# Toward an On-Demand Restricted Delegation Mechanism for Grids

Mehran Ahsant [#1], Jim Basney [*2], Olle Mulmo [#3], Adam J. Lee [%4], Lennart Johnsson [#5]

[#]*Center for Parallel Computers, Royal Institute of Technology*
*Valhallavgen 79, 10044 Stockholm, Sweden*
[1]mehrana@pdc.kth.se
[3]mulmo@pdc.kth.se
[5]johnsson@pdc.kth.se

[*]*National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign*
*1205 W. Clark St., Urbana, IL 61801 USA*
[2]jbasney@ncsa.uiuc.edu

[%]*Department of Computer Science, University of Illinois at Urbana-Champaign*
*201 N. Goodwin Ave., Urbana, IL 61801 USA*
[4]adamlee@cs.uiuc.edu

***Abstract*— Grids are intended to enable cross-organizational interactions which makes Grid security a challenging and non-trivial issue. 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 there is a trade-off between flexibility and security in the context of delegation. 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 application; on the other hand, delegating all rights provides maximum flexibility but increases exposure. In this paper, we propose an on-demand restricted delegation mechanism, aimed at addressing the shortcomings of current delegation mechanisms by providing restricted delegation in a flexible fashion as needed for Grid applications. This mechanism provides an ontology-based solution for tackling one the most challenging issues in security systems, which is the principle of least privileges. It utilizes a callback mechanism, which allows on-demand provisioning of delegated credentials in addition to observing, screening, and auditing delegated rights at runtime. This mechanism provides support for generating delegation credentials with a very limited and well-defined range of capabilities or policies, where a delegator is able to grant a delegatee a set of restricted and limited rights, implicitly or explicitly.**

## I. Introduction

Grid computing, since its emergence, has been widely regarded as a revolution in information technology. Grids provide mechanisms for the creation and management of large numbers of dynamic virtual organizations (VOs), spanning multiple heterogeneous organizations with different underlying security mechanisms [1][2].

In a Grid, users or applications may need access to resources that belong to different organizations. The jobs submitted by users of these systems may take long periods of time to execute, and resource providers usually require some form of authentication of users and authorization of requests, i.e., that a user is allowed to use the requested resources as described in the request. 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. However, the user may not be directly accessible either because of malfunction simply having disengaged after submission of a request for execution of an application. A common solution to the problem of disconnection is to delegate the authorization to an agent (program) that acts on the user's or application's behalf and that is less likely to be disengaged at any time [1].

Delegation is a key facility in Grids that makes possible an effective use of a wide range of dynamic Grid applications. However, the least privilege principle, stating that we should grant only those rights required to perform a required action to minimize vulnerability, is becoming one of the most important security aspects in regards to delegation. There are potential security risks associated with performing delegation in a way that delegated rights are not limited only to the task intended to be performed within a limited lifetime and under restricted conditions. Therefore, a fine grained policy for restricted delegation is highly desirable, yet care must be taken to not introduce unmanageable complexity.

However, performing restricted delegation in a static manner cannot meet all the requirements of dynamic execution of Grid applications, because the required access rights for completing a task cannot easily be anticipated in advance. Delegating fewer rights than required for completing the task may cause task execution to fail while delegating more rights than needed may threaten abuse by malicious parties. Therefore, utilizing a mechanism that allows determining and acquiring only required rights and credentials for completing a task, when they are needed, would be a more reasonable and robust approach for restricted delegation.

We will discuss later in this paper how current delegation approaches in Grids trade off between security and flexibility. We therefore conclude that a desirable delegation mechanism for Grid applications needs to balance the security and the

flexibility of delegation while minimizing dependencies on underlying security mechanisms used by diverse Grid systems.

