An Extended RBAC Model for Task Delegation in Workflow Systems

Khaled Gaaloul[1], Erik Proper[1,2], and François Charoy[3]

[1] Public Research Centre Henri Tudor,
L-1855 Luxembourg-Kirchberg, Luxembourg
[2] Radboud University Nijmegen
P. O. BOX 9010 6500, GL Nijmegen, The Netherlands
[3] LORIA, Université de Lorraine
BP 239, F-54506 Vandœuvre-lès-Nancy Cedex, France
{khaled.gaaloul,erik.proper}@tudor.lu,charoy@loria.fr

Abstract. In role-based access control models, delegation of authority involves delegating roles that a user can assume or the set of permissions that he can acquire, to other users. Several role-based delegation models have been proposed in the literature. However, these models consider only delegation in presence of the role type, which have some inherent limitations to *task delegation* in workflow systems. In this paper, we address task delegation in a workflow and elaborate a security model supporting delegation constraints. Delegation constraints express security requirements with regards to task's resources, user's assignment and *privileges* (delegation of authority). Further, we show how, using a role-based security model, we inject formalised delegation constraints to compute principals and privileges to be specified into delegation policies within an access control framework.

Key words: Workflow, access control, delegation, constraints, privileges, authorisation policy.

1 Introduction

With the broad adoption of workflow management systems to model and automate business processes cross organisations, security becomes a crucial and essential topic. Typically, activities that are part of a process are represented as tasks. Organisations establish a set of authorisation policies that regulate how business processes and resources should be managed within a workflow [1]. Authorisation information is given which authorises users to perform tasks. Such authorisation information may be specified using a simple access control list or more complex role-based structures [2].

In current workflow management systems, the role-based access control (RBAC) model is widely adopted, where system administrators assign roles to users. It is more convenient for administrators to manage roles than to manage users directly [3]. One important factor that affects access control (authorisation) distribution among users is delegation. Delegation involves a user passing

its authority to other users. If delegation is allowed, a delegator delegates authority (a privilege) to another active entity, called the delegatee, to carry out a task on behalf of the former. In the context of workflow systems, delegation can be very useful for real-world situations where a user who has to perform a task is either unavailable or too overloaded [4]. Hence, we define task delegation as a means for assigning a task and its access rights from a delegator to a delegatee.

The concept of delegation has been presented in [1, 5]. Significant contributions to role-based delegation can be found in [6, 7]. While much of the work in the area of delegation is limited to role-based access control, the goal of our paper is to consider task delegation constraints in workflow systems. Delegation constraints needs to tackle several issues with regards to workflow's invariants in terms of users, tasks and resources. In doing so, we need to come up with an access control model supporting the assignment of task delegation. Delegation assignment deals with delegation principals (delegator, delegatee) their respective rights (privileges) and their availability (no conflicts during task assignment). In this paper, we extend the RBAC model of Sandhu et al. [3] in two directions: (i) our formal security model defines a *Task-oriented Access Control (TAC)* model which is capable of supporting task assignment condition in workflows and (ii) we leverage TAC specifications to inject delegation constraints, thereby computing potential delegatees and their required privileges, thereby specifying them in terms of delegation policies.

The remainder of this article is organised as follows. Section 2 presents fundamental concepts of the organisational management in workflows. Section 3 defines workflow authorisation constraints during task execution. In section 4, we present a formal security model to reason about task assignment within a workflow. This model is used to integrate delegation constraints in order to compute delegatees with their respective privileges and to specify delegation policies in section 5. Section 6 presents related work. Finally, we conclude and discuss future work.

2 Background

In this section, we aim to give an overview of the organisational aspect to support human and material resources specifications in the context of workflow management systems. The aforementioned resources will play an important role to define task delegation constraints and its security requirements in Sect. 4.

2.1 Resource Management in Workflows

A workflow is made of tasks, where a task defines a unit of work that at each invocation performs the binding between different resources needed to complete a specific part of the workflow [8]. The resources that may be involved are different. We distinguish material and human resources for business objects and workflow's actors, respectively. Generally, the manipulation of material resources is interfaced by one or several entities called applications or services.

