Grid Computing Security through Access Control Modelling

Submitted By
G. Geethakumari

For the Degree of

Doctor of Philosophy
in
Computer Science

Department of Computer & Information Sciences
University of Hyderabad
Hyderabad - 500046
INDIA
February 2010

Grid Computing Security through Access Control Modelling

C E R T I F I C A T E

This is to certify that the thesis work entitled "Grid Computing Security Through Access Control Modelling" submitted by G Geethakumari (Regd No. 03MCPC10) in fulfillment of the requirement for the award of the degree of Doctor of Philosophy (Computer Science) of the University of Hyderabad, is a record of bonafide work carried out by her under our supervision.

The matter embodied in this thesis has not been submitted for the award of any other degree/diploma of any other Institute/University.

Dr Atul Negi
(Supervisor)
Reader
Department of Computer &
Information Sciences (CIS)
University of Hyderabad
Hyderabad – 500046, INDIA

Dr V N Sastry
(Supervisor)
Assoc.Professor
Institute for Development & Research
in Banking Technology (IDRBT)
Castle Hills,Masab Tank
Hyderabad – 500057, INDIA

HEAD
Department of Computer &
Information Sciences (CIS)
University of Hyderabad
Hyderabad – 500046, INDIA

DEAN
School of Maths. & Computer and
Information Sciences
University of Hyderabad
Hyderabad – 500046, INDIA

I dedicate this thesis

to my father Sri. G.Gopalan Nair

who is the guiding force in my life

G. Geethakumari

DECLARATION

I, **G. Geethakumari**, hereby declare that the work presented in this thesis has been carried out by me under the supervision of **Dr. Atul Negi**, Reader, Department of Computer and Information Sciences, University of Hyderabad, Hyderabad, India and **Dr. V. N. Sastry**, Associate Professor, IDRBT, Hyderabad, India, as per the Ph.D ordinances of the University. I declare, to the best of my knowledge, that no part of this thesis has been submitted for the award of a research degree of any other University.

G. Geethakumari

Reg No: 03MCPC10

ACKNOWLEDGEMENTS

During the course of my research program, I had the privilege to get help from many people. First and foremost, I would like to express my heart-felt gratitude to my supervisors **Dr Atul Negi** and **Dr V.N. Sastry**. The support and encouragement I received from my supervisor Dr Atul Negi has been exceptional. He advised me to be a good listener first and then a speaker. Throughout my research program, he showed me how one can be a good and effective researcher by imparting advice on all technical issues. He has always shown confidence in me. The best thing, apart from many other things, that I could learn from him is the importance of perseverance. Also throughout my research, he has been very particular about having hands-on exposure to the technical concepts. My second supervisor Dr V.N. Sastry, ensured that this work had sound mathematical foundation. He was always keen to know about the progress of the work. Apart from this, he has also been very helpful in providing me very good infrastructure and resources. He always spared time for discussion and encouragement. In all, both my supervisors contributed a lot in building my confidence as a good researcher.

I would like to express my gratitude to **Prof. Arun Agarwal**, Head of the Dept,DCIS, for his support during the course of my thesis work. I am thankful to **Prof Hrushikesha Mohanty**, Member, DRC, for his invaluable suggestions which helped me in consistently improving my work. I am also grateful to **Dr. Rajeev Wankar**, DRC Member, for his encouragement and also for ensuring that we had an excellent Grid Computing Laboratory with the latest computing facilities. My special thanks goes to **Sri Arvind Sharma**, former Director, IDRBT. He has been very kind and supportive. He believed that research talent has to be nurtured. I would like to thank the staff of IDRBT for their encouragement throughout my research. I would also like to thank IDRBT for its fellowship, without which it would have been very difficult for me to pursue this research. I am thankful to

my colleagues and friends - **Debabrata Naik**, **Pradeep Kumar**, **Pravin Metkevar**, **Ramanarayan**, **Shipa Shrivatsava**, **Padmaja**, **Kavita**, **Rajeswari**, **Srikanth**, **Prasanna** and many others who have been a continuous source of support and have made my interaction with them memorable. Directly or indirectly, they have contributed to the completion of this work.

I would like to express my gratitude to the National Institute of Technology (NIT), Warangal for their encouragement and support to research during my two year stint as a Computer Science faculty with the Institute. The NIT environment helped me in polishing my technical skills. I am also thankful to my present Institution, BITS-Pilani, Hyderabad Campus for their support. I am sure, with the kind of initiatives and research facilities here, I can scale greater heights in my research pursuits.

I am deeply indebted to my family for their love and support. This thesis is dedicated to my father **Sri G Gopalan Nair**. He has always been confident about my academic abilities. He never imposed anything on me and always inspired me to do better. It is from him that I have imbibed the never-say-die attitude in life. Inspite of many difficulties I could complete this work and the credit goes to my husband **Mr. M Deviprasad** and my loving daughter **Aishwarya**. I am really proud to have them as my family. I am also grateful to my mother **Saroja Nair** and my only sister **Jyothi** for their care and support.

G. Geethakumari

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Algorithms

ABSTRACT

The term "grid" (in the context of computing systems) refers to systems and applications that integrate and manage sharing of resources and services distributed across multiple administrative domains. Performance with scalability, optimum utilization of resources, efficient management and reliability and virtualization are some of the advantages of grid-enabled applications. Most of the existing research on grid security is focussed on authentication. Access control and authorization is a relatively less explored area, although very important for grid security. The potential of access control as a security mechanism in grids is one of the major motivating factors behind this work.

Design and development of new models for access control and authorization, specific to grid environment and its security requirements forms the core of this thesis work. The thesis proposes models for representing various access control components of grid security and suggests algorithms to show the association between these components. Our study on the access control and authorization requirements of grid systems helped us to identify the core grid security issues as grid-wide access control and authorization, indirect authorization for the geographically spread users and resources, dynamic authorization and fine-grained access control of the grid resources.

Since a grid-wide mechanism to secure the grid resources does not exist, we proposed an RBAC (Role-Based Access Control)-oriented grid authorization model for the same. This framework includes single-domain (or intra-domain) and cross-domain (or inter-domain) authorization architectures. To facilitate cross-domain authorization, we need to map the role of a given domain to its equivalent role in another domain. A role-mapping architecture is thus proposed. Role-ranking algorithms govern the working of this architecture. We implemented the grid authorization model for single-domain and cross-domain enforcement of access control. Lightweight Directory Access Protocol (LDAP) was used to represent additional user attributes apart from roles. We proposed a Role-Based Grid Delegation Model (RB-GDM) for enforcing indirect authorization based on the user's role. RB-GDM includes frameworks for delegation within a domain as well as across the domains. To enable the grid users to perform dynamic authorization, we proposed a fuzzy-based scheme called Fuzzy Trust and Delegation Model (FTDM). Finally, a fine-grained authorization framework was presented to introduce finer access control aspects in grid resource access. We implemented the models using programming techniques and fuzzy inference mechanisms.

CHAPTER I

INTRODUCTION

In this chapter, we present an introduction to grid computing systems and grid security. Then, we discuss various access control models and their features. We address the issues of grid security and access control and the proposed approaches to solve them. The organization of the thesis is presented at the end.

The idea of grid computing is widely regarded as a concept of immense potential in both industry and academia. Due to lack of consistent and widely used standards, several enterprises are concerned about the implementation of an enterprise-level grid system, though the potential of such a system is well understood. Issues related to application engineering, manageability, data management, licensing, security etc. have prevented implementation of an enterprise-wide grid solution [130]. As an issue, security is most important and needs close understanding as grid computing offers unique security challenges. Some of which are of immediate concern and some are long term issues. Issues pertaining to grid computing security can be architecture-related, infrastructure-related and management related. We focus on the architecture-related issues.

1.1 Grid Computing Systems

A grid is an environment that allows service oriented, flexible and seamless sharing of heterogeneous network of resources for compute intensive and data intensive tasks and provides faster throughput and scalability at lower costs [72]. The distinct benefits of using grids include *performance with scalability*, *resource utilization*, *management and reliability* and *virtualization*. Grid computing environment provides more computational capabilities and helps to increase the efficiency and scalability of the infrastructure [33]. Many enterprises require flexible and scalable infrastructure [73]. Therefore most of the applications running on the grid infrastructure are compute-intensive or batch type applications. Another grid requirement is the need to use the resources more efficiently. The ability to share

resources across geography makes the grid really attractive. A grid can harness the idle processing power available in computing systems located in different time zones.

Grid computing provides a single interface for managing the heterogeneous resources [44]. Another benefit of grid computing is that it can create a more robust and resilient infrastructure through the use of decentralization, fail-over and fault tolerance to make the infrastructure better suited to respond to minor or major disasters. The grid provides virtualization of heterogeneous resources resulting in better management of the resources.

Ian Foster is considered as one of the earlier proponents of grid technology. Foster defines a grid as: " A Computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high end computational capabilities" [72]. Later, Foster redefined the grid as "a computing environment concerned with the coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations".

Many of the organizations who plan to set up grid-enabled applications perceive the grid concept differently [165]. Accordingly, Oracle describes its grid vision [8] as an adaptive software infrastructure which is able to balance resources efficiently through the usage of low cost servers and storage. Sun Microsystems classifies the grid on three levels namely cluster grids, enterprise grids and global grids [6]. HP talks about grid as a utility computing environment [5] while IBM describes a grid as a autonomic computing environment having a collection of resources which create dynamic virtual organization [39], [160], [189]. From the point of industrial implications, the Gridbus project came up with the following definition for grid. Accordingly, a grid is a type of parallel and distributed system that enables the sharing of autonomous resources dynamically ensuring user's quality of service [177].

The Internet and the grid, though both are distributed systems, have their own

differences as well. The Internet is a network of communication whereas grid computing is seen as a network of computation [23] which provides tools and protocols for sharing of a variety of resources. Grid computing is also known by a number of other names such as, "grid", "computational grid" [31], "computing-on-demand" and "on-demand-computing" [43]. The term grid technology describes the entire collection of grid computing elements, middleware, networks and protocols [161].

In a grid environment, the ensemble of resources is able to work together cohesively because of defined protocols that control connectivity, coordination, resource allocation, resource management and security [164]. Generally the protocols are implemented in the middleware. The systems "glued" together by a computational grid may be in the same room, or may be distributed across the globe; they may be running on homogenous or heterogeneous hardware platforms; they may be running on similar or dissimilar operating systems; and they may be owned by one or more organizations. The goal of grid computing is to provide users with a single view and/or single mechanism that can be utilized to support any number of computing tasks. The grid leverages its extensive informatics capabilities to support the "number crunching" needed to complete a task and all that the user perceives is essentially a large virtual computer undertaking the user's work [144].

According to Ian Foster, the essence of grid computing can be captured in a simple checklist [127], according to which a grid is a system that:

- ***coordinates resources that are not subject to centralized control*** (A grid integrates and coordinates resources and users that live within different control domains, for example, the user's desktop vs. central computing; different administrative units of the same company; or different companies; and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings).

- ***using standard, open, general-purpose protocols and interfaces*** (A grid is built from multi-purpose protocols and interfaces that address such fundamental issues as authentication, authorization, resource discovery, and resource access. These protocols and interfaces need to be standard and open

so that the system is able to execute generic applications).

- ***to deliver nontrivial qualities of service*** (A grid allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating - for example - to response time, throughput, availability, and security, and/or co-allocation of multiple resource types to meet complex user demands, so that the utility of the combined system is significantly greater than that of the sum of its parts).

The grid vision requires standard protocols, interfaces and policies that are not only open but also general-purpose [126]. These protocols allow us to establish resource-sharing arrangements dynamically with any interested party and thus to create compatible and inter-operable distributed systems. The definition of standard "InterGrid" [159] protocols is the single most critical problem facing the grid community today [175]. Global Grid Forum (Open Grid Forum) is a consortium working in this direction [7]. For implementation purposes, Globus Toolkit, an open source middleware is being widely used as the platform [192], [190].

1.1.1 Elements of Grid Computing

Grid computing combines elements such as distributed computing, high-performance computing and disposable computing depending on the application of the technology and the scale of operation [26]. Grids can create a virtual supercomputer out of the existing servers, workstations and personal computers.

Present-day grids encompass the following types

- Computational grids, in which machines will set aside resources to "number crunch" data or provide coverage for other intensive workloads
- Scavenging grids, commonly used to find and harvest machine cycles from idle servers and desktop computers for use in resource-intensive tasks (scavenging is usually implemented in a way that is unobtrusive to the owner/user of the processor)
- Data grids [62], which provide a unified interface for all data repositories in an organization, and through which data can be queried, managed and secured.

- Market-oriented grids [143], which deal with price setting and negotiation, grid economy management and utility driven scheduling and resource allocation.

The key components of grid computing include the following.

- Resource management: a grid must be aware of what resources are available for different tasks
- Security management: the grid needs to take care that only authorized users can access and use the available resources
- Data management: data must be transported, cleansed, parceled and processed
- Services management: users and applications must be able to query the grid in an effective and efficient manner [114]

More specifically, grid computing environment can be viewed as a computing setup constituted by a number of logical hierarchical layers. Figure 1 represents these layers. They include grid fabric resources, grid security infrastructure, core grid middleware, user level middleware and resource aggregators, grid programming environment and tools and grid applications.

The major constituents of a grid computing system [16] can be identified into various categories from different perspectives as follows:

- functional view
- physical view
- service view

Basic constituents of a grid from a *functional* view are decided depending on the grid design and its expected use. Some of the functional constituents of a grid are

1. Security (in the form of grid security infrastructure)
2. Resource Broker
3. Scheduler

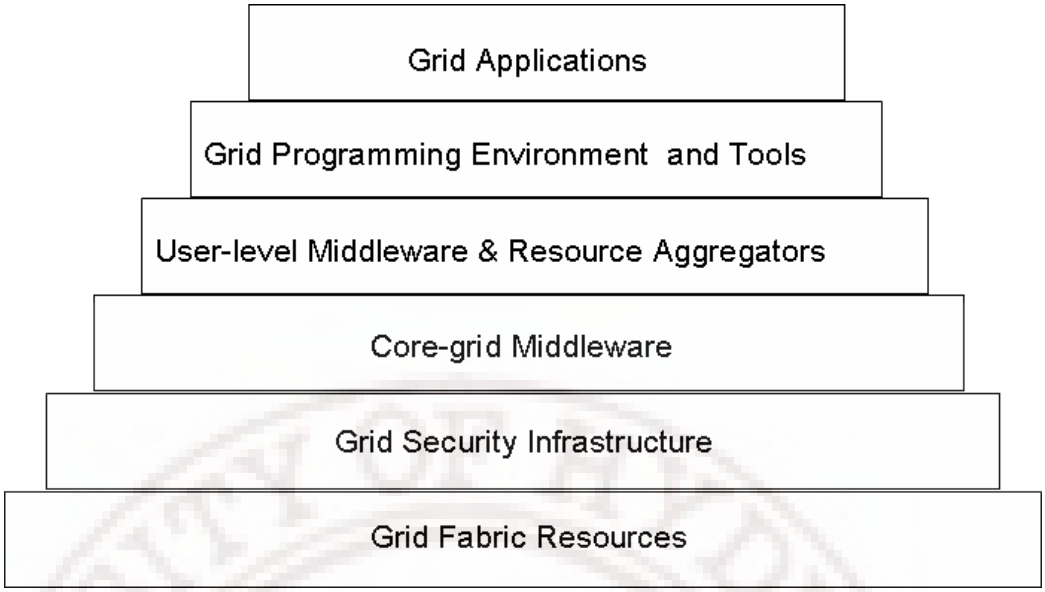


Figure 1: Overview of Grid Computing Layers

4. Data Management
5. Job and resource management
6. Resources

A *resource* is an entity that is to be shared [65]; this includes computers, storage, data and software. A resource need not be a physical entity. Normally, a grid portal acts as a user interaction mechanism which is application specific and can take many forms. A user-security functional block usually exists in the grid environment and is a key requirement for grid computing. In a grid environment, there is a need for mechanisms to provide authentication, authorization, data confidentiality, data integrity and availability, particulary from a user's point of view. In the case of inter-domain grids, there is also a requirement to support security across organizational boundaries. This makes a centrally managed security system impractical. The grid security infrastructure (GSI) provides a "single sign-on", run-anywhere authentication service with support for local control over access rights and mapping from global to local identities.

1.1.2 Grid Architecture

The architecture of a grid system is often described in terms of "layers", each providing a specific function as shown in the following figure. Higher layers are user-centric, whereas the lower layers are hardware-centric. In Figure 2, we depict a generic grid architecture showing the functionality of each layer.

Figure 2: Grid Architecture

- **Network layer:** is the bottom layer which assures the connectivity for the resources in the grid.
- **Resource layer:** is made up of actual resources that are part of the grid, such as computers, storage systems, electronic data catalogues, and even sensors such as telescopes or other instruments, which can be connected directly to the network.
- **Middleware layer:** provides the tools that enable various elements (servers, storage, networks, etc.) to participate in a unified grid environment.
- **Application layer:** which includes different user applications (science, engineering, business, financial), portal and development toolkits-supporting applications.

1.1.3 Main characteristics of Grids

The main characteristics of a grid computing environment can be listed as follows:

- **Large scale:** A grid must be able to deal with a number of resources ranging from just a few to millions.
- **Geographical distribution:** Grid resources may be spread geographically.
- **Heterogeneity:** A grid hosts both software and hardware resources that can be ranging from data, files, software components or programs to sensors, scientific instruments, display devices, personal digital organizers, computers, super-computers and networks.
- **Resource sharing and coordination:** Resources in a grid belong to different organizations that allow other organizations (i.e. users) to access them. The resources must be coordinated in order to provide aggregated computing capabilities.
- **Multiple administrations:** Each organization may establish different security and administrative policies under which resources can be accessed and used.
- **Accessability attributes:** Transparency, dependability, consistency, and pervasiveness are attributes typical to grid resource access. A grid should be seen as a single virtual computing environment and must assure the delivery of services under established Quality of Service requirements. A grid must be built with standard services, protocols and interfaces thus hiding the heterogeneity of the resources [195] while allowing its scalability. A grid must grant access to available resources by adapting to dynamic environments where resource failure is common.

1.1.4 Standard Bodies Related to Grid

The Global Grid Forum (GGF): is the community of users, developers and vendors leading the global standardization effort for grid computing. The OGSA (Open Grid Services Architecture), OGSI(Open Grid Services Infrastructure) and JSDL (Job Submission Description Language) standards were created by the GGF.

The GGF has two principal functions namely, the development of standards relating to grids and building of communities within the overall grid community.

1.1.5 Economic Aspects of Grid Systems

The economical aspects of grid computing has always drawn the attention of many researchers. The benefits gained from grid computing can translate into competitive advantages in the market place. Research contributions in the area of grid computing envision a future in which economically intelligent and economically motivated peer-to-peer and grid-like software systems will play an important role in establishing a distributed service-oriented computing paradigm. The aim of economic grid systems should be to deliver greater value to users than traditional systems [22]. For this, the economic-based resource management systems need to provide mechanisms and tools that allow resource consumers (end users) and providers (resource owners) to express their requirements and facilitate the realization of their goals [178], [24]. This means that they need

- the means to express their requirements, valuations, and objectives (value expression)
- scheduling policies to translate them to resource allocations (value translation)
- mechanisms to enforce selection and allocation of differential services, and dynamic adaptation to changes in their availability at runtime (value enforcement)

Grids need to use competitive economic models as different resource providers and resource consumers have different goals, objectives, strategies, and requirements that vary with time. Grids have the potential to

- enable resource sharing and provide transparent access to remote resources
- make effective use of computing resources, including platforms and data sets
- reduce significantly the number of servers needed by 25-75 % and allow on demand aggregation of resources at multiple sites

- reduce execution time for large-scale data processing applications
- provide access to remote databases and software and load smoothing across a set of platforms
- provide fault tolerance and also take advantage of time zone and random diversity
- provide the flexibility to meet unforseen emergency demands by renting external resources for a required period instead of owning them
- enable the realization of a virtual data center

Some applications of grid computing particularly in the scientific and engineering arenas include:

- Distributed super computing/computational science
- High-capacity/throughput computing:large-scale simulation/chip design and parameter studies
- Content sharing
- Remote software access/renting services like ASPs and web services
- Data-intensive computing like drug design , particle physics, stock prediction
- On-demand, real-time computing like mission critical initiatives
- Collaborative computing (e-science, e-engineering)
- Utility computing/service-oriented computing which involves new computing paradigm, new applications, new industries and new business

In this section, we have discussed the characteristics of grid computing environment, the elements of grid computing, the generic grid architecture, standard bodies related to grid and also the economic aspects of grid computing. In the next section, we discuss the basics of computer security and access control models, as we believe that a thorough understanding of these concepts will help us in framing an efficient grid security mechanism.

1.2 Computer Security and Access Control Models

Computer security is about protection of assets. The protection measures include *prevention* (taking measures to prevent assets from being damaged), *detection* (taking measures that allow one to detect when an asset has been damaged) and *reaction* (taking measures that allow one to recover assets from damage). The notion of computer security revolves around three core aspects namely *confidentiality*, *integrity* and *availability* [17]. Confidentiality is prevention of unauthorized disclosure of information. Integrity is about preventing unauthorized modification of information where as availability deals with prevention of unauthorized withholding of information or resources.

