一种基于移动Agent的云端(Cloud-P2P)数据复合销毁机制

徐小龙[1,2] 龚培培[1] 章 韵[1] 毕朝国[3]
(南京邮电大学计算机学院 南京 210003)[1]
(中国科学院软件研究所信息安全国家重点实验室 北京 100190)[2]
(南京财经大学江苏省现代服务业研究院 南京 210003)[3]

摘 要 云端融合计算(Cloud-P2P)融合了云计算与对等计算环境的所有节点资源,实现了最大范围的协作与资源共享。数据销毁机制是保障用户数据的安全性和可控性的重要措施之一,然而云端计算环境本身的特性也给数据的有效销毁带来了困难。针对云端数据存储系统对数据的主动销毁、定时销毁和自销毁等复合需求,提出一种基于移动Agent的数据复合销毁机制,该方法不依赖第三方,利用移动Agent技术实现对过期、废弃型数据及时、有效、灵活的销毁,并在恶意主体对数据实施攻击时主动实施防御性数据销毁,有效增强了用户数据的安全性。针对节点上数据的具体销毁,还提出一种新颖的"数据折叠"的数据覆写方法,它充分利用数据本身进行销毁,有效降低了系统的开销。

关键词 云计算,对等计算,云端计算,数据销毁,移动Agent,数据折叠
中图法分类号 TP309.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.10.029

Mobile-agent-based Composite Data Destruction Mechanism for Cloud-P2P

XU Xiao-long[1,2] GONG Pei-pei[1] ZHANG Yun[1] BI Chao-guo[3]
(College of Computer, Nanjing University of Posts & Telecommunications, Nanjing 210003, China)[1]
(State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[2]
(Jiangsu Institute of Modern Service Industry, Nanjing University of Finance and Economics, Nanjing 210003, China)[3]

Abstract Cloud-P2P combines the resources of all nodes of cloud computing and peer-to-peer computing to achieve the largest collaboration and resource sharing. The data destruction mechanism is one of the important measures to protect users' data security and controllability, which is difficult for Cloud-P2P systems. In order to meet the requirement of data destruction in Cloud-P2P storage systems, a composite data destruction mechanism based on mobile agent was put forward, which can make the expired, waste data destructed effectively, as well as defend those malicious attacks on data. In order to effectively destruct data on one node with low cost, a novel data destruction method was proposed, which realizes the data destruction by data folding.

Keywords Cloud computing, Peer-to-peer computing, Cloud-P2P computing, Data destruction, Mobile agent, Data folding

1 引言

云存储(Cloud Storage)系统提供基础的存储资源池,将多个物理设备上的存储资源虚拟为统一的存储资源池,基于资源随需分配、随需扩展的弹性存储策略,以供多租户(Multi-tenancy)共同使用[1,2]。

目前基于集中化资源管理体系的云存储存在一系列内生性问题:过度强调服务器端资源,忽视网络边缘节点能力;过度集中导致的更大的安全风险;云存储数据中心的集中部署导致设备密集运行的服务质量(Quality of Service, QoS)、能耗和制冷难题;服务的高峰期间出现服务质量难以保证和低谷期间资源浪费等问题。

本文提出了一种云端融合计算(Cloud-P2P Computing)架构[3],即将网络核心的云数据中心(Cloud Data Center)和网络边缘的终端节点上的各类资源有机聚合成更大规模的资源池,以供上层应用调用。云端数据存储系统中蕴含的节点及可用资源范围更广、数量更大,更具可伸缩性,更易满足用户日益增加的存储需求,能消除性能瓶颈,实现负载平衡和多副本冗余备份,降低数据中心能耗和成本以及避免集中数据存储带来的安全风险等,有效解决了上述问题。但是引入网络边缘节点资源的云端存储模式也带来了一系列问题,特别是数据安全难以保障的问题。

数据安全不仅包括数据加密、访问控制、数据备份及恢复,还包括数据销毁。所谓数据销毁[4],是指采用技术手段将计算存储设备中的数据予以彻底消除,避免非授权用户利用残留数据恢复原始数据信息,以达到保护用户数据私密性的目的。

到稿日期:2014-10-24 返修日期:2015-01-26 本文受国家自然科学基金资助项目(61202004,61472192),教育部科技发展中心网络时代的科技论文快速共享专项研究资助课题(2013116),江苏省高校自然科学研究计划资助项目(14KJB520014)资助。
徐小龙(1977—),男,博士,教授,主要研究方向为分布式计算、移动计算、信息安全技术等,E-mail:xuxl@njupt.edu.cn;龚培培(1989—),女,硕士生,主要研究方向为基于网络的计算机软件和信息安全技术等;章 韵(1963—),男,博士,教授,主要研究方向为计算机应用、网络计算和软件定义网络技术等;毕朝国(1978—),男,硕士,讲师,主要研究方向为计算机软件、Web技术和物联网技术等。

本文的主要贡献包括：(1)引入移动 Agent 技术，使得用户可以有效实现数据的云端托管、管理和控制；(2)提出综合采用主动销毁、定时销毁和防御型销毁的云端存储系统复合数据销毁策略来实现数据的有效销毁；(3)提出“数据折叠”的数据覆写新方法，充分利用数据自身的序列进行覆写，实现低开销数据销毁。

2 相关工作

2007 年 Perlman[5] 提出了可信删除(Assured Delete)的机制，即通过建立第三方可信机制，以用户操作或时间作为删除条件，在超过规定的时间后自动删除数据密钥，从而使得任何人都无法解密出数据明文，以达到数据销毁的目的。FADE 系统[6]设计了一种基于策略的文件删除方法，以实现云存储系统中存储数据及其副本的确定性销毁，基本思想是每个文件都对应一条或多条访问策略，不同的访问策略之间可以通过逻辑“与”和逻辑“或”组成混合策略，文件的访问者只有符合访问策略的条件时才能解密出数据明文；该方案中有第三方参与，是一种集中式的管理，存在因密钥第三方不可信而出现误删、错删或漏删的安全隐患。Vanish 系统[7]从销毁密钥的角度提出了一种基于 DHT 网络[8,9]的数据销毁机制，即在上传数据之前将数据进行加密，然后将密钥经门限密码处理后随机分发到 DHT 网络中，数据授权访问者只有获得超过 k ($k \leqslant n$)份密钥才能够正常地解密；当授权时间到达时，所有的密钥将自动销毁，使得在超过预设时间后任何人都无法恢复数据明文。然而，攻击者可以通过嗅探攻击或跳跃攻击获取到足够的密钥分片从而重构出密钥。为了解决这些问题，文献[8]对 Shamir 秘密共享算法进行改进，扩展密钥份数的长度以抵抗 Vanish 系统中存在的跳跃攻击；对于嗅探攻击的抵抗则通过公钥加解密的方式实现。文献[10]在 Vanish 系统的基础上，从销毁密钥和数据的角度将密钥和部分密文数据一起分发到 DHT 网络中，使得攻击者暴力破解不完整密文数据所需的密钥空间增大，从而增加攻击的难度和代价；由于该方案将部分密文也分发到网络中，因此增加了网络通信开销。文献[8,10]都只考虑了单个密钥和少量数据的确定性销毁，不能直接应用于海量数据存储系统，用单个密钥加密全部数据不能对数据进行细粒度的管理和操作，不能按需提供服务。文献[11]对存储前的用户数据加密及密钥管理进行研究，借鉴结构化层次密钥管理的思想，提出了一种适于云存储系统的数据销毁方法，采用基于 hash 函数的密钥派生树生成和管理密钥，从而有效减少了数据拥有者所需维护的密钥数量及暴露给外部的密钥数量。文献[12]在此基础上，对大数据分块加密，将密钥和部分数据存储在云端，利用 DHT 网络的动态变化特性确保密钥在授权时间到达后自动从网络中消失，从而实现数据销毁。

