云存储中密文数据的客户端安全去重方案

付安民,宋建业,苏 铓,李 帅
(南京理工大学计算机科学与工程学院,江苏南京 210094)

摘 要: 云存储环境下,客户端数据去重能在本地进行文件重复性检测,有效地节约存储空间和网络带宽.然而,客户端去重仍面临着很多安全挑战.首先,由于将文件哈希值作为重复性检测的证据,攻击者很可能通过一个文件的哈希值获得整个文件;其次,为了保护数据隐私,收敛加密被广泛运用于数据去重方案,但是由于数据本身是可预测的,所以收敛加密仍不可避免地遭受暴力字典攻击.为了解决上述问题,本文首次利用盲签名构造了一个安全的密钥生成协议,通过引入一个密钥服务器,实现了对收敛密钥的二次加密,有效地预防了暴力字典攻击;并进一步提出了一个基于块密钥签名的拥有权证明方法,能够有效预防攻击者通过单一的哈希值来获取文件,并能同时实现对密文文件的文件级和块级去重.同时,安全分析表明本文方案在随机预言模型下是可证明安全的,并能够满足收敛密钥安全、标签一致性和抗暴力字典攻击等更多安全属性.此外,与现有方案相比,实验结果表明本文方案在文件上传和文件去重方面的计算开销相对较小.

关键词: 客户端数据去重;收敛加密;盲签名;拥有权证明
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2017)12-2863-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.12.006

A Security Client-side Deduplication with Encrypted Data in Cloud Storage

FU An-min,SONG Jian-ye,SU Mang,LI Shuai
(*School of Computer Science and Engineering,Nanjing University of Science and Technology,Nanjing,Jiangsu* 210094,*China*)

Abstract: In cloud storage environment,client-side data deduplication can detect duplicated files at local,so as to save storage space and network bandwidth effectively. However,client-side deduplication still faces many security challenges. Firstly,since the file hash value is regarded as the evidence of duplication detection,the attacker is likely to obtain a whole file via a hash of the file. Secondly,to ensure the privacy of data,convergent encryption has been widely used in data deduplication technology,but the data itself is predictable,so that convergent encryption still inevitably suffered from violence dictionary attacks. To solve problems mentioned above,this paper uses blind signature to construct a secure key generation protocol,by introducing a key server to achieve the secondary encryption of keys,which efficiently prevents violence dictionary attacks. Furthermore,we propose a Proof of Ownership method based on block key signature. It can effectively prevent the attacker from obtaining the file through a single hash value and can realize the file-level and block-level deduplication of the encrypted file simultaneously. Meanwhile,the security analysis shows that our scheme can be proved to be secure in the random oracle model and can meet the security properties such as convergence key security,tag consistency and anti-violence dictionary attacks. In addition,compared with the existing schemes,the experimental results show that the computational overhead of our scheme is relatively small in terms of file upload and file deduplication.

Key words: client deduplication;convergent encryption;blind signature;proof of ownership

1 引言

随着云计算技术的飞速发展,大量企业和个人将他们的数据信息外包给云服务提供商(Cloud Service Provider,CSP),这样他们就可以随时随地享受 CSP 所提供的数据存储和计算服务,并能减少数据存储和维护成本.但是,当存储在云端的数据越来越多的时候,将会产生大量的冗余数据.根据 IDC(International Data

收稿日期:2016-09-14,修回日期:2017-02-22,责任编辑:梅志强
基金项目:国家自然科学基金项目(No.61572255,No.61702266);江苏省自然科学基金项目(No.BK20141404,BK20150787);中国博士后科学基金(No.2015M582622);江苏省研究生培养创新工程项目(No.KYLX16_0465)

Corporation)的报告分析[1],到 2020 年世界数据总量将预计达到 40 万亿 GB. 所以,CSP 将面临的一个严峻的挑战就是如何高效地管理持续增加的数据.

数据去重技术使得 CSP 对相同文件只存储一次,从而能够大大节省存储空间. 根据数据的粒度区分,数据去重分为文件级去重和块级去重. 相比文件级去重,块级去重能够提高去重率,极大地节约存储空间. 而根据数据去重执行位置的不同,数据去重又可以分为两种:(1)服务器端去重:用户将数据上传到云服务器,云服务器进行数据去重. 这类方法虽然节约了存储空间,但是将会消耗大量的带宽资源. (2)客户端去重:CSP 在接收到第一个用户上传请求的时候,执行数据文件的存储,对于后面相同文件的上传请求,只需分配一个原始数据文件的权限给请求上传的用户,这样用户就不需要再上传文件到云服务器. 该方法既节约了存储空间,也节约了传输带宽.

然而,客户端的数据去重面临一个很大的安全问题[2,3]:攻击者可能凭借单一的文件 hash 值就可以从云端获得相应文件的下载权限. 为了解决该问题,研究者提出了一种拥有权证明(Proof of Ownership,PoW)方法. 目前 PoW 方法主要有基于 Merkle 哈希树[5]、Bloom 过滤器[6]、签名[16,17]等. 由于基于签名的 PoW 方法具有较高的安全性,其越来越多的被用于安全去重和完整性审计等场景中.

此外,云服务器是诚实而好奇的,它可能试图窃取用户的数据信息. 因此,用户在将数据上传至云服务器之前,通常需要对数据进行加密来实现数据的隐私保护[4]. 然而,当不同用户利用各自的私钥对相同的文件进行加密时,会产生不同的密文,不利于云服务器对文件进行去重. 而基于收敛加密的云存储数据去重方法[7~16]利用数据本身产生的一个 hash 值作为数据的加密密钥,使得相同的数据文件将会产生相同的密文,从而便于云服务器进行去重处理. 因此,目前收敛加密被普遍用于客户端去重中实现密文去重. 但是,收敛加密存在很大的安全漏洞,如暴力字典攻击[13]. 对于可预测的文件,攻击者很容易推导出收敛密钥,并检测文件是否存在于云服务器中. 另外,收敛加密会产生大量的收敛密钥,这会给用户对自身密钥管理造成很大困难,所以用户通常会把密钥托管给云服务器进行管理.

可见,在客户端去重场景下,如何避免攻击者凭借单一的文件 hash 值从云端获得整个文件的下载权限是云存储密文数据安全去重必须解决的首要问题. 另一方面,为实现用户数据的隐私保护,收敛加密被广泛运用于数据去重方案,但收敛加密存在的暴力字典攻击依然是一个开放问题.

本文首次利用盲签名的方法构造了一个更加安全的密钥生成协议,通过引入一个密钥服务器(Key Server,KS),实现了对收敛密钥的二次加密,使得数据加密更加安全,能够有效地预防暴力字典攻击,特别是攻击者无法从盲化后的收敛密钥中破解收敛密钥,保证了收敛密钥的安全性. 在此基础上,进一步提出了一个基于块密钥签名的拥有权证明方法. 用户和云服务器之间必须执行一个挑战/响应协议,才能确定用户是否拥有和云端相同的文件,从而有效地预防了攻击者通过单一的 hash 值获取文件,并能够同时实现对密文数据的文件级和块级去重. 同时,安全分析表明本文方案在随机预言模型下是可证明安全的,并能够满足收敛密钥安全、标签一致性和抗暴力字典攻击等更多安全属性. 此外,与现有方案相比,本文方案在文件上传和文件去重方面的计算开销相对较小.

