KTH Computer Science
and Communication

On-demand Restricted Delegation

A Framework for Dynamic, Context-Aware, Least-Privilege Delegation in Grids

MEHRAN AHSANT

Doctoral Thesis
Stockholm, Sweden 2009

TRITA-CSC-A 2009:01
ISSN-1653-5723 KTH School of Computer Science and Communication
ISRN-KTH/CSC/A–09/01-SE SE-100 44 Stockholm
ISBN 978-91-7415-219-7 SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen i datalogi måndagen den 16 februari 2009 klockan 13:00 i Sal F3 Flodis, Lindstedtsvägen 26, Kungl Tekniska högskolan, Stockholm..

Tryck: Universitetsservice US AB

Abstract

In grids, delegation is a key facility that can be used to authenticate and authorize requests on behalf of disconnected users. In current grid systems, delegation is either performed dynamically, in an unrestricted manner, or by a secure but static method. Unfortunately, the former compromises security and the latter cannot satisfy the requirements of dynamic grid application execution. Therefore, development of a delegation framework that enables a restricted and flexible delegation mechanism becomes increasingly urgent as grids are adopted by new communities and grow in size. The main barriers in development of such a mechanism are the requirements for dynamic execution of grid applications, which make it difficult to anticipate required access rights for completing tasks in advance.

Another significant architectural requirement in grids is federated security and trust. A considerable barrier to achieving this is cross-organizational authentication and identification. Organizations participating in Virtual Organizations (VOs) may use different security infrastructures that implement different protocols for authentication and identification; thus, there exists a need to provide an architectural mechanism for lightweight, rapid and interoperable translation of security credentials from an original format to a format understandable by recipients.

This thesis contributes the development of a delegation framework that utilizes a mechanism for determining and acquiring only required rights and credentials for completing a task, when they are needed. This is what we call an on-demand delegation framework that realizes a bottom-up delegation model and provides a just-in-time acquisition of rights for restricted and dynamic delegation.

In this thesis, we further contribute the development of a credential mapping mechanism using off-the-shelf standards and technologies. This mechanism provides support for an on-the-fly exchange of different types of security credentials used by the security mechanisms of existing grids.

Keywords: Grid Security, Restricted and Context-Aware Delegation, Delegation Protocol, On-demand Delegation, Dynamic Trust Federation, Grid Interoperability, Credential Mapping

To my beloved wife, and

my parents whom I always adore …

Acknowledgments

First and foremost, I would like to thank Jim Basney from the National Center for SuperComputing Applications, University of Illinois at Urban-Champaign, whose contributions were invaluable. I feel privileged to have worked with him and I am grateful for his support, encouragement and patience; his personal and professional insights; technical feed-back; and comments on every part of my research and my publications. I also thank him for reading and commenting this thesis.

Thanks to Professor Lennart Johnsson, my advisor, for taking me on as a graduate student and for supporting my work; for his feedback on my research and for providing the financial support that was vital to my studies.

I am very grateful to Olle Mulmo, who during my first two years as a doctoral student, worked closely with me and introduced me to the area of grid computing and security. He opened many doors for me to the grid world and helped me to establish several helpful connections with experts in this field.

I would also like to thank Åke Edlund for his support during my doctoral studies at PDC and for letting me work on this thesis while I worked in the BalticGrid-II project.

Some parts of the research presented in this thesis were done in collaboration with other researchers from different projects. In particular, I would like to show my gratitude to Mike Surridge, Thomas Leonard, Ananth Krishna, E. Rowland Watkins and Joris Claessens, all of whom worked with me in the NextGrid project; Joni Hahkala, Martijn Steenbakkers and Akos Frohner, all of whom worked with me in the EGEE project. Thanks to Adam J. Lee who contributed to one of my publications. I would also like to thank Esteban Talavera González, Masashi Nakamura and Sina Khaknezhad, who helped with implementation and integration parts of the work presented in this thesis. Finally, thanks to Rachana Ananthakrishnan and Frank Siebenlist from the Argonne National Laboratory for their feedback and support in the PURSe project.

I would like to show my gratitude to Erwin Laure and Kristaps Džonsons, who read and commented on drafts of this thesis. In particular, Kristaps, who took a lot of time and helped with the proof-reading of this thesis.

I also acknowledge the help and support in the last stages of my doctoral studies by Professor Johan Håstad, Olof Ronborg and the director of graduate studies, Professor Jens Lagergren.

I am also very grateful to my friends Ali Ghodsi and Vahid Mosavat for their supports and consultations, especially Ali for reading and commenting on this thesis.

I would also like to take this opportunity and express my profound gratitude to my parents and all of my family for their continuous support and encouragement. In particular, Soheila, who has never withheld her support from me since I have been in Sweden.

Finally, my deepest gratitude and special thank to my gorgeous wife, Azadeh Jamshidi, for her love and support, for always being with me during difficult times, for always showing endless patience during bad times, and for her continuous encouragement.

I also acknowledge the projects that provided financial support for my research presented in this thesis: the EGEE project, funded by the European Union (contract IST-2003-508833); the NextGrid project co-founded by the European Commission (contract IST-2006-034567); the BalticGrid-II project funded by the European Union (contract 223807); and the SweGrid project founded by the Swedish Research Council (contract 343-2003-953).

Contents

List of Figures

Chapter 1

Introduction

The capacity and performance of Internet communication channels are increasing immensely; therefore, it is becoming feasible to solve large-scale computational problems on a single virtual computer consisting of computers connected over the Internet. "Grid Computing", where virtual computers are composed into "grids", is a paramount approach to make this possible [47, 44].

In grids, delegation is a key facility allowing effective use of a wide range of dynamic grid applications [8]. When a grid user makes use of a remote resource to e.g. execute a job, that resource may in turn need access to third-party resources (e.g. data repositories) on behalf of the user in order to complete the task. Such access may possibly span across multiple security domains. In such scenario, delegation can be used to delegate (parts of) the user's rights to the remote resource such that it in turn can access the necessary third parties [46]. In the rest of this thesis, some terminologies in regard to "delegaiton" are used frequently for which a basic and general definition can be given as the followings:

- **Delegation** is the act of transferring rights and privileges to another party (the Delegatee).
- **Delegatee** is the delegation target. It is the entity that the Delegation Credential is delegated to.
- **Delegator** is the entity that delegates the abilities and/or rights to the Delegatee.
- **Delegation Credential** is the desired result of delegation. It is a message conveying the abilities and rights from the Delegator to the Delegatee. Depending on the security infrastructure and delegation model used, the actual syntax and contents of Delegation Credentials might be different, however they are typically integrity protected and digitally signed.

The "least privileges" principle, which states that delegation should enable a delegator to grant only those rights required for performing a delegated task, is becoming one of the most important security aspects regarding delegation. Unfortunately, there are security risks associated with performing delegation where delegated rights are not limited solely to the task intended to be performed within a limited lifetime and under restricted conditions [9, 49, 85]. In a general sense, if unrestricted delegation credentials are used, an increase of sites included in the trust domain corresponds to greater probability for security holes' occurrence. This increases potential damage brought by possible attacks and stolen credentials [50, 33, 116, 97].

In extensive environments such as grids, the probability of a host losing its trust-worthiness by some irregular actions (e.g. be compromised) is likely to be (much) higher than in a highly confined and uniformly controlled environment [49]. Therefore, performing a fine-grained delegation in a restricted and secure fashion is one of the most significant concerns of grid security, a concern not yet completely resolved.

Dynamic trust establishment and interoperability across multiple and heterogeneous organizational boundaries introduces nontrivial security architectural requirements. Grids are heterogeneous environments and therefore establishing dynamic trust relationships and using them to facilitate resource sharing becomes a challenging issue.

1.1 Problem statement

In the context of grid system delegation, there is a conflict between flexibility and security: applications must choose between limited or full delegation. On one hand, delegating a restricted set of rights reduces exposure to attack, but also limits the flexibility/dynamism of the applications; on the other hand, delegating all rights provides maximum flexibility, but increases exposure. Delegating fewer rights than are required for completing a task may fail the task execution, while delegating more rights than are needed may threaten abuse by malicious parties. Supporting restricted delegation has become a challenging issue, as dynamic execution of grid applications makes it very difficult to predict the set of rights required by a delegatee to complete the task on behalf of its delegator [9].

Furthermore, implemented delegation mechanisms are mostly tied to the underlying communication protocols and supported solely by particular security infrastructure [8]. These hinder grid system interoperability and may force grid participants to use particular security infrastructures or especially a particular authorization mechanism for their back-end.

Interoperability in grids is also a significant concern [87], and one barrier to achieve it is the heterogeneity of security protocols used for authentication and identification. Organizations participating in grids may use diverse underlying security mechanisms or different grid security infrastructures, such as Public Key Infrastructure (PKI) [29], Kerberos [80] or SAML [100]. A user may hold a security credential, one from his local domain, which can not be used in relying domains (a domain that receives an assertion from an issuing party). For example, a recipient domain may be expected to receive a Kerberos ticket when it does not support Kerberos functionality and can only understand PKI-based certificates; similarly, it may be the case that the issuing and relying domain use the same authentication mechanism, however, they are missing a trust establishment. One solution is requiring users to collect a set of security credentials in different formats and/or as issued by different trust anchors. However, this obviously is not a practical, cost-beneficial and possible solution for users and organizations.

Cross-organizational authentication is clearly a challenging issue, and a credential mapping mechanism is required to address this issue. Providing such a mechanism enables grid organizations to collaborate with verifiable and understandable security credentials regardless their locally-established security mechanisms. The lack of such a mechanism may either cause the authentication to fail at any point of interaction, such as when accessing resources, or may limit the collaborations only between those organizations with fully compatible security mechanisms.

This thesis solves these problems by defining, developing and evaluating a framework that provides support for both a restricted delegation and a credential mapping mechanism. The framework developed in this thesis enables performing fine-grained and restricted delegation in a flexible, standard and interoperable manner. It also provides support for exchanging credentials for different kinds of security tokens used by current grid security infrastructures.

1.2 Approach

In this thesis we[1] introduce a novel solution for addressing the least privileges principle of delegation in dynamic, distributed environments, which no existing solution adequately addresses. We define and develop an ***on-demand delegation*** framework, which utilizes a callback mechanism for acquiring corresponding credentials required for completing tasks on-demand. This approach addresses the shortcomings of current existing delegation mechanisms by providing restricted delegation in a secure, flexible, and interoperable fashion as needed for a wide range of dynamic grid applications.

This thesis further contributes a mechanism for a lightweight, rapid and interoperable translation of security credentials. We develop a ***Credential Mapping Service (CMS)***, which leverages a simple and standard messaging protocol for exchanging different kinds of security credentials. This greatly helps to address the issue of cross-organizational authentication and enables grid entities to collaborate with verifiable and understandable security credentials regardless of their local installed security mechanisms. The implementation of this service uses off-the-shelf standards and technologies and supports the exchange of different kinds of security

[1]The term *we* is used throughout this thesis to denote the work lead and performed by the author while collaborating with other researchers. Where there are joint contributions, the parts done by the author are explicitly stated.

tokens used by existing grid security infrastructures, such as PKI, Kerberos and SAML.

1.3 Thesis organization

This thesis is divided into two parts. The first part summarizes the background, approach and results of our work. The second part consists of thesis contribution papers and one additional paper. Chapter 2 describes underlying technologies, concepts and foundations in the context of grids. Chapter 3 describes and discusses current delegation mechanisms and their shortcomings, which this thesis addresses. Chapter 4 describes the contribution of this thesis in addressing the problem. Finally, Chapter 5 presents results, related work, future work and a short description of software components developed as part of this thesis.

Part I

Background and Results

Chapter 2

Foundations

This chapter describes the foundational concepts and theory of the work presented in this thesis. We describe grids as a new paradigm of distributed computing environments. We further describe why security is a challenging issue in the context of grids and discuss how delegation mechanisms enable dynamic execution of grid applications. We briefly explore the emerging "Semantic Web" and "Semantic Grid", and we describe how they are used in grids to achieve a high degree of easy-to-use and seamless automation. We finally describe scientific workflows and discuss the potential of scientific and grid computing for accommodating workflows.

2.1 Grids

A grid is a form of distributed computing infrastructure that involves coordinating and sharing resources across Virtual Organizations that may be dynamic and geographically distributed. A Virtual Organization (VO) is a group of individuals and/or institutions/organizations with a common purpose or interest who agree on policies for the sharing of resources in order to facilitate achieving common goals. Each VO has its own policies for access, use, management and monitoring [88, 47, 44].

2.2 Grid security

Grids are in place to enable Virtual Organizations (VOs) for cross-organizational interactions. VO members need to interact with each other. This interaction may span diverse security realms and traverse different services and hosting environments. VOs, participating members of VOs and resource owners each have their own specific rules and policies that govern how to access particular grid resources. VOs are built over heterogeneous organizations with different underlying security mechanisms. Furthermore, due to the dynamic nature of VOs there is no assumption of pre-established trust between organizations and the VOs or members of VOs.

New services may be deployed and instantiated dynamically over a VO's lifetime. There is no traditional means of security administration to update a centralized policy databases or issue new credentials to access these new services. Therefore, in a general sense, cross-organizational interactions through scalable and dynamic VOs make security a very challenging issue [88, 46, 122]. Effective management of grids therefore requires mechanisms for dynamic and robust trust establishment among VOs' participants. Delegation and credential mapping, among many others, are two significant facilities for expanding trust relationships among VOs, which are the concerns of this thesis.

2.3 Delegation

Delegation may happen anywhere in daily activities. The Cambridge dictionary[1] describes the verb "delegate" as:

> "to give a particular job, duty, right, etc. to someone else so that they do it for you"

This simple description gives the main objective of delegation as getting a task or activity done by other parties. In delegating, a person assigns the authority and responsibility to another person for performing specific activities or tasks; it may either be shifting a decision-making authority to a subordinate or re-assigning a task or activity to another person. In both cases, delegation encompasses the obligation to perform the task, authority to authorize accomplishment of the task and responsibility for the possible consequences of an accomplished task [116].