3 基于 Cloud-P2P 的云端数据存储模型

3.1 模型架构

基于 Cloud-P2P 的云端数据存储模型如图 1 所示，系统中包含主管理节点(Master Server，MS)、云服务器节点(Cloud Server，CS)和边缘节点(Edge Node，EN)这 3 类节点。有存储需求的合法用户可将需要系统代为存储和管理的文件(File)切分为若干个数据块(Block)，然后上传到云端系统中，并按照一定的策略(Policy)分散存储在若干 CS 节点和 EN 节点上。MS 节点将承担管理和维护系统中所有用户、节点及数据的属性、状态和活动信息，协调整个系统的正常运转；CS 节点和 EN 节点将负责实际的数据存储和本地的资源管理；用户将维护自己拥有的且已经被托管的数据属性信息。

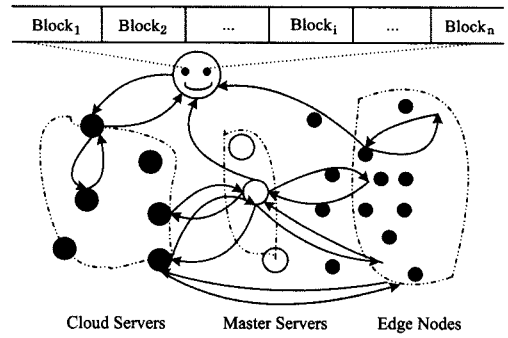

图 1 基于 Cloud-P2P 的云端数据存储模型架构

3.2 问题分析

云端数据存储系统蕴含的节点及可用资源范围更广、数量更大，更具可伸缩性，更易满足用户日益增加的存储需求，能消除性能瓶颈，实现负载平衡和多副本冗余备份，降低数据中心能耗和成本以及避免集中数据存储带来的安全风险等。但是引入网络边缘节点资源的云端存储模式也给云端数据销毁带来了如下一些问题。

(1)终端节点的动态性：EN 节点分属于不同的用户，节点具有自治性、异构性、动态性，可随意加入和退出云端存储计算环境。用户数据存储在 EN 节点上，难以保证用户发出销毁指令时存储用户数据的 EN 节点在线。

(2)节点资源的多租客共享：用户存储在云端节点的数据被多个客户访问共享，而这些租客对数据的操作(如下载等)也会对销毁造成影响。例如租客 A 在下载云端数据 B 一段时间后再次向云端上传该数据，而在这之前数据拥有者已经通知云端销毁该数据，类似这样的情况会造成云端数据销毁得不彻底。

(3)恶意节点或租客的攻击：EN 节点可能是脆弱的(Vulnerable)甚至是恶意的(Malicious)。部署到 EN 上的数据可能被该节点本身或是其上的病毒、木马等恶意代码窃取、篡改、删除，难以保障数据销毁的真实性；或者，当数据拥有者发出销毁指令之后，恶意节点(甚至云提供商本身)或租客返回假消息，即在节点数据没有完全删除的情况下，告知销毁指令发出者“销毁成功”；甚至，恶意租客发出销毁指令要求销毁云端数据，而数据上传者并不希望该数据被销毁。因此，销毁指令和返回信息的合法性也是要解决的问题之一。

(4)用户资源的操作失控：即用户需要确保销毁的数据是自己的数据及副本，不能将存储在同一节点上的其他用户的数据销毁，这个问题也是值得考虑的。

3.3 移动 Agent

为了实现云端存储系统中数据的存储与处理，并着重解决系统中数据远程销毁可能出现的各种问题，引入移动 Agent 来完成系统中的安全、稳定、可靠的数据存储、处理和销毁等任务。

移动 Agent[13]可被狭义地定义为一种用于分布式系统的粗粒度软件实体；Agent 有效地封装了方法、数据、属性和状态等信息，可以自主地在网络各节点间迁移，并通过原语性(Primitive)操作与其它软件实体进行社会性交互和协同，并完成特定的任务。

如图 2 所示，节点 $Node_A$ 上的 $Agent_1$ 可以孵化(Spawn) $Agent_2$。$Agent_1$ 称为 $Agent_2$ 的父 Agent，$Agent_2$ 称为 $Agent_1$ 的子 Agent，即 $Agent_1$ 和 $Agent_2$ 构成一种父子关系。同一个父 Agent 创建的多个子 Agent 之间则构成兄弟关系，成为兄弟 Agent。本文将节点系统初始化时本地用户创建的第一个 Agent 称为根 Agent(Root Agent，RA)。

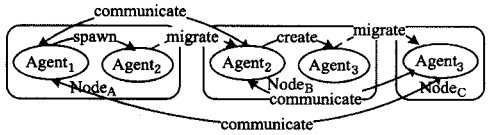

图 2　Agent 孵化、迁移及通信示意图

$Agent_2$ 被孵化后，可按一定的策略自主迁移(Migrate)到节点 $Node_B$ 上；$Agent_2$ 在节点 $Node_B$ 完成相关的任务后，可进一步孵化它的子 Agent($Agent_3$)，并将 $Agent_3$ 迁移到节点 $Node_C$ 上。Agent 的迁移本质上是将迁移前所在节点上的 Agent 序列化后基于 Agent 传输协议(Agent Transfer Protocol，ATP)发送到目标节点上，并将迁移前所在节点上的 Agent 销毁。

同一节点上的 Agent 之间以及不同节点上的 Agent 之间均可基于 Agent 通信语言(Agent Communication Language，ACL)进行通信(Communicate)，以传输消息、数据及进行协作。

定义 1　移动 Agent 是指能够有效实现携带代码、数据和状态进行上传、存储、销毁等需求的 Agent。

移动 Agent 与其执行容器 Agency 相互配合，可有效保障 Agent 及其代码和数据等信息和所处的节点双重安全。Agency 为 Agent 的运行提供通信功能(Communication)、注册服务(Registration Service)、管理功能(Management)、迁移服务(Migration Service)、持久化机制(Persistence Service)和安全保障模块(Security Module)。引入 Agent 机制之后，所有数据的传输和处理都可由 Agent 来协作完成。云端计算环境中的移动 Agent 与 Agency 的体系结构如图 3 所示。

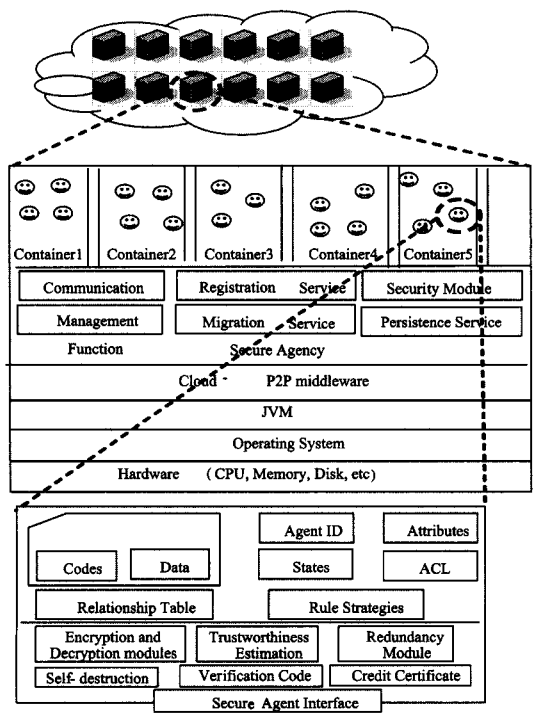

图 3　云端计算环境中的移动 Agent 与 Agency

对于云端计算环境中每个计算节点而言，节点系统的最底层为硬件设施层，其上为操作系统层；由于目前的云系统和 Agent 大都采用 Java 技术，因此系统中设置了 Java 虚拟机(Java Virtual Machine，JVM)层；JVM 之上的是云端系统中间件(Cloud-P2P middleware)层；Agency 将为源于同一用户的 Agent 创建同一个容器(Container)实例，作为 Agent 的直接运行空间，从而能将不同用户的 Agent 有效隔离，避免相互干扰甚至攻击。