2 相关工作

为了解决云存储数据安全去重问题,学者们已进行了大量研究与探讨[5~17]. 为了解决攻击者利用文件哈希值获取文件这一问题. Halevi 等人[5]首次提出了拥有权证明(PoW)模型,并给出了基于纠错码和 Merkle 散列树的文件级去重方案,有效地预防了攻击者通过单一 hash 值获取整个文件. 但当数据块很多时,构造的 Merkle 散列树高度很大,不利于计算和验证效率. 为此,Pietro 等人[14]提出了一个高效的拥有权证明方案 s-PoW,该方案通过随机选择一些比特位作为文件拥有权证明的证据,从而保证了计算开销不随数据块数量而变化. 此外,Zheng 等人[17]提出了一个基于签名的方案 POSD,它通过将公开审计和拥有权证明融合在一起,使得生成的数据块签名既可以用于数据完整性验证,也可以用于去重验证. 但该方案计算开销过大.

2013 年,Bellare 等人[8]在欧密会上利用收敛加密的思想给出了一个基于消息锁定加密的云存储数据去重方法,能够对密文数据进行安全去重. 由于基于收敛加密的数据去重方法能很好地实现密钥共享,并且有利于跨用户之间的密文数据去重,自此,大量基于收敛加密的云存储数据去重方法[9~16]被提出.

Lorena 等人[9]提出了 ce-PoW 方案,利用收敛加密的方法实现了隐私保护,通过将收敛加密产生的密文再进行哈希运算,作为块的标签,同时也是 POW 的证据,使得 ce-PoW 具有较小的计算开销. Chen 等人[16]基于收敛加密的方法提出了一个块级去重方案 BL-MLE. BL-MLE 将每一个块的签名也当作每一个块的索引,并且块密钥被封装到块标签中,用户不需要单独去存储每个块的密钥. 但是,BL-MLE 中收敛加密的安全性依然也没有得到很好地处理,同时它的计算开销也比较大. 特别是,Puzio 等人[13]指出了收敛加密存在暴力字

典攻击. 攻击者在知道部分明文的情况下,可以猜测密文和收敛密钥. 为了解决暴力字典攻击问题,他们进一步提出了一个基于文件级的去重方案,通过引入一个中间服务器,并让中间服务器利用自身私钥直接对密文和收敛密钥进行再加密,然后上传至 CSP,但该方法增加了大量通信和存储开销.

3　问题描述

3.1　系统模型

如图 1 所示,本文方案的系统模型包含 3 类实体:云服务提供商(CSP),密钥服务器(KS),用户(Users). 其中,CSP 由主服务器和存储服务器组成,它有足够的存储空间和计算能力,为用户提供数据存储和去重验证服务. KS 作为第三方服务器[3,13,14],它能够与用户进行交互,对盲化后的收敛密钥进行再加密. 通常情况下,KS 必须由可信第三方部署(比如政府部门、数字证书管理机构等). KS 可以是可信第三方部署的单台服务器,也可以是一个服务器设备群. 但为了避免出现单点故障,KS 应采用服务器设备群的方式部署. Users 包括多个普通用户,他们在上传文件到 CSP 之前,CSP 需要检查要上传的文件是否已存在于 CSP 中,若存在,用户不需要上传文件到 CSP;否则,用户上传文件到 CSP.

3.2　实现目标

为了实现在上述安全模型下密文数据的安全去重,本文方案希望实现以下目标:

(1)收敛密钥安全性:攻击者(或 KS)即使获得一些有用的用户信息,也不能够推导出收敛密钥. 我们的方案允许 KS 和 CSP 勾结,但是即使他们相互勾结也不能够恢复出收敛密钥和明文信息.

(2)数据机密性:由于数据的加密密钥包含了数据的收敛密钥和密钥服务器的私钥,我们认为被加密的数据在语义上是安全的,攻击者不能从密文中获得任何有关明文的信息.

(3)数据完整性:有两种类型的完整性检查[19]. 第一种是标签一致性证明,发生在云服务器端,即用户上传正确的标签,却上传与标签不一致的密文. 此时需要 CSP 验证该密文是否和标签来自相同文件. 另一种是消息一致性证明,发生在客户端,即合法的用户从 CSP 下载文件后,不清楚下载的文件是否被破坏或修改,甚至是否是自己所请求的文件. 此时需要用户能够验证该密文是否和标签来自相同的文件.

4　预备知识

存在 G_1 是一个 q 阶的加法循环群,G_2 是一个 q 阶的乘法循环群. P 是 G_1 的一个生成元,存在双线性对映射 $e:G_1\times G_1\to G_2$ 满足如下性质:

(1)双线性:对于所有的 $P,Q,T\in G_1$ 和所有的整数 $a,b\in Z_q$,$e(aP,bQ)=e(P,Q)^{ab}$,$e(P+T,Q)=e(P,Q)e(T,Q)$,$e(P,T+Q)=e(P,T)e(P,Q)$ 均成立.

(2)非退化:对于所有的 $P,Q\in G_1$,满足 $e(P,Q)\neq 1$.

(3)可计算:对于所有的 $P,Q\in G_1$,存在一个有效的算法可以计算出映射 $e(P,Q)$.

基于双线性映射,我们可以得到如下两个定义.

定义 1　DL(Discrete Logarithm)问题:设 P,Q 是群 G_1 的两个生成元,计算出一个整数 $n\in Z_q$,使得 $Q=n\cdot P$ 在计算上是困难的.

定义 2　CDH(Computational Diffe-Hellman)问题:设 P 是群 G_1 的一个生成元,且 $a,b\in Z_q$,已知 $P,aP,bP\in G_1$,计算 $abP\in G_1$ 是困难的.

5　密文数据的安全去重方案

设计的密文数据去重方案包括 9 个算法:密钥生成算法 KeyGen,文件初始化算法 InitFile,数据块初始化算法 InitBlock,数据块验证算法 BlockVerify,文件存储算法 ConTest,生成挑战算法 Challenge,生成证据算法 ProofGen,验证证据算法 ProofVerify,文件解密算法 Decrypt. 在 KeyGen 中,KS 为每一个数据块产生相应的块密钥;在 InitFile 中,用户初始化文件密文、文件标签等基本信息;在 InitBlock 中,用户初始化数据块标签、块签名等基本信息;在 BlockVerify 中,CSP 验证块签名的正确性,从而判断该数据块是否存在于 CSP 中;在 ConTest 中,CSP 存储首次上传的文件块签名、块标签、文件标签、文件密文等基本信息,并验证每一个数据块签名的正确性;Challenge、ProofGen 和 ProofVerify 三个阶段属于 PoW 过程,主要是用户和 CSP 执行一个“挑战-响应”协议,确保 CSP 能够判断用户是否拥有和云端相同的文件;在 Decrypt 中,用户可以根据需求从云端下载相应的文件. 根据文件级和块级去重操作的不同,方案执行的分支会有所不同,算法执行的流程如图 2 所示. 去重方案涉及到的具体的算法如下:

(1)KeyGen

用户为每一个数据块 m_i(总共有 n 个块,$1\leqslant i\leqslant n$)计算其相对应的加密密钥 k_i,密钥生成协议如图 3 所示. 具体过程如下:

① KS 随机选择一个 k-bit 的素数 q 并创建两个 q 阶的椭圆曲线方程 G_1,G_2. P,Q 是 G_1 的两个不同的生成元,并产生一个可接受的线性配对 $e:G_1\times G_1\to G_2$. KS 随机选择一个整数 $x\in Z_q$ 作为私钥,并计算 $P_{\text{pub}}=x\cdot P$. KS 公开系统参数 $\{q,G_1,G_2,e,P,Q,P_{\text{pub}}\}$,并保持 x 私有.

