A Secure Task Delegation Model for Workflows

Khaled Gaaloul* †, Andreas Schaad*, Ulrich Flegel*, François Charoy†
*SAP CEC Karlsruhe, Security & Trust Group
Vincenz-Priessnitz-Strasse 1, 76131 Karlsruhe, Germany
Email: khaled.gaaloul@sap.com, andreas.schaad@sap.com, ulrich.flegel@sap.com
†LORIA - INRIA - CNRS - UMR 7503
BP 239, F-54506 Vandœuvre-lès-Nancy Cedex, France
Email: charoy@loria.fr

***Abstract*—Workflow management systems provide some of the required technical means to preserve integrity, confidentiality and availability at the control-, data- and task assignment layers of a workflow. We currently observe a move away from predefined strict workflow enforcement approaches towards supporting exceptions which are difficult to foresee when modelling a workflow. One specific approach for exception handling is that of task delegation. The delegation of a task from one principal to another, however, has to be managed and executed in a secure way, in this context implying the presence of a fixed set of delegation events. In this paper, we propose first and foremost, a secure task delegation model within a workflow. The novel part of this model is separating the various aspects of delegation with regards to users, tasks, events and data, portraying them in terms of a multi-layered state machine. We then define delegation scenarios and analyse additional requirements to support secure task delegation over these layers. Moreover, we detail a delegation protocol with a specific focus on the initial negotiation steps between the involved principals.**

I. Introduction

Many of the complex day to day workflows in large organisations are facilitated and conducted using Workflow Management Systems (WfMS). Security is an essential and integral part of workflows, addressing the properties of *integrity, confidentiality* and *availability*. In a workflow, integrity prevents the unauthorised modification of information, whilst confidentiality implies that no data or resource is accessed by unauthorised users at anytime. Availability moreover, implies that a resource should be available when it is needed.

Within WfMS research we observe a tendency moving away from strict enforcement approaches towards mechanisms supporting exceptions that make it difficult to foresee when modelling a workflow. Along those lines one specific set of mechanisms is task delegation that allow at workflow execution time for the exception-based delegation of a task [1]. Consequently, the delegation of a task can be very useful for real-world situations where a user who is authorised to perform a task is either unavailable or too overloaded with work to successfully complete it. This can occur, for example, when certain subjects are sick or on leave. It is frequently the case that delaying these task executions will violate time constraints on the workflow impairing the entire workflow execution. Delegation is a suitable approach to handle such exceptions and to ensure alternative scenarios by making WfMS flexible and efficient.

Most of the work done in the area of security constraints and requirement modelling is focused on the infrastructure of WfMS and secure transaction management in workflow execution [2], [3]. There exists little related work in the domain of specifying task-based delegation. This observation is supported by research done by Aalst *et al.* [4] and Hung *et al.* [2]. They outlined that existing solutions, such as the Workflow Authorisation Model (WAM) [3], are static and do not support sufficient security constraints from build-time to run-time of a workflow. Moreover, model-based access control mechanisms cannot satisfy most criteria required for a secure task delegation model with regards to the aspects of users, tasks, events and data [5].

In order to tackle these problems we need to address two important issues, namely allowing the delegation task to complete, and having a secure delegation within a workflow. Allowing task delegation to complete requires a model that forms the basis of what can be analysed during the delegation protocol. Secure delegation implies the controlled propagation of authority, ensuring confidentiality at the control and data flow layers as well as availability at the task assignment layer and integrity at the data layer.

The contributions of this paper are threefold. First, we define a secure task delegation model separating the various aspects of delegation in terms of a multi-layered state machine, where the interactions between the different layers are triggered by delegation events. These delegation events imply appropriate authorisation on the delegatee side for further actions as well as contain required context for those actions. Second, we expand on earlier work we have done [6] to identify real world scenarios supporting delegation and illustrate the working of our secure task delegation model. We finally introduce a delegation protocol, in particular focusing on the initial negotiation steps. We believe that negotiation had not yet been treated in sufficient detail and accordingly, a more detailed discussion around negotiation in the delegation protocol and its main states and operations is a part of this paper.