移动 Agent 分为 3 个部分：(1)服务主体模块，指移动 Agent承担的数据处理工作，封装了服务代码和数据，包括 Agent的初始化程序和事务处理程序，产生实际执行动作；(2)属性状态等附属模块，包含了 Agent ID(Agent 系统标识)、属性(Attributes)(包括 Agent 来源、创建时间等信息)、状态(States)(记录了 Agent 执行过程中的当前状态，保存数据处理结果，实现跨平台的持续运行)、ACL、关系表(Relationship Table)(Agent 与其父、子、兄弟 Agent 的链接图)和规则策略集(Rule Strategies)；(3)安全可信保障模块，包括加解密模块(Encryption and Decryption modules)、验证码(Verification Code)(负责保障 Agent 本身在传输或运行过程中的自身完整性)、冗余模块(Redundancy Module)、信任评估(Trustworthiness Estimation)(负责评估节点与执行可信性)、自销毁(Self-destruction)(负责将 Agent 自身任务执行代码与数据等私密信息完全销毁)、信任证(Credit Certificate)(负责进入 Agency时提供身份认证与对本地资源说明等情况)和安全Agent接口(Secure Agent Interface)(Agent 与外界通信的中介，防止外界对 Agent 的非法访问)。

系统中的所有 Agent 及其所在的运行容器均需要进行标识。Agent 的全局唯一性标识可由其创建者标识和创建时赋予的本地局部唯一序列号联合构成：$OwnerID \mid AgentID$；Agent运行容器的全局唯一性标识可由节点标识和节点 Agency 创建容器时赋予的本地局部唯一序列号联合构成：$NID \mid ContainerID$。这种设计使得系统易于追踪定位任何迁移到异地的 Agent。

3.4　基于 Agent 的云端数据存储系统模型

如图 4 所示，本文利用移动 Agent 技术完成云端数据存储系统中数据的上传、认证、授权、销毁等工作。

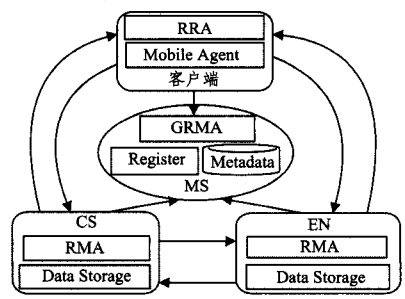

图 4　基于 Agent 的云端数据存储系统模型

基于 Agent 的云端数据存储系统模型中 CS、EN 上都存在一个本地驻守、用于管理本地资源的 Agent(Resource Manager Agent，RMA)。RMA 负责代理节点设定及管理和监控本地节点存储信息、节点平均性能表现、节点实时状态，并基于 ACL 将节点的本地数据存储(Data Storage)模块的信息发送给 MS 节点上的资源管理 Agent(Global Resource Manager Agent，GRMA)，通过注册(Register)和元数据(MetaData)模块实现资源登记和管理，维护云端存储系统全局的资源信息。

3.5 工作流程

(1)数据存储

①初始化

用户首先在本地生成一密钥对，即公钥和私钥对，记作 Ku 和 Kr；然后出具与自身有关的私有信息，如姓名或账户、口令或其他可识别的个人信息(包括本地生成的公钥 Ku)，向 MS 进行注册。MS 确保该用户的真实性后向该用户发放数字证书(Digital Certificate)，一个标准 X.509 数字证书[14,15]包括的内容如表 1 所列。

表 1 由 MS 颁发的数字证书

项目	采用标准
证书序列号	RFC 3280
持有者的标识信息	RFC 3280
公开密钥 Ku	RFC 3280
有效期	GB/T 7408-1994
认证机构的名称	GB/T 2659
数字签名	RFC 3280

其中，证书序列号是数字证书的唯一标识；持有者的标识信息包括该用户出具的相关个人信息；公开密钥指的是持有者的公开密钥；有效期指数字证书的合法期限，包括起始时间和终止时间；认证机构在颁发数字证书之前用机构的私钥对该证书进行加密形成认证机构的数字签名。用户获得 MS 颁发的数字证书后，在本地创建两个 Agent，分别为本地驻守 Agent 和移动 Agent，记作 $Agent_p$ 和 $Agent_q$。用户端初始化参数如表 2 所列。

表 2 初始化参数

参数	生成者	作用
Ku	用户	公钥，发送给云端节点以便解密用户端上传的用户数据
Kr	用户	私钥，对本地待上传数据进行加密签名
数字证书	MS	进行认证用户的真实性
$Agent_p$	用户	驻留在本地，管理本地私钥，并向云端发送指令
$Agent_q$	用户	迁移至云端完成用户数据的相关操作(如存储、被访问、销毁等)

②加密

加密即是对本地待上传文件 File 进行如下预处理，如图 5 所示。

步骤 1 由本地驻守 Agent 将 File 切分为若干系统标准大小的数据块(Block)，且 $Block=\{Block_1, Block_2, \cdots, Block_i, \cdots, Block_n\}$；

步骤 2 用初始化生成的本地私钥 Kr 对这 n 个数据块进行加密，且 $E(Block)=\{E(Block_1), E(Block_2), \cdots, E(Block_i), \cdots, E(Block_n)\}$；

步骤 3 将步骤 2 产生的 n 个密文数据块组合成一个密文文件，并对其进行 Hash 运算形成消息摘要 $H(E(Block))$，然后用 Kr 对其进行加密签名；

步骤 4 用用户本地生成的私钥 Kr 对 MS 颁发的数字证书进行加密签名。

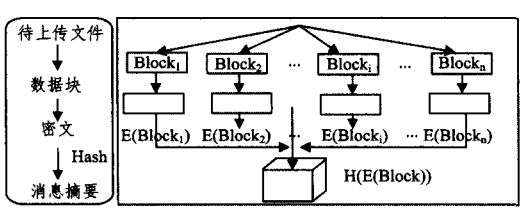

图 5 待上传文件的预处理

③上传

用户本地的资源请求 Agent(Resource Request Agent，RRA)与 MS 节点上的 GRMA 交互，目的是申请并获取符合自身需求且当前可存储 File 数据块的节点信息。GRMA 根据信息库中记录的云端节点的注册信息，按照负载均衡、保障服务质量以及充分利用 EN 节点资源等原则来选定存储节点。然后该节点上本地驻守 Agent 与用户进行交互，将节点上的公钥 Ku' 发送给用户。用户用该公钥对 File 的密文 $E(Block)$、消息摘要和经过本地私钥 Kr 签名过的数字证书进行加密。最后，将该密文和 $Agent_q$ 及本地公钥 Ku 上传至云端。云端节点用自己的私钥 Kr' 解密密文，在确保该用户的真实性和数据的完整性后进行存储。这时 GRMA 将该云端节点的基本信息发送给用户端，以便用户了解自己数据托管至云端的节点环境。

④存储

鉴于用户需求的不同、云端节点的动态性、用户数据安全等级等问题，用户数据块一般存储在多个云端节点上，即存在一个数据块的多个副本，它们可以存在同一个节点上，也可以分布在不同的节点上。产生副本的时候，来自用户端的 $Agent_q$ 会自动孵化一个 Agent，迁移到副本所在节点，起着与 $Agent_q$ 一样的作用。这些由 $Agent_q$ 孵化的 Agent 之间以及其与 $Agent_q$ 之间定期地进行通信，查看用户数据块及其副本的状态。

