云存储中基于可信第三方的安全可问责方案

王 强[1],宗 平[2]
(1. 南京邮电大学 计算机学院,江苏 南京 210003;
2. 南京邮电大学 海外教育学院,江苏 南京 210023)

摘 要:随着云存储的普及和发展,云端数据的安全问题越来越受到人们的关注。针对当存储在云端服务器中的数据文件遭到非法修改或意外损坏时,云存储用户与云端均无法提供使双方信服的凭据进行责任划分的问题,提出了一种基于可信第三方的数据安全可问责方案。该方案以可信第三方为审计的核心与桥梁,在用户与云端任何一方对数据状态持有异议时进行责任追溯。可信第三方针对每次用户数据操作都通过在线状态判断并经相应文件权限认证,只有通过可信第三方在线状态与文件权限审核的用户数据操作才能被系统所认可,并将操作记录保存在双方都无法抵赖的凭据中。实现了利用可信第三方代替用户执行数据审计与问责,可靠并高效地解决了用户对数据状态持有异议但无法追溯的问题。

关键词:云存储;可信第三方;审计;问责;数据安全
中图分类号:TP301 **文献标识码**:A **文章编号**:1673-629X(2017)10-0111-06
doi:10.3969/j.issn.1673-629X.2017.10.024

A Data Security Accountability Scheme with Trusted Third Party in Cloud Storage

WANG Qiang[1], ZONG Ping[2]
(1. College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China;
2. College of Overseas Education, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract: With the popularity and development of cloud storage, the security of the data in the cloud has been paid more and more attention. In view of the problem that the user and the cloud service provider cannot offer convincing credentials to duty partition when the data stored in the cloud has been unlawful modification or accidental damage, a data security accountability scheme based on trusted third party is put forward. It, which takes the trusted third party as the core and bond, traces the responsibility when any party has objection. For every user data operations, trusted third party is through the online status judgment and authenticated by the corresponding file permissions. The user data operations only by the trusted third party online status and user data file permissions audit can be accepted by the system and their records are stored in the credentials which the both couldn't deny. It uses the trusted third party instead of users to audit and account, and solves the problem reliably and efficiently that the user disagree the data state but cannot trace back.

Key words: cloud storage; trusted third party; audit; accountability; data security

1 概 述

云存储是以存储数据和管理数据为核心的云计算系统,是由云计算衍生出来的一种网络存储技术。云存储系统通过网络技术、集群、分布式文件系统、存储虚拟化、存储网络化等技术,将网络中大量各种不同类型的存储设备通过应用软件集合起来协同工作,共同对外提供数据存储和业务访问功能,从而将软硬件资源有限的用户从复杂繁重的数据计算和管理任务中解放出来。云存储的体系结构分层模型参见图 1。

在云存储中,当用户将数据储存到云服务器时,也就意味着失去了对数据的绝对控制权。因此,数据安全是一个不可忽视的问题。云存储系统中数据的安全性可分为存储安全性和传输安全性两部分,每部分均包含数据可用性、数据机密性和数据完整性三个方面。数据可用性是指在一定级别的存储系统环境中,数据必须是可用的。数据机密性是指数据的明文在传输和存储的过程中不能被其他任何用户和云服务提供商(Cloud Service Provider, CSP)访问,只有数据拥有者

收稿日期:2016-11-18 **修回日期**:2017-03-09 **网络出版时间**:2017-07-19
基金项目:教育部专项研究项目(2013116)
作者简介:王 强(1991-),男,硕士研究生,研究方向为云计算、云存储数据完整性;宗 平,教授,研究方向为云计算与数据处理。

和被授权用户才能访问数据明文。数据完整性[1]涉及数据存储时完整性和数据使用时完整性两个方面。数据存储时完整性是指云存储服务提供商是按照用户的要求将数据完整地保存在云端,不能有丝毫的损坏或丢失。数据使用时完整性是指当用户对某个已有权限的数据进行操作时,此数据没有被篡改或伪造。

