在拥有文件的情况下检测文件是否已经存储在服务器中；二是攻击者在没有文件的情况下，通过反复猜测可能的文件内容并试图将其存储到服务器中，达到成功猜测文件内容的目的；三是攻击者在其他用户的主机上安装恶意软件后，通过该主机向服务器存储数据时的去重功能，实现恶意软件与其远程控制中心的通信，即实现一个隐通道（convert channel）。这 3 类旁道问题是目前的客户端去重方案都还未解决的。文献[22]提出了一种结合跨用户文件级服务器端去重和本地数据块级客户端去重的去重方案，但它没有考虑文件的加密保护，并且采用跨用户文件级服务器端去重使系统的数据传输量较大。为提高去重的执行效率，文献[23]提出了一种通过随机比特抽样来预先生成挑战和应答的 PoW 方案。文献[24]针对目前在 PoW 协议中用户需要将所有数据块的摘要发送给服务器以进行重复性检测的问题，提出了一种能减小元数据传输量的 Hash Challenges 协议；但该协议中，数据块是否已经存在于服务器上是由客户端最终做出判断的，服务器无法检查该判断的真实性，导致恶意用户可以进行欺骗攻击。

综上，现有数据去重方案要么未提供对机密数据的加密保护功能，要么在机密数据去重安全性方面，特别是在抗攻击能力方面还存在许多缺陷，本文将针对这些缺陷进行改进，以期解决机密数据去重所面临的问题。

2　机密数据去重存储系统模型

2.1　模型架构

支持机密数据去重的存储系统模型架构如图 1 所示。User 端具有一个文件代理负责文件存储、访问的各项功能，并提供与用户、主服务器和存储服务器的接口。CSP 由主服务器和存储服务器构成。其中，主服务器负责执行跨用户文件重复性检测或本地数据块重复性检测、为数据块生成验证标签、管理和存储元数据；存储服务器负责存储数据块密文，以及向 User 发送其申请访问的文件。

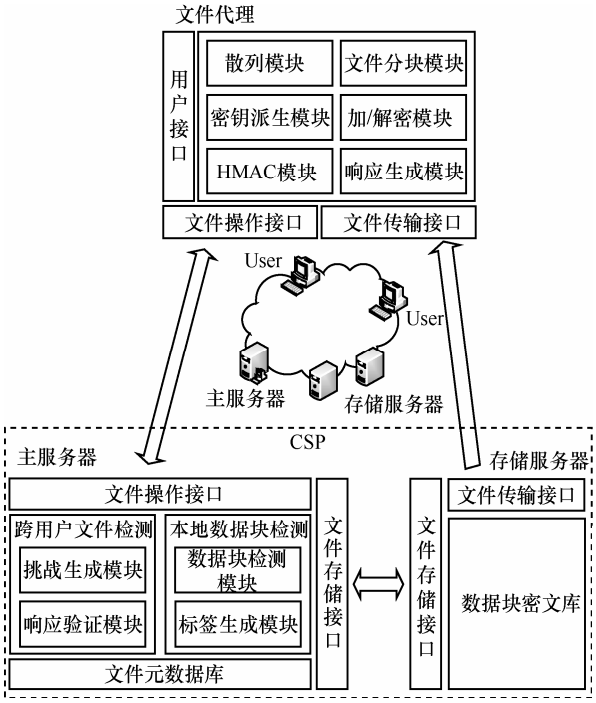

图 1　机密数据去重存储系统模型架构

2.2　模型目标

针对当前的存储系统在机密数据去重方面的问题，该模型希望实现以下目标。

1) 对机密数据的去重

User 通常将机密数据加密后存储到云中，因而 CSP 需要针对密文数据进行重复性检测。

2) 安全的数据去重

能保护数据机密性、抵抗 hash-as-a-proof 引起的攻击和对数据块的暴力搜索攻击。

3) 两级数据去重

执行跨用户文件级客户端去重和本地数据块级客户端去重的两级去重。采用客户端去重可以减小网络上的数据传输量，将跨用户文件级和本地数

据块级去重相结合，可以避免对数据块的暴力搜索攻击。

2.3 模型假设

1) CSP：CSP 能够遵从协议诚实地执行 User 的数据操作请求，但其可能利用掌握的信息查看数据的内容，并将其泄露给其他非授权实体。

2) User：当数据被存储到 CSP 后，User 会将该数据从本地存储中删除以节省存储空间。可能存在恶意 User，他们可能滥用存储系统进行内容分发。

3) 加密算法：加密算法是语义安全的。

4) 散列函数：所采用的散列函数是密码学安全的。

5) 安全信道：User 和 CSP，以及主服务器和存储服务器间都存在安全信道，该信道可以通过共享密钥或公/私钥对等方式建立。

2.4 攻击模型

主要考虑如下 3 类攻击者。

1) 恶意 User

该类 User 是存储系统的真实使用者，在某些情况下，该类用户可能滥用存储系统进行内容分发。

2) 外部攻击者

外部攻击者可能通过某些方法得到部分文件信息，如文件的摘要、个别分块的内容等，从而试图利用这些文件信息假冒真实的 User 与 CSP 交互，以达到获取文件内容或实施目标冲突攻击的目的。

3) 内部攻击者

CSP 将维护用户数据的完整性并保证存储服务的可用性，但其可能未经用户同意而搜索用户的机密数据，甚至可能将获得的数据泄露给第三方。

2.5 模型描述

该模型的算法包括 *Challenge*、*Prove* 以及 *Verify* 等，根据文件级和数据块级去重操作的不同，执行分支有所不同，算法执行流程如图 2 所示。

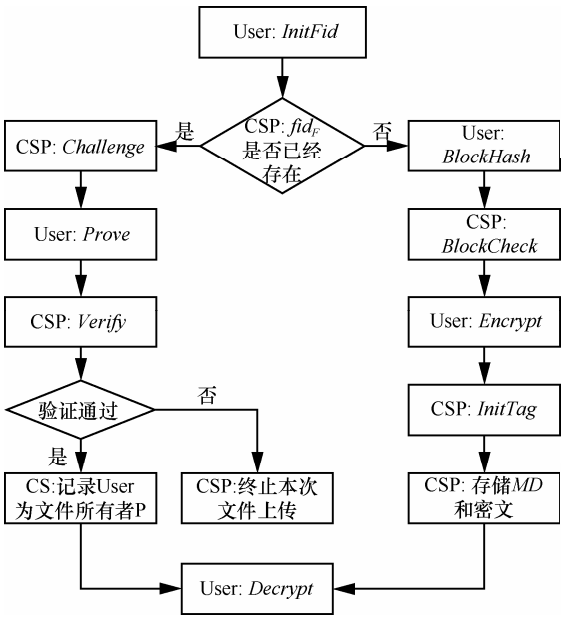

图 2 模型的算法执行流程

1) $InitFid(F)\rightarrow fid_F$

文件标识生成算法。User 利用散列函数计算文件 F 的摘要，并将其作为该文件的标识。

2) $BlockHash(F,bp_F)\rightarrow(\{F_i\},\{(i,h_{F_i})\})$

数据块摘要算法。User 根据数据块划分参数 bp_F 将文件 F 划分为 n 个块 $\{F_i\}\,(1\leqslant i\leqslant n)$，并计算得到各数据块的摘要集 $\{(i,h_{F_i})\}\,(1\leqslant i\leqslant n)$。