In this paper, we propose a novel solution for the least privilege delegation in dynamic, distributed environments, which no existing solution adequately addresses. We describe an on-demand delegation mechanism that tackles the most challenging issues of dynamic restricted delegation. This mechanism exploits ontologies and the potential power of ontological queries for determining required access rights at run-time and utilizes a callback mechanism for acquiring corresponding credentials required for completing the task at on-demand. It should also be noted that this paper deals only with the architectural issues of this mechanism and does not describe in details the implementation.

Our paper proceeds as follows. In section II, we give an overview of our on-demand delegation framework. In section III, we describe related work aimed at addressing restricted delegation. In section IV we describe how ontologies can be exploited in our proposed framework. In section V, we provide a Grid job simulation scenario as a motivating example of the potential power of this approach and finally sections VI and VII describe the security considerations and current implementation status. Our conclusion and proposed future work are both described in section VIII.

## II. On-demand delegation

On-demand delegation is a novel approach in which a delegatee obtains additional rights (in terms of additional delegated credentials) as required and requested for acting on behalf of a delegator. 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.

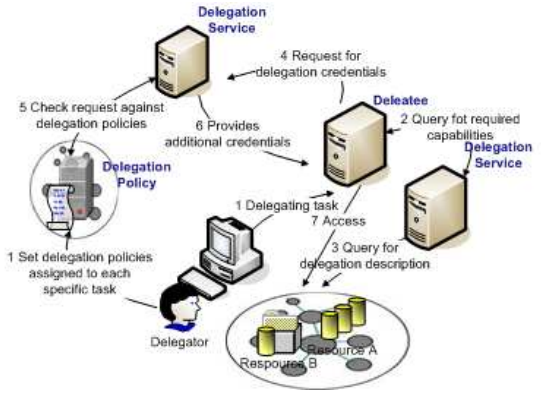


Fig. 1. An example of on-demand delegation

Fig.1 illustrates an example of performing on-demand delegation. Through this approach, a delegator initiates delegation by providing a least set of rights to a delegatee (job). This least set of rights only allows the delegatee to prove that it needs to act on behalf of its 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. The Delegatee contacts one or more Delegation Services to determine what credentials are needed and to then obtain them. In this simplest case, there is a Delegation Service associated with the Delegator and one associated with the Resource. The Delegation Service checks requests against policies established for each specific delegatee that specify the circumstances for issuance of additional credentials.

On-demand delegation allows a delegatee to leverage very simple ontological queries for determining the required access rights dynamically at run-time and to utilize a call-back mechanism to request required credentials from Delegation Services. Delegation policies are established to govern if additional credentials can be granted to a delegatee upon request and enable the delegator to keep full control over delegated rights. Ontologies are the powerful means for establishing these policies dynamically at the time of performing each delegation.

In fact, what an on-demand delegation mechanism provides is an efficient approach for performing restricted delegation in a dynamic fashion, because in practice we expect a small number of credentials to be required which can be determined by querying appropriate delegation ontologies. Ontological queries provide a very simple and efficient way of acquiring required access rights according to system descriptions.

We address the important concern of scalability by distributing Delegation Services across the Grid associated with different resources. Delegation Services need only know about local policies, and delegatees should need to query only a few Delegation Services in practice. This also provides manageability by providing local policy control points (the Delegation Service), and the semantic web community has shown [3] that ontologies can be used effectively for specifying policies in practice. By using call-backs, our on-demand framework never leaves a job stranded without credentials, but policy dictates the least privilege credentials that are delegated. When a job needs credentials that policy won't allow, we can put the job "on hold", notify the user, and wait for user intervention. If the user approves, the job can then proceed, thereby providing a very effective approach compare to other delegation mechanisms.

The approach presented in this paper is in some ways similar to the more general notion of trust negotiation [4] [5].In trust negotiation, two strangers carry out a bilateral and iterative exchange of attribute credentials to gradually establish trust in one another. We set out to solve the somewhat different problem of determining when to grant more privileges to a subjob 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 subjobs. The Delegation Service in our framework also bears some resemblance to the Traust service [6], which acts as

a stand-alone authorization service that uses trust negotiation to broker access tokens for resources in its security domain. Our Delegation Services, however, grant privileges only to a certain user's subjobs or provide policy-level information for resources in a particular security domain.