In the context of IT system, diverse models of delegation exist. As an example, batch systems for job execution read job scripts and create processes on the system on behalf of the submitter to execute the jobs. Similarly, mail client programs have permission to write mails to an inbox that is owned by an end-user.

In the context of grid security, delegation is an important facility for expanding trust relationships. It enables members of virtual organizations to endow to other entities (other members/computer programs) the ability to access and manage resources on their behalf [88, 122, 49]. In grids, resource request and use may cover extensive periods. Furthermore, requests may be generated dynamically during the execution of an application and need user approval before being submitted to a resource broker or resource provider. Resource providers usually require some form of authentication of users and authorization of requests, i.e. a user is allowed to use the requested resources as described by the request. However, the user may not be directly accessible either because of malfunction or simply by having disengaged after submission of a request for execution. A common solution to the problem of disconnection is to delegate the authorization to an agent (program) that acts on the user's/application's behalf and that is less likely to be disconnected at any time.

[1] http://dictionary.cambridge.org/

Although delegation plays an important role in expanding *trust* relationships, the level of established trust might be different in diverse contexts. In non-grid applications such as mail programs mentioned before, the system administrator authorizes the server processes to perform actions on behalf of users. This can be accomplished by running a process with more privileges than a normal user. This strategy is applicable within a single administrative domain and is possibly an effective and efficient approach to leveraging various extensions, for example, complex collections of roles, access control lists, and time-expiring access tickets. In fact, end-users of such systems *trust* local system administrators and the process with more privileges to act on their behalf. Similarly, although grid delegation is used to expand trust relationships, jobs accessing resources on behalf of end-users are running on remote sites where the end-users cannot keep control over jobs' behavior and detect malfunctions.

2.4 Ontology

In computer science, ontology is a conceptual schema to hierarchically describing and classifying entities and their relationships and rules within a certain domain. There are two kinds of ontologies, domain ontology and foundation or upper ontology. The former corresponds to a specific domain and represents the particular meanings of terms as they apply only to that domain; the latter is a model of the common objects that are generally applicable across a wide range of domain ontologies and describes general entities. In practice, a foundational ontology may be a glossary of terms used in a certain descriptive language such as a programming or modeling language. This enables creating a more specialized schema to make the data useful in making real world decisions. In this sense, ontology was described as "an explicit and formal specification of a conceptualization" by Gruber [108].

A well-known and widely used upper computational ontology is Dublin Core [58], which describes digital objects using meta-data elements such as title, creator and subject. As an example, all computer programs have a foundation ontology consisting of a processor instruction set, programming language, files in accessible file systems and so.

Recently, several ontological standards have emerged in order to describe World Wide Web content. The Semantic Web[2] is an extension of the current Web, applying explicit representation of knowledge to Web content in a manner understandable by machines. It allows more sophisticated use of the World Wide Web in order to solve problems rather than simply to retrieve information, which is its primary model of Web use today. Well-defined ontologies are required in order to formalise the relationships among web content; without them, the degree of association error rises. Semantic WEB and ontologies have strong potential in representing and reasoning over policies in the World Wide Web, distributed systems [106, 65, 107, 112, 83] and, recently, in grids.

[2]http://www.w3.org/2001/sw/

The Semantic Grid[3] is an emerging technology to add Semantic Web capabilities to the grid computing applications. Scientific processes are usually very complicated and encompass many computational steps. Each computational step may require resources from different organizations that might be represented by different models and terminologies. Resources are usually domain specific and problem dependent, and therefore an effective realization of grid computation to achieve the best effect of resources require an explicit exposure and representation of resources in a common knowledge model. Many publications have demonstrated that realization of a Semantic Grid is the key to delivering the real potential of grids and e-science [90, 36, 111].

2.5 Workflows

Workflows traditionally have been used by business organizations for modeling and controlling the execution of business processes to achieve a business objective [69]. In workflow literature, each business process can be defined and modeled as follows:

1. a collection of tasks or activities which it encompasses;
2. the applications that should perform each task; and
3. the data required for performing tasks.

A workflow separates any given organizational process into a set of well-defined tasks, related and inter-dependent, as logical steps or descriptions of pieces of work that contribute toward the accomplishment of the whole process. In this context, a Workflow Management System (WFMS) is the unit to coordinate the execution of tasks that constitute a workflow. Generally speaking, tasks in a workflow are inter-related such that initiation of one task is dependent on successful completion of another set of tasks. The order in which tasks are executed are controlled by the WFMS [14].

Recently, workflows have emerged as a paradigm for representing and managing complex distributed scientific computations and accelerating the progresses of scientific activities. For scientific applications, a workflow paradigm can be beneficial from many aspects, such as building a dynamic application for orchestrating distributed systems, utilizing resources, reducing execution costs and even promoting inter- and intra-organizational collaboration [48, 23]. Similar to business workflows, scientific workflows are also concerned with the automation of processes, and enable the composition and execution of complex scientific tasks. In scientific workflows, each step specifies a process or computation to be executed (for instance, a software program or a Web service); the workflow links the steps according to the data flow and corresponding dependencies. Processes contain tasks that are structured based on their control and data dependencies.

[3]http://www.semanticgrid.org

2.6 Summary

In this chapter, we described some of the foundational concepts on top of which this thesis has been built. Earlier, we described grids and elaborated on why security is such a challenging issue. We described the role of delegation in addressing security issues in terms of authentication and authorization, then explained how delegation can be used in expanding trust relationships in grids and why it is essential for using a wide range of dynamic grid applications. We described ontologies and how they can be used in describing hierarchical resources used by complicated scientific processes. We also described workflows as a new paradigm for representing and managing complex scientific computations. In the following chapter, we first describe how delegation is implemented and used by current existing grids. Then we discuss the shortcoming of existing approaches.

Chapter 3

Background

In this chapter, we describe the background of our work. We start by giving three examples of exiting delegation mechanisms implemented by the UNICORE[1], Globus[2] and gLite[3] grid infrastructures. Our choice of these grid middleware systems is because they are widely used and well-established, and fully implement core grid functionality. Furthermore, from the security perspective, each system approaches delegation in a different way. It should also be noted that in this chapter we only describe the original model of delegation developed by these grid infrastructures: there have been subsequent improvements and enhancement explained as related works in section 5.4.

In the remainder of this chapter, we first describe each approach individually. We further consider the pros and cons of each approach and how the shortcomings of these mechanisms can be addressed.

3.1 UNICORE delegation

UNICORE (Uniform Interface to Computing Resources) provides a ready-to-run grid system and makes distributed computing and data resources available in a seamless and secure way. UNICORE supports a task-based grid security model. In this model, when a job is created using the Job Preparation Agent (JPA), the user must specify in advance the actions to be performed, the resources needed and the system upon which the job is to run. From this job description, an Abstract Job Object (AJO) is created that instantiates the class representing UNICORE's abstract job model. AJOs can describe diverse tasks, such as computation, job management, file transfer and even some complex functions such as loops and conditionals. The AJO is signed with the end-user's certificate and then is sent to a

[1] http://www.unicore.org/
[2] http://www.globus.org/
[3] http://glite.web.cern.ch/glite/

Gateway for execution. Multi-site job execution and delegation in UNICORE is enabled by means of Sub-AJOs that are created from a parent AJO [57].

In the UNICORE security model, an "endorser" is an end-user who holds the ownership of an AJO by signing the AJO. The "consigner" is an agent that transfers the AJO to the server. It is either the end-user or another UNICORE server. "Endorsing" is the act of signing a task description by an end-user on the client site; "consigning" is the act of transferring an AJO from an end-user to a server or from a server to another server. An end-user is allowed to do both "endorsing" and "consigning" of AJOs, whereas a server can only "consign" AJOs. This means that server processes cannot create sub-AJOs dynamically. Servers do not have the ability to create grid processes dynamically, since all components of an AJO have to be pre-signed with the users' certificate. In UNICORE, all the AJOs received by sites must have been created by end-users and cannot be modified by intermediary servers. Once an AJO is received, the server checks that the endorser has properly signed the AJO. Following this, the server performs local site authorization, ensuring that the endorser is allowed to execute jobs on the site and is mapped to a local account.

This approach requires a minimal trust relationship between the sites (since the end-user has explicitly authorized the file transfer request). There is, however, no support for dynamic grid functions (e.g., reading a database from a source whose location is only discovered during execution) since the entire job tree must be signed by the end-user at the client when the job is created [57, 49].

3.2 GSI delegation

The Globus Security Infrastructure (GSI) [46] is based on the Grid Security Architecture introduced in [79]. The GSI is used to manage mutual authentication between the local and remote machine, and authorization on the remote systems. The GSI infrastructure performs delegation by means of "proxy certificates" [56]. A proxy certificate is defined as an X.509 certificate [110] issued by an end-user to a process or another entity acting on the end-user's behalf. Remote sites with support for GSI can interpret a proxy certificate as an authority of owner to perform a task on behalf of an end-user. In the GSI delegation approach, the owner of proxy certificate will be granted all rights that the end-user possesses. Proxy certificate holders further can issue other proxy certificates to other entities [121].

GSI provides delegation capability as an extension of the standard TLS protocol [40]. A proxy certificate still includes the identity of its end-user: this allows establishing SSL connections with remote sites. GSI binds a temporary private key to each proxy certificate for performing authentication. In order to support Single Sign-On [46] the private key associated with the proxy certificate is typically stored unencrypted, protected through only normal file-system mechanisms. The private key cannot be encrypted, as it must be used by a delegatee (processes that act on behalf of the end-user) without the need for entering a password by the end-user [121].

In fact, what a GSI proxy mechanism provides through a simple implementation of proxy certificates is a full impersonation of end-users by granting all rights to a subordinate for performing tasks. In the Globus approach, in order to establish the required trust relationships, the possession of a proxy certificate is sufficient to authorize work at remote sites. Any site accepting GSI delegations must trust that the originating site and all other sites in the delegation chain are properly managed and not compromised. The GSI approach imposes a transitive trust requirement on all sites participating in a particular grid (members of a particular VO). Thus the delegation protocol needs to be carefully designed because the entities (both end-users and processes) issuing proxy certificates must be careful to authenticate the identity of the process requesting the proxy so that delegated rights are given to the correct process.

3.3 gLite delegation

gLite [4] is a grid middleware system providing a framework for building Internet-wide grid applications. gLite is the largest grid computing middleware to date and is developed as part of the EGEE Project[5] [68].

gLite describes delegation as a standalone Web Services portType and provides ready-to-use library implementations of this portType. Service implementers do not need to deal with the details of the delegation mechanisms and can factor this functionality out of the internal application logic. Delegation can also be instantiated as a standalone service, front-ending a (trusted) credential repository from which the target service can extract delegated credentials in a controlled and secure manner.

The java-delegation security component of gLite [59], implements delegation by leveraging proxy certificates [110] and using a simple request-response interaction protocol between a Delegatee and a Delegator. The gLite delegation protocol was originally based on the G-HTTPS protocol described in [78], which is also used in building the delegation interface of GridSite[6], the Grid Security for the Web platforms for Grids.

In the gLite delegation components, the operations "get-proxy-request" and "put-proxy-cert" respectively implement the request and the response interactions of the delegation protocol, which are described in the followings:

1. GET-PROXY-REQUEST: A Delegator may prompt a Delegatee to perform delegation. Delegator uses the GET-PROXY-REQUEST method allowing a Delegatee to prompt a Delegator to generate an X.509 certificate request and key, and then return the certificate request to the Delegator. This request

[4]http://glite.web.cern.ch/glite/
[5]http://egee1.eu-egee.org/
[6]http://www.gridsite.org/

includes a Delegation-ID, an alphanumeric string used to identify the resulting delegation session. The Delegation-ID is used to distinguish between several delegations from the same user, using several attributes and expiration times. The server then generates an X.509 certificate request and private key, and stores the private key for retrieval using the Delegation-ID.

2. PUT-PROXY-CERT: This method is used to return the proxy certificate to a Delegatee after receiving a successful response to a GET-PROXY-REQUEST by a Delegator. The request may include a Delegation-ID header with the same delegation ID header as sent to the server in the GET-PROXY-REQUEST. Before responding, the server stores the signed certificate and chain associated with the private key the server has generated, the client's GSI identity and the Delegation ID if present.

gLite delegation provides dynamic delegation due to the fact that it uses proxy certificates as the basic means when performing delegation. This enables a grid user to delegate all (or some subset) of their privileges to another entity in a limited-time interval as described in Section 3.2. It is also dynamic in terms of involving new entities in a delegation process: in addition to persistent services and entities, delegation of privileges can be provided to services that are created dynamically or do not hold any form of identity credential. A common scenario is that a user submits a job to a computational resource and wants to delegate privileges to the job to allow that job access to other resources on the user's behalf (e.g., to access data belonging to the user on other resources or start sub-jobs on other resources). However, as described for the GSI delegation mechanism, this protocol still fails in providing a restricted delegation that can fulfill the requirements of the least privileges principle.

3.4 Discussion on current delegation approaches

We subsequently identify and address four significant concerns of delegation as the followings:

Security vs. flexibility

One the most significant concerns of current delegation mechanisms is their shortcomings in flexibly and dynamically addressing restricted and secure delegation. In general, Globus approaches a very dynamic delegation mechanism by fullly impersonating end-users and granting all rights to subordinates for performing a task. Unfortunately, an unencrypted private key associated to a proxy certificate yields a GSI proxy certificate that is valuable and relatively easy to acquire and consequently abuse.

The GSI and gLite solutions, which use proxy certificates, implement a very dynamic delegation mechanism because there is no need to know in advance execution details or resources required. However, it increases the danger of unauthorized acquisition or usage of proxy certificates. Issuing short-lived proxy certificates has been the basic and primary solution used by GSI for addressing this shortcoming. However, even a short-lived proxy certificate not restricted to a specific task can cause serious dangers to security systems. A complementary approach, limiting the usage of X.509 proxy certificates, is using the Proxy Certificate Information (PCI) extension [56]. This extension enables issuers to express their expected delegated rights and to limit the number of proxy certificates that can be further issued by a Proxy Certificate holder.

