软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(8):1831−1843 [doi: 10.13328/j.cnki.jos.004463]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种云存储环境下的安全网盘系统*

傅颖勋[1], 罗圣美[1,2], 舒继武[1]

[1](清华大学 计算机科学与技术系,北京 100084)

[2](中兴通讯股份有限公司,江苏 南京 210012)

通讯作者: 舒继武, E-mail: shujw@tsinghua.edu.cn, http://www.tsinghua.edu.cn

摘 要: 随着云存储的迅速推广,有越来越多的用户开始使用网盘系统存放数据.然而,最新的研究结果却表明:现有网盘系统普遍存在着安全漏洞.近年来,网盘泄漏用户数据的事件更是印证了这些漏洞的存在.为此,提出一种云存储环境下的安全网盘系统架构,并在此架构上设计实现了 CorsBox 系统.CorsBox 系统采用一种基于目录树的同步方式,在提高安全性的同时保证了共享操作的最终一致性,为用户提供访问权限控制、大数据断点传输、版本控制等功能.测试结果表明,安全机制仅给系统带来了很少的额外开销,说明 CorsBox 系统在提高数据安全性的同时依然具有良好的性能.

关键词: 云存储;安全网盘系统;目录树;密钥管理;最终一致性

中图法分类号: TP333

中文引用格式: 傅颖勋,罗圣美,舒继武.一种云存储环境下的安全网盘系统.软件学报,2014,25(8):1831−1843. http://www.jos.org.cn/1000-9825/4463.htm

英文引用格式: Fu YX, Luo SM, Shu JW. Secure online storage system based on cloud storage environment. Ruan Jian Xue Bao/Journal of Software, 2014,25(8):1831−1843 (in Chinese). http://www.jos.org.cn/1000-9825/4463.htm

Secure Online Storage System Based on Cloud Storage Environment

FU Ying-Xun[1], LUO Sheng-Mei[1,2], SHU Ji-Wu[1]

[1](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

[2](Zhongxing Telecom Equipment Corporation, Nanjing 210012, China)

Corresponding author: SHU Ji-Wu, E-mail: shujw@tsinghua.edu.cn, http://www.tsinghua.edu.cn

Abstract: With the rapid development of cloud storage, more and more people prefer to store their data in online storage systems. However, recent researches indicate that security problems still remain in current online storage systems, and some recent data leakage accidents of online storage systems also prove the existence of these vulnerabilities. Such security problems seriously hinder the development of online storage system. To address the issue, this study designs and implements a secure online storage system called CorsBox. CorsBox proposes a data synchronous protocol based-on directory trees for fast synchronization between data plaintext and ciphertext, designs a three-level key management scheme to enhance the security of user's data, and presents an effective method to maintain system eventual consistency. The paper finally conducts a set of intensive experiments on modern servers and the result shows that the security mechanisms only incur a little extra performance expenses, indicating that CorsBox can provide enhanced security for user's data while maintaining good performance.

Key words: cloud storage; secure online storage system; directory tree; key management; eventual consistence

随着云存储应用的推广,越来越多的用户开始使用网盘系统存放数据.网盘系统能够为用户提供数据备份、数据共享、数据同步以及用户间协同工作等功能[1−6].

* 基金项目: 国家自然科学基金(60925006, 61232003); 国家科技重大专项(2013ZX03002004-003); 中兴通讯产学研项目(One11 12300007)

收稿时间: 2012-09-05; 修改时间: 2013-04-09; 定稿时间: 2013-07-30

然而,近年来频发的网盘系统安全问题说明:现有的网盘系统存在着一些漏洞,导致用户数据的私密性、完整性受到一定的威胁.2011 年,Mulazzani 等人在分析目前主流网盘的安全性后指出:除了网盘提供方窃取数据以外,现有网盘系统普遍容易遭到哈希值操纵、偷窃宿主 ID 和直接下载等方式的攻击,并夸张地认为,云存储的黑暗时代即将来临[7].

现有主流网盘系统根据其安全机制一般可分为 3 类:

- 第 1 类网盘系统没有加密功能,直接将数据明文存放在服务器中,如 iDisk[6]等;
- 第 2 类网盘系统由服务器端对数据加密后存放在云存储中,并由服务器保管密钥,如 DropBox[2]等;
- 第 3 类网盘系统由客户端对数据加密,密钥以层级加密的方式管理,客户端自己保存根密钥,其他密钥以密文的形式通过服务器保存在云存储中,如 SpiderOak[4],Wuala[5]等.

在安全方面:

- 第 1 类网盘系统直接在服务器上存放数据明文,任何能够访问底层存储介质的用户均能查看,因此安全性很差;
- 第 2 类网盘系统虽然以密文的形式存放数据,但服务器拥有用户的数据明文和所有密钥,仍然能够随时访问或篡改用户数据;
- 第 3 类网盘系统虽然在客户端加密数据,但它们提供一些功能仍然让用户对其安全性产生怀疑,例如“在线重置数据密钥”等,同时也不能彻底防止文献[7]中提到的攻击.

针对上述安全问题,本文在第 3 类网盘的基础上提出了一套新的架构,并根据这套架构实现了一个安全网盘原型系统——CorsBox 系统.CorsBox 系统以任何情况下存储服务商无法取得用户数据明文为第一设计原则,在提高数据安全性的同时,为用户提供数据同步与共享、文件版本控制以及用户间协同工作等功能,并用一些基于数据密文的共享与同步技术,有效地防止了文献[7]中所提到的哈希值操纵等攻击方式,在保证高效性的同时提高了用户数据的安全性.

本文第 1 节介绍 CorsBox 系统的体系结构.第 2 节介绍 CorsBox 系统中的一些关键技术.第 3 节对 CorsBox 系统的安全性和性能进行测试评估.第 4 节介绍相关工作.最后对本文进行总结.

1 CorsBox 系统体系结构

CorsBox 系统用客户端-服务器-云存储端的架构,其体系结构如图 1 所示.