The remainder of this paper is organised as follows. Section 2 proceeds with an overview of a task life cycle and extends it to support delegation. In Section 3 we present our secure task delegation model within a workflow analysing relationships between users, tasks, events and data. An e-government case study is presented in Section 4 and several delegation scenarios

are defined on basis of this case study. In Section 5 we introduce a delegation protocol to support our model. Section 6 describes the related work and compares it to our approach. In Section 7 we discuss and conclude our approach, and outline several topics of potential future work.

II. Task Delegation Model

In this section, we provide the basic definitions and terminologies based on the workflow management coalition specifications [7]. In the context of a workflow, a process is composed of a number of activities which are connected in the form of a directed graph. An activity describes a piece of work that forms one logical step within a process. During execution, an activity instance includes tasks or services which are human implementations or computerised implementations of an activity.

A. Task life cycle

In this paper, our main concern is the task level. A task corresponds to a single unit of work. Each executing task is termed a *work item* [4]. In an elementary form, a task is an atomic unit of work. In a compound form, it modularises an execution order of a set of subtasks. It can define a sub-process or a block of tasks. In this paper, we consider the elementary form for simplicity's sake. The basic states for a task life cycle are *Initial, Assigned, Executed, Failed*, and *Completed* [7].

B. Basic task delegation model

Delegation can be introduced to a task model through an extension that supports additional states and transitions. *Delegate* is closely related to the *Assign* transition, where the assigned user has the authority to *Execute* or *Delegate* the task. The *Revoke* transition is derived from *Delegate* transition, such that it can be considered as the cancellation of the task delegation. The internal delegation states are Executed, Revoked and re-Assigned. The delegation behaviour remains internal according to the task model, where Completed and Failed are the final states (see *Task Layer* in Fig. 1).

III. Secure Task Delegation Model

Olivier *et al.* state that a workflow system should be considered at three levels in terms of its components (task assignment, control, data) [9]. Securing a workflow involves enforcing security principles at all three levels. Hung *et al.* developed a secure workflow model using a multi-layered state machine to manage the flow of authorisations at different layers for a secure workflow execution [2]. A multi-layered state machine can enable the analysis, simulation and validation of the WfMS under study before proceeding to implementation. In addition, it can serve as a powerful tool for modelling a secure framework at a conceptual and logical level with regards to the aspects of task, control and data [2].

We present a secure task delegation model using a multi-layered state machine within a workflow. A workflow is represented as a partially ordered set of tasks that is coordinated by a set of events. An event can be either a data event or control event. For instance, control events may refer to the task delegation transitions defined in Sect. II-B. We define three layers: *Task, Control* and *Data* (see Fig. 1).

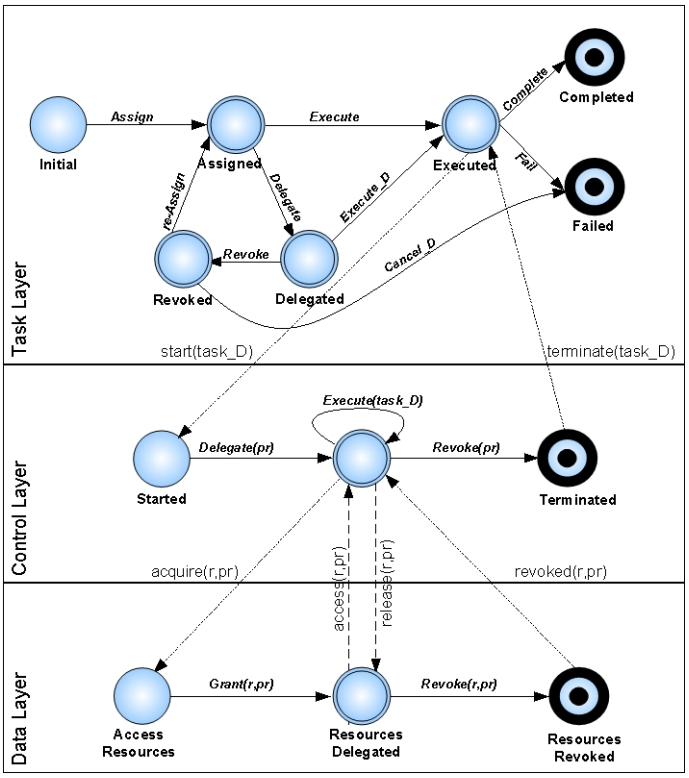

Fig. 1. **Multi-layered state machine for secure task delegation**

The novelty of this model lies in the separation of the various aspects of delegation, and in its portrayal as a multi-layered architecture. The major motivations for using a multi-layered state machine are the modelling of different aspects of authorisations in a single framework, and the ability to address different security services to handle the security properties in different layers. For instance, discretionary access control services can be applied to *Task Layer* and *Data Layer* to handle the security property of authorisation for delegating or revoking tasks and resources to and from users, respectively.

