(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102307185 A
(43)申请公布日 2012.01.04

(21)申请号 201110175541.4

(22)申请日 2011.06.27

(71)申请人 北京大学
地址 100871 北京市海淀区颐和园路 5 号

(72)发明人 沈晴霓 杨雅辉 禹熹 张力哲
吴尉泷 王丹丹 龙敏

(74)专利代理机构 北京君尚知识产权代理事务所(普通合伙) 11200
代理人 余长江

(51)Int.Cl.
H04L 29/06(2006.01)
H04L 29/08(2006.01)
G06F 21/00(2006.01)

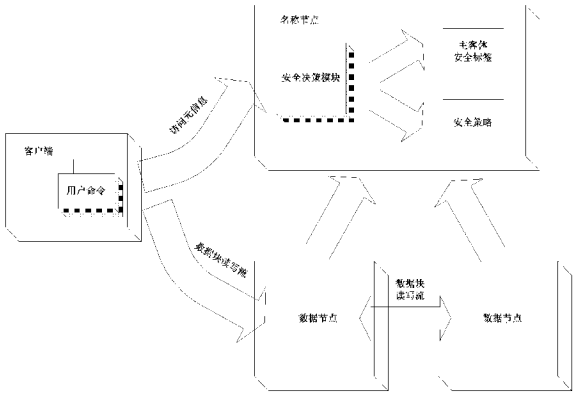
权利要求书 2 页 说明书 14 页 附图 5 页

(54)发明名称

适用于存储云内的数据隔离方法

(57)摘要

本发明涉及一种适用于存储云内的数据隔离方法,属于分布式存储领域。本方法为:1)根据组织的安全需求,在存储云系统主节点中建立该租户安全策略配置;2)创建属于该组织的主体用户,并为主体打上该组织角色模型中的角色标签;3)组织管理员为某组织用户创建的客体资源客体打上该组织 Tag 模型中的 Tag 标签;4)根据访问资源的角色权限,设置角色主体对访问资源客体的安全访问策略并将其存储到存储云系统的主节点中;5)主节点根据所述安全访问策略,判断角色主体对访问资源客体的访问是否通过,如果通过则进行访问,否则拒绝访问。本方法保证云存储企业内部数据的适度隔离,策略遵循通用性原则,适用于公有云,私有云以及混合云。



CN 102307185 A

1. 一种适用于存储云内的数据隔离方法，其步骤为：

1) 根据租户的安全需求，在存储云系统主节点中建立该租户安全策略配置，其包括主体层次化的角色模型和客体层次化的 Tag 模型，角色模型中的角色权限包括：读权限、写权限、执行权限；

2) 在存储云系统主节点中创建属于该租户的主体用户，并为主体用户打上该租户角色模型中的角色标签；

3) 在存储云系统主节点中将某租户的主体用户创建的客体资源上，打上该租户 Tag 模型中的 Tag 标签；

4) 根据访问客体资源的角色权限，设置角色主体对访问资源客体的安全访问策略并将其存储到存储云系统的主节点中；所述安全访问策略为：

a) 对于主体对客体的读访问，要求主客体所属租户一致，并且存在赋予主体的角色 k，使得角色 k 的读权限表达式在客体标签为真的情况下计算结果也为真；

b) 对于主体对客体的写访问，要求主客体所属租户一致，并且存在赋予主体的角色 k，使得角色 k 的写权限表达式在客体标签为真的情况下计算结果也为真；

c) 对于主体对客体的执行访问，要求主客体所属租户一致，并且存在赋予主体的角色 k，使得角色 k 的执行权限表达式在客体标签为真的情况下计算结果也为真；

5) 主节点根据所述安全访问策略，判断角色主体对访问资源客体的访问是否通过，如果通过则允许访问，否则拒绝访问。

2. 如权利要求 1 所述的方法，其特征在于所述主体角色模型为一具有角色继承关系的层次化标签森林，每一标签具有一角色权限。

3. 如权利要求 2 所述的方法，其特征在于所述客体层次化的 Tag 模型为一具有继承关系的层次化标签森林。

4. 如权利要求 3 所述的方法，其特征在于所述主节点采用 XML 文件存储所述安全策略配置；采用字符串形式表达所述角色权限表达式。

5. 如权利要求 4 所述的方法，其特征在于所述主节点中设置一哈希表结构的安全配置池，其中，键为租户名称，值包括哈希表结构的角色配置池、标签配置池；所述角色配置池的键为角色名称，值为角色对象；所述标签配置池的键为标签名称，值为标签对象；角色对象通过父指针指向其在所述角色模型中的父节点；标签对象通过父指针指向其在所述 Tag 模型中的父节点；主节点根据输入的租户信息、以及主体角色名或客体标签名，利用所述安全配置池查找角色对象或标签对象。

6. 如权利要求 4 或 5 所述的方法，其特征在于所述主节点判断角色主体对访问资源客体的访问是否通过的方法为：主节点将字符串形式表示的角色权限表达式转化成后序表达式树，然后根据给定表达式变量值，利用权限表达式树计算出决策结果。

7. 如权利要求 6 所述的方法，其特征在于所述主节点采用镜像文件和操作访问日志持久化所述安全策略配置。

8. 如权利要求 1 所述的方法，其特征在于角色主体对访问资源客体进行访问的方法为：

1) 角色主体向主节点发送要访问的文件名，访问的文件偏移量以及要访问客体资源的数据长度；

2) 主节点为该角色主体生成访问票据并用集群内共享的密钥对票据进行加密，然后将该角色主体要访问的数据块的标识及其所在数据节点的位置，以及加密后的访问票据发送给该角色主体；

3) 该角色主体对于每一个数据块，选择具有该数据块且距离该角色主体最近的数据节点，发送该数据块标识和访问票据，请求访问该数据块；

4) 数据节点使用集群内共享的密钥解密访问票据，判定该角色主体是否有权限访问相应的数据块，如果允许访问，则向该角色主体回送数据块的数据，否则拒绝访问。

9. 如权利要求 8 所述的方法，其特征在于所述访问票据包括：角色主体的标识符、角色主体要访问数据块的数据块标识以及访问的具体操作。

10. 如权利要求 9 所述的方法，其特征在于对于写访问操作：首先，角色主体远程调用主节点的 create 方法，开始新文件的创建，主节点采用租约方法为新创建的文件增加排它锁；然后角色主体远程调用主节点的 addBlock 方法，主节点在接收到 addBlock 请求后会返回给角色主体一个 LocatedBlock 对象，其包括新建数据块的标识以及能够写入数据块的数据节点；最后，角色主体远程调用主节点的 complete 方法，完成新文件的创建。

适用于存储云内的数据隔离方法

技术领域

[0001] 本发明提出了一种为存储云上的数据提供隔离服务的方法，并在以 HDFS(Hadoop Distributed File System) 为基础架构的云存储环境下实现了数据隔离的安全机制。本发明的技术领域涉及强制访问控制，分布式存储。

背景技术

[0002] 云计算作为一种新兴的技术和商业应用模式，近年来无论在业界还是学术界都获得了广泛的关注和巨大的推动。云计算为企业用户展现了美好的愿景。一方面，企业可以将其 IT 基础架构的管理和维护交由专业的云服务提供商负责，从而更加专注于公司本身的业务；另一方面，也是更为重要一点在于云计算提供的服务是自适应性的，具备灵活的可伸缩性。企业可以根据自身业务需求量大小租赁相应负载的云服务，不会因为公司业务需求量不足造成现有 IT 资源的浪费，也不会因为公司规模扩大导致旧有 IT 架构无法负载更大的业务量需求而需要花费昂贵的代价重新架构其 IT 基础设施。

[0003] 云计算的模式获得了巨大的成功，但研究人员对于其的关注往往多集中于如何提高云平台的可用性、可扩展性、效率性能以及容错性等，却忽视了云平台的安全性。事实上，实现安全可靠的云平台是保证企业用户从传统模式过渡到“云模式”一个关键要素，这是由云本身的特点决定的。在“云时代”，公司的应用运行在远端的云平台上，数据也存储在云上，云底层的 IT 基础设施由云服务提供商统一管理。对于应用了云计算的企业用户而言，其安全界限远远超出了公司的防火墙以外，企业本身能够控制的范围是非常有限的。公司的应用及数据均涉及到公司的商业机密，一旦泄露损失难以估量，甚至可能对整个公司造成毁灭性的打击。保证云平台的安全性对于云计算的进一步推广至关重要。如果无法为云平台提供可靠的安全保障，企业用户将很难放心的享用云带来的巨大效益。

[0004] 云存储在云计算体系中位于 IAAS(Infrastructure as a Service) 一层，主要用作云平台的存储基础设施。云存储的数据安全是云安全重要的一环。云存储出现以前，企业的数据都存储在公司内部的数据中心。“云时代”的到来使得企业将其私有数据由内部数据中心迁移到云服务提供商统一管理的公共平台以降低公司的运作成本。然而对于企业用户而言，云存储却是一把双刃剑。云存储给企业带来了收益同时也带来了安全隐患。云存储作为一种多租户的数据存储平台，存放着来自许多企业的内部私有数据。企业用户通过访问这个共享的公共平台获取属于自己公司的数据。由于云存储平台共享的特点，企业绝不希望自己的数据信息被同一平台上的其它租户窃取或者篡改。因此企业间的数据隔离是云存储亟待解决的一个重要问题。企业用户依据内部的安全需求需要对来自公司不同部门或者地域的数据进行隔离。云存储作为公共的存储平台，应该提供一种灵活多变的安全策略，它能够方便被使用云存储的企业用户定制以满足各个企业不同的安全需求。这也对云存储企业内的数据隔离提出了要求。