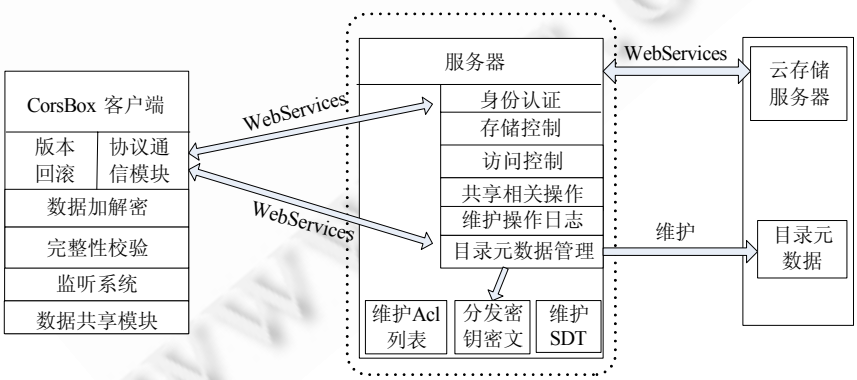


Fig.1 Architecture of CorsBox

图 1 CorsBox 的体系结构

服务器是整个系统的控制中心,它集身份认证、访问控制、数据同步以及文件版本控制等功能于一体,通过一系列数据传输与控制操作,维护客户端和存储服务器之间的数据一致性;云存储端是一个独立的云存储平台,CorsBox 系统采用 OpenStack[8]的 swift[9]项目搭建底层云存储平台来存放用户数据和系统的一些元数据信

息;客户端主要负责数据加解密以及数据的共享、同步等操作.客户端与服务器、服务器与云存储平台之间以Webservice 的方式进行通信;客户端之间以存放在云存储中的目录元数据为媒介,通过公私钥加密的方式分发用户数据的加密密钥.

这种架构有以下优点:

1) 所有数据及密钥的加解密都是在客户端进行,服务器得不到用户的数据明文,从而无法偷窥用户的数据.
2) 系统通过公私钥加密的方式传递密钥,只有数据拥有者授权且通过服务器的身份认证后,用户才能获取到数据明文,二者缺一不可,提高了用户密钥在传递过程中的安全性.
3) 系统针对目前主流网盘中存在的安全漏洞及攻击方式,设计并实现了 DirTree 同步协议,在高效进行数据同步的同时,有效地阻止了这些攻击方式,提高了系统安全性.
4) 系统具有较好的可用性、可靠性和可扩展性.云端采用的 Swift 项目是一个有较强可靠性机制的分布式存储系统.服务器之间共用的数据内容只是用户名、密码、用户服务器端目录树及其对应的日志操作等少量数据,因此可以很容易用集群实现,从而提高整个系统的效率与可扩展性.

2 CorsBox 系统关键技术

CorsBox 系统用 DirTree 协议使系统能够在数据明文与密文之间高效同步的同时,有效地防止哈希值操纵攻击;用 3 层密钥管理方式来提高数据存储与共享过程中的安全性与高效性;用一些维护一致性的技术来保证共享过程中系统的最终一致性;用数据分块与断点续传的方式提高单个大文件的传输效率与易用性;用服务器端目录树中文件号和历史文件为用户提供文件版本回滚功能;用目录树统一管理和服务器集群的方式提高系统的可扩展性与整体性能.本节将详细描述这些关键技术.

2.1 DirTree数据同步协议

数据同步是网盘系统最基本的功能之一,目前,网盘系统的同步协议中大都采用了 Rsync[10]方法.Rsync 方法是一种增量上传的同步方法,通过比对文件块的哈希值作为判断的依据来执行数据同步操作.但是,随着网盘系统中安全机制的添加,基于 Rsync 方法的数据同步方式对相同数据的明文和密文之间的同步支持并不是很好(客户端保存的是数据明文,云端保存的是数据密文),而且很容易受哈希值操纵等攻击方式(详见第 4 节相关工作)的影响.为此,本文提出了 DirTree 同步协议,通过构建和比对目录树,使得添加了安全机制后的网盘系统也能正确、高效地同步数据.

目录树是一种用于记录本地目录和文件的状态与属性的数据结构,CorsBox 系统中包括以下 3 种目录树:

1) 服务器端目录树(server directory tree,简称 SDT).服务器端目录树存放在服务器端,用于记录存储服务器中数据状态的实时情况.由于存储服务器只是存放数据的密文而并不记录数据的状态,服务器端目录树必须与存储服务器中的数据状态保持实时一致,使得系统通过此目录树中记录的状态,就可以了解云端的数据分布情况.
2) 客户端目录树(client directory tree,简称 CDT).客户端目录树存放在客户端,用于记录用户本地数据的状态.由于用户有可能在并未开启系统客户端的情况下对其本地数据进行操作,客户端目录树记录的其实是最近一次用户退出系统客户端时候的数据状态.
3) 磁盘目录树(disk directory tree,简称 DDT).磁盘目录树跟前面两棵目录树不太相同,它只是用来记录用户数据在磁盘中的实时情况,因此不需要持久化,使用时直接扫描磁盘进行构建即可.

由于 SDT 记录了数据在服务器端的基本信息,CDT 记录了数据在上次关闭客户端之时的基本信息,DDT 记录了数据在磁盘中的信息,因此通过比对这 3 棵目录树,系统就可以分析出服务器与客户端各自数据的变化情况,从而确定相应文件或目录需要进行的同步操作.

DirTree 数据同步协议的算法步骤如下:

1) 客户端通过用户名和密码向服务器确定身份.

2) 客户端下载当前用户所对应的的 SDT,并在内存中进行构建.

3) 客户端根据本地保存的文件,在内存中构建当前用户的 CDT.

4) 客户端扫描用户的工作目录,构建当前机器上的 DDT.

5) 客户端依次比对 SDT,CDT 和 DDT 中每一个节点,并记录下所有信息不一致的节点,包括节点缺失、文件的版本号不一致、文件的最后修改时间不一致等.

