Journal of Computer Applications
计算机应用,2015,35(12):3413 -3418
ISSN 1001-9081
CODEN JYIIDU
2015-12-10
http://www.joca.cn

文章编号:1001-9081(2015)12-3413-06
doi:10.11772/j.issn.1001-9081.2015.12.3413

具有细粒度访问控制和低存储空间开销的云存储系统

印凯泽*,汪海航
(同济大学 电子与信息工程学院,上海 201804)
(*通信作者电子邮箱 cogito_yin@163.com)

摘　要:针对目前公有云存储系统中存在的数据机密性和系统性能问题,提出了一个安全高效的方案,并将其应用于基于密文策略属性基加密(CP-ABE)的具有细粒度访问控制的密码学的云存储系统中。在这个方案中,原始的数据首先会经过一个(k,n)算法分割成小块,然后随机选择其中部分小块进行加密,最后发布到云上,且只保存一份副本。该方案能够提升用户撤销操作的性能和降低存储空间的开销,同时安全性分析也证明了这个系统在计算上是安全的。通过分析对比,实验结果表明:该方案优化了用户撤销,减少了数据拥有者对数据管理的时间,由于只需要保存一份数据副本,因此有效地减少了数据的存储空间。该方案实现了公有云存储中敏感数据的安全共享和高效存储。

关键词:云存储;访问控制;密文策略属性基加密;(k,n)算法;数据机密性
中图分类号:TP311.5　　**文献标志码**:A

Cloud storage system with fine-grained access control and low storage space overhead

YIN Kaize*, WANG Haihang
(*College of Electronics and Information Engineering, Tongji University, Shanghai* 201804, *China*)

Abstract: Concerning the data's confidentiality when stored in public cloud storage system and the system's performance, a secure and efficient scheme was proposed and applied in the cloud storage system of cryptography with cryptographic fine-grained access control, which was based on Ciphertext-Policy Attribute-Based Encryption (CP-ABE). In the proposed scheme, the original data were firstly divided into a number of slices by the (k, n) algorithm. Then some of slices were randomly chosen to encrypt. At last, the encrypted slices were published to the cloud storage, and only one copy of these slices was stored. The proposed scheme was proved that it could improve the performance of the user's cancel operation and reduce the cost of the storage space. At the same time, the system was also proved to be safe on calculation by the analysis of the security. By contrast, the experimental results show that, the data management time for the data owner is decreased obviously through optimizing the user revocation phase. The data storage cost is also decreased because of only storing one copy of data. The proposed scheme achieves secure sharing and efficient storage of the sensitive data in the public cloud storage.

Key words: cloud storage; access control; Ciphertext-Policy Attribute-Based Encryption (CP-ABE); (k, n) algorithm; data confidentiality

0　引言

近来,越来越多的个人用户和企业开始使用公有的云存储[1]服务来进行数据的存储和共享。国际上比较有名的云存储系统有谷歌文件系统(Google File System, GFS)[2]、Hadoop分布式文件系统(Hadoop Distributed File System, HDFS)[3]和亚马逊简单存储服务(Simple Storage Service, S3)[4]等。然而这些系统中存在的一些问题可能将阻碍云存储的发展。从用户的角度来看,他们担心的是安全问题,比如数据机密性、信息泄露和安全共享等问题。一旦他们将数据上传到云上,就失去了对数据的直接控制而只能够不情愿地去相信云服务提供商(Cloud Service Provider, CSP)。然而,这些CSP通常是不可信的,因此用户的数据就有可能被泄露。再从CSP的角度来看,他们部署的云存储系统是一个极其复杂的系统,拥有成百上千的组件。节点失效和数据丢失在这样的系统中是经常发生的,为了保证系统的稳定性和可用性,多重备份方法成为了最常用的方法。例如在GFS、HDFS和亚马逊S3中文件备份的副本数默认值为3。多重备份确实能够提供系统的可靠性,但是备份的副本却给系统带来了高昂的存储空间开销。因此,目前的云存储系统中还依然存在安全和空间开销这两个主要的问题。

针对安全问题,用户需要对数据进行加密,同时在云存储系统中实现访问控制。文献[5]提出了这样一个安全的云存储框架,叫作密码学的云存储系统。在这个框架中,数据拥有者(Data Owner, DO)首先加密数据,然后将密钥一个一个地分发给所有授权的数据使用者(Data User, DU)。这个方案可以保证数据的机密性,同时访问控制可以防止未授权用户和不可信的CSP访问数据。不管怎样,这个方案还是存在一

收稿日期:2015-05-15;**修回日期**:2015-07-13。
作者简介:印凯泽(1989 -),男,浙江宁波人,博士研究生,CCF会员,主要研究方向:云计算、访问控制;　汪海航(1965 -),男,浙江奉化人,教授,博士生导师,博士,主要研究方向:信息安全、网络与分布式计算。