(2)数据处理

数据处理可以是数据提供者对云端数据的读取、修改等操作，也可以是客户对数据的访问、云端数据更新等过程。在本文提出的架构模型中，客户想要访问云端节点上存储的数据时，在本地创建一个 Agent 与用户迁移至目标节点的 Agent 进行安全的遥控、通信和协作，以完成数据的异地操作和返回结果。

①认证

不管是作为数据提供者的用户，还是作为云端访问者的客户，对云端存储的数据进行操作之前都必须进行身份认证。首先在 MS 的信息库中查询是否有该用户(或客户)的注册信息，并核实该注册信息的真实性。

②授权

对那些通过认证的授权客户进行访问需求的授权(如读、写、删除等)。以数据提供者提供的权限为前提，为客户请求授予相应的访问权限，如查看、下载、修改、删除等。访问云端数据的基本流程如图 6 所示。

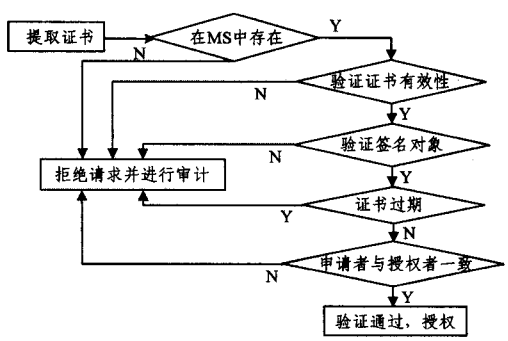

图 6 访问云端数据的基本流程

③更新

云端数据更新可以由数据提供者发出，也可以由授权客户执行。数据提供者可以根据自身需求对云端数据进行更新，即通过本地 $Agent_p$ 向云端发送更新指令；授权客户对云端数据的有权限修改也属于对云端数据更新的方式之一。另外，云端数据还可以自动地进行更新，即对于那些有规律可循的用户数据（如日期等），可以在上传数据之前对 $Agent_q$ 进行更新策略的设定，定期对云端存储的数据进行更新处理，但是，在这些更新操作之后必须更新 MS 相应的元数据信息。

4 云端存储数据复合销毁模型

4.1 销毁场景

传统的云存储系统本身存在数据删除不彻底的问题。例如，Google 云平台的 Google 文件系统（Google File System，GFS）[16,17]在删除文件后，并不立即回收磁盘空间，而是等到垃圾收集程序在文件和数据块的检查中收回。当一个文件被删除之后，主管理节点会记录下这些变化，但文件所占用的资源不会立即释放，而是给文件设定一个隐藏的名字，并注明删除的时间戳。主管理节点在定期检查名字空间时，删除超过一定时间的隐藏文件。在此之前，主管理节点以一个新的名字来读文件，还可以之前的名字恢复文件。只有当隐藏的文件在名字空间被删除之后，其元数据信息才会被删除。

云端存储系统在获得用户的删除命令时不一定会根据用户的要求彻底删除该数据及该数据的相关信息（如副本、属性记录等），终端节点随时可能会携带待删除数据暂时离开网络，从而造成用户数据销毁失败。简言之，云端存储系统中任何存储节点上的数据，都可能会因为技术问题不能被彻底销毁。

在云端存储系统中，以下几种场景需要执行数据销毁操作。

（1）数据过期销毁：云端存储系统中的过期数据主要包括到达预先设定生命周期的数据、访问频率在一定时间内低于一个预先设定值的冷门数据、更新失败的数据、冗余副本数据等。大量的过期数据浪费云端存储资源，降低了系统存储能力，增加了数据查询的代价。

（2）数据恶意攻击销毁：恶意攻击主要包括未授权访问、恶意篡改、服务提供商有意泄露、黑客攻击等。除了数据拥有者自身和授权用户之外的所有用户（包括服务提供商）均可能成为恶意攻击者。及时销毁被攻击的用户数据是维持数据的私密性的重要措施。

（3）数据残留销毁：节点数据删除不彻底、待删数据所在的存储节点暂时离线等都会造成云端存储系统的数据残留。残留数据中仍可能包含用户不希望他人获知的私密信息，同时可能影响存储空间的有效利用，用户和系统本身都有全面清除残留数据的需求。

4.2 数据销毁

云端存储的数据销毁指令可以由数据提供者、授权访问者或者云服务提供商发出。数据提供者若想要在该数据生命周期到达之前销毁云端数据，可以利用驻留在本地的 $Agent_p$ 向云端节点发送一个销毁指令或调用本节点的销毁策略将节点数据销毁；那些被授予销毁权限的客户，可以在访问该数据之后对节点数据进行销毁；当节点数据预设的时间戳到达时，可信云提供商可以对该节点数据进行销毁。但在这些销毁情况下，须对提出销毁指令的那一方进行严格的认证，以确保发出指令者是合法的；同时，必须确保删除的是待删除数据，而不能把他人的数据销毁。云端数据销毁的基本流程如图 7 所示。

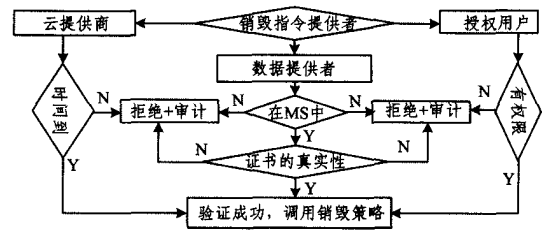

图 7 云端数据销毁的基本流程

①认证

对于数据提供者发出的销毁指令，必须验证其身份的真实性。首先在 MS 中查询该指令发出者的注册信息，核实该用户的真实性。

对于授权客户发出的销毁指令，其一定是通过了认证的客户发出的，还须判断是否对授权客户提供了销毁该节点数据的权限。若该节点数据提供该客户销毁该数据的权限，则将该销毁指令视为合法。

当那些定时模块中预设的该节点数据的时间戳到达时，调用策略模块对该节点数据进行有效销毁。

②删除

通过认证的销毁指令发出者调用策略模块中提供的销毁策略或直接调用本节点的销毁模块对该节点数据进行销毁。同时，可以根据数据保密等级的不同，选择不同的销毁方式对节点数据进行有效销毁。最后将该 MS 中的相关注册信息、云端节点和用户终端的 Agent、数据副本一并销毁。

4.3 销毁验证

虽然基于移动 Agent 技术的云端数据销毁机制不依赖于云提供商，整个过程都由移动 Agent 完成，但容易造成 Agent 的“瓶颈”。在数据的整个生命周期中，如果用户创建的移动 Agent 被恶意篡改，Agent 返回的“成功销毁”的消息将不再可靠。而且，对于那些 Agent 发出一次销毁时不在线的节点，Agent 此刻返回的“成功销毁”也是不完全的。所以，本文设计了相应的验证机制，如图 8 所示。定义上述本地创建的两个 Agent 为一对相互匹配的 Agent，$Agent_p$ 和 $Agent_q$ 存在“锁”和“钥匙”的关系，即将 $Agent_q$ 和经过预处理的用户数据以及相应的数字证书上传至云端存储时，$Agent_p$ 驻留在用户本地。一方面，$Agent_q$ 定期向本地驻留的 $Agent_p$ 发送消息，让用户了解托管数据的状态；另一方面，本地驻留的 $Agent_p$ 定期向云端 $Agent_q$ 发送消息，查看 $Agent_p$ 发送的状态并在存储节点上的数据和相应的 $Agent_q$ 都被销毁之后在预设的时间期限内定期向云端发送查询命令，看是否仍存在用户隐私数据及残留数据，完成销毁成功的验证功能。

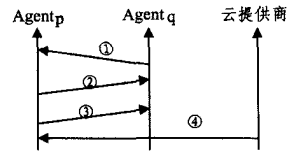

图 8 一次销毁验证全过程

①反馈

用户将在本地创建的 $Agent_q$、经过预处理的用户数据和相应数字证书上传至云端之后，将用户数据存储的节点信息通过安全信道反馈给驻留在用户本地的 $Agent_p$ 上，让用户了解托管数据的状态。