We impose security requirements on events to ensure the security properties of integrity, confidentiality and availability. During a delegation request the interactions between the different layers are triggered by delegation events. These delegation events imply appropriate authorisation on the delegatee side for further actions (starting the delegated task) as well as contain required context for those actions (accessing delegated task resources).

In the *Task Layer*, we require that *Assign* defines availability: *"For every task there must be at least one user (delegator) who is able to execute (delegate) the task"*. In addition, the assignment of the task means that the user has the authority to execute it, thereby controlling confidentiality and integrity of the assigned task. The list of potential delegators could be computed using the optimal user-activity assignment approach

defined in [10].
In the *Control Layer*, *Delegate* defines the authority of delegating a task. We define a privilege *(pr)* as a role assignment or action on a resource. We require that *"A delegatee can only perform the delegated task if and only if the task is delegated and delegated privilege is granted by the delegator"*. The control layer monitors the behaviour of the task delegation. It involves the events generated from the task assignment layer and will generate events to trigger the data layer to be executed (see *acquire (r,pr) in Fig. 1*).
In the *Data Layer*, data are stored as resources. We define *(r,pr)* as a delegated resource to the delegatee. We require that *"A delegatee can only access delegated resources if and only if the delegated privilege is granted to access the delegated resources"*. Granting and revoking resources will ensure the integrity and confidentiality of resources.

Note that we define additional events supporting concurrent states. This is a practical property for a workflow model because there may be more than one delegated task running concurrently and also a given resources can be accessed by a set of concurrently running tasks. In order to avoid an over-privileged delegatee at anytime during the execution of a task, a delegatee is asked to release the resource based on the agreement with the delegator (see *release(r,pr)* in Fig. 1).

IV. Delegation Scenarios

In previous work [6] we presented delegation scenarios inspired from case studies delivered in the European research project R4eGov [11]. Mutual Legal Assistance (MLA) is an e-government workflow scenario involving two national authorities of different European countries regarding the execution of measures for protecting a witness in a criminal proceeding (see Fig. 2). The scenario consists of the definition of a request for assistance, that leads to the preparation of the legal documents that handle the request.

The following describes a set of delegation scenarios that outline the working of our secure delegation model from task delegation to resource access and introduce the definition of a delegation protocol (see Sect. V).

- Scenario 1: We consider the delegation of *"Issuing the request of assistance"* task. The delegation is local and the delegatee is a subordinate who is able and authorised to perform the task. The delegator would also define the resources required by the delegatee. In this case, we assume that the delegator knows the resources bounded to the task thereby providing the delegated credentials (privileges) to access resources.
- Scenario 2: We consider the delegation of *"Opening the content of a WorkFile"* task. When this task is delegated, the delegator would find the list of his subordinates and so potential delegatees (delegatees pool). Delegating this task acquire the access to CMS database where resources are needed to translate, store, and forward documents. In this case, resources access is negotiated between the delegation principals and a delegated privilege is issued to grant the access. Note that the task delegation process include additional negotiation specification. For instance, the expected result defined as evidence where evidence could be quantified as a numeric value or simply qualified as a description of the delegated task [1].
- Scenario 3: We consider the delegation of *"Providing the request of assistance"* task. It is composed of several subtasks. To ensure the integrity and availability of the task, we define on the delegator side the list of delegatees authorised to execute the task as well as the required resources while keeping the privacy of the process. Note that the task delegation process include additional negotiation specification. For instance, both principals have to agree on the accepted delegated subtasks (workload). When this workload is defined and delegated, the delegator would find the list of accepted subtasks as possible evidence which the delegatee can perform.