The three other aspects which are equally important for computer security are *accountability*, *non-repudiation* and *reliability*. Accountability refers to the audit information that must be maintained and protected so that actions affecting security can be traced to the responsible party. Non-repudiation services provide unforgeable evidence that a specific action has occurred. Reliability relates to accidental failures and *safety* relates to impact of system failures on their environment. *Dependability* can be stated as a concept unifying security, reliability, integrity and availability. It is the property of a computer system such that reliance can justifiably be placed on the service it delivers. We can say that dependability summarizes all the aspects of computer security.

1.2.1 Security Models

Implementing security for an organization needs well-framed security policies. A security policy is a statement of intent to protect an identified resource from unauthorized use [131]. Organizational security policy is a set of laws, rules and practices that regulate how an organization manages, protects and distributes resources to achieve specified security policy objectives. There are also automated security policies which address issues like the definition of access control lists or firewall settings, decisions on the services that may run on devices and the security protocols used to protect network traffic. To formulate a security policy, we have to describe the entities governed by the policy and we have to state the rules that constitute our policy. Security models play an important role in the design and

evaluation of high assurance security systems [88]. The design process of such systems starts from a formal specification of the policy which the system should enforce (security model) and a high level specification of the system itself [89]. By adding more details to this high level specification, we can arrive at a series of low level specifications. We need to show that the high level specification implements the desired policy. For high assurance, a proof is required.

A security model is a mathematical, or logical expression of a set of security policies [131]. A system can be secure only if its security model is based on logically sound premises as the security features are built into such components of a computer system like the operating systems, database systems, applications, etc on the basis of their security models. Security models are also used in security evaluation, sometimes as proofs of security. There are several well-known security models, such as the Bell-LaPadula (BLP) model and the Biba model which represent the concepts for controlling accessibility, integrity, etc. of information systems [17]. All information security models use the terminologies 'subject' and 'object'. A 'subject' is an entity, such as a person, process, or device which accesses or uses information from the system. An 'object' is the information, or a piece of a larger body of information, which is accessed by a 'subject'. An 'object' may be a 'subject' in another situation or context and vice versa.

The important types of information security models [18] are

- Access Control models
- Integrity models
- State machine models
- Information flow models
- Non-interference models

Different types of information security models use different philosophies for looking at subjects and objects, for grouping and classifying them and for controlling their interactions. A specific model, which may be a well-known model or a model designed for a particular organizational environment, usually has features from different types of information models. Among the earlier models, BLP

is a state machine model which represents the confidentiality aspect of computer security. Biba model and Clark-Wilson model were designed to represent the integrity aspect of security. The Chinese Wall model represents dynamically changing access rights. The Harrison-Ruzzo-Ulman (HRU) model defines authorization systems that represent policies for changing access rights or for the creation and deletion of subjects and objects. Information flow models consider any kind of information flow, not only the direct information flow through access operations but also the indirect flow through covert channels. The following subsection describes some of the security models perceived as access control models.

Access means entry, approach, or privilege to a resource. Access control is a mechanism to secure resources from unauthorized use [15]. It constrains what a user can do directly as well as what the programs executing on the user's behalf are allowed to do. Access control models use a sets of rules, which permit or deny access for a subject to an object [108]. This ensures that information does not fall into wrong hands. The process involves a subject requesting for an object. The permission or denial of access to the object depends upon the 'right' that the subject possesses. The early years (1975 - 1985) saw the evolution of three major categories of access control models namely

- DAC (Discretionary Access Control) Model
- MAC (Mandatory Access Control) Model
- HRU (Harrison Ruzzo and Ullman) Model

The DAC Model consists of a set of Objects (O) , a set of Subjects (S) and an Access Matrix (A). Any element A[i,j] specifies the access which the subject *i* has to the object *j* . When the matrix is stored by columns it is called an Access Control List (ACL). When the matrix is stored by rows it is called a Capability List (CL). The DAC model had its own drawbacks. It does not provide real assurance on the flow of information in a system. Also it does not impose any restriction on the usage of information by a user once the user has received it. Objects depend on their owners to permit access to them for other users. Information can be copied from one object to another, so access to a copy is possible even if the owner of the original does not provide access to it.

As per the MAC Model, subjects and objects in a system have a certain classification. The read-up operation means a subject's integrity level must be dominated by the object being read whereas in write-down operation, a subject's integrity level must dominate the integrity level of the object being written. The MAC Model has its own drawbacks. The information flow can pass through covert channels in prohibited ways. There is no solution to the inference problem where the high level information is deduced by assembling and intelligently combining the low level information.

In the HRU Model, the protection system consists of a finite set of generic rights (R) and a finite set of commands (C). This model uses the DAC access matrix and has six primitive commands namely, enter R into (S,O), delete R from (S,O) , create subject S', create object O', delete subject S' delete object O'.

Originator Controlled Access Control (ORCON) [17] is an advancement over Mandatory and Discretionary Access Controls models, as both these models cannot handle environments in which the originators of documents retain control over them even after those documents are disseminated. In ORCON, a subject can give another subject rights to an object only with the approval of the creator of that object. Organizations that use categories grant access to individuals on a "need to know" basis. There is a formal, written policy determining who needs the access based on common characteristics and restrictions. The main features of ORCON are as follows.

- The owner of an object cannot change the access controls of the object
- When an object is copied, the access control restrictions of that source are copied and bound to the target of the copy
- The creator (originator) can alter the access control restrictions on a per-subject and per-object basis

In ORCON, the access control associated with the object is under the control of the originator and not the owner of the object. Possession equates to only some control. The owner of the object may determine to whom he or she gives access, but only if the originator allows the access. The owner may not override the originator.

1.2.2 Alternative Models

The decade 1985 - 1995 is called the period of alternative models which are categorized as

- Chinese Wall Model (CW)
- Task Based Authorization (TBA)
- Role Based Access Control Model (RBAC)

1.2.2.1 The Chinese Wall Model

Deriving its name from the Great Wall of China, this model deals with devising a set of rules such that no person (subject) can ever access data (objects) on the wrong side of the wall. The basis of the Chinese Wall policy is that people are only allowed access to information which is not held to conflict with any other information that they already possess. The access rule states that a subject *s* can access an object *o* only if *o* is in the same company data set as some object previously read by *s* . *o* belongs to a COI (Conflict of Interest) class within which *s* has not read any object. The information flow which causes COI should be prevented. The composition of objects according to the Chinese Wall Model is shown in Figure 3. The conflict of interest classes are A,B and C where as the company data sets are *f*,*g*,*h*.

1.2.2.2 Task Based Authorization

The Task Based Authorization deals with authorization of tasks rather than subjects and objects. These tasks may involve other tasks. The authorization is transient and it models the organizational structure.

1.2.2.3 Role Based Access Control

The standard discretionary access control and mandatory access control approaches cannot cater to an enterprise's access control requirements as the former is a user-discretion mechanism and the later an approach more suitable for Operating System security. The need to evolve a new model for enterprise-wide security lead to the development of Role Based Access Control Model or the RBAC. The Role

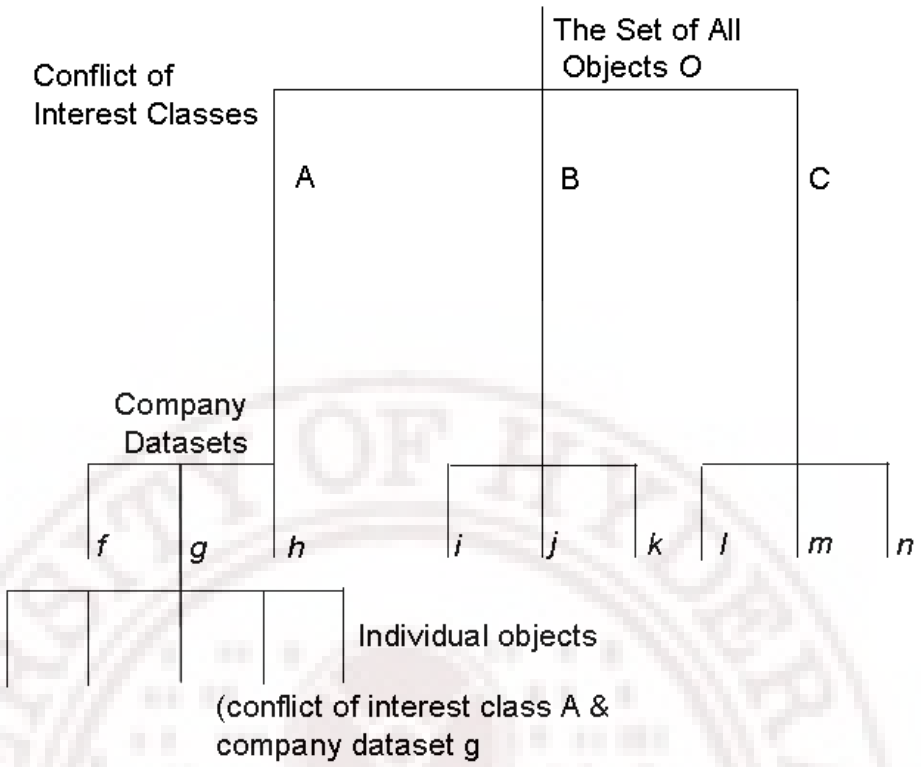


Figure 3: The Chinese Wall Model: Composition of Objects

Based Access Control Model is a technical means for controlling access to computer resources [155]. With role-based access control, access decisions are based on the *roles* that individual users have as part of an organization [174]. A *role* is the standard unit of access control in RBAC and reflects the responsibilities of a user in an organization. Users take on assigned roles (such as doctor, nurse, teller, manager). The process of defining roles should be based on a thorough analysis of how an organization operates and should include input from a wide spectrum of users in an organization. Access rights are grouped by role name, and the use of resources is restricted to individuals authorized to assume the associated role [14]. For example, within a hospital system the role of doctor can include operations to perform diagnosis, prescribe medication, and order laboratory tests; and the role of researcher can be limited to gathering anonymous clinical information for study.

The National Institute of Standards and Technology proposed the following standards for RBAC models [70]

- Flat RBAC (Core-RBAC)

- Hierarchical RBAC
- General Hierarchical RBAC
- Restricted Hierarchical RBAC
- Constrained RBAC
- Symmetric RBAC

Core RBAC sets guidelines for the basic RBAC functionalities. Figure 4 represents the set of users, roles and operations and objects (together known as privileges) as per the core-RBAC standard. It also depicts the user-role assignment and role-permission assignment. Figure 5 shows the hierarchical relationship between

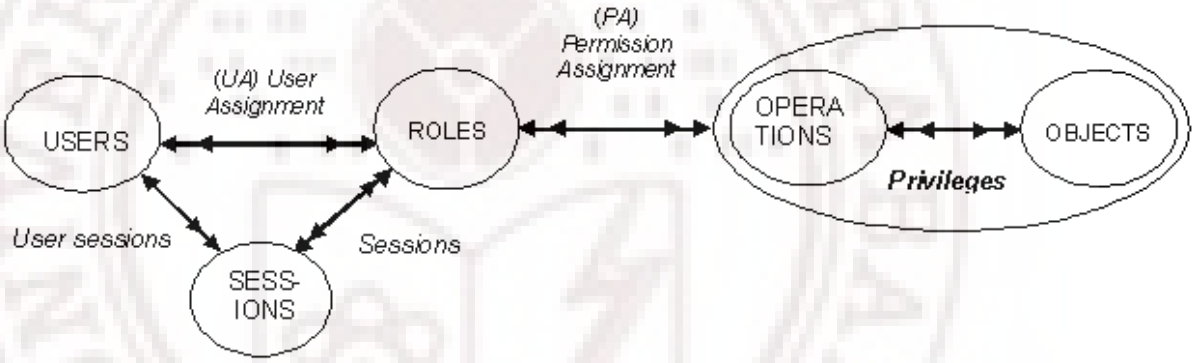


Figure 4: Core RBAC Model

various components of the RBAC model. This category of RBAC model is called hierarchical RBAC model.

Roles can be an effective means for developing and enforcing enterprise-specific security policies [150] [151] and for streamlining the security management process [163]. Under the RBAC framework, users are granted membership into roles based on their competencies and responsibilities in the organization [170]. The operations that a user is permitted to perform are based on the user's role. User membership into roles can be revoked easily and new memberships established as the job assignments dictate. Role associations can be established when new operations are instituted, and old operations can be deleted as organizational functions

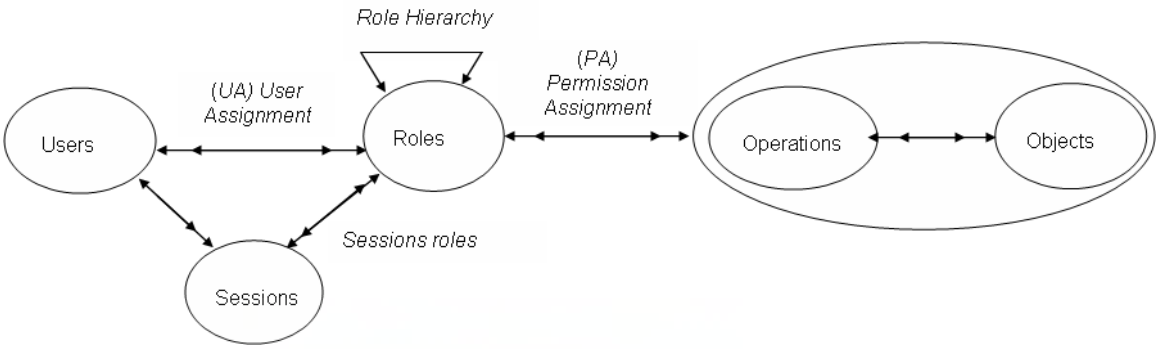


Figure 5: Hierarchical RBAC Model

change and evolve [111]. This simplifies the administration and management of privileges [162]; roles can be updated without updating the privileges for every user on an individual basis.

When a user is associated with a role, the user can be given no more privilege than is necessary to perform the job. This concept of least privilege [18] requires identifying the user's job functions, determining the minimum set of privileges required to perform that function and restricting the user to a domain with those privileges and nothing more. In less precisely controlled systems, this is often difficult or costly to achieve. Someone assigned to a job category may be allowed more privileges than needed because it is difficult to tailor access based on various attributes [11] or constraints [10], [191], [129]. Since many of the responsibilities overlap between job categories, maximum privilege for each job category could cause unlawful access.

Under RBAC, roles can have overlapping responsibilities and privileges; that is, users belonging to different roles may need to perform common operations. Some general operations may be performed by all employees. In this situation, it would be inefficient and administratively cumbersome to specify repeatedly these general operations for each role that gets created. Role hierarchies [83] can be established to provide for the natural structure of an enterprise. A role hierarchy defines roles that have unique attributes and that may contain other roles; that is, one role may implicitly include the operations that are associated with another role.

For example, in the healthcare environment, a role *Specialist* could contain the

roles of *Doctor* and *Intern*. This means that members of the role Specialist are implicitly associated with the operations associated with the roles *Doctor* and *Intern* without the administrator having to explicitly list the *Doctor* and *Intern* operations.

Role hierarchies are a natural way of organizing roles to reflect authority, responsibility, and competency. The role in which the user is gaining membership is not mutually exclusive with another role for which the user already possesses membership. These operations and roles can be subject to organizational policies [136] or constraints. When operations overlap, hierarchies of roles can be established. Instead of instituting costly auditing to monitor access, organizations can put constraints on access through RBAC. With RBAC, constraints can be placed on the physician's access to the health care data so that only those records that are associated with a particular physician can be accessed.

Organizations can establish the rules for the association of operations with roles. For example, a healthcare provider may decide that the role of clinician must be constrained to post only the results of certain tests but not to distribute them so as to protect the patient's right to privacy. Operations (permissions) can also be specified in a manner that can be used in the demonstration and enforcement of laws or regulations [173]. For example, a pharmacist can be provided with permissions to dispense, but not to prescribe, medication.

An operation represents a unit of control that can be referenced by an individual role, subject to regulatory constraints within the RBAC framework. An operation can be used to capture complex security-relevant details or constraints that cannot be determined by a simple mode of access.

For example, there are differences between the access needs of a teller and an accounting supervisor in a bank. An enterprise defines a teller role as being able to perform a savings deposit operation. This requires *read* and *write* access to specific fields within a savings file. An enterprise may also define an accounting supervisor role that is allowed to perform correction operations. These operations require read and write access to the same fields of a savings file as the teller. However, the accounting supervisor may not be allowed to initiate deposits or withdrawals but only perform corrections. Likewise, the teller is not allowed to perform any corrections once the transaction has been completed. The difference between these

two roles is the operations that are executed by different roles and the values that are written to the transaction log file.

The RBAC framework provides administrators with the capability to regulate who can perform what actions, when, from where, in what order, and in some cases under what relational circumstances. Only those operations that need to be performed by members of a role are granted to the role. Granting of user membership to roles can be limited. Some roles can only be assigned to a certain number of employees at any given period of time. The role of manager, for example, can be granted to only one employee at a time. Although an employee other than the manager may act in that role, only one person may assume the responsibilities of manager at any given time. A user can become a new member of a role as long as the number of members allowed for the role (cardinality) is not exceeded.

Apart from original roles and inherited roles, there is another way in which we can categorize roles. Based on whether a role has practically been assigned to a user, we can have two categories of roles namely enabled role and activated role. An *enabled role* is the one which is assigned to a user and ready for activation where as an *activated role* is the one which at least one subject/user has already activated it.

The advantages of RBAC are as follows. A properly-administered RBAC system enables users to carry out a broad range of authorized operations, and provides great flexibility and breadth of application [123]. System administrators can control access at an abstract level that is natural to the way that enterprises typically conduct business. This is achieved by statically and dynamically regulating users' actions through the establishment and definition of roles, role hierarchies, relationships, and constraints [104], [185]. Thus, once an RBAC framework is established for an organization, the principal administrative actions are the granting and revoking of users into and out of roles. This is in contrast to the more conventional and less intuitive process of attempting to administer lower-level access control mechanisms directly (e.g., access control lists [ACLs], capabilities, or type enforcement entities) on an object-by-object basis.

The decade 1995-2003 is called the era "After RBAC". The following models have come up during this period.

- Generalized Temporal RBAC (GTRBAC)
- Partial Outsourcing (PO)
- The Tees Confidentiality Model (TCM)

The Generalized - Temporal RBAC has the following features.

- It has a separate notion of role enabling and role activation
- It provides constraints and event expressions for enabling and activating a role

The types of constraints in GTRBAC [90] are Temporal on Role enabling/disabling and Temporal on user - role and role-permission assignments [53].

The Partial Outsourcing model [87] deals with Single Administration-Internal and Single Administration-External. The paradigm shifts to partial outsourcing. The access control administration is decentralized or jointly handled by different parties. In this model, the access control policies are motivated by the needs of different departments of the organization.

In the Tees confidentiality model [86] permissions are assigned to users irrespective of the roles. It gives override definitions for roles and permissions. As per this model, the permissions can also be inherited.

Table 1 shows the evolution of various access control models.

Table 1: Evolution of Access Control Models

Name of Access Control Model	Year of Evolution
MAC and DAC	1975-1985
Alternative Models (CW, TBA, RBAC)	1985-1995
After RBAC (GTRBAC, Partial Outsourcing	1995-2003
Recent Models (Domain Specific)	2003-Till Date

1.2.2.4 Comparison of Access Control Models

Access control models can be compared based on various factors [108] like:

- Expressive power of the models

- Inheritance of privileges
- Type of access control structures used
- Delegation of roles
- Constraints in permissions
- Existence of negative permissions
- Security analysis and evaluation
- Propagation of rights
- Facilities for Role Engineering
- Applicability in static and dynamic security environments
- Scalability
- Flexibility and ease of configuration [142]
- Implementation mechanisms
- Security assurance

Table 2 shows the comparison of RBAC model with non-RBAC models with regard to user privileges and cost comparison [78]. The table reflects the cost benefits for RBAC-oriented security mechanisms over non-RBAC mechanisms. The values given in the table are the estimated time (in minutes) required for performing access administration. We can observe a clear advantage for RBAC-oriented administrative tasks over the non-RBAC ones with regard to the time factor.

Table 2: Comparison of RBAC and Non-RBAC Models

Task	RBAC	Non-RBAC	Difference
Assign existing privileges to new users	6.14	11.39	5.25
Change existing users privileges	9.29	10.24	0.95
Establish new privileges for existing users	8.86	9.26	0.40
Termination of privileges	0.81	1.32	0.51

Though many models like the GTRBAC, Tees Confidentiality Model and the Partial Outsourcing Model have come up after RBAC, they were not as effective as RBAC to address the present day organizational security requirements. RBAC has proven to be an efficient model to represent and implement the organizational security policies.

1.2.2.5 Expressive power of the models

An access control system enforces a policy on *who may access what resources and in what manner*. Comparing the expressive power of access control models is recognized as a fundamental problem in computer security [113]. Such comparisons are generally based on simulations between different access control schemes. Policies are generally expressed in terms of the current state of the system and states that may result from prospective changes. The expressive power of access control models can be compared by perceiving the access control systems as state-machine systems. The expressive power of a model is related to the expressive power of the schemes from the model. In comparing schemes based on expressive power, we consider the types of policies that can be represented by the computing systems based on a scheme. If all policies that can be represented in scheme B can be represented in scheme A, then scheme A is at least as expressive as scheme B.

1.2.3 Access Control Approaches

Activity in a system is initiated by entities known as subjects. Subjects are typically users or programs executing on their behalf. A user may sign on to the system as different subjects on different occasions [153]. Subjects can themselves be objects. A subject can create additional subjects in order to accomplish its task. There are several approaches to access control modelling and implementation [115], [56], [119], [97]. Some of the approaches are procedure-specific where as some are domain-specific. We discuss these approaches in the following subsections.