A resource model contains the definition of human and material resources that are involved in the execution of a workflow model. While the resource model is a structured representation of organisational entities, it should be noted that both this model as well as the elements contained therein follow a life cycle and change over time. Therefore, a workflow management system not only needs to provide a mechanism to represent the organisational elements involved in the execution of workflows, but it also needs to provide mechanisms for continuous change within these elements [9]. Our change scope in this paper deals with task delegation.

2.2 Organisational Resources Analysis

During the design time, the workflow application designer has to design both the structure of the business process to be automated, and the structure of the resources that carry out the process. Resources and workflow's tasks are linked through the construct role [10]. From a process perspective, a role is a subject to authorisations that define permissions (operations) for the execution of a task. From a resource perspective, a role represents a granted authorisation for a workflow actor (so-called user). Based on these two perspectives, the design of the resource model can follow two different directions namely the material and human resources. Material resources define business objects and the way to use them. Human resources define the actors of the workflow.

From a material resource perspective, we define permissions as functions with operations to manipulate business objects. From a human resource perspective, we define a subject as an assigned user who is member of a role to claim a task instance. The task execution is added to the subject's worklist. A worklist defines the set of task instances claimed by this subject. The access to resources will be dependent on the execution model of the task. Figure 1 shows a meta model for a task-based organisational structures, which analyses the possible ways the resources access can be defined during the task execution. Figure 1 includes a white and a blue blocks. Each block defines a set of concepts and their relationships when executing a task within a workflow. The white block represents the material resource to carry out a task, and the blue block defines how a human resource (an actor) is managed to execute a task. This distinction will help us to specify our task-oriented access control model (see Sect. 4).

In figure 1, we define a task as a set of applications or services that are accessed by subjects via specific functions. These applications consist of functions that manipulate business objects. From one task several task instances can be generated. Note that we distinguish task type element from task since we assume that a task represents an instantiation of a task type during execution, equally for business objects. A task instance corresponds to an actual execution of a task. This specific execution of the task (a task instance) is allocated to only one subject through its unique worklist, where a subject defines a user selecting a role during runtime.

We aim to address issues related to the organisational management in workflow systems with regards to user's assignments, task's definition and resource's

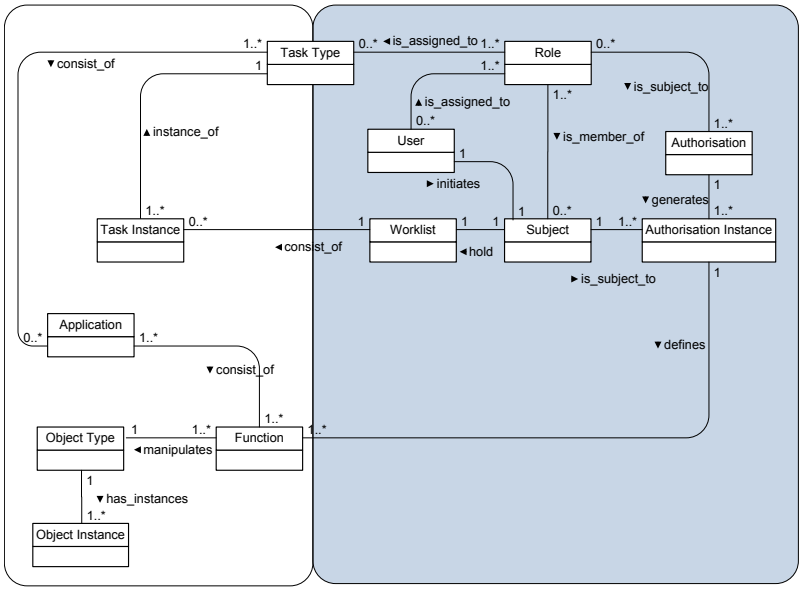

Fig. 1. A model for task-based organisational structures

access. We mainly focus on both material and human resources to analyse task's requirements. We can identify from our analysis the main relationships between tasks, roles and resources.