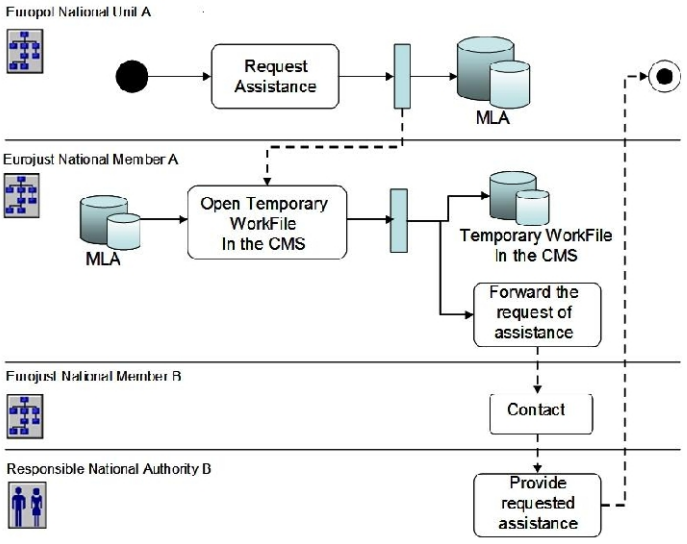

Fig. 2. **Mutual legal assistance scenario**

In summary, scenario 1 describes the delegation of tasks within a local organisation based on an organisational role hierarchy. In scenario 2, evidence on completed delegated tasks is subject to negotiation between the delegator and the delegatee. Scenario 3 describes the delegation of subtasks where workload specifications may depend on evidence issued by the delegator. In the three scenarios, we define delegated privileges to grant access to the task and its resources. The computing of delegated privileges and the definition of minimal rights as well as privacy related aspects are not discussed in this paper. In the next section, we analyse the negotiation as a first step within the delegation protocol and present the different factors that may be discussed during the negotiation such as privileges or evidence.

V. Delegation Protocol

In this section, we introduce a delegation protocol to support the dialogue between a delegator and a delegatee during a secure task delegation. The delegation protocol will support

the defined *Control Layer* with regards to the aspects of users, tasks, events and resources.

A. Protocol overview

A delegation protocol describes request and response message pairs from a delegator to a delegatee. The delegator issues a delegation request and sends it to the delegatee (see *InitDelegReq()* in Fig. 3). The first step consists of negotiating the request based on the request specifications such as deadline, evidence and workload (see *NegotiationReqIssue()* and *NegotiationReqResponse()* in Fig. 3). The delegatee will then decide whether to perform the requested operation and will send the response to the delegator (see *InitDelegResponse()* in Fig. 3). If the request is declined, the delegator will check whether another delegatee exists and then will renew his request (see *RedefineDelegReq()* in Fig. 3).

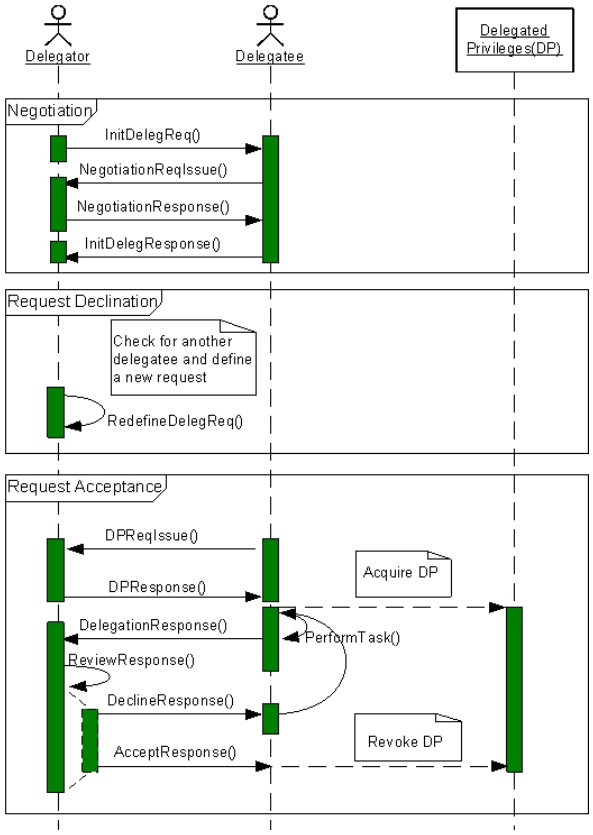

Fig. 3. **Delegation protocol model (sequence diagram)**

If the request is accepted, the delegatee will acquire delegated privileges issued by the delegator (see *DPReqIssue()* and *DPReqResponse()* in Fig. 3). Once delegated privileges are acquired, the delegatee starts performing the delegated task and then sends as a response the execution outcome to the delegator to review it based on evidence specifications defined in the negotiation step (see *DelegationResponse()* in Fig. 3). The reviewing step will lead to the acceptance or the declination of the delegation response, and so the re-assignment or the acceptance of the delegated execution task (see *DeclineResponse()* and *AcceptResponse()* in Fig. 3). Finally, the acceptance step will complete the task and revoke the delegated privileges.