In the UNICORE security model, delegated rights are restricted solely to the task for which delegation has been launched. Although the description allows some flexibility (e.g., conditional execution of parts of the job and a generic request to run anywhere suitable), there is insufficient flexibility to adapt this mechanism to the changing circumstances as required by many grid applications [49, 87, 57].

Thus, there is a conflict between addressing the flexibility and security in the context of delegation. This conflict originates in the fact that execution of grid application is highly dynamic: anticipating required access rights in advance and adapting them to policy statements for accessing resources during task execution is not straight-forward [9].

Dependency to communication protocol

In current delegation mechanisms, delegation credentials have been closely tied to the underlying authentication and communication protocols, e.g., as an optional step performed after a TLS handshake [40] as done in the first incarnations of GSI [46] or in UPL (UNICORE Protocol Layer) [57]. However, this may break the compatibility with some communication protocols such as TLS and subsequently any protocol on top of TLS, such as HTTPS [89]. Hence, a delegation framework must separate delegation from authentication at least for non-legacy software components [8]. A well-established delegation framework can make this possible by defining and implementing a stand-alone Web Services delegation portType. For this service, required supports can be embedded in the service container/application server (as a separate and standalone service), or in the application itself (by inclusion of the delegation's portType in the service description and helper libraries that implement the operations exposed by that portType) [8].

Interoperability

Current security mechanisms used by existing grids are not completely interoperable; despite that interoperability is becoming one of the most significant concerns of grid security [87]. Delegation is implemented and used in different ways by diverse grid middleware systems [34, 121, 49, 59]. Organizations may also use different technologies and security infrastructures (e.g. Kerberos [80], PKI [29] or SAML [100, 120]). Thus, assuming that current in-place security mechanisms will continue to be used, different delegation mechanisms implemented by different grid systems need to interoperate.

Observing and auditing

A robust delegation mechanism should enable keeping track of delegated tasks and accurate usage of delegated rights. By approaching current delegation mechanisms, delegators can easily lose their control over delegated rights during the task execution; further, misbehavior can not be detected immediately [49, 9]. Therefore, providing a means of enabling extensive and fine-grained monitoring and logging of delegation credentials is an essential need that is not addressed by existing mechanisms.

3.5 Summary

In this chapter, we reviewed the delegation mechanism developed and deployed by UNICORE, Globus and gLite, to date the biggest and the most widely-used grid systems. We described the shortcomings of their delegation mechanisms and discussed some of the most important issues in terms of security, flexibility, dependability and interoperability. Our discussion was aimed at making a strong context for the following chapter, in which we describe the contribution of this thesis in addressing the shortcomings of other approaches.

Chapter 4

Thesis contribution

This thesis contributes a novel delegation framework for grid environments that addresses all of the shortcomings described in Chapter 3. By leveraging the foundations described in Chapter 2, we have built a delegation framework that not only meets the requirements of the least privileges principle, but also enables performing delegation in a standard, secure and flexible manner.

4.1 On-demand delegation framework

On-demand delegation is a novel solution providing a secure, restricted and fine-grained delegation mechanism fitted for a wide range of dynamic grid applications. This solution provides a delegation framework in which a delegatee, in order to act on behalf of a delegator, needs to obtain required rights (in terms of additional delegated credentials) at run-time. On-demand delegation allows delegators to delegate privileges when the other party has proved the necessary need for those privileges. This implies delegating rights to delegatees iteratively as needed until the task has completed. Papers I and II give a detailed description of this approach.

The intuition of the on-demand delegation model is a novel concept of delegation that we call bottom-up delegation. It is well explained when compared to a traditional model of delegation in which a top-down model is approached for delegating rights from a superior to a subordinate in advance before a delegatee starts off a delegated task. In contrast, in bottom-up delegation a subordinate needs to ask a superior for acquisition of sufficient rights when it needs to perform an action on behalf of a delegator. Although top-down delegation models have been used for a quite long time by many security systems, for some use-cases, like those described earlier, they are insufficient and there is a need for a bottom-up model that enables delegating rights just-in-time and in response to a valid request [9].

Figure 4.1 illustrates a simple example of performing on-demand delegation in a single organization. As depicted in this figure, the Delegator initiates delegation by providing a minimum set of rights to the Delegatee (Step 1). This minimum set

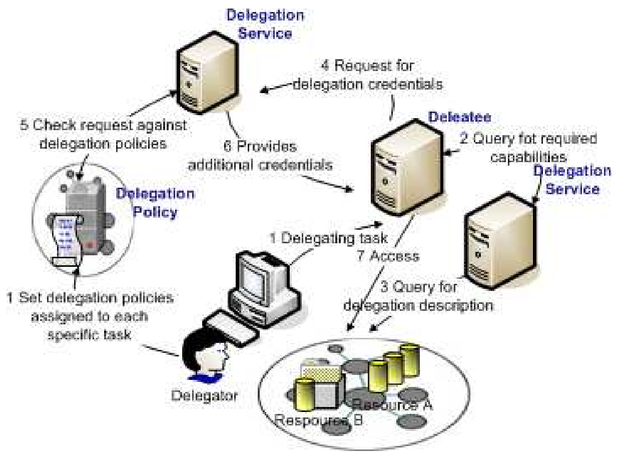
Delegation Service
4 Request for delegation credentials
Deleatee
2 Query fot required capabilities
Delegation Service
5 Check request against delegation policies
6 Provides additional credentials
Delegation Policy
1 Delegating task
7 Access
3 Query for delegation description
1 Set delegation policies assigned to each specific task
Delegator
Resource A
Respoures B

Figure 4.1: An example of on-demand delegation

only allows the Delegatee to prove that it needs to act on behalf of the Delegator. The Delegatee has no rights to access resources and therefore needs to query the delegation system model to determine what additional credentials are required to complete the task. By this query, the required privileges for executing the task are determined and disclosed to the Delegatee during the job's execution (Steps 2 and 3). The Delegatee contacts one or more Delegation Services to determine what credentials are needed and then obtains them (Step 4). In this simplest case, there is one Delegation Service associated with the Delegator and one associated with the Resource. The Delegation Service checks requests against a set of policies established for each specific delegatee that specify the circumstances for the issuance of additional credentials (Steps 5). Hidden from this picture is the process of establishing contexts. The context establishment is the process to collect all information required to assert a delegation request. These information are converted into a single internal format understandable by the Policy Engine. The Policy Engine uses a combination of delegation policies and context information to approve any delegation request: if the policies allow issuance of the requested credentials, the Delegation Service generates the credential and sends it to the Delegatee (Step 6). The Delegatee now has all required delegated credentials to access resources on behalf of the Delgegator (Step 7). A concrete example of a real grid usage has been given in Paper I Section V and Paper II section III.

On-demand delegation requires a delegatee to contact a superior in order to ask for additional rights. This gets more parties involved in the delegation process, which raises some requirements for this model as follows.

- First, there is a need for setting up a process or a service (i.e. a delegation service) that can automatically delegate a superior's rights to a delegatee without getting the superior involved.

- A delegation service is a machine and not a human being; it is, therefore, crucial to set up a policy engine that enables the service to decide if the subordinate should have the requested rights. Such established policies are essential for enabling a superior to maintain complete control over delegation.
- It is important to resolve how, in such framework, a delegation request can be approved and how a subordinate can be trusted to the truth about which rights is required and why.
- It is important to figure out how to determine required access rights and credentials when accessing resources at run-time, and how to request and obtain them from a delegation service.
- Finally, this approach needs to be reliable for delegators such that a delegation service never delegates more rights than required. It should also have the potential to optimize delegation, reducing the number of callbacks when performance is a concern and too many callbacks are required for completion of a task.

Building blocks

In the section, we describe the building blocks of an on-demand delegation framework and explain how they are used in addressing the above requirements. A detailed description can be found in Papers II, IV and V.

- **Delegation service:** A delegation service is a web service for delegating credentials to potential requesters (delegatees). It is run by a user or hosted on behalf of many users. A delegation services can delegate credentials based on a given delegation specification attached to the request. Along with the request, a delegation service requires the context information in which additional delegated rights are requested. It must also query the policy engine to check if delegation policies can be fulfilled in a particular context, permitting the delegation of additional rights. A delegation service is mainly the architectural component to address the first requirement mentioned above. Papers I and II describe in detail how a delegation service is used in an on-demand delegation framework.
- **Context manager:** A context manager is a service that is either notified by other services with new context information or that makes a query to directly obtain the context information of particular services, resources or job instances. This context information, along with the delegation request, is then sent to (or directly requested by) a delegation service in order to create an appropriate context in which a delegation request may be verified. Context information may be created to verify a delegation request that is made to

obtain additional rights for the sub-jobs created by a parent job. A context manager would typically be hosted within the service from which it obtains the information.

Contexts are established by monitoring, collecting and processing the status of jobs, resources and services. They evolve through the task life-cycle and are considered as any characterizing information about protected resources and surrounding environments. Contexts are basically associated with: i) resources to be controlled; ii) delegatees who make requests to access protected resources; and iii) delegators on whose behalf access to resources are made. Paper II describes in detail how "contexts" are created, established and leveraged in an on-demand delegation framework. A context manager is mainly used to address the second and the third requirements mentioned above.

- **Policy engine:** A policy engine is required to set up delegation policies. It is used by delegation services to check a delegation request against a set of established delegation policies specified by local administrators and/or delegators. A policy engine is mainly used to address the second and the third requirements described above. Papers I and II explain in detail how a policy engine is used in an on-demand delegation framework.

- **Credential exchanging protocol:** An on-demand delegation framework requires a protocol for requesting and receiving delegation credentials as needed. The callback mechanism leveraged by this framework requires a flexible and efficient protocol enabling the dynamic exchange of delegation credentials between delegatees and delegation services over heterogeneous security domains. The Grid Delegation Protocol (GrDP) provides such communication protocol for exchanging delegation of rights and abilities in terms of delegation credentials. The design of GrDP is based on the WS-Trust [126] specification which is not bound to a particular security mechanism and it is therefore applicable for exchanging different kinds of security tokens of common use supported by grid environments. Papers I and IV describe in detail how this protocol is defined, developed and deployed for an on-demand delegation framework.

- **Delegation ontology:** Delegation ontology is used to describe how "delegation" allows access to resources and how delegated credentials can be provided upon receiving a delegation request [9, 71]. Ontologies are important to on-demand delegation in the following general aspects:

  - to be used in *system description*, which enables both fine- and coarse-grained authorization mechanism to protect resources; and
  - to be used in dynamic establishment of *delegation policies*, which protect the exposure of delegated rights and privileges.

  Delegation ontology describes that each "Delegation" enables a "Delegator" to endow a "Capability" to a "Subject" under restricted conditions. "Credential" is the means by which "Delegation" is authorized. Each "Capability" contains one or more "Verbs", which can be accomplished on one or more "Objects". Each "Capability" may have some dependencies on other capabilities that implies a hierarchical delegation taxonomy in system description and consequently the need for additional delegation credentials.

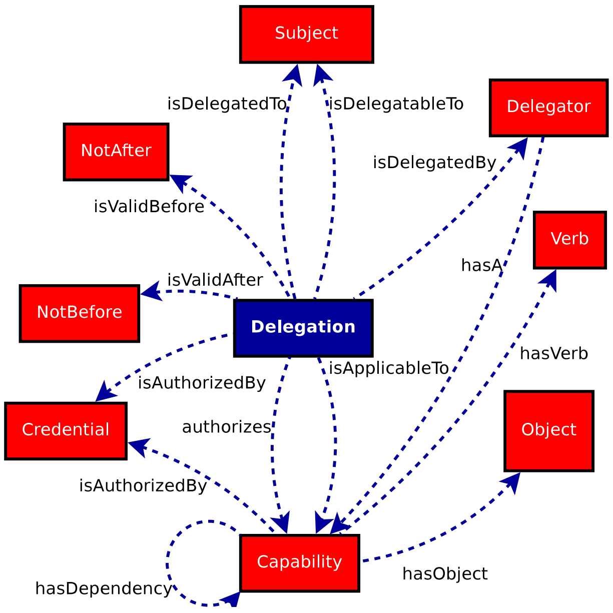

Figure 4.2: Delegation ontology

Figure 4.2 is a simple illustration of delegation ontology. This picture only illustrates the most important and relevant classes defined for delegation ontology without depicting the arrangement of these classes in a taxonomic hierarchy. It also shows defined properties and allowed values for these properties. Hidden from this picture are the values for these properties that have been filled in for instances and a knowledge base created by defining instances of these classes. On the service provider side, instances of the delegation ontology can also be used to describe the system in terms of "Objects", "Verbs" and the "Capability" that makes an appropriate relation between objects and verbs. It may also be used to specify individual instances of a "Delegator", who delegates access rights to the instances of class "Subject". Individual instances of class "Credential" can also be used to describe how the authority of the owner for accessing resources can be approved. On the Delegator side, individual instances of delegation can be used for establishing delegation policies and further specifying how credentials should be issued and appropriate constraints be applied on them. Figure 4.3 is a detailed and hierarchical illustration of all asserted classes defined for delegation ontology.

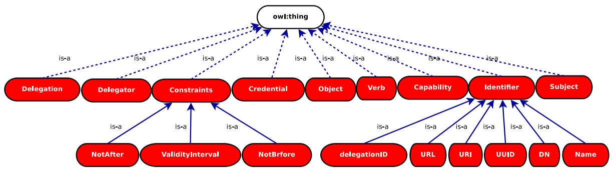

Figure 4.3: All asserted classes of delegation ontology

Delegation ontology is mainly used to address the forth requirement mentioned above. Paper I describes in detail how delegation ontology is defined and leveraged in an on-demand delegation framework. Appendix I depicts the delegation ontology deployed for this framework in OWL/XML notation.

- **Workflows:** A workflow gives a high-level and structural specification of processes used by Work Flow Management Systems (WFMS) to define the way that tasks should be ordered, scheduled and executed. The benefits of using workflow management systems in an on-demand delegation model are their support for defining, structuring, executing and controlling processes (job definitions), which encompass tasks. The task dependency, process structure and execution path requirements of a WFMS are what a bottom-up delegation model leverages to ensure that delegation is performed in a "reliable" and "optimized" way.

