ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.9, September 2009, pp.2495−2510
doi: 10.3724/SP.J.1001.2009.03324

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

基于层次角色委托的服务网格授权执行模型*

陈志刚[1], 桂劲松[1+], 郭 迎[2]

[1](中南大学 信息科学与工程学院,湖南 长沙 410083)

[2](上海交通大学 区域光纤通信网与新型光通信系统国家重点实验室,上海 200030)

Hierarchical Role-Based Delegation Authorization Execution Model for Service Grid

CHEN Zhi-Gang[1], GUI Jin-Song[1+], GUO Ying[2]

[1](School of Information Science and Engineering, Central South University, Changsha 410083, China)

[2](National Key Laboratory on Optical Communication and New Optical System, Shanghai Jiaotong University, Shanghai 200030, China)

+ Corresponding author: E-mail: jsgui06@163.com, http://www.csu.edu.cn

Chen ZG, Gui JS, Guo Y. Hierarchical role-based delegation authorization execution model for service grid. *Journal of Software*, 2009,20(9):2495−2510. http://www.jos.org.cn/1000-9825/3324.htm

Abstract: To satisfy various requirements restricted delegation of in grid applications, a hierarchical role-based delegation authorization execution model for service grid is proposed. The dynamic characteristic of delegation role granting or revocation and the associated constraint of delegation role granting are effectively supported. The fine-grained associated role dependency is implemented by adding trustworthiness. Partial delegation problem is easily solved by defining the role tree as the basic unit of delegation authorization and by pruning the role tree. The delegation spread tree with trustworthiness is defined to implement multi-step delegation in a fine-grained manner. The delegation certification is proposed to fully express temporary delegation, associated role delegation, partial delegation, multi-step delegation. Based on above works, a set of formal delegation authorization execution rules is proposed and proved, and the delegation authorization execution process is effectively controlled. The exhibited example shows that the model can satisty various requirements of restricted delegation in grid applications.

Key words: service grid; hierarchical role delegation; delegation certification; authorization execution; fine-grained control

摘 要: 针对网格应用中对委托限制的多方面需求,提出了基于层次角色委托的服务网格授权执行模型.模型支持委托角色授予与撤销功能以及相应的关联性限制特性.通过加入信任度,细化了关联性限制的表达粒度.通过定义角色树作为委托授权的基本单位并对角色树进行剪枝,改善了部分委托实现的难度.通过定义带信任度的委托传播树,细化了对委托传播限制的控制.提出的委托凭证全面支持了角色委托的临时性、关联性、部分性、传播性限制需求.对模型中的委托授权执行规则作了形式化描述,并证明了执行规则能够细粒度地控制委托授权的执行过程.实例展示表明,模型能够满足网格应用对委托限制多方面的需求.

关键词: 服务网格;层次角色委托;委托凭证;授权执行;细粒度控制

* Supported by the National Natural Science Foundation of China under Grant No.60573127 (国家自然科学基金); the Hu'nan Provincial National Natural Science Foundation of China under Grant No.07JJ3128 (湖南省自然科学基金)

Received 2007-09-08; Accepted 2008-03-27

中图法分类号: TP393 文献标识码: A

依据开放网格论坛(open grid forum,简称 OGF)的 OGSA 授权工作组(OGSA authorization working group,简称 OGSA-AUTHZ)[1]描述的通用网格授权框架,对现有典型网格授权系统进行分析:类似 CAS[2]等系统的各种目标资源的访问策略信息会预先委托给 CAS 服务器所在的虚拟组织(virtual organization,简称 VO),从而提高了决策响应的速度.但它没有考虑到资源的动态性,资源的动态变化使得 VO 的授权策略需要作相应调整.类似文献[3−5]等系统的输入信息通常是分布式的,适应了资源的动态性和网格系统的扩展性要求,但授权信息的网络传输延时会影响授权系统决策的响应速度.其次,PDP(policy decision point)输出的结果是简单的“允许”或“拒绝”.这使得 PEP(policy enforcement point)功能过于简单而 PDP 功能过于复杂,不利于独立授权过程形成流水线式的执行效果.同时,它也会降低 PDP 的决策成功率.例如,未通过决策的访问请求在随后的某个时刻又会要求再决策,这次可能因条件满足而得以通过.若在第 1 次决策时 PDP 就给出一个具有条件限制的决策结果,则会节省重复决策的开销.

按需、动态、即时构建服务虚拟组织以协同进行问题求解,是网格发展的必然趋势.这种新趋势为结合集中式和分布式授权系统的优点构建新的网格授权系统提供了基础.本文主要关注网格虚拟组织的 PEP 的工作,并提出基于层次角色委托的授权执行模型(hierarchical-role based delegation authorization execution model,简称 HDAEM)来解决授权执行的细粒度控制问题.第 1 节介绍本文的研究目的和委托的相关研究工作.第 2 节以形式化方式详细描述基于层次角色委托的授权执行模型,并与相关模型进行比较.第 3 节给出反映本文模型特点的详细示例.第 4 节总结本模型的特点并展望下一步的工作.

1 研究目的和相关工作

我们提出如图 1 所示的新授权体系结构,其特点是:

① 由分布式授权组件组成,但由 VO 构建者控制并部署在网络条件较好的区域,避免了组件间可能存在的高通信开销;

② 可以随 VO 动态构建而更新授权信息;

③ 可以采用委托的方式聚集资源的访问控制策略到 VO 中以降低跨域授权管理的复杂性;

④ 可用资源聚集时的策略协商和签定契约确保授权信息既动态可变又协调一致;

⑤ 通过在 VO 中设置一个 PEP 来分担 PDP 繁重的工作并简化资源本地域 PEP 的工作.

图 1 的上部为 VO 授权系统,其中,VRPR,VAR,VAPR,VTIP 分别为资源访问控制策略库、属性库、授权策略库以及信任信息库.这些库中的信息会随着 VO 的动态构建而更新,以便与 VO 的新目标一致.VPDP 是 VO 的授权决策组件,它依据上述库中的信息进行授权决策并输出一个授权证书.VPEP 是 VO 的授权执行组件,当它截获用户对资源的访问请求时,首先检查与请求相关的证书是否存在并且可用.若是,则依据授权证书控制授权在 VO 层次上的实施;否则,请求 VPDP 输出授权证书供 VPEP 使用或等待证书变为可用状态.图 1 的下部为自治域授权系统.LPEP 负责截获来自 VO 的访问请求,然后以本地访问者相似的方式请求对资源的访问.来自 VO 的访问请求需要同时提交来自 VO 的授权证书.通过委托的方式,能够方便地实现自治域向 VO 共享资源.自治域的资源所有者以委托的方式将自己愿意共享的资源访问控制策略提交到 VO 并存放在 VRPR 中.VO 的构建者依据 VO 目标制定的授权策略和属性信息分别存放在 VAPR 和 VAR 中.VO 维护的信任信息存放在 VTIP 中.由于 VPDP 是依据 VRPR,VAR,VAPR,VTIP 中的信息进行决策的,所以其输出的授权证书兼顾了 VO 和自治域两方面的安全需要.因此,通过 VPEP 检验的授权证书到了自治域通常是有效的,即符合本地访问控制策略.因此,它避免了无效的 VO 访问请求加重自治域授权系统的负担.本文研究 HDAEM 模型,其内容包括推理功能完备的委托授权执行规则以及表达功能完备的授权证书,我们称为委托凭证.资源提供者共享给 VO 的资源访问控制策略和 VPDP 的决策结果都需要委托凭证来表达,而委托凭证又是 VPEP 基于委托授权执行规则细粒度控制用户对资源的访问的重要依据,因此,对功能完备的受限委托机制的需求是必要的.

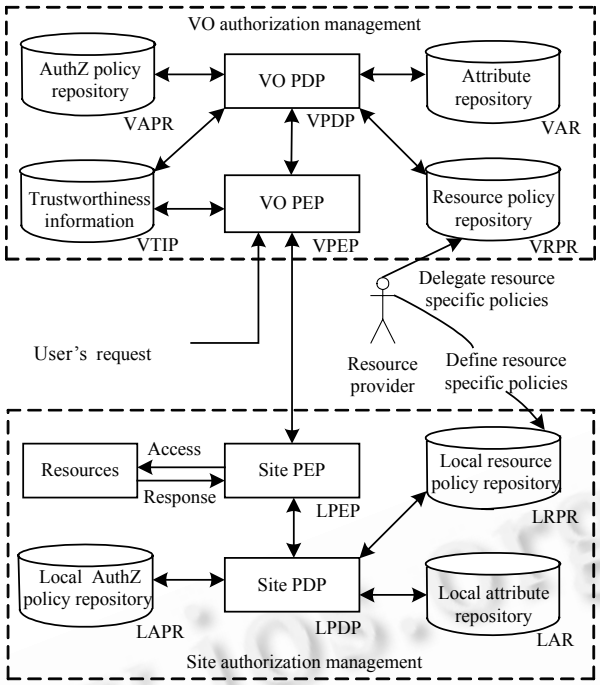


Fig.1 A grid authorization architecture

图 1 一种网格授权体系结构

网格中典型的委托相关研究工作有:网格安全基础设施(GSI)通过代理证书实现委托[6].代理证书的 PCI 扩展项具有如下功能:① 用作鉴别代理证书;② 为发布者表达委托意愿;③ 限制代理证书的路径长度.PCI 的策略方法标识符域用于指明使用哪种策略语言,而策略域中的内容是由发布者选择的方法声明的策略.在 CAS[2]中,资源提供者大粒度地将资源访问策略委托给 CAS 服务器.CAS 服务器根据用户请求签署访问资源的能力证书,并利用代理证书携带能力证书以便于网络传输.能力证书表达的是主体与权限的直接赋值关系,没有利用 RBAC[7]的优势.文献[8]描述了一种 DIS(delegation issuing service)服务,并将其应用到 PERMIS 授权系统中.DIS 替权威源或属性权威以属性证书的形式委托其权威给其他属性权威或终端实体.它的优势在于方便了委托记录的安全审计,但属性证书链的验证会变得复杂.采用基于角色的委托授权技术来实现资源提供者向 VO 的委托授权能够简化 VO 的授权管理工作.文献[9]在这方面取得了不错的研究成果,但针对我们的研究问题,仍需要对如下不足进行改进:① 对委托的临时性、关联性、部分性、传播性限制的支持不全面;② 没有对委托角色指派的动态特性进行很好的支持;③ 采用分解角色的方法实现部分委托增加了系统角色的种类,使虚拟组织管理复杂化;④ 对委托传播的支持不够,缺乏委托传播的上下文信息.