1.2.3.1 Logic approach to Access Control

Access control is central to security and is pervasive in computer systems both from the hardware and from the software point of view. It may appear with peculiar features and flaws in many applications like firewalls, virtual machines , operating systems etc. We can rely on logical ideas and tools to improve access control [54], [112]. Application of logic in access control has been substantial and beneficial [117]. The relevant issues in access control logic [109] are

- Decidability results for problems related to access control
- Logical approaches for authorizing code execution
- Formal verification of security properties

According to Abadi et.al [49], a first logic of access control is obtained by using an access control matrix which is a ternary relation (*p*,*o*,*r*) where *p* is the principal, *o* is the object and *r* represents the right. For example, if we use a predicate symbol *may-access*, then *may-access(Alice, Foo.txt, Rd)* says that Alice can perform *read* operation on object *Foo*. Constructs similar to *may-access* can support a wide range of current models for access control. These logics may include primitives pertaining to groups, roles, object containment, privilege ordering etc. The possibility of an important role for logical methods lies in the case of distributed systems whose characteristics such as size, heterogeneity, autonomy of system components, the possibility of component failures make the access control more complicated.

1.2.3.2 Language based approach to Access Control

Languages for access control aim to support expression and enforcement of the access control policies. The languages are general and flexible. Recent language designs for access control rely on logic. Also many of the language based approaches are aimed at distributed systems. Binder [68] is a good representative of this line of work.

1.2.3.3 Access Control based on execution history

In access control mechanisms which use the basic concept of access control matrix, the association of a subject with an object is defined by the right which is nothing

but an *operation* performed by the subject on the object. The association of rights with the code is a delicate issue in systems which rely on access control for security. In such systems, the run-time rights of a piece of code can be decided based on execution history [9].

1.2.3.4 Access Control in the presence of mobility

The emerging mobile computing environment draws new attention to the need for coordination among the networked components. Parties interact even when they have never met before and subsequent encounters are totally unpredictable. Reliance on centralized servers to authenticate agents and to establish data access policies is impractical since mobile networks are often decoupled from any fixed network infrastructure. Access control is a key component of security in such systems [106].

1.2.3.5 Access Control in Collaborative Environments

The characteristics of a collaborative environment require mechanisms which can integrate diverse policies [64], [105]. Some of the issues in finalizing access control mechanisms for collaborative environments [48] are as follows.

- What needs to be protected in a collaborative system ?
  Hosts, resources, data, computations
- Access control languages and policies
- Do we need ad-hoc languages for specifying access control policies for the hosts? If so what would be the most relevant features of these languages?
- User requirements
  Different hosts may provide different levels of security. How can a user specify one's security requirements when running computations? What is the assurance that the requirements have been met?
- Scalability and evolution
  Collaborative systems may encompass a very large number of nodes (hundreds or even thousands). Moreover, they can be quite dynamic, with hosts

and clients dynamically joining and leaving. So we need to design scalable access systems which are able to cope up with the required dynamism.

1.2.3.6 Access Control Structures

There are various theoretical models on access control. But equally important is an access control structure, which the system can implement so as to effectively control access to objects. An access control matrix has many disadvantages. If the number of subjects and objects is sufficiently large then the matrix will use significant amount of storage. Another drawback is that most of the entries in the matrix may be either blank (indicating no access) or the same (because implementations often provide a default setting). Also the creation and deletion of subjects and objects will require the matrix to manage its storage carefully adding to the complexity of this code. Several optimization strategies enable computing systems to use more convenient and simpler versions of the access control matrix. Access control lists and capabilities are variants based on the access control matrix. Various access control structures include [131]:

- Access control lists
- Capability lists
- Layered access control
- Propagated access control lists

1.2.3.7 Access Control Lists

Each object is associated with a set of paired elements, with each pair containing a subject and a set of rights. Figure 6 shows the access control list representation. Each entry in an access control list (ACL) is equivalent to a column in the access control matrix.

1.2.3.8 Capabilities

Conceptually a capability is like the row of an access control matrix. Each subject has associated with it a set of paired elements, with each pair containing an object and a set of rights. The subject associated with this list can access the named object

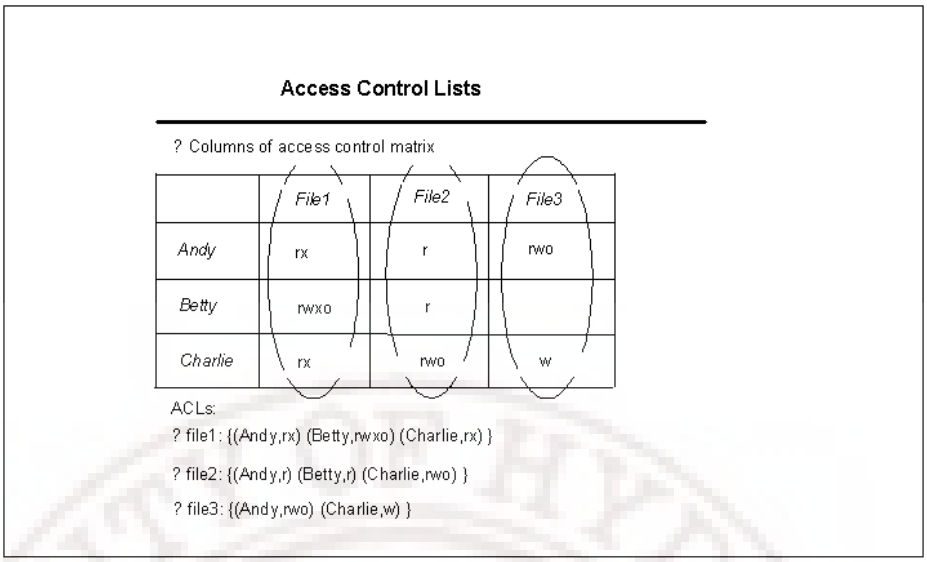


Figure 6: Access Control Lists

in any of the ways indicated by the named rights as shown in Figure 7. Each entry in a capability is equivalent to a row of the access control matrix.

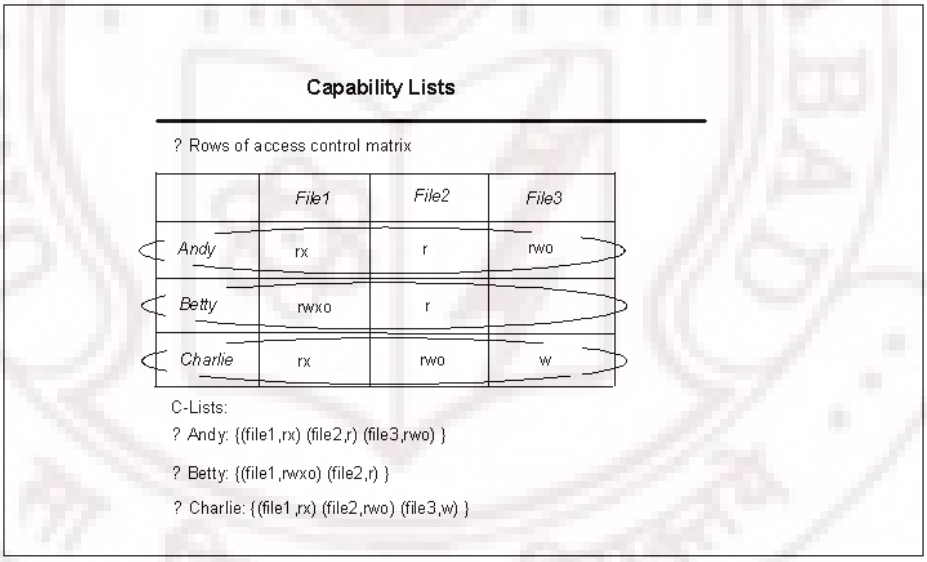


Figure 7: Capability Lists

1.2.3.9 Access Control Functions

An access control function takes five parameters: the tuple of attributes, the requesting agent's credentials, the operation, the pattern used in the request and the

owner's profile. Upon requesting operations, an agent sends a portion of its profile such as its ID, a third-party authentication etc as its credentials. This type of identification is necessary as mobile applications are open and dynamic where the coordinating agents appear and disappear sporadically. A set of credentials can be viewed as a tuple of attributes such as (att_ name,type,value).

The second parameter is operation. The possible operations are represented as a set O which contains string representations of the different operations like "rd","reaction","migration" etc. The third parameter is Requested Tuple. The access control function operates over the tuple to be returned from an operation. Tuples provide fine grained access control because access privileges are based on individual data items. The fourth parameter is pattern. This parameter is particularly important in content based operation. The pattern gives the owning agent information about the requesting agent's knowledge of the data it is attempting to access. The last parameter of access control function is owner's profile. It considers the owner's current state. This is very important as access policy is determined for dynamic environments.

1.3 Grid Computing Security

Computational grids combine computational and data resources across organizational boundaries. The sharing of code and data on the grid gives rise to the need for security. Security is the most challenging issue offered by grid systems due to its unique characteristics [85]. The desired characteristics of grid security include.

- the ability to verify the identity of an individual
- the capacity to allow or restrict access to some or all resources on the grid

Grid systems like Alchemi [51] and cloud initiatives like Aneka [32] discuss security in terms of trust relationships, recovery, tracing and data integrity.

1.3.1 Taxonomy of Grid Security Issues

Securing a grid environment presents a distinctive set of challenges. Grid security requirements can be divided as generic and project-specific. Generic requirements include those which apply to existing systems, but are stretched by scalability and

administration complexity. Remote delegation and distributed authorization are requirements new to the highly distributed grid systems. Security requirements within the grid environment are driven by the need to support scalable, dynamic, distributed virtual organizations (VOs) which are collection of diverse and distributed individuals that seek to share and use diverse resources in a coordinated fashion [180]. Figure 8 shows the categorization of the different security issues in a grid as derived from [29]. Broadly, grid security issues can be classified as architecture-related issues, infrastructure-related issues and management-related issues [29].

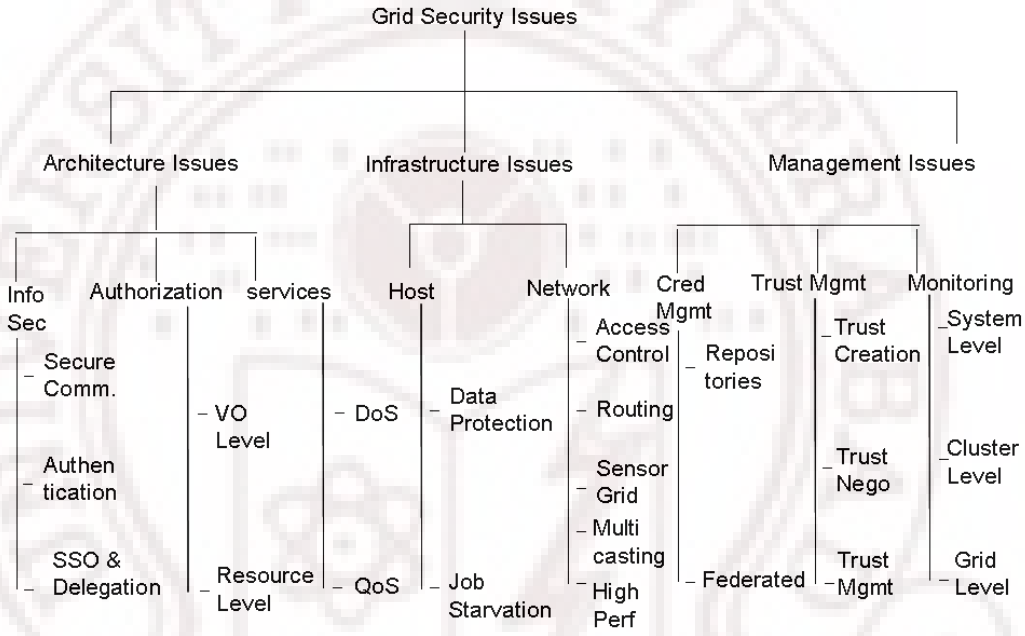


Figure 8: Taxonomy of Grid Security Issues:derived from [29]

Architecture issues address the concerns pertaining to the architecture of the grid. The requirements under this category are classified as information security, authorization and services. Authorization may be at the service level or resource level. Infrastructure related issues relate to the network and host components which constitute the grid infrastructure. Host level issues are those issues that make a host apprehensive about affiliating itself to the grid system. External jobs should not reduce the priority of local jobs and hence lead to job starvation. Similarly, if the host is a server, it can be concerned about its own availability. The third set of issues pertains to the management of grid. Credential management

is a critical issue due to the heterogeneous nature of the grid infrastructure and applications. Grid systems also require some amount of source monitoring for auditing purpose. Architecture level issues address the concern of the grid system as a whole.

Issues related to information security, authorization and service level security generally destabilize the whole grid system and so an architecture level solution is needed to prevent them. Information security is the security related to the information exchanged between different hosts or between hosts and users. The main information security issues are secure communication, authentication, single sign on and delegation [128]. Secure communication issues include those security concerns that arise during the communication between two entities like confidentiality and integrity issues.

Confidentiality is the property which ensures that the data is not disclosed to unauthorized users where as integrity ensures that the data is not modified by unauthorized entities. The Grid Security Infrastructure (GSI) addresses some of the above mentioned information security related issues. To ensure secure communication, GSI uses cryptographic techniques as the basis for creating secure grids and SSL/TLS for data encryption. Authentication is achieved in GSI with the help of a certificate [81]. Every user and service on the grid is identified via a certificate, which contains information vital to identifying and authenticating the user or service. The GSI provides single sign on and delegation capability which helps in reducing the number of times the user must enter his/her pass phrase when multiple resources are used.

Another core security issue in grids is the authorization of users. Like any resource sharing system, grid systems also require resource specific and system specific authorizations. Many authorization systems are possible in the context of grids. Mainly, we need to have Virtual Organization (VO) [12] level and resource level authorizations. VO level authorization works for the entire virtual organization. A VO normally consists of a set of users and several resource providers. The user presents the credentials to the resource to gain access to the resource, which means that resource holds the key to either granting / denying the access to the

users. Community Authorization Systems (CAS), Virtual Organization Monitoring Service (VOMS) and EALS (Enterprise Authorization and Licensing System) are some examples of VO implementations [33], [12].

The resource level authorization systems implement the decision to authorize the access to a set of resources. Some of the resource level authorization system include Akenti, PERMIS (Privilege Management Infrastructure Standards Validation) [28] and also Gridmap system. Thus, VO level and resource level authorization systems look at two different aspects of the grid authorization. Service level security is another important aspect of grid security. The possible attacks in this category include Denial of Service (DoS) and Quality of Service (QoS) violation. Denial of Service attack can be prevented in two ways: using preventive methods or through reactive techniques. QoS violation can be curbed by introducing some amount of monitoring and metering systems which detect the QoS levels of the systems and then make decisions to raise alarms.

Infrastructure-related issues are concerned with the grid infrastructure consisting of grid nodes and the communication network. Broadly they can be categorized as

- host security issues
  Solutions to host security issues include
  - data protection
  - job starvation
- network security issues
  The network security issues can be addressed by the following means.
  - Access control and isolation
  - secure routing
  - secure multicasting

The different management issues that concern grid administration are credential management, trust management and monitoring related issues. The major credential management systems are credential repositories and credential federation systems. Another important management issue is trust management. Trust is

a multi-dimensional factor which depends on a host of different components like reputation of an entity, policies and opinions about the entity. The trust life cycle is composed of three phases. Trust creation phase, trust negotiation phase and trust management phase. Trust management solutions can be broadly categorized into reputation based and policy based. Another management issue is that of monitoring of resources. This is essential in a grid scenario as different organizations or departments can be charged based on their usage. Secondly, resource related information can be logged for auditing or compliance purposes. The monitoring stages include data collection, data processing, data transmission, data storage and data presentation. Monitoring can be done at various levels such as system level, cluster level or at the grid level.

The activities that are needed to secure a grid environment can be grouped into four categories: naming and authentication; secure communication; trust, policy and authorization; and enforcement of access control [141]. One of the critical differences between grid security and host or site security is *site autonomy*. In contrast to the site security where a system administrator has complete control to modify the security mechanisms and security policies, resource providers for the virtual organization of the grid must understand and accommodate mechanisms and policies that are strictly under their control. While the virtual organization certainly could define some common security mechanisms and policies across it, resource utilization by the nonlocal members of the VO must be made possible.

1.3.2 Generic Grid Security Model

As grid computing provides a virtualized view of the underlying grid resources, we require a loosely-coupled platform independent model of securing applications within and across organizations. Anirban Chakrabarti [29] suggested such a generic model for grid computing security. We depict various components of this model in Figure 9.

1.4 Research Objectives and Problem Statement

In the previous sections we have detailed the grid security requirements, the core aspects of computer security and the existing access control paradigms. In this

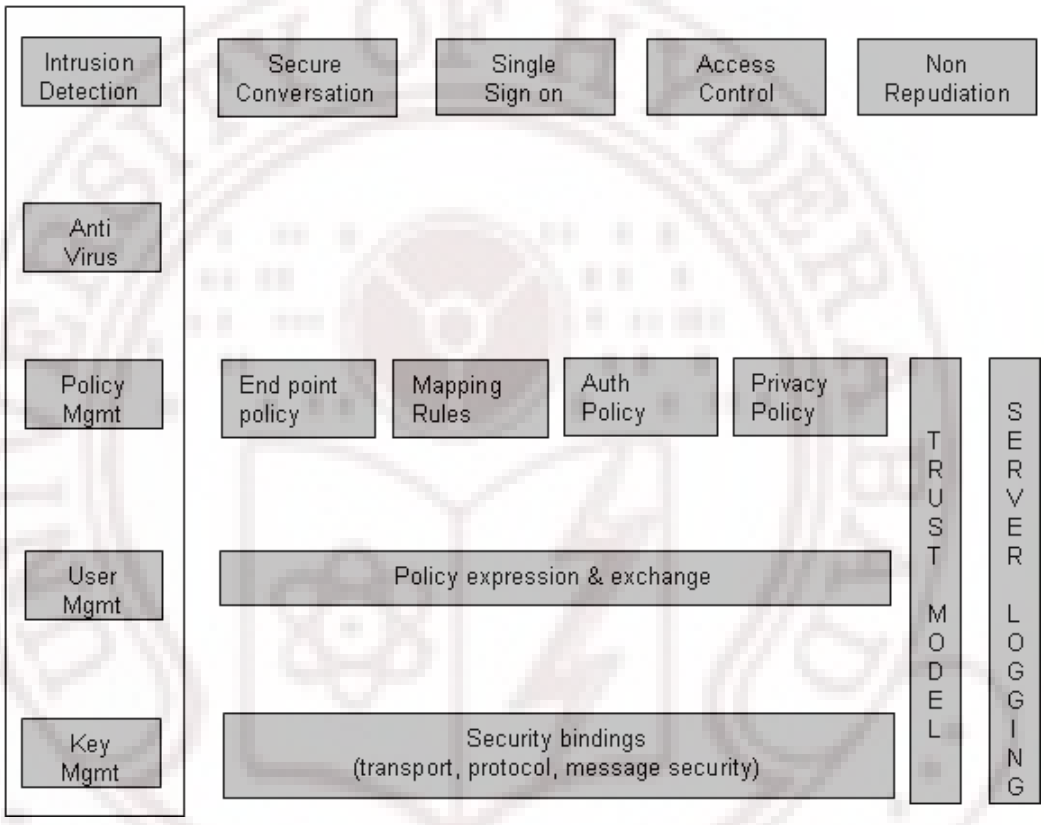


Figure 9: Components of the Grid Security Model: derived from [29]

section we give the research objectives and the problem statement for this thesis. We aim to come up with an access control model which can address all issues of grid computing security. Traditional means of security administration that involves manual editing of policy databases or issuance of credentials cannot meet the dynamic grid access control scenarios. The grid environment poses additional challenges that have not been addressed by classical access control mechanisms. Grids assemble heterogeneous resources from different providers. The physical storage location of data is transparent to the users and data may be automatically replicated by the grid infrastructure for a more effective access. Therefore a user who stores sensitive data on a grid wants the access control to be effective for all its replicas, no matter who owns the actual storage devices. To allow users to selectively share data using the grid, the access control system must allow them to delegate access rights to other entities. The possibility to delegate access rights also makes the system more scalable, since it allows to distribute the burden of access control administration. We require a user-driven security model that allows users to create entities and policy domains in order to facilitate creation and coordination of resources within the virtual organizations. Access control models provide a formalism and framework for specifying, analyzing and evaluating security policies. Grid entities need to trust each other to initiate authorization.

Proposing security mechanisms for an environment requires thorough understanding of the environment, its security issues, requirements and challenges. Like any resource sharing system, grid systems also require resource specific and system specific access control and authorization. The issues and challenges of grid access control are as follows.