些问题。首先,一个一个分发密钥效率低下,尤其是在文件和 DU 数量很大的情况下;其次,访问策略撤销的代价很高,因为 DO 需要检索到数据,再重新加密重新发布到云上。其中第一个问题可以用一个叫作密文策略的属性基加密(Ciphertext-Policy Attribute-Based Encryption, CP-ABE)算法[6]的公钥算法来解决。在一个基于 CP-ABE 的访问控制系统中,每一个 DU 都会分配一个属性集 S, 对称密钥 k 会被 DO 的公开密钥(Public Key, PK)和访问结构树 T 加密,然后密文 $E'_T(k)$ 和 $E_k(F)$ 会被发布出去。当且仅当 DU 的属性集 S 满足 T 时,才可以解密 $E'_T(k)$,最后得到明文 F。因此,DO 可以通过指定访问策略来分配密钥而不需要一个一个地分发。假设一个属性集为 S 的 DU 已经被授权可以访问文件 F, 现在 DU 被撤销了。那么一个完整的安全撤销过程就会按如下步骤执行:首先,DO 检索得到密文 $E_k(F)$ 解密得到 F; 接着,DO 选择另一个随机密钥 k',用 k' 加密 F 得到密文 $E_{k'}(F)$; 然后,DO 计算出一个 S 不能满足的新的访问结构树 T' 和密文 $E'_T(k')$; 最后,DO 将 $E_{k'}(F)$ 和 $E'_T(k')$ 发布出去,删除原来的 $E_k(F)$ 和 $E'_T(k)$。然而第二个问题在基于 CP-ABE 的访问控制系统中的用户撤销阶段依然存在。因此本文会提出一个具有高效用户撤销的基于 CP-ABE 的密码学的云存储系统。

在目前的存储系统中,安全和性能总是对立的。当引入安全性的时候也就意味着将花费更多的时间。但是,可以通过降低存储空间来对系统的安全性和整体的开销作一个权衡。因此在本文的云存储系统中,通过只保留 1 份数据副本来降低存储的开销,同时可以保证跟多重备份一样的可靠性。

首先,将研究如何降低基于 CP-ABE 的云存储系统中用户撤销的开销。可以知道撤销的开销是跟数据大小有关系的,因此如果撤销操作只在整个数据块的一小块或几小块上进行,那么撤销的开销就会大大地下降。根据这个思想,可以将原始数据 F 分割成 n 小块, 这样 CP-ABE 算法只在某几个小块上进行。同时为了保证其余小块的机密性,分割的操作可以在一个 $(k,n)(n \geqslant k)$ 秘密共享方案[7]下进行。根据秘密共享方案的特性,只要 $(n-k+1)$ 个小块通过 CP-ABE 算法加密,那么任何人都不可能在不经过解密的情况下从任意的 k 个小块中恢复出数据 F,因为选择任意 k 个小块中至少有 1 个是被加密的。同理,只要用户可以解密这些加密的小块,那么他/她就可以从任意的 k 个小块中恢复数据 F。所以只需要保存 1 份这些小块的副本就可以达到与多重备份相同的错误容忍度,只要 (k,n,R) 满足公式 $R-1=n-k$ 即可。比如多重备份副本数 $R=3$, 那么同一时间最多可以允许两个副本失效。只要调整秘密共享方案为(3,5),就也可以允许同一时间最多两个副本失效。根据这些技术,可以构建一个具有细粒度访问控制和低存储空间开销的安全高效的云存储系统。

1 CP-ABE 算法和秘密共享方案

1.1 CP-ABE 算法

Goyal 等[8]提出了属性基加密(Attribute-Based Encryption, ABE)算法作为一种在加密条件下访问控制的新的方法。后来,ABE 分成了密钥策略属性基加密(Key-Policy Attribute-Based Encryption, KP-ABE)和 CP-ABE。在 KP-ABE 系统中,访问策略是由 DU 制定的。相反地,在 CP-ABE 系统中,访问策略是由 DO 制定的,因此 CP-ABE 算法更适合应用于访问控制应用中。

CP-ABE 算法主要包括如下四个步骤:

1) Setup: 产生一个主密钥(Main Key, MK)和公开密钥 PK。

2) C = Encrypt(PK,F,T):将数据 F 用 PK 和访问结构树 T 加密得到密文 C。

3) SK = KeyGen(MK,S):输入主密钥 MK 和属性集 S,输出一个私有密钥(Secret Key, SK)。

4) Decrypt(C,SK):只要 SK 中包含的属性集 S 满足访问结构树 T,就可以通过 SK 解密密文 C 得到数据 F;反之则不行。

1.2 秘密共享方案

秘密共享方案(Secret Sharing Scheme, SSS)[7],也叫作 (k,n) 门限方案,可以按如下方式将秘密 S 分割成 n 个数据块 $S_1,S_2,\cdots,S_n$:

1) 知道任意 k 个或更多的 S_i 数据块时可以轻松地计算出 S;

2) 知道任意 $k-1$ 个或更少的 S_i 数据块时就完全不能得出 S。

有一些秘密共享方案被证明是在信息论理论上安全的,叫作无条件的安全秘密共享方案。但是这些方案有一个缺点,就是结果数据的大小是原始数据的 n 倍。例如,原始数据块很大,是 1 GB,分片的数量是 10,那么结果数据大小就是 10 GB。毫无疑问,在实际的云存储应用中,DO 和 CSP 都不会接受这个严重的额外的存储开销。因此,必须放弃这个无条件的安全需求,而采用其他可替代的方法来增加秘密共享方案的效率同时保证足够的安全性。