It should also be noted that for a system with a high level of granularity of access rights, making multiple calls to a Delegation Service could hurt performance. Performance evaluation and optimization is an important area for our future work, discussed in brief in section VIII.

## III. Related Work

There is significant prior work on restricted delegation for Grids and other distributed computing environments. In this section, we describe the prior work that we find most relevant to our own.

UNICORE[1], which is one of the widely used Grid infrastructures, addresses delegation and multi-site job execution by creating Sub-AJOs from a parent Abstract Job Object (AJO) [7]. In this approach, all components of an AJO are signed with the end-users' certificate at creation time, granting a limited set of rights to the specific job. This implies a secure but static delegation mechanism. "Explicit Trust Delegation" proposed in [7] is aimed to address this shortcoming by introducing trusted agents that are allowed to create and sign AJOs on behalf of end-users. By this approach end-users need to trust other agents for endorsing Sub-AJOs on their behalf and if job runs on a site which its server is not trusted, execution fails. More ever end-users loose their control on Sub-AJOs during the execution of task. Scalability also is an issue in this approach. In order to ensure that an endorser is authorized to endorse a Sub-AJO on behalf of end-user many information should be coded explicitly in authorization database at each site.

The Globus Security Infrastructure (GSI) implements delegation by means of "proxy certificates", which can provide full impersonation of end-users by granting all rights to a subordinate [8]. This provides a dynamic delegation mechanism, as there is no need to know the details of the execution in advance. However, this exposes all of the user's rights to possible compromise. Issuing short-lived proxy certificates is one solution for limiting the danger caused by unauthorized acquisition or usage of a proxy identity. Another approach is to use the PCI extension to carry a policy statement that limits the delegated rights [8].

The Community Authorization Service (CAS) [9] 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) [10] 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 delegation, considering that it is often difficult to determine the rights needed by a Grid job in advance.

The "Workflow-based Authorization Service" (WAS) [11] 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 the Grid opportunistically.

"Multiple Authorizations" is a concept suggested in [12] for restricted delegation 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 before delegating the task. In this approach, a protected operation cannot be executed unless the additional authorizations by other partners are provided. When a "work agent" needs to execute a delegated task, it has to collect approvals from all the corresponding "authorization agents" for the task. This approach utilizes a call back mechanism to ask for required grants on-demand, but the collection of required grants that authorizes task execution needs to be determined in advance.

Rein [13] 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 mainly be used for cryptographically asserting delegations for policy decision making. 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 delegation of least privileges in Grids.

None of above approaches address all the requirements of restricted delegation described earlier. In general, there is a compromising situation for addressing flexibility and security in the context of delegation. The most challenging issue of restricted delegation, namely dynamically anticipating access rights required for completing a task, is still unresolved. Some of these approaches have tried to address this challenge partially, though not through a dynamic and generic solution. All these approaches are strongly dependent on a specific

[1] http://www.unicore.org

and particular underlying authorization mechanism or Grid infrastructure. There is little support for observing and auditing of the delegation process that could be adapted to the dynamic requirements of Grid applications.

## IV. Delegation ontology

In computer science, an ontology is a conceptual schema that describes and classifying entities, the relationships between entities, and rules within a certain domain by means of a hierarchical data structure. It implies a more specialized schema for making the data useful for making real-world decisions. In this sense, Tom Gruber and R. Studer describe ontology as "an explicit and formal specification of a conceptualization". The Semantic Web[2] is a direct extension of the current Web to the explicit representation of knowledge by giving meaning and semantics, in a manner understandable by machines, to the content of documents on the Web. In Grids there are also many possible applications of knowledge-based problem-solving functionality which potentially can exploit ontologies. Therefore, the Semantic Grid[3] is also emerging to add Semantic Web capabilities to the Grid computing applications. We propose that on-demand delegation is a paradigm in which ontologies can support the automatic process of determining, requesting and delegating credentials.