②事先轮询

用户本地驻留的 $Agent_p$ 在数据销毁之前主动定期向云端对应的 $Agent_q$ 发送查询指令，了解存储在云端的用户的存储状态(包括是原数据还是副本、存储节点有没有改变、有无恶意攻击等潜在危险等)。

③事后轮询

在 $Agent_q$ 反馈数据安全销毁之后，$Agent_p$ 在预设时间内定期向云端发送轮询指令，查看是否仍存在本该销毁的数据或残留数据，并在云提供商“强制反馈”之后向云端再次进行查询，务必确保云端已不存在本该销毁的用户数据。

④强制反馈

若事后轮询仍发现用户数据残留，则通过与云服务提供商交涉的办法，强制云服务提供商销毁该残留数据，并保证在之后的轮询中不会再发现该残留数据。

5 云端数据复合销毁机制

(1)数据节点的选择

对于第一类过期数据，可以随机地逐个销毁。对于第二类恶意攻击数据，则将先将被攻击的节点数据销毁，然后逐一删除该节点数据的原数据、副本节点数据以及与该数据相关的 Agent 信息。对于第三类残留数据，定期进行轮询，并将本地信息库中没有的以及没有相应完整 Agent 信息的节点数据销毁。

(2)数据销毁的条件

云端存储数据销毁的时间影响用户数据的安全以及云端空间资源的充分利用。对于那些有预设时间的节点数据，当预设时间到达时立即调用销毁策略进行销毁。一旦节点数据上的 Agent 发现存储环境异常或者是被未授权访问恶意攻击时，立即销毁该节点数据。此外，对于那些没有预设时间的、过期的、信息陈旧的、残留的数据，云端创建一个轮询 Agent 定期查询，一旦发现有类似现象的数据，立即销毁。

(3)数据销毁的策略

对于过期数据、多副本、残留数据等数据的销毁，应能实现主动销毁，这样就不需要额外的人力、技术去执行销毁操作；对于那些有预设值的节点数据的销毁，应能完成定时销毁；对于被恶意攻击、欺骗等的数据，其在被攻击、欺骗的“萌芽”阶段应能实现自销毁，这样就能够避免用户数据的泄露，从而保证用户数据的安全性。

云端数据自身、副本放置、安全等级等都是多样的，没有统一的标准；授权访问者需求更是多样的；恶意攻击者能力的不确定性，这些往往使得单一销毁方式不能完全完成任务。本文提到的复合销毁模式主要有主动销毁、定时销毁和防御型销毁 3 种。

5.1 主动销毁

定义 2(主动销毁) 即不被任何外力所干预，只根据其存储系统的内在设置对存储数据进行合理的自销毁。

例如，基于 DHT 网络[18]的动态性，超过一定时限的数据就会主动销毁，不能被任何方式获取；当一个用户数据的副本数大于该系统的上限时，主动销毁最先设置的副本或者销毁所有副本等。用户上传数据至云端之前先对该云服务器进行一定的了解，比如一般形成几份数据副本、销毁机制等内部设置。一般遇到以下情况时使用主动销毁模式。

(1)数据备份过多：即数据副本超出系统规定副本数，此时驻留在该节点上的 Agent 应调用销毁指令对原数据和所有副本文件进行整合，销毁那些超出系统规定的副本。

(2)数据存储时间过长：即对于那些长期存储在云端，并在某一长时间段内没有任何价值的用户数据，节点上的 Agent 应调用销毁指令对该节点原数据及其所有副本一并删除。

(3)存储数据节点不在线：并不是所有的云端服务器都是永久在线的，对于那些短暂不在线的服务器，存储在其上的用户数据很容易因为不在线而不受控制，以致用户数据泄露。此时，驻留在该节点上的 Agent 应能调用销毁指令对该不在线节点上的副本进行销毁。

5.2 定时销毁

定义 3(定时销毁) 即预先设定一个阈值，一旦到达这个阈值就销毁该阈值作用的节点数据。

云端用户隐私安全保护一直备受大家的关注，现在很多研究都基于 DHT 网络的动态性，利用其限定时间来保护云端用户数据不被泄露。但是，有时用户刚将数据上传至云端就希望取消数据托管，这时 DHT 网络的限定时间还没有到，该用户数据不能按用户的需求立即删除；有时，DHT 网络限定时间到达时，用户仍希望存储在云端的数据再保留一段时间，但此刻 DHT 网络中的数据都会被自动销毁。这时，就需要一个定时机制，让其根据用户的需求，可以动态调节时间，这也就是定时销毁机制所要实现的功能。

用户可以在上传数据之前设置好生命周期，然后利用 Agent 携带用户数据及定时器上传数据，这个定时器的设置完全由用户控制。用户想要改变这个定时器时长时，向存储该数据节点的 Agent 发送命令，Agent 调用定时模块更新这个时间。

5.3 防御型销毁

定义 4(防御型销毁) 指在用户数据面临潜在危险的情况下立即对节点上的数据进行销毁，具有超前性，如图 9 所示。

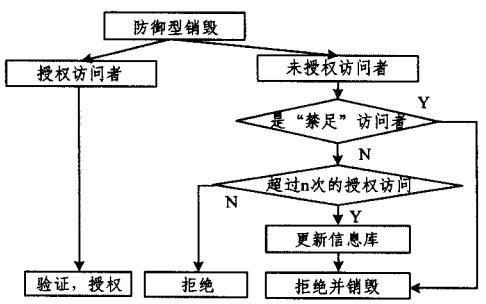

图 9 防御型销毁

主动销毁和定时销毁都是基于一个“限定时间”的销毁策略，具有滞后性。而防御型销毁是用户在上传数据之前规定了授权用户和“禁足”用户，在 Agent 携带访问者的数字证书进行匹配的过程中，若发现该访问者是一个“禁足”用户，则拒

绝访问并立即调用销毁指令对该节点数据进行删除；若发现该访问者既不是授权用户也不是“禁足”用户，则拒绝访问，但如果该访问者在拒绝访问之后仍多次通过类似伪造身份的方式强制访问该节点上的数据，此时将该访问者写入“禁足”用户模块，并立即销毁该节点上的数据。

5.4 典型应用

下面通过一个例子来说明这 3 种销毁模式的应用情况。

以 $User_1$ 试图销毁数据块 DB_3 为例，一次成功的数据销毁步骤为：

Step 1 $User_1$ 登录系统，在通过主管理节点身份认证后，主管理节点会为其颁发一张令牌；

Step 2 $User_1$ 查阅本地的 $File_x$ 元数据信息，查询 DB_3 所在的存储节点，向所有存储节点发出数据销毁请求；

Step 3 所有节点验证 $User_1$ 的令牌后，根据用户的销毁请求调用本地的数据覆写模块销毁本地的 DB_3 数据副本；

Step 4 节点在数据销毁操作成功后，将更新本地的元数据信息，并向 $User_1$、主管理节点发回确认信息；

Step 5 $User_1$ 和主管理节点收到确认信息，更新各自的元数据信息；若节点当前不在线，或是本次数据销毁操作并未成功完成，$User_1$ 将记录下在规定时间期限内未返回确认信息的节点，并在一段时间后再次向该节点发出数据销毁请求。

上述的数据销毁步骤采用的是主动销毁策略，但对于反复发出数据销毁请求却不得回复的节点来说，显然达不到理想的效果。因此本文在上述主动销毁策略的基础上，联合定时销毁策略，一旦节点上的数据到达失效期，就立即调用销毁指令对其进行删除。同时，为了避免主动销毁和定时销毁带来的滞后性，本文引进防御型销毁模式，即对那些被数据拥有者明确指出是未授权的恶意访问者，应该在初步验证其身份的过程中，为了避免该节点上用户数据泄露而立即销毁该节点上的用户数据；并根据上述防御型销毁模式对那些潜在的未授权用户进行排查规避，以最大限度保障用户数据不被恶意泄露。