② 用户将文件 M 分成 n 个块:$M=m_1\parallel m_2\parallel\cdots\parallel$

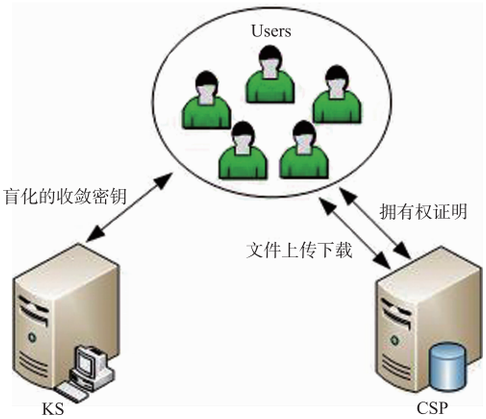

图1 系统模型

m_n,并为每一个数据块 m_i 计算收敛密钥 $h_i = h(m_i)$,其中哈希函数 $h(\cdot):\{0,1\}^* \rightarrow G_1$;然后,用户随机选择 $r \in Z_q$ 作为盲化因子,再计算 $a_i = h_i + r \cdot P$,并将 a_i 上传到 KS.

③ KS 计算 $b_i = a_i \cdot x$,并将 b_i 返回给用户.

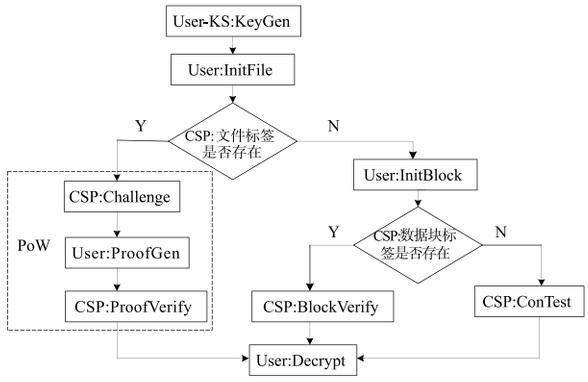

图2 算法执行流程图

④ 用户计算 $d_i = b_i - r \cdot P_{pub}$,并验证 d_i 的正确性:$e(d_i,P) \stackrel{?}{=} e(h(m_i),P_{pub})$. 若正确,则用户令 $k_i = d_i$ 作为每一个块 $m_i(1 \leqslant i \leqslant n)$ 的加密密钥. 等式 $e(d_i,P) \stackrel{?}{=} e(h(m_i),P_{pub})$ 的正确性可以由式(1)得出.

$$\begin{aligned} e(d_i,P) &= e(b_i - r \cdot P_{pub},P) \\ &= e((h_i + r \cdot P) \cdot x - r \cdot x \cdot P,P) \\ &= e(h_i \cdot x,P) = e(h(m_i) \cdot x,P) \\ &= e(h(m_i),x \cdot P) \\ &= e(h(m_i),P_{pub}) \end{aligned} \tag{1}$$

(2) InitFile

用户根据所获得的块密钥,计算每一个块的密文和文件标签,并对密钥进行加密运算. 具体过程如下:

① 使用对称加密的方法为文件 M 的每一个块 m_i $(1 \leqslant i \leqslant n)$,计算块密文 $C_i = \mathrm{Enc}(k_i,m_i)$,并计算 $c_i = H_1(C_i)$,其中哈希函数 $H_1(\cdot):\{0,1\}^* \rightarrow Z_q$.

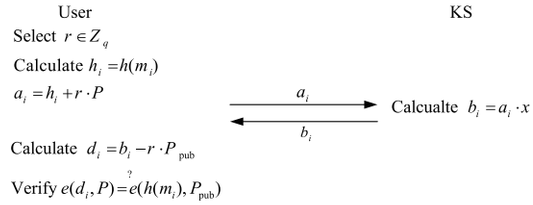

图3 密钥生成协议

② 计算 $T_1 = H_1(C_1 \| C_2 \| \cdots \| C_n)$,$T_2 = H_1(M) \cdot P$,(其中 T_1 用于数据完整性验证,T_2 用于块签名的验证). 并将文件 M 的标签 $T = (T_1,T_2)$ 上传至 CSP.

③ 随机选择自身私钥 sk,并对块密钥 $k_i(1 \leqslant i \leqslant n)$ 进行对称加密,即计算 $C_{key} = \mathrm{Enc}(sk,k_1 \| k_2 \| \cdots \| k_n)$.

(3) InitBlock

当 CSP 检测到文件标签 T 不存在时,用户需要为每一个数据块计算块标签和块签名. 具体过程如下:

① 为每一个文件块 $m_i(1 \leqslant i \leqslant n)$,计算块标签:$\tau_i = H_1(M) \cdot h(m_i)$.

② 计算块签名:$\sigma_i = H_1(M) \cdot (k_i + c_i \cdot Q)$.

(4) BlockVerify

当收到来自用户的块标签 τ_i 和块签名 σ_i,CSP 检测块标签 τ_i 是否存在. 若存在,则执行块去重. CSP 通过验证块签名 σ_i 的正确性,从而判断用户是否拥有和 CSP 相同的数据块. 具体过程如下:

CSP 验证 $\sigma_i = \sigma'_i$ 是否成立,其中 σ'_i 是 CSP 存储的数据块的签名. 若成立,说明用户拥有和 CSP 相同的数据块,CSP 只需为该数据块添加用户的身份 ID_{user};否则,返回一个错误消息.

(5) ConTest

当收到来自用户的块标签 τ_i 和块签名 σ_i,CSP 检测块标签 τ_i 是否存在. 若不存在,需请求用户上传对应的数据块密文 C_i 和密钥的密文 C_{key}. 具体过程如下:

① 用户上传所有的块密文 $C_i(1 \leqslant i \leqslant n)$ 和密钥的密文 C_{key}.

② CSP 验证 $T_1 = H_1(C_1 \| C_2 \| \cdots \| C_n)$ 是否成立. 若成立,说明用户上传的密文和标签是一致的;否则,返回一个错误消息.