在基于角色的受限委托研究方面也有不少工作.文献[10]的工作支持层次角色环境下的多步委托(传播性),但未对多步委托限制作出描述;文献[11]只支持层次角色环境下的时间限制(临时性);文献[12]实现部分委托时,对系统角色种类进行增添,因而增加了系统管理的复杂性;文献[13]只支持扁平角色环境下的临时性、委托角色激活与去活的关联性限制.这种关联性限制可以表达动态职责分离约束,但不能表达静态职责分离约束,而且对关联性限制的表达粒度不够细.例如,当甲向银行贷款要求乙担保,现有关联性限制将其表示为:在乙激活"担保人"角色的前提下,甲才能激活"贷款人"角色.这样的表达粒度显然不够细,因为银行会根据贷款额度的不同对乙激活"担保人"角色的信任度也会有区别,而现有表达方式无法体现这种区别;文献[14]采用信任度控制委托传播深度方向上的委托角色的激活,但未涉及委托传播宽度控制的问题,而此问题在本文研究的 HDAEM 模型中显得尤为必要.当资源提供者将资源共享给 VO 时,它可能向 VO 委托其资源访问权,对 VO 的每个用户都有

个性化的约束要求,需要为各种约束提供单独的细粒度表达方式.其他与角色委托和受限委托相关的国内外工作在文献[9,13,14]等文献中都有详细阐述,故本文从略.

本文在文献[13]工作的基础上增加对层次角色环境下部分委托限制、委托传播限制的支持,以完善对委托限制的多种需求,并增加委托角色授予与撤销的关联性限制功能来克服文献[13]所不能表达的静态职责分离约束的缺陷.为了简化 VO 授权管理工作,我们只在 VO 中使用各自治域委托给 VO 的委托角色而不设置 VO 角色,网格用户可以通过获得委托角色来访问网格资源.我们提出动态授予或撤销委托角色的方法来解决委托角色数量相对较少与用户数量相对较多的矛盾.我们在关联性限制中增加信任度约束以克服现有关联性限制表达能力弱的缺陷,在部分委托限制中采用对层次角色剪枝的方法来降低分解角色实现部分委托的复杂性,在委托传播宽度和深度两个方向上,采用整数值控制委托角色的授予以及采用信任度控制已授予委托角色的激活来细粒度控制权限传播.通过定义完整表达委托需求的委托凭证以满足委托授权对策略表达的多方面需求,并在此基础上提出了一套完备的委托授权执行规则作为细粒度控制授权执行的手段.

2 基于层次角色委托的授权执行模型

2.1 模型定义

RBDM0[15]是支持扁平角色的委托模型,文献[13]提出的 CRDM 是基于 RBDM0 的.RBDM1[15]是支持层次角色的委托模型,本文基于 RBDM1 对 CRDM 进行扩展并提出 HDAEM 模型.文献[7]定义的部分基础元素为:① $U=\{u_1,u_2,\ldots,u_m\}$是所有用户的集合;② $R=\{r_1,r_2,\ldots,r_n\}$是所有角色的集合;③ $UA\subseteq U\times R$ 是从用户集合到角色集合的多对多映射.RBDM0 将 *UA* 划分为 *UAO* 和 *UAD*,分别表示用户与被授予的角色的关系和用户与其被委托的角色的关系.常规用户是由系统管理员指派角色的用户,委托用户是通过委托指派角色的用户.在 RBDM1 中,控制委托授权的语义有:

① $\forall r_1,r_2\in R,(r_1,r_2)\in can\_delegate$ 表示 r_1 的显性或隐性常规用户能使得 r_2 的显性或隐性常规用户成为层次低于或等于 r_1 的任意其他角色的显性委托用户,即可以向下委托;

② $\forall r_1,r_2\in R\wedge r_1\succeq r_2\Rightarrow(r_2,r_1)\notin can\_delegate$ 表示低层次角色的用户不能委托角色到高层次角色的用户,即向上委托无意义;

③ $\forall r_1,r_2\in R\wedge(r_1,r_2),(r_2,r_1)\in can\_delegate\rightarrow r_1\|r_2$ 表示无角色继承关系的成员之间具有相互委托权,用符号$\|$表示,即可以交叉委托.

通常由系统管理员对常规用户进行角色指派,所以 *UAO* 是一个静态集合,在系统运行过程中不能动态改变.而委托用户的角色是由其他用户动态指派的,也可以视情况随时撤销.因此,*UAD* 是一个动态集合.在层次角色环境下,委托角色与被委托者需要满足特定的约束关系,即 RBDM1 的委托授权语义.我们将满足 RBDM1 委托授权语义的用户、角色对(u,r)的集合称作资格集合 *CAN_UAD*.只有 *CAN_UAD* 的元素才能视用户请求情况动态进入或退出集合 *UAD*.

委托角色激活依赖[13]是一个集合,表示为 $dep\in DEP$,其中,任意元素表示为 *uad* 或$\neg uad$,$uad\in UAD$.但正如前面所述,这种表示方式粒度较粗,我们定义带信任度的委托角色激活依赖来提供细粒度的表示.

定义 1(带信任度的委托角色激活依赖).

• 带信任度的委托角色激活依赖是一个集合,表示为 $dept\in DEPT$,其中任意元素表示为 uad^T,*uad* 或$\neg uad$,$uad\in UAD$.$T\in[0,1]$表示信任度阈值,*uad* 表示依赖的处于激活状态的用户、委托角色对,$\neg uad$ 表示依赖的处于非激活状态的用户、委托角色对,uad^T 表示依赖的处于激活状态的信任度不低于 *T* 的用户、委托角色对.对于无上标 *T* 的 *uad*,默认信任度阈值 $T=0$.

例 1:激活用户、委托角色对$(u',r')\in UAD$ 依赖于集合 $dept=\{(u,r)^T\}$.若用户 *u* 的信任度不低于 *T* 且(u,r)处于激活状态,则(u',r')可以被激活.

定义 1 的依赖关系可以表达动态职责分离约束,但不能表达静态职责的分离约束.因此,我们需要定义带信任度的委托角色授予依赖.

定义 2(带信任度的委托角色授予依赖、函数 *grant*()、函数 *nogrant*()).

• 带信任度的委托角色授予依赖是一个集合,表示为 $degt\in DEGT$,其中,任意元素表示为 uad^T,uad 或 $\neg uad$,$uad\in UAD$.$T\in[0,1]$表示信任度阈值,uad 表示依赖的处于授予状态的用户、委托角色对,$\neg uad$ 表示依赖的处于非授予状态的用户、委托角色对,uad^T 表示依赖的处于授予状态的信任度不低于 T 的用户、委托角色对.对于无上标 T 的 uad,默认信任度阈值 T=0.

• $grant(degt)=\{uad|uad\in degt\}$,得到依赖的授予状态的用户、委托角色对.

• $nogrant(degt)=\{uad|\neg uad\in degt\}$,得到依赖的非授予状态的用户、委托角色对.

• $\forall degt:DEGT\bullet grant(degt)\cap nogrant(degt)=\varnothing\wedge degt=grant(degt)\cup\{\neg x|x\in nogrant(degt)\}$.

在定义 2 中,$uad=(u,r)$形式的元素表示用户 u 被授予角色 r 是授予某用户某委托角色的必要条件,$\neg uad=\neg(u,r)$形式的元素正好相反.在定义 2 中,第 4 条性质表示任意授予依赖 $degt$ 仅由定义中两种形式的用户、委托角色对组成,并且它们不相交.即不存在既依赖于某个用户授予特定角色,又依赖于某个用户非授予特定角色的情况.定义 2 中,uad 的信任度阈值 T 只有在有具体要求时才以上标形式标出,若无具体要求一般不标出.

例 2:用户 u 不能同时担任会计和出纳工作,角色 r 被赋予做会计工作的权限.用户 u 担任出纳工作的静态职责分离约束表示为 $degt=\{\neg(u,r)\}$,而动态职责分离约束则表示为 $dept=\{\neg(u,r)\}$.

基于定义 1 和定义 2,我们将 CRDM[13]的五元组委托票据 $dt=(uad,pt,n,ae,dep)$扩展为六元组委托票据 $dtt=(uad,pt,n,ae,dept,degt)$.

定义 3(角色树、委托传播树、委托凭证).

• 定义角色树 $r_{tree}=(r_0(r_1(r_{11}(r_{111}(\dots),\dots,r_{11n}(\dots)),\dots,r_{1m}(\dots)),\dots,r_k(r_{k1}(\dots))))$作为委托授权的基本单位.

• 在委托票据 $dtt=(uad,pt,n,ae,dept,degt)$中,用 r_{tree} 替代 $uad=(u,r)\in UAD$ 中的 r,并将 r 看成 r_{tree} 的未给出树形结构的简写形式.

• 定义 $spr_{tree}=(dtt_0^{T0}(dtt_1^{T1}(dtt_{11}^{T11}(dtt_{111}^{T111}(...),...,dtt_{11n}^{T11n}(...)),...,dtt_{1m}^{T1m}(...)),...,dtt_k^{Tk}(dtt_{k1}^{Tk1}(...))))$ 为委托传播树.树的根结点拥有从常规用户得到的委托票据 dtt_0.$dtt_1,dtt_2,\dots,dtt_k$ 是根结点可签发的委托票据,根结点可将 dtt_0 的角色树的一部分或全部委托给 $dtt_1,dtt_2,\dots,dtt_k$ 的持有者.$dtt_{11},dtt_{12},\dots,dtt_{1m}$ 是 dtt_1 的持有者可签发的委托票据,dtt_1 的持有者可将 dtt_1 的角色树的一部分或全部委托给 $dtt_{11},dtt_{12},\dots,dtt_{1m}$ 的持有者,其余依次类推.委托票据 dtt^T 的上标 T 为信任度阈值.在 dtt^T 的 $uad=(u,r)$中,u 的信任度不低于 T 才能激活 r.