3) $BlockCheck(\{(i,h_{F_i})\},MD_0)\rightarrow\{(i,fid_{F'},lb_j,mb_j)\}$

重复数据块检测算法。CSP 将文件 F 的数据块摘要集 $\{(i,h_{F_i})\}$（$1\leqslant i\leqslant n$）与该 User 已存储文件的所有数据块摘要对比，若 h_{F_i} 已经存在，则为数据块 i 生成一个元数据项 $(i,fid_{F'},lb_j,mb_j)$。其中，MD_0 为 User 的所有已存储文件的元数据，它包含了文件标识、数据块的摘要、数据块密钥的密文 lockbox、以及与密钥派生树中的密钥节点的映射 mapbox 等信息；$fid_{F'}$ 为文件 F' 的标识，它是 User 存储在云服务器中并且其数据块 F_j' 与 F_i 相同的文件；lb_j 和 mb_j 分别为 F_j' 的 lockbox 和 mapbox 条目。

4) $Encrypt(\{F_i\},p,n,1^\lambda,ft,fs,\{(i,fid_F,lb_j,mb_j)\})\rightarrow(C_F,MD_F)$

文件加密算法。User 对文件 F 的各数据块加密。其中，p 为密钥派生树的高度，n 为数据块数目，λ 为密钥安全参数，ft 和 fs 分别为文件 F 的类型和大小，C_F 为各数据块的密文集，MD_F 为文件 F 加密后得到的元数据。

5) $InitTag(C_F,MD_F)\rightarrow MD'_F$

验证标签生成算法。CSP 对密文文件 C_F 计算验证标签 T_F，并将其添加到 MD_F 中得到新的元数据 MD'_F。

6) $Challenge(n,MD_{F''})\rightarrow(bp_{F''},c,r,\tau_{F''},\{(i,lb_i,mb_i)\}_{F''})$

数据块挑战生成算法。为检查 User 是否确实拥有所申请上传的文件 F，CSP 生成一个数据块挑战。其中，n 和 $MD_{F''}$ 分别为文件 F'' 的数据块数目和元数据，并且满足 $fid_{F''}=fid_F$，即文件 F'' 与 F 具有相同的标识。c 为被挑战的数据块数目，r 为一个随

机参数。$\tau_{F''}$ 为文件 F'' 的根密钥的密文，$\{(i,lb_i,mb_i)\}_{F''}$（$1\leqslant i\leqslant n$）为 $MD_{F''}$ 中各数据块的序号以及 lockbox 和 mapbox 条目的集合。

7) $Prove(F,ft,fs,bp_{F''},c,r,\tau_{F''},\{(i,lb_i,mb_i)\}_{F''})\rightarrow R$

响应生成算法。针对 CSP 的挑战，User 为文件 F 生成相应的响应 R。其中，R 根据检查协议的不同具有不同的内容。

8) $Verify(R,MD_{F''})\rightarrow y$

响应验证算法。CSP 利用文件 F'' 的元数据，验证 User 所返回响应的正确性，验证结果 $y\in\{accept,reject\}$。

9) $Decrypt(C_F,\{i,lb_i,mb_i\}_F)\rightarrow F$

文件解密算法。User 访问文件 F 时，利用从 CSP 获得的数据块密文集 C_F 和 F 的元数据 $\{(i,lb_i,mb_i)\}_F$（$1\leqslant i\leqslant n$）即可解密出该文件。

2.6 模型安全性

该模型执行本地数据块级客户端去重时，由于这些文件被相同的 User 所有，因而在执行文件级去重时不存在目标冲突攻击或将存储系统作为内容分发网络网络等问题。因而，只定义模型在文件级去重上的安全性。

执行文件级去重时的挑战证明协议即是一个 PoW 协议，以 $PoW=(S,\prod)$ 表示该挑战证明协议。其中，$S(F,1^\lambda)$ 表示对文件 F 的初始化存储（$F\in\{0,1\}^n$、安全参数 $\lambda\in N$），$\prod(P(F,1^\lambda)\leftrightarrow V(F'',1^\lambda))$ 表示 CSP 和 User 之间的挑战证明协议。具体地，$S(F,1^\lambda)$ 包括 *InitFid*、*BlockHash*、*BlockCheck*、*Encrypt* 和 *InitTag* 算法，$\prod(P(F,1^\lambda)\leftrightarrow V(F'',1^\lambda))$ 包括 *Challenge*、*Prove* 和 *Verify* 算法。

$PoW=(S,\prod)$ 协议的安全性包括 2 方面：一是任意真实的文件都可以通过验证，称其为协议的正确性；二是任意通过验证的文件都是真实的，称其为协议的完备性。

正确性：若对于任意文件 $F\in\{0,1\}^n$ 和安全参数 $\lambda\in N$，$\prod(P(F,1^\lambda)\leftrightarrow V(S(F,1^\lambda)))\Rightarrow True$ 都成立，则称 $PoW=(S,\prod)$ 是正确的。

完备性：对于任意文件 $F,F''\in\{0,1\}^n$ 和安全参数 $\lambda\in N$，当 $\prod(P(F,1^\lambda)\leftrightarrow V(S(F'',1^\lambda)))\Rightarrow True$ 时，都满足 $F=F''$，则称 $PoW=(S,\prod)$ 是完备的。

3 机密数据去重方法

3.1 MHT-Dedup 方法

1) *InitFid*：User 利用散列函数 $h()$计算文件 F 的摘要，即 $fid_F=h(F)$。这里，除特殊说明外函数 $h()$均采用 SHA-256。

2) *BlockHash*：User 根据数据块划分参数 bp_F，按固定大小分块或变长分块等方法将文件 F 划分为 n 个数据块 $\{F_i\}$（$1\leqslant i\leqslant n$），并计算每个数据块的摘要 $h_{F_i}=h(F_i)$（$1\leqslant i\leqslant n$）。