- Permit fine-grained access control on resources, where the sources of authority that may issue permissions are individual users owning the data.
- Dynamic authorization and access control
- Indirect authorization through delegation
- Group authorization through role-assignment to users
- Providing flexible policy-driven access control and federating policies from several independent sources

- Allowing long-running jobs to be treated as objects whose management is subject to access control decisions
- Providing a means of publishing, negotiating and exchanging policy statements where the policies are written in an expressive policy language
- Appropriate trust support services to address the dynamic and complex relationship among organizations across trust domains comprising of dynamic resources

The major objectives of the thesis are to:

- investigate various grid access control issues and requirements
- examine the existing models of access control in grids
- propose a solution to the access control challenges through authorization modelling.
- suggest a policy representation and implementation mechanism for the models using XML based access control language which can support legacy access mechanisms

The thesis addresses the general issues of access control in grid environment with specific emphasis on delegation, fine-grained access control, dynamic authorization, cross-domain access control and formulation of platform independent access policies. The following are the key concerns addressed in this thesis.

- How to provide a grid-wide access control and authorization mechanism?
- How to achieve indirect authorization of grid users?
- How to devise a mechanism for dynamic authorization of user's access rights?
- How to incorporate fine-grained access control on grid resources?

The thesis work considers access control through authorization as the major mechanism for the policy formulation, modelling and implementation for grid computing security. Accordingly, access control models have been designed to represent solutions to the various problems being addressed in this thesis. In this

work, we first addressed the problem of providing grid-wide authorization by suggesting architectural frameworks for single-domain and cross-domain scenarios. We proposed authorization algorithms to govern the working of these architectures. We then studied the indirect authorization issues and came up with models for delegation and revocation. The grid-wide authorization, delegation and revocation mechanisms are built on the NIST Role Based Access Control (RBAC) standard. These frameworks were implemented for a grid platform and found to suit the grid access control and security requirements. We integrated RBAC with the grid environment. Next, we addressed the problem of providing access control in dynamic context using trust factor. Finally we suggested a fine-grained access control mechanism for grid resources based on trust quantification. The models proposed through this thesis have been implemented on a grid middleware platform.

1.5 Contributions of the Thesis

We summarize the main contributions of the thesis below.

- **Direct Authorization through RBAC.** We have integrated the standard Role Based Access Control with the grid environment to support policy formulation and enforcement of security. This solution provides a grid-wide access framework called the grid authorization model.

- **Indirect Authorization through Delegation and Revocation of roles.**

  Here, we have proposed a novel approach to grid delegation whereby the *role* of a grid entity can be used as the standard unit of delegation in place of the credential-based GSI proxy delegation. This approach is built on the Role Based Access Control model and Role Based Delegation. We have proposed a framework called Role Based Grid Delegation Model (RB-GDM).

- **Dynamic Authorization of User's Access Rights.** We proposed a new scheme called Fuzzy Trust and Delegation Model (FTDM). It uses a two-stage fuzzy inference process based on trust relationships among the grid entities. The privileges are granted depending on the trust values.

- **Fine-Grained Access Control for Grid Resources.** To restrict access to resources by unknown grid entities (guest users), we require granular access control. We suggested a flexible, trust-aware scheme. The access decision (grant/deny or partial grant), is governed by the trust relationships among the domains.

- **Enforcement of Grid Access Control.** We implemented the frameworks for direct authorization and delegation on a grid simulation platform.

We conducted a case study of Garuda Grid, the National Grid Computing Initiative of India. The existing security measures in Garuda Grid include the following aspects: developing Garuda security policies, evolving responsibilities for resource providers as well as users, user registration using PURSE (Portal based User Registration Service which uses web based applications on GSI), credential delegation, single sign-on and user mapping.

1.6 Organization of the Thesis

The thesis aims at modelling the access control and authorization requirements of grid computing systems. In order to model these requirements, four major aspects of grid resource access namely direct authorization, indirect authorization (delegation), dynamic access and finegrained control have been considered. The thesis makes use of the NIST RBAC standard, Fuzzy Inference Systems and other well known standards for modelling grid security. The thesis is organized into eight chapters.

Chapter I: Introduction. The chapter provides an overview of grid computing, various access control models and grid security requirements. It presents the taxonomy of grid security issues and existing generic grid security model.

Chapter II: Survey of Grid Security Models and Mechanisms. This chapter presents the related research work in the area of grid computing security with specific emphasis to grid access control and authorization.

Chapter III: Role-Based Grid Authorization Model. This chapter presents architectural frameworks for facilitating scalable access control and authorization in grids. These frameworks are built on the standard RBAC. Considering the grid as

one logical enterprise, we introduced a single-domain authorization mechanism . Then we extended this framework for cross-domain authorization. The first requirement for cross-domain authorization is to map the role of a given domain to its equivalent role in another domain. To address the issue of resolving a local role to the corresponding global role, we suggested a role-mapping architecture. This architecture maps a role in a local domain to its equivalent global role. We used the approach of weighted role-ranking for the same. The authorization decision is taken based on the mapped global role ranking and the resource access policies.

In summary, the chapter proposes a secure inter-operable mechanism for access to grid resources within a single physical domain and also across multiple domains in a grid VO.

Chapter IV: Role-Based Grid Delegation Model. Role-based delegation is suggested as a useful authorization mechanism for grids. As credential-based delegation has its own limitations in a dynamic grid environment, a new conceptual model is required to effectively formulate the grid delegation requirements. In this chapter, we present a framework called Role-Based Grid Delegation Model (RB-GDM) for delegating access rights in a single grid enterprise. The basic unit of delegation in this model is *role*. Derived from the standard RBAC formalisms, this framework explores various approaches for indirect authorization through delegation. We present the RB-GDM interconnection framework for various scenarios like intra-domain and inter-domain delegation (both peer-to-peer as well as master/slave topologies). We also present mechanisms by which one can revoke the delegated rights. The revocation methods include grant-dependent, grant-independent as well as timeout revocation. We also extend the model for cross-domain delegation. We have also implemented these frameworks on a grid middleware platform.

Chapter V: Trust-Based Dynamic Delegation Model. The chapter presents a Fuzzy Trust and Delegation Model (FTDM) to address the dynamic access control requirements of grids. As the grid resources are distributed in space and time, we need to have authorization mechanisms supporting dynamic context. Also, trust relationships are central to the implementation of security in the dynamic and ever evolving communities like grids. The model presents a two-stage fuzzy inference

process wherein the first stage computes and quantifies the trust values of the sites involved. The criterion is the effectiveness of the site security mechanisms. In the second stage we determine the degree of delegation based on the trust values obtained from the first stage. The technique suits the dynamic grid environment which is fuzzy.

Chapter VI: Fine-Grained Access Control Framework for Grid Resources. In this chapter, we propose a flexible, trust-aware fine-grained access framework. Presently grid security mechanisms maintain confidentiality and integrity of resource access through coarse-grained grant/deny permissions to known entities. In the proposed framework, trust relationships govern the access policies. We consider various trust categories namely direct trust, transitive trust and asymmetric trust. The trust values are quantified through fuzzy inferencing and then mapped to access control decisions. We consider varying degrees of access like full access, partial access (fine grained) or access denial (no access).

Chapter VII: Implementation Aspects and Case Study. Here, we present the implementation aspects of the models proposed in the thesis. The enforcement mechanism is built on the grid middleware platform. We opted for Globus, as it provides support for implementation of security interfaces. We also conduct a case study on the security requirements of Garuda and suggest ways to incorporate some of the models to improve Garuda security. The existing security mechanism of Garuda relies on authentication. Though Garuda supports authorization through grid-map file, it is not fine-grained. PURSE, the user registration portal used in Garuda is just sufficient as an authentication scheme. We suggest the following measures to provide better security for resource access in Garuda Grid.

- Incorporating a role-based authorization mechanism
- Introducing role-based delegation to facilitate indirect authorization
- Incorporating revocation mechanisms to revoke the delegated rights on the resources
- Establishment of trust relationships among domains to support dynamic access control

Chapter VIII: Conclusion and Future Scope. In this chapter, we summarize the major contributions of the research work presented in the thesis and give our observations. We also highlight the future scope and further research directions.

Overall, this thesis covers both the theoretical aspects as well as implementation aspects of grid computing security through access control modelling and its implementation. Theoretical characterization of the proposed models has been done through algorithmic representations. The applied aspect of the research reported in the thesis is reflected in the implementation conducted on the grid middleware platform.

1.7 Chapter Summary

This chapter provides an introduction to grid computing, an emerging technology of enormous promise. We discuss the elements which constitute the grid and also its economic aspects. Next, we discuss the basics of computer security and present the existing security models. From security models, we then show the evolution of access control models and approaches adopted for different environments and systems. We discuss the grid computing security and present the taxonomy of grid security issues. The problem definition and the contribution of the thesis are presented along with the organization of the thesis.

CHAPTER II

LITERATURE SURVEY ON GRID SECURITY MODELS AND MECHANISMS

2.1 Introduction

The previous chapter of this thesis describes various aspects of grid computing environment and the security issues associated with grids. In this chapter, we present a detailed survey of relevant literature in grid computing security. The aim of this survey work is to present current research trends from the point of view of the security models and authorization mechanisms available in the grid computing environment. Efforts have been made to present an up-to-date research information with details of approaches, algorithms and implementation systems that have been developed and used during the last decade of grid computing evolution.

The chapter has been organized as follows. First we review the literature work in the general area of grid computing security. In section 3, we present various schemes in the grid access control. Section 4 discusses the grid authorization attempts. Section 5 presents the research in trust relationships in grid computing. Next, we present the security issues identified as part of this survey. Section 7 discusses the need for grid access control and the characteristics of an ideal grid security mechanism. Finally we summarize the chapter.

2.2 Grid Security

Grid computing is an interesting and a highly potential area for most enterprises [74] of recent times. Security is one of the major impediments in wide-spread grid adoption. By security, we mean the concepts, techniques, technical measures and administrative measures used to protect the information assets from deliberate or

inadvertent unauthorized acquisition, damage, disclosure, manipulation, modification, loss or use. The standardization effort of grid security has led to the design of security standards in grids which is defined under Grid Security Infrastructure(GSI) [135], [190]. Global Grid Forum(GGF), a consortium of researchers and practitioners was set up for exchanging information and defining standards for grid computing. The Open Grid Standards Architecture (OGSA) proposed by Ian Foster and his group, defines web services [33] for different systems to communicate and share the heterogeneous grid resources.

A grid defines the concept of a Virtual Organization (VO). In a VO, different individuals, enterprises, organizations come together to share resources and services under a set of rules or policies guiding and governing the extent and conditions of sharing. VO can be formed across different universities, across different enterprises as well as within an enterprise also. The level of heterogeneity is high in such situations. Thus the main aspect that separates grid systems and its security from other systems are heterogeneity and complex policies [61]. Marty Humphrey, Mary Thompson et.al, in their work on grid security [139] came up with a comprehensive study on the security aspects of grids. According to them, the most significant challenge for grid computing is to develop a comprehensive set of mechanisms and policies for securing grids. Their work presented the state of the art with regard to grid security and identified open issues. This paper grouped the activities in grids that need to be secured into four categories: naming and authentication; secure communication; trust, policy, and authorization; and enforcement of access control. According to them significant challenges remain in cases like privacy management, denial of service, and integrated cross-domain auditing.

"Security for Grid Services" [128], another work in this area by Von Welch et.al, discusses how one must deal with diverse local mechanisms, support dynamic creation of services, and enable dynamic creation of trust domains. They have also described how these issues are addressed in various versions of the Globus Toolkit, a grid middleware. The combination of dynamic policy overlays and dynamically created entities drives the need for three key functions in a grid security model namely, multiple security mechanisms, dynamic creation of services and dynamic

establishment of trust domains. According to this work, security can be implemented as a service. In summary, the work suggests that secure operation in a grid environment requires that applications and services be capable of supporting a variety of security functionality, such as authentication, authorization, credential conversion, revocation [52], auditing and delegation.

Ian Foster et.al, [128] analyzed the unique security requirements of large-scale distributed (grid) computing and developed a security policy and the corresponding security architecture. An implementation of the architecture within the Globus meta-computing toolkit was also discussed. To identify the security requirements in any environment, the primary step is to understand the environment's distinctive characteristics.

Accordingly the main characteristics of grid are identified as: large and dynamic user population, large and dynamic resource pool, dynamic computation involving group of processes running on different resources and sites, unicast or multicast communication among the processes, resources requiring different authentication and authorization mechanisms. Also individual user may be associated with different local name spaces, credentials, or accounts, at different sites and finally the resources and the users may be located in different countries. Based on these characteristics the following grid security requirements were identified: authentication, access control, integrity, privacy and non-repudiation. Specifically we need to provide authentication and authorization solutions for users, processes and resources of the grid. Subsequent works outline the generic requirements for security in grid systems and the problems often cited with current grid software. The outcome was the possibility of these issues being resolved by a federation of both users and resources. Also a comprehensive set of grid usage scenarios were presented and analyzed by M Humphrey and M R Thompson in the year 2001 [139], with regard to security requirements such as authentication, authorization, integrity, and confidentiality. There were also works which made suggestions for implementing security without much performance degradation in grids.

2.3 Grid Access Control

There have been some attempts at providing security to the resources of a grid environment through access control. Here we discuss some of such attempts. Kevin Kane and James C. Browne, in their research paper [92], presented a classification of implementations of access control systems based on a lattice taxonomy. Pietro Mazzoleni et.al [186] discussed ways to provide fine-grained access control in large scale grid services. They developed a novel resource broker service for grids that integrates access control with resource scheduling. Weizhong Qiang et.al [202], proposed a general authorization and access control architecture, RBGACA, for grid environment. It is based on the classical access control mechanism - Role Based Access Control (RBAC). The architecture provides convenient policy evaluation and decision making approach. They provided a framework for access control management that treats the whole grid as a series of independent, interrelated, dynamic domains. Though this approach introduced RBAC in grids, it did not address many issues like delegation, scalability and trust relationships. Also, no provision for cross-domain authorization was made.

Xinwen Zhang et.al [201], in their work, "A Usage-based Authorization Framework for Collaborative Computing Systems", proposed a usage control (UCON) based authorization framework for collaborative applications. Usage control policies are defined using subject and object attributes [11], along with system attributes as conditions. Earnesto Damiani et.al [69], in their work titled "New Paradigms for Access Control in Open Systems" suggested ways of providing access control in open distributed systems. They surveyed the current state of this research area and came up with the possible future trends in open system access control. There are a few new access control paradigms like the attribute-based models and the semantics-based access control [21] which outline some research and development challenges that should be addressed in this field. "XACML Policy Integration Algorithms", a work done by P. Mazzoleni et.al [100], suggested the use of XACML, an OASIS [1] standard language for the specification of authorization and entitlement policies. Works like [76] by Gabriel Kuper and [103] by Nathan N Vuong discuss the use of markup languages to present generalized

security views. In [107], Rafae Bhatti et.al proposed an XML-based policy specification framework and architecture for enterprise-wide access control. Virtual enterprises are usually built through collaboration of several autonomous subjects sharing their resources. Their requirements have been addressed by industrial outfits like the IBM [36], [38]. James B.D.Joshi et.al proposed an access control language for multi-domain environments in [47]. Weizhong et.al [118] proposed a VO-based access control model for grids.

2.4 Grid Authorization Systems

In the literature on grid security research, the focus has been on authentication. Here, we discuss some of the research works and also the publications in the area of grid authorization. Access control in any computing environment involves three phases '**A**uthentication', '**A**uthorization' and '**A**ccountability and auditing'. These three phases are called the three **A**s of access control. Though authentication forms the foundation for most of the grid security solutions, authorization also has evolved as an important stage of access control [148]. Lorch et.al PRIMA [95], proposes a Policy-Reduced Integrity Measurement Architecture (PRIMA) for grids. They proved how a remote party can verify the integrity properties using PRIMA. R. Alferi and R.Cecchini [12], discussed the evolution of grid authorization from grid map file to Virtual Organization Membership Service (VOMS). A simple authorization implementation, based on a direct user registration on the resources, is not sufficient for large scale environments like grids. According to the authors, VOMS allows fine grained control of the use of the resources both to the users' organizations and to the resource owners. But the drawback of this approach is that through a membership in VOMS the user gets access to the resources without any fine-grained control. Mary Thompson et.al [99] presented a mechanism called Akenti, for authorization in grids. But, it does not directly address the determination of rights to be granted.

Attribute-based authorization schemes have also been suggested for grid access which can streamline and broaden access control by allowing authorization decisions to be made based not only on user identity, but also user attributes and roles. Community Authorization Service or CAS proposed by Laura Pearlman et.al [196]

is an authorization system built on the Grid Security Infrastructure (GSI). CAS allows resource providers to specify course-grained access control policies in terms of communities as a whole, delegating fine-grained access control policy management to the community itself. The drawbacks of CAS are, it completely removes access control from local resource or service, which is in violation of one of the main properties of grid namely "*site autonomy*". CAS also has scalability issues because of central management. In addition, the approach of limiting the group's privileges [60], through restrictions, to generate a subset of privileges applicable to an individual community member presents a subtractive security mechanism and violates the "*least privilege principle*" [17]. Authorization and account management in Open Science Grid has been discussed in [98]. Some authentication and authorization mechanisms for multi-domain environments have been discussed in [94].

Another issue is scalable authorization. The work by David Cordes et.al [35] addressed the authorization problem in grid system environments. They came up with a solution for authorization within the *Globus* system. The authorization approach is based on distributed authorization servers and extensions to *Globus Metacomputing Directory Service* (MDS). "A Rule-Based Framework for Role-Based Delegation and Revocation" [200], though a contribution not specific to grids, focuses on user-to-user delegation, where a user delegates his/her role to another user. The authors proposed a rule-based frame work for role-based delegation. A rule-based system is one where all behaviors are governed by a set of explicit rules. The major requirements of role-based delegation were identified as support for multi-step delegation, support for different revocation schemes, support for constraints and support for partial delegation. But this work was a general framework meant for enterprises. Issues specific to grids were not addressed in this paper.

James B.D.Joshi et.al [183] suggested an integer programming (IP) approach for secure inter-operation involving RBAC policies. But their work also does not reflect the distinct characteristics and requirements of grid authorization. Another proposed approach is user-credential based role-mapping where by a user's credentials associated with the role form the basis for role-mapping [146]. As the

fundamental unit of RBAC is a *role* and since one user can be mapped to multiple roles, we cannot use the user's credentials as the sole criteria for role mapping. Liang Chen et.al [30], proposed an inter-domain role mapping technique based on the principle of least privilege. They suggested a minimal cardinality for a role within a domain to avoid misuse of access. But this constraint may work against the dynamic and heterogeneous access requirements of grids.

Some of the existing grid authorization mechanisms are Permis [67], [28], Akenti [99], Shibboleth [181], VOMS [12] and CAS [196]. Though Permis, Akenti and CAS introduce the concept of roles in a grid environment, they are not role-based access control implementations like the standard RBAC. Also they lack the flexibility of RBAC and are static in nature. CAS is primarily a community based authorization service, it allows resource providers to specify course-grained access control policies in terms of communities as a whole, delegating fine-grained access control policy management to the community itself. The major drawback is the lack of scalability and denial of basic right for every node to decide its users.

"A Multi-policy Authorization Framework for Grid Security", a work by Bo Long et.al [55], suggested a framework for authorization in a grid system based on multiple policies across the domains. The grid environment needs to be flexible and scalable to support multiple security policies. The absence of a standard role-mapping mechanism to address the grid authorization and access control issues combined with the fact that the present form of RBAC for single enterprises does not support grid access control motivated us to develop a new architecture which can truly reflect a multi-domain grid access environment.

2.5 Trust in Grid Systems

Significant amount of research has been done in the area of trust relationships in distributed systems in general and grid environment in particular. Woodas et.al [169], proposed trust models for distributed systems and suggested that distributed trust models assume asymmetrical trust. They also talked about combining trust values of different applications. Snelling et.al, [188] suggested the use of explicit trust relationships for providing security in dynamic grids. Farag Azzedin

et.al [13], proposed a trust-aware resource management mechanism for grids. Resource optimization in grids has been another area where the concept of trust has been suggested [140]. Liu et.al [156], et.al proposed a mission-aware trust model for grid computing systems. Internet applications also extensively use the concept of trust [132], [133], [134]. The usefulness of trust in ubiquitous computing has been highlighted in [184]. Jeffrey Dwoskin et.al [85], in their research work, defined three generic grid security scenarios: mutual trust, partial trust (distrusted user) and mutual distrust. According to them, decreasing levels of trust enable one to expose new vulnerabilities and show the increasing levels of security support required.

2.6 Shortcomings of the Existing Solutions

The existing approaches to grid security fail to provide a holistic solution to grid security. Primarily, authentication has been in use for providing security in grids. Authorization has proven to be a successful alternative or addition to authentication in providing security to today's enterprise applications. Also, security in dynamic context is another aspect which we need to consider. Building trust relationships between the organizations in a grid can contribute to grid security. Fine-grained access control of the resources is another important requirement in grids. Providing security to the geographically spread resources is a real challenge, which, many of the existing solutions seem to neglect. We need to come up with solutions for grid computing security keeping in mind its characteristics. i.e, We need to provide a specific security mechanism for grids in place of the existing generic approaches. As a first step in this regard, we identified various security issues in grids, which led us to the need for access control in grids. From the findings of the literature survey we have arrived at the characteristics of an ideal grid access control mechanism which we explain in the following section. These characteristics govern the design of the authorization models which we have proposed through this thesis.