• 委托凭证由 $dc=(n_d^T,n_b^T,spr_{tree})\in DC$ 组成,其中,n_d 表示委托权限传播的许可步数,即委托深度;n_b 表示在每步中权限被委托的许可用户数,即委托宽度;n_d^T 和 n_b^T 的上标 T 为信任度阈值.在委托传播深度和 宽度上每一个 dtt 的 $uad=(u,r)$中,u 的信任度不低于 T 才能激活 r.对于无上标的 n_d 和 n_b,默认信任度阈值 T=0.

• 委托凭证序列的集合组成委托凭证序列集,记作 DCS.

图 2 是某个角色 r_0 的层次关系,通常是一个有向无环图.r_0 是入度为 0 的结点,根据有向无环图的拓扑排序方法,可穷尽以 r_0 为初始结点的线性序列.若将 r_0 看作一棵树的根,全部序列看作从根到叶的路径,则可将有向无环图转换成一个倒转树形结构,以广义表形式表示为$(r_0(r_1(r_{11}(r_{111},r_{121}),r_{12}(r_{121},r_{211})),r_2(r_{21},r_{22}(r_{111},r_{211}))))$.

利用角色树 r_{tree} 容易实现部分委托,通常将不允许委托的角色分枝剪去即可.例如:

若角色 r_0 的用户欲将 $r_{0_tree}=(r_0(r_1(r_{11}(r_{111},r_{121}),r_{12}(r_{121},r_{211})),r_2(r_{21},r_{22}(r_{111},r_{211}))))$中除$(r_2(r_{21},r_{22}(r_{111},r_{211})))$外的角色权限委托给角色 r_{12} 的用户,则剪枝后的角色树为 $r'_{0\_tree}=(r_0(r_1(r_{11}(r_{111},r_{121}))))$;

若角色 r_1 的用户欲将 $r_{1_tree}=(r_1(r_{11}(r_{111},r_{121}),r_{12}(r_{121},r_{211})))$中除$(r_{12}(r_{121},r_{211}))$外的角色权限委托给角色 r_2 的用户,则剪枝后的角色树为 $r'_{1\_tree}=(r_1(r_{11}(r_{111},r_{121})))$.

spr_{tree} 的定义在形式上同 r_{tree} 但含义不同.r_{tree} 表示的是角色继承的层次关系,而 spr_{tree} 表示允许委托角色传播的范围.图 3 是一个委托传播树的例子(图中略去了信任度阈值的标注),可用 k 叉树来表示,k 对应 n_b,树高对应 n_d.其广义表表示形式如下:

$$(dtt_0^{T0}(dtt_1^{T1}(dtt_{11}^{T11}(dtt_{111}^{T111},dtt_{12}^{T12}(dtt_{121}^{T121},dtt_{122}^{T122})),dtt_2^{T2}(dtt_{21}^{T21},dtt_{22}^{T22}(dtt_{221}^{T221},dtt_{222}^{T222})))).$$

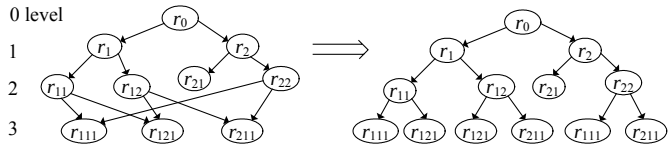


Fig.2 An example of hierarchal role relationship

图 2 层次角色关系示例

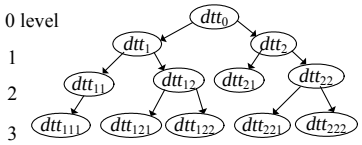


Fig.3 Role delegation spread tree

图 3 角色委托传播树

n_d^T 和 n_b^T 的上标 T 标出的信任度阈值是委托源对委托传播树上委托角色激活的最低信任度要求,但通过 spr_{tree} 的结点 dtt^T 的上标 T 可以给出更严格的个性化限制,以满足不同结点的个性化需求.

例 3:在 $dc_0=(1^{0.6},2^{0.6},(dtt_1^{0.6}))$中,委托源(即资源提供者)通过 dc_0 限制委托传播树上激活委托角色的用户最低信任度不低于 0.6, $dtt_1^{0.6}$ 为委托源签发的委托票据. $dtt_1^{0.6}$ 的持有者(即 $dtt_1^{0.6}$ 的 $uad=(u,r)$中的用户 u)也可签发 $dc_1=(1^{0.6},2^{0.6},(dtt_1^{0.6}(dtt_{11}^{0.7},dtt_{12}^{0.8})))$给 $dtt_{11}^{0.7}$ 的持有者以及 $dtt_{12}^{0.8}$ 的持有者,并且限定 $dtt_{11}^{0.7}$ 的持有者信任度不低于 0.7 以及 $dtt_{12}^{0.8}$ 的持有者信任度不低于 0.8 才能激活各自被授予的委托角色.若不采取更严格的限制,则至少不低于 0.6(这是委托源的最低要求).

定义 4(函数 *mapt*()、函数 *up*()、符号 $tv_{uad.u}$).

- 令 $uad\in UAD,dtt\in DTT,mapt(dtt,uad)=\{dtt|dtt.uad=uad\}$,函数返回某委托票据集中作用于特定被委托用户和委托角色对的委托票据集合.
- $up(x)=\begin{cases}T, & x=a^T\\ 0, & x=a\end{cases}$,函数 $up()$返回输入参数的上标值.
- 令 $uad=(u,r)\in UAD$,符号 $tv_{uad.u}$ 表示用户、委托角色对(u,r)中的用户 u 的信任度值.

定义 5(角色委托链、角色委托链集、函数 *acqu_rdc*()、函数 *acqu_uadu*()、函数 *acqu_u*()、函数 *acqu_uxj*()、函数 *acqu_dtt*()).

- 角色委托链定义为委托传播树上任意一条从根到叶的路径,记作 rdc.
- 角色委托链表示为 $rdc=((uad_1,u_0),(uad_2,u_1),\ldots,(uad_n,u_{n-1})),uad_i=(u_i,r_{i\_tree})\in UAD,r_{i-1\_tree}\succeq r_{i\_tree},u_{i-1}\in U$ 是委托者,$1\le i\le n$.
- 在角色委托链 rdc 中,对 $uad_i,u_{i-1},i\in\{k|1<k<n\}$而言,$uad_j,u_{j-1},1\le j<i$ 称为其上游,$uad_j,u_{j-1},i<j\le n$ 称为其下游,并用符号 $\triangleright$ 表示上、下游关系.例如, $a\triangleright b$ 表示 a 是 b 的上游,b 是 a 的下游.
- 在角色委托链 rdc 中,使用符号 $\triangleright|$ 表示邻接关系,例如, $a\triangleright|b$ 表示 a 是 b 的上游且与 b 邻接.
- 同一委托传播树的角色委托链的集合组成角色委托链集,表示为 $RDCS$.
- $acqu\_rdc(dc.spr_{tree})=\{rdc|rdc\in RDCS\},dc\in DCS$,函数返回某一委托传播树的所有角色委托链.$dc.spr_{tree}$ 表示委托凭证 dc 中的委托传播树 spr_{tree}.
- $acqu\_uadu(rdc,(ur,v))=\{(uad,u)\,|\,(ur,v)\triangleright rdc(i).(uad,u),1\le i\le n\}$,$ur\in UAD,v\in U,rdc\in RDCS$,函数返回角色委托链 rdc 中(ur,v)的所有下游$(uad,u).rdc(i).(uad,u)$表示角色委托链 rdc 中第 i 个节点的(uad,u).
- $acqu\_u(dc,v)=\{u\,|\,(\exists rdc\in acqu\_rdc(dc.spr_{tree}))(\forall i\in n)rdc(i).u\triangleright v\}$,$v\in U,dc\in DCS$,函数返回委托凭证 dc 中某角色委托链 rdc 中 v 的所有上游 $u.rdc(i).u$ 表示角色委托链 rdc 中第 i 个节点的 u.
- $acqu\_uxj(dc,v)=\{u\,|\,(\forall rdc\in acqu\_rdc(dc.spr_{tree}))(\exists i\in n)v\triangleright|\,rdc(i).u\}$,$v\in U,dc\in DCS$,函数返回委托凭证 dc 中 v 的所有邻接下游 u.
- $acqu\_dtt(dc.spr_{tree})=\{dtt\,|\,dtt\in\{nodes\_of\_(dc.spr_{tree})\}\}$,$dc\in DCS$,函数返回委托传播树 spr_{tree} 中的所有委托票据 dtt.

定义 6(委托角色激活/去活请求).

- 委托角色激活/去活请求表示为一个二元组$(uad,act_ad)\in DRAD$,其中,$uad\in UAD$ 是用户、委托角色对,$act_ad\in\{activate,deactivate\}$表示请求进行的操作.
- 委托角色激活/去活请求序列表示为 $DRADS$,$DRADS(t)$表示在时间点 t 发生的委托角色激活/去活请求的

集合.

• 用户委托角色激活/去活请求序列表示为 *UDRADS*,其中任意元素 *UDRADS*(*t*)中的委托角色激活/去活请求都来自用户.

定义 7(委托角色授予/撤销请求).

• 委托角色授予/撤销请求表示为一个三元组(*uad*,*opgr*,*act_gr*)∈*DRGR*,其中,*uad*=(*u*,*r*)∈*UAD* 是用户、委托角色对,*opgr*∈*U* 表示进行授予/撤销的操作者,*act_gr*∈{*grant*,*revoke*}表示请求进行的操作.

• 委托角色授予/撤销请求序列表示为 *DRGRS*.*DRGRS*(*t*)表示在时间点 *t* 发生的委托角色授予/撤销请求的集合.

• 用户委托角色授予/撤销请求序列表示为 *UDRGRS*,其中任意元素 *UDRGRS*(*t*)中的委托角色授予/撤销请求都来自用户.

定义 8(委托角色激活状态).

• 委托角色激活状态是二元组(*u*,*r*)的集合,其中,(*u*,*r*)∈*UAD*,表示为 *DRAS*.

• 委托角色激活状态序列表示为 *DRASS*,它的第 *t* 个元素 *DRASS*(*t*)∈*DRAS* 表示在时间点 *t* 的委托角色激活状态.

定义 9(委托角色授予状态).