In on-demand delegation, ontologies can be populated for sharing a common understanding of a delegation concept among different Grid systems with different underlying security mechanisms. The delegation ontology can provide a formal description of the delegation concept to be instantiated specifically for each administrative domain. This provides strong support for reusing and analyzing the domain knowledge required to meet the requirements of dynamic restricted delegation. In a general sense, on-demand delegation exploits ontologies for:

- Describing systems and resources which eventually result in determining and providing required credentials for access to resources and;
- Establishing delegation policies to automate the process of decision making.

This implies that the delegation ontology can be populated to describe the service provider's requirements for resource access as well as the Delegator's policies for performing delegation. Later in this paper we show how ontologies can be used in a real Grid usage scenario to describe the delegation mechanism in a particular system for enabling fine-grained access to protect resources in a highly descriptive fashion.

Ontologies also have a strong potential for making more efficient, adaptive, and intelligent queries on any system description. With ontologies in place, one can start adding reasoning capabilities for automatic disclosure of privileges according to delegation polices. Furthermore, ontologies make defining and managing delegation polices easier [14][15]. Ontology can be used for detecting conflicts and inconsistencies in the system description, which increase the risk of unauthorized access to resources. Even more, fulfillment of delegation policies could be assessed by analyzing resource usage and task definition information described using ontologies. Ontologies can also be used to determine the least privileges required to fulfill a request for delegation.

The delegation ontology depicted in Fig. 2, describes that each "Delegation" enables a "Delegator" to endow a "Capability" to a "Subject" under restricted conditions. "Credential" is also the means by which "Delegation" is authorized. It also depicts that 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, which implies a hierarchical delegation taxonomy in system description and consequently the need for further required delegation credentials.

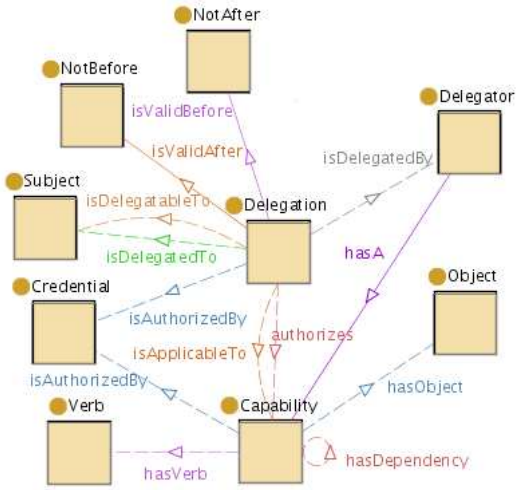


Fig. 2. Delegation Ontology

Fig. 2 is a simple illustration of a delegation ontology populated for on-demand delegation. 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 which have been filled in for instances and a knowledge base which is also 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" which makes an appropriate relation between the objects and verbs. It can also be used to specify individual instances of "Delegator" who can delegate 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.

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

## V. A GRID USAGE SCENARIO

In this section, we demonstrate the potential benefits of on-demand delegation by thoroughly discussing the example depicted in Fig.3. This picture describes a Grid scenario for job submission that utilizes on-demand delegation to perform a simulation job on the Grid. We describe in detail how on-demand delegation can be used to provide the job acting on behalf of Alice with the required delegation credentials for completing the task during its execution lifecycle.

Alice needs to submit an earthquake simulation job to the PDCGrid cluster. A Delegation Service is running on Alice's side to mediate between Alice and Alice's job for providing additional delegation credentials. The delegation service is aware of the domain delegation ontology that represents relationships between resources, users and delegates (jobs). This simulation job may need to get its input from a database on the NCSAGrid cluster, then perform a complex computation task on the SunetGrid cluster, and finally store the results in a database located in the PDCGrid cluster.