5.5 数据折叠

针对云端存储数据删除不彻底的缺陷，文中设计了云端数据的多安全级数据销毁机制，以达到数据的安全销毁。不管是上述哪种销毁模式，最终都要调用销毁指令来删除数据，这里所说的销毁指令多数指的是数据覆写。根据用户数据安全等级的高低，采用的销毁方式、覆写的方式都不同。

目前的数据销毁方式[19]分为硬销毁和软销毁两种。数据硬销毁是指采用消磁、焚化和熔炼等物理破坏或化学腐蚀的方法把记录涉密数据的物理载体完全破坏掉，从而在根本上解决数据泄露问题的销毁方式。这对于需反复使用的云存储系统而言显然是不合适的。数据软销毁又称逻辑销毁，即通过删除文件、格式化硬盘和文件粉碎等软件方法销毁数据。

可采用数据覆写法来实现数据的远程软销毁。数据覆写法[20]是用预先定义的无意义、无规律的信息反复多次覆盖硬盘上原先存储的数据，从而达到销毁数据的目的。数据覆写分为逐位覆写、跳位覆写、随机覆写等模式。根据时间、密级要求的不同，可组合使用上述模式。美国国防部网络与计算机安全标准和北约的数据覆写标准规定了覆写数据的次数和覆写数据的格式。由数据覆写法处理后的硬盘可以循环使用，特别是需要对某一具体文件进行销毁而其他文件不能破坏时，这种方法更为可取。

目前，主流的覆写标准[21]有 DOD5220.22-M 简单覆写标准、DOD5220.22-M7 次檫除标准、全零覆写标准、RCMP TSSIT OPS 标准和 Gutmann 等标准。覆写算法如表 3 所列。

表 3 现有典型的覆写算法

算法	描述
全零	向文件中依次全部覆写一次 0
DOD5220.22-M	1)产生一个随机数，用该随机数覆写文件； 2)取该随机数的反码，用该反码覆写文件； 3)生成另一个随机数，用该随机数覆写文件。
DOD5220.22-M7	1)产生一个随机数，用该随机数覆写文件； 2)取该随机数的反码，用该反码覆写文件； 3)产生另一个随机数，用该随机数覆写文件； 4)产生另一个随机数，用该随机数覆写文件； 5)取该随机数的反码，用该反码覆写文件； 6)产生另一个随机数，用该随机数覆写文件； 7)产生另一个随机数，用该随机数覆写文件。
RCMP TSSIT OPS	该覆写算法一共覆写 8 次，奇数次产生随机数并用该随机数覆写文件，偶数次用上次随机数的反码覆写文件。
Gutmann	覆写 35 次，覆写速度慢，时间长。对大文件来说，覆写效率太低，会对系统的性能造成很大的影响。

不同的标准安全性不同，同时消耗的时间和资源也不同，对系统的性能产生的影响也不同。对那些政府、军事等保密性极高的数据(即保密性大于代价)，只能用硬销毁方式。对于大多数的云端存储数据，只要使用一定的覆写方式即能满足用户的删除要求。有时候，也可以一次使用多种覆写方式。现今效果最好的是 Gutmann，但是它是以 35 次之多的覆写次数来换取彻底销毁的结果，对大文件来说，覆写效率太低，会对系统的性能造成很大的影响。因此，寻找以有限的覆写次数达到最大覆写程度的覆写方式仍值得进一步研究。

本文提出了一种关于“数据折叠”的数据销毁方法。该方法充分利用数据本身序列进行覆写，不需要事先生成覆写序列，减少了覆写的时间和系统开销。所谓“数据覆写”就是将连续存储数据序列的“首”和“尾”的位置对折，这样引起整个序列两两对应，然后将这两两对应的数据分别进行“模 2 加”操作，充分利用自身数据序列完成数据覆写，从而最终实现数据的安全销毁，主要包括以下步骤。

Step 1 确定要覆写的数据序列 ListData(假设长度为 n)，将 ListData 设定为原数据(Raw Data，RD)，找到 RD 的第一位和最后一位，即其“首”(ListDatafirst)和“尾”(ListDatalast)。

Step 2 第一次折叠(折叠-展开-对称的过程)：将步骤 1 中找到的 ListDatafirst 和 ListDatalast 位置进行对折，即将后半段的序列从末尾开始写入了前半段序列(即折叠)，也就是说对前半段序列进行了一次覆写。但是由于是用原数据的后半段序列来覆写的，因此通过这样的一次折叠结束后，仍保留着后半段序列的信息(形式、位置改变了而已)，而且前半段的序列信息也可以通过现有的一些恢复软件恢复出来。

因此，本文引入“模 2 加”操作，即在数据序列两两对应之后，对相应位置上的数据两两分别进行“模 2 加”操作，这样一次折叠之后的结果就不再携带后半段序列的数据信息，同时给前半段序列的恢复带来了难度。

最后，将这前半段的折叠结果从原始数据的最后一位开

始重新将后半段序列写满数据，即将“模 2 加”的结果同时写入参与操作的两个位置上（即展开），这样就缩短了覆写的时间。到此，第一次折叠完成。

Step 3　第二次折叠：从第一次折叠之后的序列的 $n/2$ 位置向前查询，将第一个不是“0”的那个位置定义为此刻的最后一位 $m(m\leqslant n/2)$（记作 ListDatalast_m），序列的第一位位置不变；这样，对于不同的数据序列的折叠，从第二次开始，ListDatalast 的位置是不定的，给数据恢复带来了难度。然后，按照 Step 2 开始第二次折叠。

不同的是，将这段按照步骤 2 收尾折叠后的序列从该次折叠前确定的 ListDatalast_m 的后一位（$m+1$）开始连续写入未进行折叠的序列位置上，即对未进行第二次折叠的数据位进行覆写；到此，第二次折叠完成。

Step 4　第 N 次折叠：从上一次进行折叠的最后一位开始向前查询，找到第一个不是“0”的位置，并将其定义为该次折叠的最后一位（记作 ListDatalast_n），然后，按照 Step 3 进行折叠。到此，第 N 次折叠完成。

事实上，一般的用户数据通过 2～3 次的覆写就能满足用户的需求。但作为备选方案，按照上述步骤折叠到不能再折叠时，将该时刻的序列当作原序列进行下一轮折叠，以满足用户对数据销毁程度的各类需求。

6　实验

6.1　系统环境实现

本节将详细阐述基于移动 Agent 的云端数据销毁原型系统，具体包括客户端、服务器端和移动 Agent 3 个部分，分别包括图 3 所示的相关模块。原型系统采用 Java 语言以及 NetBeans 工具来开发，移动 Agent 基于 JADE 开发。JADE 符合 MASIF 和 FIPA 规范。

本地驻守 Agent 实现的方法是：由客户端创建一个 Agent，在本地利用 Java 中的 Keytool 工具实现仿真数字证书（携带用户相关信息），用这个 Agent 与服务器端进行交互，将该数字证书发送至服务器端进行注册，并获取一个数字凭证。

移动 Agent 的类的内容如表 4 所列。Agent 实现的方法是：由用户端创建 Agent，该 Agent 包含数据属性、定时模块、策略模块以及 Agent 自身的属性。然后，将该 Agent 迁移至目标节点实现云端数据存储与处理任务。本地 Agent 的主界面如图 10 所示。