本文将采用一个叫作带里德-所罗门码的全或无转换(All-Or-Nothing Transform-Reed Solomon, AONT-RS)方法[9],这个方法就是在用信息分散算法(Information Dispersal Algorithm, IDA)[10]对数据进行分片之前使用全或无转换(All-Or-Nothing Transform, AONT)方法[11]进行预处理。AONT 方案可以看作是一个 $(n+1,n+1)$ 的门限方案,它将一个文件编码分割成 $n+1$ 份,可以保证任意小于门限的分片数量都不能够解密数据。IDA 算法是一个数据分片算法,类似于 SSS 同样会配置一个门限,但是它的分片结果数据不会随着(碎片数量/门限)因子增大。例如,门限方案是(10,15),那么总的分片大小就是原始数据的(15/10)倍。IDA 和 AONT 的详细介绍可以分别在文献[10]和文献[11]中查询。

2 安全高效的云存储系统

2.1 系统框架

本文提出的安全高效的云存储系统是基于 CP-ABE 的,并使用 AONT-RS 方案来进行性能优化。这个云存储系统的框架如图 1 所示,其中主要有三个参与者:数据拥有者、数据使用者和 CSP。系统的主要流程可以分为三个阶段:数据发布、数据检索和用户撤销。

在数据发布阶段开始时,DO 运行 Setup 算法产生公开密钥 PK 和一个主密钥 MK。接着运行 KeyGen 算法得到每个 DU 的私有密钥 SK,然后将 SK 通过一个安全的通道发送给 DU。正如引言中分析的,这个方案需要将原始数据 F(假设数据 F

包含 t 个字,每一个字有 w 位) 分割成碎片。因此 DO 会先运行数据分片(Data Splitting, DS) 算法来分割数据,相应算法如算法 1 所示。在这个算法中,原始数据 F 首先会通过 AONT 方法将原始的 t 个字编码成 $t+1$ 个字,其中第 $t+1$ 个字 c_{t+1} 用来检查重构数据的完整性。生成的密钥 K_1 会被用于重构阶段,并在数据发布阶段被 CP-ABE 算法加密。然后通过 (k,n) IDA 算法将处理过的数据分割成 n 个分片。接着 DU 就可以通过数据发布(Data Publishing, DP)算法将数据发布到云上,相应算法如算法 2 所示。在这个算法中,会随机选择 $n-(k-1)$ 个分片进行加密,这样未授权的 DU 就无法通过其余 $(k-1)$ 个分片来恢复数据,同时在云端只需要保存一份数据副本。

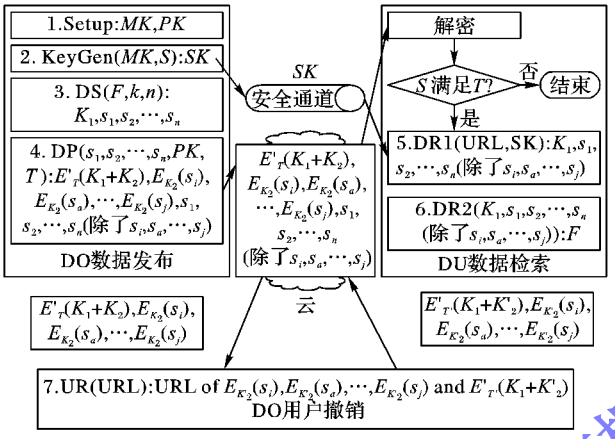

图 1　安全高效云存储系统框架

算法 1　数据分片算法(DS 算法)。

1) 输入原始数据 $F(m_1,m_2,\cdots,m_t)$,IDA 算法的参数 k 和 n;

2) 计算 $m_{t+1}=\text{hash}(m_1,m_2,\cdots,m_t)$;

3) 选择一个随机密钥 K;

4) 计算 $c_i=m_i\oplus E_K(i)$,其中 $1\leqslant i\leqslant t+1$;

5) 计算 $K_1=K\oplus h_1\oplus h_2\oplus\cdots\oplus h_{t+1}$,其中 $h_i=\text{hash}(c_i)$;

6) 输入 $c_1,c_2,\cdots,c_{t+1}$ 运行 (k,n) IDA 算法,输出 n 个分片 $s_1,s_2,\cdots,s_n$;

7) 返回 $K_1,s_1,s_2,\cdots,s_n$。

算法 2　数据发布算法(DP 算法)。

1) 输入 $s_1,s_2,\cdots,s_n,K_1,PK$,和 T;

2) 随机选择 $s_i,s_\alpha,\cdots,s_j$ 分片(总共$(n-(k-1))$ 个分片);

3) 选择一个随机密钥 K_2;

4) 计算 $E_{K_2}(s_i),E_{K_2}(s_\alpha),\cdots,E_{K_2}(s_j)$;

5) 计算 $E'_T(K_1+K_2)=\text{Encrypt}(PK,K_1+K_2,T)$,其中 E' 是 CP-ABE 算法,T 是访问结构树;

6) 将 $E'_T(K_1+K_2),E_{K_2}(s_i),E_{K_2}(s_\alpha),\cdots,E_{K_2}(s_j)$ 和 $s_1,s_2,\cdots,s_n$(除了 $s_i,s_\alpha,\cdots,s_j$) 发布到云端;