3) *BlockCheck*：CSP 收到 User 发送的待上传文件 F 的数据块摘要集 $\{(i,h_{F_i})\}$（$1\leqslant i\leqslant n$）时，查找该 User 的已存储文件是否包含摘要为 h_{F_i}（$1\leqslant i\leqslant n$）的数据块。若包含该数据块，则为其生成一条元数据项 $(i,fid_{F'},lb_j,mb_j)$，从而 User 可以确定文件 F 的第 i 个数据块与文件 F' 第 j 个数据块相同，F_j' 的数据密钥的密文为 lb_j，并且其控制密钥在密钥树中的位置为 mb_j。对于每条元数据项 $(i,fid_{F'},lb_j,mb_j)$，CSP 还需要生成一条元数据项 (i,T_j) 用于 *InitTag* 算法中更新文件 F 的元数据，其中 T_j 为 F_j' 的验证标签。

4) *Encrypt*：由于文件 F 未存储于云服务器中，因而 User 需要将其加密后上传。

首先，User 初始化一棵高度为 p、包含 n 个叶节点密钥的密钥树[25]，并将叶节点密钥转换为变换叶节点密钥，以变换叶节点密钥作为控制密钥。其中，树高度 p 与数据块数目 n 满足 $2^{p-1}<n\leqslant 2^p$。User 根据安全参数 λ 随机选择一个根密钥 $k_{0,1}$，再利用左右孩子派生规则 f_L 和 f_R 逐级计算得到树节点密钥。当 $k_{i,j}$ 为非叶节点密钥时，其左、右孩子节点密钥分别为 $k_{i+1,2j-1}$ 和 $k_{i+1,2j}$。其中，$k_{i+1,2j-1}=f_L(k_{i,j})=h(k_{i,j}\|i\|(2j-1))$，$k_{i+1,2j}=f_R(k_{i,j})=h(k_{i,j}\|i\|2j)$，“$\|$”表示字符串的联接。完成密钥树的初始化后，对叶节点密钥 $k_{p,i}$（$1\leqslant i\leqslant n$）运算得到变换叶节点密钥 $k'_{p,i}=f(k_{p,i})=h(k_{p,i}\|p\|i\|fa)$。其中，$fa=h(ft\|fs)$ 被称为文件属性。

其次，User 执行数据块加密。采用密钥长度为 256 bit 的 AES 算法进行加密。对于数据块 F_i，若 i 不包含于 $(i,fid_{F'},lb_j,mb_j)$ 之内，则 User 为其随机生成一个数据密钥 k_i，得到数据块密文 $C_i=E_{k_i}(F_i)$、该数据块的 lockbox 项 $lb_i=E_{k'_{p,i}}(k_i)$ 以及其 mapbox 项 $mb=(p,i)$。其中，mapbox 项记录了该数据块的控制密钥在密钥树中的位置。而对于数据块 F_i，若 i 包含于 $(i,fid_{F'},lb_j,mb_j)$ 之内，则 User 从文件 F_i 的根密钥 $k'_{0,1}$ 派生得到位置为 mb_j 的叶节点密钥，并计算得到相应的变换叶节点密钥（由于 F' 为 User

自身的文件，因而User知道F'的fa）。进一步地，User利用该变换叶节点密钥解密lb_j得到F'的一个数据密钥，并将该密钥作为数据块F_i的数据密钥k_i。此时，User只需要计算$lb_i = E_{k'_{p,i}}(k_i)$并记录$mb = (p,i)$，而不需要再计算$C_i = E_{k_i}(F_i)$。

再次，User对根密钥$k_{0,1}$加密。

User首先以属性fa为密钥计算文件F的HMAC-SHA256值，即$HMAC_F = HMAC - SHA256_{fa}(F)$，再按$\tau_F = HMAC_F \oplus k_{0,1}$加密根密钥$k_{0,1}$。

最后，User将加密结果组成数据密文$C_F = \{(h_{F_i}, C_i)\}(1 \leqslant i \leqslant n$，但不包含与$F'$重复的数据块)和元数据$MD_F = (fid_F, bp_F, \tau_F, \{(i, h_{F_i}, lb_i, mb_i)\}_F)(1 \leqslant i \leqslant n)$。

5) *InitTag*：首先，CSP以散列运算计算各数据块的验证标签。

对数据密文$C_F = \{(h_{F_i}, C_i)\}$中的每个密文块C_i进行散列运算，并以该摘要值为数据块的验证标签，即$T_i = h(C_i)$。而对于在*BlockCheck*算法中记录的所有元数据项(i, T_j)中的数据块，将其验证标签赋值为$T_i = T_j$。其次，CSP以所有数据块的验证标签T_j（$1 \leqslant i \leqslant n$）为叶节点，并以数据块序号$i$为序，由叶节点逐级往上生成一棵如图3所示的验证二叉树$Tree_F$，并以$T_{root,F}$表示树根节点。最后，CSP更新文件F的元数据为$MD'_F = (fid_F, bp_F, \tau_F, T_{root,F}, \{(i, h_{F_i}, lb_i, mb_i, T_i)\}_F)(1 \leqslant i \leqslant n)$。

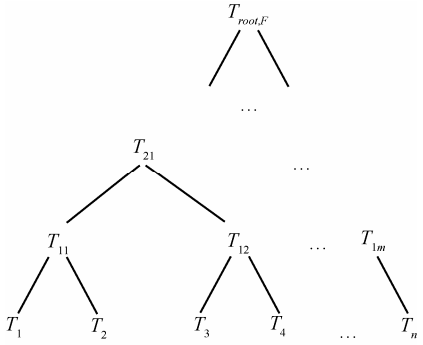

图3 验证二叉树

6) *Challenge*：CSP随机选择一个数c（$1 \leqslant c \leqslant n$）作为被挑战的数据块数目，并生成一个随机数$r$作为伪随机置换的密钥。另外，CSP还需要从元数据$MD_{F''}$中提取$bp_{F''}$、$\tau_{F''}$和$\{(i, lb_i, mb_i)\}_{F''}$（$1 \leqslant i \leqslant n$），并将它们与$c$、$r$一起返回给User。

7) *Prove*：首先，User解密得到F''的根密钥$k''_{0,1}$。

利用待上传文件F的类型ft和大小fs计算得到文件属性$fa = h(ft \| fs)$，并进而计算得到$HMAC_F = HMAC - SHA256_{fa}(F)$，以及$k_{0,1}'' = HMAC_F \oplus \tau_{F''}$。

其次，User对文件分块并加密各数据块。

根据数据块划分参数$bp_{F''}$将文件F划分为n个块$\{F_i\}(1 \leqslant i \leqslant n)$，并由根密钥$k''_{0,1}$派生得到位置为$mb_i(1 \leqslant i \leqslant n)$的$n$个叶节点密钥$k_{p,i}(1 \leqslant i \leqslant n)$，从而可得到变换叶节点密钥$k'_{p,i} = f(k_{p,i}) = h(k_{p,i} \| p \| i \| fa)(1 \leqslant i \leqslant n)$。进一步地，User利用$k'_{p,i}$解密相应的$lb_i$以得到数据块密钥$k_i = E^{-1}_{k'_{p,i}}(lb_i)(1 \leqslant i \leqslant n)$，并按$C_i = E_{k_i}(F_i)(1 \leqslant i \leqslant n)$加密各数据块。