• 委托角色授予状态是(*uad*,*opgr*)的集合,其中,*uad*=(*u*,*r*)∈*UAD*,*opgr*∈*U*,表示为 *DRGS*.

• 委托角色授予状态序列表示为 *DRGSS*.它的第 *t* 个元素 *DRGSS*(*t*)∈*DRGS* 表示在时间点 *t* 的委托角色授予状态.

定义 10(委托角色激活历史).

• 委托角色激活历史表示为 *DRAH*,它的第 *t* 个元素表示为 *DRAH*(*t*),是二元组(*u*,*r*)的集合,其中, (*u*,*r*)∈*UAD*,表示在时间点 *t* 成功激活委托角色的操作.

定义 11(委托角色授予历史).

• 委托角色授予历史表示为 *DRGH*.它的第 *t* 个元素表示为 *DRGH*(*t*),是(*uad*,*opgr*)的集合,其中,*uad*=(*u*,*r*)∈*UAD*,*opgr*∈*U*,表示在时间点 *t* 成功授予委托角色的操作.

定义 12(函数 ***indrgh***()、函数 ***udrg***()、函数 ***udrgg***()).

• $indrgh(opg,drgh,t)=\begin{cases}1, & opg\in drgh(t).opgr\\ 0, & opg\notin drgh(t).opgr\end{cases}$,*opg*∈*U*,*drgh*∈*DRGH*,*t*≥0,函数根据 *opg* 是否在时间点 *t* 的委托角色授予历史中返回 1 或 0.

• $udrg([begin,end],opg,drgh)=\sum_{t=begin}^{end} indrgh(opg,drgh,t)$,[*begin*,*end*]是一个时间段,*opg*∈*U*,*drgh*∈*DRGH*,函数计算给定时间段[*begin*,*end*]内 *opg* 委托授权的次数.

• $udrgg(pt,opg,drgh)=\sum_{[begin,end]\in\Pi(pt)} udrg([begin,end],opg,drgh)$,*pt*∈*PT*,*opg*∈*U*,*drgh*∈*DRGH*,函数计算给定周期时间 *pt* 中 *opg* 委托授权的次数.

周期时间 *pt*∈*PT* 的定义见文献[13],函数 $\Pi()$的定义见文献[16].使用次数统计函数 *indrgh*(),*udrg*(), *udrgg*()会根据委托角色授予历史序列计算出指定周期时间或时间段内某委托者的委托授权次数,此值不应超过 n_b.

定义 13(函数 ***indepth***()、函数 ***depth***()).

• $indepth(pt,opg,drgh)=\begin{cases}1, & udrgg(pt,opg,drgh)\neq 0\\ 0, & udrgg(pt,opg,drgh)=0\end{cases}$,*pt*∈*PT*,*opg*∈*U*,*drgh*∈*DRGH*,函数根据给定周期时间 *pt* 中用户 *opg* 是否做过委托授权而返回 1 或 0.

• $depth(pt,drgh,fatnd)=\sum_{opg\in fatnd} indepth(pt,drgh,fatnd)$,*pt*∈*PT*,*opg*∈*U*,*drgh*∈*DRGH*,*fatnd*=*acqu_u*(*dc*,*udr*.*opgr*),*dc*∈*DCS*,*udr*=(*uad*,*opgr*,*grant*)∈*UDRGRS*(*t*),函数计算给定周期时间 *pt* 中委托凭证 *dc* 中用户 *opgr* 的做过委托授

权的上游用户数.

使用次数统计函数 *indepth*(),*depth*()会根据委托角色授予历史序列计算出指定周期时间或时间段内委托权限的传播次数,此值不应超过 n_d.

定义 14(符号 $\veebar$、符号 $\wedge$、函数 ***gsetnum***()、函数 ***qinch***()、函数 ***dinch***()).

- 定义 $a \veebar b$ 为:若 *a* 的执行结果为真则不执行 *b*,否则执行 *b*.
- 定义 $a \wedge b$ 为:若 *a* 的执行结果为假则不执行 *b*,否则执行 *b*.
- *gsetnum*(*set*)返回集合 *set* 中的元素个数.
- 令 $udr=(uad,opgr,grant)\in UDRGRS(t),dc\in DCS$,

$$qinch(udr,dc)=\begin{cases}\text{true}, & (\exists rdc\in acqu\_rdc(dc.spr_{tree}))(\exists i\in N)(rdc(i).uad.u=\\ & \quad udr.uad.u\wedge rdc(i).u=udr.opgr\wedge rdc(i).uad.r\succeq udr.uad.r)\\ \text{false}, & \text{else}\end{cases}.$$

若函数返回真,则表明委托授权请求 *udr* 满足委托凭证 *dc* 的部分委托限制和委托传播限制;若函数返回假,则需要另作判断.

- 令 $udr=(uad,opgr,grant)\in UDRGRS(t),dc\in DCS$,

$$dinch(udr,dc)=\begin{cases}\text{true}, & (\exists rdc\in acqu\_rdc(dc.spr_{tree}))(\exists i\in N)(rdc(i).uad.u=\\ & \quad udr.opgr\wedge rdc(i).uad.r\succeq udr.uad.r)\\ \text{false}, & \text{else}\end{cases}.$$

若函数返回真,则表明委托授权请求 *udr* 请求的委托者在委托凭证 *dc* 的某个角色委托链上并且请求的权限不超过委托者所拥有的权限;否则函数返回假.

2.2 模型语义

在模型语义描述中,我们将文献[13]的假设 1 扩展为假设 Extension_1.

Extension_1:同一时间点只处理冲突角色请求中的去活角色请求,且常规角色请求优先于委托角色请求;同一时间点也只处理冲突委托角色授予/撤销请求中的委托角色撤销请求;若前后两个时间点相同,则去活角色请求优先于委托角色撤销请求.

在 CRDM[13]模型语义定义中,整个动态过程是通过一个快照的序列来表现的,每个快照是某个时间点的状态,并用四元组(*srres*,*orass*,*rass*,*rah*)表示,称作系统历史,我们将它扩展为系统状态空间.

定义 15(系统状态空间).

- 系统状态空间表示为 $SysStatusSpace\subseteq(DRADS\cup DRGRS)\times DRASS\times DRAH\times DRGSS\times DRGH$.
- 对于一个委托凭证序列 $dcs\in DCS$、一个用户委托角色授予/撤销请求序列 $udrgrs\in UDRGRS$、一个用户委托角色激活/去活请求序列 $udrads\in UDRADS$,它们的系统状态空间的状态元素表示为一个五元组($sdrrs,drass,drah,drgss,drgh$),其中 $sdrrs\in DRADS\cup DRGRS$ 是系统委托角色激活/去活请求序列或系统委托角色授予/撤销请求序列,$drass\in DRASS$ 是委托角色激活状态序列,$drah\in DRAH$ 是委托角色激活历史,$drgss\in DRGSS$ 是委托角色授予状态序列,$drgh\in DRGH$ 是委托角色授予历史,其中在任意时间点,$t\geq 0$,$drass(t)\subseteq UAD$,$drgss(t)\subseteq DRGS$.

定义 16(委托授权执行规则). 系统状态空间的一个状态($sdrrs,drass,drah,drgss,drgh$)是 $dcs\in DCS,udrgrs\in UDRGRS,udrads\in UDRADS$ 确定的一个执行状态,当且仅当初始状态下 $sdrrs(0),drass(0),drah(0),drgss(0),drgh(0)$的任意元素均为空并且对任意 $t>0$ 必须满足如下委托授权执行规则 1~规则 12.

规则 1. 在时间点 $t-1$ 满足委托凭证中某委托票据时间限制,而在 *t* 不满足限制的委托角色授予状态将导致产生在时间点 *t* 的系统委托角色撤销请求,形式化表示为

$$\begin{pmatrix}\forall ur:UAD\\ \forall opgr:U\end{pmatrix}\bullet\begin{pmatrix}\exists dc:DC\bullet dc\in dcs,\exists dtt:DTT\bullet dtt\in acqu\_dtt(dc.spr_{tree})\wedge ur=dtt.uad\\ \wedge(ur,opgr)\in drgss(t-1)\wedge t-1\in Sol(pt)\wedge t\notin Sol(pt)\end{pmatrix}\rightarrow$$
$$(ur,opgr,revoke)\in sdrrs(t).$$

规则 2. 用户对委托角色的授予请求会生成相应的系统委托角色授予请求,形式化表示为

$$\forall(ur,opgr,grant):DRGR\bullet\begin{pmatrix}(ur,opgr,grant)\in udrgrs(t)\wedge\\(ur,opgr)\notin drgss(t-1)\wedge t\in Sol(pt)\end{pmatrix}\rightarrow(ur,opgr,grant)\in sdrrs(t).$$

规则 3. 用户对委托角色的撤销请求不仅会导致产生相应的系统委托角色撤销请求,还可能会导致角色委托链集中,此用户所处的所有链中的该用户的下游用户产生相应的系统委托角色撤销请求,形式化表示为

$$\begin{pmatrix}\forall(ur,opgr,revoke):DRGR\bullet\begin{pmatrix}(ur,opgr,revoke)\in udrgrs(t)\wedge\\(ur,opgr)\in drgss(t-1)\wedge t\in Sol(pt)\end{pmatrix}\rightarrow(ur,opgr,revoke)\in sdrrs(t)\end{pmatrix}\wedge$$
$$\begin{pmatrix}\begin{pmatrix}\forall ur:UAD\\\forall opgr:U\end{pmatrix}\bullet\begin{pmatrix}\exists dc:DC\bullet dc\in dcs,\forall rdc:RDC\bullet rdc\in acqu\_rdc(dc.spr_{tree}),(\forall ur':UAD,\forall opgr':U)\bullet\\(ur',opgr')\in acqu\_uadu(rdc,(ur,opgr))\wedge(ur',opgr')\in drgss(t-1)\wedge t\in Sol(pt)\end{pmatrix}\rightarrow\\(ur',opgr',revoke)\in sdrrs(t).\end{pmatrix}.$$