  To achieve a reliable delegation we introduce the notation of Delegation Safety Invariant (DSI) to assuring the following: i) privileges can be delegated either at the time of launching tasks or during the execution (just-in-time), ii) no privileges can be delegated from a superior to a subordinate unless they are restricted to an intended delegated task (bound-to-task), and iii) no privileges can be delegated for accessing a resource unless they are less/equally powerful as the privileges held by the superior for accessing the same resource or executing the same task (limited-to-boundaries).

  To achieve an optimized delegation we introduce the Risk of Delegation (RoD) parameter that can be set by security administrators or individual delegators to specify the level of security risks and threats when delegating rights. It gives the ability to adjust risk factors based on the current threat landscape and enforces the security model to operate accordingly. A higher value for this parameter indicates a higher level of security risks and misbehavior when delegating rights; a lower value implies less security risks and threats associated with delegation. The RoD parameter controls how restricted privileges can be delegated to potential delegatees. It also determines how often during a workflow execution a delegatee is required to ask (via making call-backs) for more rights. For example, if delegation is performed in safe and restricted environments to highly trusted delegatees, the value of RoD can be set to a lower level which allows providing more rights via a fewer number of callbacks; and in case of stronger potential of security threats and risks, a lower value can be set to the RoD parameter to enforce using a delegation model which requires a larger number of callbacks and more restricted range of delegated rights. This enables performing delegation in different level of efficiency and restrictions. In Paper III, based on the concept of ROD parameter we present a formal definition and develop appropriate algorithms for three different models of bottom-up delegation.

Figure 4.4 decouples the on-demand delegation framework into its architectural components described earlier. It also explains how these components are related and should interact with each other. What Figure 4.4 illustrates are the components of an on-demand delegation model in a generic grid infrastructure. The generic architectural components (white boxes) illustrated in this picture can be provided by many of currently-used grid systems; the components represented by gray boxes are what on-demand delegation provides to a grid infrastructure. In Paper II, we give a concrete design of this generic architecture for a real grid system and we describe how the components with generic names are replaced by the real components in Globus which is one of the most widely used grid infrastructures.

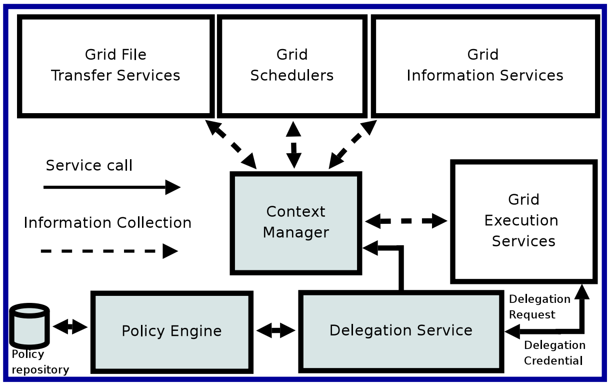

Figure 4.4: Architectural components of on-demand delegation framework

On-demand delegation strengths

The strengths of this approach, particularly those that address the shortcomings described earlier in Chapter 3 and those that fulfill the requirements explained above, are as follows:

- *Fine-grained delegation*

  This framework provides support for generating delegation credentials with a very limited and well-defined range of capabilities or policies. A delegator is able to implicitly or explicitly entitle a delegatee solely a set of restricted and limited rights for the task to be performed.

- *Dynamic determination and provisioning of required rights*

  On-demand delegation mechanisms enable dynamic determination of required rights and utilizes a callback mechanism for requesting and acquiring additional rights at run-time. Thus, there is no need to know about required rights in advance.

- *Context-aware delegation*

  On-demand delegation framework is capable of establishing "contexts" in which the need for additional delegated rights can be verified. Supporting contexts addresses the requirements of active security systems: in active security systems, any assigned permission might be activated or deactivated when its associated context is evolving. Operations are then permitted if the associated permission is currently activated. In such security systems, access control models can distinguish between permission assignment and activation by considering different levels of "context" when processing an access operation on an object [27, 52, 55, 103].

- *Uniformly performing delegation*

  On-demand delegation framework provides a protocol for requesting and exchanging delegation credentials when they are needed by delegatees to complete tasks. The GrDP protocol implements a flexible security token exchange protocol enabling requester and delegation services to interact dynamically over heterogeneous security domains[1].

- *Exploitation of delegation Ontology*

  The resource description and associated delegation specification are utilized for dynamic determination of access rights and delegation requirements when accessing resources. In order to support an automatic process for establishing delegation policies and dynamic disclosure of required access rights, this framework provides a formal and abstract description of the concept of delegation that can be instantiated for any administrative domain.

- *Integration with workflows*

[1]In 2005 and in order to harmonize delegation between different grid middlewares and projects we established a "Task Force" group and invited partners from different projects and organizations who have been experiencing delegation in their products such as: Globus, EGEE and GridSite. The preliminary goal set to gain a consensus on a single set of syntax and semantics for delegation based on the WS-Trust specification and creating a strawman document for discussion in a wider community/standard body. In order to solicit comments from the grid security community the idea was presented at the thirteenth Global Grid Forum (GGF13) in the WS-Delegation AdHoc BoF session.

Using workflows in an on-demand delegation framework ensures to meet the requirements of DSI factor and increases the level of reliability of this approach in a sense that delegated privileges are not more than required for executing a process (job) and delegation is always bound to intended job(s). Using workflows we also presented three different models of bottom-up delegation: Per-step delegation, Multi-step delegation and One-Step delegation. These models provide delegation in different level of efficiency and restriction based on the current threat landscape specified by the RoD parameter described earlier.

- Observability and monitoring

  In on-demand delegation frameworks, a delegator must be able to establish relevant delegation policies at the time of initiating tasks. These policies govern if the request for additional rights and privileges can be granted to a delegatee (specific task/program) at run-time. This enables a delegator to keep full control over delegated rights and privileges during the task execution and to take appropriate action in response to any potential misbehavior or unexpected situation.

4.2 Security credential mapping

We identified cross-organizational authentication and identification as a significant and challenging issue in grids: members participating in different VOs need to interact with each other. The requests for accessing grid services may cross organizational boundaries with different underlying security models. One part of this thesis contributes a credential mapping mechanism that enables grid organizations to collaborate with verifiable and understandable security credentials regardless of their locally-established security mechanisms.

This thesis addresses this problem with a mechanism for on-the-fly exchange of different types of security credential profiled by OASIS[2], including Kerberos [80] and X.509 proxy certificates [110]. Our work presented in this thesis is the only *full-mesh* and *standard* credential mapping mechanism that has been developed as a light-weight, easy-to-integrate and open-source service for grids. In this work, we only deal with authentication token mapping that is in general about the syntax aspects of security credentials. Other aspects of "cross-organizational" interoperability, such as authorization and attribute mapping, are covered in collaboration with other researchers as it is described in Paper VI .

Previously, we mentioned the development of a standalone delegation service for issuing security tokens as defined by the WS-Trust specification. Using such a standard and flexible interface as proposed by the WS-Trust specification, and adding some more functionality, we have built a service capable of exchanging security tokens for different formats or syntax. In this case, the service will function

[2]http://www.oasis-open.org/specs/

as a Credential Mapping Service (CMS) in our framework and can be used to address the cross-organizational authentication challenge described earlier. As a usage example, if a user holds a Kerberos ticket asserting his identity and needs to be authenticated by a target service that needs X.509 certificates for authentication, the Kerberos ticket can be presented to CMS, which then issues the holder with an X.509 certificate asserting the same identity in a different format. CMS developed in our framework provides a full translation between security credentials common in grids in addition to those profiled by the OASIS standard body[3]. This includes UsernameTokens [77], X.509 Certificates [56], SAML Assertions [100], Kerberos tickets [80] and X.509 proxy certificates [110].

Figure 4.5 depicts a design diagram of a general mapping scenario using the CMS service. In this framework, we assume that a prospective user, who is willing to request CMS for exchanging a security token, holds a valid credential obtained from his local identity provider. In follows, we describe the steps of this mapping as depicted in Figure 4.5:

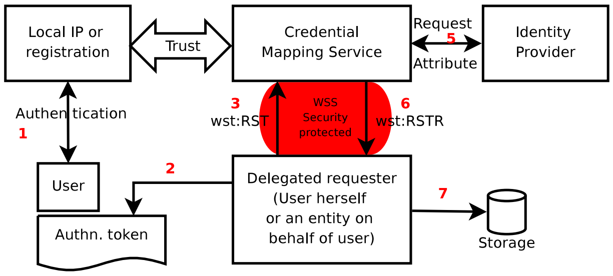

Figure 4.5: A general use-case of exchanging security tokens using CMS

1. The User authenticates with his local identity provider in a domain in which he is already registered. As the result of this authentication, the User obtains an authentication token from the local identity provider (IdP). This is basically a native authenticator token generated by the local authentication mechanism.

2. The requester (the user or an entity on behalf of the user) generates a request message. Depending on the exchanging scenario requested, the requester attaches all the “Claims” required by CMS.

3. The requester signs, encrypts and sends the message to CMS. How a request is signed and encrypted may differ for each particular mapping scenario and the way that the trust relationship is established between the local IdP and the CMS. We elaborate this when we describe each particular scenario separately.

[3]www.oasis-open.org/

4. CMS receives the message, verifies the signature and decrypts it. If the verification is successful, it checks the request against existing policies and checks if all required "Claims" are attached to the request. It then uses an appropriate mapping mechanism (a plug-in) to generate the requested credentials.
5. If CMS is already configured to use a third-party attribute provider, it requests for a set of attributes to be incorporated in a newly-generated security credential.
6. The generated credential then will be packed, encrypted, signed and sent back to the requester.
7. The requester receives the response message, verifies the CMS's signature, decrypts the message, extracts the generated credential from the message and stores that in a configured credential storage.

In this section, we briefly described a generic model of credential mapping architecture. Papers V and VI give a detailed explanation and design diagram for each particular mapping scenarios. They further describe some real grid usage scenarios in which credential mapping has been used.

4.3 Summary

In this chapter, we briefly described the contribution of this thesis. We introduced an on-demand delegation framework that provides a novel approach for performing delegation in grid environments. We described the building blocks, components, technologies and protocols for achieving such a framework and explained how on-demand delegation addresses the shortcomings of existing delegation mechanisms. An implementation of this framework for a real grid usage scenario based on the Globus environment is fully described in Paper II. We further described our contribution in providing a credential mapping mechanism developed for addressing the cross-organizational grid authentication.

Part II of this thesis includes a number of papers that give a more detailed explanation of our approach. In regard to building an on-demand delegation framework more information can be found from Papers I, II, III and IV. More elaboration on credential mapping mechanism can be found from Papers V, VI and VII.

Chapter 5

Results

In this chapter, we present a summary of the papers contributed in this thesis. We also give a short description of the software components developed and implemented for building the framework described earlier. In brief, Papers I, II, III, and IV give a detailed description of the on-demand delegation framework and Papers V, VI and VII give a detailed description of the credential mapping mechanism and its usage in some real grid scenarios. In this chapter, we further summarize related works and give a conclusion of this thesis and highlight some works for future studies.

5.1 Summary of thesis contribution papers

Paper I : Toward An On-demand Restricted Delegation Mechanism for Grids

This conference paper[1] identifies the shortcomings of current delegation mechanisms in grids and highlights the existing conflict in addressing restricted delegation in current grid security systems. It further proposes to approach delegation on-demand by leveraging a call-back mechanism, in order to achieve a secure and flexible restricted delegation in grids. The paper presents the concept of delegation ontology and describes the potentials of on-demand delegation framework in using a delegation ontology.

Paper II : Context-Aware, Least-Privilege Grid Delegation

This conference paper[2] recognizes the need of supporting "contexts" for establishing delegation policies in an on-demand delegation framework. In this paper, we

[1]This paper was published in Proceedings of the 7th IEEE/ACM International Conference on Grid Computing, Barcelona, September, 2006.

[2]This paper was published in Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, Austin, Texas, September, 2007.

introduce the concept of "active delegation" in our framework for enabling a just-in-time acquisition of delegated rights in an "associated context". We describe the architectural components of the on-demand delegation framework and explain the implementation and integration of this framework in a real grid usage scenario based on the Globus grid infrastructure.

Paper III : Workflows in Dynamic and Restricted Delegation

This conference paper[3] describes the integration of workflows with an on-demand delegation framework. In this paper, we describe the development of three different models of bottom-up delegation as: One-step, Multi-step and Per-step delegation that enable delegating rights in different level of efficiency and restriction based on the current threat landscape indicated by the Risk of Delegation (RoD) parameter. The notation of Delegation Safety Invariant (DSI) is also presented in this paper. This paper uses some formal notations of a standard RBAC authorization model and a graph-based workflow model to define, analyze and evaluate these delegation models.

Paper IV : Grid Delegation Protocol

This paper[4] highlights the lack of a standard delegation interface and further proposes a standard and interoperable protocol for performing delegation in grids. The GrDP protocol introduced by this paper, approaches delegation in grids through a standard and interoperable way based on the WS-Trust specification. This paper describes how the GrDP protocol can operate across diverse security realms and organizations for performing delegation in an interoperable and independent manner.

Paper V : Security Credential Mapping in Grids

This conference paper[5] describes the development of a credential mapping mechanism for grids. The paper highlights the significant challenge of cross-organizational identification and authentication in girds as one of the main barriers to dynamic trust federation over short time scales. The paper describes the architecture and development of a credential mapping mechanism, based on off-the-shelf standard specifications and technologies. The paper describes the design diagram and the implementation of this framework for different credential mapping scenarios.

[3]This paper will be published in Proceedings of the 4th International Conference on Availability, Reliability and Security (ARES/CISIS 2009), Fukuoka, Japan, March 2009.

[4]This paper was published in the Proceedings of the Workshop on Grid Security Practice and Experience (UK e-Science Security Task Force), Oxford, July, 2004.