表 4　移动 Agent 的类说明

变量	
File	待删除文件
AgentState	用于描述 Agent 的状态
Location	用于描述待删数据所在节点地址
AgentType	用于表示 Agent 的类型
Certification	用于描述数字证书的内容
Ontology	用于描述销毁策略
方法	
getFileName()	用于获取待删除文件名
getCertification()	用于获取数字证书
init()	用于初始化 Agent
live()	JADE 定义的 Agent 的入口函数
move()	用于使 Agent 迁移
runCertification()	当 Agent 迁移到节点时进行认证
runFileName()	当 Agent 迁移到节点时调用销毁策略销毁数据

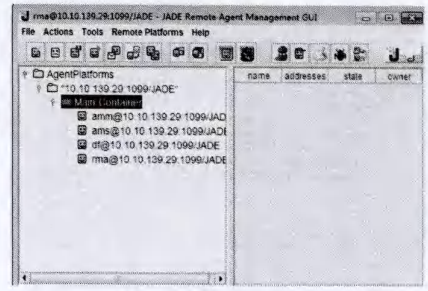

图 10　本地 Agent 的主界面

本地 Agent 主界面为 Agent 的迁移提供环境，负责创建、移动 Agent，并查看 Agent 的通信状态。用户 Gpp 在本地创建一个名为 Delete 的 Agent，并将其迁移到目标节点 Alice 后执行策略模块中设定的销毁指令（这里指定销毁目标节点上 E:\test. txt 文件），迁移过程如图 11、图 12 所示。

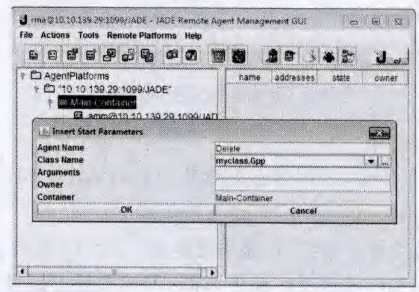

图 11　在本地创建 Agent

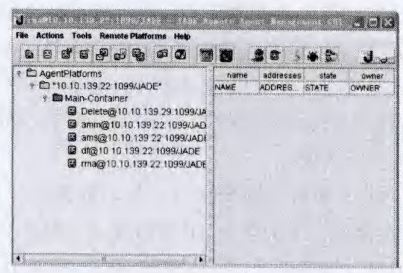

图 12　Agent 迁移至目标节点

在 Delete 完成任务之后，Alice 向 Gpp 发送一个名为 Reply 的 Agent 作为答复，并携带 Delete 任务执行的结果（若完成则 Success，失败有 Not Found 或 Failure 等），图 13、图 14 是一个任务执行成功的答复结果。

用户 Gpp 在接收到 Alice 的“Success”销毁成功的消息之后，定期向 Alice 端发送名为 Delete 的 Agent 以确保在销毁成功之后不会因为恢复软件、重新上传等原因再次出现本该销毁的用户数据。如图 15 所示，Alice 端不再存在用户 Gpp 指定销毁的数据。

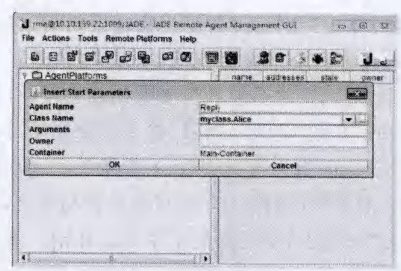

图 13　目标节点创建名为 Reply 的 Agent

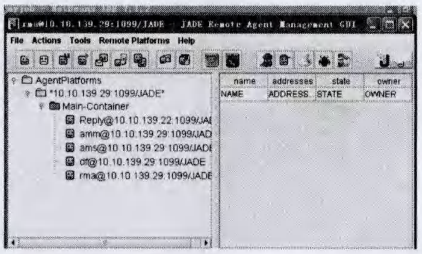

图 14　用户 Gpp 接收到任务执行成功的答复

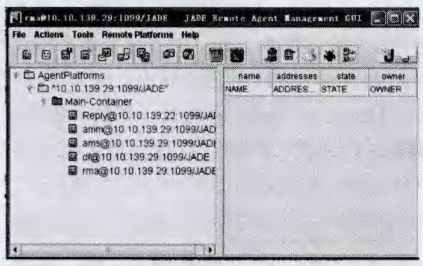

图 15　不存在指定销毁的数据

6.2　性能分析

下面从安全、开销等方面对基于移动 Agent 的云端数据远程销毁机制进行进一步分析。

(1)安全性。首先,在该系统模型中 Agent 起到了承上启下的作用,与传统的系统模型不同,不再依赖云提供商,并且数据托管至云端后对用户而言不再是不可控的;而且,用户在上传数据之前对待传数据进行了一系列预处理(如加密、签名等),保证了用户数据的完整性和用户的真实性。重要的是,该系统对云端数据的任何操作(如访问、更新等)都要经过严格的认证,用户身份的真实性以及用户权限的可靠性都需要经过 MS 的多级认证。

(2)有效性。传统的数据销毁只是对数据进行多次的覆写工作,而本文结合主动销毁、定时销毁、防御型销毁等策略更加全面地对云端数据进行有效销毁;而且,本文引入了验证模块,该模块的功能是在数据生命周期到达之前,与云端数据存储节点进行定时通信,并在数据被销毁后定期与云端进行交互,以确保不再存在本该已被销毁的数据。

(3)系统开销。大量的 Agent 模块功能都是在用户本地完成的,不会造成系统的开销。到达目的节点后的节点内部的 Agent 通信开销远比节点间 Agent 的通信开销小,而该系统着重就是 Agent 在节点间的通信,这就会造成一定的通信开销。而引入"数据折叠"的销毁方法,充分利用待删数据自身的序列进行有效覆写,不需要事先产生覆写序列,这将大大降低系统开销。

结束语　本文针对云端数据远程销毁机制的问题,结合移动 Agent,设计了基于移动 Agent 的云端数据复合销毁机制,结合主动销毁、定时销毁和防御型销毁实现对过期、废弃型数据的及时、有效、灵活的销毁,并在恶意主体对数据实施攻击时主动实施防御性数据销毁,有效增强了用户数据安全性;还提出一种新颖的"数据折叠"的数据覆写方法,其充分利用数据本身进行销毁,有效降低了系统的开销。本文的研究成果对于解决各类分布式存储系统中的数据销毁具有一定的借鉴意义。

参 考 文 献

[1] 陈康,郑纬民. 云计算:系统实例与研究现状[J]. 软件学报,2009,20(5):1337-1348
Chen Kang, Zheng Wei-ming. Cloud computing: system instances and current research[J]. Journal of Software, 2009, 20(5): 1337-1348

[2] 金海,吴松,廖小飞,等. 云计算的发展与挑战[M]// 2009 中国计算机科学技术发展报告. 北京:机械工业出版社,2010:21-51
Jin Hai, Wu Song, Liao Xiao-fei, et al. Cloud computing and challenges[M] // China Computer Federation 2009 Computer Science and Technology Development Report. Beijing: Machinery Industry Press, 2010: 21-51

[3] 徐小龙,程春玲,熊婧夷. 基于 Multi-Agent 的云端计算融合模型的研究[J]. 通信学报,2010,31(10):203-211
Xu Xiao-long, Cheng Chun-lin, Xiong Jing-yi. Conjunct model of cloud & client computing based on multi-agent[J]. Journal on Communications, 2010, 31(10): 203-211

[4] 邓谦. 基于 Hadoop 的云计算安全机制研究[D]. 南京:南京邮电大学,2013
Deng Qian. Secure mechanism research based on the Hadoop in Cloud[D]. Nanjing: Nanjing University of Posts and Telecommunications, 2013

[5] Perlman R. File system design with assure delete[C]// Procession of the 3rd IEEE International Security in Storage Workshop. IEEE, 2007: 83-88

[6] Tang Y, Lee P P C, Lui J C S, et al. FADE: Secure overlay cloud storage with file assured deletion[J]. IEEE Transactions on Dependable and Secure Computing, 2012, 9(12): 903-916