7) 返回 $E'_T(K_1+K_2),E_{K_2}(s_i),E_{K_2}(s_\alpha),\cdots,E_{K_2}(s_j)$ 和 $s_1,s_2,\cdots,s_n$(除了 $s_i,s_\alpha,\cdots,s_j$) 的统一资源定位符(Uniform Resource Locator, URL)。

在数据发布之后,每一个 DU 都可以得到发布的数据,但是只有授权的 DU 才可以通过数据检索 DR1(Data Retrieving) 算法解密数据,相应算法如算法 3 所示。之后 DU 可以调用数据重构 DR2(Data Reconstructing) 算法得到原始数据 F,相应算法如算法 4 所示。

DO 可以运行用户撤销(User Revocation, UR) 算法来修改访问策略,相应算法如算法 5 所示。比如可以撤销一个 DU u_i 访问文件 f_j 的权限,也可以授予一个 DU u_j 访问文件 f_i 的权限。这个算法只是简单地更新 $E'_T(K_1+K_2)$ 和 $E_{K_2}(s_i),E_{K_2}(s_\alpha),\cdots,E_{K_2}(s_j)$ 来执行撤销和授权的操作。

算法 3　数据检索算法(DR1 算法)。

1) 输入 URL 和 DU 的私有密钥 SK;

2) 检索得到 $E'_T(K_1+K_2)$ 和 $E_{K_2}(s_i),E_{K_2}(s_\alpha),\cdots,E_{K_2}(s_j)$ 中任意一个(假设为 $E_{K_2}(s_i)$) 和 $s_1,s_2,\cdots,s_n$(除了 $s_i,s_\alpha,\cdots,s_j$);

3) 如果 S 不满足 T,那么退出;

4) 解密 $E'_T(K_1+K_2)$ 得到 K_1 和 K_2;

5) 解密 $E_{K_2}(s_i)$ 得到 s_i;

6) 返回 K_1,s_i 和 $s_1,s_2,\cdots,s_n$(除了 $s_i,s_\alpha,\cdots,s_j$)。

算法 4　数据重构算法(DR2 算法)。

1) 输入 K_1,s_i 和 $s_1,s_2,\cdots,s_n$(除了 $s_i,s_\alpha,\cdots,s_j$);

2) 输入 s_i 和 $s_1,s_2,\cdots,s_n$(除了 $s_i,s_\alpha,\cdots,s_j$),运行 (k,n) IDA 算法得到 $c_1,c_2,\cdots,c_{t+1}$;

3) 计算密钥 $K=K_1\oplus h_1\oplus h_2\oplus\cdots\oplus h_{t+1}$,其中 $h_i=\text{hash}(c_i)$;

4) 计算 $m_i=c_i\oplus E_K(i)$,其中 $1\leqslant i\leqslant t+1$;

5) 计算 $m'_{t+1}=\text{hash}(m_1,m_2,\cdots,m_t)$;

6) 如果 $m'_{t+1}!=m_{t+1}$,那么退出;

7) 返回 $F(m_1,m_2,\cdots,m_t)$。

算法 5　用户撤销算法(UR 算法)。

1) 输入 URL;

2) 得到 $E'_T(K_1+K_2)$ 和 $E_{K_2}(s_i),E_{K_2}(s_\alpha),\cdots,E_{K_2}(s_j)$;

3) 解密 $E'_T(K_1+K_2)$ 得到 K_1 和 K_2;

4) 解密 $E_{K_2}(s_i),E_{K_2}(s_\alpha),\cdots,E_{K_2}(s_j)$ 得到 $s_i,s_\alpha,\cdots,s_j$;

5) 选择另一个随机密钥 K'_2;

6) 计算 $E_{K'2}(s_i),E_{K'2}(s_\alpha),\cdots,E_{K'2}(s_j)$;

7) 计算 $E'_{T'}(K_1+K'_2)=\text{Encrypt}(PK,K_1+K'_2,T')$,其中 E' 是 CP-ABE 算法,T' 是新的访问结构树;

8) 将 $E'_{T'}(K_1+K'_2),E_{K'2}(s_i),E_{K'2}(s_\alpha),\cdots,E_{K'2}(s_j)$ 发布到云端并删除原来的副本;

9) 返回 $E'_{T'}(K_1+K'_2),E_{K'_2}(s_i),E_{K'_2}(s_\alpha),\cdots,E_{K'_2}(s_j)$ 的 URL。

2.2　安全性分析

在 2.1 节中,详细叙述了这个方案,在本节中将分析它的安全性。首先在这个系统中,主要有两类攻击者,一类是不可信的 CSP,另一类是未授权的或者已经撤销的 DU。密文 $E'_T(K_1+K_2)$ 是由 CP-ABE 算法产生的,它已经被证明是安全的[6]。密文 $s_1,s_2,\cdots,s_n$(除了 $s_i,s_\alpha,\cdots,s_j$) 由 AONT-RS 方案产生,至少需要 $s_i,s_\alpha,\cdots,s_j$ 的其中一个才可以重构出数据 F。然而 $s_i,s_\alpha,\cdots,s_j$ 已经被加密成了密文 $E_{K_2}(s_i),E_{K_2}(s_\alpha),\cdots,E_{K_2}(s_j)$。由于 AONT-RS 方案是可计算安全的[9],所以攻击者在不解密 $E_{K_2}(s_i),E_{K_2}(s_\alpha),\cdots,E_{K_2}(s_j)$ 其中一个的前提下是不可能从 $s_1,s_2,\cdots,s_n$(除了 $s_i,s_\alpha,\cdots,s_j$) 中得到任何信息。