2.7 *Security Issues Identified in Grids*

Based on the extensive literature survey conducted with regard to grid computing environment in general and grid security and authorization in particular, we have come up with some observations, which guide us through the design and development of a grid access control mechanism. We first identified the security issues in grids, the need for access control in grid systems and finally came up with the desirable features of a grid access control and authorization mechanism. As grids are decentralized, dynamic and distributed in nature, they require security mechanisms that allow seamless access to resources to the users of different organizations [58]. The most significant challenge for grid computing is to develop a comprehensive set of mechanisms and policies for securing the Grid. The security constraints resulting from the distinctive characteristics of grid computing environment include:

- Large and dynamic user population which belong to many physical organizations and which may change frequently
- Large and dynamic resource pool contributed by different physical organizations which can change rapidly.
- The possibility that a computation may acquire, start processes on, and release resources dynamically during its execution. In other words, throughout its lifetime, a computation is composed of a dynamic group of processes running on different resources and sites
- Resources may require diverse authentication and authorization mechanisms and policies
- Resources and users may be located in different locations

Other security issues derived from the characteristics of the grid environment include [141]:

1. **Single sign-on:** A user should be able to authenticate once and initiate computations that require resources, use resources, release resources without further authentication of the user

2. **Delegation:** A user must be able to endow to a program/user the ability to run on that user's behalf, so that the program/user is able to access the resources on which the user is authorized. The program/user should also be able to further delegate to another program/user

3. **Inter-operability with local security solutions:** While the security solutions may provide inter-domain access mechanisms, access to local resources will typically be determined by a local security policy that is enforced by a local security mechanism

4. **Naming:** Users in the grid must be identified unambiguously and globally which means that a user needs a distinguished name which is unique through out the grid. The X.500 naming structure is an attempt to define enough components that people can be named uniquely and meaningfully. X.509 names, derived from the X.500 standard, are commonly used in grids to provide global names for users and hosts

2.8 Need for Access Control in Grids

Access Control is the secure evaluation of whether an established identity can have access to a particular resource, also referred to as an object. The following grid characteristics point to the need for enforcing access control in grids.

1. The resources in a grid environment are owned by multiple institutions. Each individual institution may give preference to local users (belongs to the same institution) than users of other domains.

2. The service providers restrict the access to their resource

3. At the organization level users of higher authority may get more preference than users with lower authority

4. Resources in the grid environment are valuable, confidential and need different levels of authorization

5. Resources have their own restrictions to take into account before providing services to the requests [37] (for example, a server may consider availability of its free memory, CPU load etc before accepting the request)

2.8.1 Characteristics of An Ideal Grid Access Control Mechanism

As discussed in section 2.2, a grid is a virtual organization which consists of a set of institutions. Therefore, an efficient grid access control system should meet the following requirements:

- **Suitability for the Organization's Needs:** Each organization, in real world consists of hierarchy of users and different set of permissions associated for the respective users.
- **Scalability:** Grids consist of large and dynamic users. So, the access control mechanism must be able to deal with large number of users.
- **Maintainability:** User populations and resources are often large and unstable in a grid system environment. Access control information between users and resources may be adjusted and changed at any time. The grid authorization service must ensure that the associated administration and maintenance work is able to keep pace with the dynamic nature of the grid system.
- **Variety of security policies:** The authorization scheme must be able to express a variety of security policies. Since each organization unit and resource may have different ways of specifying its own security policy, this flexibility is essential.

2.9 Chapter Summary

This chapter focuses on the related research which has taken place over the past few years in the area of grid computing security. The last decade has seen tremendous development with regard to grid technology. We carried out an extensive literature survey, on the research work done in the grid computing area in general and grid security and access control in particular. The chapter starts with a brief introduction to grid technology. In the later sections, we discuss the research work which has taken place in grid computing security, followed by research on grid access control and grid authorization. We explain in detail, various research papers published in this area, the major contributions and their drawbacks. Various

authorization attempts in grid security implementation have been discussed thoroughly. We also discussed the importance of trust relationships in grids and implementing security based on trust values across domains. We presented a summary of major security issues in grids, the need for access control in a grid environment and also the desirable features of a grid access control mechanism.

The literature survey on related work motivated us to take up the following research directions. In Chapter III, we come up with a role-based grid authorization model which is the resultant of our study on Role Based Access Control. In Chapter IV, we have suggested a delegation model for grids. It is the outcome of the study and survey we conducted on the delegation requirements in grids. Our research on the dynamic delegation requirements in grids motivated us to propose a trust-based dynamic delegation model for grids. This is explained in Chapter V of the thesis. To facilitate a fine-grained access mechanism for grid resource access we have created a framework as shown in Chapter VI. During the course of the literature survey we felt the lack of sufficient work as well as survey material in the area of implementation of grid security. To fill the gaps and also to test our models, we prepared an implementation set up, the details of which are explained in Chapter VII of this thesis.

CHAPTER III

ROLE BASED GRID AUTHORIZATION MODEL

3.1 Introduction

The previous chapter dealt with a literature survey on grid computing research with specific emphasis on grid security models and mechanisms. In this chapter we address one of the core security issues of grids namely protection of resources from unauthorized access. We provide solutions for direct as well as indirect authorization issues. Our solutions include the proposed architectural frameworks for single-domain (intra-domain) role-based authorization and cross-domain (inter-domain) authorization. These designs are based on the standard Role Based Access Control Model (RBAC) [124] and have been supported by implementation done on the Globus middleware platform. RBAC is yet to be fully utilized in grids.

Apart from authorization, secure inter-operability has been a growing concern for multi-domain computing systems like the grids [183]. This is due to the fact that a grid comprises diverse local administrative domains collaborating as a virtual domain. Through the proposed cross-domain (inter-domain) authorization framework, we solve the issue of secure inter-operability. As a first step in this direction, we propose a role-mapping architecture. A *role* in one administrative domain may have a different meaning in another domain as roles reflect the responsibilities of the users in the organization [171], [71]. We establish role-equivalence among the domains by mapping a local domain role to its equivalent global role.

This chapter is organized as follows. The first section presents the grid authorization and access control framework. We propose the system architecture for single domain (intra-domain) authorization in the next section. Section 3 depicts the cross-domain (inter-domain) role-mapping and authorization architecture. In the last section, we summarize the contributions made in this chapter.

3.2 Grid Authorization and Access Control Framework

We address the problem of authorization in a grid environment using RBAC. Mandatory authorization schemes were initially used to define access control policies [194], [80]. The evolution of RBAC as a reliable standard for single enterprises [19], [91], [78] motivated researchers to think of ways in which it could be incorporated into grid environment. A grid is often viewed as a logical organization formed of multiple physical organizations or enterprises and hence the integration of RBAC into grid is a logical extension of the standard RBAC implementation. For a grid system usually formed by multiple domains which are maintained by different companies, organizations or institutions, inter-operability is a major issue.

A *role*, though the basic unit of access control as per the RBAC, signifies different meanings in different organizational contexts. A mechanism has to be in place by which we can map the role of one enterprise into its new semantics in another enterprise. Delegation and revocation mechanisms also are based upon role. In our work, we present an architecture for implementation of access control through authorization with supportive delegation and revocation mechanisms for a single domain environment. Later, we extend this work for authorization of users across various domains.

3.2.1 Requirements

- We need an access control model which is suitable for an organization's requirements and is also scalable for grid. Role Based Access Control is suitable because of its ability to represent real world organizational role hierarchy [41] and scalability. In order to implement an access control model we need:

    1. **Access control policies:**
       An access control policy is a security policy consisting of a set of rules that an organization adopts to govern *who can have access to what resource*. In RBAC, access control is based on the *role* of the user and not on the user's identity
       XACML is used as the policy language to express the access control information. The language provides built-in features to express complex access control policies

2. **Framework to operate with policies:**
   We need a mechanism which can make the access decision based on user's role, the resource being accessed, the action being performed by the user on the resource
   An XACML framework is available to operate with the access control policies written in XACML.

- We require a data store, which holds user-role, role-hierarchy, role-delegation and such information.
- A Graphical User Interface (GUI) for the administrator to perform operations specified in the functional specifications of RBAC. (assign/delete roles to users, specify the role-hierarchy).
- We need a GUI for the Policy Administrator to create, delete and modify the access control policies.
- GUI for the clients to know the information about user-role relationships in the organization, support for delegation of his/her role to other users and the log information about delegation (to whom he/she delegated and what are his/her delegated roles).
- We need a server which is populated with the details of users and resources with in the organization so that clients can know the organizational structure

3.3 System Architecture for Single Domain Grid Authorization

We suggest an architecture for authorization in a grid environment envisaged as single enterprise. We also design a back end policy database. The main components of our framework include the following.

- **XACML Policy Framework:** It is designed as a separate service running on remote system. It accepts the requests from multiple authorization modules of the middleware and makes access decisions. Policies are specified using XACML. These policies have been placed on the same system where the XACML framework resides

- **Globus container:** This component contains all the services and accepts requests from the grid clients and runs on a separate system
- **Database:** It contains information about user-role, role-delegation etc. The Policy Enforcement Point (PEP) of the policy framework queries the database for user, role and permission details
- **LDAP(Lightweight Directory Access Protocol)server:** It runs on a separate system and accepts request from the LDAP clients. It contains the complete details of users and resources in the organization. It also responds to the queries from the Policy Enforcement Point (PEP) for additional user attributes
- **Admin GUI:** An interface to enable administration of the access control decisions, it resides in the same system where policies are stored
- **Client GUI:** It is an interface which enables the grid client to send access requests. It resides in the grid client system

The architectural components and their interfacing is presented in a basic framework as shown in Figure 10, which is a simplified version of the detailed system with the components for a single domain authorization and their associations shown in Figure 11.

An administrator in the proposed architecture has the following functionalities.

- **Create Organizational Hierarchy:** The administrator specifies the roles at each level
- **Assign Roles to Users:** The administrator assigns roles to users. The constraint in role assignment is that one user has at most one role only. We do not deal with the Separation of Duties (SoD) and cardinality issues of RBAC as these are not in the purview of our work
- **Modifying User-Role Information:** The administrator can delete or change user's roles. The administrator has to maintain all the details about users and resources of the organization
- **Specify Access Control Policies:** The administrator is responsible for creating new policies, modify and delete the access control policies according to

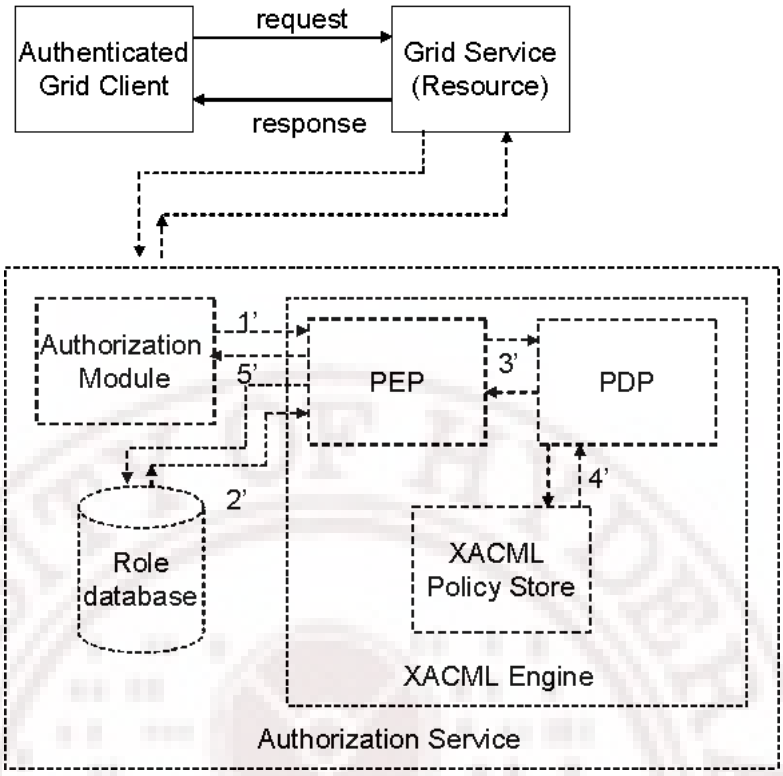


Figure 10: Basic Architecture for Single Domain Authorization

the organizational needs. The access control policies are based on the role information of the user

- **Check the Audit files:** The authorization module in the grid node logs information about user activities. The administrator checks these log files and takes appropriate decisions

The administrator needs to interface with the policy database to assign/manipulate the user-role or role-permission information. It also needs to have another interface for extracting additional user attributes from the Lightweight Directory Access Protocol (LDAP) server. Thus, the two interfaces for the Administrator include:

- **Interface to Database:** With this interface, the Administrator can do all the user-role, role-permission assignments.
- **Interface to LDAP server:** Support for adding, deleting and modifying entries in the directory at the LDAP server.

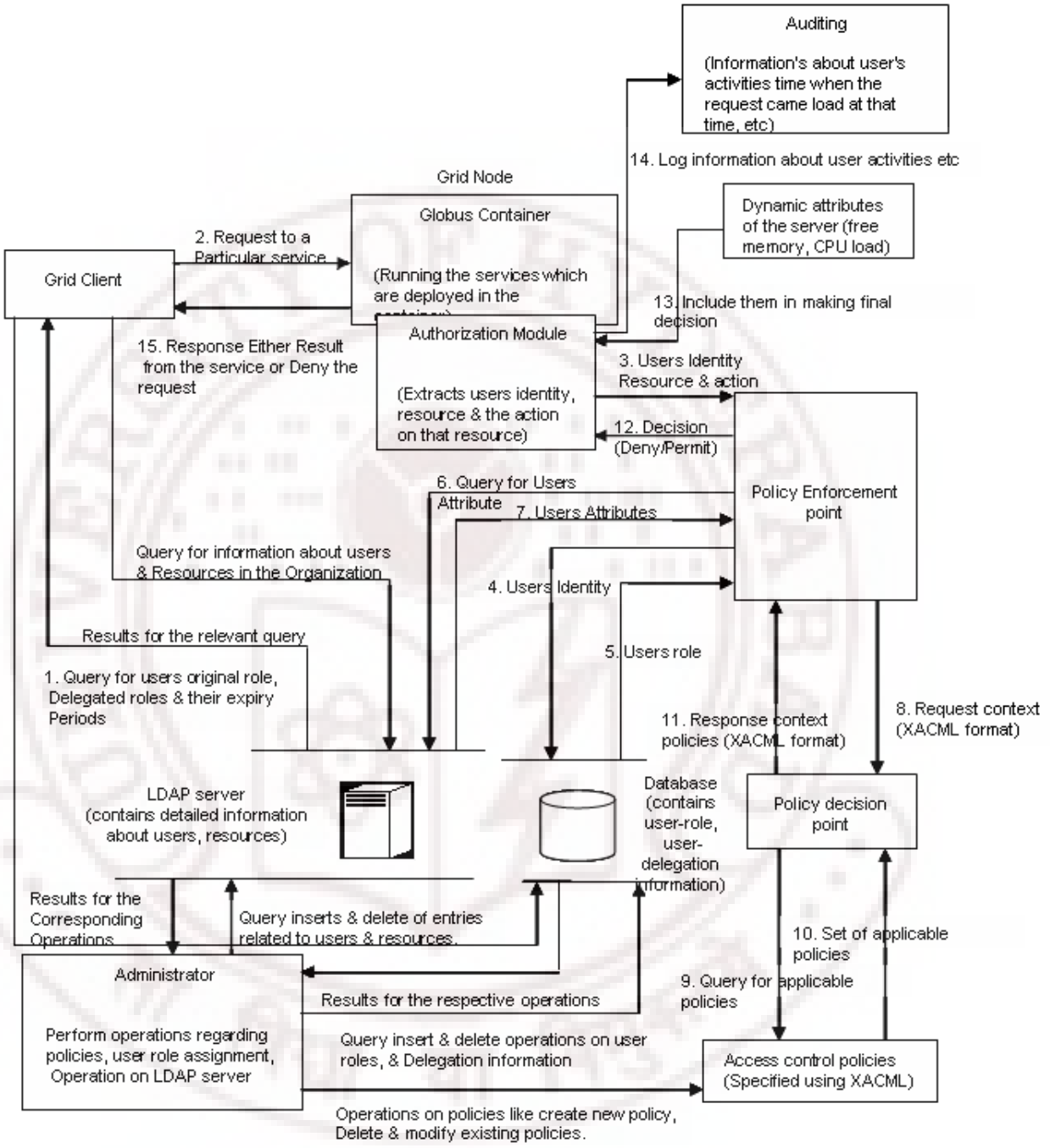


Figure 11: High Level System Architecture for Single Domain Authorization

- **Policy Interface:** This interface helps the administrator to create new access control policies or update the existing policies.

Client Side

A client in the grid environment must have a valid credential in order to access the resources. A grid client is presumed to have obtained a user certificate from the Certificate Authority (CA). Generally clients initiate proxies for most of the operations in the grid. The proxy-certificate has limited life time, generally 12 hours. We do not deal with the authentication aspects of grid client as it is not in the purview of our work. We provided two interfaces for the client:

- **Interface to LDAP server:** With this interface the client can query the LDAP server and get information regarding other users and resources in the organization
- **Interface to Database:** Using this interface the client can find user-role information, log information about past delegations by the client and to the client

Server side: When the *Globus* container is started, it displays all the services which are currently available. Whenever a request for a service comes,

1. The container first checks for the user's valid credential (identity) using the GSI mechanism available with it
2. The custom authorization module controls access to the service with the help of the decision engine (XACML Framework). It also takes account of the system's dynamic attributes (free memory, cpu load) to take final authorization decision. It logs information about the request, the response and other attributes.

3.3.1 Access Control Mechanism

The access control mechanism which we implement takes the following inputs: the user's selected role (by default, user's original role is activated), the services the user requests and the operations the user intends to perform. The outcome of the access control decision i.e. *deny* or *permit* will be logged. The access control mechanism involves three modules namely Custom Authorization Module,

Policy Enforcement Point (PEP) and Policy Decision Point (PDP) according to the XACML [1] standard.

3.3.1.1 Custom Authorization Module

The custom authorization module works in one of the following ways. If the service is not configured to use custom authorization plug-in, then the global client is mapped to its local identity and the client can then access the service. It is a form of coarse-grained access control. Otherwise if the request comes to a service which is configured to use custom authorization plug-in, then the corresponding module will be called. This module extracts the details of the client from the client-request.

- client's identity (distinguished name)
- client-requested service (set of operations)
- particular operation in the service

The authorization module sends the *request (user's Identity, service name, operation)* to the XACML framework (decision engine) via the network.

3.3.1.2 Policy Enforcement Point

The decision engine which runs on a separate system accepts the request. The Policy Enforcement Point (PEP) tries to get the client's selected role, provided the user explicitly selects one of its delegated roles. Otherwise, it tries to get the client's default role (original role) from the database, using client-identity (distinguished name) as the key. If PEP does not get any meaningful role for the client, it sends response to the authorization module with *Deny* as the decision. Otherwise, the PEP gets the role for the client which may be one of the user's delegated roles or the original role. If PEP gets the delegated role of user, then it tries to get the information about the delegated role's expiry period (date, time). If PEP gets the user's original role, it returns the ResponseContext. In case of delegated role, PEP takes current date and time, *if (current(date, time)* $\leqq$ *expiry(date, time) of delegated role*, then through the Policy Decision Point (PDP), the query will be sent to find out the applicable policies. If expiry date and time is less than the current date and time, then it returns *Deny* to the authorization module. If all the above conditions

are satisfied, the PEP may query the LDAP server to get more attributes about the user i.e, for more fine-grained access control policies. Then the PEP formulates the RequestContext with the available information like

- user's role
- resource/service in request
- action/operation
- other attributes of the user/resource (optional)

and sends the RequestContext to Policy Decision Point (PDP).

3.3.1.3 Policy Decision Point

The Policy Decision Point (PDP) parses the RequestContext and extracts the attribute values of elements

- subject (user's role)
- resource (service name)
- action (operation)

The PDP notes the time when it received the RequestContext, checks whether it is within the time-range for accessing a service or not. The time-range for resource access is specified in XACML. If the time when the RequestContext is received is not in the time-range of the service, then PDP sends the ResponseContext to PEP with *Deny* as the final decision. Otherwise, it continues with the next steps. The PDP uses the extracted attributes (from RequestContext) to find the set of applicable policies. For this the PDP matches

- Policy target or
- Rule target
- may be both with the extracted attributes

If any policy or rule target matches with these attributes, then that policy is said to be applicable for this Requestcontext. If there are no policies applicable for the

RequestContext, the PDP sends the ResponseContext to PEP with decision as "Not Applicable". If more than one rule or policy is matched with the RequestContext, the PDP uses the Rule Combining Algorithm or the Policy Combining Algorithm to combine the individual decisions and draw the final decision. The rule or policy combining algorithm is specified in the policy or policy set. Some of the policy/rule combining algorithms are:

- **Deny-overrides:** It allows a single evaluation of *Deny* to take precedence over any number of permit, not applicable results
- **Permit-overrides:** It allows a single evaluation of *Permit* to take precedence over any number of deny, not applicable or indeterminate results

If any final decision is drawn, then PDP prepares the ResponseContext with the final decision either *Deny* or *Permit* and sends the decision to PEP.

The sequences given in Figure 11 work as follows. In step 1, a query for a user' role is sent to the database and once the results are fetched, the grid client sends a request for a particular service to the grid container as shown in step 2. In step 3, the authorization module deployed as part of the Globus container sends the user's identity and the required resource action (operation) to the Policy Enforcement Point (PEP) [Appendix B]. The PEP then sends the user identity to the *rbac* database and fetches the corresponding role as in step 5. In step 6, the PEP queries the LDAP server for additional user attributes. Step 7 shows the reply for this query. After this the PEP sends a RequestContext to the policy Decision Point (PDP)[Appendix B] as shown in step 8. In step 9, a query for applicable policies is sent to the access policy module and the set of applicable policies fetched in step 10. Next in step 11, the PEP receives the ResponseContext from PDP. Based on the ResponseContext, the PEP now sends the access decision to the authorization module. The final authorization decision is made after incorporating the dynamic attributes of the server like the available memory, CPU load etc as shown in step 14. These sequences have been consolidated in the sequence diagram for the single domain authorization mechanism as in Figure 12.