再次，User为文件F生成一棵验证二叉树。

对各数据块密文C_i计算摘要值作为验证标签，即$T_i = h(C_i)(1 \leqslant i \leqslant n)$，并以$T_i(1 \leqslant i \leqslant n)$为叶节点、数据块序号$i$为序，由叶节点逐级往上生成一棵验证二叉树。

最后，User返回验证二叉树的部分节点作为响应R。

以文件F的块数n、被挑战数据块数目c和随机数r为伪随机置换函数σ的输入，得到c个数据块序号$i_j = \sigma_r(c,n)(1 \leqslant j \leqslant c, 1 \leqslant i_j \leqslant n)$。User将验证二叉树中被挑战块$i_j(1 \leqslant j \leqslant c)$的叶节点及其到根节点路径上的兄弟节点作为响应$R$返回给CSP。

8) *Verify*：CSP利用响应R中的节点生成一棵验证二叉树，比较该树的根节点$T'_{root,F}$与$T_{root,F''}$是否相等。

若相等则表示User拥有文件F，并且文件F与F''是同一文件；若不相等，则表示User所申请上传的文件F与CSP所存储的文件F''具有相同的标识，但文件内容不同，两者中存在文件内容与标识不相符的情况。当出现文件内容与标识不相符的情况时，为确定文件F与F''中哪个为伪造的，CSP（或一个可信的仲裁方）需要要求该User出示文件F的明文，并同时要求文件F''的所有者出示解密F''的根密钥及文件属性等信息，由CSP（或可信的仲裁方）分别对文件F和F''的明文计算摘要，若所计算摘要与文件标识不相等，则表示该文件为伪造的。

9) *Decrypt*：User需要访问文件F时，从CSP

获得其密文 C_F 和元数据 $\{i,lb_i,mb_i\}_F(1\leqslant i\leqslant n)$，再利用本地存储的根密钥 $k_{0,1}$ 和文件参数 *ft*、*fs*，即可按生成 *F* 的密钥树、计算变换叶节点密钥、解密出数据密钥以及解密数据块的顺序得到文件 *F*。

3.2 hMAC-Dedup 方法

除 *InitTag*、*Challenge*、*Prove* 和 *Verify* 算法外，hMAC-Dedup 的 *InitFid* 和 *Encrypt* 等算法与 MHT-Dedup 相同。

1) *InitTag*：CSP 以同态 MAC[13]计算各数据块的验证标签。

首先，初始化同态 MAC 运算的参数。CSP 选择一个 *k* bit 的素数 *q*，使其满足 $C_i\in Z_q(1\leqslant i\leqslant n)$。同时，CSP 随机选择 2 个参数 *s* 和 α，使其满足 $s\in Z_q^*$ 和 $\alpha\in Z_q^*$，并计算 $s_i=G(s,i)(1\leqslant i\leqslant n)$。其中，*G*() 为一个安全的伪随机数发生器，并且是对所有 User 公开的。

其次，CSP 计算每个数据块的验证标签。对数据密文 $C_F=\{(h_{F_i},C_i)\}$ 中的每个密文块 C_i，CSP 计算 $T_i=\alpha C_i+s_i \bmod q$。而对于在 *BlockCheck* 算法中记录的所有元数据项 (i,T_j) 中的数据块，将其验证标签赋值为 $T_i=T_j$。从而，得到所有数据块的验证标签 $T_i(1\leqslant i\leqslant n)$。

最后，CSP 更新文件 *F* 的元数据为 $MD'_F=(fid_F,bp_F,\tau_F,s,\alpha,\{(i,h_{F_i},lb_i,mb_i,T_i)\}_F)(1\leqslant i\leqslant n)$。

2) *Challenge*：CSP 随机选择一个数 $c(1\leqslant c\leqslant n)$ 作为被挑战的数据块数目，并生成 2 个随机数 *r* 和 β。其中，$r\in Z_q^*$ 为一个系数因子，β 为一个伪随机置换的密钥。CSP 以文件 *F*" 的块数 *n*、被挑战数据块数目 *c* 和随机数 β 为伪随机置换函数 σ 的输入，得到 *c* 个数据块序号 $i_j=\sigma_\beta(c,n)(1\leqslant j\leqslant c$，$1\leqslant i_j\leqslant n)$。从而，CSP 将参数 *c*、*r* 和元数据 $bp_{F"}$、$\tau_{F"}$、$\{(i_j,lb_{i_j},mb_{i_j})\}_{F"}(1\leqslant j\leqslant c)$一起返回给 User。

3) *Prove*：首先，User 解密得到 *F*" 的根密钥 $k"_{0,1}$。

利用待上传文件 *F* 的类型 *ft* 和大小 *fs* 计算得到文件属性 $fa=h(ft\,\|\,fs)$，并进而计算得到 $HMAC_F=HMAC-SHA256_{fa}(F)$，以及 $k_{0,1}"=HMAC_F\oplus\tau_{F"}$。

其次，User 对文件分块并加密被挑战的数据块。

根据数据块划分参数 $bp_{F"}$ 将文件 *F* 划分为 *n* 个块 $\{F_i\}(1\leqslant i\leqslant n)$，并由根密钥 $k"_{0,1}$ 派生得到位置为 $mb_{i_j}(1\leqslant j\leqslant c)$的 *c* 个叶节点密钥 $k_{p,i_j}(1\leqslant j\leqslant c)$，从而可得到变换叶节点密钥 $k'_{p,i_j}=f(k_{p,i_j})=h(k_{p,i_j}\,\|\,p\,\|\,i_j\,\|\,fa)(1\leqslant j\leqslant c)$。进一步地，User 利用 k'_{p,i_j} 解密相应的 lb_{i_j} 以得到数据块密钥 $k_{i_j}=E^{-1}_{k'_{p,i_j}}(lb_{i_j})(1\leqslant j\leqslant c)$，并按 $C_{i_j}=E_{k_{i_j}}(F_{i_j})(1\leqslant j\leqslant c)$加密被挑战数据块。

最后，User 计算一个被挑战数据块的验证值 $A=\sum_{j=1}^{c}r^{i_j}C_{i_j} \bmod q$，并将 *A* 作为响应 *R* 返回给 CSP。

4) *Verify*：CSP 对被挑战数据块的标签计算一个值 $B=\sum_{j=1}^{c}r^{i_j}T_{i_j} \bmod q$，并计算 $S=\sum_{j=1}^{c}r^{i_j}G(s,i_j) \bmod q$。从而，CSP 检查 $B\equiv\alpha A+S(\bmod q)$ 是否成立。若成立则表示 User 拥有文件 *F*，并且文件 *F* 与 *F*" 是同一文件；若不成立，则表示 User 所申请上传的文件 *F* 与 CSP 所存储的文件 *F*" 具有相同的标识，但文件内容不同，两者中存在文件内容与标识不相符的情况，需要对文件内容的真假进行仲裁。