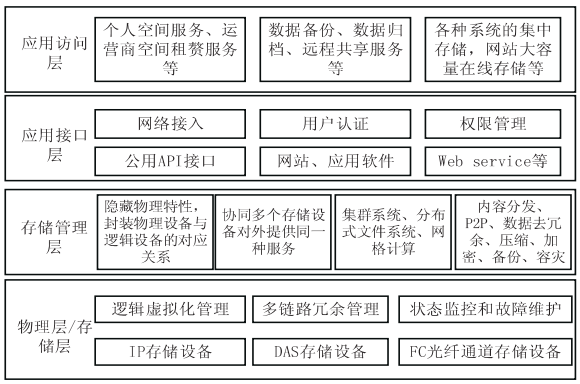

图 1 云存储系统结构模型

数据的完整性是当前用户将数据存储在云端所要考虑的一个重要方面。一方面,存储在云端的数据易被黑客攻击或被云服务提供商的内部人员恶意篡改,并且由于不同用户之间所持有的数据资源通常采用逻辑的方式隔离,因此恶意的攻击者可以通过伪装成合法用户从内部发起攻击,窃取或破坏其他用户的数据。另一方面,云服务提供商并不能保证云端软件或硬件一直处于正常运行状态,因此一些不可避免的因素也会导致数据的损坏。同时,由于缺乏监管,云服务提供商出于利益的考虑可能会隐瞒用户数据的丢失或损坏,甚至泄露用户的敏感数据,并且使用户相信他们的数据仍然被安全完整地存储在云服务器上[2-3]。因此,数据的完整性问题是云存储数据安全方面不可忽视的重要问题之一。

虽然用户在使用云存储服务之前会与云服务提供商订立相应合约,但是合约的履行缺乏实际监督,所以当数据遭到破坏,或者用户对数据的状态提出疑问时,双方均无法提供一个可信的凭据进行责任划分。因此,建立一个完善的数据监督和问责机制来对用户和云端进行行为监督,不仅能够加强用户与云服务提供商之间的信任,还能及时准确的解决云端数据产生的纠纷问题[4-5]。

为了将云服务和问责技术结合起来,在云存储环境下建立了一个完善的问责机制。Andreas Haeberlen[6] 和 Siani Pearson[7] 在云计算的基础上归纳总结出了问责机制的基本需求,并提出了问责机制的设计框架和潜在的技术挑战等。Ryan 等[8] 进一步完善了可信云计算中问责机制的框架,并提出了云问责的生命周期以及三个抽象层。Volkmar 等[9] 提出云问责需要在更加多样的安全机制和更加严格的业务流程中进行。Mainul Mizan 等[10] 提出一种基于时间属性安全问责的大规模可扩展系统。Zou Jun 等[11] 提出一种可问责云服务模型(Accountable Cloud Service,ACS),该模型是由动态逻辑系统扩展而来的可问责的动态逻辑混合系统。

基于上述考虑,文中提出一种基于可信第三方的验证在线状态和权限的可问责方案。该方案以可信第三方作为问责和审计流程的核心,对用户提出的文件操作请求进行在线状态和文件权限验证。同时,在云端将用户提出的相应操作执行完毕后,可信第三方将双方用于确认操作的签名和操作记录保存在凭据中,从而保证了审计结果的不可抵赖性。

2 基于可信第三方的可问责方案

2.1 方案架构设计

提出了一种基于可信第三方(Trusted Third Party,TTP)的可问责系统。一个可靠的可信第三方问责系统需要提供以下三个功能[12-13]:

(1)用户对云端数据的任何操作在可信第三方和云端均有记录;

(2)当出现纠纷时,可信第三方能够准确查找到纠纷责任方,并进行责任判定

(3)用于判断责任方的凭证具有无法抵赖性和无法推卸性。

因此,每次用户、云端和可信第三方交互时,可信第三方都会新建或更新一种用于问责审计时不可抵赖的凭单。

云存储中基于可信第三方的数据完整性问责方案主要在用户(User)、云服务提供商、可信第三方这三个角色之间执行。云服务提供商提供云存储以及周边相关服务。用户使用云存储服务,将文件数据等存储在云端。可信第三方负责监督用户和云端的行为,并记录每次数据操作,当出现纠纷时,可信第三方可出具不可抵赖的问责凭据,并判断责任方。

用户通过浏览器或者其他客户端对云端文件进行操作。可信第三方的服务器数据库中存有云端文件访问控制列表、文件加密解密密钥表、问责凭单列表以及用户临时令牌表。云端存储文件和文件操作记录表等其他数据。同时,对于除文件以外的数据,在传输时均要进行加密,为此每方都持有己方的私钥和另外两方的公钥。

用户每次通过浏览器或其他客户端登录云端时,都会从可信第三方处获得一个临时令牌 token。当用户一段时间未进行数据操作或退出登录时,可信第三方会将该用户对应的临时令牌 token 重置。用户在对