[0005] 以下是目前可查到的与云存储或分布式存储数据隔离或者访问控制相关的专利情况。

[0006] 公开号为 102014133A，发明名称“在云存储环境下一种安全存储系统的实现方法”的专利提供了在云存储环境下一种安全存储系统的实现方法，其特征在于：在服务器中根据用户需求建立信任域，在其中利用公钥基础设施 PKI 进行身份认证，利用用户空间的文件系统 FUSE，实现了存储系统与底层系统无关，利用哈希算法 SHA1 算法以块为单位对文件计算哈希值，再利用密钥和对称加密算法 AES 算法对以块为单位对文件块加密，再将文件密文上传到云存储区中的文件服务器，保证了文件的机密性和完整性，文件所有者通过在访问控制列表中指定具有对该文件进行访问的用户及其权限，在发生权限撤销操作时，将对文件进行重新加密的操作推迟，只有当用户对文件内容进行修改时，由该用户重新加密修改内容所在的文件块，系统实行了文件块密钥、安全元数据文件密钥和信任域服务器密钥三层密钥管理，既保证了权限撤销时文件的安全性，又不增加系统的管理负担。

[0007] 此专利的侧重点在于保护云存储上数据的机密性和完整性，实现机制是主要是通过加密算法保护数据机密性，通过哈希值来保证数据完整性，其中也粗略的提到了简单的数据访问控制机制，数据保护的粒度较粗。不同的是，本发明的访问控制安全策略是有针对性的，根据云存储这种多租户的特殊环境制定了一系列相关的安全策略，通过强制访问控制机制，极大的保证了企业间数据的强隔离性，企业内部数据的适度隔离，并进而考虑企业数据共享的情况。可以说两者得安全目标是截然不同的。

[0008] 公开号为 102006300A，发明名称“一种云存储数据加密方法、装置及系统”的专利涉及一种云存储数据加密方法、装置及系统，所述方法包括：根据预置时间内预期存储数据量 X，本地存储空间占用比例 R 及数据安全级别 Z 计算出应生成随机种子的大小 H；根据每次明文数据量 Y 计算出对随机种子采集的次数 u；按次数 u 对生成的 H 大小随机种子多次采集数据生成明文加密位标识数据串；利用该数据串选择二分之一以上明文数据加密形成密文。此发明还提供了一种云存储数据加密装置及系统。此发明在不牺牲数据安全性前提下，减少了待存储数据的加密数量，大大提升了云存储数据的存储速度。

[0009] 此专利属于云存储安全领域，但其关注点仍在云端数据的私密性。通过提出云存储的数据加密方法和装置，在保证系统性能的大前提下提升数据的安全性。它同本发明的保障云存储多租户环境下数据隔离的安全目标是截然不同的。

[0010] 公开号为 101316273，发明名称”一种分布式安全存储系统”的专利目的在于克服现有基于证书的安全存储系统中安全管理器负载重，用户权限管理复杂的问题。此发明包括连入网络的应用客户端、存储设备、安全与策略管理器和元数据服务器；安全与策略管理器存储并管理全系统的访问控制项、访问控制策略和规则，依据全系统访问控制项、访问控制策略和规则对存储设备进行访问策略控制和权限控制，包括改变访问控制项的优先级和继承规则、添加和删除访问控制项。

[0011] 此专利提出了一种主 / 从式的安全存储架构，通过成员或者角色的访问控制项来控制主体对数据资源的访问。同本发明类似，其实现针对的对象亦是主 / 从式的存储架构。不同的是此专利侧重于安全存储架构的设计，仅通过较为简单的自主访问控制列表来保障分布式存储的数据安全。本发明在此基础上更进了一步，主要面向云存储多租户的复杂环境，通过制定了一套灵活通用的强制访问控制安全策略来实现预定的数据隔离安全目标，具备环境直接针对性和更多的安全保障性。

[0012] 公开号为 101605137，发明名称“安全分布式文件系统”的专利提出了一种安全分

布式文件系统设计，包括认证服务器、元数据服务器、对象存储设备和客户端，认证服务器用于验证用户身份，并向通过验证的对应用户提供用以请求访问文件的目录票据和第一临时会话密钥；元数据服务器根据接收的目录票据和利用所述第一临时会话密钥加密的访问请求，向对应用户提供用以请求访问文件的文件票据、头文件和第二临时会话密钥；对象存储设备根据接收的文件票据和利用第二临时会话密钥加密的访问请求，向对应用户发送密文文件，客户端利用第一和第二临时会话密钥加密对应的访问请求以及通过头文件来解密密文文件供对应用户读写。本发明可以对用户的文件访问提供安全有效的保障。

[0013] 此专利侧重整套安全存储架构的设计，其核心思想使用的是票据，通过认证服务器认证颁发认证票据，通过元数据服务器决策颁发文件访问票据，最后使用文件访问票据访问存储设备上的数据。然而此专利完全没有关注与访问控制相关决策的策略，即元数据服务器如何做出访问控制决策。

发明内容

[0014] 本发明针对云存储这样一个多租户的存储环境，就其数据隔离问题，提出了相应的安全解决方案。云存储数据隔离的安全目标简单来说是为了保证存储在云间的数据只能被授权的企业用户访问，而无法被恶意的用户访问或者篡改。

[0015] 本发明提出了一套云存储系统安全访问控制服务，通过基于安全标签的强制访问控制策略为企业用户提供了数据隔离服务。本文针对云存储的特殊环境，以 RBAC(Role based Access Control) 为基础，结合组织标签和多种安全属性的逻辑组合，提出了一种灵活的访问控制策略，一方面能够保证云端不同企业间数据的强隔离性，使得企业用户无法越权去访问其它企业用户的数据；另一方面该策略能够保证云存储企业内部数据的适度隔离，企业用户可以根据公司自身的安全需求灵活定制企业内的访问控制策略，隔离来自不同部门和地域的数据；最后该策略通过虚拟组织的概念在企业间数据强隔离的情况下实现了可能的数据共享，并通过中国墙策略保障同一冲突集中的企业是不能共享数据的。本文提出的策略遵循通用性原则，适用于公有云，私有云以及混合云。

[0016] 本发明的技术方案为：

[0017] 一种适用于存储云内的数据隔离方法，其步骤为：

[0018] 1) 根据租户的安全需求，在存储云系统主节点中建立该租户安全策略配置，其包括主体层次化的角色模型和客体层次化的 Tag 模型，角色模型中的角色权限包括：读权限、写权限、执行权限；

[0019] 2) 在存储云系统主节点中创建属于该租户的主体用户，并为主体用户打上该租户角色模型中的角色标签；

[0020] 3) 在存储云系统主节点中将某租户的主体用户创建的客体资源上，打上该租户 Tag 模型中的 Tag 标签；

[0021] 4) 根据访问客体资源的角色权限，设置角色主体对访问资源客体的安全访问策略并将其存储到存储云系统的主节点中；所述安全访问策略为：

[0022] a) 对于主体对客体的读访问，要求主客体所属租户一致，并且存在赋予主体的角色 k，使得角色 k 的读权限表达式在客体标签为真的情况下计算结果也为真；

[0023] b) 对于主体对客体的写访问，要求主客体所属租户一致，并且存在赋予主体的角

色 k，使得角色 k 的写权限表达式在客体标签为真的情况下计算结果也为真；

[0024] c) 对于主体对客体的执行访问，要求主客体所属租户一致，并且存在赋予主体的角色 k，使得角色 k 的执行权限表达式在客体标签为真的情况下计算结果也为真；

[0025] 5) 主节点根据所述安全访问策略，判断角色主体对访问资源客体的访问是否通过，如果通过则允许访问，否则拒绝访问。

[0026] 进一步的，所述主体角色模型为一具有角色继承关系的层次化标签森林，每一标签具有一角色权限。

[0027] 进一步的，所述客体层次化的 Tag 模型为一具有继承关系的层次化标签森林。

[0028] 进一步的，所述主节点采用 XML 文件存储所述安全策略配置；采用字符串形式表达所述角色权限表达式。

[0029] 进一步的，所述主节点中设置一哈希表结构的安全配置池，其中，键为租户名称，值包括哈希表结构的角色配置池、标签配置池；所述角色配置池的键为角色名称，值为角色对象；所述标签配置池的键为标签名称，值为标签对象；角色对象通过父指针指向其在所述角色模型中的父节点；标签对象通过父指针指向其在所述 Tag 模型中的父节点；主节点根据输入的租户信息、以及主体角色名或客体标签名，利用所述安全配置池查找角色对象或标签对象。

[0030] 进一步的，所述主节点判断角色主体对访问资源客体的访问是否通过的方法为：主节点将字符串形式表示的角色权限表达式转化成后序表达式树，然后根据给定表达式变量值，利用权限表达式树计算出决策结果。