3.3 元数据管理

数据块加密完成后，User 将加密结果发送给 CSP，并在本地为文件 *F* 存储 fid_F、$k_{0,1}$、*fs*、*ft*、*n* 等参数。CSP 调用 *InitTag* 算法对数据块密文计算验证标签，并将元数据和数据密文分别存储到主服务器和存储服务器。其中，MHT-Dedup 中文件 *F* 的元数据包括 fid_F、bp_F、τ_F、$T_{root,F}$ 以及存储数据块参数的 Block-Info 表；hMAC-Dedup 中文件 *F* 的元数据包括 fid_F、bp_F、τ_F、*s*、α 和 Block-Info 表。Block-Info 表的结构如表 1 所示。

表 1　　Block-Info 元数据

块序号	块摘要	lockbox	mapbox	块标签
1	h_{F_1}	lb_1	mb_1	T_1
2	h_{F_2}	lb_2	mb_2	T_2
…	…	…	…	…
n	h_{F_n}	lb_n	mb_n	T_n

4 安全性证明

4.1 MHT-Dedup 的安全性

正确性：显然，MHT-Dedup 中对于任意文件 *F*，$PoW=(S,\prod)$ 协议在 $S(F,1^\lambda)$ 的 *InitTag* 算法和 $\prod(P(F,1^\lambda)\leftrightarrow V(F,1^\lambda))$ 的 *Prove* 算法中将生成相同

的验证二叉树，从而 $\prod(P(F,1^\lambda)\leftrightarrow V(S(F,1^\lambda)))\Rightarrow$ True 对任意文件都成立，即 MHT-Dedup 的 $PoW=(S,\prod)$ 协议是正确的。

完备性：MHT-Dedup 的 $PoW=(S,\prod)$ 协议的完备性证明与文献[10]的第一种方案的安全性证明类似，若其不满足完备性，则攻击敌手可以找出一个对散列函数 $h()$的碰撞，它与散列函数是密码学安全的假设相违被。限于篇幅，这里不再详述证明过程。

4.2 hMAC-Dedup 的安全性

正确性：hMAC-Dedup 中对于任意文件 F，$PoW=(S,\prod)$ 协议在 $S(F,1^\lambda)$ 的 *InitTag* 算法中生成的标签集 T_{i_j} $(1\leqslant j\leqslant c)$、$\prod(P(F,1^\lambda)\leftrightarrow V(F,1^\lambda))$ 的 *Prove* 算法生成的 A，以及 *Verify* 算法生成的 B 和 S，满足以下等式

$$\begin{aligned}B&=\sum_{j=1}^{c}r^{i_j}T_{i_j}\bmod q\\&=\sum_{j=1}^{c}r^{i_j}(\alpha C_{i_j}+s_{i_j})\bmod q\\&=\sum_{j=1}^{c}\alpha r^{i_j}C_{i_j}+r^{i_j}s_{i_j}\bmod q\\&=\sum_{j=1}^{c}\alpha r^{i_j}C_{i_j}+\sum_{j=1}^{c}r^{i_j}s_{i_j}(\bmod q)\\&=\alpha\sum_{j=1}^{c}r^{i_j}C_{i_j}+\sum_{j=1}^{c}r^{i_j}G(s,i_j)(\bmod q)\\&\equiv\alpha A+S(\bmod q)\end{aligned}$$

所以，hMAC-Dedup 的 $PoW=(S,\prod)$ 协议是正确的。

完备性：由于 hMAC-Dedup 对数据块进行抽样检查，在完备性证明中令 $c=n$，即在挑战文件 F 的所有数据块的情况下证明 hMAC-Dedup 的完备性。

假设存在一个攻击敌手 A 能以至少 0.5 的概率通过 $PoW=(S,\prod)$ 的检查，即 $P(\prod(P(F,1^\lambda)\leftrightarrow V(S(F'',1^\lambda)))\Rightarrow\text{True})\geqslant 0.5$。若能找出一个算法可以从敌手 A 恢复出文件 F''，则表明敌手 A 至少拥有一个文件 F 满足 $F=F''$。

$A=\sum_{i=1}^{n}r^iC_i\bmod q$ 是一个以 r 为变量且度数为 n 的多项式。对于任意一次 $\prod(P(F,1^\lambda)\leftrightarrow V(S(F'',1^\lambda)))\Rightarrow$ True 的执行，*Prove* 算法生成的值 A_j 都是该多项式取变量值 r_j 的实例，即 (r_j,A_j) 为该多项式的一个取值对。因而，当 CSP 对已存储文件 F'' 向敌手 A 发起足够多次挑战，使 $\prod(P(F,1^\lambda)\leftrightarrow V(S(F'',1^\lambda)))\Rightarrow$ True 至少成功执行 $n+1$ 次时，可以得到 $n+1$ 个不同的多项式取值对 $(r_j,A_j)(1\leqslant j\leqslant n+1)$，从而根据拉格朗日插值公式可以唯一解出该多项式的系数 $\{C_i\}(1\leqslant i\leqslant n)$。集合 $\{C_i\}(1\leqslant i\leqslant n)$即为文件 F'' 的密文块集，因而可以从敌手 A 恢复出文件 F'' 的密文。在加密算法安全的假设下，即可以从敌手 A 恢复出文件 F''。

所以，当 CSP 对文件 F 的所有数据块进行挑战（即 $c=n$）时，hMAC-Dedup 的 $PoW=(S,\prod)$ 协议是完备的。

当 hMAC-Dedup 在 $PoW=(S,\prod)$ 协议中对数据块进行抽样检查时，根据文献[26]对外包数据抽样检查的证明结论，若 User 所掌握的文件 F'' 的信息量与其最小熵的比率和 $PoW=(S,\prod)$ 协议的置信率都一定时，要达到该置信率每次所需检查的分块数是一定的，它与文件 F'' 的总分块数无关。例如，当 User 掌握了文件 F'' 99%的最小熵时，每次只需检查 460 个分块即可以 99%的概率发现该 User 不拥有文件 F''。因而，hMAC-Dedup 对数据块进行抽样检查也能以较高概率保证 $PoW=(S,\prod)$ 协议的完备性，同时降低了协议执行中的计算开销。

4.3 具有的安全特性

由于 PoW 协议中的检查对象为密文文件而不是文件（或密文文件）的摘要，因而 MHT-Dedup 和 hMAC-Dedup 避免了 hash-as-a-proof 的弊端，达到了以下目的。