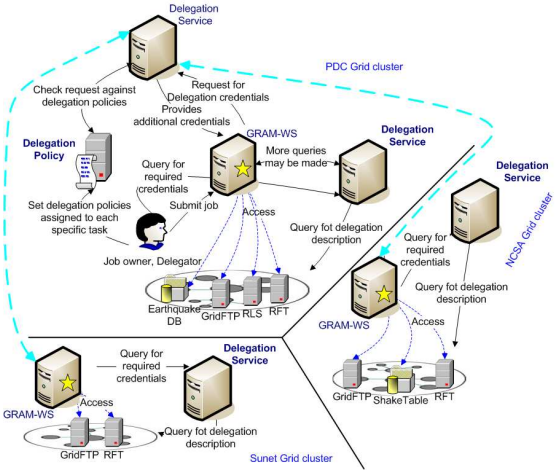


Fig. 3. An on-demand delegation scenario

Alice needs to generate an Independent Proxy Certificate (IPC), which is required for creating an independent identity for the submitted job with which to associate Delegation Service policies. These policies specify how further privileges could be linked later to this credential for enabling its owner to act on her behalf for completing a specific task. For example, such policies may specify where (on which site/cluster) this submitted job could be executed or for what operations on which services this job can act on behalf of Alice.

Alice creates an IPC and submits the job. A mutual authentication happens between Alice and the cluster's job manager. Once authentication is complete, the job manager will verify that Alice is authorized to submit the job. Alice also registers the IPC with the Delegation Service as a valid job's identity with permission to make use of her privileges to access resources.

Now the job has started on PDCGrid job scheduler. During its execution the job needs to locate some input data for the simulation process. First it needs to access a replica location service (RLS). The job sends a query to the Delegation Service running on NCSAGrid cluster to determine what permissions are required to invoke the RLS on behalf of Alice. This query is depicted in Fig. 4 and the result is depicted in Table I. This states that access to the RLS on behalf of Alice is authorized through a proxy certificate.

TABLE I
REQUIRED CREDENTIALS TO ACCESS RLS

| | |
|---|---|
| Delegator | Alice |
| Capability | Access_RLS |
| Dependency | NoCapability |
| Credential | X509PC |
| Object | RLS_SERVICE |
| EPR | http://grid.pdc.se/wsrf/RLSService |

```
SELECT   ?delegator  ?capability
         ?dependency ?credential
         ?object ?EPR
WHERE {
?delegator:isIdentifiedBy  ?iden.
?delegator:hasA  ?capability.
?capability:hasDependency ?dependency.
?capability:isAuthorizedBy ?credential.
?capability:hasObject ?object.
?object:isIdentifiedBy  ?objiden.
?objiden:EPR ?EPR.
FILTER regex(str(?iden), "Alice_DN") .
FILTER regex(str(?capability),
"http://grid.pdc.se/wsrf/RLSService")}
```

Fig. 4. Query on access to RLS service

The Delegation Service returns the signed result of the query back to the job. These requested privileges might be exposed as a portion of policy or even in terms of a specific role or attribute. It depends on the ontology that describes the delegation mechanism in each particular domain.

The job constructs a request message based on the information received from the Delegation Service and sends the request to the designated Delegation Service that performs delegation in Alice's domain. It authenticates the request by signing it with the IPC.

The Delegation Service receives the request. It verifies the request signature, parses the request and compares it against Alice's issuance policies, which she specified when she submitted the job. Ultimately, if the policy allows issuance of the requested credentials, the Delegation Service generates the credential and sends it to the job.

Now Alice's job can authenticate to the RLS and use the service to locate its input data. The RLS determines that input data can be obtained from the ShakeTable on the NCSAGrid cluster. However, to obtain the input data it may need to determine the required credentials and request them from the Delegation Service again.