云端数据进行操作时,都会将缓存在客户端的临时令牌加密后发送给可信第三方。可信第三方将收到的 token 解密后与用户临时令牌表中的对应记录进行比对,在匹配成功后,可信第三方会通过文件访问控制表验证用户是否拥有被访问文件的访问权限。只有上述验证全部通过后,用户才能被可信第三方和云端认可并进行后续的操作。当云端对用户的请求进行响应后,用户均会通过客户端对云服务器上请求操作的数据进行在线检查。用户确认操作数据成功后会发送确认信息给可信第三方进行用户方确认。

每次用户对文件进行操作后,可信第三方都会生成新密钥对文件进行加密,并更新文件密钥版本。因此在进行数据共享时,可信第三方只需维护文件访问控制列表,无需管理文件密钥与用户的关系。同时,文件密钥表的使用,使得密钥的生成算法被独立分割出来,即加密和密钥生成算法不依赖于系统,相对独立。因此,系统可使用多种密钥生成算法来生成密钥。

用户和云端对数据的每次操作都会生成问责凭单项,并保存在可信第三方的凭单表中。当用户提出问责或双方出现纠纷时,可信第三方将根据云端的文件操作记录和本地的凭单来进行责任判断。基于可信第三方的数据完整性问责架构如图 2 所示。

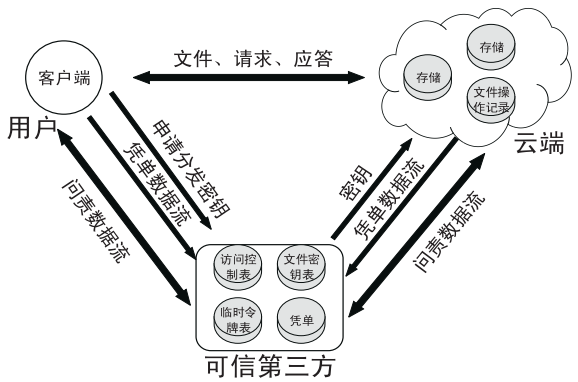

图 2　基于可信第三方的可问责架构

2.2　凭单设计

可信第三方的凭单表(Credential Table,CT)中记录了云端每个文件的凭单链表(Credential List,CL),其结构如图 3 所示。

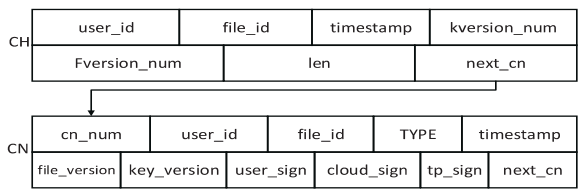

图 3　凭单链表

这些凭单链表和文件是一一对应的关系。凭单链表的表头节点(Credential Head,CH)保存了该凭单链表的基本信息。user_id 为文件拥有者的唯一标识;file_id 为该凭单链表对应文件的唯一标识;timestamp 为最近一次操作文件的时间戳;kversion_num 为该文件的密钥版本数,即密钥最大版本号;fversion_num 为该文件版本数,即文件最大版本号;len 为 CL 不包括 CH 的剩余长度;next_cn 为该表头节点的下一个凭据节点(Credential Node,CN)的编号。凭据节点中保存了每一次文件操作的基本信息,cn_num 为该节点的编号;user_id 为操作文件用户的唯一标识;file_id 为文件的唯一标识;TYPE 为文件操作类型,该类型为枚举型,包含 CREATE、DETELE、READ、UPDATE 四种类型;timestamp 为此次操作的时间戳;key_version 和 file_version 分别对应操作完成后的密钥版本号和文件版本号;user_sign、cloud_sign、tp_sign 分别为用户签名、云端签名、可信第三方签名,用来保证凭据的不可抵赖性和完整性;next_cn 为该凭据节点的下一个凭据节点的编号。

2.3　用户登录流程

为了防止云端内部伪装成合法用户登录系统,从而欺骗可信第三方进行数据操作,所采用方案的用户登录管理权限由可信第三方来执行。用户在登录时,将登录数据发送给可信第三方进行审核。审核通过后,可信第三方会将用户登录数据按照事先与云端商量好的格式进行打包,并发送给云端。云端记录用户登录状态,并返回用户此次在线期间内数据交互时需要的临时数据。最后,可信第三方将云端返回的数据连同临时令牌 token 和用户私钥一起返回给用户。用户将这些数据信息进行缓存或保存到临时文件中。