密钥 K_2 是由 DO 产生并完全在本地进行加密的。因此，CSP 就无法解密任何一个 $E_{K_2}(s_i)$，$E_{K_2}(s_\alpha)$，…，$E_{K_2}(s_j)$，也就是说 CSP 得不到任何数据 F 有用的信息。现在假设 CSP 可以无限期地保存数据。当一个用户撤销发生时，CSP 可能已经拥有了 s_1，s_2，…，s_n（除了 s_i，s_α，…，s_j）和 $E_{K_2}(s_i)$，$E_{K_2}(s_\alpha)$，…，$E_{K_2}(s_j)$。即使这样，CSP 还是无法得到数据 F。因为撤销的过程不会泄露密钥 K'_2 和 s_i，s_α，…，s_j 的信息给 CSP。所以这个系统对于不可信 CSP 的攻击是安全的。

当一个授权的 DU 被撤销了，那么这个 DU 就无法再解密 $E'_{T'}(K_1 + K'_2)$ 得到 K'_2，从而也就无法再得到数据 F。因此这个系统已经撤销的 DU 的攻击也是安全的。

这个方案无法抵御缓存攻击，也就是说撤销的 DU 可能会保存 s_i，s_α，…，s_j 分片。不过，缓存攻击在实际的系统中是可以被忽略的，因为如果 DU 可以缓存这些分片的话，那么他/她同样也可以保存原始的数据 F。

3 系统开销评估

3.1 理论分析

首先，将对这个系统的理论的时间和存储空间开销进行评估。表 1 显示了所需的符号，并给定了假设的初始值。其中的编解码率由相应算法的实际测试结果为了计算方便而取近似值得到。最后，会将本文的方案与最初使用完全撤销和多重备份方法的原始方案（Original Secure Cloud Storage Scheme，OSCSS）进行比较，这里将本文方案命名为安全高效的云存储方案（Secure and Efficient Cloud Storage Scheme，SECSS）。

表 1 理论分析中使用到的符号

符号	假设的值	描述
R	3	完全备份方案中的副本数
k	10	(k,n) IDA 算法的参数 k
n	12	(k,n) IDA 算法的参数 n
w	8 B	一个字的长度
w_k	32 B	密钥长度
F	12.5 M 字	原始数据大小
B	10 MB/s	DO 的带宽
E_{aes}	40 MB/s	基于密钥的加密算法的编码率
D_{aes}	40 MB/s	基于密钥的加密算法的解码率
E_{cpabe}	1.5 MB/s	CP-ABE 算法的编码率
D_{cpabe}	3 MB/s	CP-ABE 算法的解码率
E_{ida}	50 MB/s	IDA 算法的分割速率
D_{ida}	50 MB/s	IDA 算法的重构速率
E_{hash}	100 MB/s	哈希函数编码率
E_{xor}	500 MB/s	⊕操作的带宽

让 T_{DS} 表示算法 1 将原始数据 F 分割成 n 份所需的时间，它可以根据式(1) 计算得到：

$$T_{DS} = \frac{Fw}{E_{hash}} + \frac{(F+1)w}{E_{aes}} + \frac{2(F+1)w}{E_{xor}} + \frac{(F+1)w}{E_{hash}} + \frac{(F+1)w}{E_{ida}} \tag{1}$$

算法 2 所需的时间 T_{DP} 为：

$$T_{DP} = \frac{(n-(k-1))(F+1)w}{kE_{aes}} + \frac{2w_k}{E_{cpabe}} + \frac{(n(F+1)w)/k + 2w_k}{B} \tag{2}$$

则数据发布阶段所需的时间 $T_{publish}$ 就是：

$$T_{publish} = T_{DS} + T_{DP} \tag{3}$$

类似地，算法 3 和算法 4 所需的时间 T_{DR1} 和 T_{DR2}，以及数据检索阶段所需的时间 $T_{retrieve}$ 分别为：

$$T_{DR1} = \frac{(n(F+1)w)/k + 2w_k}{B} + \frac{2w_k}{D_{cpabe}} + \frac{(F+1)w}{kD_{aes}} \tag{4}$$

$$T_{DR2} = \frac{(F+1)w}{D_{ida}} + \frac{(F+1)w}{E_{hash}} + \frac{Fw}{E_{hash}} + \frac{(F+1)w}{E_{aes}} + \frac{2(F+1)w}{E_{xor}} \tag{5}$$

$$T_{retrieve} = T_{DR1} + T_{DR2} \tag{6}$$

算法 5 用户撤销阶段所需的时间 T_{revoke} 为：

$$T_{revoke} = \frac{2[(n-(k-1))(F+1)w/k + 2w_k]}{B} + \frac{2w_k}{D_{cpabe}} + \frac{(n-(k-1))(F+1)w}{kD_{aes}} + \frac{(n-(k-1))(F+1)w}{kE_{aes}} + \frac{2w_k}{E_{cpabe}} \tag{7}$$