[5]This paper will be published in Proceedings of the 4th International Conference on Availability, Reliability and Security (ARES/CISIS 2009), Fukuoka, Japan, March 2009.

Paper VI : Dynamic Trust Federation in Grids

This conference paper[6] investigates on trust federation and mapping mechanisms for managing aggregated security and trust relationships in dynamic virtual organizations. The work presented in this paper describes a joint collaboration with other researchers to design and develop a complete solution for achieving a dynamic trust federation in grids. Our contribution to the work presented in this paper has been carrying out a state-of-the-art analysis of trust federation and credential mapping mechanisms and their applicability to the architecture proposed for the next generation of grids. In this work we integrated our credential mapping mechanism with a dynamic authorization framework, contributed by the co-authors, for enabling dynamic federation of resources based on a short-term, rapidly formed business relationship.

5.2 Summary of other papers

Paper VII : Streamlining Grid Operations: Definition and Deployment of a Portal-based User Registration Service

This journal paper[7] describes PURSe, a Portal-based User Registration System (PURSe) and credential management tool for grid users. The tool is aimed at providing end-users with an easy-to-use, semi-automatic, yet secure way of obtaining and managing credentials necessary to access grid resources. The tool is implemented as a set of highly customizable components suitable for portal integration. Our contribution to the work presented in this paper has been the improvement of its security design and development of corresponding software components that enable PURSe to support external Certification Authorities at back-end. We have also extended the functionality of PURSe by integrating this portal with our Credential Mapping Service (CMS)[8].

5.3 Software

For both the on-demand delegation framework and the credential mapping mechanism contributed in this thesis, we have provided a detailed architectural design and a prototype implementation of appropriate software components to demonstrate

[6]This paper was published in Proceedings of The 4th International Conference on Trust Management, Italy, May, 2005.

[7]The first edition of this paper was published in Proceeding of the Workshop on Grid Applications: from Early Adopters to Mainstream Users In conjunction with GGF14, (June 27, Chicago, USA). The second edition of this paper was published in Journal of Grid Computing, Volume 4, Number 2 / June, 2006.

[8]The development of software components which integrate PURSe with CMS was accomplished in Google Summer of Code 2008 program (GSoC 2008) and the result are available from the Google Code web site.

the feasibility of our approach. The followings are the most important software components developed for these frameworks:

- **Delegation Service** is a kind of WS-Trust security token service [76, 126] that can be used by any hosting domain in on-demand delegation framework for providing delegation credentials to a potential requester. Upon receiving a request a delegation service checks the validity of request again established delegation policies and issue delegation credentials based on a given set of delegation specifications. Depending on received delegation request and type of requested delegated credential the delegation service may use different mechanisms at back-end to obtain requested credentials.

- **Grid Delegation Protocol (GrDP)** is a delegation protocol to describe delegation as a standalone Web Services portType based on the WS-Trust specification. A set of ready-to-use library implementations of this portType were also developed to enable service implementers to support delegation without a need for understanding implementation details. This helps to factor out delegation functionality from the applications. Furthermore, GrDP instantiates delegation as a standalone service, frontending a (trusted) credential repository, from which the target service can extract the delegated credential in a controlled and secure manner.

- **Ontology-based software components** are a set of Java libraries developed for our on-demand delegation framework by using the Protégé-OWL API [26, 117]. These software components are used for both: populating and instantiating delegation ontology and generating, parsing and evaluating ontological queries dynamically.

- **Credential Mapping Service (CMS)**, is a Web Service that issues security tokens as defined by the WS-Trust specification. This service can be used to exchange a security token which is not in a format or syntax understandable by recipient domains. For example, if a user holds a Kerberos ticket asserting their identity, but the target service needs an X.509 certificate, the Kerberos ticket can be presented to CMS, which will issue the holder with an equivalent X.509 certificate asserting the same identity. The CMS implementation shares the same libraries with delegation service as both of these services are using the same interface.

- **PURSe** provides an end-user portal for credential management in grids and it is currently being used as an end-user portal for SweGrid[9] and the Earth System Grids (ESG)[10]. We have contributed in development of PURSe by providing a set of software components that improve the security and functionality of this portal. Our contributed software components enables PURSe

[9]http://www.swegrid.se/
[10]http://www.earthsystemgrid.org

to obtain certificates trough a secure communicate with an external Certification Authority. They also make PURSs to interacting with a credential mapping service (CMS) for generating different types of security tokens [45, 10].

- **gLite-java-delegation** is a set of Java libraries that provide support for performing delegation within the context of gLite grid middleware[11] as described in Section 3.3. The gLite-java-delegation component consists of three subcomponents. The gLite-delegation-interface, defines a delegation interface based on the G-HTTPS request-response interaction protocol as described in Section 3.3. The gLite-delegation-java provides a collection of Java libraries to perform delegation. These APIs not only implement the gLite delegation interface, but also provides developer with a set of standalone Java libraries to perform delegation in a flexible and arbitrary way. Finally, the gLite-delegation-service implements a stand-alone delegation service, which can be used by clients (Delegatees) to obtain delegation privileges. More information can be achieved through the technical documentations of relevant APIs [6].

5.4 Related Work

In this section we describe related works in the area of grid delegation and other relevant areas including active security models, and credential mapping mechanisms.

There are significant prior works on delegation in grids and other distributed computing environments. In this section, we describe some of these works that we found most relevant to our own.

Delegation Issuing Service (DIS) [34] is a service that issues X.509 Attribute Certificates (AC) on behalf of an Attribute Authority (typically a manager), integrated into the PERMIS authorization infrastructure. DIS only issue Attribute Certificates and lacks a standard protocol for interactions with an Attribute Authority who requests to issue ACs. It still needs a human to decide what is the least set of privileges to delegate to another entity and there is no support for an automated logic that can determine what are the least privileges to delegate.

The Community Authorization Service (CAS) [84] is a third-party, trusted by resource owners and used by end-users to obtain rights to access resources. It issues credentials, which limit the rights of the holder to only those agreed on between the VO and the resource providers. CAS has been primarily developed to set PCI extensions to limit the delegated rights to the intersection of rights between VOs and resource owners. The Virtual Organization Membership Service (VOMS) [12] has also been developed to solve this problem. It grants authorization data to users at the VO level by providing support for group membership, roles and capabilities. Although CAS and VOMS allow users to obtain and delegate specific rights via tags and roles during a session, they do not address the need for dynamic, on-demand

[11] http://glite.web.cern.ch/glite/

delegation, considering that it is often difficult to determine the rights needed by a grid job in advance.

"Explicit Trust Delegation"(EDT) described in [49] introduces "trusted agents" to provide support of dynamic delegation in the UNICORE security model (We fully described delegation in UNICORE in Section 3.1). These trusted agents (e.g. portals) are the entities that are allowed to create and sign Sub-AJOs on behalf of end-users. Scalability, the lack of providing auditing and robust control over delegated rights are some other issues and concerns of this approach.

Currently the GSI approach, described in Section 3.2, uses identity credentials with gridmap-files, a kind of ACL (access control list), in its authorization system. However, emerging works like Akenti [105], PERMIS [35] and, PRIMA [74] have been aimed at providing support for policy-based authorization system in GSI. Akenti and PERMIS provide a distributed policy-based authorization system. Akenti designed for grid environments using attribute certificates and delegated authorization. PERMIS also uses X.509 attribute certificates to hold roles/attributes and currently is fully integrated into the Globus GT4 authorization framework. Privilege Management and Authorization system (PRIMA) embeds authorization credentials in GSI proxy certificate to enable transport of authorizations in GSI. PRIMA can also embed XACML [101] privilege statements in GSI proxy credentials to be compared with XACML resource policies in PRIMA access control engine.

Rein [63] is an open and extensible approach for representing policies and provides a unified way of decision making by reasoning over policies and delegation networks. It uses Ontologies for specifying and reasoning about access policies in heterogeneous policy domains with different policy languages. Ontologies are used for describing and modeling different information in this framework such as policies, requests and delegation of authority and trust. However, in regard to delegation, this framework can only be used for simple scenarios and mainly for asserting delegation chains and delegation constraints cryptographically. From this perspective, Rein is analogous to other policy frameworks developed to support delegation of authority and trust to address delegation and can not be used as a complete solution for addressing the issue of principle of least privileges in grids.

Rein basically uses high level Rei concepts [61, 64], which is a policy framework that we used for reasoning over delegation policies in our framework. It has strong potential to meet the requirements of active delegation framework described in Paper II. The Rei policy framework permits specifying, analyzing and reasoning over declarative policies defined as norms of behavior [62, 61]. Rei policies can restrict domain actions that an entity can/must perform on resources in the environment, allowing policies to be deployed as contextually constrained deontic concepts, i.e., permission, prohibition, obligation and dispensation. In the Rei policy specification[12], context conditions can be specified by defining one or more *constraints*. A constraint, which may be simple or Boolean, i.e., the boolean combination of a pair of simple constraints, defines a set of actors or a set of actions that fulfill a certain

[12]http://www.cs.umbc.edu/ lkagal1/rei/

property. Rei's support for contexts makes it well-suited for our purposes.

In order to use Rei, as described in Paper II, we have extended SpeechAct delegation by incorporating a new property called *pre-condition*, which is defined as the conditions that need to be true before the delegation speech act can be performed. These pre-conditions are in fact constructing the "context" in which delegation can be activated and a delegation speech act can be triggered in response to a request. We need to recognize the distinction between the *pre-condition* property and the *condition* property defined originally in the Rei specification. The *pre-condition* determines when rights may be delegated, while the *condition* determines when rights may be used. Grid tasks can take longer than the validity interval of a context for which delegation is activated, which implies that constraints that must be satisfied before delegating rights should not necessarily remain true to make the delegated rights usable.

"Multiple Authorization" is a concept suggested in [116] for restricted delegation to prevent abusing of delegated tasks in a management system based on mobile agents. It proposes to share the responsibility for protected operations and to supervise actions of subordinates instead of transferring rights or even chains between agents before delegating the task. In this approach, a protected operation cannot be executed unless additional authorizations by other partners are provided. This approach binds authorization to particular and protected operations under special circumstances. This enables a fine-grained access control on a delegated task. This approach introduces two kinds of agents; work agents and authorization agents. Authorization agents are owned by the entities that their approvals are required for executing the task. When the work agent needs to execute a delegated task, it has to request for the authorizations that are needed to complete the task. Thus it sends a message to all agents and asks them to send their corresponding authorization agents. When all the authorization agents are collected, they will grant the execution of task by the work agent on behalf of user.

The "Workflow-based Authorization Service" (WAS) [66] proposes an authorization architecture for supporting restricted delegation and rights management. The WAS architecture uses a task workflow, created from the task source code, to obtain the sequence of required rights for executing the task. This can provide a useful way of determining the required rights in advance for deterministic jobs. However, in practice we still need on-demand delegation, because even if we can predict the job's behavior, the environment is dynamic, and we need the flexibility to use different services on grids opportunistically.

Trust negotiation is an approach for establishing trust in open distributed systems [25, 130]. In trust negotiation, two strangers carry out a bilateral and iterative exchange of attribute credentials to gradually establish trust in one another. On-demand delegation mechanism sets out to solve somewhat the different problem of determining when to grant more privileges to a sub-job running on behalf of a user. This approach complements trust negotiation in that our work could be used in a system employing trust negotiation to determine when a user's sensitive attribute certificates should be accessible to his sub-jobs. The Delegation Service in on-demand delegation framework also bears some resemblance to the Traust Service [70], which acts as a stand-alone authorization service that uses trust negotiation to broker access tokens for resources in its security domain.

The notation of context and active security system are observed in many works. In some of them diverse delegation models are also supported. However, as far as we are aware, the delegation concept in those works has not been investigated from the perspective of our own work presented in this thesis. Most of the works in this context are based on the RBAC96 family [93], which supports role activation within sessions to provide an active model of authorization management and access control models, such as the TeaM-based Access Control (TMAC) model, which is introduced in [102] and extended in [51] to a family of context aware access control models. TMAC basically recognizes the importance of context information for just-in-time activation of permissions. Similar to the TMAC approach, the Task-Base Access Control (TBAC) model [104] is also used for management of authorizations that encapsulate a group of permissions, in a way that they are turned-on only in a just-in-time fashion and synchronized with the processing of authorizations in progressing tasks. The OASIS RBAC model [127] is also an extension to the role-based access control architecture. OASIS RBAC does not use role delegation but instead defines the notion of appointment. One could consider applying our work to these systems by employing an active security model to determine when a team or session needs to be created or a new member needs to be joined into an established context (team).

None of above approaches addresses all the requirements of delegation described earlier. For the works around active security systems, although they provide support for contexts, they lack the ability to determine the "need" for establishing contexts "dynamically", which is an essential requirement for highly dynamic environments like grids. This is where we introduce the Context Manager as an architectural component. We also recognize a very special requirement of grid delegation when we distinguish the constraints applied at authorization time from the constraints applied at delegation time. The former implies that a delegation must be valid as long as the conditions are fulfilled, while the latter specifies that a delegation can be activated only when a set of pre-conditions are met. This is where we extend the Rei policy language to incorporate "pre-conditions" in the delegation SpeechAct as described earlier. What distinguishes our work from others is the adaptation of the active delegation model to grids and providing a supporting framework which can be integrated into current grid systems and leveraged by existing security mechanisms. Our framework further provides more support for observing and auditing the delegation process adapted to the dynamic requirements of grid applications.

In the context of integration of workflows when performing delegation, the work presented in [119, 118] provides an authorization model to handle delegation in an RBAC workflow environment. Their model extends the RBAC model to support a user-to-user delegation, user to a group of users delegation and revocations. Their workflow delegation model addresses dynamic constraint handling by defining specific delegations. Multiple delegation and partial delegation through the chain of delegation is also supported by this model.