- The set of *Tasks*, *Applications*, *Objects* and *Functions* defines the material requirements that are necessary to carry out a task.
- The set of *Roles* assigned to a task. A role inherits permissions based on the defined authorisation from an access control model.
- The set of *Authorisations* defines the condition of assignment that a role must have as permissions to execute a task, where a permission defines which action/function to perform over a business object of a task.
- The set of *Subjects*, *Worklists*, *Task Instances* and *Authorisation Instances* defines the task assignment condition.

Authorisation constraints need to be specified under access control models and expressed in terms of authorisation policies that regulate business processes and resources within a workflow [1]. In the rest of this paper, we explain how such constraints are specified and expressed in an extended role-based access control model supporting task delegation.

3 Workflow Authorisation Constraints

A workflow comprises various activities that are involved in a business process. Activities that are part of a process are represented as tasks [11]. Authorisation information is given which authorises users to perform tasks. Such authorisation information may be specified using a simple access control list or more complex role-based structures [12].

A task instance is created and then assigned to a user. The assigned user can start or delegate the task which gathers all operations and rights over the business objects related to task's resources (see Fig. 1). Seeing a task as a block that needs protection against undesired accesses, access control will depend on the specified authorisation information.

We define a permission as an authorisation allowing a user to perform a task. Authorisation makes an explicit binding between a user, a task resource (business object) and his rights over it (function/action). In our work, we define a task oriented access control model based on the RBAC model. We focus on task's requirements to analyse and specify security constraints while accessing workflow's data. Data access defines permissions on business objects related to task's resources.

4 Task-oriented Access Control Model

We propose a *Task-oriented Access Control (TAC)* model to support authorisation requirements in workflow systems (see Fig. 2). Authorisation information will be inferred from access control data structures, such as user-role assignment (URA) and task-role assignment (TRA) relations. In addition, we model permission assignment relations for tasks and roles in order to support the task execution context. The remaining relations are generic relations based on the RBAC model [3].

Formally, we define sets *U, R, OU, T, P, S* and *TI* as a set of users, roles, organisations units, tasks, permissions, subjects and task instances, respectively. We use a subject to denote the time a user selects roles for a session. During the task instantiation assignment, we create a user's current active role set and define it as a subject (see Fig. 2). For example, the user *Alice* with the role *clerk* defines a subject to execute the instance of a task 'Check credit' in a bank loan process.

We define RH (Role Hierarchy), where RH is a partial order on R, r_i and $r_j \in R$. RH denotes that r_i is a role superior to r_j, as a result, r_i automatically inherits the permissions of r_j.

We define RM (Role Mapping), where $RM \subseteq OU_i \times OU_j$ with OU_i and OU_j two organisations units. RM defines external roles accessing distributed resources cross-organisations. It provides a decentralised access control mechanism where externally known roles are publicly available:
$r_k \in OU_i$ and $r_l \in OU_j$, RM denotes that r_l is a role mapped to r_k, as a result, r_l shares the permissions of r_k.

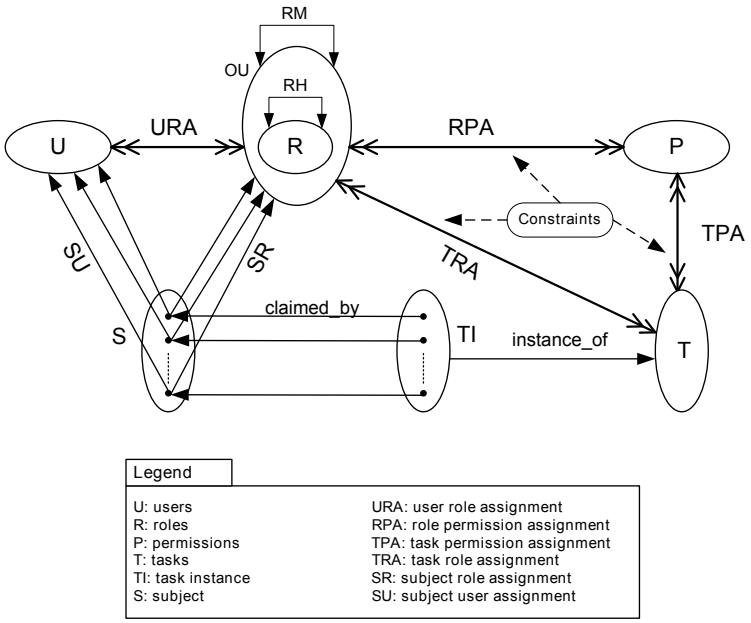

Fig. 2. Task-oriented access control (TAC) model.