2.4　数据交互流程

2.4.1　基础交互架构

User、CSP 和 TTP 三方进行数据交互时,首先 User 会向 CSP 发送文件操作请求,如果此时操作类型为“新建”或“更新”,则需同时上传文件。与此同时,User 会向 TTP 发送密钥申请请求以及在线标识。TTP 在收到 User 发来的数据后验证 User 在线状态,并进行密钥表操作,随后向 CSP 发送密钥和其他附加数据。CSP 收到密钥后进行相应文件操作,并告知 User 操作结果,同时向 TTP 发送表示操作完成的签名。User 收到 CSP 的文件操作完成后进行检查,并向 TTP 发送表示检查结果的签名。具体流程如图 4 所示。

2.4.2　创建新文件流程

User 在创建新的文件对象时,首先向 CSP 和 TTP 发送通过私钥加密后的请求。向 CSP 发送新文件、user_id、file_id、timestamp,向 TTP 发送的是申请第三方向云端发送加密密钥的请求,该请求包括值为“CREATE”的 TYPE、在线标识 token 以及 file_id、user_id、timestamp。

TTP 收到 User 发来的发送密钥请求后,利用 User

公钥对请求进行解密,随后识别 user_id 和 token 是否匹配。匹配通过后,TTP 新建与 user_id 和 file_id 对应的文件密钥表,并将利用密钥生成算法生成的密钥对保存在该表中。最后,TTP 利用己方私钥将加密密钥与 file_id 进行加密,向 CSP 发送。

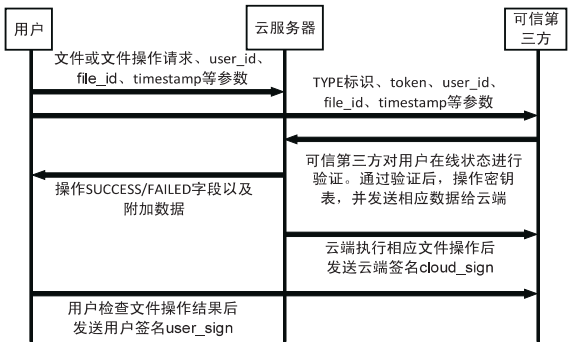

图 4 基础交互架构

CSP 在收到 User 和 TTP 发来的数据后,先对 TTP 发来的数据进行解密,然后利用密钥对 User 发来的文件进行加密和存储。随后向 User 发送“SUCCESS_CREATE”字段表示新建文件对象成功,并将 user_id、file_id 以及初始操作编号“1”加密作为签名发送给 TTP。如果 CSP 新建文件失败,则发送“FAILED_CREATE”字段给 User,并将 user_id、file_id 以及失败编号“0”加密作为签名发送给 TTP。

User 收到 CSP 的创建文件对象响应后,直接通过浏览器或其他客户端检查文件对象是否新建成功。如果检查通过,则将 user_id、file_id 以及初始操作编号“1”加密作为签名发送给 TTP;否则,将 user_id、file_id 以及未通过检查标识“0”加密作为签名发送给 TTP。

TTP 收到 User 和 CSP 发来的签名后,根据与此次操作相关的 file_id、user_id、timestamp、TYPE、user_sign、cloud_sign,创建一个只含一个 CN 的 CL,其中 CH 的 kversion_num、fversion_num、len 和 next_cn 置 1,CN 的 key_version、file_version、cn_num 置 1,next_cn 置空,tp_sign 中保存的是利用 TTP 私钥加密 user_id、file_id、key_version 和 file_version 生成的 TTP 签名。

2.4.3 读取文件流程

User 在读取文件对象时,同时向 CSP 和 TTP 发送私钥加密过的请求。向 CSP 发送的读文件请求包括 user_id、file_id 和 timestamp,向 TTP 发送的是申请 TTP 向云端发送解密密钥的请求,该请求包括值为“READ”的 TYPE、在线标识 token 以及 file_id、user_id、timestamp。

TTP 在收到 User 的发送密钥请求后,首先验证 User 是否拥有对应文件的读权限,并匹配 User 的 token 是否正确。通过验证后,TTP 在对应 file_id 的 CL 中查找最新 CN 的 cn_num 和 key_version,并在文件密钥表中查找该 key_version 的解密密钥。同时,TTP 利用密钥生成算法生成一对新的密钥对保存到文件密钥表中。最后,TTP 将该旧解密密钥、新解密密钥、file_id 以及最新 CN 的 cn_num 进行加密后发送给 CSP。