规则 4. 系统委托角色授予请求会为未授予某角色且满足委托凭证限制的用户指派此角色,表示为

$$\forall udr=(ur,opgr,grant):DRGR\bullet$$
$$\begin{pmatrix}(ur,opgr,grant)\in sdrrs(t)\wedge ur\in CAN\_UAD\wedge(ur,opgr,revoke)\notin sdrrs(t)\wedge\\\begin{pmatrix}\begin{pmatrix}\forall dc:DC\bullet dc\in dcs,\\\forall dtt:DTT\bullet dtt\in acqu\_dtt(dc.spr_{tree})\end{pmatrix}\wedge\begin{pmatrix}mapt(acqu\_dtt(dc.spr_{tree}),ur)=\phi\vee\\\begin{pmatrix}dtt\in mapt(acqu\_dtt(dc.spr_{tree}),ur)\wedge t\in Sol(pt)\wedge\\(\exists ur':UA,\exists opgr':U)\bullet(ur',opgr')\in drgss(t-1)\wedge\\(ur'.u=opgr)\wedge(ur'.r\geq ur.r)\wedge(ur,opgr)\notin drgss(t-1)\\\wedge grant(dtt.degt)\subseteq drgss(t).uad\wedge\\(\forall x:UA\bullet x\in grant(dtt.degt)\rightarrow tv_{x.u}\geq up(x))\wedge\\(\forall y:UA\bullet y\in nogrant(dtt.degt)\rightarrow y\notin drgss(t).uad)\\\wedge(qinch(udr,dc)\underline{\vee}(dinch(udr,dc)\underline{\wedge}(dtt.ae=all\\\wedge depth(dtt.pt,drgh,acqu\_u(dc,opgr))<dc.n_d)\\\underline{\wedge}(dtt.ae=all\wedge udrgg(dtt.pt,opgr,drgh)<dc.n_b)\\\underline{\wedge}(gsetnum(acqu\_uxj(dc,opgr))<dc.n_b)))\end{pmatrix}\end{pmatrix}\rightarrow\\ur\in UAD,(ur,opgr)\in drgss(t),(ur,opgr)\in drgh(t).\end{pmatrix}$$

规则 5. 系统委托角色撤销请求会将指定的非激活状态的委托角色撤销,形式化表示为

$$\forall(ur,opgr,revoke):DRGR\bullet\begin{pmatrix}(ur,opgr,revoke)\in sdrrs(t)\wedge ur\notin drass(t-1)\\(ur,opgr)\in drgss(t-1)\wedge t\in Sol(pt)\end{pmatrix}\rightarrow ur\notin UAD,(ur,opgr)\notin drgss(t).$$

规则 6. 如果没有显式地授予或撤销系统请求,*drgss* 不会自动发生变化,形式化表示为

$$\begin{pmatrix}\forall ur:CAN\_UAD\\\forall opgr:U\end{pmatrix}\bullet\begin{pmatrix}(ur,opgr)\in drgss(t-1)\wedge((ur,opgr),revoke)\notin sdrrs(t)\rightarrow(ur,opgr)\in drgss(t)\\(ur,opgr)\notin drgss(t-1)\wedge((ur,opgr),grant)\notin sdrrs(t)\rightarrow(ur,opgr)\notin drgss(t)\end{pmatrix}.$$

规则 7. 用户成功激活委托角色必须满足委托凭证的限制,表示为

$$\forall(ur,activate):DRAD\bullet$$
$$\left(\begin{array}{c}(ur,activate)\in sdrrs(t)\wedge ur\in UAD\wedge(ur,deactivate)\notin sdrrs(t)\wedge\\ \left(\begin{array}{c}\forall dc:DC\bullet dc\in dcs,\\ \forall dtt:DTT\bullet dtt\in acqu\_dtt(dc.spr_{tree})\end{array}\right)\wedge\left(\begin{array}{c}mapt(acqu\_dtt(dc.spr_{tree}),ur)=\phi\vee\\ \left(\begin{array}{c}dtt\in mapt(acqu\_dtt(dc.spr_{tree}),ur)\wedge\\ t\in Sol(pt)\wedge active(dtt.dept)\subseteq drass(t)\wedge\\ (\forall x:UAD\bullet x\in active(dtt.dept)\rightarrow\\ tv_{x.u}\geq up(x))\wedge(\forall y:UAD\bullet\\ y\in inactive(dtt.dept)\rightarrow y\notin drass(t))\\ \wedge(tv_{ur.u}\geq up(dtt))\wedge(dtt.uad.r\succeq ur.r)\\ \wedge((dtt.ae=all\wedge uraa(dtt.pt,ur,drah)<dtt.n)\vee\\ (dtt.ae=each\wedge ura(Pti(t,dtt.pt),ur,drah)<dtt.n))\end{array}\right)\end{array}\right)\end{array}\right)$$
$$\rightarrow ur\in drass(t),ur\in drah(t).$$

函数 *active*(),*inactive*(),*uraa*(),*ura*(),*Pti*()的定义参见文献[13],函数 *Sol*()的定义参见文献[16].

我们对文献[13]的规则 1、规则 4、规则 5、规则 10、规则 11 使用六元组委托票据 *dtt* 进行功能扩展,并使用本文定义符号进行改写.这些规则与上述 7 个规则构成完整的委托授权执行规则.

规则 8. 在时间点 *t*–1 满足委托凭证中某委托票据时间限制而在 *t* 不满足限制的委托角色激活状态将导致产生在时间点 *t* 的系统委托角色去活请求,形式化表示为

$$\forall ur:UAD\bullet\left(\begin{array}{c}\exists dc:DC\bullet dc\in dcs,\exists dtt:DTT\bullet dtt\in acqu\_dtt(dc.spr_{tree})\wedge ur=dtt.uad\\ \wedge ur\in drass(t-1)\wedge t-1\in Sol(pt)\wedge t\notin Sol(pt)\end{array}\right)\rightarrow(ur,deactivate)\in sdrrs(t).$$

规则 9. 用户对委托角色的激活请求会生成相应的系统委托角色激活请求,形式化表示为

$$\forall(ur,activate):DRAD\bullet\left(\begin{array}{c}(ur,activate)\in udrads(t)\wedge ur\in UAD\wedge\\ ur\notin drass(t-1)\wedge t\in Sol(pt)\end{array}\right)\rightarrow(ur,activate)\in sdrrs(t).$$

规则 10. 用户对委托角色的去活请求会生成相应的系统委托角色去活请求,形式化表示为

$$\forall(ur,deactivate):DRAD\bullet\left(\begin{array}{c}(ur,deactivate)\in udrads(t)\wedge ur\in UAD\wedge\\ ur\in drass(t-1)\wedge t\in Sol(pt)\end{array}\right)\rightarrow(ur,deactivate)\in sdrrs(t).$$

规则 11. 系统委托角色去活请求会导致委托角色状态的相应变化,形式化表示为

$$\forall(ur,deactivate)\left(\begin{array}{c}(ur,deactivate)\in sdrrs(t)\wedge\\ ur\in UAD\end{array}\right)\rightarrow ur\notin drass(t).$$

规则 12. 如果没有显式的激活或去活的系统委托角色请求,*drass* 不会自动发生变化,形式化表示为

$$\forall ur:UAD\bullet\left(\begin{array}{c}ur\in drass(t-1)\wedge(ur,deactivate)\notin sdrrs(t)\rightarrow ur\in drass(t)\\ ur\notin drass(t-1)\wedge(ur,activate)\notin sdrrs(t)\rightarrow ur\notin drass(t)\end{array}\right).$$

定理 1. 系统状态空间的任意状态(*sdrrs*,*drass*,*drah*,*drgss*,*drgh*)始终满足其相应的委托凭证序列 *dcs*.

证明:在 *t*=0 时,各个元素均为空,所以一定满足 *dcs*.假定在 *t*=*k* 时,仍然满足 *dcs*.可能导致不满足 *dcs* 的情况有:① 用户委托角色激活请求不满足临时性约束;② 用户委托角色激活请求不满足委托角色激活依赖 *dept*;③ 请求激活委托角色的用户信任度低于激活此角色所需的信任度阈值;④ *dcs* 的委托票据中适用于角色激活状态的时间段过期;⑤ 用户委托角色授予请求不满足委托角色授予依赖 *degt*;⑥ 用户委托角色授予请求不满足角色委托链的邻接关系依赖,即邻接关系的上游未授权;⑦ 用户委托角色授予请求不满足 *dcs* 的部分委托限制或委托传播限制;⑧ 用户委托角色撤销请求不满足委托角色依赖,即此委托角色处于激活状态;⑨ *dcs* 的委托票据中适用于委托角色授予状态的时间段过期.

(1) 在情况①~情况③中,根据规则 9,在时间点 *k*+1 的用户委托角色激活请求会进入 *sdrrs*(*k*+1)中,在处理过程中,它们不满足规则 7,所以请求会被拒绝.

(2) 情况④中,在时间点 *k* 上,*ur*∈*drass*(*k*),而时间点 *k*+1 不在它对应的委托凭证中某委托票据使用的时间段

内.这时根据规则 8,系统将自动生成一个去活 *ur* 的请求,根据规则 11,*ur* 不会出现在 *drass*(*t*+1)中,并且根据规则 7,若同时有一个激活的 *ur* 请求,则该激活请求将被忽略.

(3) 在情况⑤~情况⑦中,根据规则 2,在时间点 *k*+1 的用户委托角色授予请求会导致生成在时间点 *k* 未授权的该用户、委托角色对的系统委托角色授予请求(加入 *sdrrs*(*t*+1)).根据规则 4,上述用户、委托角色对不会出现在 *UAD*,*drgss*(*k*+1)和 *drgh*(*t*+1)中,而根据规则 7,上述用户、委托角色对也不会出现在 *drass*(*t*+1)和 *drah*(*t*+1)中.

(4) 在情况⑧中,根据规则 3,在时间点 *k*+1 的用户委托角色撤销请求会导致生成在时间点 *k* 已授权的该用户、委托角色对的系统委托角色撤销请求(加入 *sdrrs*(*t*+1)),同时还会导致角色委托链集中此用户、委托角色对所处的所有链中的该角色对的下游产生相应的系统委托角色撤销请求.而根据规则 5,上述用户、委托角色对不会从 *UAD*,*drgss*(*k*+1)中撤销.