6) 客户端根据步骤 5)中记录的信息,依次生成数据同步所需要执行的操作(例如文件上传、下载),并以一定的格式保存到操作队列中.由于此步骤的情况较多,本文在此通过以下典型案例进行简单的描述:

- 若某文件在 SDT 中存在,而在 CDT 和 DDT 中均不存在,则说明该文件是在其他机器上被上传的,对该文件进行的同步操作是下载该文件;
- 若某文件在 DDT 中存在,而在 SDT 与 CDT 中均不存在,则说明该文件是在用户离线时被创建的,需要对该文件进行上传操作处理;
- 若某文件在 SDT,CDT 和 DDT 中都存在,但是 CDT 和 DDT 中记录的最后修改时间不一致,则说明该文件用户在离线时做了修改,因此需要将文件的版本号加 1 后,执行上传操作;
- 若某目录在 SDT 中存在,而在 CDT 和 DDT 中不存在,则说明该目录为其他用户离线创建,需要下载该目录及其子目录下的所有文件;
- 若某文件在 SDT,CDT 和 DDT 中都存在,但是 SDT 和 CDT 中的版本号不一致,则说明该文件被其他用户做了修改,因此需要去服务器下载最新的版本.

7) 依次执行操作队列中节点所记录的操作,完成同步过程.

由此可见,DirTree 同步协议在整个同步过程中都没有用到任何哈希值,从而彻底杜绝一切以哈希值为媒介的方式对系统进行攻击.在同步操作的产生过程中,该协议不涉及文件内容的相关属性,因此能够较好地支持数据明文和数据密文之间的同步.

2.2 层级密钥管理分发方式

在安全网盘系统中,如何对密钥进行管理分发是一个非常重要的问题,因为它将直接影响到整个系统的机密性、稳定性以及高效性.密钥管理的关键在于如何减少管理密钥的开销和如何能够快速获得所需的密钥,并保障数据共享过程中密钥的机密性.CorsBox 系统在清华大学 Corslet[11]系统的密钥管理基础上,用文件密钥、目录密钥、用户公私钥三层式的结构对众多密钥进行管理与分发(如图 2 所示).

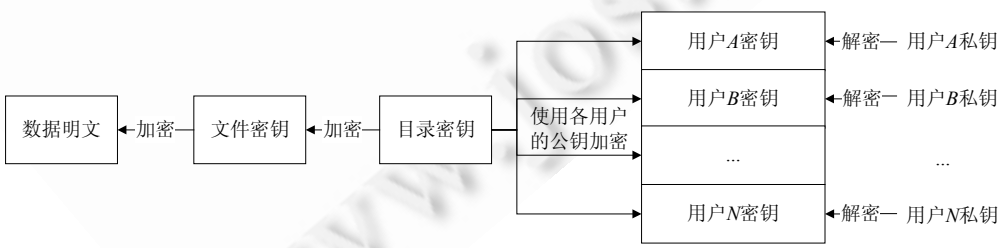


Fig.2 Level-Based key management

图 2 层级密钥管理

文件密钥以密文的形式与原始文件的密文一同存放在上传文件中,是层级密钥的第 1 层.CorsBox 系统使用文件明文的哈希值作为密钥对文件进行加密,这种加密方式的好处在于:

第一,数据明文的哈希值可以用作完整性校验,减少了存储空间的使用;

第二,相同内容的文件经加密后所生成的密文也相同,为系统后期进行重复数据删除操作提供了可行性,能够极大地提高云存储平台的空间利用率.

目录密钥以密文形式保存在目录元数据中,负责数据共享时的密钥分发,是层级密钥的第 2 层.在 CorsBox 系统中,当目录在被设置为共享时,服务器需要为该目录生成目录元数据并保存在云端.目录元数据中包含目录的绝对路径、拥有者 ID、拥有者密钥密文、访问控制列表这 4 项,其访问控制列表中的每一项又包含了共享者的 ID 和该共享者所对应的目录密钥密文,以控制共享用户的访问权限.此外,CorsBox 系统中的每个用户都有一个主密钥,然后,在主密钥的基础上生成目录密钥,以用来加密文件密钥(即文件明文的哈希值).一般来说,非共享目录所对应的目录密钥就是主密钥本身,由用户设置并保管.当用户需要共享某目录时,系统根据用户的配置选择是否重新生成目录密钥,并根据需要重新计算该目录及其子目录下的所有的文件密钥,然后上传至服务器替换原有的文件密钥.当目录的拥有者想将某共享目录授权给某用户时,只需使用该用户的公钥加密此共享目录的目录密钥,然后将其存放在目录元数据的访问控制列表中即可.

用户公私钥对由用户自行保管,通过配置路径的方式使用,是层级密钥中的第 3 层.用户在注册新帐号的同时生成公私钥对,并将公钥存放在服务器端;私钥则是用户的绝密信息,不能让其他任何人知道,一旦泄漏则必须立即向服务器申请更换公私钥对,同时撤销所有跟此用户有关的授权信息.当用户需要访问共享目录时,首先由服务器端验证其访问权限后,将数据密文和目录密钥密文返回给该用户所在的客户端,之后再由客户端使用用户私钥解密出目录密钥,最后解密出数据明文并对其进行完整性校验.

下面分别以文件的上传、下载、目录共享和添加访问权限为例,揭示 3 层密钥的存放与运行机理(此处省略了客户端向服务器确定身份的环节):

文件上传算法步骤如下:

1) 客户端计算待上传文件的哈希值,并以此为密钥加密文件.
2) 若为普通文件,客户端利用主密钥(即目录密钥)加密哈希值;若为共享文件,客户端则从临时文件中获取目录密钥来加密哈希值.
3) 客户端将哈希值密文和文件密文合并成一个文件,并上传至服务器.
4) 服务器将合并后的文件保存至后台云存储中,并更新该用户的服务器端目录树.