The job sends a query to the Delegation Service on the NCSAGrid cluster to determine what credentials are needed

TABLE II

REQUIRED CREDENTIALS FOR READING THE INPUT

| Delegator | Alice | Alice | Alice |
|---|---|---|---|
| Capability | Read_DB | Read_DB | Read_DB |
| Dependency | Access_RFT | Access_ GrdiFTP | Read_DB |
| Credential | X509PC | X509PC | UserNameToken |
| Object | RFT_ Service | GridFTP_ Service | Shake_Table |
| EPR | http://ncsa. teragrid.org/ RFTService | http://ncsa. teragrid.org/ GridFTP Service | gsiftp:// ncsa.teragrid. org/shake/1352 |

TABLE III

REQUIRED CREDENTIALS FOR COMPUTATION JOB

| Delegator | Alice |
|---|---|
| Capability | Execute_JOB |
| Dependency | NoCapability |
| Credential | SAMLAssertion |
| Object | JOBProcess |
| EPR | null |

TABLE IV

REQUIRED CREDENTIALS FOR WRITING THE OUTPUT

| Delegator | Alice | Alice | Alice |
|---|---|---|---|
| Capability | Write_DB | Write_DB | Write_DB |
| Dependency | Access_RFT | Access_ GrdiFTP | Write_DB |
| Credential | X509PC | X509PC | UserNameToken |
| Object | RFT_ Service | GridFTP_ Service | EarthQuake_Table |
| EPR | http://grid. pdc.se/ RFTService | http://grid. pdc.se/ GridFTP Service | gsiftp://grid. pdc-se. /quake/1445 |

TABLE V

CONSTRAINTS OF DELEGATION CREDENTIAL

| Subject | Job_1 |
|---|---|
| Capability | Submission_JOB |
| ValidNotBefore | 2006-04-10T12:00:00 |
| ValidNotAfter | 2006-04-11T12:00:00 |
| isApplicableTo | NoCapability |
| isDelegatableTo | NoSubject |
| Dependency | NoCapability |

to authorize the capability "Read_DB" on behalf of delegator Alice. This query is depicted in Fig.5. The Delegation Service queries the ontology and sends the result back as it is depicted in Table II.

```
SELECT ?delegator ?capability ?dependency
       ?credential ?object ?EPR
WHERE {
?delegator:isIdentifiedBy  ?iden.
?delegator:hasA  ?capability.
?capability:hasDependency ?dependency.
?dependency:isAuthorizedBy ?credential.
?dependency:hasObject ?object.
?object:isIdentifiedBy  ?objiden.
?objiden:EPR ?EPR.
FILTER regex(str(?EPR),
"gsiftp://ncsa.teragrid.org/shake/1352")
FILTER regex(str(?iden), "Alice_DN") .
FILTER regex(str(?capability),"Read_DB")}.
```

Fig. 5. Query on READ_DB

The results show the required access rights and associated credentials to read data from the ShakeTable on behalf of Alice. It also illustrates how the attribute "Dependency" enables an automatic reasoning for determining all required access rights only by one query. The job requests the credentials again from Alice's Delegation Service as described earlier. Once Alice's job has obtained the required delegation credentials from Alice's Delegation Service, it can get its input data and continue on to completing its task.

This process continues to obtain all the required credentials iteratively until the job execution has completed. Table IV shows the results of appropriate queries made to Delegation Services to determine what credentials are required to authorize Alice's job to perform the computation task and store output data in a proper location.

Ontologies are also utilized by Alice's Delegation Service to decide when a credential should be issued. Fig. 6 shows a sample query to answer the question, "What are the parameters and constraints of delegation which authorize Alice's job, identified by JOB_1_DN, for job submission?".

```
SELECT ?Subject ?Identifier ?capability
       ?validNotBefore ?validNotAfter
       ?isApplicableTo ?isDelegatableTo
WHERE {
?cap:authorizes ?capability.
?delegaion:isDelegatedTo ?Subject.
?idn:isIdentifiedBy ?Identifier.
?nb:isValidNotBefore ?validNotBefore.
?na:isValidNotAfter ?validNotAfter.
?ap:isApplicableTo ?isApplicableTo.
?de:isDelegatableTo ?isDelegatableTo.
FILTER regex(str(?capability),
"Submission_JOB"
FILTER regex(str(?Identifier),
"Job_1_DN").}
```

Fig. 6. Query on delegation description

The results of the query in Table V specify the constraints that should be considered when a delegation credential is issued for Alice's job identified by "JOB_1_DN".