第一，任何用户都只有通过 PoW 协议的验证后才能被 CSP 记录为文件所有者，因而恶意用户不能通过共享文件摘要来使其他用户从云中获得文件，能够有效防止恶意用户利用存储系统进行内容分发的行为。

第二，即使外部攻击者通过某种方式获得了文件摘要或部分文件内容，也不能通过 PoW 协议的验证，因而一方面他无法获得文件内容，另一方面若其进行目标冲突攻击，CSP 将发现具有相同摘要但 PoW 检查却无法通过的文件，从而通过仲裁揭露攻击者的身份，阻止了攻击者将污染数据分发给其他用户的企图。

第三，由于文件在用户端进行加密，CSP 只对密文文件计算 PoW 协议的检查标签，因而能有效防止内部攻击者访问和获取用户机密数据

的企图。

另外，MHT-Dedup 和 hMAC-Dedup 采用跨用户文件级客户端去重和本地数据块级客户端去重相结合，能够避免对数据块的暴力搜索攻击。

5　性能分析

5.1　计算开销

以密码运算来度量各阶段的计算开销，分别以 T_h、T_{HM}、T_σ 和 T_E 分别表示 $h(\cdot)$、$HMAC(\cdot)$、$\sigma(\cdot)$ 和对称加/解密 $E(\cdot)/E^{-1}(\cdot)$ 的单次运算开销。

对于一个文件 F'' 只需要进行一次初始化存储 $S(F'',1^\lambda)$，而可能反复多次执行 $\prod(P(F,1^\lambda)\leftrightarrow V(S(F'',1^\lambda)))$ 协议，并且在一个具有数据去重功能的存储系统中，用户希望通过高效的重复文件验证协议获得良好的使用体验。因而，只对 MHT-Dedup 和 hMAC-Dedup 在 $\prod(P(F,1^\lambda)\leftrightarrow V(S(F'',1^\lambda)))$ 协议中的计算开销进行分析。

MHT-Dedup 在 $\prod(P(F,1^\lambda)\leftrightarrow V(S(F'',1^\lambda)))$ 协议中，User 执行 *Prove* 算法，解密文件 F'' 的根密钥的计算开销为 T_h+T_{HM}；生成变换叶节点密钥的计算开销为 $(2n+2^p-2)T_h$；解密数据密钥和加密数据块的计算开销为 $2nT_E$；计算数据块验证标签的计算开销为 nT_h；生成验证二叉树的计算开销为 $(n-1)T_h$；伪随机置换的计算开销为 cT_σ。该协议中，User 的总计算开销为 $(2n+2^p-2)T_h+T_{HM}+2nT_E+cT_\sigma$。CSP 执行 *Challenge* 和 *Verify* 算法，根据 c 个被挑战数据块的标签及其兄弟路径节点计算验证二叉树的开销最大为 $(n-1)T_h$。

hMAC-Dedup 在 $\prod(P(F,1^\lambda)\leftrightarrow V(S(F'',1^\lambda)))$ 协议中，User 执行 *Prove* 算法，解密文件 F'' 的根密钥的计算开销为 T_h+T_{HM}；生成 c 个变换叶节点密钥的计算开销最大为 $(2c+2^p-2)T_h$；解密数据密钥和加密数据块的计算开销为 $2cT_E$；计算验证值 A 的开销为 T_Σ。其中，以 T_Σ 表示 $X=\sum_{i=1}^{c}r^{i_j}x_{i_j} \bmod q$ 的计算开销，它由 c 的大小和序号 i_j 的大小决定。该协议中，User 的总计算开销为 $(2c+2^p-1)T_h+T_{HM}+2cT_E+T_\Sigma$。CSP 执行 *Challenge* 和 *Verify* 算法，伪随机置换的计算开销为 cT_σ；计算值 B 和 S 的开销为 $2T_\Sigma$，因而总计算开销为 $cT_\sigma+2T_\Sigma$。

MHT-Dedup 和 hMAC-Dedup 中的计算开销如表 2 所示。

表 2　MHT-Dedup 和 hMAC-Dedup 的计算开销

方案		*Prove*	*Challenge*/*Verify*
MHT-Dedup	User	$(4n+2^p-1)T_h+T_{HM}+2nT_E+cT_\sigma$	/
	CSP	/	$(n-1)T_h$
hMAC-Dedup	User	$(2c+2^p-1)T_h+T_{HM}+2cT_E+T_\Sigma$	/
	CSP	/	$cT_\sigma+2T_\Sigma$

计算开销测试的实验环境为：Intel(R) Core(TM) i3 2.93 GHz 双核处理器，4 GB 内存，Windows 7 Professional 系统，密码算法采用 ActivePerl-5.14.2.1402-MSWin32-x86-295342 和 Win32OpenSSL-0.9.8v，编程环境为 Microsoft Visual C++ 6.0。散列运算采用 SHA-1，AES 算法密钥长度为 128 bit，采用 AES_ecb_encrypt 进行电子密码本模式加密。另外，以 AES 算法来实现 $\sigma(\cdot)$ 运算。

对一个 64 GB 的文件按 4 kB 进行分块，并在每次协议运行中挑战 500 个分块，测试结果为：MHT-Dedup 中运行一次挑战-证明协议，User 的计算开销为 1 000 s，CSP 的计算开销为 200 s。而在 hMAC-Dedup 中运行一次挑战-证明协议，User 和 CSP 的计算开销分别为 220 s 和 200 ms。可以看出，一方面由于 User 在这 2 个协议中需要生成一棵叶节点数为 n 的密钥树，因而其计算开销比 CSP 大；另一方面，由于 MHT-Dedup 中 User 和 CSP 都需要生成一棵验证二叉树，而 hMAC-Dedup 对数据块抽样并采用同态运算进行验证，因而 MHT-Dedup 比 hMAC-Dedup 的计算开销大。

5.2　存储开销

MHT-Dedup 中文件 F 的元数据包括 fid_F、bp_F、τ_F 和 $t_{root,F}$ 以及存储数据块参数的 Block-Info 表。其中，fid_F、bp_F、τ_F 和 $t_{root,F}$ 的大小相对于文件 F 都可以忽略，主要的额外存储开销来自于 Block-Info 表。在一个 Block-Info 条目中，块序号和 mapbox 值的大小共为 $3\log n$ bit，而块摘要、lockbox 值以及块标签的大小均为 256 bit。所以 MHT-Dedup 中 CSP 的主要额外存储开销为 $n(3\log n+768)$ bit。以 S_F 表示文件 F 的大小，u 表示存储了文件 F 的 User 的数目（即不进行数据去重的情况下，文件 F 被存储的数目），则 CSP 的额外存储率 $ExRate_{\text{CSP}}$ 为

$$ExRate_{\text{CSP}}=\frac{n(3\log n+768)}{uS_F}$$

同样，hMAC-Dedup 主要的额外存储开销也来自于 Block-Info 表。hMAC-Dedup 中，由于验证标签 $t_i \in Z_q$ 和密文块 $c_i \in Z_q$，因而标签集 T 的大小可用密文 C 的大小来衡量，则 CSP 的主要额外存储开销为 $n(3\log n + 512) + S_F$ bit，其额外存储率 $ExRate_{\text{CSP}}$ 为

$$ExRate_{\text{CSP}} = \frac{n(3\log n + 512) + S_F}{uS_F}$$

在额外存储率指标中，数据块数目 n 是由分块策略决定的，它反映了分块策略对额外存储率的影响；存储文件 F 的用户数 u 表示在文件级去重的程度，反映了数据去重率对额外存储率的影响。因而，额外存储率公式也反映了额外存储开销、分块策略和数据去重率之间的约束关系。