The work presented in [11] incorporates delegation in a workflow management system. There is a delegation module, which is invoked whenever a delegation decision for task assignment must be made. There is a delegation table that only displays static constraints and relationships of delegations; then when a workflow assigns a task, this assignment is screened by the delegation module to determine whether delegation is required. The delegation module in turn inquires delegation tables and reply "yes" or "no" to the assignment request. It will notify the workflow engine to generate a new task assignment for a new delegatee. This model uses a static delegation table along with a static time constraints model; the authorization and the workflow are not accommodated and therefore if a task is completed sooner than anticipated, the delegated authorization being valid for longer than required.

All these approaches perform delegation in a full-ahead-plan tied to a particular user or group. They are mainly intended to business workflows which are more static, predictable and controlled environments as described in Paper III.

In the context of security credential mapping our work was originally inspired from the KX.509 protocol [41], which provides a mechanism for obtaining X.509 identity certificates based on a Kerberos domain login identity. Two main components in this protocol are i) the KCA (Kerberized Certification Authority) [67], a Kerberized service, running inside a Kerberos domain, and provides the functionality of an X.509 certification authority, and ii) the KX.509 that is a standalone client program to acquire a short-lived X.509 certificate from the KCA for an authenticated Kerberos user. As for the differences, CMS uses a standard interface for exchanging security tokens where the KX.509 does not. Our approach is more general and provides support for more use cases in the sense that it can let both the user or any delegated entity request for exchanging a security token (not only a Web browser as approached by KX.509).

MyProxy [81] is a software for managing PKI security credentials. It combines an on-line credential repository with an online certification authority to allow users to securely obtain credentials when and where needed. MyProxy supports storage of multiple credentials (X.509 proxy certificates) per user and retrieval as needed using username and password.

SACRED (Securely Available Credentials) [54] is an IETF framework and protocol for credential portability to allow secure and authenticated access to security credentials. This project provides SACRED client and server implementations.

CredEx [114], is a Web Service which provides a secure storage of credentials and enables exchanging of different credential types using the WS-Trust token exchange protocol. It can be seen as a credential mapping mechanism for exchanging username-password token to X.509 certificates.

The works described above basically develop an on-line credential repository to ease and secure the process of credential management. Although such on-line credential repositories are also required for many use-cases, however, we should emphasize that the work presented in this thesis was intentionally designed for just not storing credentials on a remote and on-line repository; it is rather designed to perform an on-the-fly credential mapping to make a non-vulnerable, lightweight, easy operated and quick to configure service. Furthermore, these solutions do not provide a full mesh credential exchange as provided by the CMS solution presented in this thesis.

5.5 Conclusion

In this thesis we demonstrated the importance of delegation in a wide range of grid applications and use-cases. We highlighted the shortcomings of existing mechanisms in providing a dynamic, restricted, standard and interoperable delegation system for grid applications. We discussed that current approaches are unable to provide secure and restricted delegation for dynamic grid applications. We demonstrated that current delegation mechanisms are mostly coupled to the underlying security mechanisms and are not interoperable between different grid security systems; furthermore, we demonstrated that there is a lack of standard interfaces and protocols for performing delegation in heterogeneous grid environments.

We solved these problems with a novel approach providing restricted delegation for dynamic execution of grid applications. We analyzed and developed the building blocks of a delegation framework, which allows on-demand determination, requesting and provisioning of delegated credentials. This framework has the benefits of observing, screening and auditing of delegated rights during the execution of tasks. We described how this framework, which leverages a call-back mechanism and a flexible and interoperable delegation protocol (GrDP), restricts the scope of delegated rights solely to those required for completing an intended task. In order to address some security concerns and preventing violation of delegation policies, we extended our framework to be capable of establishing "contexts" in which the need for additional delegated rights could be verified.

We discussed that on-demand delegation, although having the benefits of real-time control and auditing at the delegation service, might penalize delegation service performance for obtaining rights. In that regard, we developed a strategy to optimize on-demand delegation by delegating more rights to jobs either at the time of launching delegation or during each callback; thus, a delegatee does not have to callback to a delegation service so frequently. Determining these set of rights was a challenging issue and we addressed this challenge by integrating on-demand delegation framework with workflows.

We introduced the concept of Risk of Delegation (RoD) parameter in on-demand delegation framework. We further implemented appropriated algorithms for three different models of bottom-up delegation: Per-step delegation, Multi-step delegation and One-Step delegation. These models were designed to perform delegation in different levels of efficiency and restriction based on the current threat landscape indicated by the RoD parameter.

Finally, in order to address the cross-organizational authentication and identification, we developed a credential mapping service for on-the-fly exchange of security credentials in grids. This service was built using off-the-shelf components, standards and technologies.

5.6 Future Work

We proposed using the Rei policy framework for on-demand delegation frameworks. However, we believe that any other policy language incorporating the concept of contexts and providing support for delegation can also be used to describe active delegation policies. Therefore, future work could be investigating how other policy languages may be used in such framework. One alternative is XACML [101], a standard and general-purpose policy system designed to support the needs of modern authorization systems. The current policy schema of XACML version 2.0 does not support delegation; however, version 3.0 of this policy language will incorporate delegation concepts in the XACML policy schema, making XACML a strong and standard alternative for deployment in on-demand delegation frameworks.

We have not yet integrated our framework with a real grid workflow system. We only gave a formal definition and evaluation of that idea, and therefore another future task is to provide a concrete integration of the on-demand delegation with a real grid workflow system. Adopting the concept of bottom-up delegation to other grid middlewares and frameworks is another task and topic of future study.

The level of granularity of resource and system description can also affect the complexity of restricted delegation. Fine-grained system descriptions and access rights may result in unnecessarily complicated processes for restricting delegation and high overhead. This may sacrifice the usability of the whole system. Therefore, an investigation should be performed as to what is the best level of granularity to describe the system and grid resources without losing security and introducing unnecessary complexity.

Bibliography

[1] Martin Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15(4):706–734, 1993.

[2] Nabil R. Adam, Vijayalakshmi Atluri, and Wei-Kuang Huang. Modeling and analysis of workflows using petri nets. *J. Intell. Inf. Syst.*, 10(2):131–158, 1998.

[3] Gail-Joon Ahn and Ravi Sandhu. Role-based authorization constraints specification. *ACM Trans. Inf. Syst. Secur.*, 3(4):207–226, 2000.

[4] M. Ahsant, J. Basney, and L. Johnsson. Dynamic, context-aware, least-privilege grid delegation. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing (Grid2007)*, pages 209–216. IEEE Press, Sept. 2007.

[5] M. Ahsant, M. Surridge, T. Leonard, A. Krishna, and O. Mulmo. Dynamic trust federation in grids. In *In Proceedings of the 4th International Conference on Trust Management*, volume 3986 of *Lecture Notes in Computer Science*, pages 3–18. Springer-Verlag, 2006.

[6] Mehran Ahsant. glite delegation, java apis for glite delegation. Technical report, Royal Institute of Technology (KTH), October 2004.

[7] Mehran Ahsant and Jim Basney. Workflows in dynamic and restricted delegation. In *Proceedings of the 4th International Conference on Availability, Reliability and Security (ARES/CISIS 2009).* IEEE Computer Society, March 2009.

[8] Mehran Ahsant, Jim Basney, and Olle Mulmo. Grid delegation protocol. In *Proceedings of the Workshop on Grid Security Practice and Experience*, volume YCS-2004-380, pages 81–91. Oxford, UK, July 2004.

[9] Mehran Ahsant, Jim Basney, Olle Mulmo, Adam J. Lee, and Lennart Johnsson. Toward an on-demand restricted delegation mechanism for grids. In *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing (Grid2006)*, pages 152–159. IEEE Press, Sep. 2006.

[10] Mehran Ahsant, Esteban Talavera González, and Jim Basney. Security credential mapping in grids. In *Proceedings of the 4th International Conference on Availability, Reliability and Security (ARES/CISIS 2009)*. IEEE Computer Society Press, March 2009.

[11] Kumar Akhil. A framework for handling delegation in workflow management systems. In *Workshop on Information Technology and Systems (WITS)*, December 1999.

[12] Roberto Alfieri, Roberto Cecchini, Vincenzo Ciaschini, Luca dell'Agnello, Akos Frohner, Alberto Gianoli, Karoly Lorentey, and Fabio Spataro. Voms, an authorization system for virtual organizations. In *European Across Grids Conference*, pages 33–40, 2003.

[13] R. J. Anderson. A security policy model for clinical information systems. In *SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy*, page 30, Washington, DC, USA, 1996. IEEE Computer Society.

[14] Vijay Atluri. Security for workflow systems. Technical Report 2, Elsevier Science Ltd., 2001.

[15] Vijayalakshmi Atluri and Wei-Kuang Huang. An extended petri net model for supporting workflow in a multilevel secure environment. In *Proceedings of the tenth annual IFIP TC11/WG11.3 international conference on Database security: volume X : status and prospects*, pages 240–258, London, UK, UK, 1997. Chapman & Hall, Ltd.

[16] Vijayalakshmi Atluri and Wei kuang Huang. An authorization model for workflows. In *ESORICS 96: Proceedings of the 4th European Symposium on Research in Computer Security*, pages 44–64, London, UK, 1996. Springer-Verlag.

[17] Vijayalakshmi Atluri and Janice Warner. Supporting conditional delegation in secure workflow management systems. In *SACMAT*, pages 49–58, 2005.

[18] Olav Bandmann, Babak Sadighi Firozabadi, and Mads Dam. Constrained delegation. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 131, Washington, DC, USA, 2002. IEEE Computer Society.

[19] E. Barka and R. Sandhu. Framework for role-based delegation models. *acsac*, 0:168, 2000.

[20] E. Barka and R. Sandhu. Framework for role-based delegation models. In *Proceedings of the 16th Annual Computer Security Applications Conference*, December 2000.

[21] E. Barka and R. Sandhu. Framework for role-based delegation models. In *ACSAC '00: Proceedings of the 16th Annual Computer Security Applications Conference*, page 168, Washington, DC, USA, 2000. IEEE Computer Society.

[22] E. Barka and R. Sandhu. A role-based delegation model and some extensions. In *Proceedings of the 23rd National Information Systems Security Conference*, October 2000.

[23] Adam Barker and Jano van Hemert. Scientific workflow: A survey and research directions. In *PPAM*, pages 746–753, 2007.

[24] J. Basney, W. Yurcik, R. Bonilla, and A. Slagell. The credential wallet: A classification of credential repositories highlighting myproxy. In *Proceedings of the 31st Research Conference on Communication, Information and Internet Policy (TPRC 2003)*, Arlington, Virginia, September 19-21 2003.

[25] Jim Basney, Wolfgang Nejdl, Daniel Olmedilla, Von Welch, and Marianne Winslett. Negotiating trust on the grid. In *Semantic Grid: The Convergence of Technologies*, Dagstuhl Seminar Proceedings, 2005.

[26] Sean Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, L.A. Stein, and F.W. Olin. Owl web ontology language reference, w3c recommendation. Available online from http://www.w3.org/TR/owl-features, March 2004.

[27] D. Bell and L. La Padula. Secure computer systems: Unified exposition and multics interpretation. Technical report, MITRE Corporation, Bedford, MA, March 1976.

[28] S. M. Bellovin and M. Merritt. Limitations of the kerberos authentication system. *SIGCOMM Comput. Commun. Rev.*, 20(5):119–132, 1990.

[29] M. Benantar. The internet public key infrastructure. *IBM Syst. J.*, 40(3):648–665, 2001.

[30] Elisa Bertino, Elena Ferrari, and Vijay Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.*, 2(1):65–104, 1999.

[31] Elisa Bertino, Elena Ferrari, and Vijayalakshmi Atluri. A flexible model supporting the specification and enforcement of role-based authorization in workflow management systems. In *RBAC 97: Proceedings of the second ACM workshop on Role-based access control*, pages 1–12, New York, NY, USA, 1997. ACM.

[32] Matt Bishop. *Computer Security: Art and Science.* Addison-Wesley Professional, 2002.

[33] Claudio Calvelli and Vijay Varadharajan. An analysis of some delegation protocols for distributed systems. In *CSFW*, pages 92–110, 1992.

[34] David W. Chadwick. Delegation issuing service for x.509. In *Proceedings of the 4th Annual PKI R&D Workshop*, pages 66–77, USA, April 2005. NIST Technical Publication, IR 7224.

[35] David W. Chadwick, Alexander Otenko, and Edward Ball. Role-based access control with x.509 attribute certificates. *IEEE Internet Computing*, 7(2):62–69, 2003.

[36] L. Chen, S. J. Cox, F. Tao, N. R. Shadbolt, C. Puleston, and C. Goble. Empowering resource providers to build the semantic grid. In *WI '04: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 271–277, Washington, DC, USA, 2004. IEEE Computer Society.

[37] Shreyas Cholia. Profile for slcs x.509 public key certification authorities with secured infrastructure version 2.1. Available online from http://www.tagpma.org/files/SLCS-2.1.pdf, August 2008.

[38] Eve Cohen, Roshan K. Thomas, William Winsborough, and Deborah Shands. Models for coalition-based access control (cbac). In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 97–106, New York, NY, USA, 2002. ACM Press.

[39] Ewa Deelman, James Blythe, Yolanda Gil, and Carl Kesselman. Workflow management in griphyn. In *Grid resource management: state of the art and future trends*, pages 99–116, Norwell, MA, USA, 2004. Kluwer Academic Publishers.

[40] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), jan 1999. Obsoleted by RFC 4346, updated by RFC 3546.

[41] William Doster, Marcus Watts, and Dan Hyde. The kx.509 protocol. Technical report, University of Michigan, February 2001.

[42] Bertino E., Bonatti P.A., and Ferrari E. Trbac: A temporal role-based access control model. *ACM Trans. Inf. Syst. Secur.*, 4(3):191–233, August 2001.

[43] Babak Sadighi Firozabadi, Marek J. Sergot, and Olav L. Bandmann. Using authority certificates to create management structures. In *Revised Papers from the 9th International Workshop on Security Protocols*, pages 134–145, London, UK, 2002. Springer-Verlag.

[44] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. Available online from http://www.globus.org/alliance/publications/papers/ogsa.pdf, June 2002.