(5) 情况⑨中,在时间点 *k* 上,(*ur*,*opgr*)∈*drgss*(*k*),而时间点 *k*+1 不在它对应的委托凭证中某委托票据使用的时间段内.这时根据规则 1,系统将自动生成一个撤销(*ur*,*opgr*)的请求,根据规则 5,(*ur*,*opgr*)不会出现在 *drgss*(*k*+1)中,并且根据规则 4,若同时有一个(*ur*,*opgr*,*grant*)请求,则该请求将被忽略.

(6) 根据规则 6 和规则 12,只有 *sdrrs*(*k*+1)会导致 *drgss*(*k*+1)或 *drass*(*k*+1)发生变化.

综上所述,系统状态空间的任意状态在时间点 *k*+1 满足 *dcs*.得证. □

定理 2. 给定委托凭证序列、用户委托角色授予/撤销请求序列和用户角色激活/去活请求序列,则它们有唯一的系统状态.

根据定义 16 和假设 Extension_1 容易证明,故略去细节.

2.3 模型实现

本文模型对应于图 1 中的 VPEP,它的构成细节示例如图 4 中的虚线框所示.

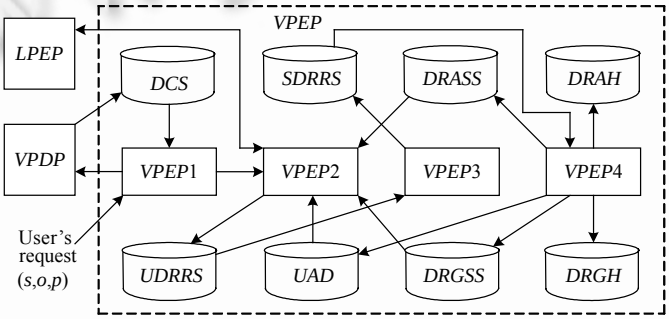


Fig.4 An example of VPEP structure detail

图 4 VPEP 构成细节示例

*VPEP*1 负责截获访问请求(*s*,*o*,*p*)(*s*,*o*,*p* 分别表示为主体、客体、权限,在本文中分别代表网格用户、网格服务、网格服务的操作),然后查询凭证库 *DCS* 中是否存在对应的处于可使用状态的委托凭证 *dc*.若存在,则将(*s*,*o*,*p*)与 *dc* 绑定;若无,则将(*s*,*o*,*p*)递交 *VPDP* 进行授权,决策产生相应的处于可使用状态的 *dc* 再绑定.若存在但暂时不可用,则等待 *VPDP* 根据系统状态变化进行再决策,改变 *dc* 为可使用状态时再绑定.若成功绑定一对((*s*,*o*,*p*),*dc*),则 *VPEP*1 启动一个 *VPEP*2 进程处理它,而自己继续监听是否有下一个请求到来.若有,则重复上述过程.*VPEP*1 可采用定时器来为每个(*s*,*o*,*p*)的处理设置超时间隔,若超时,则放弃处理.

*VPEP*2 根据((*s*,*o*,*p*),*dc*)产生相应的用户、委托角色对(*s*,*r*)(其中委托角色 *r* 是依据 *dc* 确定的,它包含使用操作 *p* 的权限),然后查询(*s*,*r*)是否出现在 *UAD* 中.若无,则产生相应的用户委托角色授予请求((*s*,*r*),*VO*,*grant*)并置入 *UDRRS*(包括 *UDRGRS* 和 *UDRADS*)中.再查询((*s*,*r*),*VO*)是否出现在 *DRGSS* 中.若在,则产生一个用户委托角色激活请求((*s*,*r*),*activate*)并置入 *UDRRS* 中.继续查询(*s*,*r*)是否出现在 *DRASS* 中.若在,则将((*s*,*o*,*p*),*dc*)发送到资源所在自治域的 *LPEP* 并等待返回访问结果.若在 *UAD* 中查询到(*s*,*r*),则不需要产生((*s*,*r*),*VO*,*grant*)而直接产生

((*s*,*r*),*activate*).通常,若有访问结果返回,则 *VPEP*2 产生用户委托角色去活请求((*s*,*r*),*deactivate*)并置入 *UDRRS* 中.至于是否产生用户委托角色撤销请求((*s*,*r*),*VO*,*revoke*)并置入 *UDRRS* 中,要根据应用的特点和系统资源的情况依据事先制定的策略进行处理.无论是否产生用户委托角色撤销请求,都会结束 *VPEP*2 进程.为避免过多 *VPEP*2 进程占用系统资源,可采用定时器为查询操作和等待过程设置超时间隔.若超时仍未在 *DRGSS* 中查询到((*s*,*r*),*VO*),则结束 *VPEP*2 进程.若超时仍未在 *DRASS* 中查询到(*s*,*r*),则 *VPEP*2 进程在产生用户委托角色撤销请求后结束.若在等待访问结果时超时,则 *VPEP*2 进程在产生用户委托角色去活请求和撤销请求后结束. *UDRRS* 中的元素都带有产生时间标识.由于为提高处理(*s*,*o*,*p*)的效率而使 *VPEP*2 组件多进程运行,因此在时间轴上,不同(*s*,*o*,*p*)的用户委托角色请求可能存在交错排列.

*VPEP*3 按产生时间顺序处理 *UDRRS* 中的用户委托角色请求,依据委托授权执行规则 2、规则 3、规则 9、规则 10,将生成的系统委托角色请求置入 *SDRRS* 中,并且定期按照委托授权执行规则 1 和规则 8 检查委托角色授予或激活状态是否符合委托凭证时间限制,若不符合,则产生系统委托角色撤销或去活请求并置入 *SDRRS* 中.

*VPEP*4 从 *SDRRS* 中逐一取出系统委托角色请求(按其产生的时间顺序)进行处理.若是授予请求,则按规则 4 处理,若成功,则将(*s*,*r*)置入 *UAD* 中,将((*s*,*r*),*VO*)分别置入 *DRGSS* 和 *DRGH* 中.若是撤销请求,则按规则 5 处理,若成功,则将(*s*,*r*)从 *UAD* 中删除,将((*s*,*r*),*VO*)从 *DRGSS* 中删除.若是激活请求,则按规则 7 处理,若成功,则将(*s*,*r*)分别置入 *DRASS* 和 *DRAH* 中.若是去活请求,则按规则 11 处理,将 *DRASS* 中的(*s*,*r*)删除.若 *SDRRS* 为空,则根据规则 6 和规则 12,*DRGSS* 和 *DRASS* 不会发生变化.在处理过程中,若遇到因暂时条件不满足而不能成功处理的系统委托角色请求,则暂时跳过以便及时处理后面的,而后面的成功处理可能会使得前面未处理的请求条件得到满足.因此,*VPEP*4 需要依据一定策略再次处理以前没有成功处理的请求.

2.4 与相关模型的比较

文献[9]的模型要求 *VO* 具有生成 *VO* 角色以及继承和分解委托角色的功能,而我们通过要求自治域采用基于层次角色的访问控制系统以便在 *VO* 中省略这些功能.基于文献[9]相同的示例,我们给出了统计局的一个与工业统计工作有关的层次角色关系(广义表表示的树形结构)如:$r_{DH\text{-}tree}=(r_{DH}(r_{PS}(r_1,r_2),r_{EI}(r_3,r_4),r_{EPI}(r_5,r_6)))$.其中,最底层的 r_1~r_6 是只具有单一权限的原子角色,具体含义见表 1.统计局将业务服务 BS_{PS},BS_{EI} 和 BS_{EPI} 共享给虚拟空间 VS_{EI}.为防止造假,统计局权限系统将这 3 个业务服务的操作权限分别分配给 3 个角色 r_{PS},r_{EI} 和 r_{EPI},并设定它们必须互斥.统计局向 VS_{EI} 发布的使用业务服务 BS_{PS},BS_{EI} 和 BS_{EPI} 的委托凭证,见表 2.

Table 1 Permission assignment table

表 1 权限分配表

Atom role	Permission	Operation of business service
r_1	PW_{PS} (writing report forms of production and vendition)	Writing operation of BS_{PS}
r_2	PR_{PS} (reading report forms of production and vendition)	Reading operation of BS_{PS}
r_3	PW_{EI} (writing report forms of economic index)	Writing operation of BS_{EI}
r_4	PR_{EI} (reading report forms of economic index)	Reading operation of BS_{EI}
r_5	PW_{EPI} (writing report forms of economic plan index)	Writing operation of BS_{EPI}
r_6	PR_{EPI} (reading report forms of economic plan index)	Reading operation of BS_{EPI}

Table 2 Using permissions of business services delegated to VS_{EI}

表 2 委托给 VS_{EI} 的业务服务的使用权限

Business	Delegation certification	Delegation ticket
BS_{PS}	$dc_{PS}=(n_d,n_b,(dtt_{VO\text{-}PS}))$	$dtt_{VO\text{-}PS}=((VS_{EI},r_{PS\text{-}tree}),pt,n,ae,\{\neg(VS_{EI},r_{EI\text{-}tree}),\neg(VS_{EI},r_{EPI\text{-}tree})\},\varnothing)$
BS_{EI}	$dc_{EI}=(n_d,n_b,(dtt_{VO\text{-}EI}))$	$dtt_{VO\text{-}EI}=((VS_{EI},r_{EI\text{-}tree}),pt,n,ae,\{\neg(VS_{EI},r_{PS\text{-}tree}),\neg(VS_{EI},r_{EPI\text{-}tree})\},\varnothing)$
BS_{EPI}	$dc_{EPI}=(n_d,n_b,(dtt_{VO\text{-}EPI}))$	$dtt_{VO\text{-}EPI}=((VS_{EI},r_{EPI\text{-}tree}),pt,n,ae,\{\neg(VS_{EI},r_{EI\text{-}tree}),\neg(VS_{EI},r_{PS\text{-}tree})\},\varnothing)$

发改局用户 *F* 需要拥有读 BS_{EPI} 的权限.VS_{EI} 只需依据 dc_{EPI},VS_{EI} 授权策略和用户 *F* 属性进行授权决策,输出委托凭证 $dc_{F\text{-}EPI}=(n_d,n_b,(dtt_{VO\text{-}EPI}(dtt_{F\text{-}EPI})))$.其中,*F* 的委托票据为 $dtt_{F\text{-}EPI}=((F,r'_{EPI\text{-}tree}),pt,n,ae,\{\neg(F,r'_{EI\text{-}tree}),\neg(F,r'_{PS\text{-}tree})\},\varnothing)$.其中,角色树 $r'_{EPI\text{-}tree}$ 为$(r_{EPI}(r_6))$,而表 2 的角色树 $r_{EPI\text{-}tree}$ 则为$(r_{EPI}(r_5,r_6))$.VS_{EI} 授予 *F* 的权限为 VS_{EI} 获得的相应权限的子集,因此不会违背自治域安全策略.VS_{EI} 的 *VPEP* 利用委托授权执行规则,依据委托凭

证对用户的资源访问请求进行验证,不符合委托凭证要求的访问请求在 *VO* 层次都会被过滤掉.因此,与文献[9]模型相比,本文模型能够减轻自治域的验证负担.