[7] Geambasu R, Kohno T, Levy A A, et al. Vanish: Increasing Data Privacy with Self-Destructing Data [C] // USENIX Security Symposium. 2009: 299-316

[8] Wolchok S, Hofmann O S, Heninger N, et al. Defeating Vanish with Low-Cost Sybil Attacks Against Large DHTs[C]// In Proceeding of NDSS. 2010

[9] 王铁军,刘恒,孙明,等. 资源定位服务的分布式生成树 模型及算法研究[J]. 电子学报,2011,39(1):364-369
Wang Tie-jun, Liu Heng, Sun Ming, et al. Research on the model and algorithms based on distributed spanning tree for resource location service[J]. Acta Electronica Sinca, 2011, 39(1): 364-369

[10] Zeng L F, Shi Zh, Xu Sh J, et al. SafeVanish: An Improved Data Self-Destruction for Protecting Data Privacy [C]// IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom). 2010: 531-528

[11] 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(2):266 -272
Wang Li-na, Ren Zheng-wei, Yu Rong-wei, et al. A certainty delete method suitable for cloud storage data[J]. Acta Electronica Sinca, 2012(2): 266-272

[12] 岳风顺. 云计算环境中数据自毁机制研究[D]. 长沙:中南大学,2011
Yue Feng-shun. Research on data self-destruct mechanism in Cloud[D]. Changsha: Central South University, 2011

[13] Agent Technology [EB/OL]. 2012-2-1 [2013-3-22]. http://baike.baidu.com/view/144699.htm

(下转第 179 页)

binational models[J]. Journal of Systems and Software, 2007, 80(4): 606-615

[7] 刘逻,郭立红,肖辉,等. 基于参数动态调整的动态模糊神经网络的软件可靠性增长模型[J]. 计算机科学,2013,40(2):186-190
Liu Luo, Guo Li-hong, Xiao Hui, et al. Software Reliability Growth Model Based on Dynamic Fuzzy Neural Network with Parameters Dynamic Adjustment [J]. Computer Science, 2013, 40(2): 179-182

[8] 张柯,张德平,汪帅. 软件可靠性混沌神经网络模型[J]. 计算机科学,2014,41(4):172-177
Zhang Ke, Zhang De-ping, Wang Shuai. Chaotic Neural Network Model for Software Reliability[J]. Computer Science, 2014, 41(4): 172-177

[9] Yang Bo, Li Xiang, Xie Min. A generic data-driwen software reliability model with mining technique[J]. Reliability Engineering and System Saft, 2010, 95: 671-678

[10] 崔正斌,汤光明,乐峰. 遗传优化支持向量机的软件可靠性预测模型[J]. 计算机工程与应用,2009,45(36):71-74
Cui Zheng-bin, Tang Guang-ming, Yue Feng. Software reliability prediction model based on support vector machine optimized by genetic algorithm[J]. Computer Engineering and Applications, 2009, 45(36): 71-74

[11] 侯雪梅,高飞,宋瑞丽,等. 基于量子粒子群的软件模糊可靠性分配模型[J]. 信息工程大学学报,2013,14(1):124-128
Hou Xue-mei, Gao Fei, Song Rui-li, et al. Software Fuzzy Reliability Allocation Based on Quantum Particle Swarm Algorithm [J]. Journal of Information Engineering University, 2013, 14(1): 124-128

[12] Cai Kai-yuan, Cai Lin, Wang Wei-dong, et al. On the neural network approach in software reliability modeling[J]. The Journal of Systems and Software, 2001, 58(1): 47-62

[13] Castillo E. Functional Networks[J]. Neural Processing Letters, 1998, 7(3): 151-159

[14] Castillo E, Cobo A, Gomez-Nesterkin R, et al. A general framework for functional networks[J]. Networks, 2000, 35(1): 70-82

[15] Castillo E, Gutiérrez J M. Nonlinear time series modeling and prediction using functional networks. Extracting Information M asked by Chaos[J]. Physics Letter Apply, 1998, 244(1): 71-84

[16] Iglesias A, Arcay B, Ctos J M, et al. A Comparison between Functional Networks and Artificial Neural Networks for the Prediction of Fishing Catches[J]. Neural computer&applied, 2004, 13: 24-31

[17] 崔强,武春友,匡海波. BP-DEMATEL 在空港竞争力影响因素识别中的应用[J]. 系统工程理论与实践,2013,33(6):1471-1478
Cui Qiang, Wu Chun-you, Kuang Hai-bo. Influencing factors research of airports competitiveness based BP-DEMATEL model [J]. Systems Engineering-Theory & Practice, 2013, 33(6): 1471-1478

[18] 崔春生. 基于泛函网络的组合推荐算法[J]. 系统工程理论与实践,2014,34(4):1034-1042
Cui Chun-sheng. Hybrid recommendation based on functional network[J]. Systems Engineering-Theory & Practice, 2014, 34(4): 1034-1042

[19] 徐仁佐,谢旻,郑人杰. 软件可靠性模型及应用[M]. 北京:清华大学出版社,1994:125-129
Xu Ren-zuo, Xie Min, Zheng Ren-jie. Software Reliability Model and Application [M]. Beijing: Tsinghua University press, 1994: 125-129

[20] Lyu M R, Nikora A. CASRE-A computer-aided software reliability estimation tool[C]// Computer-Aided Software Engineering Proceedings. 1992: 264-275

[21] Vladicescu F P. Performance evaluation of computers courses technical, University of Denmark Informatics and Mathematical Modeling[OL]. 2006[2014-3-11]. http://www.imm.dut.dk/popentiu/pec/pec.html

[22] Castillo E, Gutiérrez J M, Cobo A. A minimax method for learning functional networks[J]. Neur Proc Lett, 2000, 1(1): 39-49

[23] Malaiya Y K, Karunanithi N. Predictability measures for software reliability models[C]// Fourteenth Annual International Computer Software and Applications Coference, 1990(COMPSAC 90). 1990, 1: 7-12

[24] 王涛,杨娟,陈代国. 基于贝叶斯网络的构件软件体系建模及可靠性研究[J]. 四川兵工学报,2009,30(5):4-7,13

(上接第 146 页)

[14] Wang C, Wang Q, Ren K, et al. Privacy-preserving public auditing for data storage security in cloud computing[C]// Proceeding of IEEE INFOCOM. 2010: 1-9

[15] The format of the digital certificate[EB/OL]. 2009-5-21 [2010-9-29]. http://baike.baidu.com/view/356572.htm

[16] Woo J. Market Basket Analysis algorithms with MapReduce [J]. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2013, 3(6): 445-452

[17] Qin J, Zhang Y P, Zong P. Research on Data Destruction Mechanism with Security Level in HDFS[J]. Advanced Materials Research, 2014, 834: 1795-1798

[18] Shen H Y, Li Z, Li J. A DHT-Aided Chunk-Driven Overlay for Scalable and Efficient Peer-to-Peer Live Streaming[J]. IEEE Transactions on Parallel and Distributed Systems, 2013, 11(11): 2125-2137

[19] Zou G L, Ma J J. The Data Self-Destruction Technology Research Used in Marine Environmental Monitoring Buoy System [J]. Applied Mechanics and Materials, 2013, 336: 2195-2198

[20] Croft J, Signorile R. Secure distribution of confidential information via self-destructing data[C]// Proceedings of the 8th World Scientific and Engineering Academy and Society (WSEAS) International Conference on on Data Networks, Communications, Computers Baltimore. USA, 2009

[21] 郑光,苏锦海,孙万忠. 闪存数据应急销毁算法的研究与设计[J]. 计算机应用与软件,2013,30(9):305-308
Zheng Guang, Su Jin-hai, Sun Wan-zhong. Research and design of flash memory data emergency disposal algorithm[J]. Computing Applications and Software, 2013, 30(9): 305-308