[0031] 进一步的，所述主节点采用镜像文件和操作访问日志持久化所述安全策略配置。

[0032] 进一步的，角色主体对访问资源客体进行访问的方法为：

[0033] 1) 角色主体向主节点发送要访问的文件名，访问的文件偏移量以及要访问客体资源的数据长度；

[0034] 2) 主节点为该角色主体生成访问票据并用集群内共享的密钥对票据进行加密，然后将该角色主体要访问的数据块的标识及其所在数据节点的位置，以及加密后的访问票据发送给该角色主体；

[0035] 3) 该角色主体对于每一个数据块，选择具有该数据块且距离该角色主体最近的数据节点，发送该数据块标识和访问票据，请求访问该数据块；

[0036] 4) 数据节点使用集群内共享的密钥解密访问票据，判定该角色主体是否有权限访问相应的数据块，如果允许访问，则向该角色主体回送数据块的数据，否则拒绝访问。

[0037] 进一步的，所述访问票据包括：角色主体的标识符、角色主体要访问数据块的数据块标识以及访问的具体操作。

[0038] 进一步的，对于写访问操作：首先，角色主体远程调用主节点的 create 方法，开始新文件的创建，主节点采用租约方法为新创建的文件增加排它锁；然后角色主体远程调用主节点的 addBlock 方法，主节点在接收到 addBlock 请求后会返回给角色主体一个 LocatedBlock 对象，其包括新建数据块的标识以及能够写入数据块的数据节点；最后，角色主体远程调用主节点的 complete 方法，完成新文件的创建。

[0039] 云存储作为一个多租户共享的存储基础架构，存在着自身安全的复杂性和特殊性。由云服务提供商所管理的公有云存储上存放着来自不同企业组织的私有数据，而由企

业自身负责管理维护的私有云存储上也存放着来自企业不同部门的私有数据，这些不同的企业组织之间，以及企业内部的不同部门之间就构成了云存储的多租户。无论是来自不同企业之间的数据，还是来自公司内部不同部门的数据在公共的存储环境上都需要被安全的隔离，以保证来自不同租户数据的私密性。因此，不论是公有云或者是私有云，亦或者是两者构成的混合云，都需要有一定的安全的和排他的虚拟存储环境。这样才能够保证存放在云上的数据的安全性，才能够促进云的进一步推广和应用，本策略正是基于此而提出的。

[0040] 安全策略设计的目标如下：

[0041] 1. 企业之间数据的强隔离性。对于公有云存储而言，云服务提供商必须要严格保障来自不同公司的数据被安全的隔离。公有云上存储着来自许多不同企业的数据，同时使用同一云服务提供商服务的某些企业之间甚至是同一块市场的激烈竞争者。一个企业内部的私密数据对其而言是重要的生命线，一旦泄露后果不堪设想，因此必须严格的保障企业和企业间数据的强隔离性。

[0042] 2. 灵活的企业内部数据隔离。无论对于存储在公有云上来自同一企业不同部门的数据，还是存储在企业内部私有云上的不同部门的数据，都需要进行一定程度的安全隔离。企业的每一个部门有自身的职责划分，它们分管企业某一块任务拼图的运作，比如生产部负责产品的制造，人事部负责人才的招聘和管理，以及财务部负责公司的收入和支出管理。根据系统安全的最基本原则”最小特权原则”，安全的系统一方面应该给予主体必不可少的权限，保证所有的主体都能够在赋予的权限之下完成所需要的任务和操作，另一方面也只给予主体必不可少的权限，这样就能够限制每个主体所能进行的操作，确保蓄意或者不小心越权造成的错误和事故导致的损失最小。因此不同部门间的数据也是需要进行一定程度的安全隔离以保证最小特权原则。因为公司内部部门存在着一定的协作关系，部分数据的共享在所难免，它们之间数据隔离应当具备适当的灵活性，不应当使用企业之间数据的强隔离特性。

[0043] 3. 企业之间数据共享的情况。当企业之间有协作关系时，也会存在着一定数据共享的情况。如果双方企业都使用云作为存储平台，那么公有云存储可以成为公司间数据共享的很好媒介平台。企业的应用可以在不用改变数据访问的接口的情况下直接去访问存储在云上的另一个企业的数据。然而企业间数据的共享需求和企业间数据的隔离需求存在着很大程度的矛盾和冲突。需要有一种机制能够在保障企业间数据强隔离的情况下，完成企业数据的共享。另外一些企业是同一块市场的激烈竞争者，如中国移动和中国联通，中国石油和中国石化。这些公司由于双方利益存在很大的冲突，它们可能不希望与竞争对手共享数据，应当有相关的机制防止误操作或恶意操作导致它们之间数据的共享。

[0044] 策略设计原则

[0045] 本发明提出的安全策略基于如下的设计原则：

[0046] 1. 通用性。无论是多个企业共享的公有云，还是企业内部使用的私有云，或者是公有云和私有云构成的混合云，它们都存在着多租户数据隔离的安全需求，因此安全策略应该是通用的，能够适用于各种不同的云环境。

[0047] 2. 灵活性。各个企业存在着自身不同的安全需求，因此存在多租户的云存储上的安全策略应当是灵活的，能够适用于各个公司不同的情况。企业用户可以根据自身的安全需求灵活的定制安全策略，实现其安全目标。

[0048] 3. 层次性。此安全策略是主要针对企业用户设计的。大部分企业的划分具有一定的层次性，安全策略中主体和资源客体的属性应该能够体现出企业的这种层次的特性。

[0049] 具体策略描述

[0050] 本发明所设计的安全策略是基于 RBAC 模型的，它实现了文中描述的安全目标，提供了云存储的数据隔离服务。策略的核心是基于主客体安全标签的强制访问控制。本节分为三小节，分别描述主体的安全标签设计，客体的安全标签设计以及基于主客体安全标签的访问控制规则。文中所指的组织，企业，公司，租户均可认为是同一概念，只是在不同的应用场景中使用。

[0051] 1> 主体安全标签

[0052] 层次化的角色模型

[0053] RBAC 模型预先定义了一组具备相应权限的角色 (role)，在遵循最小特权原则的基础上为相关用户赋予相应的角色。给用户赋予角色的过程实际上也是为用户赋予一定权限的过程。扩展的 RBAC 模型更具备灵活性，角色和角色之间可以存在继承的关系。角色继承的意义主要在于不用重复定义一些已经定义过的角色的权限，而是可以通过继承的方式自动承接所有祖先节点的角色权限，在此基础上进行角色权限的特定扩展。图 1 显示了三组定义好的角色继承的层次树结构。可以称作角色的森林。

[0054] 主体安全标签定义

[0055] 由于每个公司有不同的安全需求，因此其也会有不同的角色定义的需求。因此这里允许每个公司拥有自己不同的角色定义的森林。为了区分不同公司的角色定义，需要在资源访问的主体安全标签内加上特定公司的标签。一方面它可以区分不同公司的角色森林，另一方面它也可以实现公司间数据的隔离，关于这一点将在后面说明。

[0056] 云存储中资源访问的主体标签如下：

[0057] $\langle org_1, role_1, role_2, role_3, \ldots, role_n \rangle$ 或 <org，role list>

[0058] 图 2 的示例进一步说明了主体标签的角色定义。

[0059] 2> 客体安全标签

[0060] 层次化的标签模型

[0061] 使用标签 (Tags) 来对资源访问的客体进行标识。如果采用 Web 上常用的平面标签模型，确实它非常的灵活，但却存在两方面的问题。首先由于它的平面结构，它无法表达公司或者组织内部的层次化机构，这点对于公司而言是很不合理的；其次由于在标签定义上没有控制，很容易造成标签定义爆炸，会定义大量语义相同然而名字不同的标签，给整个系统的维护和运作带来困难。因此我们决定在此基础上对其进行改进，将平面的标签模型进行扩展成为层次化的对象标签模型。

[0062] 层次化的对象标签模型应用场景如下：首先由公司内部的高层管理人员根据公司需求，按照不同的准则（如地域，部门等）对公司进行具体划分，这样就能构成由很多树构成的森林，树上的每一个节点即代表了这个公司内部的一个可控的标签。通过这种方式，大大限制了系统中存在的标签个数，并且很好的表达出了公司内部的层次化的结构。在层次化的标签模型中，子节点相当于自动继承了其祖先节点的标签，因此也就拥有多个标签。

[0063] 图 3 说明了这种层次化的标签模型，可以称作标签的森林。

[0064] 客体安全标签定义

[0065] 同主体的安全标签类似，每个公司也会针对于自身的安全需求来创建自己专属的不同的标签森林。在客体的标签上需要对不同公司的标签进行区分，当然这也是保证公司之间数据隔离的需要。

[0066] 云存储中资源访问的客体标签如下：

[0067] $<org_1, tag_1, tag_2, tag_3, \ldots, tag_n>$ 或 <org，tag list>

[0068] 图 4 进一步说明了客体标签的定义。

[0069] 3> 访问控制规则

[0070] 角色的权限定义

[0071] 由于系统的强制访问控制是基于 RBAC 的，RBAC 中的角色脱离了权限的定义是无意义的，本小节主要对角色权限进行了定义。根据客户在云存储的访问需求，这里为每个 role 分别制定相应的读 (R)，写 (W) 以及执行 (X) 权限，即为 role：$(perm_r, perm_w, perm_x)$。