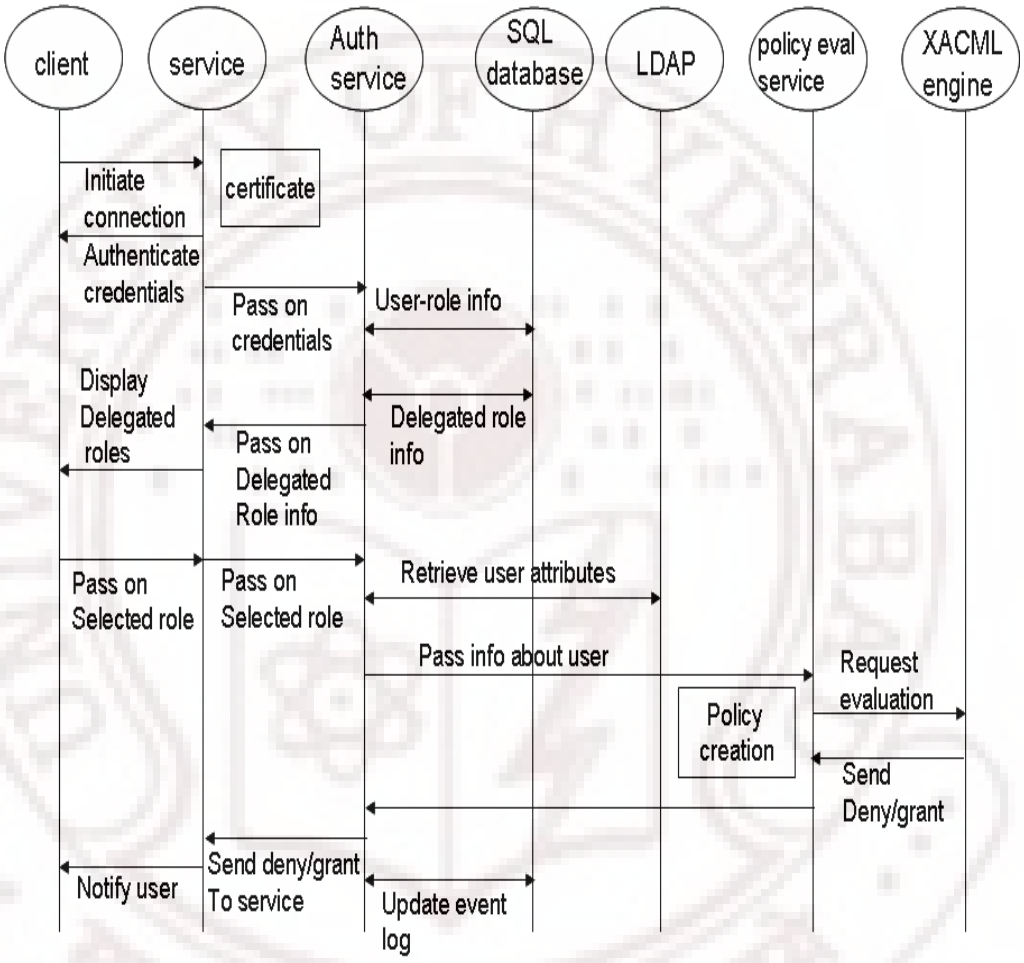


Figure 12: Sequence Diagram for Single Domain Authorization

3.3.2 Database Design

The access control mechanism works in close association with the data base. The data base contains user-role and role-permission mappings. Apart from this, the database also contains information regarding other mappings like role-level, role-delegate, and usr-password. Accordingly, we have designed a database named *rbac* with the following tables.

role_level
(role, level)
usr_role
(usr, role)
usr_delegate
(fusr, tusr, role, cdate, ctime, edate, etime, flag)
role_act
(role, service, act1, act2, act3, act4, act5)
usr_pass
(usr, pass)
usr_selected
(usr, role)

The details of each of the tables in the database is as follows.

- **role_level:** It contains details about roles at each level within the organization. *role* is the primary key for this table
- **usr_role:** contains information about the roles of the users. We have restricted to single role per user. ***usr*** is the primary key for this table
- **usr_delegate:** It gives the delegation information of the user, who is delegated (*fusr* or from user) what role (role) to whom (*tusr* or to user), when it is delegated (*cdate* or commencing date, *ctime* or commencing time) and for how much time it will be valid (*edate* or end date, *etime* or end time), whether the role is further delegatable or not (flag)
- **role_act:** Contains details about what are the services and the operations of that service available to a particular role

- **usr_pass:** This contains information about user's login name and password details
- **usr_selected:** Contains the details of the user's selected role

3.4 System Architecture for Cross-Domain Authorization

A grid system usually consists of more than one domain in a hierarchical/nested fashion. Therefore, cross domain authorization is an essential factor for grid-wide multi domain access control. The first step is to map the role of a given domain to equivalent role in another domain. If the request is from a client in a different domain, we map the roles using the concept of role ranking. Normally, the roles in a domain follow a hierarchical relationship. We make use of this hierarchy, to give the roles a rank on a scale of 10. The request for the resource is passed on to the central authorization server which passes the request to the Authorization Server (AS1) of the domain in which the user is a part of.

The AS1 retrieves the user's role and ranks it in its domain, creates a token and passes it on to central authorization server. It adds the rank for the domain and normalizes the rank of the client on a scale of 1 and passes the token to AS2. Here, the subsequent rank of the source is added, normalized and compared with the rank of the client. If the required rank is greater, then access is denied, otherwise it is granted. If there are more sub domains, then we find the normalized rank with respect to the first common ancestor between the client and the resource. If the user wants to delegate the role, then the user passes on the produced token with the normalized roles. If the user who is being delegated is from a different domain, then the role normalization is done again with respect to the correct ancestor and a token is recreated.

The whole grid environment is separated into different domains and sub domains. The sub domains in every domain are given ranks on a scale of 10. The roles in a local domain are also ranked on a scale of 10 based on their hierarchy. The role in a local domain is translated to its global ranking based on the value of its own ranking and also the rank of its ancestor domains up the tree. We represent the role-mapping architecture as a weighted tree and arrive at the globally mapped

role by comparing the global rank of a role with respect to its first common ancestor. The hierarchy of domains is represented as in Figure 13.

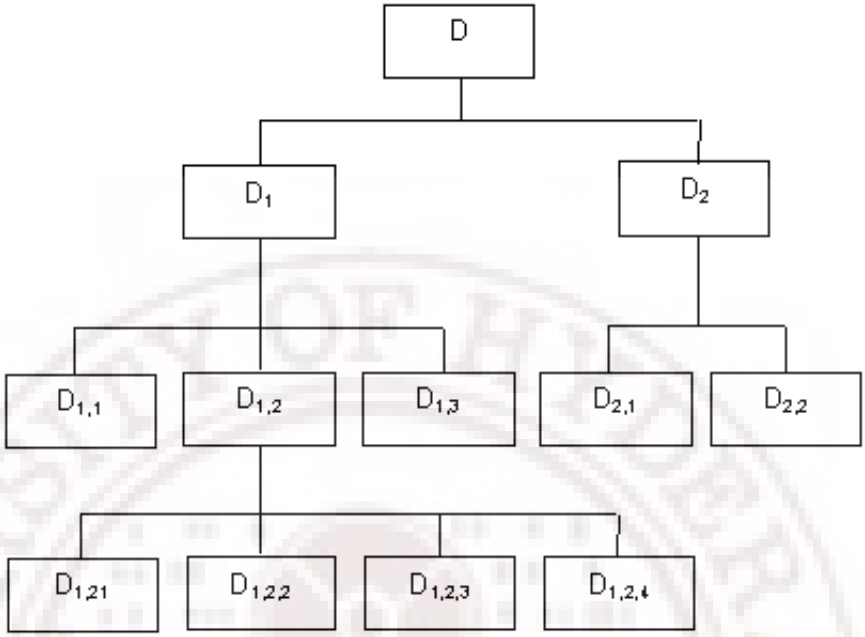


Figure 13: Hierarchy of Domains

The details of the user's delegated and revoked roles between two domains are stored with their common ancestor central server. As detailed in Chapter I, the RBAC standard uses *role* as the basic unit of authorization and incorporates features such as role hierarchy [102], static and dynamic separation of duties and so on [75]. In a typical RBAC environment, a user is assigned roles based on the responsibilities of the user in the organization. For example, in a university domain, the potential roles could be Professor, Associate Professor and so on. For an industrial domain the roles could be CEO, General Manager, Manager and so on. The semantics of roles in a given domain will not have relevance in another domain. The role in an organization has to be mapped to its corresponding meaning in another to make cross-domain resource sharing possible.

We address this issue with a ranking based weighted role approach. Our architecture enables mapping of a local role to a global ranking. We consider a nested and hierarchical domain architecture reflecting the real life grid scenario. The roles in a particular domain follow a local role hierarchy. The cross domain architecture consists of the following components.

- At the organizational level we consider two Domains A and B
- Domains A and B consist of sub-Domains A_ a and B_ b
- Further, Domains A_ a and B_ b have grid nodes as their constituents
- Authorization Server1 (AS1) is the local Authorization server for grid nodes from Domain A_ a
- Authorization Server2 (AS2) plays a similar role in Domain B_ b
- Ranking servers RS1 and RS2 for the two respective Domains A and B store the rating of the subdomains

Figure 14 depicts the role mapping architecture which forms the basis for ranking of roles across domains. A simple view of the individual authorization server architecture is shown in Figure 15.

Figure 16 shows the user-role ratings for authorization.

The grid user is granted/denied access to the requested resource through the following procedure.

1. The user from Domain A_ a sends his identity, path, the requested resource and also requested operation to Domain B_ b
2. The user in domain b forwards the details to its Authorization Server (AS2) and awaits a deny or grant
3. The Authorization Server AS2 executes the algorithm *Authorize*
4. The credentials are passed up the hierarchy for role mapping as the user is from a different domain
5. The credentials reach AS1 where the attributes like the user's role, rating etc are retrieved and a Token is created.
6. The Token follows the same path in reverse and at every stage, the rating of the domain gets weighted
7. AS2 gets the final version of the Token. The general expression by which the user rating is computed is as follows.

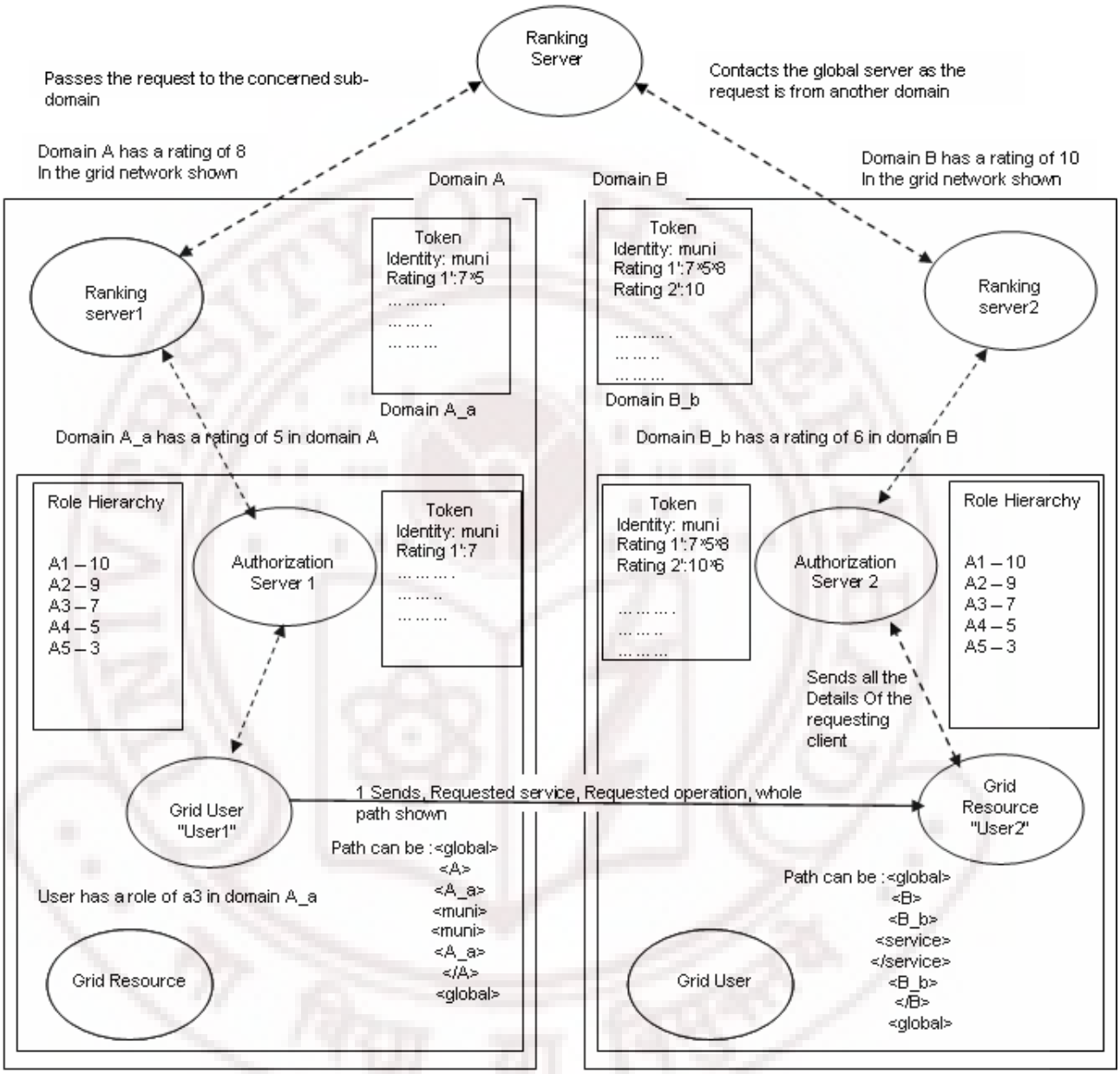


Figure 14: Cross Domain Role Mapping Architecture

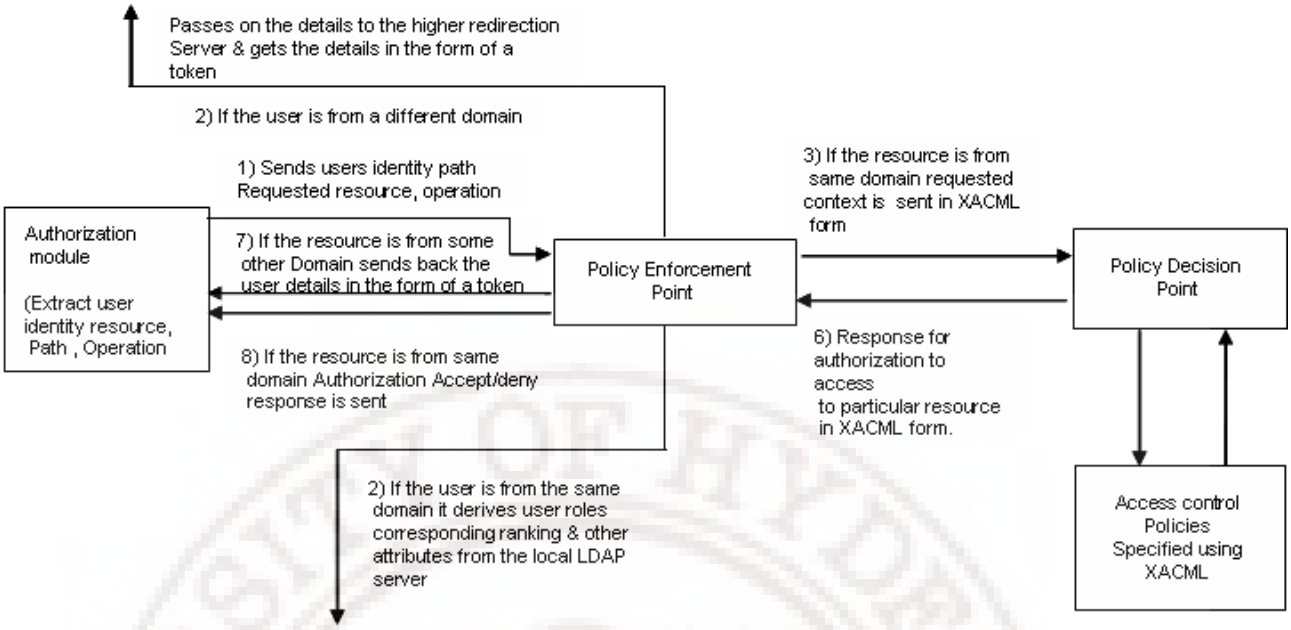


Figure 15: Authorization Server Architecture

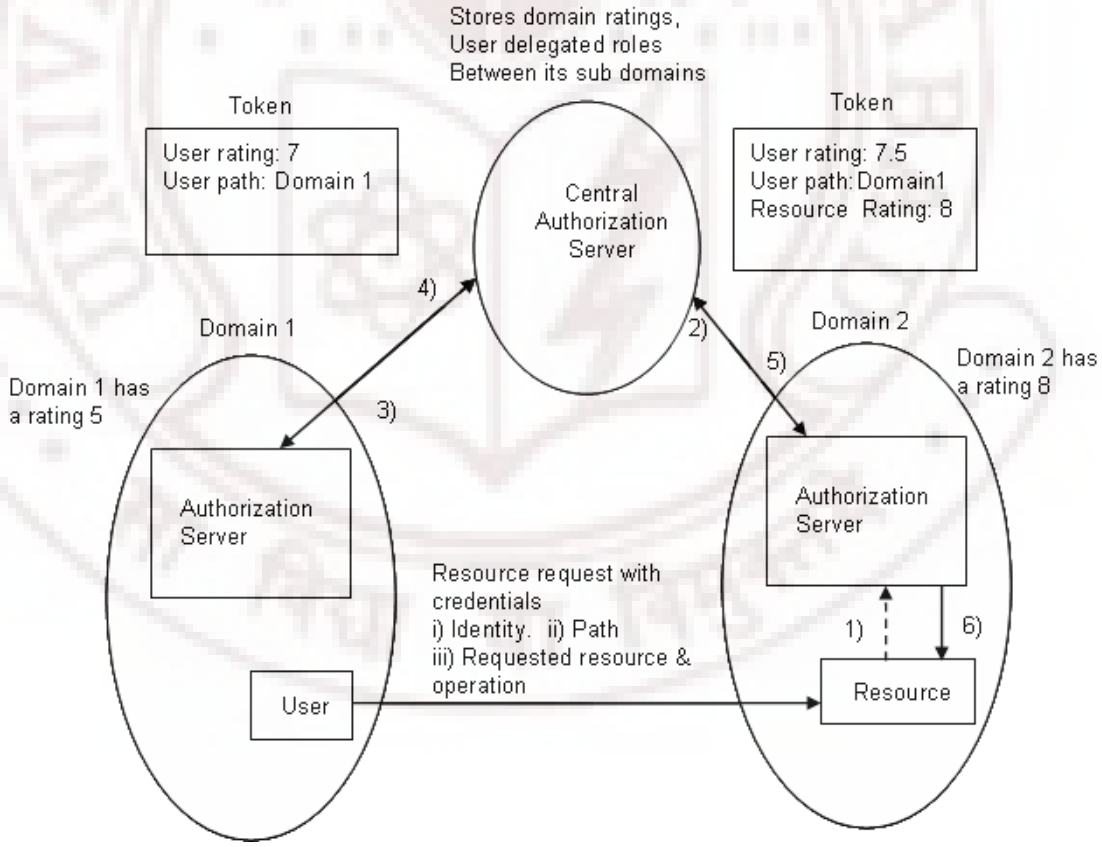


Figure 16: User-Role Ratings for Authorization

The normalized user rating = (local rank of the role * rank of the sub domain * rank of the domain) / 10 * 10 * 10

For example, for a grid user A3 in Domain A the normalized user rating will be 7*5*8 / 10*10*10 = 0.280

8. AS2 finds the minimal rating of the role needed to access the resource. The normalized rating of the resource is similarly found as 10*8*5 / 10*10*10 = 0.400

9. AS2 can integrate this ranking comparison with other local policies to either deny or grant access to the user

Here, we give a generic algorithm for mapping the role of a domain to its equivalent role in another domain as shown in Algorithm 1. We also describe an algorithm for authorization which is presented in Algorithm 2.