文件下载算法步骤如下:

1) 客户端下载文件,若是共享文件,则同时下载当前用户所对应的目录密钥密文.
2) 若是共享文件,则客户端使用当前用户的私钥解密出目录密钥,并保存在临时文件中;若是非共享文件,当前用户的主密钥即是目录密钥.
3) 客户端利用目录密钥从下载的文件中解密出文件密钥,继而解密出文件明文.
4) 客户端重新计算文件明文的哈希值,并与文件密钥做比对,进行完整性校验.

目录共享算法步骤如下:

1) 客户端根据当前配置,选择是否需要更换目录密钥,若更换,则随机生成新的目录密钥,并用新的目录密钥加密待共享目录及其子目录中所有的文件密钥.
2) 客户端使用当前用户(即拥有者)的公钥加密目录密钥,并和重新生成的文件密钥密文一起上传至服务器.
3) 服务器为共享目录生成目录元数据,并在目录元数据中插入绝对路径、拥有者 ID 和拥有者密钥密文(即步骤 2)中生成的目录密钥密文).
4) 服务器更新当前用户的服务器端目录树,将该目录设置为共享状态.
5) 服务器用新的文件密钥,依次替换相应的旧文件密钥(若用户选择不更换目录密钥,则跳过此步骤).

添加访问权限算法步骤如下:

1) 客户端从服务器(或本地临时文件)中获取并解密目录密钥.
2) 客户端使用待添加用户的公钥加密目录密钥,并上传至服务器.
3) 服务器在该共享目录所对应目录元数据的访问控制列表中插入一条新纪录,其中包含共享用户 ID 及步骤 2)中生成的目录密钥密文.

通过这样的3层密钥管理的方式,CorsBox系统只需花费少量的时间与空间,就能方便地对数量众多的密钥进行管理分发,在保证系统功能的同时,在很大程度上降低了所需的密钥管理与维护开销.

2.3 数据共享与一致性

CorsBox 系统的最小共享单位是文件夹,用户可以将自己的文件夹(及其包含的所有文件)共享给任意一个用户,前提是用户必须知道此用户的用户名.相对于以文件为最小共享单位而言,以文件夹为最小单位的共享方式在提高安全性的同时,不仅能够减少客户端大量的加解密开销,还能够有效地减少用户的冗余操作.此外,CorsBox 系统客户端的监听模块能够即时监听用户对 CorsBox 工作目录下的文件或文件夹下进行的移动、删除、重命名等操作,当这些操作发生时,监听模块立刻调用相关接口对操作进行实时的处理.

数据共享引发了系统的冲突问题.在网盘系统中,如何正确、高效地处理冲突以保证一致性,是一个很重要的问题.CorsBox 系统在目录树的基础上提出了一种一致性协议,能够保证系统的最终一致性(eventual consistency)[12,13],即,若系统保证没有新的更新提交,则最终所有的访问都将获得最后更新的版本.

在 CorsBox 系统中,最终一致性分为数据内容的最终一致性和路径的最终一致性.

- 数据内容的最终一致性

在数据同步时,由于操作的非实时性,用户上传文件时可能存在内容上的冲突.例如,用户 *A* 和 *B* 都是文件 foo 的合法用户(包括数据拥有者和授权用户,下同),用户 *A* 修改了文件 foo,而用户 *B* 在用户 *A* 上传到云端前也修改了 foo 文件,然后 *A* 执行数据同步将 foo 文件上传到云端.此后,当用户 *B* 执行同步时就会产生数据内容的冲突,由此导致了数据内容的不一致.针对这类情况,CorsBox 系统用在目录树中记录文件版本号的方式,在客户端上传文件的同时,将文件的版本号加 1,并在其服务器端目录树中记录文件最新的版本号.这样一来,当用户 *B* 进行数据同步时,将会发现 foo 在服务器端的版本号不小于自己客户端目录树中的版本号,就不执行上传操作,而是转入冲突处理:将本地的 foo 重命名后上传至云端并下载云端的 foo 文件,再由合法用户根据需要进行处理.

- 数据路径的最终一致性

在 CorsBox 系统共享机制下,任何合法用户都可以对共享的目录/文件进行移动、重命名操作,而云端却只保留一份数据(加上 *n* 个副本,*n* 由用户指定),这种多对一的关系势必会引起目录/文件名的不一致.为了解决这个问题,CorsBox 专门设置了日志机制:每个共享目录均对应一个日志,用来记录所有授权用户对该共享目录及其子目录进行的操作.当某个授权用户同步时,服务器根据日志先在服务器端同步目录/文件名,然后根据情况将部分日志内容返回给客户端,客户端先进行目录/文件名同步,然后再进行数据同步.

2.4 大数据断点传输机制

受网络或其他因素的影响,许多网盘系统都对文件大小有限制.例如,skydrive[14]系统将文件大小限制在300M,金山快盘则是近期才从100M提升到了300M.为了突破单个文件的大小限制,CorsBox系统使用大数据分块断点续传的机制,在不限制文件大小的同时提高了传输效率.

网盘系统中的大数据是指相对较大的数据文件.在 CorsBox 系统中,我们将 64M 以上的单个数据文件定义为大数据文件(可自行配置).为了支持大数据文件上传,CorsBox 系统在客户端按照固定的大小对其进行分块,再将数据块加密后逐一上传;当所有的数据块均上传至云存储后,服务器再调用云存储平台的“合并”指令,将所有的小块合并成原来的单个文件.若上传过程意外中止,待下次同步时,客户端会根据用户名、当前机器及服务器端目录树中的断点信息进行判断,并根据判断结果决定是否从中断点开始续传.

大数据的断点传输机制不仅解决了 CorsBox 单个文件的大小限制的问题,而且还提供断点续传的功能,极大地提高了系统的实用性与大文件的传输效率.

2.5 文件版本控制

CorsBox 系统在云存储端保存用户数据的所有历史版本,以提供数据回滚功能.在用户上传文件时,客户端根据同步比对结果,将当前文件版本的版本号上传至服务器,服务器根据版本号将上传文件重命名后,再保存至云存储服务器,同时更新该用户的服务器端目录树.这样,当用户需要回滚到某文件的历史版本时,系统只需根

据服务器端目录树中记录的文件版本信息,将该文件所有历史版本的版本号及上传时间返回给客户端,客户端就能根据自己需要的文件选择合适的版本进行回滚.