## VI. SECURITY CONSIDERATIONS

We must recognize that any form of delegation entails some risks that we cannot eliminate completely. If the job runs at a compromised site, it may be hijacked and its credentials may be misused. However, we believe that on-demand restricted delegation provides mechanisms to help us manage these risks.

In particular, the callback mechanism enables us the possibility of detecting misbehaviors and raising appropriate alarms to indicate that a job might be hijacked and compromised. The Delegation Service can log what credentials the job obtained, so after the compromise is detected, we can determine how the compromise spread and what credentials need to be revoked. Even more, we can consider:

- An exception mechanism for handling unexpected situations by blocking the job, notifying the user, and waiting for the user's decision on how to proceed, or by granting the rights so the job can proceed, but notifying the user, and letting the user terminate the job if it has overstepped its bounds.
- A learning mechanism that can be used to modify the policies according to the credentials that the job required when it last ran and even build up the policies by leveraging an interactive mechanism which involves the owner's approval for provided credentials over and over until all needed credentials are acquired.

These mechanisms assist the user in building up restricted policies to limit the vulnerability of delegated credentials and allow the user to monitor jobs to detect when changes in policy are required or when jobs are misbehaving.

## VII. Implementation and validation

Currently, we demonstrate the feasibility of our approach through a prototype implementation, which we have successfully tested with a simple Grid application. In our test scenario we have implemented a client program for performing a third party data transfer in Grids, which is run with no pre-generated proxy certificates. It further obtains credentials to use with GridFTP [16] services on demand and solely for the particular files specified by its command line arguments. For this we have implemented a Delegation Service and a communication protocol for requesting and exchanging credentials as described in [17]. In our implementation we have used the Ontology Web Language (OWL) [18] for creating delegation ontology which includes descriptions of classes, properties and the instances of delegation ontology described earlier. The SPARQL query language [19] is also used to make efficient, adaptive, and intelligent queries on system description and delegation ontology. We have also used the Protege OWL_API [20] to develop a set of software components for generating, parsing and evaluating queries and also populating and instantiating ontologies.

## VIII. Conclusion and future work

In this paper we claimed that the lack of a flexible least privilege delegation mechanism necessitates the design of an on-demand delegation framework. We believe that provisioning restricted delegation in a flexible way is the most significant and challenging issue for delegation in Grids.

We proposed an on-demand delegation framework that utilizes a callback mechanism for provisioning of restricted credentials containing only the rights that are actually needed. We described how ontologies can be utilized for determining the access rights required by a delegatee to complete a task and further how a Delegation Service can use ontologies for performing delegation. Approaching on-demand delegation has benefits of real-time control and auditing at the Delegation Service. However, it may hurt performance if it requires multiple callbacks to the Delegation Service for obtaining rights. In this regard, one strategy to optimize on-demand delegation would be delegating more rights to the job either at the time of launching or during each callback, so that a delegatee does not have to callback to a Delegation Service so often. Determining this set of rights in an optimal way is a challenging issue that can be considered part of the future work of this research.

Rein [13], the policy framework described earlier, supports an ontology-based mechanism for describing delegation and therefore it has certainly strong potential to be used in our proposed on-demand delegation. Then one future work would be investigating on how to use Rein policy framework in this framework. The level of granularity of resource and system description and access rights can also affect the complexity of restricted delegation and even the security of the whole system. Therefore, one additional target point of future work would be the optimization of the delegation description to find an optimized level of granularity that yields a less complex and more usable system without significantly compromising the security.