在安全保障方面,若用户 *F* 同时请求读 BS_{EPI} 和 BS_{EI} 的权限(这会违背 r_{EPI} 和 r_{EI} 必须互斥的自治域安全策略),则 VS_{EI} 的 *VPDP* 分别产生委托凭证 $dc_{F\text{-}EPI}=(n_d,n_b,(dtt_{VO\text{-}EPI}(dtt_{F\text{-}EPI})))$和 $dc_{F\text{-}EI}=(n_d,n_b,(dtt_{VO\text{-}EI}(dtt_{F\text{-}EI})))$,其中,$dtt_{F\text{-}EI}=((F, r'_{EI\text{-}tree}),pt,n,ae,\{\neg(F, r'_{EPI\text{-}tree}),\neg(F, r'_{PS\text{-}tree})\},\varnothing)$.若 VS_{EI} 的 *VPEP* 先执行读 BS_{EPI},则$(F, r'_{EPI\text{-}tree})$会出现在 *DRASS* 中.这将使得随后 *VPEP* 执行读 BS_{EI} 的请求时,会因委托票据 $dtt_{F\text{-}EI}$ 的依赖关系$\neg(F, r'_{EPI\text{-}tree})$得不到满足,而将此访问请求过滤掉,从而在 *VO* 层次维护了自治域安全策略.而文献[9]模型需要自治域来将这种情况检测出来.

3 应用举例

构建一个“软件培训”虚拟组织 *VST* 需要聚集广域分布的闲散资源——课件.我们将课件封装成网格服务,它对外提供 4 个基本操作:读、下载、写、上传.这些操作的使用权限分别表示为 *R*,*D*,*W*,*U*,并分别被指派给 4 个基本角色:r_R,r_D,r_W,r_U.自治域 *AD*1 愿意共享基础课服务,而自治域 *AD*2 愿意共享专业课服务.表 3 列出了一些服务与其分配的角色.*AD*1 和 *AD*2 的角色层次关系片断(广义表表示的树形结构)分别表示为 $r_{MT\text{-}tree}=(r_{MT}(r_M(r_R,r_D,r_W,r_U),r_S(r_R,r_D,r_W,r_U),r_C(r_R,r_D,r_W,r_U)))$和 $r_{ST\text{-}tree}=(r_{ST}(r_E(r_R,r_D,r_W,r_U),r_T(r_R,r_D,r_W,r_U),r_O(r_R,r_D,r_W,r_U)))$.

VST 根据自己的目标与 *AD*1 协商获得“应用数学 *M*”和“应用统计 *S*”的“读”和“下载”权,有效期从 7 月 1 日~8 月 31 日.设定 $n_d^T=1^{0.6}$, $n_b^T=30^{0.6}$ 以限制 *VST* 只能同时委托授权给 30 位用户,并且用户的信任度必须不低于 *T*=0.6 才能激活委托角色,其委托凭证为 $dc_{VST\text{-}AD1}=(1^{0.6},30^{0.6},dtt_{VST-AD1}^{0.6})$.其中,委托票据为 $dtt_{VST\text{-}AD1}=((VST, r'_{MT\text{-}tree}),pt,n,ae,dept,degt), r'_{MT\text{-}tree}=(r_{MT}(r_M(r_R,r_D),r_S(r_R,r_D))),pt=([07\text{-}01,0\text{-}31],P),pt$ 的详细解释见文献[13]. $r'_{MT\text{-}tree}$ 表示由完整的角色树 $r_{MT\text{-}tree}$ 剪枝得到的子角色树.委托票据的 *dept* 和 *degt* 都为空,表示没有关联性限制.其他未解释的参数在本例中对授权无限制,下同.*VST* 根据自己的目标与 *AD*2 协商获得“现代软件工程 *E*”、“软件度量与测试 *T*”和“面向对象与构件 *O*”的“读”和“下载”权,其委托凭证为 $dc_{VST\text{-}AD2}=(1^{0.6},30^{0.6},dtt_{VST\text{-}AD2}^{0.6})$.其中,委托票据为 $dtt_{VST\text{-}AD2}=((VST, r'_{ST\text{-}tree}),pt,n,ae,dept,degt), r'_{ST\text{-}tree}=(r_{ST}(r_E(r_R,r_D),r_T(r_R,r_D),r_O(r_R,r_D)))$. $r'_{ST\text{-}tree}$ 的解释类似 $r'_{MT\text{-}tree}$.其他未解释的参数同前.

Table 3 Assignment table of courseware services and their roles

表 3 课件服务和角色分配表

Basic lesson service	Role	Speciality lesson service	Role
Application maths	r_M	Modern sofe engineering	r_E
Application statistic	r_S	Software measurement and test	r_T
Combinatorics	r_C	Object oriented and component	r_O

VST 主要有两种用户类别:教师(*te*)和学生(*st*).我们约定用符号 $User_{class}$ 表示某类别的特定用户,*User* 表示用户名,下标 *class* 表示用户类别.例如,Li_{st} 代表名字为 *Li* 的学生用户,Any_{te} 代表任意教师用户.在 *VST* 的授权策略中有如下规定:① 必须有一定信任度的教师任教某门课,学生才能选修;② 必须有一定信任度的教师在线指导,学生才能在线学习基础课;③ 任意教师类用户都不能既任教基础课又任教专业课.为使系统有限的委托角色授予资源可以被充分利用以服务更多的用户,*VST* 系统规定,对申请任教某门课的教师用户保留任教资格一周,而对申请学习某门课的学生用户只保留学习资格一天.超过上述期限的用户都要重新被授予相应角色以获得任教或学习资格.

在 7 月 1 日,*VST* 系统被初始化后,学生 Li_{st} 发出浏览应用数学 *M* 课件的请求(Li_{st},*M*,*R*);然后,教师 $Chen_{te}$ 发出浏览应用数学 *M* 课件的请求($Chen_{te}$,*M*,*R*);在 7 月 2 日,教师 $Chen_{te}$ 发出浏览应用数学 *M* 课件的请求($Chen_{te}$,*M*,*R*);然后学生 Li_{st} 发出浏览应用数学 *M* 课件的请求(Li_{st},*M*,*R*);在 7 月 3 日,学生 Sun_{st} 发出浏览现代软件工程 *E* 课件的请求(Sun_{st},*E*,*R*);然后教师 $Chen_{te}$ 发出浏览现代软件工程 *E* 课件的请求($Chen_{te}$,*E*,*R*).

*VPEP*1 截获用户的首次浏览请求并递交 *VPDP* 进行授权决策而产生的委托凭证,见表 4.由 *VPEP*2 根据

((*s*,*o*,*p*),*dc*)产生的用户委托角色请求序列见表 5.在 7 月 1 日~7 月 3 日期间,Li_{st},$Chen_{te}$,Sun_{st}的信任度变化情况见表 6.由 *VPEP*3 和 *VPEP*4 依据委托凭证序列、用户委托角色请求序列和规则 1~规则 12 产生的系统状态空间见表 7.

Table 4 Delegation certifications and delegation tickets

表 4 委托凭证和委托票据

Delegation certification	Delegation ticket
$dc_{(Li,M)}=(1^{0.6},30^{0.6},dtt^{0.6}_{VST\text{-}AD1}dtt^{0.7}_{Li\text{-}VST})$	$dtt_{Li\text{-}VST}=((Li_{st},r''_{MT\text{-}tree}),pt,n,ae,\{(Any_{te},r''_{MT\text{-}tree})^{0.8}\},\{(Any_{te},r''_{MT\text{-}tree})^{0.85},\neg(Any_{te},r_{ST\text{-}tree})\})$; $r''_{MT\text{-}tree}=(r_{MT}(r_M(r_R)))$
$dc_{(Chen,M)}=(1^{0.6},30^{0.6},(dtt^{0.6}_{VST\text{-}AD1}dtt^{0.8}_{Chen\text{-}VST})$	$dtt_{Chen\text{-}VST}=((Chen_{te},r''_{MT\text{-}tree}),pt,n,ae,\varnothing,\{\neg(Chen_{te},r_{ST\text{-}tree})\})$; $r''_{MT\text{-}tree}=(r_{MT}(r_M(r_R)))$
$dc_{(Sun,E)}=(1^{0.6},30^{0.6},(dtt^{0.6}_{VST\text{-}AD2}dtt^{0.7}_{Sun\text{-}VST})$	$dtt_{Sun\text{-}VST}=((Sun_{st},r''_{ST\text{-}tree}),pt,n,ae,\varnothing,\{(Any_{te},r''_{ST\text{-}tree})^{0.85},\neg(Any_{te},r_{MT\text{-}tree})\})$; $r''_{ST\text{-}tree}=(r_{ST}(r_E(r_R)))$
$dc_{(Chen,E)}=(1^{0.6},30^{0.6},(dtt^{0.6}_{VST\text{-}AD2}dtt^{0.8}_{Chen\text{-}VST})$	$dtt_{Chen\text{-}VST}=((Chen_{te},r''_{ST\text{-}tree}),pt,n,ae,\varnothing,\{\neg(Chen_{te},r_{MT\text{-}tree})\})$; $r''_{ST\text{-}tree}=(r_{ST}(r_E(r_R)))$