③ CSP 计算 $c_i = H_1(C_i)$,并验证 σ_i 的正确性:$e(\sigma_i,P) \stackrel{?}{=} e(\tau_i,P_{pub}) \cdot e(Q,T_2)^{c_i}$. 若成立,存储 T、τ_i、σ_i、C_i 和 C_{key},并添加用户的身份 ID_{user} 到 ID_M,则 $ID_M = ID_M \cup ID_{user}$,其中 ID_M 代表所有拥有文件 M 的用户身份的集合;否则,CSP 返回一个错误消息. 等式 $e(\sigma_i,P) \stackrel{?}{=} e(\tau_i,P_{pub}) \cdot e(Q,T_2)^{c_i}$ 的正确性可以通过式(2)得出.

$$\begin{aligned} e(\sigma_i,P) &= e(H_1(M) \cdot (k_i + c_i \cdot Q),P) \\ &= e(H_1(M) \cdot k_i,P) \cdot e(H_1(M) \cdot c_i \cdot Q,P) \\ &= e(H_1(M) \cdot x \cdot h(m_i),P) \cdot e(Q,H_1(M) \cdot P)^{c_i} \end{aligned}$$

$$=e(H_1(M)\cdot h(m_i),x\cdot P)\cdot e(Q,T_2)^{c_i}$$
$$=e(\tau_i,P_{\text{pub}})\cdot e(Q,T_2)^{c_i} \quad (2)$$

(6) Challenge

当 CSP 检测到文件标签 T 存在时,执行文件去重. CSP 生成挑战信息发送给用户. 具体过程如下:

① 根据文件 M 的总块数 n,从 1 ~ n 中随机生成 c 个数,组成 $I=\{s_1,s_2,\cdots,s_c\}$,且对于 $\forall s_i,s_j\in I(i\neq j)$,$s_i$ 和 s_j 是相互独立的.

② 对于 $\forall i\in I$,随机选择一个 $v_i\in Z_q$,组成挑战信息 chal = $\{i,v_i\}_{i\in I}$,并将 chal 发送给用户.

(7) ProofGen

当收到来自 CSP 发送的挑战信息 chal 后,用户计算响应证据 $P_V=\sum_{i\in I}((k_i+c_i\cdot Q)\cdot H_1(M)\cdot v_i)$,其中 $c_i=H_1(C_i)$,并将 P_V 发送给 CSP.

(8) ProofVerify

当 CSP 收到来自用户的响应证据后,通过验证响应证据是否正确,从而判断用户是否拥有和 CSP 相同的文件. 具体过程如下:

CSP 验证 $\sum_{i\in I}(\sigma_i\cdot v_i)=P_V$ 是否成立. 若成立,说明用户拥有和 CSP 相同的文件,用户只需上传 C_{key},ID_{user}到 CSP. 然后 CSP 存储 C_{key},并添加用户的身份 ID_{user}到 ID_M,则 $\text{ID}_M=\text{ID}_M\cup\text{ID}_{\text{user}}$,其中 ID_M 代表所有拥有文件 M 的用户身份的集合. 否则,CSP 返回一个错误消息.

注:一个完整的拥有权证明过程包括 Challenge, ProofGen, ProofVerify 三个过程,它能够高效地执行一个客户端的文件级去重任务,详细的交互过程如图 4 所示.

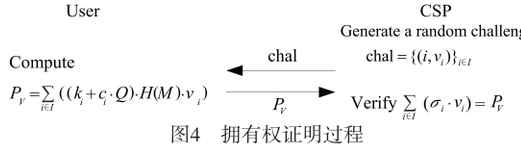

图4　拥有权证明过程

(9) Decrypt

当需要从 CSP 中下载文件 M 时,用户通过文件标签 T 和身份 ID_{user}从 CSP 获取文件 M. 具体过程如下:

① 首先,用户发送文件 M 的标签 T 和身份 ID_{user}到 CSP.

② CSP 收到文件标签 T 和用户身份 ID_{user}后,验证文件标签 T 和用户身份 ID_{user}是否正确. 如果正确,将其对应的 C_{key}和 $C_i(1\leqslant i\leqslant n)$传送给用户;否则,返回一个错误消息.

③ 用户收到 C_{key}和 $C_i(1\leqslant i\leqslant n)$后,验证 $T_1=H_1(C_1\parallel C_2\parallel\cdots\parallel C_n)$是否成立. 如果成立,用户用私钥 sk 解密每一个块密钥 $k_1\parallel k_2\parallel\cdots\parallel k_n=\text{Dec}(\text{sk},C_{\text{key}})$,然后利用块密钥解密每一个数据块 $m_i=\text{Dec}(k_i,C_i)$ $(1\leqslant i\leqslant n)$;否则说明 CSP 返回给了用户不正确的密文.

6　安全分析

6.1　安全证明

定理 1　如果 CDH 问题是成立的,本文密钥生成协议中的盲签名满足不可伪造性.

证明　假设 B 是一个可以生成伪造的(t',ε')算法,即运行时间为 t',优势为 ε'. 那么存在一个(t,ε)算法 A 可以求解 CDH 问题,且满足 $t\leqslant t'+(q_H+q_S+1)\cdot\text{time}_{G_1}$和 $\varepsilon\geqslant\varepsilon'/(e+q_Se)$,其中 B 最多产生 q_H 个哈希询问和 q_S 个签名询问,$e=\lim_{q_S\to\infty}(1+1/q_S)^{q_S}$,$G_1$ 上的点乘数运算时间记为 time_{G_1}.

已知挑战(x,xP,bP),A 利用该挑战来构建一个公钥 $P_{\text{pub}}=xP$,为 B 模拟生成一个盲签名. 在此过程中,B 可以向 A 提交两类询问:哈希询问和签名询问. 设 B 最多产生 q_H 个哈希询问和 q_S 个签名询问. 不失一般性的,我们假设 A 可以提交连续的询问,即对于每一个数据块 m_i 上的签名询问,A 已经产生了一个在对应的 m_i 上哈希询问.

假设 B 触发一个哈希询问,对于该哈希询问,A 投掷硬币,以概率 p_x 显示 0,否则显示 1,概率 p_x 将在下文中确定. 然后,A 随机选择一个 $t\in Z_q$,如果硬币的结果显示为 0 的话,令 $h(m_i)=tP=h_i$,那么 A 返回的结果为 $h_i=tP$,进一步计算得 $a_i=h_i+rP=tP+rP$,其中 $r\in Z_q$;否则令 $h(m_i)=tbP=h_i$,A 返回的结果为 $h_i=tbP$,进一步计算得 $a_i=h_i+rP=tbP+rP$. 最后,A 输出哈希询问的结果给 B. 因为 t 是在哈希询问中随机选取的且是在 Z_q 中均匀分布,tP 和 tbP 在 G_1 中有着相同的均匀分布. 因此,B 无法通过观察哈希询问结果来得知投掷硬币结果.

假设 B 对于数据块 m_i 触发一个签名询问. 按照前面的假设,一个在对应的 m_i 上的哈希询问已经触发了. 如果 A 在对应的哈希询问中投掷的硬币的结果为 0,那么 A 失败并退出. 否则,A 返回 $h_i=h(m_i)=tP$ 以及 $a_i=h_i+rP$. 此时,A 为 m_i 返回签名 $\theta=x\cdot a_i=xtP+xrP$.

最终,B 对于 m^* 输出一个伪造为 θ^*. 根据我们的假设,一个对于 m^* 的哈希询问已经触发. 如果 A 对应的投掷硬币的结果不为 0,那么 A 失败. 否则,哈希询问的结果为 $h_i^*=tbP$,以及 $a_i^*=h_i^*+rP$. 其对应的签名询问的结果为 $\theta^*=x\cdot a_i^*$,其中 t 是由 A 选择得到的. 然后,A 就能够计算 $xbP=(t^{-1}\bmod q)\cdot(\theta^*-\theta)-P_{\text{Pub}}$.

因此,已知(x,xP,bP),如果 B 可以成功计算出一个伪造,则 A 可以输出 xbP. A 成功的概率为 $p_x^{q_S}(1-p_x)\varepsilon'$,该概率在 $p_x=q_S/(q_S+1)$取得最大值,然后,该概率的上界为 $1/e\cdot(1+q_S)$,其中 $e=\lim_{q_S\to\infty}(1+1/q_S)^{q_S}$.