在式(2)、(4)和(7)中，为了简化，用 $2w_k$ 代替了 $E'_{T'}(K_1 + K_2)$。当然，$E'_{T'}(K_1 + K_2)$ 通常只比 $2w_k$ 略大。同样的，本文也认为加密后的数据大小也跟原始数据一样大。因为跟 MB 级别的原始数据相比，几十 KB 的密文数据可以忽略不计。因此 SECSS 的存储开销 S 可以近似地为：

$$S = k^{-1}n(F+1)w + 2w_k \tag{8}$$

可以直接得出 OSCSS 的数据发布时间 $T'_{publish}$、数据检索时间 $T'_{retrieve}$、用户撤销时间 T'_{revoke} 和存储开销 S'：

$$T'_{publish} = \frac{Fw}{E_{aes}} + \frac{w_k}{E_{cpabe}} + \frac{Fw + w_k}{B} \tag{9}$$

$$T'_{retrieve} = \frac{Fw + w_k}{B} + \frac{w_k}{D_{cpabe}} + \frac{Fw}{D_{aes}} \tag{10}$$

$$T'_{revoke} = T'_{publish} + T'_{retrieve} \tag{11}$$

$$S' = R(Fw + w_k) \tag{12}$$

SECSS 与 OSCSS 的理论性能和空间开销的比较如图 2 所示。从图 2 中可知，SECSS 在用户撤销阶段的性能和存储空间开销上都比 OSCSS 更加有效率。不管怎样，SECSS 在数据发布和检索阶段要比原始的方案花费更多的时间。当然，在性能上，SECSS 的主要目的是为了降低 DO 的工作量。假设 DO 在将数据发布到云上以后，需要执行 x 次撤销操作，那么 DO 所需的管理时间 T_{manage} 为：

$$T_{manage} = T_{publish} + xT_{revoke} \tag{13}$$

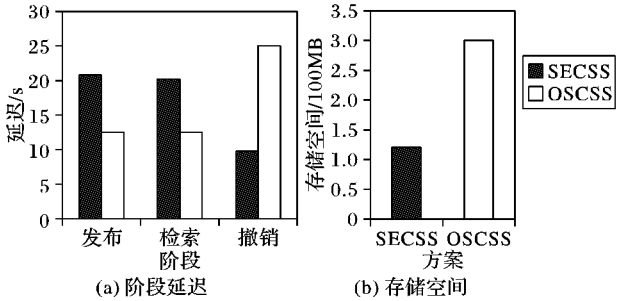

图 2 SECSS 与 OSCSS 理论上的性能和空间开销的比较

图 3 表示的是 SECSS 与 OSCSS 中 DO 的管理时间如何随

着 x 的增加而变化。可以发现,随着撤销操作次数的增加,SECSS 要比 OSCSS 更加有利。因此 SECSS 在撤销操作频繁和数据块很大的情况下要比 OSCSS 更加高效。

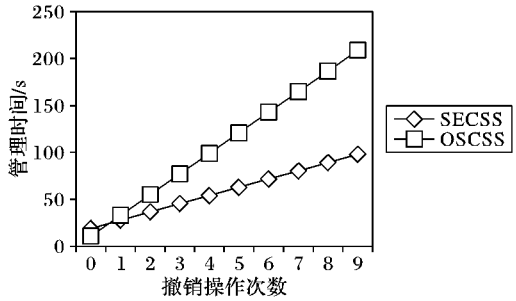

图 3　不同撤销操作次数下 DO 管理时间的变化

3.2　实验结果

本文实现了一个原型系统,并同时对 SECSS 和 OSCSS 的性能和存储空间开销进行了测试。在这个原型系统中,部署了 6 台虚拟机,其中 2 台作为 DO 和 DU 的客户机,另外 4 台作为云存储的服务器。每台客户机的配置为 2.8 GHz 的单核 CPU 和 1 GB 内存。每台服务器的配置为 2.8 GHz 的双核 CPU,2 GB 内存和 500 GB 硬盘。所有机器运行 Ubuntu 14.04 操作系统。在这个系统中,部署了本地的 HDFS[3] 作为云存储服务,使用 OpenSSL 加密库实现高级加密标准 192 位(Advanced Encryption Standard-192 bits, AES-192)和安全散列算法 256 位(Secure Hash Algorithm-256 bits, SHA-256)算法,最后采用 CP-ABE 工具包[12] 来实现 CP-ABE 方案。

首先测试了在不同数据块大小下,数据发布、检索和用户撤销阶段的性能以及存储空间的开销。数据块的大小从 10 MB 变化到 200 MB,同时系统其他参数配置如下:$R = 3$,$k = 10$,$n = 12$。其中测试结果如图 4 所示。根据实验数据可知,数据发布和数据检索性能在数据块增大后性能下降明显。在实际的应用场景中,企业或组织的机密信息文件的大小并不会这么大,例如文献[13]基于云的电子健康记录系统中,具有隐私信息的病人医疗档案多以文档形式存储,其大小一般在几十兆字节。