2.6 目录树缓存与服务器集群

由于 CorsBox 服务器需要同时处理很多个客户端的请求,很容易成为整个系统的瓶颈,因此,本文用服务器集群来实现.为了保证集群后系统的正确性与一致性,本文设计并实现了目录树服务器,对用户的服务器端目录树进行统一管理,并添加缓存机制以减少服务器端目录树的构建与写回次数.

服务器集群如图 3 所示,客户端将请求发送到服务器集群,由集群中随机的一台服务器进行响应.客户端与服务器之间的通信采用 Session 机制,能够保证客户端在 Session 失效前的所有请求均由同一台服务器处理.

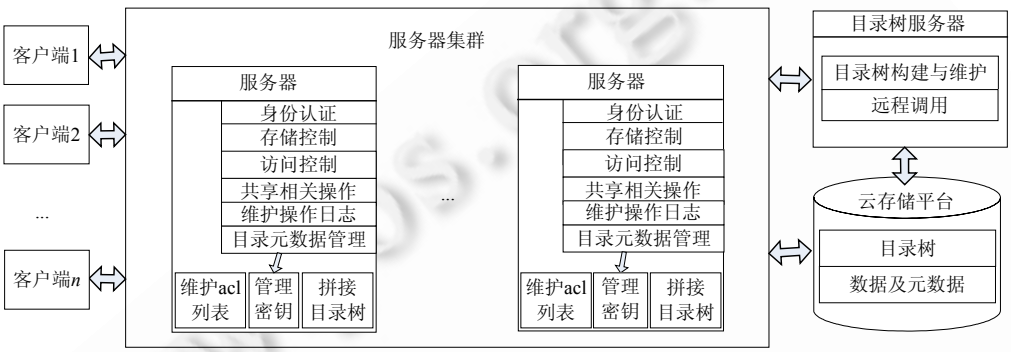


Fig.3 Directory tree cache and the cluster of servers

图 3 目录树缓存与服务器集群

由于用户的服务器端目录树经持久化后保存在云端,当某台 CorsBox 服务器构建用户的服务器端目录树时,可能有其他服务器在修改此目录树后,还没来得及写入云端,从而导致系统出错.针对此问题,本文采用目录树服务器对所有用户的服务器端目录树进行统一的管理,并添加目录树缓存机制以提高效率.当 CorsBox 服务器需要构建某个目录树时,首先向目录树服务器发起请求,目录树服务器查看本地缓存是否有本次请求需要的目录树:

- 若有,则直接返回;
- 若没有,则从云端构建后返回给 CorsBox 服务器,同时将此目录树添加到缓存中.

为了保证系统的正确性,在目录树服务器将某个目录树交给 CorsBox 服务器时,需要对该目录树锁定,直到服务器传回后再解除锁定.由于构建目录树所需要的内存资源较少,用户的服务器端目录树的使用时间也比较集中(主要在数据同步过程中使用),因此,目录树服务器采用相对简单的 LRU 替换算法,在替换发生时,将最近最少使用的目录树写回云存储.

服务器端目录树的缓存机制使得系统减少了从云存储中读取目录树的次数,并在提高系统效率的同时,使得服务器能够方便用集群实现,从而突破了系统中单个服务器难响应更多用户的瓶颈,在很大程度上提高了系统的可扩展性和整体性能.

3 测试与分析

本文通过了一系列的测试来评估 CorsBox 系统的安全性与性能,包括检验系统在不可信的网络及存储环境下提供的安全功能的正确性,以及测试生成同步操作队列、上传下载以及权限变更等操作的性能.

3.1 测试环境

CorsBox 系统测试使用 10 台机器,其中,4 台服务器用来部署云存储端,3 台服务器用来部署 CorsBox 服务

器(目录树服务器搭建在这 3 台服务器中的 1 台上),另外 3 台 PC 机作为客户端,分别以用户 *A*、用户 *B* 和用户 *C* 登录服务器并进行相关操作.7 台服务器的硬件环境是 Intel(R) Xeon(R) X5472,主频为 3.0GHz 的四核 CPU,8GB 内存,服务器间以千兆局域网连接;软件环境是 Ubuntu Linux 2.6.33 内核,openstack-swift 1.3.0, bcprov-jdk16-146.jar.3 台客户端的硬件环境为 Intel(R) Core(TM) 2,主频为 2.40GHz 的双核 CPU,2GB 内存,软件环境为 Windows 7 操作系统.服务器与云存储之间的连接网络为高速内部网络;客户端与服务器之间的连接网络为普通的公用网络(校园网).

在安全算法的选择上,使用 SHA-1 函数来进行哈希值的计算,以 AES-256 系列函数作为加解密函数,以 ecb 作为加密模式(用户也可以通过配置文件来选择加密算法和加密模式),以 X509 系列函数来实现身份验证;云存储为每份文件保留 3 份副本.

3.2 安全性测试与分析

本文分别从数据机密性、数据完整性、用户权限管理的正确性和 CorsBox 服务器与共享用户共谋这 4 个方面对 CorsBox 系统进行安全性测试与分析.具体的测试内容与测试结果见表 1.

Table 1 Security evaluations of CorsBox

表 1 CorsBox 安全性测试

测试内容	测试结果
数据机密性保护:服务器绕过 CorsBox 系统,直接调用云端接口查看用户数据内容.	数据内容为加密后的乱码,无法得知被查看数据明文.
数据完整性校验:服务器绕过 CorsBox 系统篡改用户数据.	CorsBox 系统弹出数据已遭到破坏的警告.
权限管理:测试某个用户在其被授予某个共享目录的使用权限之前、被授予权限之后,以及被撤销权限之后的结果.	被授予权限前,用户看不到其他用户的任何数据;被授予权限后,用户可以查看并使用所有共享给自己的数据;被撤销权限后,用户的共享文件被从共享夹中移除,作为自己的本地文件保存在磁盘中.
CorsBox 服务器与共享用户共谋:共享用户将解密后的目录密钥发送给 CorsBox 服务器.	若用户共享时选择为共享目录重新生成密钥,服务器只能解密出该密钥所对应的共享目录下的内容;若用户共享时选择不重新生成目录密钥,则服务器能够解密出该用户上传的所有内容.