[45] I. Foster, V. Nefedova, M. Ahsant, R. Anantha krishnan, L. Liming, R. Madduri, O. Mulmo, L. Pearlman, and F. Siebenlist. Streamlining grid operations: Definition and deployment of a portal-based user registration service. *Journal of Grid Computing*, 4(2):135–144, June 2006.

[46] Ian Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A security architecture for computational grids. In *ACM Conference on Computer and Communications Security*, pages 83–92, 1998.

[47] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, 2001.

[48] Geoffrey C. Fox and Dennis Gannon. Special issue: Workflow in grid systems: Editorials. *Concurr. Comput. : Pract. Exper.*, 18(10):1009–1019, 2006.

[49] David F.Snelling, Sven van den Berghe, and Vivian Qian. Explicit trust delegation: Security for dynamic grids. *FUJITSU Sci.Tech.Journal*, 40:282–294, 2004.

[50] Morrie Gasser and Ellen McDermott. An architecture for practical delegation in a distributed system. *sp*, 00:20, 1990.

[51] Christos K. Georgiadis, Ioannis Mavridis, George Pangalos, and Roshan K. Thomas. Flexible team-based access control using contexts. In *SACMAT*, pages 21–27, 2001.

[52] H. M. Gladney. Access control for large collections. *ACM Trans. Inf. Syst.*, 15(2):154–194, 1997.

[53] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[54] D. Gustafson, M. Just, M. Nystrom, A. Arsenault, and S. Farrell. Securely available credentials (sacred). RFC 3760 (Proposed Standard), April 2004.

[55] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in operating systems. *Commun. ACM*, 19(8):461–471, 1976.

[56] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459 (Proposed Standard), jan 1999. Obsoleted by RFC 3280.

[57] Valentina Huber. Unicore: A grid computing environment for distributed and parallel computing. In *PaCT*, pages 258–265, 2001.

[58] The Dublin Core Metadata Initiative. Available online from: http://dublincore.org/, February 2006.

[59] EGEE Security JRA3. Global security architecture, for web and legacy services. Available online from https://edms.cern.ch/document/487004/, September 2004.

[60] L. Kagal, T. Finin, and A. Joshi. Developing secure agent systems using delegation based trust management. In *Security of Mobile MultiAgent Systems (SEMAS 02)*, 2002.

[61] Lalana Kagal. Rei : A Policy Language for the Me-Centric Project. Technical report, HP Labs, September 2002. http://www.hpl.hp.com/techreports/2002/HPL-2002-270.html.

[62] Lalana Kagal. *A Policy-Based Approach to Governing Autonomous Behavior in Distributed Environments*. PhD thesis, University of Maryland Baltimore County, Baltimore MD 21250, September 2004.

[63] Lalana Kagal, Tim Berners-Lee, Dan Connolly, and Daniel Weitzner. Self-describing delegation networks for the web. In *POLICY '06: Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, pages 205–214, Washington, DC, USA, 2006. IEEE Computer Society.

[64] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, page 63, Washington, DC, USA, 2003. IEEE Computer Society.

[65] Lalana Kagal, Timothy W. Finin, and Anupam Joshi. A policy based approach to security for the semantic web. In *International Semantic Web Conference*, pages 402–418, 2003.

[66] Seung-Hyun Kim, Jong Kim, Sung-Je Hong, and Sangwan Kim. Workflow-based authorization service in grid. In *GRID '03: Proceedings of the Fourth International Workshop on Grid Computing*, page 94, Washington, DC, USA, 2003. IEEE Computer Society.

[67] Olga Kornievskaia, Peter Honeyman, Bill Doster, and Kevin Coffman. Kerberized credential translation: a solution to web access control. In *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, pages 18–18, Berkeley, CA, USA, 2001. USENIX Association.

[68] E Laure, S M Fisher, A Frohner, C Grandi, Peter Z Kunszt, A Krenek, O Mulmo, F Pacini, F Prelz, J White, M Barroso, P Buncic, F Hemmer, A Di Meglio, and A Edlund. Programming the grid with glite. Technical Report EGEE-TR-2006-001, CERN, Geneva, Mar 2006.

[69] Peter Lawrence, editor. *Workflow handbook 1997*. John Wiley & Sons, Inc., New York, NY, USA, 1997.

[70] Adam J. Lee, Marianne Winslett, Jim Basney, and Von Welch. Traust: a trust negotiation-based authorization service for open systems. In *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, New York, NY, USA, 2006. ACM Press.

[71] Travis Leithead, Wolfgang Nejdl, Daniel Olmedilla, Kent E. Seamons, Marianne Winslett, Ting Yu, and Charles C. Zhang. How to exploit ontologies for trust negotiation. In *ISWC Workshop on Trust, Security, and Reputation on the Semantic Web.*, volume 127 of *CEUR Workshop Proceedings*, Hiroshima, Japan, Nov. 2004. Technical University of Aachen (RWTH).

[72] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. Inf. Syst. Secur.*, 6(1):128–171, 2003.

[73] Liberty. Liberty alliance project. Available online from https://www.projectliberty.org/, May 2003.

[74] M. Lorch, D. B. Adams, D. Kafura, M. S. R. Koneni, A. Rathi, and S. Shah. The prima system for privilege management, authorization and enforcement in grid environments. In *GRID '03: Proceedings of the Fourth International Workshop on Grid Computing*, page 109, Washington, DC, USA, 2003. IEEE Computer Society.

[75] Ravi K. Madduri, Cynthia S. Hood, and William E. Allcock. Reliable file transfer in grid environments. In *LCN '02: Proceedings of the 27th Annual IEEE Conference on Local Computer Networks.*, pages 737–738, Washington, DC, USA, 2002. IEEE Computer Society.

[76] Paul Madsen. Ws-trust: Interoperable security for web services. http://webservices.xml.com/pub/a/ws/2003/06/24/ws-trust.html, June 2003.

[77] Paul Madsen. Web services security usernametoken profile. Available online from http://www.oasis-open.org/committees/documents.php, February 2004.

[78] Andrew McNab, Joni Hahkala, and Akos Frohner. Description of implementation and use of g-https by edg. GGF, draft doc5361, February 2004.

[79] N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist, V. Welch, I. Foster, , and S. Tuecke. The security architecture for open grid services., 2006.

[80] B. Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, September 1994.

[81] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the grid: Myproxy. In *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*. IEEE Press, August 2001.

[82] M. Nystrom and B. Kaliski. Certification request syntax specification. RFC 2986 (Proposed Standard), november 2000.

[83] Kunal Patel and Gopal Gupta. Semantic processing of the semantic web. In *International Semantic Web Conference*, pages 80–95, 2003.

[84] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A community authorization service for group collaboration. In *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02).*, page 50, Washington, DC, USA, 2002. IEEE Computer Society.

[85] S. Piger, C. Kunz, C. Grimm, and R. Groeper. Enhancing security in grids through self-restricted delegation of rights with user-based policies. In *Proceedings of the 26th IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN 08)*, pages 597–075. ACTA Press, 2008.

[86] Eric Prud'hommeaux and Andy Seaborne. Rdf data access working group. sparql query language for rdf. Available online from http://www.w3.org/TR/rdf-sparql-query/, April 2006.

[87] Michael Rambadt and Philipp Wieder. Unicore - globus interoperability: Getting the best of both worlds. In *HPDC*, page 422, 2002.

[88] Daniel A. Reed, Celso L. Mendes, Chang da Lu, Ian Foster, and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure - Application Tuning and Adaptation.* Morgan Kaufman, San Francisco, CA, second edition, 2003. ch. 16.

[89] E. Rescorla. HTTP Over TLS. RFC 2818 (Informational), may 2000.

[90] David De Roure, Nicholas R. Jennings, and Nigel R. Shadbolt. The semantic grid: A future e-science infrastructure. In *Grid Computing: Making the Global Infrastructure a Reality*, pages 437–470. John Wiley & Sons, 2003.

[91] Wasim Sadiq and Maria E. Orlowska. Analyzing process models using graph reduction techniques. *Inf. Syst.*, 25(2):117–134, 2000.

[92] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, Sep 1975.

[93] Ravi Sandhu. Rationale for the rbac96 family of access control models. In *RBAC '95: Proceedings of the first ACM Workshop on Role-based access control*, page 9, New York, NY, USA, 1996. ACM Press.

[94] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The nist model for role-based access control: towards a unified standard. In *Proceedings of the fifth ACM workshop on Role-based access control (RBAC 00)*, pages 47–63, New York, NY, USA, 2000. ACM.

[95] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[96] Shibbol. Shibboleth project. Available online from http://shibboleth.internet2.edu/, May 2008.

[97] Geoff Stoker, Brian S. White, Ellen Stackpole, T. J. Highley, and Marty Humphrey. Toward realizable restricted delegation in computational grids. In *HPCN Europe*, pages 32–41, 2001.

[98] M. Surridge, S. Taylor, D. De Roure, and E. Zaluska. Experiences with gria; industrial applications on a web services grid. In *Proc. First International Conference on e-Science and Grid Computing*, pages 98–105, 2005.

[99] S. Taylor, M. Surridge, and D. Marvin. Grid resources for industrial applications. In *Proc. IEEE International Conference on Web Services*, pages 402–409, 2004.

[100] OASIS TC. Security assertion markup language (saml). Available online from http://www.oasis-open.org/specs/index.php, August 2003.

[101] OASIS TC. extensible access control markup language (xacml) version 2.0. Available online from http://docs.oasis-open.org/xacml/2.0/, February 2005.

[102] Roshan K. Thomas. 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*, pages 13–19, New York, NY, USA, 1997. ACM Press.

[103] Roshan K. Thomas and Ravi S. Sandhu. Towards a task-based paradigm for flexible and adaptable access control in distributed applications. In *NSPW '92-93: Proceedings on the 1992-1993 workshop on New security paradigms*, pages 138–142, New York, NY, USA, 1993. ACM Press.

[104] Roshan K. Thomas and Ravi S. Sandhu. 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*, pages 166–181, London, UK, UK, 1998. Chapman & Hall, Ltd.

[105] Mary R. Thompson, Abdelilah Essiari, and Srilekha Mudumbai. Certificate-based authorization policy in a pki environment. *ACM Trans. Inf. Syst. Secur.*, 6(4):566–588, 2003.

[106] Alessandra Toninelli, Jeffrey Bradshaw, Lalana Kagal, and Rebecca Montanari. Rule-based and ontology-based policies: Toward a hybrid approach to control agents in pervasive environments. In *Proceedings of the Semantic Web and Policy Workshop*, November 2005.

[107] Gianluca Tonti, Jeffrey M. Bradshaw, Renia Jeffers, Rebecca Montanari, Niranjan Suri, and Andrzej Uszok. Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder. In *International Semantic Web Conference*, pages 419–437, 2003.

[108] Gruber T.R. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[109] TrustBridge. Trustbridge project., June 2002.

[110] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet x.509 public key infrastructure (pki) proxy certificate profile. RFC 3820 (Proposed Standard), jun 2004.

[111] Andrzej Uszok, Jeffrey M. Bradshaw, and Renia Jeffers. Kaos: A policy and domain services framework for grid computing and semantic web services. In *iTrust*, pages 16–26, 2004.

[112] Andrzej Uszok, Jeffrey M. Bradshaw, Matthew Johnson, Renia Jeffers, Austin Tate, Jeff Dalton, and Stuart Aitken. Kaos policy management for semantic web services. *IEEE Intelligent Systems*, 19(4):32–41, 2004.

[113] Wil M. P. van der Aalst, Alexander Hirnschall, and H. M. W. (Eric) Verbeek. An alternative way to analyze workflow graphs. In *CAiSE 02: Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, pages 535–552, London, UK, 2002. Springer-Verlag.

[114] David Del Vecchio, Marty Humphrey, Jim Basney, and Nataraj Nagaratnam. Credex: User-centric credential management for grid and web services. In *ICWS '05: Proceedings of the IEEE International Conference on Web Services*, pages 149–156, Washington, DC, USA, 2005. IEEE Computer Society.

[115] Karin Venter and Martin S Olivier. The delegation authorization model: A model for the dynamic delegation of authorization rights in a secure workflow management system. In *ISSA2002*, Muldersdrift, South Africa, 2002. Published electronically.

[116] G. Vogt. Delegation of tasks and rights. In INRIA, editor, *Proceedings of the 12th Annual IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 2001)*, pages 327–337, Nancy, France, Oct 2001. IFIP/IEEE, INRIA Press.

[117] Bechhofer R. Volz and P. Lord. Cooking the semantic web with the owl api. In *Proceedings of International Semantic Web Conference*, pages 659–675, Sanibel Island, 2003.

[118] Jacques Wainer and Paulo Barthelmess. Wrbac : A workflow security model incorporating controlled overriding of constraints. *International Journal of Cooperative Information Systems*, 12:2003, 2003.

[119] Jacques Wainer, Akhil Kumar, and Paulo Barthelmess. Dw-rbac: A formal security model of delegation and revocation in workflow systems. *Inf. Syst.*, 32(3):365–384, 2007.

[120] Jun Wang, David Del Vecchio, and Marty Humphrey. Extending the security assertion markup language to support delegation for web services and grid services. In *ICWS '05: Proceedings of the IEEE International Conference on Web Services (ICWS'05)*, pages 67–74, Washington, DC, USA, 2005. IEEE Computer Society Press.

[121] V. Welch, I. Foster, Kesselman C.and Mulmo O., Pearlman L., Tuecke S., Gawor J., and Meder S.and Siebenlist F. X.509 proxy certificate for dynamic delegation. In *Proceedings of the 3rd Annual PKI Workshop*, pages 20–25, Gaithersburg MD, USA, April 2004.

[122] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for grid services. In *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, page 48, Washington, DC, USA, 2003. IEEE Computer Society.

[123] WS-Federation. Web services federation language (ws-federation). Available online from http://www.ibm.com/developerworks/library/specification/ws-fed/, December 2006.

[124] WS-Policy. Web services policy framework (ws-policy). Available online from http://www.ibm.com/developerworks/library/ws-polfram, March 2006.

[125] WS-SecurityPolicy. Web services security policy language (ws-securitypolicy). Available online from http://www.ibm.com/developerworks/library/ws-secpol/, July 2005.