因此,如果在我们的密钥生成协议下,B 是一个可以生成伪造盲签名的(t',ε')算法,那么就存在一个(t,ε)算法 A 可以在时间 t 内,以优势 ε 求解出 CDH 问题,

其中 $t \leqslant t' + (q_H + q_S + 1) \cdot \text{time}_{G_1}$ 和 $\varepsilon \geqslant \varepsilon'/(e + q_S e)$. 显然,这与 CDH 假设相矛盾. 所以,敌手不能够伪造出一个合法的盲签名.

定理 2 如果 CDH 问题是成立的,那么在我们的方案中计算一个伪造的数据块签名是计算上不可行的.

证明 假定存在一个敌手 A 能够在不知道 KS 私钥的前提下伪造一个签名,那么我们就能够构造另一个算法 B 去解决 CDH 问题. 根据定义 2,给定 $(P, aP, bP) \in G_1$,目标是计算出 abP.

在下面的游戏中,我们把哈希函数 h 当作一个随机预言机,每一个数据块仅能询问 h 一次,哈希函数 H_1 是一个抗碰撞的单向哈希函数. 敌手 A 能够有选择的做 h-hash 询问和 Key-extract 询问和 TagGen 询问.

Setup:G_1 和 G_2 是两个 q 阶的循环群,算法 B 令 $P_{\text{pub}} = aP$,然后随机选择 $t \in Z_q$,令 $Q = tP$. 之后返回 $\{q, G_1, G_2, e, P, Q, P_{\text{pub}}\}$ 给敌手 A,$j \in \{1, \cdots, q_h\}$ 是挑战的数据块的索引.

h-hash 询问:当敌手 A 做数据块 m_i 的 h-hash 询问时,如果 $i \neq j$,B 随机选择 $t \in Z_q$,令 $h(m_i) = tP = h_i$,否则,B 随机选择 $t \in Z_q$,令 $h(m_i) = tbP = h_i$,并将 (m_i, h_i, t) 加入 h-list 之中.

Key-extract 询问:当敌手 A 做数据块 m_i 的 Key-extract 询问时. B 按照下述的方式进行响应:

(1)如果 $i \neq j$,B 在表 h-list 中查找 (m_i, h_i, t),返回 $k_i = taP$.

(2)否则,B 拒绝并终止游戏.

TagGen 询问:当敌手 A 做 (m_i, M) 的 TagGen 询问时,B 按照如下的方式进行响应:

(1)B 核对 m_i 是否在列表 h-list 中. 若在其中,B 返回 (m_i, h_i, t),否则,B 做 m_i 的 h-hash 询问.

(2)如果 $i = j$,B 拒绝并终止游戏,否则 B 计算数据块 m_i 的签名 $R = H_1(M)$,$C_i = \text{Enc}(k_i, m_i)$,$c_i = H_1(C_i)$,$B$ 令 $\delta_i = R \cdot t \cdot P_{\text{pub}} + R \cdot c_i \cdot Q$. 然后,将 (m_i, M, C_i, δ_i) 加入列表 Tag-list 中.

(3)将 (m_i, C_i, δ_i) 发给敌手 A.

Output:最终敌手输出一个伪造签名 (m^*, C^*, δ^*),敌手 A 赢得游戏当且仅当满足如下限制条件:

(1)(m^*, M^*) 没有在 TagGen 询问中被查询过;

(2)$m^* = m_j$;

(3)(m^*, C^*, δ^*) 是关于 m^* 的一个有效签名.

如果 (m^*, C^*, δ^*) 是一个有效的签名,我们可以得到下面的等式:$e(\delta^*, P) = e(R^* h_i, P_{\text{pub}}) e(Q, R^* P)^{c^*}$,其中 $R^* = H_1(M^*)$,$c^* = H_1(C^*)$,从而我们可以得出:$abP = (1/tR^*)(\delta^* - QR^* c^*)$. 这就意味着敌手 A 能够以不可忽略的概率 ε' 来解决 CDH 问题,这与 CDH 假设相矛盾. 所以,敌手 A 不能伪造一个合法的数据块签名.

6.2 安全属性分析

(1)收敛密钥安全性:在密钥生成协议中,用户首先选择一个随机数 $r \in Z_q$ 作为盲化因子,对收敛密钥进行盲化处理,并将盲化后的值 $a_i = h(m_i) + r \cdot P$ 发送给 KS,KS 再对盲化后的收敛密钥进行加密. 在整个传输过程中,收敛密钥始终是盲化的,由于 $h(\cdot)$ 是单向散列的,并且 $r \in Z_q$ 是随机选取的,所以攻击者(甚至是 KS)即使获得了交互过程中的一些信息,也不能推导出收敛密钥和数据块 m_i.

(2)数据机密性:在本文方案中,数据是以加密形式存储在 CSP. 由于数据块的加密密钥只在用户端生成,且此加密密钥中包含了数据块的收敛密钥和 KS 的私钥,特别是收敛密钥安全性是得到有效保护的,所以攻击者很难获得或破解此数据块的加密密钥,也就不可能获取数据明文信息.

(3)标签一致性:我们基于文件的密文产生文件标签 T_1,当用户上传 T_1 和每一个块密文 C_i $(1 \leqslant i \leqslant n)$ 后,云服务器能够利用收到的文件标签和密文来验证 $T_1 = H_1(C_1 \parallel C_2 \parallel \cdots \parallel C_n)$ 是否成立,从而有效地预防用户上传与标签不一致的文件,确保数据标签的一致性.

(4)消息一致性:当用户从 CSP 成功下载文件后,能够利用下载的密文和自身所拥有的文件标签验证 $T_1 = H_1(C_1 \parallel C_2 \parallel \cdots \parallel C_n)$ 是否成立,以避免 CSP 篡改文件,从而保证消息的一致性.

(5)抗暴力字典攻击:本文方案中,数据块的加密密钥采用盲签名的方法对收敛密钥进行二次加密,使得攻击者不可能推导出收敛密钥和数据块的加密密钥. 由于加密密钥中包含 KS 的私钥,而 KS 的私钥是随机选取的,所以加密的密文存在更大的无关联性. 这样攻击者在对密文发起暴力字典攻击的过程中,更加难以猜测明文,从而能有力地预防暴力字典攻击.

为了更好的阐述本文方案的安全属性,我们与文献[16,17]中基于签名的去重方案进行了对比分析,具体如表 1 所示,从表中我们可以看出,本文方案能够支持更多安全属性.

表 1 功能与安全属性对比

Features	BL-MLE[16]	POSD[17]	Our scheme
拥有权证明	Y	Y	Y
文件级去重	Y	Y	Y
块级去重	Y	Y	Y
数据机密性	Y	N	Y
收敛密钥安全性	N	N	Y
标签一致性	N	N	Y
消息一致性	N	N	Y
抗暴力字典攻击	N	N	Y

7　性能分析

7.1　检测成功率

在拥有权证明过程中,CSP 随机发送 c 个挑战块给用户,用户只需计算这 c 个块的聚合响应值发送给 CSP. 这其中可能存在一些被损坏(修改或丢失)的数据块没有被检测到,所以我们需要选择合适的挑战块数 c,从而提升检测成功率.