4.1 Definitions of Map Relations

Formally, we define a set of relations as follows:

- $URA \subseteq U \times R$, the user role assignment relation mapping users to roles they are member of.
- $RPA \subseteq R \times P$, the permission role assignment relation mapping roles to permissions they are authorised to.
- $TPA \subseteq T \times P$, the task permission assignment relation mapping tasks to permissions. This defines the set of permission required to execute a task (see Definition 2).
- $TRA \subseteq T \times R$ the task role assignment relation mapping roles to tasks they are assigned to.

4.2 Definitions of Functions

Formally, we define a set of functions as follows:

- $SU: S \rightarrow U$ a function mapping a subject to the corresponding user.
- $SR: S \rightarrow R$, a function mapping each subject to a role, where $SR(s) = r, (SU(s), r) \in URA\}$ with a subject s having a permission $p|(r, p) \in RPA\}$.
- $instance_{of}: TI \rightarrow T$, a function mapping a task instance to its task type.

– $claimed_{by}$: $TI \rightarrow S$, a function mapping a task instance to a subject to execute it. It defines the user-task assignment condition s = $claimed_{by}(t_{i1})$ where : $\{t_i = instance_{of}(t_{i1}), (r,u) \in URA | (SR(s) = r \bigwedge SU(s) = u), (t_i, r) \in TRA\}$.

4.3 Definitions of Constraints

Here we discuss Separation of duty (SoD) and Binding of duty (BoD) constraints. It defines security constraints between two tasks that compose a business process [13]. Such constraints help to verify whether a user is not allowed to execute a task due to some conflicts (e.g., conflict of interest). We define an exclusive relation between tasks for SoD, and a binding relation between tasks for BoD :

$$TT_{SOD} : \{(t_i, t_j) \in T \times T \mid t_i \textit{ is exclusive with } t_j\}$$

$$TT_{BOD} : \{(t_i, t_j) \in T \times T \mid t_i \textit{ is binding with } t_j\}$$

If $(t_i, t_j) \in TT_{SOD}$, then t_i and t_j cannot be assigned to the same user. If $(t_i, t_j) \in TT_{BOD}$, then t_i and t_j must be assigned to the same user which defines a binding relation between two tasks.

4.4 Model Contributions

The main contribution of the TAC model is to specify the task assignment relation where two conditions have to be verified: (1) the first condition is related to task's resources requirements. The role's permissions defined in RPA (role-permission assignment) needs to satisfy the permissions defined in TPA (task-permission assignment). (2) the task is executed if and only if the user/role is assigned to it. Basically, having a permission to execute a task but not being assigned to it will not satisfy the outlined conditions and, therefore, will deny the access to its resources.

Definition 1 (Task Assignment). *A task instance t_i is assigned to a user u with an active subject s if and only if :*
$(t, r) \in TRA \Rightarrow \{p \in P | (t, p) \in TPA\} \subseteq \{p | (r, p) \in RPA\} \bigwedge claimed_{by}(t_i) = s$, *where* $(SR(s) = r \bigwedge SU(s) = u)$.

The user-task assignment requires the $claimed_{by}$ function. For instance, a task t_i is assigned a set of permissions based on the TPA relation in order to carry out this task. A user u_1 with a role r_j is assigned to t_i if and only if u_1 verifies the TRA and $claimed_{by}$ conditions. However, if we consider another user u_2 member of same role r_j having the same permissions based on the RPA relation but u_2 is not defined in $claimed_{by}(t_i)$, which means not assigned to this task. In this case, u_2 is not allowed to execute t_i since he does not fulfil the user-task assignment relation (see condition 2).