从表 1 中可以看出:

- 在共享用户不与 CorsBox 服务器方共谋的情况下,CorsBox 系统无法通过客户端或从用户上传的内容中窃取或分析出所需的密钥,也就无法获得数据明文;
- 由于 CorsBox 服务器无法获得用户的目录密钥(主密钥),也就无法伪造经目录密钥加密后的哈希值,从而保证完整性校验的正确性.

若共享用户与 CorsBox 服务器串通,并将自己解密出的目录密钥交给 CorsBox 服务器,即使在这种情况下,若用户共享时选择了重新生成目录密钥,CorsBox 服务器只能窃取到相应的共享目录下的数据(该共享用户不与服务器共谋也能合法获取或修改的部分内容);若用户选择不重新生成目录密钥,这种情况下则面临着所有数据均被偷窥或篡改的风险.由于共享用户和 CorsBox 服务器共谋在实际应用中只是一个概率很小的情况,因此,CorsBox 系统在客户端为用户提供了相应的配置选项,用户只需根据数据的实际情况,在安全性和易用性之间做出平衡的选择.

3.3 性能测试与分析

DirTree 同步协议的核心是生成同步操作队列,为了测试该协议的效率,本文对生成同步操作队列的时间开销做了具体的测试与分析.在 CorsBox 系统中,数据上传、下载、共享与权限变更的过程中包含了加解密操作.因此,本文对这些过程中各个部分的时间开销进行了测试,以分析加解密操作为系统带来的额外开销.

3.3.1 生成同步操作队列的开销

生成同步操作是指客户端通过比对目录树,对系统工作目录中的数据状态进行检验,以知道哪些数据需要执行怎样的同步操作,然后将需要执行同步操作按照约定的格式生成同步操作节点并保存至同步操作队列中

的过程,具体可以分为下载服务器端目录树、构建目录树(包括客户端目录树、服务器端目录树和磁盘目录树)、比对目录树并生成操作队列 3 个部分.我们分别对工作目录中包含 500,1 000,2 000 和 5 000 个文件的生成同步操作的时间开销做了测试,其结果如图 4 所示.

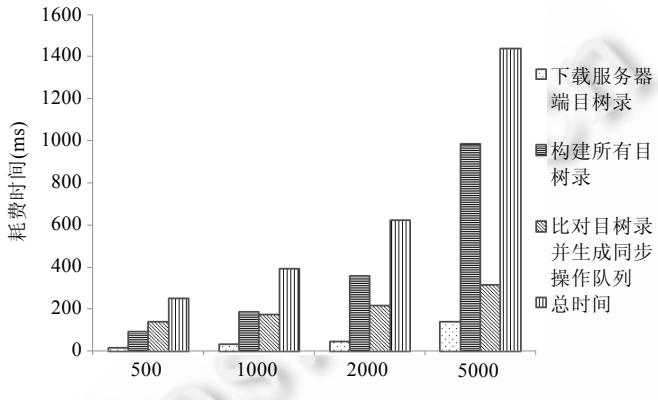


Fig.4 Time cost of each step of generating synchronous operations

图 4 生成同步操作各步骤的时间开销

从图 4 中可以看出:生成同步操作队列的时间开销主要花费在客户端构建目录树上,而下载服务器端目录树和目录树的比对的过程花费的时间很少.包含 5 000 个文件的工作目录只需大约 1.6s 的时间就能完成同步操作队列的生成操作,相对于传输操作(例如上传、下载等)的执行时间来说很少,因此在整个数据同步过程中只占着极小的比例.

此外,同步所花费时间大致随工作目录中的文件个数增长而线性递增,说明基于 DirTree 协议的同步方式具有良好的可扩展性和性能.

3.3.2 数据传输开销

在生成同步操作队列后,客户端将根据此队列中每个同步操作节点中记录的信息,依次向服务器请求并执行相应的同步操作,例如上传、下载等.我们分别对 16MB,32MB,64MB 文件上传下载的各部分时间开销做了测试,其结果如图 5 所示.

从图 5 中可以看出:

- 在上传过程中,最大的时间开销在数据从客户端上传至服务器部分,大约占总开销的 80%;
- 其次是服务器将数据保存至云端,根据文件大小不同所占的比例也不一样,大约占总开销的 10%~20%;
- 数据加解密的开销只占总开销的 6%左右,客户端读取本地文件和计算哈希值占的时间很少,几乎可以忽略不计.

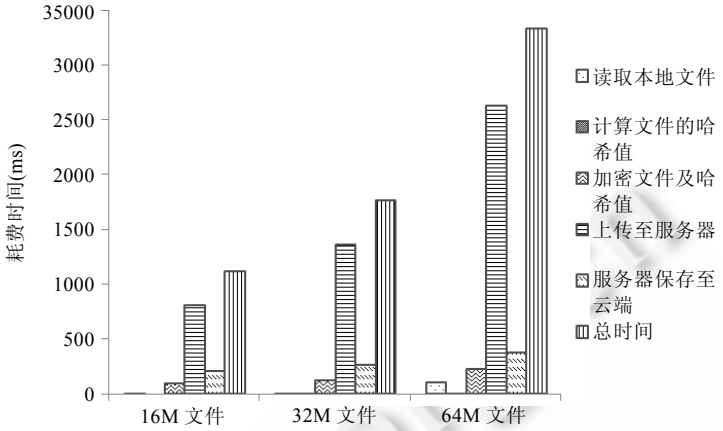
类似地,在下载过程的时间开销中,数据传输占了约 75%,加解密占了约 15%,服务器读数据和客户端写入磁盘占了大约 10%,完整性校验所占的时间几乎可以忽略.