假设用户声称他拥有 CSP 中已存在的原始文件 M 的全部块数 n 个数据块,但实际上其中 t 个数据块已经被损坏. 在拥有权证明的过程中,CSP 随机发送 c 个挑战块,X 是被检测到损坏块的块数,P_X 是至少一个损坏块被检测到的概率,$P_X = P\{X \geqslant 1\} = 1 - P\{X = 0\} = 1 - \frac{n-t}{n} \cdot \frac{n-t-1}{n-1} \cdot \ldots \cdot \frac{n-t-c+1}{n-c+1}$.

因为 $\frac{n-t-i}{n-i} > \frac{n-t-i-1}{n-i-1}$,所以 $1-(\frac{n-t}{n})^c \leqslant P_X \leqslant 1-(\frac{n-t-c+1}{n-c+1})^c$,即 $P_X \geqslant 1-(1-\frac{t}{n})^c$.

如果 $t = n \times 1\%$,$P_X \geqslant 1-0.99^c$,当 $c \geq 460$ 时,$P_X \geqslant 1-0.99^c \geqslant 1-0.99^{460} \approx 0.99$. 这就意味着在损坏率(修改或丢失)为 1% 的情况下,若挑战块数达到 460,检测成功的概率将大于 99%. 为更好地分析检测成功率与挑战数的关系,我们分别给出了在损坏率为 1%、5% 和 10% 的情况下,检测成功率 P_X 随挑战块数 c 的变化情况,具体如图 5 所示. 从图中我们可以看出,挑战块数越多,检测成功率越高,特别是在损坏率为 5% 和 10% 的情况下,要达到 99% 的检测成功率,分别至少需要挑战块数为 90,22. 此外,当损坏率越大的时候,所需挑战的块数也就越少.

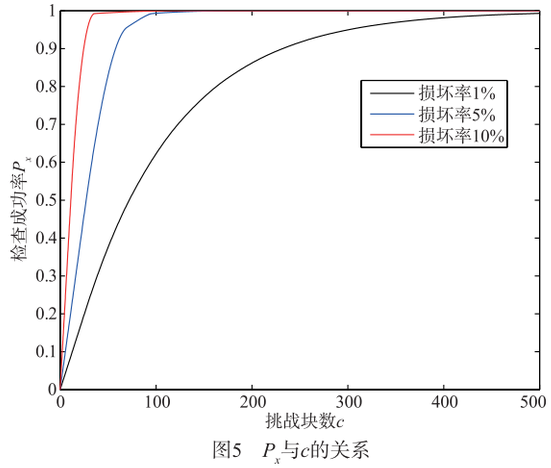

图5　P_x与c的关系

7.2　性能评估

我们选取了同样基于签名的文献[16,17]方案和本文方案进行对比分析. 在我们的性能评估中,本文主要从文件上传和文件去重两个方面来分析通信和计算开销. 需要说明的是,由于本文方案数据块去重验证 BlockVerify 算法只涉及到一个简单的块签名等于验证运算,开销很小,故没有考虑. 此外,本文文件解密 Decrypt 算法与文献[16]方案均直接采用传统的对称加解密算法,只是密钥的选取有差别,所以本文没有对他们单独进行对比分析.

7.2.1　通信开销

在数据去重方案中主要存在两种操作:文件上传和文件去重. 在文件上传过程中,用户需要上传文件标签 T、块标签 τ_i、块签名 σ_i、密文 C_i 和密钥的密文 C_{key},因此上传文件的通信开销是 $2|q| + 2(n+1)|G| + n|w|$;在文件去重过程中,用户和 CSP 需要传输挑战消息 chal 和一个聚合响应值 P_V 用于完成拥有权证明,因此文件去重的通信开销是 $c|\text{id}| + c|q| + |G|$. 其中,$n$ 是块的总数,m 是每个块的扇区数,c 是挑战块数,$|\text{id}|$ 是数据块标识的大小,$|G|$ 是 G_1 中元素的大小,$|T|$ 是 G_T 中元素的大小,$|q|$ 是 Z_q 中元素的大小,$|w|$ 是数据块的密文 C_i 的大小.

表 2　通信开销对比

	BL-MLE[16]	POSD[17]	Our scheme
文件上传	$(n+1)\|G\| + n\|T\| + n\|w\|$	$3n\|q\| + n\|G\| + n\|w\|$	$2\|q\| + (2n+1)\|G\| + n\|w\|$
文件去重	$c\|\text{id}\| + c\|q\| + \|G\|$	$c\|\text{id}\| + (c+m)\|q\|$	$c\|\text{id}\| + c\|q\| + \|G\|$

表 2 给出了本文方案与文献[16,17]方案的通信开销对比. 从表 2 可以看出,文件上传的通信开销与总块数 n 存在线性关系,而文件去重的通信开销与挑战块数 c 存在线性关系.

7.2.2　计算开销

假设用户想要上传文件 M,首先将文件分成 n 个块,并将每一个块分为 m 个扇区. 在进行拥有权证明时,CSP 随机请求挑战 c 个块. 其中 $n >> m$,$m \geqslant 1$,$c \geqslant 1$. 为便于描述各方案的计算开销,表 3 给出了所涉及的一些基本密码学操作.

表 3　密码学操作的符号及解释

符号	解释
Hash_{G_1}	将消息 m 进行 hash 运算映射到 G_1 域的一个值
Hash_{Z_q}	将消息 m 进行 hash 运算映射到 Z_q 域的一个值
$\text{Mul}_{G_1 * Z_q}$	将 G_1 域的一个值和 Z_q 域的一个值进行乘运算
$\text{Mul}_{G_1 * G_1}$	将 G_1 域的两个值进行乘运算
$\text{Mul}_{Z_q * Z_q}$	将 Z_q 域的两个值进行乘运算
$\text{Exp}_{G_1 * Z_q}$	G_1 域和 Z_q 域的值分别为底数和指数,进行幂运算
$\text{Exp}_{Z_q * Z_q}$	Z_q 域的值分别为底数和指数,进行幂运算
Pair	执行 $e(u,g)$ 线性对运算,其中 $u,g \in G_1$

表4 给出了本文方案与文献[16,17]方案的计算开销对比情况. 我们分别从文件上传和文件去重两方面,统计了不同阶段的计算开销. 值得注意的是,在文件去重的过程中,文献[17]方案中用户每次响应挑战时,虽然用户只需计算部分响应值,而 CSP 却需要执行大量的运算去完成验证.