然后在数据块大小固定、改变 (k,n) 参数情况下,测试了撤销延迟和存储空间开销。数据块的大小固定为 100 MB,(k,n) 参数从(5,7)变化到(15,17)。从图 5 的测试结果中可以发现 SECSS 随着 (k,n) 参数的增加,撤销延迟和存储空间开销会逐渐减少。

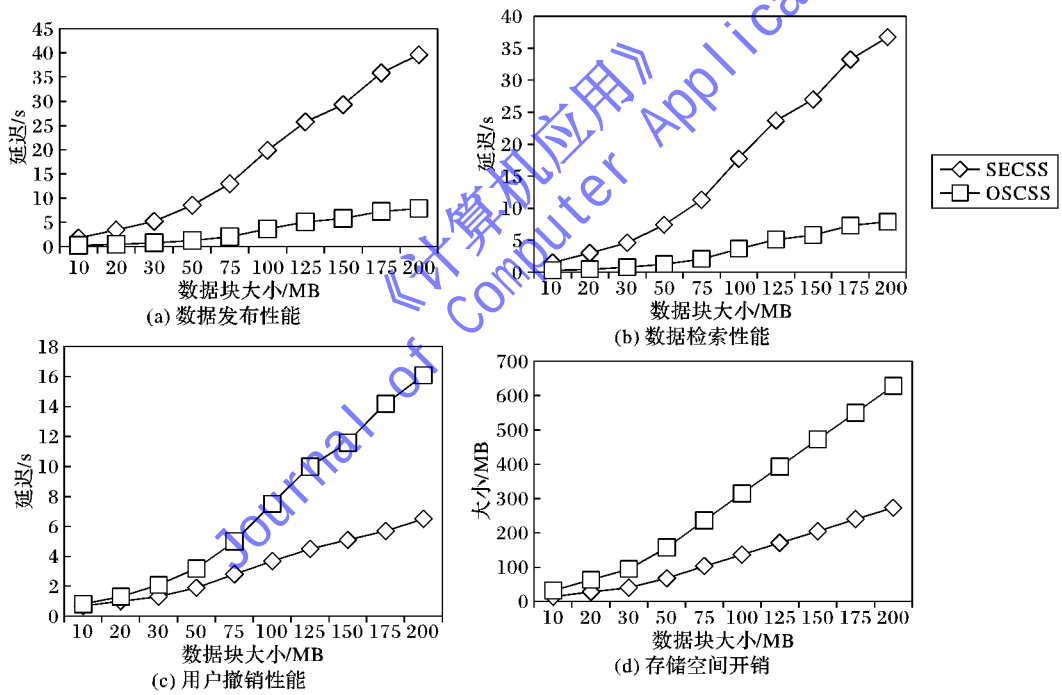

图 4　不同阶段的性能和存储空间开销测试结果

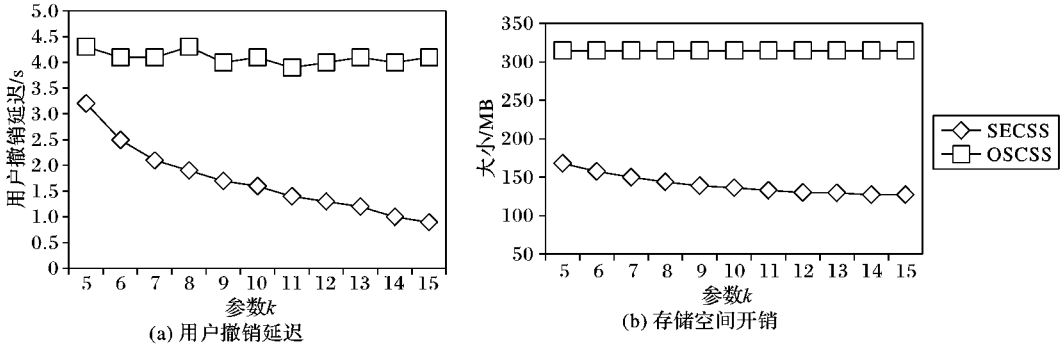

图 5　用户撤销延迟和存储空间开销在不同的 (k,n) 参数下测试结果

从实际的实验结果可以发现,SECSS 很明显地降低了存储空间的开销,同时保证了与多重备份方案相同的可用性。虽然增加了 DO 在数据发布阶段所需的时间,但是在用户撤销阶段得到了明显的改善。在实际的系统中,DO 一般只要

进行一次数据发布的操作而需要进行多次的用户撤销操作，因此 SECSS 在性能的提升上是非常显著的。

4 结语

本文提出了一个安全高效的云存储系统。这个系统是一个基于 CP-ABE 的密码学的访问控制系统，同时在只保存一份数据副本的思想下，提出了一个基于数据分片和秘密共享的高效存储方案。这个方案可以显著降低 DO 的工作量和 CSP 的存储空间开销，这样可以有效地促进密码学的云存储系统的使用。同时，安全分析证明了这个系统在计算上是安全的。从理论分析和实际的测试结果中都可以看出，SECSS 在用户撤销和存储空间开销上都比 OSCSS 更加高效。因此在撤销操作频繁和数据量大的情况下，DO 和 CSP 都将从中得益。总的来说，本文的优化方案是在系统安全性和整体开销上作出了一个最优化的权衡。

参考文献：

[1] SNIA Technical Position. Cloud Data Management Interface (CDMI) v1.0.2 [EB/OL]. [2015-05-01]. http://snia.org/sites/default/files/CDMI%20v1.0.2.pdf.