Algorithm 1 Rolemap

```
Input : User Credentials(Token)
Output : Token containing updated values
  begin
    if(user is from different domain or sub-Domain)
        Token T = call Rolemap(credentials)
        Add rating of the sub-Domain to
        the Global rating of the resource in the Token
        return T
    if(user is from same Domain)
        Token T = call Authorize(Credentials)
        Add Rating of Sub - Domain to T
        Return T
  end
```

Algorithm 2 Authorize

```
Input : User Credentials
Output : Grant/Deny a Token
  begin
    If(user is from a different domain)
    call Rolemap(credentials)
    find minimum rated role to access resource in the domain
    find the normalized global rating of that role GLR
    retrieve the Normalized global rating of user from certificate GLU
    if GLU≥GLR
        return accept
    else
        return deny
    else
        retrieve user roles, rating and other credentials from database
    if(resource is from other domain)
        create token T containing the user details
        return T
    else
        find minimum rated role MR to access resource in the domain
    if(MR > user's role-rating) return accept
    else return deny
  end
```

3.5 Chapter Summary

The framework proposed in this chapter contributes to the enforcement of grid computing security in a major way. First, we propose architectural frameworks for authorization within a grid enterprise. Later we extend it to suit the requirements of authorization across domains, where inter-operability and security go hand in hand. The role mapping architecture helps the grid nodes across domains to map the roles and thereby interact and authorize users for resource access. This architecture also supports reusability of role-ranking mechanism, as the token once created between two domains can be used for future interactions between them. Domains can also formulate additional access control policies like giving access to only semantically sensible organizations for that particular resource, apart from just comparing the global ranking of the user. Both the architectures have also been represented through sequence diagrams. In total, these architectures form the core of the grid access control and authorization systems.

CHAPTER IV

ROLE-BASED GRID DELEGATION MODEL

4.1 Introduction

We discussed two major concerns of grid computing security namely direct authorization and secure inter-operability [91] in the previous chapter. In this chapter we address another equally important aspect of grid security namely, indirect authorization (delegation). Grid delegation is a procedure by which a valid user endows another user or program or a service with the ability to act on that users behalf. Delegation is in fact the primary form of authorization in grids. The large and geographically distributed, dynamic, heterogeneous and scalable grid environment poses unique delegation requirements. Presently there are no standard mechanisms to guide grid delegation. As credential delegation (proxy delegation) has its own limitations in a dynamic grid environment, the need for a new conceptual model is felt to effectively formulate the grid delegation requirements. Here, we present a framework called Role- Based Grid Delegation Model (RB-GDM) for delegating access rights in grids. The basic unit of delegation in our model is *role*. Derived from the standard RBAC formalisms, this framework explores various approaches for authorization and revocation of delegation.

The chapter begins with an introduction to the delegation requirements in grid environment. The rest of the chapter is organized as follows. We first discuss the delegation requirements in grids in section 2. We also present the existing approaches to delegation. In section 3, we propose the role-based grid delegation framework for implementing delegation and revocation in grids. The RB-GDM interconnection framework has been presented in section 4. It includes intra-domain as well as inter-domain topologies. We also extend the delegation model to meet the requirements of delegation and revocation across domains (inter-domain or cross-domain delegation. Finally section 5 summarizes the chapter.

4.2 Delegation Requirements in Grids

Delegation is an important aspect of grid security. In general, delegation is defined as the action whereby an active entity in a system authorizes (delegates) its authority to another entity to execute some functions on its behalf. Delegation in computing systems can take many forms: human to human, human to machine, machine to machine and even machine to human [200]. The consequence of delegation is propagation of access rights [172]. Access rights propagation in a decentralized collaborative system like the grid presents difficult challenges for traditional access control mechanisms and models.

A delegation activity in a grid environment needs to be monitored and controlled in such a manner that the resource inside the domain can stay protected. Though various forms of delegation are possible, the most common delegation activities in present day grids are user to user delegation and user to machine delegation. Most of the resource access in grids takes place through remote authorization and is achieved by user to machine/process delegation. User to machine delegation is the mechanism whereby a user in a distributed environment authorizes a system to access remote resources on his behalf [93]. In some cases the user may delegate the rights to one of the several permissible roles or identities [110]; in order to limit the actions of the process to some subset of the rights that user is authorized to. This form of delegation is generally referred to as *limited* or *restricted delegation*.

The support of roles is crucial in an open computing environment like the grids, where not all individuals may be registered with a service and requests may arrive from previously unknown parties. In such a context, the ability to invoke a service often depends on the *capacity in which a user can operate* rather than on *who the user actually is*. This capacity is characterized as a *role*. The concept of roles is different from that of groups in the sense that unlike groups, roles carry a dynamic behavior and can be activated and deactivated by users and service providers as required. Though current grid systems support delegation through impersonation (by issuing a proxy certificate), the revocation of delegation tracing and multi step delegation still remain as the major challenges.

We propose a grid delegation approach built on Role Based Access Control

(RBAC). RBAC has been accepted in the access control design and architecture of security in enterprises as an alternative to traditional discretionary and mandatory access control models. RBAC is designed to suit large-scale enterprise-wide systems and works on the notion that permissions are associated with roles. As explained in Chapter I of the thesis, the users are assigned to appropriate roles and users acquire permissions by being members of roles. Roles can be granted new permissions and can be easily revoked from roles as and when needed. Thus RBAC provides a way to empower individual users and service providers through role-based delegation [120] in a fully distributed environment like grids.

4.2.1 Existing Delegation Approaches

The core theme behind delegation is that of delegating rights, obligations, privileges and authority. The basic idea of role-based delegation is that users themselves may delegate role authorities (rights) to others to carry out some functions authorized to the former [45]. Related delegation approaches like [82], [83] generally assumes that the rights to be delegated have already been decided by a separate mechanism, and it is just a matter of encoding and checking these rights. The general need for delegation is exemplified by the Digital Distributed System Security Architecture, a comprehensive collection of security services and mechanisms for general-purpose distributed systems [27]. Gasser.M and McDermott.E [157] discuss various issues on implementing delegation in the distributed environments. Target restrictions and delegate restrictions are the common forms of delegation-related restrictions. Calvelli et.al [25] analyzed the use of delegation protocols for distributed systems. [63] suggested a weighted graph approach to authorization delegation and conflict resolution.

Delegation of rights is a significant problem in middleware like Globus. Grid Security Infrastructure, the security component of Globus supports an interface for delegation. But, it is based on credentials and proxies. The reliance on GSS-API in Globus hinders efforts at delegation in Globus, because GSS-API does not offer a clear solution to delegation and impedes restricted delegation. Delegating the user's full rights could result in impersonation. There have been some efforts in Globus to adopt and support the Generic Authorization and Access- Control API

(GAA API). However, it does not propose any systematic procedure by which we can determine the rights that should be delegated. Mehran et.al in [101] suggested a delegation protocol for grid computing systems.

The study conducted on the existing delegation attempts lead to the conclusion that delegation, in its present form is not effective to meet the grid authorization requirements. We, therefore, present a delegation model for grid resource access based upon roles through which only the rights assigned to roles can be delegated, thereby making it fine-grained. Unlike the existing identity-based approaches, this framework treats grid delegation as an authorization problem.

4.3 Grid Delegation Framework

In this section, we present a grid delegation approach called the Role-Based Grid Delegation Model (RB-GDM). This model is aimed at supporting role-based delegation and revocation. Our work is built on the following reference models: RBAC96 model [155], the RBDM0 model [45], RDM2000- A Rule-Based Framework for Role-Based Delegation and Revocation [200], Rule-Based Delegation Model 1 (RBDM1) [46] and RB-GACA: A RBAC Based Grid Access Control Architecture [202].

4.3.1 Basic Elements from Reference Models

The Role Based Access Control Model (RBAC) has the following basic elements: users U, roles R and permissions P. Roles are assigned to individual users and permissions are assigned to roles. According to this model, a *user* is an individual; a *role* is a job function indicative of the responsibility and *permission* indicates the approval to execute some method or perform some action. We generalize the term *user* to make it represent an individual, an intelligent autonomous agent, or a machine in the grid environment. We consider the end user of a grid resource as the basic user or original user. There are two sets of users associated with a role r : as given in.

- Original users: Those users who are directly assigned roles
- Delegated users: Those users who are indirectly assigned (delegated) the roles by original users

Constraints on User to User Role Delegation

1. At any time a user can have its original role and the roles which it has acquired through delegation by other users
2. A user is authorized to delegate its original role only or the roles it has acquired through delegation. It is meaningless for a user to delegate other users' roles
3. Roles follow a hierarchy. Thus a user with lower role is not authorized to delegate a role to a user of higher role
4. It is meaningless for a user to delegate a particular role to another user to which he already belongs
5. A user of higher role has privileges to delegate its junior roles also. This enforces the principle of least privilege
6. A user will not be able to delegate a role which has no further delegation allowed

Each constituent domain of the grid virtual domain (VO) may have its own role hierarchies. As mentioned in Chapter I, a role hierarchy is a structure reflecting an organization's lines of authority and responsibility and is represented using a partial ordering operator $\geq$. For example, if role1$\geq$role2, then *role1* inherits permissions of *role2*. Though role hierarchies [46] in different local domains vary in their semantics, they are required to enforce the "Principle of Least Privilege" [17] which states that in a particular abstraction layer of a computing environment, every module (such as a process, a user or a program on the basis of the layer we are considering) must be able to access only such information and resources that are necessary to its legitimate purpose. The fundamental components of RBAC and RBDM0 which form the basis for our model (named RB-GDM) are depicted as follows in Table 3 and Table 4 respectively. These reference models have been detailed in [155] and [45].

Table 3: RBAC Components

• U,R,P,S are the sets of users, roles,permissions and sessions respectively • UA $\subseteq U \times R$ is a many to many user-to-role assignment • PA $\subseteq P \times R$ is a many-to-many permission to role assignment relation, where P and R are the set of permissions and roles respectively • RH $\subseteq R \times R$ is a partially ordered hierarchy

Table 4: RBDM0 Components:

• UA = UAO $\cup$ UAD,where • UAO $\subseteq U \times R$ is a many to many original user-to-role assignment relation • UAD $\subseteq U \times$ R is a many to many delegated user-to-role assignment relation • Users : $R \rightarrow 2^U$ is a mapping of each role to a set of users Users(r) = $\{u \mid (u, r) \in UA\}$ • Users(r) = Users_ $O(r) \cup Users\_ D(r)$,where users O(r) and D(r) are the original users and the delegated users respectively.

4.3.2 Role-Based Delegation Model

The delegation model RBDM0 [45] suggests only user-to-user delegation, which is the simplest form of delegation. Our model is grid specific. As mentioned earlier, the end-user is considered as the basic user. The end-users are geographically distributed, can be dynamic and may get disconnected from the virtual organization (VO) after the submission of a job. Therefore, end-user-to-end-user delegation is not possible grids. As the end user cannot directly delegate an entity, we need to look at other forms like user-to-machine, machine-to-machine or user-to-process delegation. The basic idea of delegation is to decide *who is delegating or revoking what to whom*. Thus the entities in delegation are the grantor, the grantee and the rights granted [182]. To suit the requirements of a grid environment and to distinguish delegation from direct authorization, we call the grantor of delegation as a delegator, the grantee as delegatee and the rights that are granted through the roles as delegated rights.

As discussed in Chapter I, the RBAC role hierarchy includes flat roles and hierarchical roles. For flat roles, no inheritance of permissions between roles is involved. With regard to delegation, we define roles as either further delegatable or non-delegatable based on whether a subset of roles can further be delegated or

not.

The first challenge in framing a grid delegation model is to identify the roles assigned to the users in the grid environment. The subject or active entity in a grid transaction can be:

- a user (an individual)
- a process or
- a machine

Among users, there can be:

- administrators (who possess all the administrative and super user privileges -eg.a domain administrator)
- basic users (who can submit large scale jobs, access resources etc)
- developers (whose main objective is to handle the programming and middleware aspects) or
- guest users (who can only browse the information or submit very small scale jobs)

The components that constitute a grid delegation mechanism include the delegator, the delegatee, the delegated rights and the associated constraints. Multi-step delegation is possible with delegatable roles whereas single step delegation applies to roles which are non-delegatable. Sometimes the delegator grants full permissions to the delegatee. We call this as total delegation. If the delegator assigns only a subset of the rights to the delegatee, it is called partial delegation. We make the following design criteria in RB-GDM.

- A role is considered to be the basic unit of delegation and is a job function
- The term user in our model represents an end-user, a process or a machine in the broader perspective
- Delegation between entities in the same role is not allowed since the same roles will have same permissions

- Both single step delegation and multi-step delegation are permissible
- Delegation can be granted in total or can be partial
- Each delegating role r has two types of members namely the original members O_r and delegated members D_r
- Grid user can be a domain administrator, an advanced user, a basic user or a guest user

Before we build a model, it is essential to define the components which constitute the model. Therefore we present the following definitions 1 to 4 which are detailed in Table 5, Table 6, Table 7 and Table 8.

1. In Definition 1, the basic components like grid domain, users, roles, permissions and the user-role, role-permission assignments which are required to build the Role-Based Grid Delegation Model (RB-GDM) are explained
2. Definition 2 explains how the access rights can be delegated through authorization.
3. Definition 3 shows as to how we can give different role permissions like delegable and nondelegable
4. Definition 4 depicts the meaning of "revocation of delegation"

The interaction of these components facilitates a delegation mechanism as per the algorithms for authorization and revocation of delegation. Algorithm 3 gives the method for authorization of delegation as per Definition 2. Algorithm 4, 5 and 6 show the methods for revoking the delegated rights by various means such as grant-dependent, grant-independent or time-out. First, we present the intra-domain (single domain) delegation procedure and in the next subsection, we show as to how we can perform delegation across domains.

There are two possible ways in which revocation can be done. The first approach is by revocation using time out (attaching a time constraint to every assigned delegation). Timeout revocation though self triggering, is not enough to ensure secured delegation. For tasks, which take much longer periods to complete, this method is not suitable. Tracking the delegator is another issue with this type

Table 5: Definition 1 RB-GDM Components

• Let D = $\{D_l\}$, where l = 1 to N be the set of domains of a Grid G • $U^l = \{u^l{}_i\}$, where i = 1 to m_l be the set of users of D_l domain • $R^l = \{r^l{}_j\}$, where j = 1 to n_l be the set of roles of D_l • $P^l = \{p^l{}_k\}$, where k = 1 to q_l be the set of permissions of R^l • $UA_l \subseteq U^l \times R^l$ is the set of relations of user to role assignment in D_l where $UA_l = UAO_l \cup UAD_l$ • $PA_l \subseteq P^l \times R^l$ is the set of relations of permission to role assignment in D_l • $U^l(r^l{}_j) : R^l \rightarrow 2^{U^l}$ is a function derived from UA_l mapping each role r to a set of users where $U^l(r^l{}_j) = \{u^l{}_i | (u^l{}_i, r^l{}_j) \in UA_l\}$ • $P^l(r^l{}_j) : R^l \rightarrow 2^{P^l}$ is a function derived from PA_l, mapping each role r to a set of permissions, where $P^l(r^l{}_j) = \{p^l{}_k | (p^l{}_k, r^l{}_j) \in PA_l\}$

Table 6: Definition 2 RB-GDM Delegation

RB-GDM controls role-based delegation by means of predicate can_$delegate \subseteq R^l \times R^{l*}$ between two domains of a grid G. It is a non-reflexive relation since the user in a role cannot delegate his membership to another user in the same role. The meaning of the above relation is that if $u^l{}_1 \in R^l$ and $u^l{}_2 \in R^{l*}$, then ($r^l{}_1$,$r^l{}_2$) $\in$ can_$delegate$,i.e., ($u^l{}_2$, $r^l{}_1$) $\in UAD$, provided the constraints are satisfied.

Table 7: Definition 3 RB-GDM Role-Permissions

Each role can have two types of permissions • Delegatable permissions PD_l • Non-delegatable permissions PN_l From the RBDM0 model, we extend the following permission components for RB-GDM, where P^l is a set of regular permissions • $PA_l = PDA_l \cup PNA_l$, where PA_l,PDA_l and PNA_l are $\subseteq P^l \times R^l$ which include many to many role to permission assignment relation, many to many delegatable role to permission assignment relation and many to many non-delegatable role to permission assignment relation respectively. • $P^l(r^l{}_j) = \{p^l{}_k | (p^l{}_k, r) \in PA_l\}$ Similarly for $PDA^l(r^l{}_j)$ and $PNA^l(r^l{}_j)$ as well

of delegation and revocation. These issues can be handled by explicit revocation (either grant-dependent or grant independent).

Table 8: Definition 4 RB-GDM Revocation

The revocation procedure can be through any of the following ways • Remove one or more pieces of permissions from delegation • Revoke delegation role owned by a fixed delegable role • Remove one or more pieces of permissions from a fixed delegable role to its regular (original) role

The procedure for role-based delegation is shown in algorithm 3 below. It shows how the permission is associated with the delegated role and how delegation is granted or denied. Revocation of delegation is as important and crucial as

Algorithm 3 Role-Based Delegation

Input:
[1] A delegation request ($u^l{}_i$,$r^l{}_j$)
[2] UA_l
[3] PA_l
Output:
True, if the delegation is granted
False, if delegation is denied
Begin
Step 1: $U^l(r^l{}_j) \leftarrow$ get(UA_l); /* from Definition 2, returns the user-role assignment set for the domain*/ UA_l is such that $UA_l = UAO_l \cup UAD_l$
Step 2: for $U^l(r^l{}_j) = \{u^l{}_i | (u^l{}_i, r^l{}_j) \in UA_l\}$
if $P^l(r^l{}_j) = \{p^l{}_k | (p^l{}_k, r^l{}_j) \in PDA_l\}$ /* returns true if the permission is associated with the delegated role and delegation is granted, otherwise the condition returns false and delegation is denied. This decision is guided by the policies of the role-permission relationship set of the given domain*/
if $((u^l{}_i, r^l{}_j) \in can\_delegate)$
Step 3:return true;
else
Step 4:return false;
End
/*Example : Let the delegated user be in a role "basic user". It sends a request for delegation along with the role and access specifications. The delegator for example, an administrator maps the user-role relations and the role-permission relations and grants or rejects delegation accordingly.*/

authorization of delegation. The revocation procedure (Grant Dependent) is done as represented in the algorithm 4.

Algorithm 4 Revocation of Delegation-Grant Dependent

Input:
[1] UA_l
Output:
True, if the delegation is revoked
False, if delegation is not revoked
Begin
if($(u^l{}_i, r^l{}_j) \in can\_revoke$)
Step 1:return true;
else
Step 2:return false;
End
/* Example : Let the delegatee user be a process and the delegator be an administrator. The execution of the algorithm results in revocation of delegated rights by the delegator (in this case the administrator) himself.*/

Algorithm 5 shows the steps by which grant-independent delegation is revoked. Algorithm 6 gives the methodology for delegation revocation based on

Algorithm 5 Revocation of Delegation-Grant Independent

Input:
[1] UA_l
Output:
True, if the delegation is revoked
False, if delegation is not revoked
Begin
if (($u^l{}_i, r^l{}_j$) $\in can\_revoke$*)
Step 1: return true;
/*where $u^l{}_i$ belongs to the original user set UAO_l*/ **else**
Step 2:return false;
End
/* Example : Let us consider a user $\{u^l{}_i\}$ in a role "basic user" and the delegator be another basic user who is granted rights by a site administrator. When this algorithm is invoked, it matches the user (delegatee) with its corresponding role i.e., *guest* and the delegated rights are revoked by the administrator and not by the delegator.*/

timeout concept.

4.4 RB-GDM Interconnection Framework

We propose three approaches, one for intra-domain access control and the remaining two for inter-domain access control [167]. These interconnection networks are implemented using realized based on the definitions 1 to 4 given in section 4.3.2.

Algorithm 6 Revocation of Delegation-Using Timeout

Input:
[1] UA_l
[2] t (Duration of delegation)
Output:
True, if the delegation is revoked
False, if delegation is not revoked
Begin
if ($(u^l{}_i, r^l{}_j) \in can\_revoke(t)$
Step 1: return true;
/*t is the duration after which the delegation is revoked*/ **else**
Step 2:return false;
End
/* Example : Consider a user $\{u^l{}_i\}$ in a role "guest". When this algorithm is invoked, it matches the user with its corresponding role i.e., *guest* and if the duration *t* has expired, the delegated rights are revoked.*/

The interactions between the components of the RB-GDM are governed by the algorithms . The basic procedure to authorize and revoke delegation is the same for all the three interconnection networks except that RB-GDM components will be organized in different forms in each case.

4.4.1 Intra-Domain Role Based Delegation

It is essentially similar to an enterprise wide role delegation where the entities within a domain delegate tasks to others using a hierarchical relationship. The appropriate association would be that of the superior entity delegating to the subordinate. Figure 17 illustrates this framework. The RB-GDM core model that supports intra-domain role-based delegation is shown in Figure 18.

4.4.2 Inter-Domain Role Based Delegation

Role-based delegation across domains can be done mainly in two ways; by treating the two domains as equals (peers) in the association or by considering a master-slave relationship.