[0072] 权限定义针对云存储上的文件和目录的概念是不一样的，如下所示：

[0073] 1> 针对普通文件的权限定义：

[0074] • 读权限 (R)：对此文件拥有读权限

[0075] • 写权限 (W)：对此文件拥有写权限

[0076] • 执行权限 (X)：针对于云存储的特性，执行权限不适用于普通文件。

[0077] 2> 针对目录 / 文件夹的权限定义：

[0078] • 读权限 (R)：可以查看此文件夹下的文件和子文件夹列表（类似于允许执行 ls 操作）

[0079] • 写权限 (W)：可以对此文件夹下创建文件，删除文件，修改文件名

[0080] • 执行权限 (X)：可以进入此文件夹（类似于允许执行 cd 操作）

[0081] 对于读，写，执行权限的具体定义是一致的，都是使用客体的 Tag 标签加上于 (&&)，或 (||)，非（！）三种逻辑运算符组合构成的权限表达式 (expr)。具体定义可以用下面的表达式描述所示．逻辑运算符的优先级为非（！）＞与 (&&) ＝或 (||)，从左到右计算，可以使用括号来改变权限表达式的运算次序。

[0082] 1.entity: ＝ {all available tags}.

[0083] 2.predicate: ＝ {&&, ||, ！ }

[0084]

3. *expr* := ç | *entity* | *(expr)* | *! expr* | *expr && expr* | *expr* ||*expr*

[0085]

i. *expr -> ç*

[0086] ii.expr-＞entity

[0087] iii.expr-＞ (expr)

[0088] iv.expr-＞！ expr

[0089] v.expr-＞ expr&&expr

[0090] vi.expr-＞ expr//expr

[0091] 4.perm: ＝ expr

[0092] 5.role：$<r:perm_r, w:perm_w, x:perm_x>$

[0093] 另外，根据之前的层次性的模型，角色还是自动继承其祖先节点的权限，即读权限

还会扩展继承所有祖先的读权限，写权限还会扩展继承所有祖先的写权限，执行权限还会扩展继承所有祖先的执行权限。

[0094] 访问规则定义

[0095] 访问规则是整个策略的核心，无论是主体标签还是客体标签的定义都是服务于最终的访问规则的，访问规则才是策略的重点所在。这里针对于读，写，执行权限定义了三套访问规则。

[0096] 1. 预备定义

[0097] Tag 继承树定义

[0098] i> 定义 explict_tags(o) 为客体标签 <org，tag_1，tag_2，，...，tag_n> 中明确定义的 tag 集合。

[0099] ii> 定义 implicit_tags(o) 为 explict_tags(o) 中所有 tag 的祖先 tag 的集合。

[0100] iii> 定义 all_tags(o) 为 explict_tags(o) 和 implicit_tags(o) 的并集。

[0101] Role 继承树定义

[0102] i> 定义 explict_roles(s) 为主体标签 <org，$role_1$，$role_2$，...，$role_n$> 中明确定义的 role 集合。

[0103] ii> 定义 implicit_roles(s) 为 explict_tags(s) 中所有 role 的祖先 role 的集合。

[0104] iii> 定义 all_roles(s) 为 explict_roles(s) 和 implicit_roles(s) 的并集。

[0105] 2. 访问规则定义

[0106] 1) 读访问规则

[0107] 假定有主体 S 和客体 0，主体 S 能读客体 0 当且仅当以下条件满足：

[0108] a)org(S) = org(0)，

[0109] b) $\exists k \in all\_roles(S)$，使得在 all_tags(o) 中的 tag 变量值均为 true 的情况下，$perm_r(k)$ 的逻辑表达式计算结果也为 true（客体的所有 Tag 标签决定了当前权限表达式中 Tag 变量的输入值，如有客体有这个标签，那么权限表达式中的 Tag 变量为 true，否则为 false。这里就是说用客体所有的 tag 代入权限表达式的情况下，能够使得权限表达式（这里即 permr(k)）计算结果为 true；即存在赋予主体 S 的角色 R，使得 permr(R) 在客体 Tag 变量输入的条件下逻辑表达式计算结果为真）；

[0110] 即要求主客体所属组织一致，并且存在赋予主体的角色 k 使得此角色的读权限表达式（$perm_r(k)$）在客体 Tag 标签为真的情况下计算结果也为真。

[0111] 2) 写访问规则

[0112] 假定有主体 S 和客体 0，主体 S 能写客体 0 当且仅当以下条件满足：

[0113] a)org(S) = org(0)

[0114] b) $\exists k \in all\_roles(S)$，使得在 all_tags(o) 中的 tag 变量值均为 true 的情况下，$perm_w(k)$ 的逻辑表达式计算结果也为 true（即存在赋予主体 S 的角色 R，使得 $perm_w(R)$ 在客体 Tag 变量输入的条件下逻辑表达式计算结果为真）

[0115] 即要求主客体所属组织一致，并且存在赋予主体的角色 k 使得此角色的写权限表达式（$perm_w(k)$）在客体 Tag 标签为真的情况下计算结果也为真。

[0116] 3) 执行访问规则

[0117] 假定有主体 S 和客体 0，主体 S 能执行客体 0 当且仅当以下条件满足：

[0118] a)org(S) = org(0)

[0119] b) $\exists k \in all\_roles(S)$，使得在 all_tags(o) 中的 tag 变量值均为 true 的情况下，$perm_x(k)$ 的逻辑表达式计算结果也为 true（即存在赋予主体 S 的角色 R，使得 $perm_x(R)$ 在客体 Tag 变量输入的条件下逻辑表达式计算结果为真）

[0120] 即要求主客体所属组织一致，并且存在赋予主体的角色 k 使得此角色的执行权限表达式（$perm_x(k)$）在客体 Tag 标签为真的情况下计算结果也为真。

[0121] 4> 虚拟组织和中国墙

[0122] 企业之间有时因为相互协作关系存在着数据的共享，然而企业间数据的强隔离和企业数据的共享是相悖的。为了满足企业之间数据共享的特殊需求，这里引入虚拟组织的概念。虚拟组织指的是希望共享数据的几个企业共同建立的虚拟的组织，在现实中是不存在的。虚拟组织也可以有自己的用户和数据，通过为存在协作关系的企业用户之间创建虚拟组织的用户，并给这些企业某些私有数据打上虚拟组织的标签，那么根据之前定义访问控制规则便可以实现企业间数据的共享。此时，主体和共享的客体的企业属性都是此虚拟组织，它们是匹配的，因此允许主体访问客体资源。

[0123] 某些企业用户之间共享同一市场，之间存在着激烈的竞争关系，如中国移动和中国联通，中国石油和中国石化。这些公司由于双方利益存在很大的冲突，它们可能不希望和竞争对手共享数据，应当有相关的机制防止误操作或恶意操作导致它们之间数据的共享。这里可以应用中国墙模型（参考：THE CHINESE WALL SECURITY POLICY，Dr.David F.C.Brewer and Dr.Michael J.Nash），通过制定冲突集，保证同一冲突集内的企业无法建立虚拟组织。如将中国移动和中国联通放置在同一冲突集内，将中国石油和中国石化放在另一冲突集内，这样便可以防止它们数据的共享。

[0124] 与现有技术相比，本发明的积极效果为：

[0125] 本发明针对云存储的特殊环境，以RBAC(Role based Access Control)为基础，结合组织标签和多种安全属性的逻辑组合，提出了一种灵活的访问控制策略，一方面能够保证云端不同企业间数据的强隔离性，使得企业用户无法越权去访问其它企业用户的数据；另一方面该策略能够保证云存储企业内部数据的适度隔离，企业用户可以根据公司自身的安全需求灵活定制企业内的访问控制策略，隔离来自不同部门和地域的数据；最后该策略通过虚拟组织的概念在企业间数据强隔离的情况下实现了可能的数据共享，并通过中国墙策略保障同一冲突集中的企业是不能共享数据的。本文提出的策略遵循通用性原则，适用于公有云，私有云以及混合云。

附图说明

[0126] 图 1 层次化的主体角色模型；

[0127] (a) 角色层次树 a (b) 角色层次树 b (c) 角色层次树 c

[0128] 图 2 与图 1 对应的主体标签定义示例；

[0129] 图 3 层次化的客体 Tag 模型；

[0130] (a) 标签层次树 a (b) 标签层次树 b (c) 标签层次树 c

[0131] 图 4 与图 3 对应的客体标签定义示例；

[0132] 图 5 基于 HDFS 的安全体系结构设计；

[0133] 图 6 安全配置池的动态结构；

[0134] 图 7 权限表达式转化为后序表达式树示意图；

[0135] 图 8 后序表达树计算权限表达式值示意图；

[0136] 图 9 基于票据的访问控制示意图。

具体实施方式

[0137] 本发明基于 Hadoop 分布式文件系统 (HDFS) 开发了数据隔离的安全机制。分布式文件系统广泛用作云存储的底层基础设施，商业化的云存储 Cloudera 其底层使用正是 HDFS。使用开源的 HDFS 开发安全机制原型有利于对整个系统的效果评估。