As described in the sequence diagram, we can identify three main steps: *Negotiation*, *Request Declination*, and *Request Acceptance*. The two last steps depend on the negotiation. We consider *Negotiation* as the trigger point for the main operations in the delegation protocol.

B. Negotiation

We consider the negotiation step as a fundamental step for the delegation protocol. It involves all the principals (delegator and delegatee) and negotiation specifications (e.g. evidence, time). Our intention is to envisage a wide-ranging request that gives flexibility for the delegation request. Negotiation specifications will ensure this flexibility. In the following, we present a taxonomy of factors that are relevant in the delegation scenarios described in Sect. IV.

- **F1. Scope:** This factor describes the scope of delegation (e.g. multistep) [12]. Basically, the delegator proposes the degree of delegation regarding the context of delegation (e.g. local, global).
- **F2. Time:** This factor defines one of the delegation constraints: the deadline. Delegation may be actually temporary for some security reasons. This involves a time constraint specifying a time window for the delegatee. This constraint utility can avoid also a long period of inactivity of the delegatee.
- **F3. Workload:** The negotiation here deals with the delegated amount of work. In fact, the delegatee may be overwhelmed by the number of tasks assigned to him, a situation which can be sorted by reducing the workload (see scenario 3 in Sect. IV).
- **F4. Evidence:** This factor is a specific type of business object that can be manipulated by a task. Task execution may generate evidence for review by the delegator [1]. The negotiation here deals with the reviewing specifications issued by the delegator to validate the completion of delegation.
- **F5. Privileges:** This factor can be a role assignment or an action on a resource. Privileges will be granted to the delegatee later on to execute tasks or access specified resources (see Sect. V-E). Privileges can be permanent or temporal depending on time constraints.

In the following, we identify the relationship between delegation principals and negotiation factors (see Table I). We assume that the five factors can be defined by the issuer of the request, the delegator. In fact, he is the user initially assigned to execute tasks and authorised to delegate, and so expresses their delegation specifications. As part of the negotiation some of the factors can be modified by the delegatee. In fact, both principals can negotiate time, workload and evidence. This consists of giving the delegatee the ability to extend the deadline, reduce the workload, and propose suitable evidence, respectively. The other factors are exclusive to the delegator for security reasons.

TABLE I
NEGOTIATION FACTORS SPECIFIED BY DELEGATION PRINCIPALS

Factors	Delegator	Delegatee
F1	✓	
F2	✓	✓
F3	✓	✓
F4	✓	✓
F5	✓	

C. Request Declination

The declination step occurs when the negotiation failed and the proposed specifications are rejected. We consider this step as a precondition to renew the delegation request.

D. Request Acceptance

Our delegation protocol supports several operations that can be requested during acceptance request. The acceptance step consists of granting delegated privileges, performing the task, and reviewing it in order to complete the request. The core operations are defined in the following:

DPReqIssue(): Creates a new privilege according to the needs of the delegatee. The result is a delegated privilege that delegates the negotiated access rights from the delegator to the delegatee.

PerformTask(): Delegated privileges are associated to the delegatee to execute the task and access resources. The delegatee is authorised to perform a task and send feedback to the delegator based on negotiated evidence.

ReviewResponse(): The reviewing operation is much more complicated. The reviewer (the delegator) has to decide whether to accept or decline the *DelegationResponse()* based on negotiated evidence. This split decision will either lead to the re-execution of the task, or to the acceptance of the response and so the revocation of privileges afterwards.

E. Delegated Privileges

Delegated privileges are granted by the delegator to the delegatee. In order to control access to the delegated task's resources, both authentication and authorisation are needed. Authentication and authorisation requirements are both defined in the delegated privileges.