Table 5 User delegation role requests

表 5 用户委托角色请求

Time	User request sequence (*UDRGRS* or *UDRADS*)
07-01AM	$((Li_{st},r''_{MT\text{-}tree}),VST,grant),((Chen_{te},r''_{MT\text{-}tree}),VST,grant),((Chen_{te},r''_{MT\text{-}tree}),activate)$
07-01PM	$((Chen_{te},r''_{MT\text{-}tree}),deactivate)$
07-02AM	$((Chen_{te},r''_{MT\text{-}tree}),activate),((Li_{st},r''_{MT\text{-}tree}),VST,grant),((Li_{st},r''_{MT\text{-}tree}),activate)$
07-02PM	$((Li_{st},r''_{MT\text{-}tree}),deactivate),((Li_{st},r''_{MT\text{-}tree}),VST,revoke),((Chen_{te},r''_{MT\text{-}tree}),deactivate)$
07-03AM	$((Sun_{st},r''_{ST\text{-}tree}),VST,grant),((Chen_{te},r''_{ST\text{-}tree}),VST,grant)$

Table 6 User's trustworthiness

表 6 用户的信任度值

Time	Li_{st}	$Chen_{te}$	Sun_{st}
07-01	0.6	0.8	0.6
07-02	0.7	0.85	0.8
07-03	0.6	0.7	0.7

Table 7 System state space

表 7 系统状态空间

Time	*sdrrs*∈*DRADS*∪*DRGRS*	*drass*∈*DRASS*	*drah*∈*DRAH*	*drgss*∈*DRGSS*	*drgh*∈*DRGH*
07-01AM	$((Li_{st},r''_{MT\text{-}tree}),VST,grant)$ $((Chen_{te},r''_{MT\text{-}tree}),VST,grant)$ $((Chen_{te},r''_{MT\text{-}tree}),activate)$	$(Chen_{te},r''_{MT\text{-}tree})$	$(Chen_{te},r''_{MT\text{-}tree})$	$((Chen_{te},r''_{MT\text{-}tree}),VST)$	$((Chen_{te},r''_{MT\text{-}tree}),VST)$
07-01PM	$((Chen_{te},r''_{MT\text{-}tree}),deactivate)$			$((Chen_{te},r''_{MT\text{-}tree}),VST)$	
07-02AM	$((Chen_{te},r''_{MT\text{-}tree}),activate)$ $((Li_{st},r''_{MT\text{-}tree}),VST,grant)$ $((Li_{st},r''_{MT\text{-}tree}),activate)$	$(Chen_{te},r''_{MT\text{-}tree})$ $(Li_{st},r''_{MT\text{-}tree})$	$(Chen_{te},r''_{MT\text{-}tree})$ $(Li_{st},r''_{MT\text{-}tree})$	$((Chen_{te},r''_{MT\text{-}tree}),VST)$ $((Li_{st},r''_{MT\text{-}tree}),VST)$	$((Li_{st},r''_{MT\text{-}tree}),VST)$
07-02PM	$((Li_{st},r''_{MT\text{-}tree}),deactivate)$ $((Li_{st},r''_{MT\text{-}tree}),VST,revoke)$ $((Chen_{te},r''_{MT\text{-}tree}),deactivate)$			$((Chen_{te},r''_{MT\text{-}tree}),VST)$	
07-03AM	$((Sun_{st},r''_{ST\text{-}tree}),VST,grant)$ $((Chen_{te},r''_{MT\text{-}tree}),VST,grant)$			$((Chen_{te},r''_{MT\text{-}tree}),VST)$	

在表 4 的委托凭证 $dc_{(Li,M)}$中, $dtt^{0.6}_{VST\text{-}AD1}$ 的持有者的信任度需由委托源(资源提供者)验证,其值不低于 0.6 是 $dc_{(Li,M)}$被委托源验证通过的前提. $dtt^{0.7}_{Li\text{-}VST}$ 是虚拟组织 *VST* 签发给学生 Li_{st} 的委托票据,它限定 Li_{st} 的信任度必须

不低于 0.7 才能激活被授予的委托角色.*pt*,*n*,*ae* 未给出具体值,默认继承委托者的值.*dept*={$(Any_{te}, r''_{MT\text{-}tree})^{0.8}$}体现了 *VST* 授权策略中的规定②.*degt*={$(Any_{te}, r''_{MT\text{-}tree})^{0.85}$,¬$(Any_{te}, r_{ST\text{-}tree})$}体现了 *VST* 授权策略中的规定①、规定③.其他凭证类似,不再详述.在时间 07-01 之前没有任何用户请求,在 07-01 开始有用户请求.表 5 列出的用户委托角色请求序列反映了第 2.3 节所述的 *VPEP*1 和 *VPEP*2 的实现过程以及 *VST* 系统的资源使用策略.

表 7 展示了第 2.3 节模型实现过程的状态变化,其中体现了如下关联性限制特征:委托角色授予依赖、带信任度的委托角色激活依赖、带信任度的委托角色授予依赖、静态职责分离.我们也可以看到,各自治域通过委托凭证的宽度限制域表达了对 *VST* 系统的负荷限制.*VST* 系统利用委托角色的动态授予与撤销的动态特性达到了充分利用有限委托角色资源以服务更多用户的目的.

4 结束语

本文的 HDAEM 在层次角色环境下,对委托的临时性、关联性、部分性、传播性限制进行了全面支持,克服了现有的相关模型只支持其中某些方面的不足,并很好地支持了委托角色授予或撤销的动态特性.此外,还增强了关联性限制功能、降低了部分委托实现的难度、细化了委托传播控制,并支持对委托传播上下文信息的表达.本文提出的委托凭证将上述特征进行了统一封装,并作为授权执行过程细粒度控制的依据.在此基础上提出的委托授权执行规则细粒度地控制了委托授权执行过程.下一步将对服务网格的授权决策模型进行研究.

References:

[1] Open Grid Forum. Project: OGSA-AUTHZ-WG. 2007. https://forge.gridforum.org/sf/projects/ogsa-authz

[2] Pearlman L, Welch V, Foster I, Kesselman C, Tuecke S. A community authorization service for group collaboration. In: Proc. of the IEEE 3rd Int'l Workshop on Policies for Distributed Systems and Networks. Washington: IEEE Computer Society, 2002. 50−59. http://www.globus.org/alliance/publications/papers/CAS_2002_Revised.pdf

[3] Chadwick DW, Otenko A. The PERMIS X.509 role based privilege management infrastructure. In: Proc. of the 7th ACM Symp. on Access Control Models and Technologies (SACMAT 2002). Monterey: ACM, 2002. 135−140. http://sec.cs.kent.ac.uk/download/SACMATfinal.pdf

[4] Thompson M, Essiari A, Mudumbai S. Certificate-Based authorization policy in a PKI environment. ACM Trans. on Information and System Security (TISSEC), 2003,6(4):566−588.

[5] Lorch M, Adams D, Kafura D, Koneni M, Rathi A, Shah S. The PRIMA system for privilege management, authorization and enforcement in grid environments. In: Stockinger H, Baker M, eds. Proc. of the 4th Int'l Workshop on Grid Computing (Grid 2003). Los Alamitos: IEEE Computer Society, 2003. 109−116.

[6] Welch V, Foster I, Kesselman C, Mulmo O, Pearlman L, Tuecke S, Gawor J, Meder S, Siebenlist F. X.509 proxy certificate for dynamic delegation. In: Proc. of the 3rd Annual PKI Workshop. 2004. 20−25. http://www.globus.org/alliance/publications/papers/pki04-welch-proxy-cert-final.pdf

[7] Ferraiolo DF, Sandhu R, Gavrila S. Proposed NIST standard for role-based access control. ACM Trans. on Information and System Security, 2001,4(3):224−274.

[8] Chadwick DW. Delegation issuing service for X.509. In: Proc. of the NIST 4th Annual PKI Workshop. 2005. 62−73. http://middleware.internet2.edu/pki05/proceedings/chadwick-delegation_issuing.pdf

[9] Sun WQ, Shan BH, Zhang C, Liu C. A role-based delegation access control model for virtual organization in service grid. Chinese Journal of Computers, 2006,29(7):1199−1208 (in Chinese with English abstract).

[10] Zhang LH, Ahn GJ, Chu BT. A rule-based framework for role-based delegation. In: Sandhu RS, Jaeger T, eds. Proc. of the 6th ACM Symp. on Access Control Models and Technologies. New York: ACM Press, 2001. 153−162.

[11] Joshi JBD, Bertino E, Ghafoor A. Temporal hierarchy and inheritance semantics for GTRBAC. In: Proc. of the 7th ACM Symp. on Access Control Models and Technologies. New York: ACM Press, 2002. 74−83. http://cobweb.ecn.purdue.edu/~dmultlab/Security/sacmat2002.pdf

[12] Zhang XW, Oh S, Sandhu R. PBDM: A flexible delegation model in RBAC. In: Ferrari E, Ferraiolo D, eds. Proc. of the 8th ACM Symp. on Access Control Models and Technologies. New York: ACM Press, 2003. 149−157.

[13] Xu Z, Li L, Feng DG. A constrained role-based delegation model. Journal of Software, 2005,16(5):970−978 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/970.htm

[14] Zhai ZD, Feng DG, Xu Z. Fine-Grained controllable delegation authorization model based on trustworthiness. Journal of Software, 2007,18(8):2002−2015 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/2002.htm

[15] Barka ES. Framework for role-based delegation models [Ph.D. Thesis]. Fairfax Virginia: George Mason University, 2002.

[16] Bertino E, Bonatti PA, Ferrari E. TRBAC: A temporal role-based access control model. ACM Trans. on Information and System Security, 2001,4(3):191−233.

附中文参考文献:

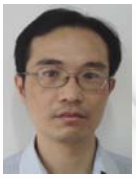
[9] 孙为群,单保华,张程,刘晨.一种基于角色代理的服务网格虚拟组织访问控制模型.计算机学报,2006,29(7):1199−1208.

[13] 徐震,李斓,冯登国.基于角色的受限委托模型.软件学报,2005,16(5):970−978. http://www.jos.org.cn/1000-9825/16/970.htm

[14] 翟征德,冯登国,徐震.细粒度的基于信任度的可控委托授权模型.软件学报,2007,18(8):2002−2015. http://www.jos.org.cn/1000-9825/18/2002.htm

陈志刚(1964－),男,湖南长沙人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络计算,网络与信息安全.

郭迎(1975－),男,博士,副教授,CCF 会员,主要研究领域为网络与信息安全.



桂劲松(1968－),男,博士生,讲师,CCF 会员,主要研究领域为网络与信息安全.