可以看出，MHT-Dedup 中 CSP 的额外存储开销只与数据块数目相关，而与文件大小、用户数目等因素无关，但其额外存储率则与数据块数目、文件大小和存储文件 F 的用户数目相关；hMAC-Dedup 中 CSP 的额外存储开销与数据块数目和文件大小相关，除这 2 个因素外，其额外存储率还与存储文件 F 的用户数目相关。

例如，对一个 64 GB 的文件按 4 kB 进行分块，并假设该文件被 100 个用户所存储，则 MHT-Dedup 中 CSP 的额外存储开销约为 1.64 GB，其额外存储率约为 0.026%；hMAC-Dedup 中 CSP 的额外存储开销约为 65.14 GB，其额外存储率约为 1.02%。

5.3 传输开销

与计算开销一样，只分析在 $\prod(P(F,1^\lambda) \leftrightarrow V(S(F'',1^\lambda)))$ 协议中的传输开销。

MHT-Dedup 中 CSP 执行 *Challenge* 算法，它向 User 所发送挑战的主要传输开销为 $\{(i,lb_i,mb_i)\}_{F''}$ $(1 \leqslant i \leqslant n)$ 的大小 $n(3\log n + 256)$ bit。User 执行 *Prove* 算法，他向 CSP 返回的响应 R 中最多包括 n 个验证二叉树节点，因而其传输开销为 $256n$ bit。

hMAC-Dedup 中 CSP 向 User 所发送挑战的主要传输开销为 $\{(i_j,lb_{i_j},mb_{i_j})\}_{F''}$ $(1 \leqslant j \leqslant c)$ 的大小 $c(3\log n + 256)$ bit。User 向 CSP 返回的响应 R 只包含验证值 A，其大小为 $O(k)$。

例如，对一个 64 GB 的文件按 4 kB 进行分块，在每次协议运行中挑战 500 个分块，并设参数 k 取 1 024 bit，则 MHT-Dedup 中挑战和响应消息的主要传输开销分别为 560 MB 和 375 kB，hMAC-Dedup 中挑战和响应消息的主要传输开销分别为 20 kB 和 1 024 bit。

5.4 典型方案对比

MHT-Dedup 和 hMAC-Dedup 与其他典型的数据去重方案（对于文献[10,11,23]分别取它们的 Streaming Protocol、STRONG-CSD 和 s-POW3 方案）的特性对比如表 3 所示。其中，c 表示被挑战的数据块（或比特）的数目，n 表示数据块的总数目，s 表示数据块的子块数目，m 表示文件 F 的大小，t 表示预计算的验证标签的数目，p 表示密钥树的高度。检查概率中，P_{un} 表示证明方（User 或攻击者）未掌握的数据块数目（或比特数）与文件 F 的总块数（或比特数）的比率。安全性中，“是”和“否”分别表示该方案能否抵抗 hash-as-a-poof 引起的攻击。另外，存储复杂度指 CSP 为数据去重而增加的额外存储量；通信和计算复杂度指 PoW 协议中的开销。

可以看出，MHT-Dedup 和 hMAC-Dedup 对数据提供了加密保护，实现了两级数据去重，并能抵抗 hash-as-a-proof 所引起的攻击。因为对密文文件而不是文件摘要进行验证以确定用户是否拥有文件，因而 MHT-Dedup 和 hMAC-Dedup 避免了 hash-as-a-proof 的弊端，能够防止恶意用户利用存储系统进行内容分发，以及对数据机密性的内外部攻击。同时，当攻击者进行目标冲突攻击时，CSP 将发现具有相同摘要但 PoW 检查却无法通过的文件，从而通过仲裁揭露攻击者的身份，阻止了攻击者将污染数据分发给其他用户的企图。

在 MHT-Dedup 和 hMAC-Dedup 之间，一方面，二者分别实现了确定型和概率型的机密数据重复性检测。因而，对于机密性强度较高的文件，可以采用 MHT-Dedup 确保用户绝对拥有该文件；而对于机密性强度较低的文件，可以采用 hMAC-Dedup 以较高的概率检查用户是否拥有该文件。另一方面，MHT-Dedup 比 hMAC-Dedup 具有更低的额外存储开销，但 hMAC-Dedup 比 MHT-Dedup 具有更低的通信开销，以及 PoW 协议中更低的计算开销。因而，hMAC-Dedup 适用于利用智能手机、ipad 等移动终端在无线环境下进行云存储的情况；MHT-Dedup 则可以应用在宽带网络和计算能力较强的情况下，能在数据去重的同时进一步减小额外存储开销。

表 3　　典型的数据去重方案对比

指标		Streaming	STRONG-CSD	s-POW3	文献[25]	本文	
						MHT-Dedup	hMAC-Dedup
去重对象		数据的纠删码编码	数据密文	数据明文	数据的纠删码编码	数据密文	数据密文
去重类型		跨用户文件级客户端去重	跨用户文件级客户端去重	跨用户文件级客户端去重	跨用户文件级客户端去重	跨用户文件级客户端去重、本地数据块级客户端去重	跨用户文件级客户端去重、本地数据块级客户端去重
安全性		否	否	是	是	是	是
检查概率		1	1	$1-(1-P_{un})^c$	$1-(1-P_{un})^c$	1	$1-(1-P_{un})^c$
存储复杂度		$O(1)$	$O(1)$	$O(tc)$	$O(n)$	$O(n \log n)$	$O(n \log n)$
通信复杂度		$O(c \log n)$	$O(c \log n)$	$O(c)$	$O(c)$	$O(n \log n)$	$O(n \log n)$
计算复杂度	CSP	$O(n)$	$O(n)$	$O(tc)$	$O(s+c)$	$O(n)$	$O(c+2^p)$
	User	$O(m)$	$O(m)$	$O(c)$	$O(m)$	$O(n+c)$	$O(c)$

6　结束语

对云存储中的机密数据去重需要解决诸多问题，如去重与加密的矛盾、防止恶意用户滥用存储系统进行内容分发等。为此，本文基于 MHT 和同态 MAC 提出了 2 种分别称为 MHT-Dedup 和 hMAC-Dedup 的数据去重方案，实现了对机密数据的跨用户文件级客户端去重和本地数据块级客户端去重，并解决了所提出的几类安全问题。hMAC-Dedup 比 MHT-Dedup 具有更低的计算和传输开销，但 MHT-Dedup 具有更低的额外存储开销。同时，MHT-Dedup 提供确定性的文件拥有证明，而 hMAC-Dedup 通过对数据块抽样提供概率性的文件拥有证明。

下一步将在此基础上，研究降低方案存储复杂度的方法，并进一步研究支持数据块更新的机密数据去重方案。

参考文献：

[1] DUTCH M, FREEMAN L. Understanding data de-duplication ratios[EB/OL]. http://www.sina.org/.2012.