同样大小的文件,其下载速度明显要比上传快,这是因为 CorsBox 系统在文件上传之前对数据做了一定的编码优化,减少了接收方(即服务器)的内存使用,但是带来了编码转换的开销.

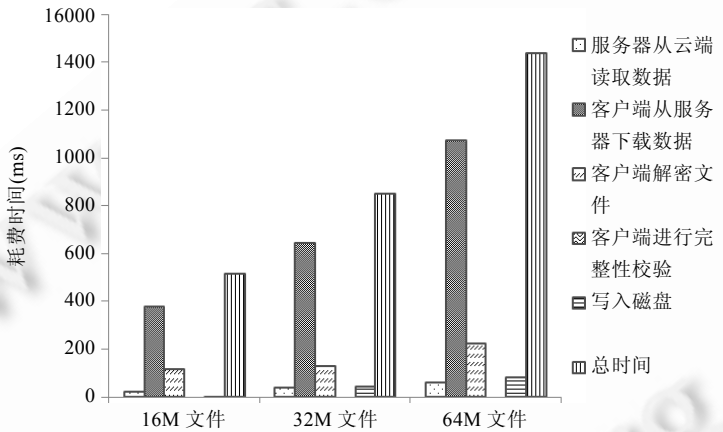
对用户来说,上传操作在系统后台进行,而下载操作直接影响文件使用,因此,CorsBox 系统对上传操作进行编码优化是合理的.

此外,当系统的客户端与服务器之间的连接为速度较慢的 Internet 网络时,受网速的影响整个传输过程所需的时间开销将会变大,因此加解密操作所占的比例也将进一步降低.

总的来说,机密性与完整性保护所带来的额外开销在用户可以接受的范围内.



(a) 上传操作各步骤的时间开销



(b) 下载操作各步骤的时间开销

Fig.5 Time cost of data transmissions

图 5 数据传输的时间开销

3.3.3 权限变更的开销

权限变更的开销测试分为共享目录、添加权限、撤销权限和撤销共享者这 4 个部分:用户 *A* 在其客户端选择共享自己的某个目录,然后添加用户 *B* 对此目录的访问权限,添加成功后再撤销用户 *B* 对此目录的访问权限,最后取消共享此目录.

从表 2 中可以看出:用户共享目录和取消共享的执行速度很快;添加权限过程需要获取待添加用户的公钥,执行速度相对较慢;撤销权限只需在目录元数据的访问控制列表中删除掉该用户对应的项即可,因此速度较快.总的来说,权限变更操作的时间开销在合理的范围之内.

Table 2 Time cost of modifying access permissions

表 2 权限变更的各项时间开销

操作描述	时间开销(ms)
共享目录	634
添加权限	4 338
撤销权限	1 712
取消共享	518

4 相关工作

云计算的诞生为存储系统带来许多新的发展,许多企业也开始搭建自己的云存储平台.目前应用比较广泛的主要有亚马逊的简单存储服务(S3)[2]、RackSpace 的 Openstack[8]以及微软的 Azure[15].云存储平台催生了许多云存储应用系统,网盘系统便是著名应用系统之一.目前比较出名的网盘系统主要有 DropBox[3],SkyDrive[13],SpiderOak[5],Wuala[6],Ubuntu One[16],iDisk[6]等.DropBox 是一个基于商业应用的在线存储系统,底层采用亚马逊的简单存储服务,通过 AES-256 加密算法对数据进行加密存储,提供了数据同步及文件共享等服务.但是,由于 DropBox 的所有密钥均由服务器进行保管,很难真正保障用户数据的机密性.SkyDrive 是微软公司在其云存储平台 Azure 上搭建的网盘系统,和 DropBox 一样,也是由服务器保管密钥,因此,其用户数据的机密性较差.SpiderOak 是一个安全的云存储网盘系统,对外提供数据同步及共享等功能.其特点在于,用户可以自定义一个密钥对数据进行加密,服务器方在任何情况下都不知道用户数据的明文;共享密钥使用文件密钥生成.Wuala 是基于瑞士联邦理工学院研发的安全网盘系统.它和 SpiderOak 一样,将数据加密后再上传至服务器,由用户自己管理文件密钥.但由于 SpiderOak 和 Wuala 均提供了“外链”的数据共享方式与“在线重置密码”的功能,因此用户有理由怀疑其安全性.Ubuntu One 和 iDisk 均为与操作系统相结合的网盘,通过内嵌在操作系统中的方式,为用户提供数据备份等服务(见表 3).

Table 3 Typical online storage systems comparison

表 3 典型的网盘系统对比

网盘系统	是否加密传输	传输与同步协议	密钥管理方式	是否允许数据共享	服务器是否能够查看用户数据	是否能够防止操纵哈希值攻击
DropBox	是	Proprietary	服务器统一管理	是	能	不能
SpiderOak	是	Proprietary	主密钥、密钥派生函数和公私钥管理	是	不能	未知
Wuala	是	Cryptree	通过加密树结构,高效地进行管理	是	不能	未知
SkyDrive	是	WebDav	服务器统一管理	是	能	不能
Ubuntu One	是	Ulstorage	服务器统一管理	是	能	不能
iDisk	否	WebDav	数据未加密	否	能	不能
CorsBox	是	DirTree	文件哈希、目录密钥和用户公私钥三层管理	是	不能	能

针对云存储的安全问题,目前学术界提出了一些改善安全问题的云存储系统,比较典型的有 Tahoe[17],Farsite[18]以及 Corslet[11]系统等.

- Tahoe 是一个分布式安全文件系统,被部署在一个商业的备份服务中,提供访问控制、加密、完整性检查等功能,并采用了纠删码技术进行容错.
- Farsite 通过多副本机制提供文件的可靠性与可用性,提供一个集中式文件服务器的功能,同时可以防止一个拜占庭错误来保证数据的完整性.
- Corslet 是清华大学高性能计算所研发的一个栈式安全存储系统.它可以直接架在已有的存储系统上而不需要对其进行任何修改,同时可以提供端到端的数据机密性、完整性保护和访问权限控制等功能.