A Privilege Management Infrastructure (PMI) is to authorisation what a Public Key Infrastructure (PKI) is to authentication. Consequently there are many similar concepts shared between PKIs and PMIs. A public key certificate (PKC) is used for authentication and maintains a strong binding between a user's name and his public key, whilst an attribute certificate (AC) is used for authorisation and maintains a strong binding between a user's name and one or more privilege attributes. Therefore we consider PKC as a passport and AC as a visa.

We assume that both PKC and AC will be issued by the delegator to the delegatee. Certification can not be negotiated by the delegatee since the delegator has to align his certification request with the authorisation policy of the process in general. Specially, certification is based on the authorisation policy of the domain application. Domain application can be defined as an RBAC model for specifying authorisations. RBAC has the advantage of scalability over DAC, and can easily handle large numbers of users [13]. For instance, we developed a Role Based Privilege Management Infrastructure. It supports authentication and authorisation requirements where a delegator access resources via an application gateway including authentication and authorisation functions. Our technology mapping is inspired from the PERMIS project infrastructure [13].

VI. RELATED WORK

The Workflow Management Coalition (WfMC) summarises a number of security services for a conceptual workflow model including authentication, authorisation, access control, data integrity, security management and administration [14]. Unlike our approach, WfMC does not consider different layers of authorisations among tasks, control and data flow layer during the the modelling and execution of task delegation for WfMS.

Role-based access control (RBAC) is recognised as an efficient access control model for large organisations. Most organisations have some business rules related to access control policy. Delegation of authority is among these rules [15]. Sandhu *et al.* extended the RBAC96 model by defining some delegations rules [5]. They proposed a role-based delegation model (RBDM). Users however may want to delegate a piece of permission from a role [16]. Zhang *et al.* proposed a flexible delegation model named Permission-based Delegation Model (PBDM). Neither RBAC nor PBDM models however, proposed a secure task delegation model supporting integrity, confidentiality and availability.

The Workflow Authorisation Model (WAM) presents a conceptual, logical and execution model that concentrates on the enforcement of authorisation flow in task dependency and transaction processing [3]. Though WAM discusses the synchronisation of authorisation flow with the workflow and specification of temporal constraints in a static approach, it is not sufficient to support workflow security in general and task delegation in particular. This is due to workflows needing a more dynamic approach to synchronise the flow of authorisations during the workflow execution. WAM does not discuss the order of operation flow such as task delegation within a workflow. In a workflow, we need to investigate the delegation control in different aspects such as tasks, events and data by leveraging the required authorisations to secure our delegation.

Russel *et al.* proposed an approach supporting delegation [4]. They described the life cycle of a work item in the form of a state transition diagram with a particular focus on the resource allocation perspective. One of the main drawbacks of this approach is that it defines a static binding of all work items associated with a task to a single resource. This is a static approach that ignores additional events (transitions) during delegation execution and does not support secure and dynamic interactions within a workflow with regards to aspects of users,

tasks, events, and data. This is the major contribution of this paper.

In this paper, we do not distinguish the delegation of privileges based on *grant* or *transfer* [17]. Crampton *et al.* developed a comprehensive delegation model for role-based access control that provides support for both grant and transfer delegation policies [17]. In addition, authors focused 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 of models. This will be an immediate priority in our future work to enrich the task delegation model by supporting hierarchical delegations and enforcing authorisation mechanisms based on delegation policies.

VII. Conclusion and Future Directions

In this paper we presented a secure task delegation model to manage and in parts enforce integrity, confidentiality and availability within a workflow. The novelty of this model lies in the separation of the various aspects of delegation, and in its portrayal as a multi-layered state machine. The interaction between different layers is triggered by delegation events. In fact, delegation events ensure the appropriate authorisation of principals to delegate or revoke a task and access resources, thereby supporting security properties. We proposed a delegation protocol based on our delegation model and discussed negotiation factors and their impact on the protocol and later concrete technical realisation of privileges. Our analysis is based on real world processes from an e-government case study.

Note that in this context, we abstracted from the eventual technical realisation of the privileges on purpose. Privileges could be anything, ranging from a cryptographic token in a distributed context, to a simple additional entry in an ACL or Capability. We believe that there is a strong dependency to the properties of the negotiation step detailing, for example, that the duration of a delegated task will also serve as a certificate expiry variable. A list of such factors has been identified (Sect. V-B). Future work will now concentrate on a classification of our identified factors against basic authorisation technologies.