[2] CNET. Who owns your files on google drive[EB/OL]. http://news.cnet.com/8301-10233-57420551-93/who-owns-your-files-on-google-drive/.2016.2013.

[3] Dropbox. Dropbox privacy policy[EB/OL]. https://www.dropbox.com/privacy.2013.

[4] Google. Google terms of service[EB/OL]. http://www.google.com/policies/terms/.2013.

[5] Apple Inc. Apple privacy policy (Covering iCloud)[EB/OL]. http://www.apple.com/privacy/.2013.

[6] Microsoft. Microsoft services agreement[EB/OL]. http://windows.microsoft.com/en-US/windows-live/microsoft-service-agreement. 2013.

[7] Wired.com. Dropbox left user accounts unlocked for 4 hours sunday[EB/OL].http://www.wired.com/threatlevel/2011/06/dropbox/. 2013.

[8] Twitter. Tweetdeck[EB/OL]. http://money.cnn.com /2012/03/30/ technology /tweetdeck-bug-twitter/.2013.

[9] DOUCEUR J R, ADYA A, BOLOSKY W J, *et al*. Reclaiming space from duplicate files in a serverless distributed file system[A]. Proc. of ICDCS'02[C]. 2002.617-624.

[10] SHAI H, DANNY H, BENNY P, *et al.* Proofs of ownership in remote storage systems[A]. Proc. of the 18th ACM conference on Computer and communications security (CCS'11)[C]. New York, USA, 2011. 491-500.

[11] XU J, CHANG E C, ZHOU J Y. Leakage-Resilient Client-side Deduplication of Encrypted Data in Cloud Storage[R]. Cryptology ePrint Archive, Report 2011/538, 2011.

[12] Dropship. Dropbox api utilities[EB/OL]. https://github.com/driverdan/dropship.2013.

[13] CHANG E C, XU J. Remote integrity check with dishonest storage server[A]. Proc. of ESORICS '08: European Symposium on Research in Computer Security: Computer Security[C]. Berlin, Heidelberg, 2008.223-237.

[14] 敖莉, 舒继武, 李明强. 重复数据删除技术[J]. 软件学报，2010，21(5):916-929.
AO L, SHU J W, LI M Q. Data deduplication techniques[J]. Journal of Software, 2010, 21(5):916-929.

[15] 王灿, 秦志光, 冯朝胜, 等. 面向重复数据消除的备份数据加密方法[J]. 计算机应用，2010，30(7):1763-1766,1781.
WANG C, QIN Z G, FENG C S, *et al*. Deduplication-oriented backup-data encryption method[J]. Journal of Computer Applications, 2010，30(7):1763-1766,1781.

[16] ANDERSON P, ZHANG L. Fast and secure laptop backups with encrypted de-duplication[A]. Proc. of the 24th International Conference on Large Installation System Administration (LISA'10)[C]. 2010. 29-40.

[17] MARK W S, KEVIN G, DARRELL D E, *et al*. Secure data deduplication[A]. Proc. of the 4th ACM International Workshop on Storage security and survivability[C]. New York, USA, 2008. 1-10.

[18] 王珂, 刘川意, 王春露. 基于代理重加密的安全重复数据删除机制的研究[EB/OL]. http://www.paper.edu.cn.2013.
WANG K, LIU C Y, WANG C L. Research on secure de-duplication based on proxy-reencryption[EB/OL]. http://www.paper.edu.cn.2013.

[19] LIU C Y, LIU X J, WAN L. Policy-based de-duplication in secure cloud storage[J]. Trustworthy Computing and Services Communications in Computer and Information Science, 2013, 320:250-262.

[20] KEONG N W, WEN Y G, ZHU H F. Private data deduplication proto-

cols in cloud storage[A]. Proc. of the 27th Annual ACM Symposium on Applied Computing (SAC'12)[C]. New York, USA, 2012.441-446.

[21] DANNY H, BENNY P, ALEXANDRA S P. Side channels in cloud services – the case of deduplication in cloud storage[J]. IEEE Security and Privacy Magazine, special issue of Cloud Security, 2010, 8(6): 40-47.

[22] TAN Y J, JIANG H, FENG D, *et al*. SAM: a semantic-aware multi-tiered source de-duplication framework for cloud backup[A]. 2010 39th International Conference on Parallel Processing[C]. San Diego, CA, 2010.614-623.

[23] ROBERTO D P, ALESSANDRO S. Boosting efficiency and security in proof of ownership for deduplication[A]. Proc. of the 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS'12)[C]. New York, USA, 2012.81-90.

[24] JOÃO B, LUÍS V, PAULO F. Hash challenges: stretching the limits of compare-by-hash in distributed data deduplication[J]. Information Processing Letters, 2012, 112:380-385.

[25] 王丽娜, 任正伟, 余荣威, 等. 一种适于云存储的数据确定性删除方法[J]. 电子学报, 2012,40(2):266-272.
WANG L N, REN Z W, YU R W, *et al*. A data assured deletion approach adapted for cloud storage[J]. Acta Electronica Sinica, 2012, 40(2): 266-272.

[26] ATENIESE G, BURNS R, CURTMOLA R, *et al*. Provable data possession at untrusted stores[A]. Proc of ACM-CCS'07[C]. Alexandria, Virginia, USA, 2007.598-609.

作者简介：

陈越（1965-），男，河南开封人，博士，解放军信息工程大学教授、博士生导师，主要研究方向为网络与信息安全。

李超零（1985-），男，四川眉山人，博士，解放军 78179 部队助理工程师，主要研究方向为云计算与数据安全。

兰巨龙（1962-），男，河北张北人，博士，解放军信息工程大学教授、博士生导师，主要研究方向为宽带信息网络。

金开春（1985-），男，河南商丘人，解放军信息工程大学博士生，主要研究方向为网络与信息安全、可重构安全计算。

王仲辉（1976-），男，四川射洪人，硕士，解放军 78179 部队工程师，主要研究方向为信息安全和计算机应用。