[0138] 图 5 描绘了基于 HDFS 架构的安全体系结构设计。

[0139] 原型系统的核心是安全决策模块。为了判定主体是否有权限访问资源客体，安全决策模块需要首先获取主客体安全标签的信息及当前系统的安全策略。在 HDFS 中，文件系统的元信息存储在主节点 Namenode 一端，因此安全标签及安全策略这类安全元信息按照原有设计也应当存储在 Namenode 端，并由 Namenode 加以管理；从安全的角度上看亦是如此。如果将敏感的安全元信息分布在集群内多个 Datanode 节点上，那么势必增加安全的风险，因为整个系统需要保护的安全界限明显的增大了，保护单节点的 Namenode 会比保护多节点的 Datanode 要简单的多；将安全元信息放在单节点的 Namenode 上也便于系统的管理维护。由此可见，将系统安全元信息放在 Namenode 一端是良好的设计，它既遵循了原有系统的设计也便于管理维护，并且安全可靠性更高。

[0140] Namenode 端的安全决策模块在作出访问控制判定后，需要在整个分布式的系统中实施安全决策，包括数据块读写请求决策的实施。安全决策的实施是通过网络节点的交互完成的，包括 DFSClient 和 Namenode 之间元信息交互，DFSClient 和 Datanode 的数据流交互以及在写数据块过程中 Datanode 和 Datanode 之间的数据流交互。

[0141] 在 DFSClient 端提供了一些管理命令，通过这些命令接口可以查询修改 Namenode 上的安全元信息。当然绝大数这些命令是提供给管理员的，普通用户没有权限使用这些命令。

[0142] 下面说明各个关键模块的设计与实现。

[0143] 1> 主客体安全标签的设计与实现

[0144] 依照 HDFS 的原有设计，为了保证系统的性能，文件系统的元信息都是保存在 Namenode 内存中以供运行时快速访问。因此对于需要经常访问的主客体安全标签这类元信息，也应当存放在 Namenode 内存中以防止降低系统效率。客体标签可以存放在代表文件目录树节点的类 INode 内，但由于不存在现有的数据结构存放主体信息，主体标签需新建数据结构保存，可以通过哈希表的索引结构存储系统中所有的主体标签，在需要获取相应主体标签时根据主体名从哈希表中获取。

[0145] 内存中的主客体安全标签信息需要实现持久化机制以在系统断电后长期保存。Namenode 同时采用镜像文件 fsimage 和操作访问日志 edits 持久化内存中系统的元信息。镜像文件 fsimage 记录了整个文件系统的元信息，而操作日志 edits 则记录运行时对元信息的更新操作。Namenode 每次启动时会首先载入镜像文件 fsimage 以装载系统的元信息，

并在此基础上重放操作日志 edits 中的每一条操作记录将内存中的元信息更改至最新。在系统运行时，如果系统有操作修改了 Namenode 内存中的元信息，Namenode 会同时向操作日志写入该条操作的信息，以便下次重放。系统会不定时的将内存中最新的元信息重新写入 fsimage，并清空操作日志。通过应用操作日志的方法，Namenode 可以避免每次系统元信息更新都需要重新写入整个系统元信息镜像以防止系统异常崩溃导致的元信息更新丢失，同时还可以减少系统 I/O 保证性能。因此通过镜像文件 fsimage 和操作日志 edits 部分的代码修改来实现主客体安全标签的持久化 / 非持久化。

[0146] 2> 系统安全策略的设计与实现

[0147] 安全策略是另一类需要经常访问的信息。每个组织均可根据内部安全需求建立特有的主体角色模型和客体标签模型，存放在以 xml 文件表示的安全策略配置文件中。XML 层次化的格式使得它能够很好的适应这种角色。安全策略配置文件中描述了每个组织特有的角色层次树和 Tag 层次树，并为组织内所有角色定义了以字符串形式表达的读，写以及执行所要求的权限表达式。

[0148] 系统在初始化时载入所有企业用户的安全策略配置以提升运行时效率。在 Namenode 内存中为每个企业维护一个 Role 配置池 (RolePool) 和一个 Tag 配置池 (TagPool)，其中的 Role 对象和 Tag 对象即保存了企业用户相关的安全策略配置。在系统运行时刻，如果主体需要引用 Role 标签或客体需要引用 Tag 标签，可以直接通过安全配置池快速获取，而不用反复读取安全配置文件进行耗时的 I/O 操作。

[0149] 图 6 描述了安全配置池在内存中的动态结构。

[0150] 安全配置池通过两级索引可以快速定位到具体的 Role 对象和 Tag 对象。Organization Pool 为每个组织维护一个配置池，使用的是哈希表的索引结构，其主键为组织的名字 (orgName)，值包含 Role 配置池 (role pool) 和 Tag 配置池 (tag pool)。Role 配置池和 Tag 配置池同样使用的是哈希表结构。Role 配置池的键为角色的名字，值为角色对象 (Role Object)。Tag 配置池的键为 Tag 的名字，值为 Tag 对象 (Tag Object)。Role 对象使用 parent 指针指向在角色模型中的父亲节点，Role 对象通过 parent 指针这种相互引用的关系形成了一种层次性的结构，但并不是所有的 Role 都有这种祖先后辈的联系。从最上层看就形成了一片 Role 对象的森林。对于 Tag 对象也一样，Tag 对象使用 parent 指针相互引用最终也形成了一片 Tag 对象的森林。

[0151] 组织内的主体用户可以由该组织的管理员创建，主体的组织标签取决于创建它的管理员所属的组织，即某组织管理员创建的主体用户都隶属于该组织，该管理员不能创建其它组织的用户；客体资源的组织标签由创建客体的主体用户决定，即某组织的主体用户创建的客体资源默认属于该组织所有。组织管理员可以为组织内的主体用户增加角色标签，亦可为组织内的客体资源增加 Tag 标签，前提是该角色标签和 Tag 标签都在该组织的安全配置中有定义。在组织管理员为主体或客体执行增加角色标签或者 Tag 标签时，系统可以根据管理员用户所属的组织信息以及管理员提供的主体角色名或客体 Tag 名快速通过两次哈希表的查找操作迅速定位到相应的 Role 对象或者 Tag 对象，并且这些 Role 对象和 Tag 对象已经通过 parent 指针形成了树形的层次性结构。这样可以避免系统在运行时反复的读写安全策略配置文件，减少 I/O 操作，极大提高系统的性能。

[0152] 3> 安全决策模块的设计与实现

[0153] 安全决策模块的作用在于判定给定主体是否有权限访问给定的客体。由于角色权限的定义是由 Tag 标签变量和逻辑运算符所组成的逻辑表达式，如何在给定客体 Tag 集合的情况下高效的计算出权限逻辑表达式的值是非常关键。云存储系统在同一时间内需要负荷来自不同公司的成千上万的客户的访问，而权限判定是客户在访问每一个文件的时候都必须通过的，如果权限判定不够高效，一方面会降低客户端的访问速度，另一方面也由于服务器的负载增大导致单位时间内可以访问云存储的客户数量减少。因此访问控制决策判定的性能是非常重要的。

[0154] 为了把服务性能的损耗降低到最小，项目原型在 Hadoop 分布式文件系统初始化的时候将所有角色以字符串形式表示的权限表达式转化成后序表达式树这种中间形式。在系统运行时，HDFS 可以直接根据权限表达式树很快计算出决策结果，而不用再对权限表达式做字符串分析。这样一方面可以减少运行时客户访问文件系统的时间，提升系统的性能；另外一方面也可以避免同一角色在访问不同文件时反复对相同权限表达式进行分析，避免重复工作。

[0155] 权限表达式计算的算法思想如下：

[0156] 1> 系统初始化时动作：

[0157] 把角色权限定义的逻辑表达式转化为以表达树形式描述的后序表达式 (Postfix representation，又称逆波兰序)，见图 7。

[0158] 2> 系统运行时动作：

[0159] 在给定表达式变量值（即 Tag 标签值）输入的情况下，通过内存中的后序表达式树快速计算

[0160] 权限表达式的值，见图 8。

[0161] 4> 安全决策的实施

[0162] 读请求的安全控制

[0163] 根据电影票售票模式，HDFS 可以采用类似的安全实施方式，称为”基于票据 (Ticket) 的访问控制”。客户端在访问 Namenode 以获取数据块的位置信息时，Namenode 会首先进行根据主客体的安全标签相关安全决策判定，如果通过的话则会为客户端生成一个包含了访问决策的票据。客户端在访问 Datanode 上的数据时需要向 Datanode 出示 Namenode 授予的票据才能够正常的访问数据。为了防止恶意的客户端对票据进行伪造和篡改，Namenode 使用和所有 Datanode 共享的一组密钥对其进行加密，因此只有 Datanode 才能够解密票据查看其中的访问决策，从而判定当前的客户端是否有权利访问相关的数据块。