In the banking process example, let user *Bob* a member of role *Clerk* but not from the same bank agency. Bob is not allowed to perform the task 'Check credit' since he is not assigned by the system to execute it. Within organisations, users can share different roles but are not assigned to the same tasks. This is due to privacy and security constraints such as the separation of duty. Therefore, we leverage condition 2 as an additional constraint when claiming a task instance by a user.

5 Securing Task Delegation

In this section, we leverage the user-task assignment conditions to support task delegation assignment with regards to the delegatees and its required privileges. We use computed privileges to specify delegation policies within an existing access control framework.

5.1 Access Control over Delegation

Delegation is a mechanism that permits a user to assign a subset of his assigned authorisations (privileges) to other users who currently do not possess it.

Definition 2 (Delegation Relation). *We define a delegation relation* $DR \subseteq T \times U \times U \times 2^{DC}$ *where* T *a set of tasks,* U *a set of users and* DC *a set of delegation constraints. A task delegation relation is defined as* DR = (t,u_1,u_2,{DC}), t *is the delegated task and* t ∈ T*,* u_1 *the delegator and* u_2 *the delegatee* ∈ U*.*

For instance, delegation constraints (DC) can be related to time or evidence specifications [4]. In addition, organisational constraints regarding roles mapping cross organisations or role hierarchies within an organisation define user-to-user delegation constraints (see RM and RH relations of the TAC model in Fig. 2). For instance, a subordinate in an organisation hierarchy can act on behalf of his superior where the latter is the delegator and the former is the delegatee.

Here, a delegation relation defines the main constraints to be considered when delegating privileges with regards to users/roles, task and resources. Our focus is to integrate such constraints in a secure manner. In doing so, we leverage the TAC (task-oriented access control) model specifications to compute delegatees and privileges. The TAC model allows to compute the list of potential delegatees using the RPA (role-permission assignment) relation that may satisfy the delegated task requirements based on the TPA (task-permission assignment) relation. In doing so, we define a method for access control over task delegation using TAC. In the following, we detail our method and describe how valid delegatees are checked and whether they need delegated privileges grant.

Input: u_1, $u_2 \in U$; r_1, $r_2 \in R$; t_i, $t_j \in T$.

1. Defining the role and permission assignments for each user (URA and RPA);

2. Instantiating the task t_{i1} and assigning it to the delegator s_1 who is the current user u_1;
3. Checking security constraints before delegation (SoD and BoD);
4. Computing the delegatee s_2, who is the current user u_2, based on his permissions assignment ($(t_i, p_{r2}) \in TPA$) or;
5. Granting privileges for s_2 based on the task instance permissions assignment ($p'_{r2} \leftarrow p_{r2} \cup p_{ti}$) which is defined in the $claimed_{by}$ function;

Output: Delegation relation instance : $dr_1 = (t_{i1}, s_1, s_2, \{DC\})$;

The main contribution of this method is to specify the delegated task assignment conditions based on Definition 2. If the two conditions are satisfied, then the task t_i is delegated to the delegatee u_2. However, if u_2 does not have the permission required and there is no conflicts (BoD or SoD) to execute t_i. Then the delegated privileges are granted for u_2 based on the $claimed_{by}$ function.

The computation of the privileges is based on the TRA and $claimed_{by}$ specifications defined in our TAC model (see $claimed_{by}$ condition for permissions). Basically, we provide a method to compute the least privileges to delegate based on the current requirements of the task instances t_{i1} which is generated from the delegated task. At this stage, delegated privileges are done manually supporting a user-to-user delegation. However, the administration of new access rights has to be specified later into authorisation policies in a compliant and dynamic manner. Authorisation policies will regulate how the business process and resources should be managed during delegation.

5.2 Delegation Policies

We introduce authorisation policies based on our access control (TAC) model. We then identify the delegation constraints that have to be specified in the delegation policies. An access control has to be defined to check the authorisation of the initiating user so-called subject. An authorisation makes an explicit binding between a role (subject), a task resource (object) and his rights (action) over it. This binding is defined based on the main relations: user-role assignment (URA), task-permission assignment (TPA) and task-role assignment (TRA) in our access control model (TAC). Subsequently, an authorisation expresses a user's permissions on a task's resources, where a permission is the right to execute an action on a resource.