CSP 在收到 User 发来的读取请求和 TTP 发来的数据后,对文件进行解密。待解密成功后,将文件、最新 CN 的 cn_num 和“SUCCESS_READ”字段一并发送给 User 所使用的客户端。随后利用新的加密密钥将解密完成后的文件进行重新加密。同时,CSP 将 user_id、file_id 和 cn_num+1 加密作为签名发送给 TTP。如果操作失败,则向 User 发送“FAILED_READ”字段,并将 user_id、file_id 和失败编号“0”加密作为签名发送给 TTP。

User 收到 CSP 的读取文件响应后,直接通过客户端检查文件对象是否是自己期望的内容和版本,如果检查通过,User 将 user_id、file_id 和 cn_num+1 加密作为签名发送给 TTP。否则,将 user_id、file_id 以及未通过检查标识“0”加密作为签名发送给 TTP。

TTP 收到 User 和 CSP 发来的签名后,根据与此次操作相关的 file_id、user_id、timestamp、TYPE、user_sign、cloud_sign 生成 CN,该 CN 的 key_version 和 file_version 均为最新的版本号。同时,该 CN 中的 tp_sign 保存的是利用 TTP 私钥加密 user_id、file_id、key_version 和 file_version 生成的 TTP 签名。最后,TTP 将该 CN 插入到对应 CL 的 CH 后,更新 CH 中的 timestamp、kversion_num、fversion_num、len 和 next_cn。

2.4.4 更新文件流程

User 在更新文件对象时向 CSP 和 TTP 发送通过私钥加密后的请求,向 CSP 发送的请求包括更新的文件、file_id、user_id、timestamp,向 TTP 发送的是申请第三方向云端发送加密密钥的请求,该请求包括值为“UPDATE”的 TYPE、在线标识 token 以及 file_id、user_id 和 timestamp。

TTP 在收到 User 发来的发送密钥请求后,利用 User 公钥对请求进行解密,随后验证 User 是否拥有对应文件的写权限,并识别 user_id 和 token 是否匹配。匹配通过后,TTP 在对应 file_id 的 CL 中查找最新 CN 的 cn_num,同时,TTP 利用密钥生成算法生成一对新的密钥对保存到文件密钥表中。最后,TTP 将新加密密钥、cn_num、file_id 进行加密后发送给 CSP。

CSP 在收到 User 发来的更新文件和 TTP 发来的数据后,对原文件进行更新和重新加密。随后向用户发送“SUCCESS_UPDATE”字段以及最新 CN 的 cn_num,同时将 user_id、file_id 和 cn_num+1 加密作为签名发送给可信第三方。如果 CSP 操作失败,则向 User 发送“FAILED_UPDATE”字段,并将 user_id、file_id 以

及失败标识“0”加密作为签名发送给 TTP。

User 收到 CSP 的更新响应后,直接通过客户端检查文件对象是否更新成功。如果检查通过,User 将 user_id、file_id 和 cn_num+1 加密作为签名发送给 TTP,否则,将 user_id、file_id 以及未通过检查标识“0”加密作为签名发送给 TTP。

TTP 收到 User 和 CPS 发来的签名后,根据与此次操作相关的 file_id、user_id、timestamp、TYPE、user_sign、cloud_sign 生成 CN,该 CN 的 key_version 和 file_version 均为最新的版本号。同时,该 CN 中的 tp_sign 保存的是利用 TTP 私钥加密 user_id、file_id、key_version 和 file_version 生成的 TTP 签名。最后,TTP 将该 CN 插入到对应 CL 的 CH 后,更新 CH 中的 timestamp、kversion_num、fversion_num、len 和 next_cn。

2.4.5　删除文件流程

User 在删除文件对象时,同时向 CSP 和 TTP 发送删除文件请求。向 CSP 发送的删除文件请求中包括 user_id、file_id、timestamp,向 TTP 发送的是判断是否有删除文件的权限的请求,该请求包括值为“DETELE”的 TYPE 标识、在线标识 token 以及 file_id、user_id 和 timestamp。