[0164] 根据 HDFS 客户端的访问流程，在加上了基于 Ticket 的访问控制之后框架图如图 9 所示：

[0165] 加粗的部分即是修改了 HDFS 网络通信的部分，主要是传递新增加的票据信息，其它的部分均是 HDFS 原本存在的部分。整个流程如下所述：

[0166] 1. **Client -> Namenode**:<filename，offset，length>

[0167] 客户端向 Namenode 发送要访问的文件名 (filename)，访问的文件偏移量以及要访问的数据的长度 (length)。

[0168] 2. **Namenode -> Client:** multiple<block handle，block locations>+AccessTicket

[0169] Namenode 在判定客户端对文件的访问权限通过之后，生成访问票据并用集群内共享的密钥对票据进行加密。Namenode 向客户端返回其要访问的多个数据块的标识 (block handle) 以及它们所处的 Datanode 的位置 (block locations)，最后是附加的加密后的访问票据。

[0170] 3. **Client -> Datanode:** <block handle，AccessTicket>

[0171] 客户端在获取了数据块的具体位置以及访问票据之后，对于每一个数据块选择离它最近 DataNode，向其发送客户端要访问的数据块标识 (block handle) 和访问票据 (AccessTicket)，请求访问数据块。由于访问票据被加密过了，客户端是不能随便篡改其中的内容的。

[0172] 4. **Datanode -> Client**: block stream

[0173] Datanode 取得 AccessTicket 之后，使用集群内共享的密钥解密 AccessTicket 判定当前客户端是否有权限访问相应的 block。如果允许访问，则向 client 回送数据块的数据，否则拒绝客户端的访问。

[0174] AccessTicket 包含了安全决策的信息，是由 Namenode 生成，之后通过 Client 转发给 Datanode，其定义如下：

[0175] $\text{AccessTicket} = E\{\text{user-id}, \langle \text{file block IDs} \rangle, \text{op}\}_{\text{secret}}$

[0176] 访问票据包括访问资源主体的标识符 (user-id)，主体要的所有的 block id(<file block IDs>)，以及访问的具体操作 (op)。为了防止客户端篡改 Ticket，Ticket 使用 HDFS 集群内共享的一组密钥 (secret) 加密。

[0177] 写请求的安全控制

[0178] HDFS 0.18 版本只支持创建写操作 (create)，文件一旦创建成功后就不能再被修改，这样也部分简化了写请求安全控制的工作量。

[0179] 客户端在分布式文件系统上创建新的文件时，遵循如下步骤：

[0180] 1. 客户端（即租户的主体用户，角色主体）开始新文件的创建

[0181] **DFSClient -> Namenode**: void ClientProtocol.create(...)

[0182] 客户端通过 rpc 远程调用 Namenode 实现的 create 方法，开始新文件的创建。Namenode 采用了一种称为租约 (Lease) 的方法，为新创建的文件增加排它锁。除了启动文件创建的用户可以访问此文件外，其他用户是不能访问文件的。此时尚未完成创建的文件在 Namenode 内存中以 INodeFileUnderConstruction 对象表示。

[0183] 2. 客户端写入文件的数据块

[0184] 对于每一个数据块，执行如下操作：

[0185] i> **DFSClient -> Namenode**: LocatedBlock ClientProtocol.addBlock(...)

[0186] 客户端 rpc 远程调用 Namenode 实现的 addBlock 方法，为文件新增数据块。Namenode 在接收到 addBlock 请求后会返回给客户端一个 LocatedBlock 对象。此对象中包含了新建数据块的标识，并说明了应该往哪几个 Datanode 中写入数据块。

[0187] ii> **DFSClient -> Datanode**: block stream

[0188] 客户端和 LocatedBlock 对象中包含的 Datanode 之间建立起 pipeline 连接，通过网络 I/O 流往 Datanode 中写入数据块的数据。

[0189] 3. 客户端完成文件的创建

[0190] **DFSClient -> Namenode**: void ClientProtocol.complete(...)

[0191] 客户端 rpc 远程调用 Namenode 实现的 complete 方法，完成新文件的创建。Namenode 将 INodeFileUnderConstruction 对象转化为正常的 INodeFile 对象，并解除文件上的排他锁。

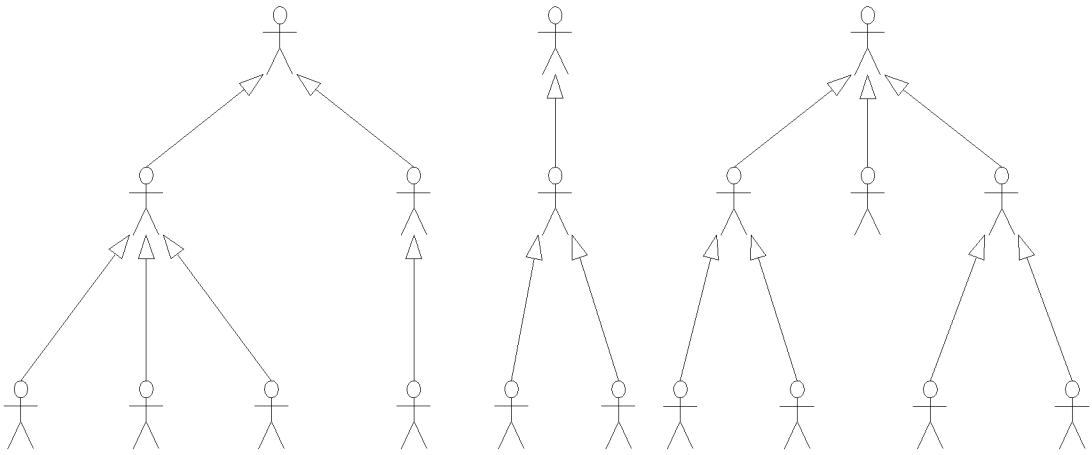
[0192] 根据以上的数据写访问过程，HDFS 的写访问控制不需要向读访问控制那样使用基于票据的方式。由于在 HDFS 的 Namenode 端实现了基于 Lease 的排他锁，其它的用户是不能够访问尚未创建完成的文件的。而在文件创建完成之后就不能修改，这样大大简化了文件写请求的安全控制。只需要在 DFSClient 向 Namenode 发起 create 远程调用开始文件创建的时候对其进行访问控制即可。

[0193] 5> 用户命令的设计

[0194] 为了方便各个组织管理员的操作，在系统的客户端实现了相应的管理员命令。可以使用这些命令对系统用户，用户角色以及客体的 Tag 标签进行管理。所有的这些命令都实现在 Hadoop 原有的 FSShell 中，命令说明如下：

[0195]

	命令格式	命令描述
增加用户	$./hadoop dfs -addUser <user name> {<role name>}*	组织管理员可以使用-addUser 命令为组织内成员创建用户，并可赋予新建用户相关角色。
删除用户	$./hadoop dfs -removeUser <user name>	组织管理员可以使用-removeUser 命令删除组织内的用户。
为已有用户赋予角色	$ ./hadoop dfs -addUserRole <user name> {<role name>}*	组织管理员可以使用-addUserRole 命令为已有的用户赋予新的角色
删除用户已有角色	$./hadoop dfs -removeUserRole <user name > {<role name>}*	组织管理员可以使用-removeUserRole 命令删除用户已有的角色
为文件或目录增加标签	$./hadoop dfs –addTag <file path> {<tag name>}*	组织管理员可以使用-addTag 命令为系统的文件或者目录增加 Tag 标签
删除文件或目录已有标签	$./hadoop dfs -removeTag <file path> {<tag name>}*	组织管理员可以使用-removeTag 命令删除文件或目录已有的 Tag 标签。
显示文件或目录的标签信息	$./hadoop dfs -listTag <file path>	可以使用-listTag 命令显示文件或者目录的标签信息