Definition 3. *We define a policy* $P \subseteq target \times rule \times 2^C$*, where* target *defines where a policy is applicable,* rule *is a set of rules that defines the policy decision result, and* C *the policy constraints set that validates the policy rule.*

A target defines the entities of an access request. It is composed of a role associated to the subject and an action on a business object of a task type. A pseudo formal expression of a target is:

```
<target>
  <Subject>[role]
  <Resource>[object]
  <Action>[operation]
  <Task>[task type]
< /target>
```

A rule effect defines an authorisation decision. It can return as a result a permit, a deny or an indeterminate request [14]. Constraints are related to the workflow authorisation specifications. For instance, the separation of duty (SoD) is a constraint for a user-task assignment. In the aforementioned banking process, a pseudo formal expression of a policy for a subject member of role *clerk* on the task T1 'Check credit' on a business object 'bo1' is:

```
<Policy>
  <target>[clerk,bo1,read,T1]
  <rule>[Permit]
  <C>[none]
< /Policy>
```

The policy decision returns the result "Permit" where the user Alice member of role *clerk* can access to the resource 'bo1' of task ' Check credit ' and read it.

5.3 Deployment

We use the PERMIS policy editor for creating and editing delegation policies. PERMIS is a policy based authorisation system, a Privilege Management Infrastructure [15]. Given a username, a target and an action, the PERMIS decision engine says whether the user is granted or denied access based on the policy for the target. The policy is role/attribute based where users are given roles/attributes and roles/attributes are given permissions to access targets.

The interface to the PERMIS decision engine has been enhanced to support dynamic delegation of authority [16]. It can be considered as a lightweight authorisation decision engine. In order to execute our delegation request, we use the policy tester which is a tool used to test PERMIS policies created by the policy editor. The PERMIS Policy Tester can also allow dynamic updates of policies. This offers a suitable solution to add new delegation rules that grant or revoke delegated privileges. However, this tool does not support dynamic policies and any further changes in policy will be made externally from PERMIS. A prototype of this implementation can be found in [4] (cf. pp. 156-166).

6 Related Work

Barka et al. proposed a role-based delegation model based on the RBAC model. Their unit of delegation is a role. Authors focused also on role-based models supporting role hierarchies when studying delegation in the context of both

RBAC0 model (flat roles) and RBAC1 model (hierarchical roles) of the RBAC96 family [6]. In this paper, we motivated additional requirements where users may want to delegate a piece of permission. This is the case when computing delegated privileges which are not covered by RBAC.

Task-based access control (TBAC) aims to provide a task context during permission assignments [17]. A workflow system consisting of tasks is assumed. Each of these tasks is then assigned a "protection state", providing information as to who gets to have which permission on a task basis. According to the current state of the workflow system moving through the process instance, different permission assignments are activated or deactivated as ordered by the protection state. The TBAC design is process oriented, however, ignoring human-centric interactions such as user-to-user delegation. Delegation involving users is discussed in the TAC model and aligned with the workflow invariants in terms of tasks, users, and resources.

Team based access control (TMAC) is an access control scheme similar to RBAC, but it provides the assignment of both users and permissions to teams [18]. Each team then is bound to the task it was created for. At runtime, more than one team can be created out of the same template, but each team will be working on a different task instance and accordingly will need access to different object instances. TMAC model is out of the scope of this paper where we consider constraints on tasks and users rather than a team.

There exists several work about delegation policies. In [16, 19], authors investigated how an authorisation management system based on the XACML (eXtensible Access Control Markup Language) can be extended to use flexible delegation mechanisms. The proposed architecture offers a flexible and dynamic way to manage users credentials and administrate delegation policies. However, it is not enough to support dynamic delegation of authority. Delegating a task requires more effort and involves additional specifications related to delegation constraints. In this paper, we proposed an approach to inject delegation constraints within an access control model as a means to specify dynamic delegation policies within workflows.

7 Conclusion

In this paper, we integrated task delegation constraints into a formal security model. In doing so, we analysed task authorisation constraints to support security requirements for delegation. Based on the RBAC model, we proposed the task-oriented access control (TAC) model. This model can grant authorisations based on workflow specifications and user authorisation information. It offers a fine grained access control protocol to support delegation. Moreover, we presented a method to compute potential delegatees and their delegated privileges, thereby specifying delegation policies in existing access control framework.