[126] WS-Trust. Web service trust language(ws-trust). Available online from http://www.ibm.com/developerworks/library/specification/ws-trust/, February 2005.

[127] Walt Yao, Ken Moody, and Jean Bacon. A model of oasis role-based access control and its support for active security. In *SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies*, pages 171–181, New York, NY, USA, 2001. ACM Press.

[128] Gang Yin, Huai min Wang, Dian xi Shi, Yan Jia, and Meng Teng. A rule-based framework for role-based constrained delegation. In *InfoSecu '04: Proceedings of the 3rd international conference on Information security*, pages 186–191, New York, NY, USA, 2004. ACM Press.

[129] Jia Yu and Rajkumar Buyya. A novel architecture for realizing grid workflow using tuple spaces. In *GRID 04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 119–128, Washington, DC, USA, 2004. IEEE Computer Society.

[130] T. Yu, M. Winslett, and K.E.Seamons. Automated trust negotiation over the internet. In *The 6th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, FL, July 2002.

[131] Longhua Zhang, Gail-Joon Ahn, and Bei-Tseng Chu. A rule-based framework for role based delegation. In *SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies*, pages 153–162, New York, NY, USA, 2001. ACM Press.

[132] Longhua Zhang, Gail-Joon Ahn, and Bei-Tseng Chu. A role-based delegation framework for healthcare information systems. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 125–134, New York, NY, USA, 2002. ACM Press.

Appendix I: Delegation ontology in OWL/XML notation

```
<?xml version="1.0"?>

<!DOCTYPE owl2xml:Ontology [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY daml "http://www.daml.org/2001/03/daml+oil#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY assert "http://www.owl-ontologies.com/assert.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY  "http://www.nada.kth.se/Delegation.owl#" >
    <!ENTITY p1 "http://www.daml.org/2003/01/periodictable/
PeriodicTable#" >
]>

<owl2xml:Ontology xmlns="http://www.nada.kth.se/Delegation.owl#"
      xml:base="http://www.w3.org/2006/12/owl2-xml#"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:="http://www.nada.kth.se/Delegation.owl#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
      xmlns:p1="http://www.daml.org/2003/01/periodictable/
PeriodicTable#"
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:assert="http://www.owl-ontologies.com/assert.owl#"
      xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
      xmlns:Delegation="http://www.nada.kth.se/Delegation.owl#"
```

```
  owl2xml:URI="http://www.nada.kth.se/Delegation.owl">
<owl2xml:Import
    >http://www.owl-ontologies.com/assert.owl</owl2xml:Import>
<owl2xml:Import
    >http://www.owl-ontologies.com/assert.owl</owl2xml:Import>
<owl2xml:SubClassOf>
    <owl2xml:Class owl2xml:URI="&;Capability"/>
    <owl2xml:Class owl2xml:URI="&owl;Thing"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
    <owl2xml:Class owl2xml:URI="&;Capability"/>
    <owl2xml:ObjectSomeValuesFrom>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasObject"/>
        <owl2xml:Class owl2xml:URI="&;Object"/>
    </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
    <owl2xml:Class owl2xml:URI="&;Capability"/>
    <owl2xml:ObjectSomeValuesFrom>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasVerb"/>
        <owl2xml:Class owl2xml:URI="&;Verb"/>
    </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
    <owl2xml:Class owl2xml:URI="&;DN"/>
    <owl2xml:Class owl2xml:URI="&;Identifier"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
    <owl2xml:Class owl2xml:URI="&;Name"/>
    <owl2xml:Class owl2xml:URI="&;Identifier"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
    <owl2xml:Class owl2xml:URI="&;NotAfter"/>
    <owl2xml:Class owl2xml:URI="&;Constraints"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
    <owl2xml:Class owl2xml:URI="&;NotBefore"/>
    <owl2xml:Class owl2xml:URI="&;Constraints"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
    <owl2xml:Class owl2xml:URI="&;URI"/>
    <owl2xml:Class owl2xml:URI="&;Identifier"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
```

```
        <owl2xml:Class owl2xml:URI="&;URL"/>
        <owl2xml:Class owl2xml:URI="&;Identifier"/>
    </owl2xml:SubClassOf>
    <owl2xml:SubClassOf>
        <owl2xml:Class owl2xml:URI="&;UUID"/>
        <owl2xml:Class owl2xml:URI="&;Identifier"/>
    </owl2xml:SubClassOf>
    <owl2xml:SubClassOf>
        <owl2xml:Class owl2xml:URI="&;ValidityInterval"/>
        <owl2xml:Class owl2xml:URI="&;Constraints"/>
    </owl2xml:SubClassOf>
    <owl2xml:SubClassOf>
        <owl2xml:Class owl2xml:URI="&;delegationIdn"/>
        <owl2xml:Class owl2xml:URI="&;Identifier"/>
    </owl2xml:SubClassOf>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;authorizes"/>
        <owl2xml:Class owl2xml:URI="&;Credential"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;authorizes"/>
        <owl2xml:ObjectUnionOf>
            <owl2xml:Class owl2xml:URI="&;Capability"/>
            <owl2xml:Class owl2xml:URI="&;Delegation"/>
        </owl2xml:ObjectUnionOf>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:InverseObjectProperties>
        <owl2xml:ObjectProperty owl2xml:URI="&;delegates"/>
        <owl2xml:ObjectProperty owl2xml:URI="&;isDelegatedBy"/>
    </owl2xml:InverseObjectProperties>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;delegates"/>
        <owl2xml:Class owl2xml:URI="&;Delegator"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;delegates"/>
        <owl2xml:Class owl2xml:URI="&;Delegation"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasA"/>
        <owl2xml:Class owl2xml:URI="&;Delegator"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasA"/>
        <owl2xml:Class owl2xml:URI="&;Capability"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasDependency"/>
        <owl2xml:Class owl2xml:URI="&;Capability"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasDependency"/>
        <owl2xml:Class owl2xml:URI="&;Capability"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasObject"/>
        <owl2xml:Class owl2xml:URI="&;Capability"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasObject"/>
        <owl2xml:Class owl2xml:URI="&;Object"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:SubObjectPropertyOf>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasValidityInterval"/>
        <owl2xml:ObjectProperty owl2xml:URI="&;isRestrictedBy"/>
    </owl2xml:SubObjectPropertyOf>
    <owl2xml:InverseObjectProperties>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasVerb"/>
        <owl2xml:ObjectProperty owl2xml:URI="&;isVerbOf"/>
    </owl2xml:InverseObjectProperties>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasVerb"/>
        <owl2xml:Class owl2xml:URI="&;Capability"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasVerb"/>
        <owl2xml:Class owl2xml:URI="&;Verb"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:FunctionalObjectProperty>
        <owl2xml:ObjectProperty owl2xml:URI="&;identifies"/>
    </owl2xml:FunctionalObjectProperty>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;identifies"/>
        <owl2xml:Class owl2xml:URI="&;Identifier"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;identifies"/>
        <owl2xml:ObjectUnionOf>
```

```
            <owl2xml:Class owl2xml:URI="&;Credential"/>
            <owl2xml:Class owl2xml:URI="&;Delegation"/>
            <owl2xml:Class owl2xml:URI="&;Delegator"/>
            <owl2xml:Class owl2xml:URI="&;Object"/>
            <owl2xml:Class owl2xml:URI="&;Subject"/>
        </owl2xml:ObjectUnionOf>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;isApplicableTo"/>
        <owl2xml:Class owl2xml:URI="&;Delegation"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;isApplicableTo"/>
        <owl2xml:Class owl2xml:URI="&;Capability"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:InverseObjectProperties>
        <owl2xml:ObjectProperty owl2xml:URI="&;isAuthorizedBy"/>
        <owl2xml:ObjectProperty owl2xml:URI="&;authorizes"/>
    </owl2xml:InverseObjectProperties>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;isAuthorizedBy"/>
        <owl2xml:ObjectUnionOf>
            <owl2xml:Class owl2xml:URI="&;Capability"/>
            <owl2xml:Class owl2xml:URI="&;Delegation"/>
        </owl2xml:ObjectUnionOf>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;isAuthorizedBy"/>
        <owl2xml:Class owl2xml:URI="&;Credential"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;isDelegatableTo"/>
        <owl2xml:Class owl2xml:URI="&;Delegation"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;isDelegatableTo"/>
        <owl2xml:Class owl2xml:URI="&;Subject"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;isDelegatedBy"/>
        <owl2xml:Class owl2xml:URI="&;Delegation"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;isDelegatedBy"/>
        <owl2xml:Class owl2xml:URI="&;Delegator"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:TransitiveObjectProperty>
        <owl2xml:ObjectProperty owl2xml:URI="&;isDelegatedTo"/>
    </owl2xml:TransitiveObjectProperty>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;isDelegatedTo"/>
        <owl2xml:Class owl2xml:URI="&;Delegation"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;isDelegatedTo"/>
        <owl2xml:Class owl2xml:URI="&;Subject"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:InverseObjectProperties>
        <owl2xml:ObjectProperty owl2xml:URI="&;isIdentifiedBy"/>
        <owl2xml:ObjectProperty owl2xml:URI="&;identifies"/>
    </owl2xml:InverseObjectProperties>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;isIdentifiedBy"/>
        <owl2xml:ObjectUnionOf>
            <owl2xml:Class owl2xml:URI="&;Credential"/>
            <owl2xml:Class owl2xml:URI="&;Delegation"/>
            <owl2xml:Class owl2xml:URI="&;Delegator"/>
            <owl2xml:Class owl2xml:URI="&;Object"/>
            <owl2xml:Class owl2xml:URI="&;Subject"/>
        </owl2xml:ObjectUnionOf>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;isIdentifiedBy"/>
        <owl2xml:Class owl2xml:URI="&;Identifier"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:InverseObjectProperties>
        <owl2xml:ObjectProperty owl2xml:URI="&;isObjectOf"/>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasObject"/>
    </owl2xml:InverseObjectProperties>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;isObjectOf"/>
        <owl2xml:Class owl2xml:URI="&;Object"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;isObjectOf"/>
        <owl2xml:Class owl2xml:URI="&;Capability"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:InverseObjectProperties>
```

```
        <owl2xml:ObjectProperty owl2xml:URI="&;isOwned"/>
        <owl2xml:ObjectProperty owl2xml:URI="&;hasA"/>
    </owl2xml:InverseObjectProperties>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;isOwned"/>
        <owl2xml:Class owl2xml:URI="&;Capability"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;isOwned"/>
        <owl2xml:Class owl2xml:URI="&;Delegator"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:SubObjectPropertyOf>
        <owl2xml:ObjectProperty owl2xml:URI="&;isValidNotAfter"/>
        <owl2xml:ObjectProperty owl2xml:URI="&;isRestrictedBy"/>
    </owl2xml:SubObjectPropertyOf>
    <owl2xml:SubObjectPropertyOf>
        <owl2xml:ObjectProperty owl2xml:URI="&;isValidNotBefore"/>
        <owl2xml:ObjectProperty owl2xml:URI="&;isRestrictedBy"/>
    </owl2xml:SubObjectPropertyOf>
    <owl2xml:ObjectPropertyDomain>
        <owl2xml:ObjectProperty owl2xml:URI="&;isVerbOf"/>
        <owl2xml:Class owl2xml:URI="&;Verb"/>
    </owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectPropertyRange>
        <owl2xml:ObjectProperty owl2xml:URI="&;isVerbOf"/>
        <owl2xml:Class owl2xml:URI="&;Capability"/>
    </owl2xml:ObjectPropertyRange>
    <owl2xml:DataPropertyDomain>
        <owl2xml:DataProperty owl2xml:URI="&;EPR"/>
        <owl2xml:Class owl2xml:URI="&;Identifier"/>
    </owl2xml:DataPropertyDomain>
    <owl2xml:DataPropertyRange>
        <owl2xml:DataProperty owl2xml:URI="&;EPR"/>
        <owl2xml:Datatype owl2xml:URI="&xsd;string"/>
    </owl2xml:DataPropertyRange>
    <owl2xml:DataPropertyDomain>
        <owl2xml:DataProperty owl2xml:URI="&;hasValidityInterval"/>
        <owl2xml:Class owl2xml:URI="&;Delegation"/>
    </owl2xml:DataPropertyDomain>
    <owl2xml:DataPropertyRange>
        <owl2xml:DataProperty owl2xml:URI="&;hasValidityInterval"/>
        <owl2xml:Datatype owl2xml:URI="&xsd;int"/>
    </owl2xml:DataPropertyRange>
    <owl2xml:DataPropertyDomain>
```

```
        <owl2xml:DataProperty owl2xml:URI="&;isValidNotAfter"/>
        <owl2xml:Class owl2xml:URI="&;Delegation"/>
    </owl2xml:DataPropertyDomain>
    <owl2xml:DataPropertyRange>
        <owl2xml:DataProperty owl2xml:URI="&;isValidNotAfter"/>
        <owl2xml:Datatype owl2xml:URI="&xsd;string"/>
    </owl2xml:DataPropertyRange>
    <owl2xml:DataPropertyDomain>
        <owl2xml:DataProperty owl2xml:URI="&;isValidNotBefore"/>
        <owl2xml:Class owl2xml:URI="&;Delegation"/>
    </owl2xml:DataPropertyDomain>
    <owl2xml:DataPropertyRange>
        <owl2xml:DataProperty owl2xml:URI="&;isValidNotBefore"/>
        <owl2xml:Datatype owl2xml:URI="&xsd;string"/>
    </owl2xml:DataPropertyRange>
    <owl2xml:DataPropertyAssertion>
        <owl2xml:DataProperty owl2xml:URI="&assert;notEmpty"/>
        <owl2xml:Individual owl2xml:URI="&;Delegator"/>
        <owl2xml:Constant owl2xml:datatypeURI="&xsd;string"
            >SELECT   ?Delegator   ?Capability
WHERE {   ?Delegator:hasA   ?Capability
             }</owl2xml:Constant>
    </owl2xml:DataPropertyAssertion>
</owl2xml:Ontology>
<!———————>
```