表4 计算开销对比

		BL-MLE[16]	POSD[17]	Our scheme
文件上传	块签名生成	$n((m+1)\mathrm{Exp}_{G_1*Z_q}+m\mathrm{Mul}_{G_1*G_1})$	$2n\mathrm{Exp}_{Z_q*Z_q}+2(n+m-1)\mathrm{Mul}_{Z_q*Z_q}$ $+n\mathrm{Hash}_{G_1}+\mathrm{Exp}_{G_1*Z_q}+n\mathrm{Mul}_{G_1*G_1}$	$2n\mathrm{Mul}_{G_1*Z_q}+\mathrm{Hash}_{Z_q}$
	块签名验证	$n(3\mathrm{Pair}+\mathrm{Mul}_{G_1*G_1}+m\mathrm{Exp}_{G_1*Z_q})$	$n(2\mathrm{Pair}+\mathrm{Hash}_{G_1}+\mathrm{Exp}_{G_1*Z_q}+\mathrm{Mul}_{Z_q*Z_q})$	$n(3\mathrm{Pair}+\mathrm{Mul}_{G_1*G_1}+\mathrm{Exp}_{G_1*Z_q}+\mathrm{Hash}_{Z_q})$
文件去重	响应挑战	$c[(m+2)\mathrm{Exp}_{G_1*Z_q}+m\mathrm{Mul}_{G_1*G_1}]$	$m(c-1)\mathrm{Mul}_{Z_q*Z_q}$	$c(2\mathrm{Mul}_{G_1*Z_q}+\mathrm{Mul}_{Z_q*Z_q})+\mathrm{Hash}_{Z_q}$
	验证挑战	$c\mathrm{Exp}_{G_1*Z_q}+(c-1)\mathrm{Mul}_{G_1*G_1}$	$(2c+1)\mathrm{Exp}_{G_1*Z_q}+(m+2)\mathrm{Exp}_{Z_q*Z_q}$ $+(m+c)\mathrm{Mul}_{Z_q*Z_q}+2Pair+(2c-1)\mathrm{Mul}_{G_1*G_1}$	$c\mathrm{Mul}_{G_1*Z_q}$

7.3 实验结果

为了具体评估方案的性能,我们使用 JPBC[20] 库,选用基域的大小为 159bit 和嵌入度为6 的椭圆曲线,分别对本文方案和文献[16,17]方案中的算法进行了实现. 所有的算法都运行在 Intel Core i7 - 2600 CPU @ 3.4GHz,内存为4GB,硬盘为 850GB 的主机上.

表5 通信开销结果对比

	BL-MLE[16]	POSD[17]	Our scheme
文件上传(Kb)	75536.313	78036	75536.625
文件去重(Kb)	94.648	95.898	94.648

在实验中,我们设置 G_1,G_T 中元素的大小均为 320b,即$|G|=320$b,$|T|=320$b,Z_q 中元素的大小为 160b,即$|q|=160$b,数据块标识大小$|\mathrm{id}|=50$b,每一个数据块大小为 size = 4Kb,我们假设每一个密文块大小也为 4Kb,即$|w|=4$Kb,每个数据块的扇区数 $m=10$,使用随机抽样方法[18],在丢失率 1% 的情况下,CSP 选择 460 个数据块,就能以 99% 以上的概率检测出被损坏的数据块,为此我们设置 $c=460$,并以 64Mb 的文件大小为例,统计了其通信开销的实验结果,如表 5 所示. 从表 5 我们可以看出,本文方案和文献[16,17]方案的通信开销相差不大. 在文件上传的过程中,我们方案比 BL-MLE[16] 多了 0.312Kb,这是由于本文方案为了解决标签一致性的问题,增加了一个额外的标签 T_1. 而且相对于 64Mb 文件而言,增加 0.312Kb 是非常小的,在可接受的范围内. 在文件去重过程中,本文方案和文献[16]方案是最小的,它们只需传输挑战消息 chal 和响应值 P_V 这两个值.

在计算开销方面,由于本文方案利用盲签名进行二次加密以生成块密钥,因此,我们首先统计了每一个块密钥的生成时间开销大概为 0.078 秒,并随着块数的增加,密钥生成时间也将呈线性变化. 但引入盲签名进行二次加密以生成块密钥增加的时间开销相比文件上传和文件去重等方面很小. 图6、图7 分别给出了不同方案中块签名生成时间和块签名验证时间的对比情况. 从图可以看出,块签名生成时间和块签名验证时间均随文件大小增加而增加. 这是因为随着文件大小的增加,而数据块大小不变,这样产生的数据块的个数将会越来越多,导致块标签和块签名的计算量也越来越多. 从图6、图7 和表4 中,我们可以看出 BL-MLE 方案中的块签名生成时间、块签名验证时间以及文件上传总时间都非常大,是因为 BL-MLE 方案需要运行大量开销较大的幂运算和双线性对运算. 而本文方案时间开销最小,这是因为本文方案将每个数据块看成单独的个体,没有再分扇区,并且我们在生成块签名的过程中,成功避免了一些开销较大的幂运算.

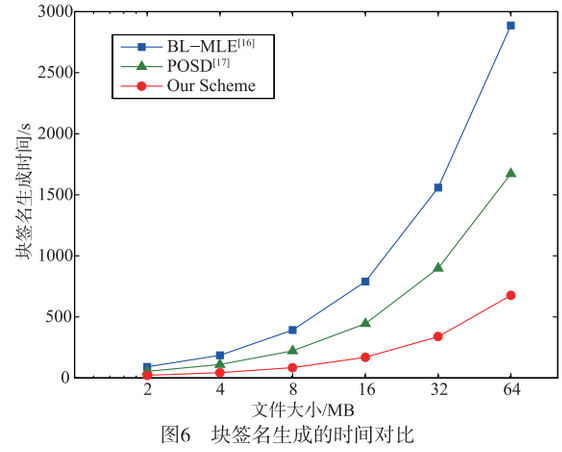

图6 块签名生成的时间对比

为进一步分析文件去重开销跟挑战块数 c 的关系,我们对响应挑战时间、验证挑战时间以及去重总时间跟挑战块数 c 的关系进行了实验验证. 图8 ~ 10 分别给

出了不同方案中,文件去重过程中响应挑战时间、验证挑战时间以及去重总时间对比情况. 从图 8 和表 4 可以看出,POSD[17]方案中的响应挑战时间是最小的,这是因为在 POSD 中用户响应挑战时,只做了计算量较小的 $\mathrm{Mul}_{Z_q^* Z_q}$ 运算. 但从图 9 可以看出,POSD 方案在验证挑战时,所消耗的计算开销远远大于本文方案中验证挑战的时间开销,这是因为 POSD 需要计算大量的辅助值来完成验证,而我们的方案和 BL-MLE 只需聚合被挑战的 c 个块的签名,然后和聚合响应值 P_V 进行比较即可. 而从图 10 可以看出,本文方案文件去重总时间开销依然最小.