## Acknowledgment

Adam J. Lee was supported by a Motorola Center for Communications graduate fellowship and by the NSF under grants IIS-0331707, CNS-0325951, and CNS-0524695.

## References

[1] D. A. Reed, C. L. Mendes, C. da Lu, I. Foster, and C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure - Application Tuning and Adaptation*, 2nd ed. San Francisco, CA: Morgan Kaufman, 2003, ch. 16.

[2] 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)*. Washington, DC, USA: IEEE Computer Society, 2003, p. 48.

[3] T. Leithead, W. Nejdl, D. Olmedilla, K. E. Seamons, M. Winslett, T. Yu, and C. C. Zhang, "How to exploit ontologies for trust negotiation," in *ISWC Workshop on Trust, Security, and Reputation on the Semantic Web*, ser. CEUR Workshop Proceedings, vol. 127. Hiroshima, Japan: Technical University of Aachen (RWTH), Nov. 2004.

[4] J. Basney, W. Nejdl, D. Olmedilla, V. Welch, and M. Winslett, "Negotiating trust on the grid," in *Semantic Grid: The Convergence of Technologies*, ser. Dagstuhl Seminar Proceedings, 2005.

[5] 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.

[6] A. J. Lee, M. Winslett, J. Basney, and V. 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: ACM Press, 2006.

[7] D. F.Snelling, S. van den Berghe, and V. Qian, "Explicit trust delegation: Security for dynamic grids," *FUJITSU Sci.Tech.Journal*, vol. 40, pp. 282–294, 2004.

[8] V. Welch, I. Foster, K. C. M. O., P. L., T. S., G. J., and M. S. S. F., "X.509 proxy certificate for dynamic delegation," in *Proceedings of the 3rd Annual PKI Workshop*, Gaithersburg MD, USA, April 2004, pp. 20–25.

[9] 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)*. Washington, DC, USA: IEEE Computer Society, 2002, p. 50.

[10] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, A. Gianoli, K. Lorentey, and F. Spataro, "Voms, an authorization system for virtual organizations." in *European Across Grids Conference*, 2003, pp. 33–40.

[11] S.-H. Kim, J. Kim, S.-J. Hong, and S. Kim, "Workflow-based authorization service in grid," in *GRID '03: Proceedings of the Fourth International Workshop on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2003, p. 94.

[12] G. Vogt, "Delegation of Tasks and Rights," in *Proceedings of the 12th Annual IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 2001)*, INRIA, Ed., IFIP/IEEE. Nancy, France: INRIA Press, Oct. 2001, pp. 327–337.

[13] L. Kagal, T. Berners-Lee, D. Connolly, and D. 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)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 205–214.

[14] L. Kagal, T. Finin, and A. Joshi, "A Policy Based Approach to Security for the Semantic Web," in *2nd International Semantic Web Conference (ISWC2003)*, September 2003.

[15] A. Toninelli, J. Bradshaw, L. Kagal, and R. 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.

[16] R. K. Madduri, C. S. Hood, and W. E. Allcock, "Reliable file transfer in grid environments," in *LCN '02: Proceedings of the 27th Annual IEEE Conference on Local Computer Networks*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 737–738.

[17] M. Ahsant, J. Basney, and O. Mulmo, "Grid delegation protocol," in *Proceedings of the Workshop on Grid Security Practice and Experience (UK e-Science Security Task Force)*, Oxford, UK, July 2004, pp. 81–91.

[18] (2004) Owl web ontology language reference, w3c recommendation. [Online]. Available: http://www.w3.org/TR/owl-features/

[19] (2004) Rdf data access working group. sparql query language for rdf. [Online]. Available: http: //www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/

[20] B. R. Volz and P. Lord, "Cooking the semantic web with the owl api," in *Proceedings of International Semantic Web Conference*, Sanibel Island, 2003, pp. 659–675.