TTP 在收到 User 的请求后,利用 User 公钥对请求进行解密,随后判断 User 对文件是否有写权限,并识别 user_id 和 token 是否匹配。匹配通过后,TTP 向云端发送允许删除文件的“DELETE_ALLOW”字段、cn_num、file_id。

CSP 收到 User 删除请求和 TTP 发来的“DELETE_ALLOW”字段后,删除文件,并向 User 发送“SUCCESS_DELETE”字段以及最新 CN 的 cn_num,同时将 user_id、file_id 和 cn_num+1 加密作为签名发送给 TTP。如果操作失败,则向 User 发送“FAILED_DELETE”字段,并将 user_id、file_id 和失败编号“0”加密作为签名发送给 TTP。

User 收到 CSP 发来的删除文件成功的消息后,通过客户端检查是否删除成功,如果通过检查,则将 user_id、file_id 和 cn_num+1 加密作为签名发送给 TTP,否则,将 user_id、file_id 以及未通过检查标识“0”加密作为签名发送给 TTP。

TTP 收到 User 和 CSP 发来的签名后,根据与此次操作相关的 file_id、user_id、timestamp、TYPE、user_sign、cloud_sign 生成 CN,该 CN 的 key_version 和 file_version 均为最新的版本号。同时,该 CN 中的 tp_sign 保存的是利用 TTP 私钥加密 user_id、file_id、key_version 和 file_version 生成的 TTP 签名。最后,TTP 将该 CN 插入到对应 CL 的 CH 后,更新 CH 中的 timestamp、kversion_num、fversion_num、len 和 next_cn。

2.5　审计和问责流程

当 User 对文件内容或版本有异议时,可向 TTP 提出问责审计请求。TTP 首先会判别 User 是否具有权限对文件进行读或写。当 User 权限不够时,TTP 拒绝此次审计请求,并告知 User 原因。TTP 还会判断 User 申请审计的文件版本是否合理,如果申请的版本小于对应凭单链表中记录的版本,则拒接此次审计请求,并告知 User 原因。

TTP 要求 CSP 提供用户请求审计的相应版本文件以及云端记录的该文件的操作历史。如果 CSP 无法提供文件或操作记录的任何一个,则判别为 CSP 责任,并告知 User 原因。

TTP 根据文件对应版本的内容,以及 CL 中和 CSP 提供的文件操作记录进行责任判别。通过 CL 长度与文件操作记录的数量判别文件是否被其他非法用户进行额外操作;通过检查 CN 中的 user_sign、cloud_sign、tp_sign 判别各方是否承认对应操作成功;通过检查文件内容判别文件是否在逃避记录的情况下被修改。审计问责流程如图 5 所示。

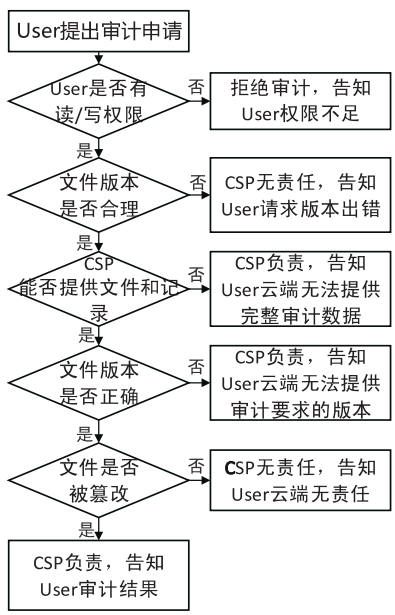

图 5　审计和问责流程

3　安全性分析

云存储可问责系统主要关注导致数据文件处于某个状态的缘由,即对数据文件的各种操作。在面临安全问题时,可问责系统能够根据对文件的操作记录进行审计和问责。表 1 列出了系统可能面临的安全问题以及审计方法。

(1)云端执行用户操作失败或返回错误的响应码。

云端收到用户的操作请求后,进行一系列的数据处理,随后将处理结果提供给用户进行检查。只有通过用户检查确认,此次数据操作才被认可。同时,可信

第三方会将用户的确认信息记录在凭据中,以便后续审计使用。

表 1 可问责系统面临的安全问题和审计方法

安全问题	审计方法
云端执行用户操作失败	用户可直接发现
云端返回错误的响应码	用户可直接发现
用户否认文件的创建、删除、更新	第三方审计
云端否认文件的创建、删除、更新	第三方审计
非法的创建、删除、更新	第三方审计
云端模拟用户登陆系统	可信第三方控制
非法的读取文件	可信第三方控制