[2] GHEMAWAT S, GOBIOFF H, LEUNG S T. The Google file system [J]. ACM SIGOPS Operating Systems Review, 2003, 37(5): 29 – 43.

[3] BORTHAKUR D. The hadoop distributed file system: architecture and design [EB/OL]. [2015-05-01]. http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

[4] Amazon. Amazon simple storage service [EB/OL]. [2015-05-01]. http://aws.amazon.com/s3/.

[5] KAMARA S, LAUTER K. Cryptographic cloud storage [C]// Financial Cryptography and Data Security, LNCS 6054. Berlin: Springer, 2010: 136 – 149.

[6] BETHENCOURT J, SAHAI A, WATERS B. Ciphertext-policy attribute-based encryption [C]// Proceedings of the 2007 IEEE Symposium on Security and Privacy. Washington, DC: IEEE Computer Society, 2007: 321 – 334.

[7] SHAMIR A. How to share a secret [J]. Communications of the ACM, 1979, 22(11): 612 – 613.

[8] GOYAL V, PANDEY O, SAHAI A, *et al.* Attribute-based encryption for fine-grained access control of encrypted data [C]// Proceedings of the 13th ACM Conference on Computer and Communications Security. New York: ACM, 2006: 89 – 98.

[9] RESCH J K, PLANK J S. AONT-RS: blending security and performance in dispersed storage systems [C]// Proceedings of the 9th USENIX Conference on File and Storage Technologies. Berkeley: USENIX Association, 2011: 1 – 12.

[10] RABIN M O. Efficient dispersal of information for security, load balancing, and fault tolerance [J]. Journal of the ACM, 1989, 36(2): 335 – 348.

[11] RIVEST R L. All-or-nothing encryption and the package transform [C]// Proceedings of the 4th International Workshop on Fast Software Encryption. New York: ACM, 1997: 210 – 218.

[12] BETHENCOURT J, SAHAI A, WATERS B. CP-ABE toolkit [EB/OL]. [2015-05-01]. http://acsc.cs.utexas.edu/cpabe/.

[13] ALABDULATIF A, KHALIL I, MAI V. Protection of Electronic Health Records (EHRs) in cloud [C]// Proceedings of the 35th IEEE Conference on Engineering in Medicine and Biology Society. Piscataway: IEEE, 2013: 4191 – 4194.

（上接第 3412 页）

[2] RIVEST R L, ADLEMAN L, DERTOUZOS M L. On data banks and privacy homomorphisms [C]// Foundations of Secure Computation. New York: Academic Press, 1978: 169 – 179.

[3] RIVEST R L, SHAMIR A, ADLEMAN L. A method for obtaining digital signatures and public-key cryptosystems [J]. Communications of the ACM, 1978, 21(2): 120 – 126.

[4] GENTRY C. Fully homomorphic encryption using ideal lattices [C]// STOC'09: Proceedings of the 41st Annual ACM Symposium on Theory of Computing. New York: ACM, 2009: 169 – 178.

[5] GENTRY C. A fully homomorphic encryption scheme [D]. Stanford: Stanford University, 2009: 1 – 45.

[6] GENTRY C, HALEVI S. Implementing Gentry's fully-homomorphic encryption scheme [C]// Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology — EUROCRYPT 2011, LNCS 6632. Berlin: Springer, 2011: 129 – 148.

[7] GENTRY C B. Fully homomorphic encryption method based on a bootstrappable encryption scheme: U. S. Patent, 8630422 [P]. 2014-01-14.

[8] van DIJK M, GENTRY C, HALEVI S, *et al.* Fully homomorphic encryption over the integers [C]// Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology — EUROCRYPT 2010, LNCS 6110. Berlin: Springer, 2010: 24 – 43.

[9] BRAKERSKI Z, VAIKUNTANATHAN V. Efficient fully homomorphic encryption from (standard) LWE [C]// FOCS'11: Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science. Washington, DC: IEEE Computer Society, 2011: 97 – 106.

[10] BRAKERSKI Z, VAIKUNTANATHAN V. Fully homomorphic encryption from ring-LWE and security for key dependent messages [C]// Proceedings of the 31st Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2011, LNCS 6841. Berlin: Springer, 2011: 505 – 524.

[11] DITTRICH J, QUIANÉ-RUIZ J-A. Efficient big data processing in Hadoop MapReduce [J]. Proceedings of the VLDB Endowment, 2012, 5(12): 2014 – 2015.

[12] TANG D, ZHU S, CAO Y. Faster fully homomorphic scheme over integer [J]. Computer Engineering and Applications, 2012, 48(28): 117 – 122. (汤殿华, 祝世雄, 曹云飞. 一个较快速的整数上的全同态加密方案[J]. 计算机工程与应用, 2012, 48(28): 117 – 122).

[13] HOWGRAVE-GRAHAM N. Approximate integer common divisors [C] // Proceedings of the CaLC 2001 International Conference on Cryptography and Lattices, LNCS 2146. Berlin: Springer, 2001: 51 – 66.