The next stage of our work also needs to address the further formalisation of our model by supporting it through an abstract state machine specification and verification, thus addressing properties such as completeness, satisfiability and safety.

VIII. Acknowledgment

This work has been partially funded by the EU Commission under the contract number IST-2004-026650 through the EU integrated project R4eGov as well as by the German Federal Ministry of Economy and Technology under the promotional reference 01MQ07012. The authors take the responsibility for the contents.

References

[1] Andreas Schaad. A framework for evidence lifecycle management. In *Web Information Systems Engineering, Proceedings of the WISE 2007 International Workshops, Nancy, France*, Lecture Notes in Computer Science, pages 191–200. Springer, 2007.

[2] Patrick C. K. Hung and Kamalakar Karlapalem. A secure workflow model. In *ACSW Frontiers '03: Proceedings of the Australasian information security workshop conference on ACSW frontiers*, pages 33–41. Australian Computer Society, Inc., 2003.

[3] Vijayalakshmi Atluri, Wei-Kuang Huang, and Elisa Bertino. An execution model for multilevel seccure workflows. In *Proceedings of the IFIP TC11 WG11.3 Eleventh International Conference on Database Securty XI*, pages 151–165, London, UK, 1998. Chapman & Hall, Ltd.

[4] Nick Russell, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and David Edmond. Workflow resource patterns: Identification, representation and tool support. In *Proceedings of the Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, Portugal*, pages 216–232, 2005.

[5] E. Barka and R. Sandhu. Framework for role-based delegation models. In *Proceedings of the 16th Annual Computer Security Applications Conference*, pages 168–176, Washington, DC, USA, 2000. IEEE Computer Society.

[6] Khaled Gaaloul, François Charoy, Andreas Schaad, and Hannah Lee. Collaboration for human-centric egovernment workflows. In *Web Information Systems Engineering, Proceedings of the WISE 2007 International Workshops, Nancy, France*, Lecture Notes in Computer Science, pages 201–212. Springer, 2007.

[7] Workflow Management Coalition. Workflow Management Coalition Terminology and Glossary. Document Number WFMC-TC-1011, February 1999.

[8] Web Services Human Task. (WS-HumanTask), Version 1.0, June 2007. http://www.active-endpoints.com/documents/documents/1/WS-HumanTask-v1.pdf.

[9] Martin S. Olivier, Reind P. van de Riet, and Ehud Gudes. Specifying application-level security in workflow systems. In *DEXA '98: Proceedings of the 9th International Workshop on Database and Expert Systems Applications*, pages 346–351, Washington, DC, USA, 1998. IEEE Computer Society.

[10] Mathias Kohler and Andreas Schaad. Avoiding Policy-based Deadlocks in Business Processes. In *Proceedings of the The Third International Conference on Availability, Reliability and Security, ARES 2008, pages 709-716, Technical University of Catalonia, Barcelona, Spain*, IEEE Computer Society.

[11] R4eGov Technical Annex 1. Towards e-Administration in the large, March 2006. http://www.r4egov.eu.

[12] Longhua Zhang, Gail-Joon Ahn, and Bei-Tseng Chu. A rule-based framework for role-based delegation and revocation. *ACM Transactions on Information and System Security*, 6(3):404–441, 2003.

[13] David W. Chadwick and Alexander Otenko. The PERMIS X.509 role based privilege management infrastructure. SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies, June 3-4, 2002, pages 135-140, Monterey, California, USA, ACM.

[14] Workflow Management Coalition. Workflow Security Considerations. White Paper, Document Number WFMC-TC-1019, 2001.

[15] Andreas Belokosztolszki, David M. Eyers, and Ken Moody. Policy Contexts: Controlling Information Flow in Parameterised RBAC. In *POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, page 99, Washington, DC, USA, 2003. IEEE Computer Society.

[16] Xinwen Zhang, Sejong Oh, and Ravi Sandhu. PBDM: a flexible delegation model in RBAC. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 149–157, New York, NY, USA, 2003. ACM Press.

[17] Jason Crampton and Hemanth Khambhammettu. 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, pages 174–191. Springer, 2006.