The next stage of our work is the implementation of our approach within an existing workflow system supporting human interactions. Intalio Tempo is a set

of runtime components that support human-centric workflow within a service-oriented architecture. The main goal is to provide a complete and extensible workflow solution with a bias towards interoperable technologies such as BPEL, BPEL4People, RBAC, and web services. In this context, we will work on extending the security framework based on RBAC with the delegation of authority constraints defined in our model.

References

1. Atluri, V., Warner, J.: Supporting conditional delegation in secure workflow management systems. In: SACMAT '05: The tenth ACM symposium on Access control models and technologies, New York, NY, USA (2005) 49–58
2. Crampton, J., Khambhammettu, H.: On delegation and workflow execution models. In: SAC '08: Proceedings of the 2008 ACM symposium on Applied computing, New York, NY, USA, ACM (2008) 2137–2144
3. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer **29**(2) (1996) 38–47
4. Gaaloul, K.: A Secure Framework for Dynamic Task Delegation in Workflow Management Systems (2010) Ph.D. thesis, The University of Henri Poincaré, Nancy, France.
5. Crampton, J., Khambhammettu, H.: Delegation in role-based access control. In: Proceedings of the Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006. Lecture Notes in Computer Science, Springer (2006) 174–191
6. Barka, E., Sandhu, R.: Framework for role-based delegation models. In: Proceedings of the 16th Annual Computer Security Applications Conference, Washington, DC, USA, IEEE Computer Society (2000) 168–176
7. Zhang, X., Oh, S., Sandhu, R.: PBDM: a flexible delegation model in RBAC. In: SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies, New York, NY, USA, ACM Press (2003) 149–157
8. Russell, N., van der Aalst, W.M.P., Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: Identification, representation and tool support. In: Proceedings of the Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, Portugal. (2005) 216–232
9. Zur Muehlen, M.: Workflow-based Process Controlling. Foundation, Design, and Application of workflow-driven Process Information Systems. Logos Verlag Berlin (2004)
10. Curtis, B., Kellner, M.I., Over, J.: Process modeling. Commun. ACM **35**(9) (1992) 75–90
11. WFMC, The Workflow Management Coalition: Workflow Management Coalition Terminology and Glossary (1999) Document Number WFMC-TC-1011.
12. Crampton, J., Khambhammettu, H.: Delegation and satisfiability in workflow systems. In: SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies, New York, NY, USA, ACM (2008) 31–40
13. Botha, R.A., Eloff, J.H.P.: Separation of duties for access control enforcement in workflow environments. IBM Systems Journal **40**(3) (2001) 666–682
14. Tim Moses: eXtensible Access Control Markup Language (XACML) Version 2.0 (2005) Committee specification, OASIS.

15. Chadwick, D.W., Otenko, A.: The permis x.509 role based privilege management infrastructure. In: SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies, New York, NY, USA, ACM (2002) 135–140
16. Chadwick, D.W., Otenko, S., Nguyen, T.A.: Adding support to xacml for multi-domain user to user dynamic delegation of authority. Int. Journal Information Security **8**(2) (2009) 137–152
17. Thomas, R.K., Sandhu, R.S.: Task-based authorization controls (tbac): A family of models for active and enterprise-oriented autorization management. In: Proceedings of the IFIP TC11 WG11.3 Eleventh International Conference on Database Securty XI, London, UK, UK, Chapman & Hall, Ltd. (1998) 166–181
18. Thomas, R.K.: Team-based access control (tmac): a primitive for applying role-based access controls in collaborative environments. In: RBAC '97: Proceedings of the second ACM workshop on Role-based access control, New York, NY, USA, ACM (1997) 13–19
19. Seitz, L., Rissanen, E., Sandholm, T., Firozabadi, B.S., Mulmo, O.: Policy administration control and delegation using xacml and delegent. In: GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, Washington, DC, USA, IEEE Computer Society (2005) 49–54