但是,这类系统并非针对网盘这一特定的应用,也未能很好地解决网盘系统中存在的安全问题.

文献[7]对目前的主流网盘系统的安全性做了一定的分析,并指出,除了网盘提供商窃取用户数据之外,目前主流网盘系统中存在的一些攻击方式,包括操纵哈希值攻击(hash value manipulation attack)、偷窃宿主 ID 攻击(stolen HostID attack)和直接下载攻击(direct download attack)等.

- 操纵哈希值攻击利用网盘系统通过哈希值来判断文件是否需要上传的漏洞:攻击者只需要获取到某文件的哈希值(例如从“.torrent”文件中获得),然后告知网盘服务器上传该文件(实际上,攻击者并没有此文件的全文);网盘服务器在接到上传请求后,向用户索要此文件的哈希值,并根据此哈希值判定服务器中是否已有此文件,若有,则攻击者不需要上传文件内容(上传操作已完成,网盘系统认为攻击者拥有此文件的全文);待下次同步发生时,攻击者即可从服务器中下载文件的文,从而实现了根据哈希值,获取文件内容的攻击.

- 宿主 ID 是网盘系统将客户端和宿主机绑定生成,在整个系统中能够唯一标识的 ID,用来验证用户的身份.攻击者若通过某种非法的方式偷窃到他人的宿主 ID(例如从中间件中获得),则能够获取此用户的所有上传内容.
- 直接下载攻击是指通过哈希值,直接向网盘服务器请求下载该哈希值所对应的文件,其性质与操纵哈希值攻击相似.

在这 3 类攻击中,操纵哈希值攻击不仅应用广泛,而且为服务器所难以察觉,是网盘系统中最具有威胁的攻击方式.CorsBox 系统用的 DirTree 协议以最后修改时间和文件版本号共同作用,取代哈希值作为文件上传的判定条件,有效地防止了操纵哈希值攻击和直接下载攻击,安全、高效地实现了数据明文与密文之间的同步.用宿主 ID 和密码同时校验、用户自行保管(记住)主密钥的方式,以防止偷窃宿主 ID 攻击;提供了一套多粒度的数据共享与密钥管理分发机制,用户只需保存一个主密钥和自己的私钥就可以为共享数据提供两种密钥粒度的选择,提高了数据的机密性;使用了大数据的断点续传机制,能够支持大数据的高效传输.

5 结 论

本文介绍了一种云存储环境下的安全网盘系统——CorsBox 系统的体系结构及其实现的几个关键技术.CorsBox 系统解决了目前主流网盘系统易遭受攻击和服务提供方不可信的问题,能够在提供数据同步、数据共享、文件版本控制等功能的同时为用户数据提供较强的机密性、完整性保护.测试结果表明,CorsBox 系统生成同步操作的效率很高,因添加安全机制而导致系统额外的性能开销大约在 6%~15%之间,说明 CorsBox 系统在添加安全机制之后仍然具有良好的性能.

References:

[1] Amazon S3. Amazon simple storage service (Amazon S3). http://aws.amazon.com/s3

[2] DropBox. http://www.dropbox.com/business

[3] Kingsoft kuai pan. http://www.kuaipan.cn/

[4] SpiderOak. https://spideroak.com/

[5] Wuala. http://www.wuala.com/

[6] IDisk. http://en.wikipedia.org/wiki/IDisk

[7] Mulazzani M, Schrittwieser S, Leithner M, Huber M, Weippl E. Dark clouds on the horizon: Using cloud storage as attack vector and online slack space. In: Proc. of the 20th USENIX Conf. on Security. 2011.

[8] Openstack. http://openstack.org/

[9] Swift. http://openstack.org/software/openstack-storage/

[10] Rsync. http://en.wikipedia.org/wiki/Rsync

[11] Xue W, Shu JW, Liu Y, Xue M. Corslet: A shared storage system keeping your data private. Science China Information Science, 2011,54(6):1119−1128. [doi: 10.1007/s11432-011-4259-y]

[12] Mahajan P, Setty S, Lee S, Clement A, Alvisi L, Dahlin M, Walfish M. Depot: Cloud storage with minimal trust. In: Proc. of the 9th Conf. on Symp. on Operation Systems Design and Implementation. Berkley: USENIX Association, 2010. 307−322. [doi: 10.1145/2063509.2063512]

[13] Serafini M, Dobre D, Majuntke M, Bokor P, Suri N. Eventually linearizable shared objects. In: Proc. of the 29th ACM SIGACT-SIGOPS Symp. on Principes of Distributed Computing (PODC). New York: ACM, 2010. 95−104. [doi: 10.1145/1835698.1835723]

[14] Skydrive. http://windows.microsoft.com/en-us/skydrive

[15] Windows azure. http://www.windowsazure.com/

[16] Ubuntu one. http://one.ubuntu.com

[17] Wilcox-O'Hearn Z, Warner B. Tahoe: The least-authority filesystem. In: Proc. of the 4th ACM Int'l Workshop on Storage Security and Survivability. New York: ACM, 2008. 21−26. [doi: 10.1145/1456469.1456474]

[18] Adya A, Bolosky WJ, Castro M, Cermak G, Chaiken R, Douceur JR, Howell J, Lorch JR, Theimer M, Wattenhofer RP. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In: Proc. of the 5th Symp. on Operating Systems Design and Implementation. Berkley: USENIX Association, 2002. 1−14. [doi: 10.1145/1060289.1060291]

附中文参考文献:

[3] 金山快盘. http://www.kuaipan.cn/

傅颖勋(1986－),男,湖南永州人,博士生,主要研究领域为云存储安全,可靠性系统及其关键技术.
E-mail: mooncape1986@126.com

罗圣美(1971－),男,高级工程师,CCF 会员,主要研究领域为云计算,移动互联网,通信网络.
E-mail: luo.shengmei@zte.com.cn

舒继武(1968－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络/云存储,存储安全与可靠性,大数据存储,并行处理技术.
E-mail: shujw@tsinghua.edu.cn