(2)用户或云端否认文件的创建、删除、更新。

可信第三方保存的凭单链表中保存了用户与云端每次交互生成的数据信息。用户签名、云端签名以及可信第三方签名更是三方均不可抵赖的确认凭据。同时,当云端数据出现了凭单链表中无法检查到的更改时,可认为数据被非法用户更改,为云端责任。

(3)非法创建、删除、更新。

若一个访问者不在被操作文件的访问控制列表中,但仍能对文件进行读或写,则该访问者可认为是非法用户。非法用户在对数据进行操作时,即使在云端和可信第三方中均没有留下操作痕迹,但在审计阶段对文件内容进行检查时,仍可发现文件被篡改,因此可判断为云端责任。

(4)非法登陆系统。

为了防止云端模拟用户登录系统,系统的登录权限由可信第三方进行管理控制,云端不持有用户的登录密钥。

(5)非法读取文件。

可信第三方中保存并控制文件的读写权限。正常用户向云端和可信第三方发起文件读取请求时,都会被可信第三方进行权限检查。同时,为了防止非法用户绕过可信第三方对云端数据进行直接读取,云端将数据采取一次一密的加密方式。即使非法用户获取到解密密钥,但下次读取时数据的密钥对已经更换,从而保证了云端数据的安全性。

4 结束语

随着云计算的发展,云存储的安全问题必然受到更多的需求和关注。在云计算数据存储环境下提出了可信第三方的概念,利用可信第三方监督云端与用户之间的数据交互,并作为中间方进行公平公正的审计与问责。方案中使用一种不可更改的凭单链表来记录用户对云端数据的操作。链表由可信第三方保存,用户和云端均无权修改,从而保证了凭据的安全性。同时,方案提出了一种基于用户在线的凭单生成协议,在每次进行数据交互时,可信第三方都会对用户的在线状态和 token 进行验证,只有用户在线且持有正确的 token 时,可信第三方才认可用户对数据的操作。可信第三方仅负责生成、保存和发送密钥,对于具体的密钥生成算法并没有严格限制,大大增强了该方案的可用性和可扩展性。

参考文献:

[1] 冯登国,张 敏,张 妍,等. 云计算安全研究[J]. 软件学报,2011,22(1):71-83.

[2] Ateniese G,Burns R,Curtmola R,et al. Remote data checking using provable data possession[J]. ACM Transactions on Information and System Security,2011,14(1):12.

[3] Wang Q,Wang C,Li J,et al. Enabling public variability and data dynamics for storage security in cloud computing[C]// Proceedings of ESORICS. [s. l.]:[s. n.],2009:355-370.

[4] Riedel E,Kallahalla M,Swaminathan R. A framework for evaluating storage system security[C]//Proceedings of the conference on file and storage technologies. Berkley:USENIX Association,2002:15-30.

[5] Kamara S, Lauter K. Cryptographic cloud storage financial cryptography and data security[C]//Proceedings of the 14th international conference on financial cryptography and data security. Berlin:Springer,2010:136-149.

[6] Haeberlen A. A case for accountable cloud[J]. ACM SIUOPS Operating Systems Review,2010,44(2):52-57.

[7] Pearson S. Toward accountability in the cloud[J]. IEEE Internet Computing,2011,15(4):64-69.

[8] Ko R K L,Bu S L,Pearson S. Towards achieving accountability,auditability and trust in cloud computing[C]//Proceedings of international conference on ad-vances in computing and communications. [s. l.]:[s. n.],2011:432-444.

[9] Sendor J,Lotz V,de Oliveira A S. Control as a means towards accountable services in the cloud[J]. Computer Systems Science and Engineering,2013,28(6):377-386.

[10] Mizan M,Rahman M L,Khan R,et al. Accountable proof of ownership for data using timing element in cloud services [C]//International conference on high performance computing and simulation. [s. l.]:IEEE,2013:57-64.

[11] Zou J,Wang Y,Orgun M A. Modeling accountable cloud services based on dynamic logic for accountability[J]. International Journal of Web Services Research,2015,12(3):48-77.

[12] Yao J H,Chen S P,Wang C,et al. Accountability as a service for the cloud[C]//Proceedings of IEEE international conference on services computing. [s. l.]:IEEE,2010:81-88.

[13] Yemerefendi A R,Chase J S. Strong accountability for network storage[J]. ACM Transactions on Storage,2007,3(3):11-33.