（a） （b） （c）

图 1

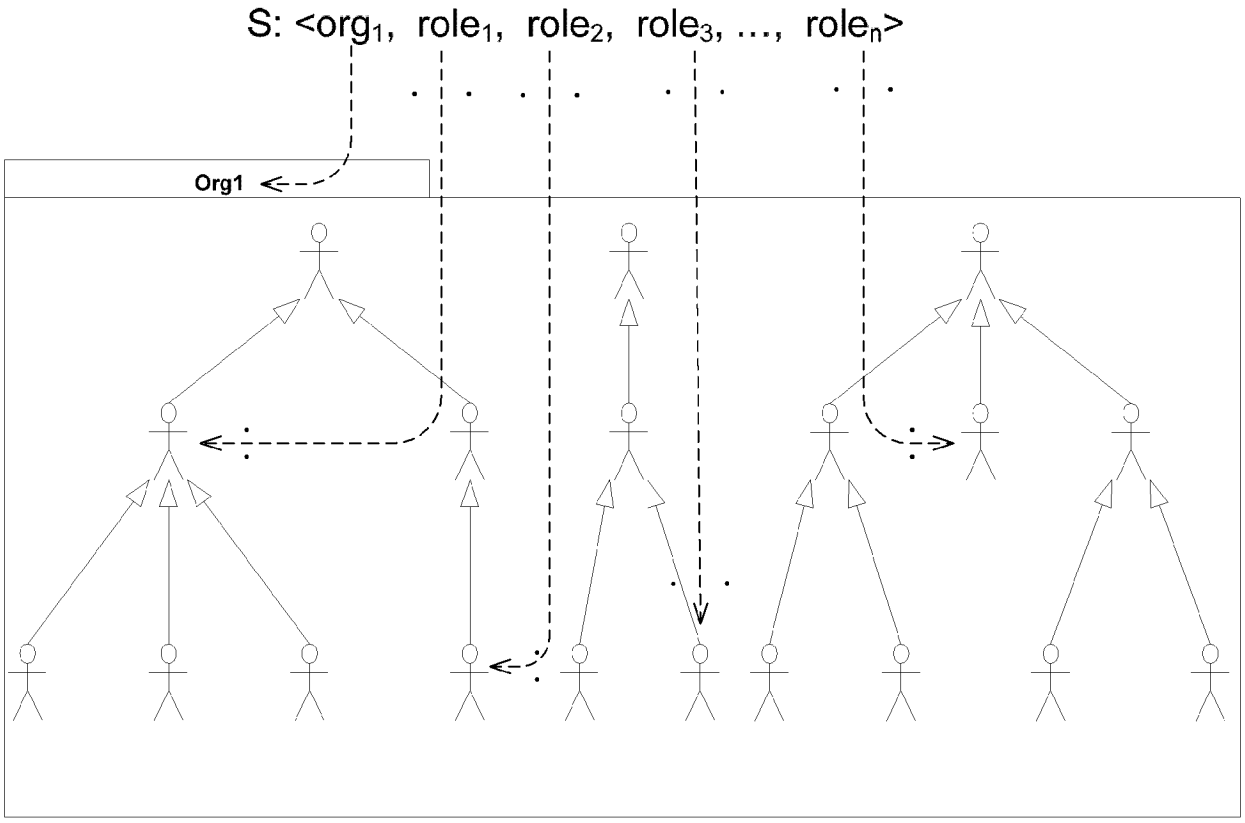
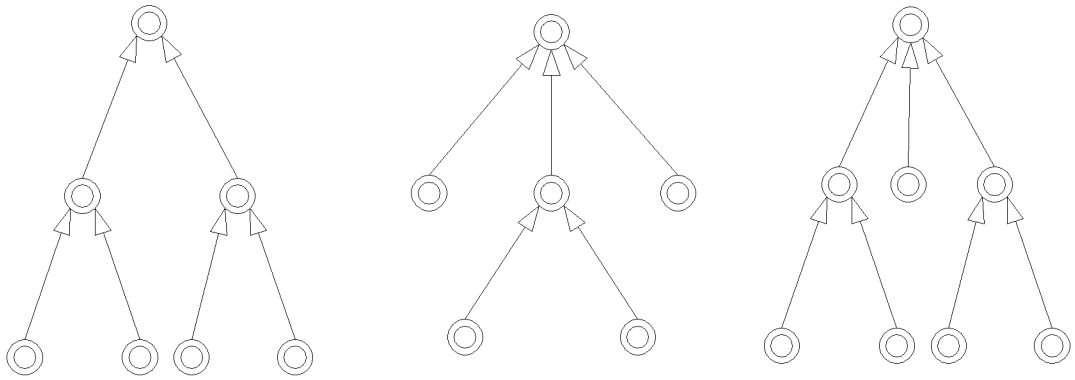