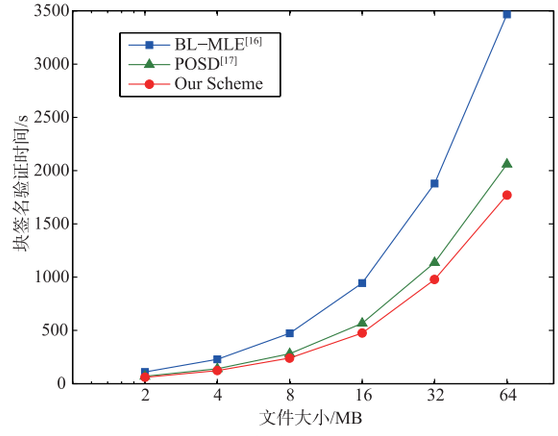

图7　块签名验证的时间对比

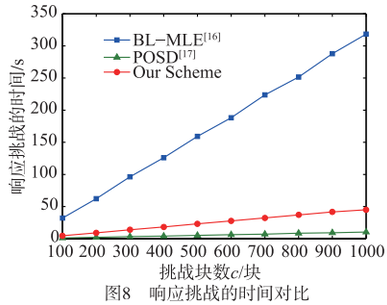

图8　响应挑战的时间对比

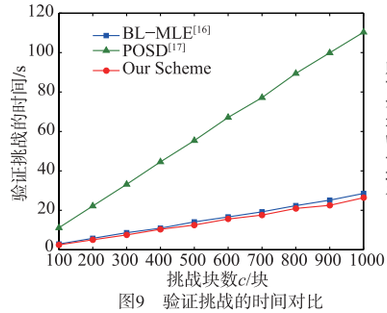

图9　验证挑战的时间对比

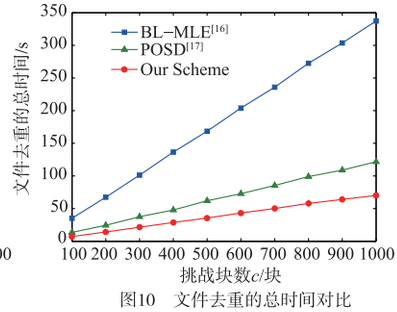

图10　文件去重的总时间对比

8　结论

本文借助盲签名的方法构造了一个安全有效的密钥生成协议,实现了对收敛密钥的二次加密,使得数据加密更加安全,能够有效地预防暴力字典攻击. 特别是,我们进一步构造出了一个基于块密钥签名的拥有权证明方法,能够有效地预防攻击者通过单一的哈希值来获取文件,并且它能够同时实现对密文数据的文件级和块级去重. 同时,安全分析表明本文方案在随机预言模型下是可证明安全的,能够满足收敛密钥安全、标签一致性和抗暴力字典攻击等更多安全属性. 此外,与现有方案相比,实验结果表明本文方案在文件上传和文件去重方面的计算开销相对较小.

参考文献

[1] Gantz J, Reinsel D. The Digital Universe in 2020: Big Data, Bigger Digital Shadows, Biggest Growth in the Far East, Dec. 2012 [DB/OL]. http://www.emc.com/collatera/analystreports/idc-the-digital-universe-in-2020.pdf, 2016.5.

[2] 俞能海,郝卓,徐甲甲,等. 云安全研究进展综述[J]. 电子学报,2013,41(2):371-381.
Yu NH, Hao Z, Xu JJ, et al. Review of cloud computing security [J]. Acta Electronica Sinica, 2013, 41(2): 371-381. (in Chinese)

[3] 熊金波,张媛媛,李凤华,等. 云环境中数据安全去重研究进展[J]. 通信学报,2016,37(11):1-12
Xiong JB, Zhang YY, Li FH, et al. Research progress on secure data deduplication in cloud[J]. Journal on Communications, 2016, 37(11): 1-12 (in Chinese)

[4] 付安民,秦宁元,宋建业,等. 云端多管理者群组共享数据中具有隐私保护的公开审计方案[J]. 计算机研究与发展,2015,52(10):2353-2362.
Fu AM, Qin NY, Song JY, et al. Privacy-preserving public auditing for multiple managers shared data in the cloud [J]. Journal of Computer Research and Development, 2015, 52(10): 2353-2562 (in Chinese)

[5] Halevi S, Harnik D, Pinkas B, et al. Proofs of ownership in remote storage systems[A]. Proceedings of the 18th ACM Conference on Computer and Communications Security [C]. Chicago, Illinois, USA. NewYorK: ACM, 2011. 491-500.

[6] Blasco J, Di Pietro R, Orfila A, et al. A tunable proof of ownership scheme for deduplication using bloom filters [A]. 2014 IEEE Conference on Communications and Network Security (CNS) [C]. San Francisco, CA, USA. USA: IEEE, 2014. 481-489.

[7] Douceur JR, Adya A, Bolosky WJ, et al. Reclaiming space from duplicate files in a serverless distributed file system

[A]. Proceedings of the 22nd International Conference on Distributed Computing Systems [C]. Vienna, Austria. USA:IEEE,2002. 617 – 624.

[8] Bellare M, Keelveedhi S, Ristenpart T. Message-locked encryption and secure deduplication[A]. Annual international conference on the theory and applications of cryptographic techniques [C]. Springer, Berlin, Heidelberg, Germany: ACM,2013. 296 – 312.

[9] González-Manzano L, Orfila A. An efficient confidentiality-preserving proof of ownership for deduplication [J]. Journal of Network and Computer Applications,2015,50: 49 – 59.

[10] Li J, Chen XF, Li MQ, et al. Secure deduplication with efficient and reliable convergent key management[J]. IEEE Trans. on Parallel and Distributed Systems,2014,25(6): 1615 – 1625.

[11] 杨超,张俊伟,董学文,等. 云存储加密数据去重删除所有权证明方法[J]. 计算机研究与发展,2015,52(1): 248 – 258
Yang C, Zhang JW, Dong XW, et al. Proving method of ownership of encrypted files in cloud de-duplication deletion[J]. Journal of Computer Research and Development, 2015,52(1):248 – 258 (in Chinese)

[12] Yan F, Tan YA, Zhang QX, et al. An effective RAID data layout for object-based de-duplication backup system[J]. Chinese Journal of Electronics,2016,25(5):832 – 840.

[13] Puzio P, Molva R, Onen M, et al. ClouDedup: secure deduplication with encrypted data for cloud storage [A]. IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom) [C]. NewTork, USA:IEEE,2013. 363 – 370.

[14] Di Pietro R, Sorniotti A. Boosting efficiency and security in proof of ownership for deduplication[A]. Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security [C]. Seoul, Korea. New-York:ACM,2012. 81 – 82.

[15] Li J, Li YK, Chen XF, et al. A hybrid cloud approach for secure authorized deduplication[J]. IEEE Trans on Parallel and Distributed Systems,2015,26(5):1206 – 1216.

[16] Chen RM, Mu Y, Yang GM, et al. BL-MLE: block-level message-locked encryption for secure large file deduplication[J]. IEEE Trans on Information Forensics and Security,2015,10(12):2643 – 2652.

[17] Zheng QJ, Xu SH . Secure and efficient proof of storage with deduplication[A]. Proceedings of the second ACM Conference on Data and Application Security and Privacy [C]. New York, NY, USA: ACM,2012. 1 – 12.

[18] 王宏远,祝烈煌,李龙一佳. 云存储中支持数据去重的群组数据持有性证明[J]. 软件学报,2016,27(6):1417 – 1431.
Wang HY, Zhu LH, Li LJY. Group provable data possession with deduplication in cloud storage [J]. Journal of Software,2016,27(6):1417 – 1431. (in Chinese)

[19] Li J, Chen XF, Huang XY, et al. Secure distributed deduplication systems with improved reliability [J]. IEEE Trans on Computers,2015,64(12):3569 – 3579.

[20] Stanford University. The Java Pairing Based Cryptography (JPBC) Library Benchmark,2013 [DB/OL]. http://gas.dia. unisa. it/projects/jpbc/ benchmark. html,2016. 2.

作者简介

付安民 男,1981 年 5 月出生,湖北通城人,副教授,博士生导师,研究方向为密码学与网络安全. E-mail:fam_0522@163. com

宋建业 男,1991 年 8 月出生,江苏淮安人,硕士生,研究方向为云存储安全.

苏 铓 女,1987 年 12 月出生,内蒙赤峰人,博士,讲师,研究方向为云计算安全.

李 帅 男,1992 年 8 月出生,山东青岛人,硕士生,研究方向为云计算安全.