图 2



（a） （b） （c）

图 3

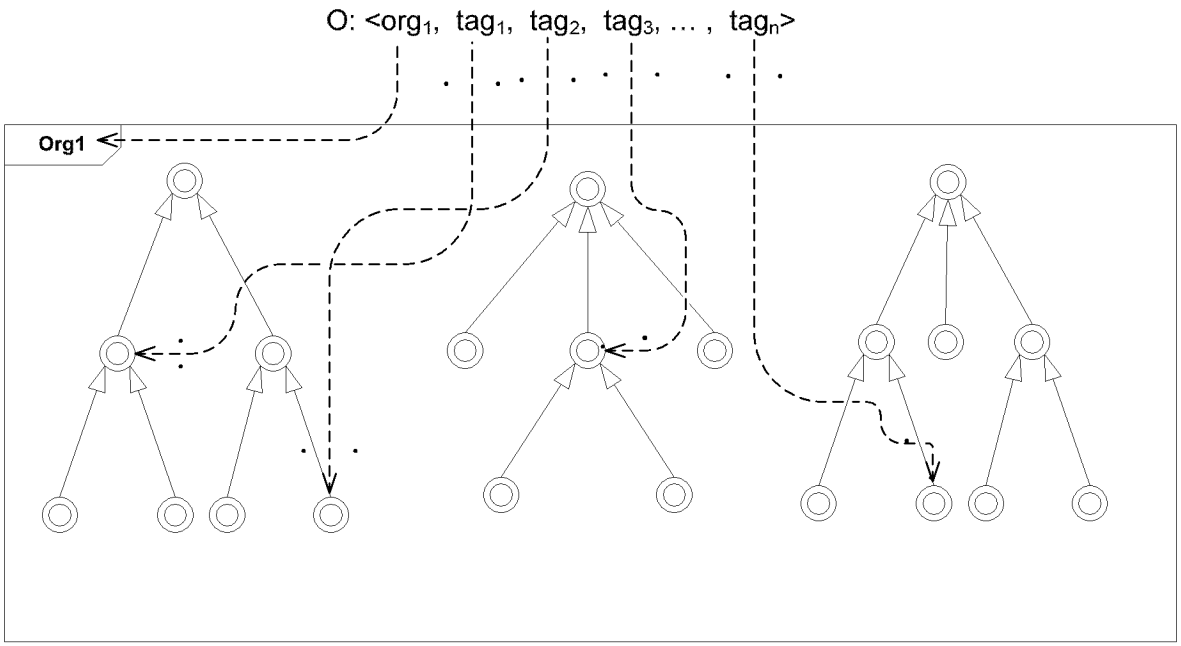


图 4

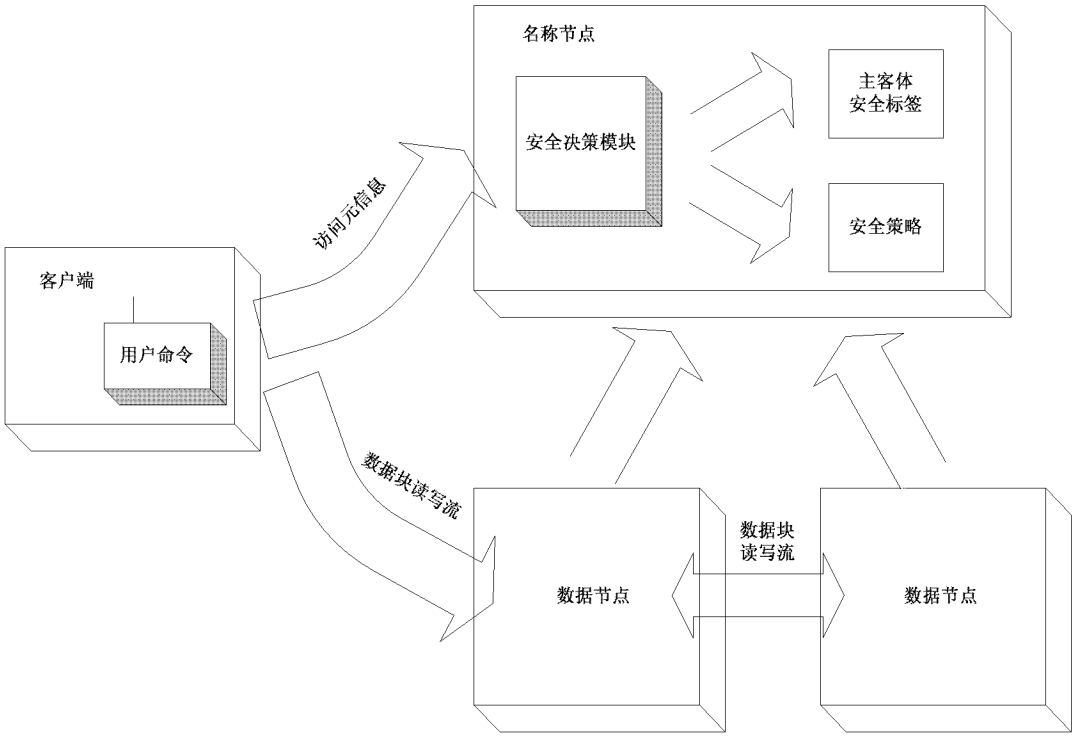

名称节点
安全决策模块
主客体
安全标签
安全策略
访问元信息
客户端
用户命令
数据块读写流
数据块
读写流
数据节点
数据节点

图 5

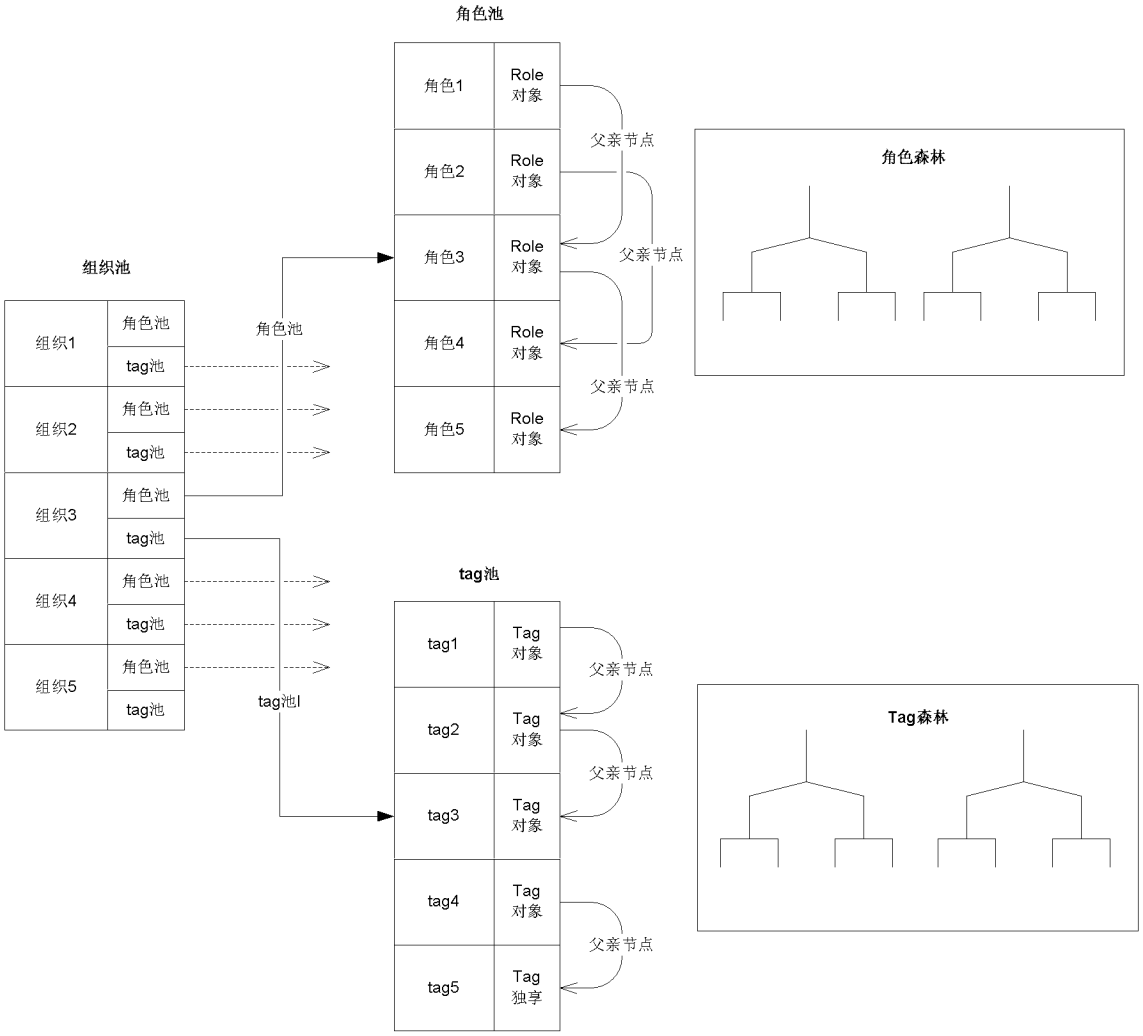

角色池
角色1
Role 对象
角色2
Role 对象
角色3
Role 对象
角色4
Role 对象
角色5
Role 对象
父亲节点
父亲节点
父亲节点
角色森林
组织池
组织1
角色池
tag池
组织2
角色池
tag池
组织3
角色池
tag池
组织4
角色池
tag池
组织5
角色池
tag池
角色池
tag池l
tag池
tag1
Tag 对象
tag2
Tag 对象
tag3
Tag 对象
tag4
Tag 对象
tag5
Tag 独享
父亲节点
父亲节点
父亲节点
Tag森林

图 6

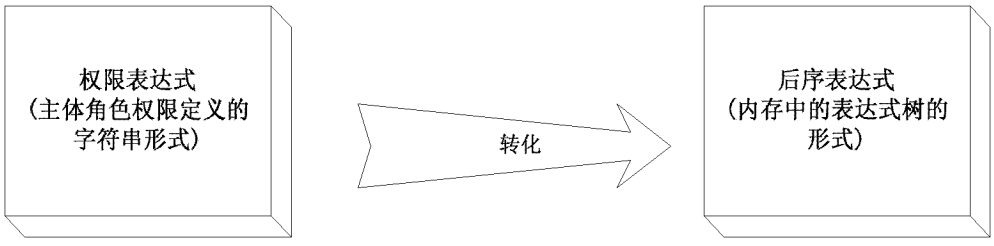

权限表达式
(主体角色权限定义的
字符串形式)
转化
后序表达式
(内存中的表达式树的
形式)

图 7

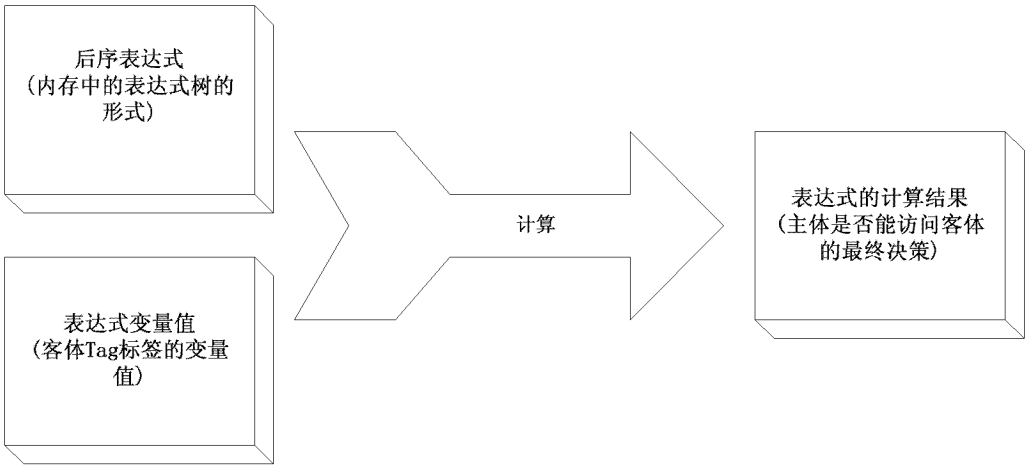

后序表达式
（内存中的表达式树的形式）
表达式变量值
（客体Tag标签的变量值）
计算
表达式的计算结果
（主体是否能访问客体的最终决策）

图 8

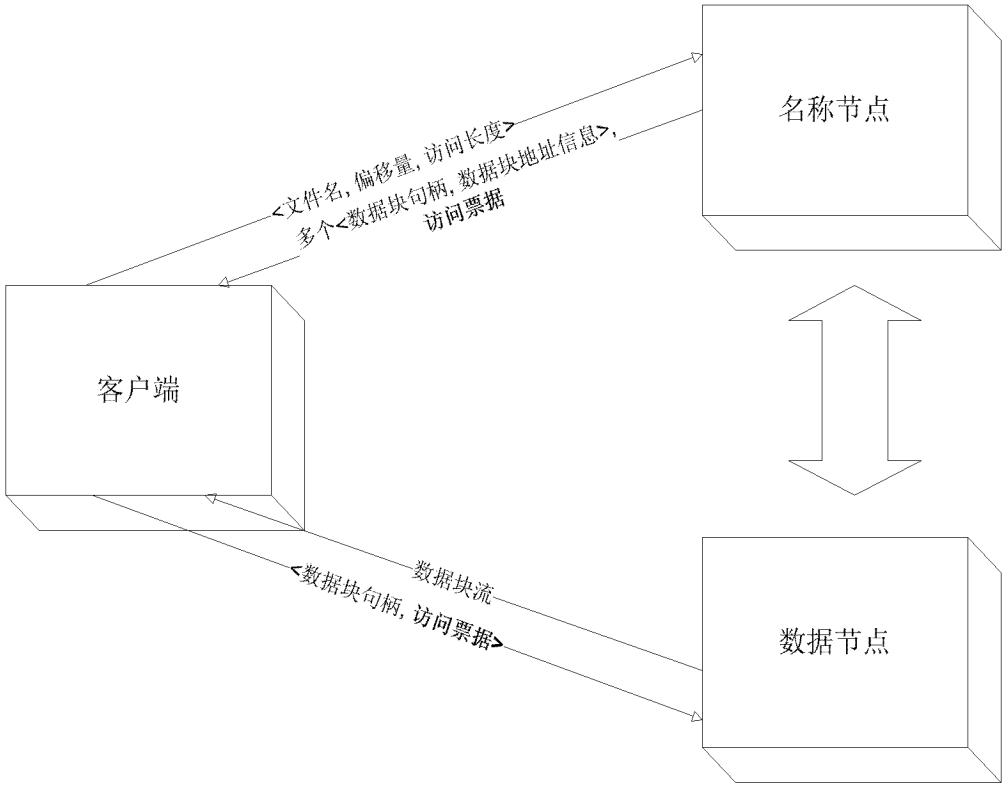

名称节点
<文件名，偏移量，访问长度>
多个<数据块句柄，数据块地址信息>，
访问票据
客户端
数据块流
<数据块句柄，访问票据>
数据节点

图 9