4.4.2.1 Peer-to-Peer Role-Based Delegation

This approach is suitable when transactions take place between two domains of flat association i.e. there exists no hierarchical relationship (for example, between two

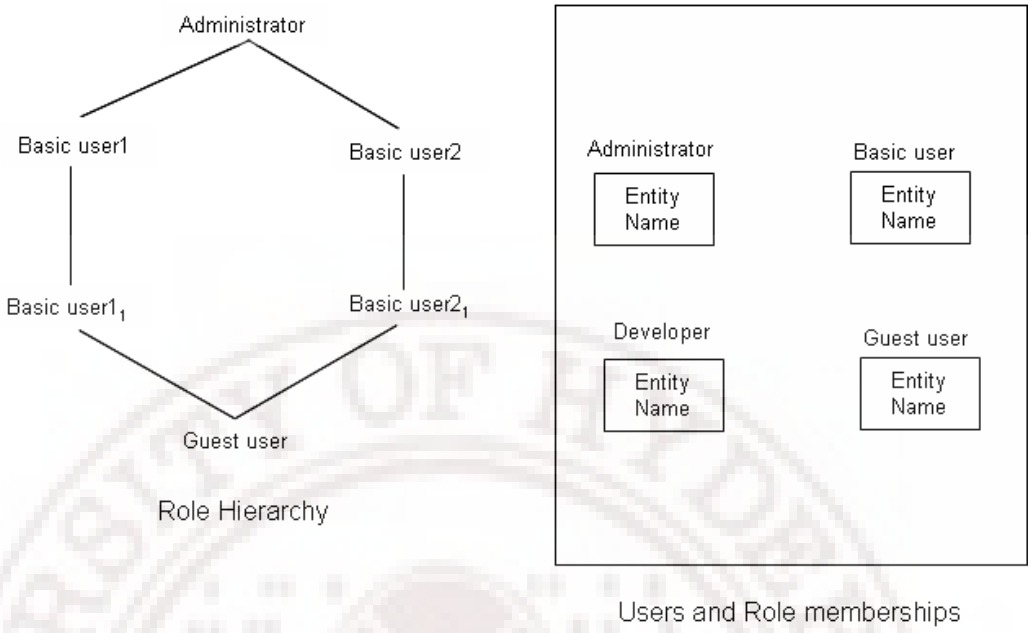


Figure 17: Intra-Domain Role-Delegation Framework

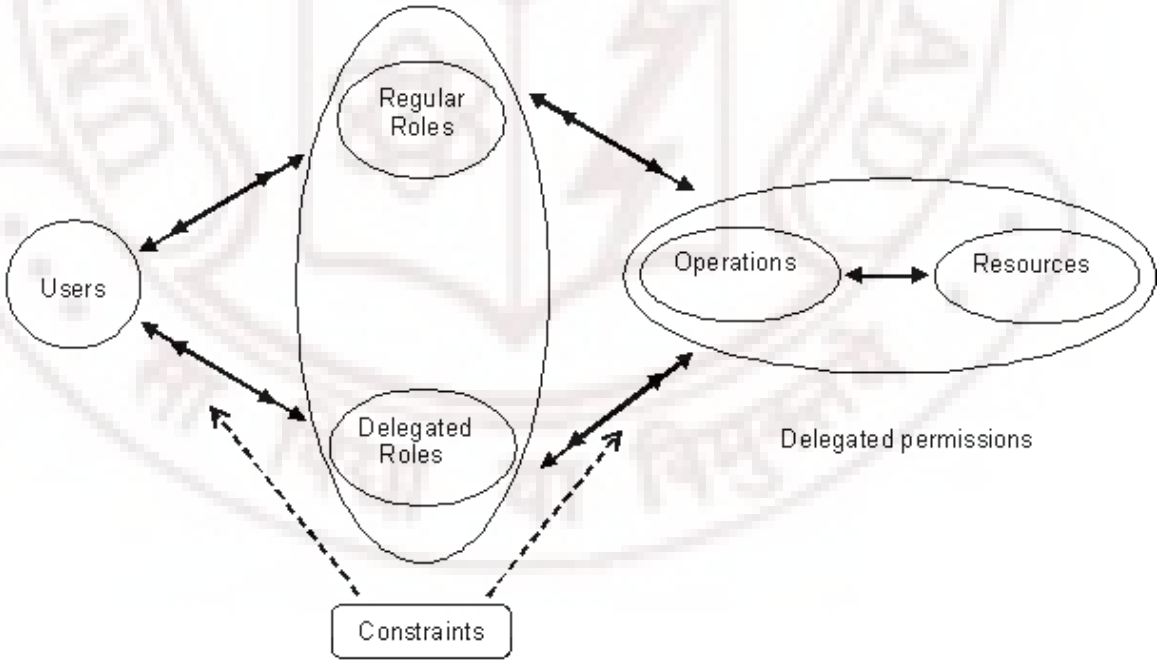


Figure 18: RB-GDM Core Model

investment banking domains). This type of delegation is based on the mapping between a role in the delegator domain to its corresponding equivalent role in the delegatee domain both being peer roles. Figure 19 represents the peer-to-peer interconnection framework.

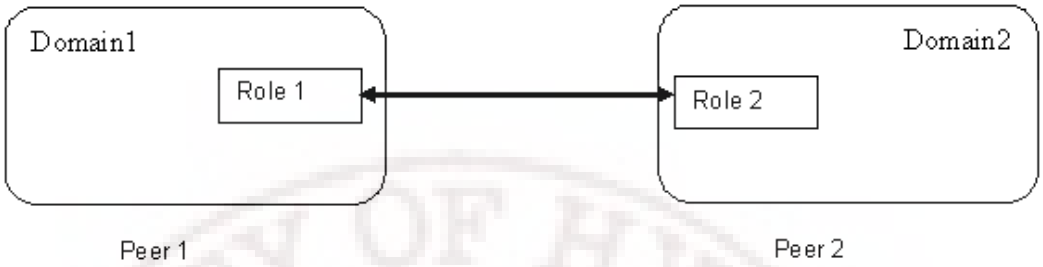


Figure 19: Peer to Peer Interconnection Framework

The RB-GDM (Role-Based Grid Delegation Model) for peer-to-peer set up is as in Figure 20.

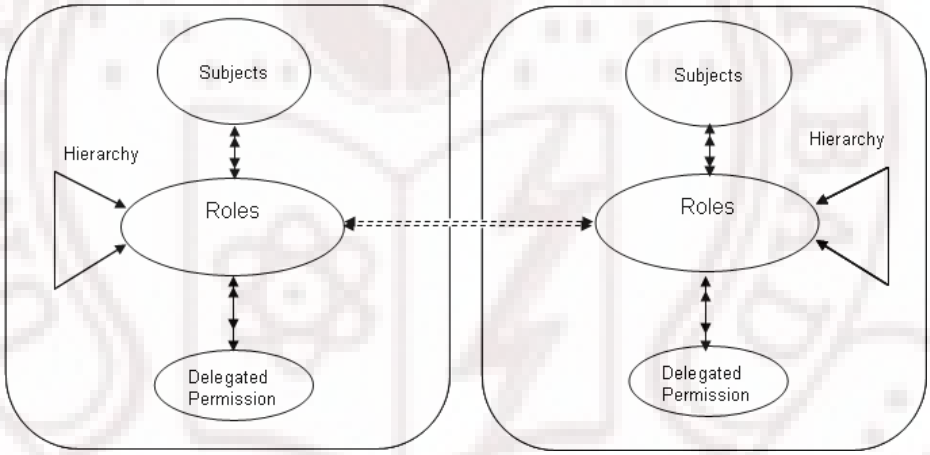


Figure 20: Peer-to-Peer Inter Domain RB-GDM

4.4.2.2 Master/Slave Role Based Delegation

This approach helps to grant role delegation between two domains, which have a master/slave kind of association between them. It implies that the roles of one domain act as superiors to the roles of another domain. An example of such an arrangement is the association between a central bank domain and a subsidiary bank domain (hierarchy between two different domains). Typical hierarchical domain relationship can also exist in a grid between a virtual organization (VO) or

virtual domain and the member domains of the VO.

Figure 21 shows the hierarchy topology for inter-domain role delegation.

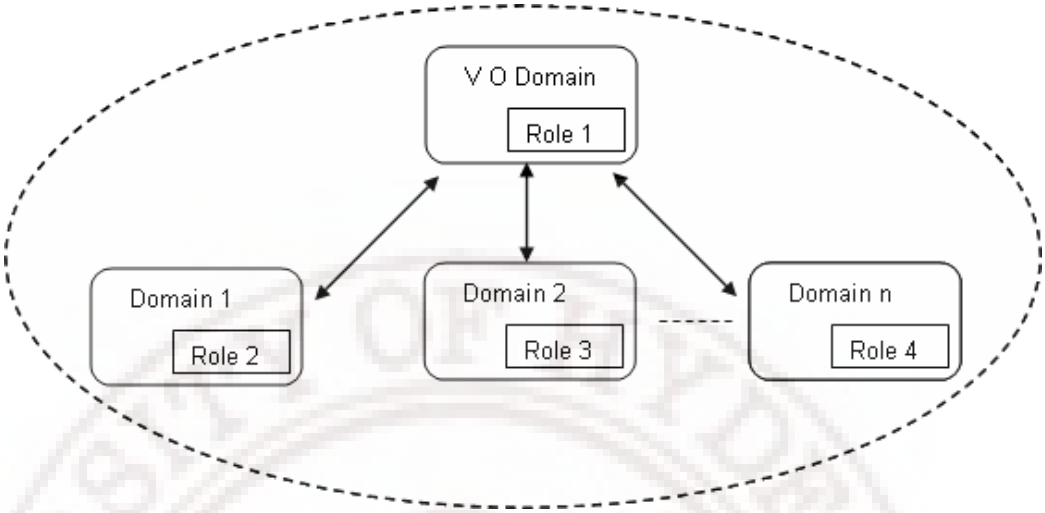


Figure 21: Hierarchy Topology of Inter Domain Role Delegation

The RB-GDM for master/slave inter domain role based delegation is as in Figure 22.

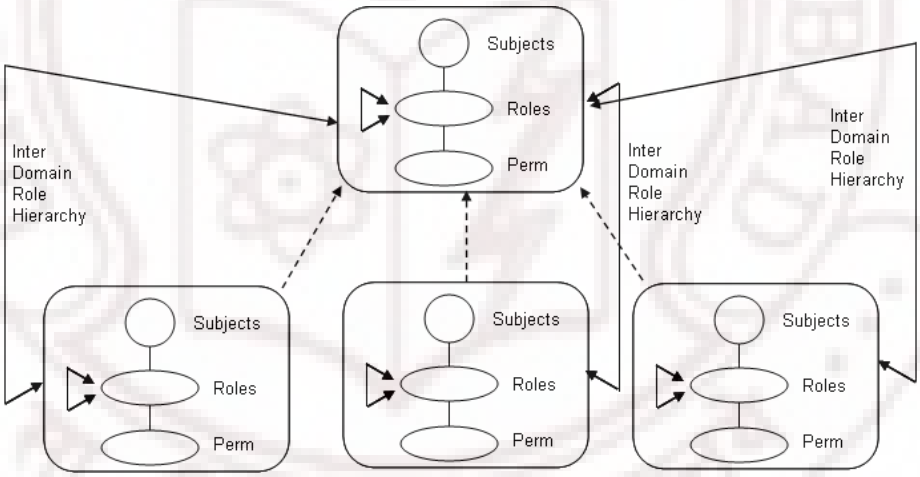


Figure 22: Master/Slave Inter-Domain RB-GDM

4.4.3 Inter-Domain/Multi-Domain Role Delegation Model

We propose a grid delegation model for delegation across domains. Jajodia et.al [84] proposed a generic model for multi-domain delegation. Our model is based on the role-equivalence established between roles of different domains. We rank

the roles using the role mapping architecture [168] which we proposed in Chapter III. When a request comes from a user in one domain for a resource in another domain, we derive the user's equivalent role in the second domain. Assume the grid user *user1* has a role *a3* as shown in Figure 14. He wants to access a resource which requires a minimum role of *b4* in the hierarchy. Since the global rank of *a3* is more than that of *b4*, user1 will be granted access. At the same time he will be given a temporary role of *b2* in the domain B_b. Our model takes care that the global rank of the temporary assigned role is closely equal to the global rank of the requested role. Now in the above mentioned scenario the user 'user1' will be assigned a temporary role of *b2* (suppose) in the domain B_b. Figure 23 shows the sequence diagram for cross-domain role-mapping.

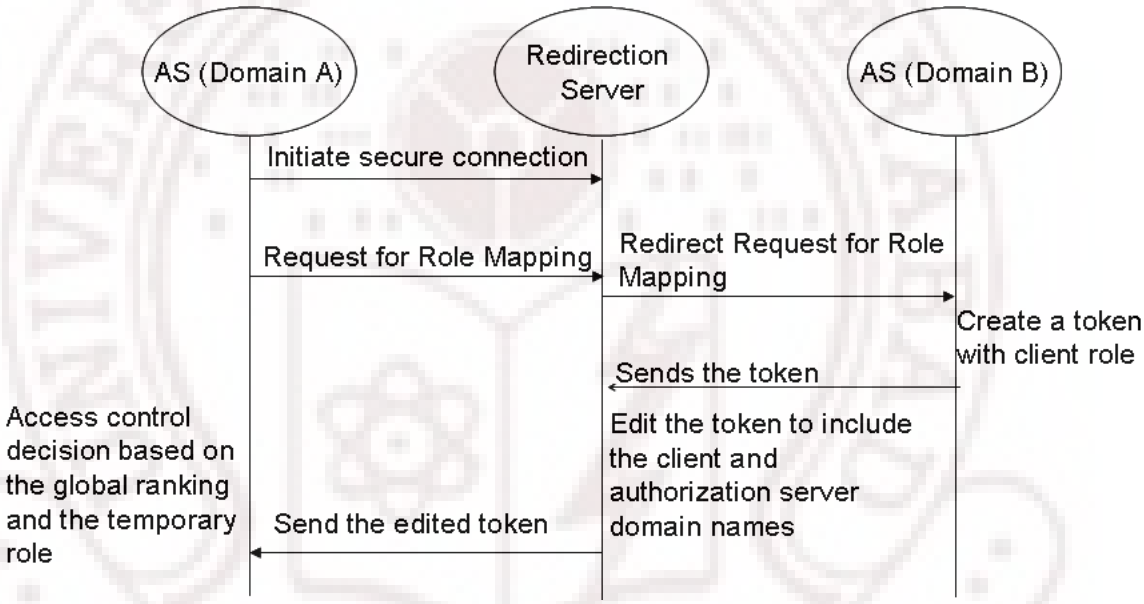


Figure 23: Sequence Diagram for Cross Domain Role Mapping

Figure 24 depicts delegation across different domains of a grid computing environment.

Algorithm 7 shows the steps through which a user is delegated across another domain.

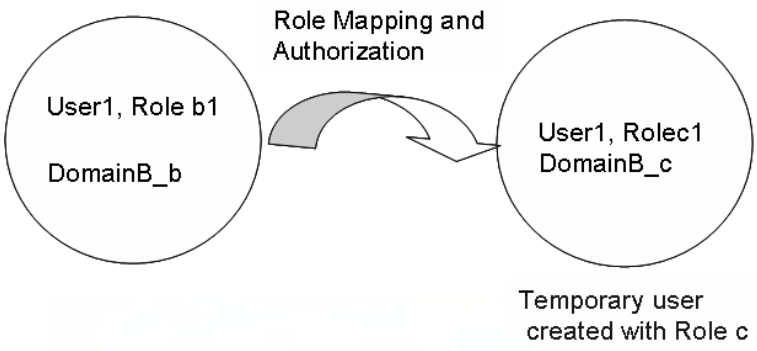


Figure 24: Cross-Domain Delegation

Algorithm 7 AcceptDelegate

Input : User Token
Output : Delegation deny/Grant
begin
Authorize User
Create user with username and other credentials
Add user to the Temporary user set
retrieve the role whose global rating is closest-
to that of user and assign user with that role
end

4.4.3.1 *Revocation of Rights-Cross Domain Environment*

The rights once granted, should not be held for long by the delegate. Such a scenario may invite misuse. It is necessary to have a robust access control mechanism which encompasses a strong revocation model to revoke the rights passed on by a user. Earlier, in the chapter, we presented an algorithm for revocation of intra-domain or single domain delegation. Here, we incorporate revocation in the cross-domain environment. In a scenario where a user, *user1* wants to revoke all his assigned roles, then the user invokes the *doRevoke* method of the domain *B_b*. It recursively calls the same method from the domains where it has further delegated the user's rights. It then stops all the jobs which are running with that user's rights, then removes the user from the temporary users' set and then returns the token. Thus the revocation of user's rights happens recursively in the above mentioned fashion as shown in Figure 25.

The algorithm for the recursive revocation procedure is as shown in the Algorithm 8.

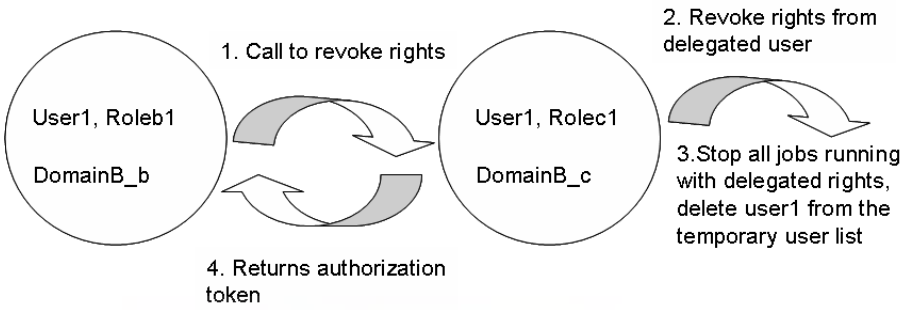


Figure 25: Cross-Domain Revocation of Delegation

Algorithm 8 doRevoke

```
begin
    call doRevoke(delegated users)
    Stop all jobs concerning the user who holds -
        the right to be revoked
    delete user from temporary user set
    send back the token
end
```

Revocation of delegated rights in a multi-domain environment can be done through any of the following means.

- Revocation-Grant Dependent: In this model, the delegate can revoke any right that he has delegated at any point of time. This means that he can issue revocation at any time to the Authorization server of the domain to which the right has been delegated
- Revocation- Time Dependent: In this model the delegated right expires after a stipulated amount of time, when it is revoked automatically in the delegated domain according to the revoke algorithm as given above
- Revocation- Administrator Dependent: In this model, the administrator of the delegate domain or the domain in which the role has been delegated can revoke the role. We define a delegation path as "a set of domains to which the role has been delegated from one domain to another in order". This path would contain the global rank of the role delegated in that particular domain with the names of the domain.

4.5 Chapter Summary

The main contribution of this chapter is the proposed Role-Based Grid Delegation Model (RB-GDM). The model is based on the standard RBAC and RBDM models and provides a flexible indirect authorization mechanism for grids. This framework is an innovative approach towards delegation in grid environment, as it uses a *role* as the criterion instead of a proxy credential. By categorizing the grid users into various roles and delegating access rights through roles, we also achieve granular access control. We provided three different inter-connection frameworks to depict the delegation scenarios in the intra-domain and inter-domain grid setup. Our model supports both peer to peer as well as hierarchical role relationships. Delegation based on role hierarchies ensures restricted access rights to the resources. The proposed model also provides mechanisms for revocation of delegated rights. RB-GDM is a common framework to support single domain as well as multi-domain grid delegation and revocation.

CHAPTER V

TRUST BASED DYNAMIC DELEGATION MODEL

5.1 Introduction

In the previous chapter we presented a framework for role-based delegation. This chapter presents a dynamic delegation model based on fuzzy trust values. As mentioned in Chapter IV, delegation is a way of indirect authorization where by a user is granted access to a resource through delegated rights rather than directly acquired rights. Delegation is an effective access control mechanism in dynamic situations. As the grid resources are distributed in space and time, direct authorization methods are not always sufficient. In this chapter, we introduce the concept of *trust* in grid systems. Trust relationships are central to the implementation of security in the dynamic and ever evolving communities like grids.

The rest of the chapter is organized as follows. In section 2, we give a brief introduction to Fuzzy Inference Systems (FIS), the computing technique which we have used. In the next two sections, we discuss about the need for dynamic grid delegation followed by the challenges of modelling and implementing them in grid computing systems. Section 5 presents the Fuzzy Trust and Delegation Model in detail. This consists of two stages namely computation of trust levels and computation of fuzzy delegation levels. And finally, we summarize the contributions of this chapter.

5.2 Fuzzy Inferencing and Trust

Fuzzy Inference Systems use fuzzy logic to control the input-process-output cycle. Introduced by Lotfi Zadeh [199] as a way to generate decisions involving imprecise information and approximate reasoning, fuzzy logic is an ideal methodology for representing complex systems involving ambiguity and vagueness [198]. Trust

is often expressed in linguistic terms rather than numeric values. Fuzzy Inferencing is a suitable technique to quantify trust among domains which may be a resource site (provider) or application site (consumer). The utility of fuzzy inference systems has not been explored fully in the area of security control. Shanshan Song and Kai Hwang had proposed fuzzy trust integration for security enforcement in grids [152]. We use fuzzy inferencing to quantify and analyze trust values so that we can derive access control decisions for the virtual grid domain.

The trust relationships among grid sites are hard to assess due to the uncertainties and impreciseness involved in the selection and measurement of input variables for evaluation process. These uncertainties may also appear in the form of security holes such as operating system blind spots, privacy traps, hardware deficiencies in resource/application sites and software bugs. These security holes adversely affect the trust index of sites.

The fuzzification interface maps the crisp input values into linguistic values in relevant fuzzy sets. The fuzzy inference engine receives fuzzified inputs and interacts with fuzzy rule base to determine the outcome. The fuzzy rule base is a collection of *if-then* rules. The de-fuzzification interface aggregates fuzzy inference results and produces a crisp output value. Various de-fuzzification methods such as centroid, bisector and middle-of -max exist.

Fuzzy inferencing helps in quantifying the uncertainty and imprecision involved in assessing trust relationships. Fuzzy inference approach helps in developing different inference rules. The linguistic terms used in fuzzy approach are close to reality.

Fuzzy logic extends the Boolean logic and introduces a set of membership functions that maps between linguistic elements to numerical values in the context of fuzzy sets. A fuzzy membership function indicates the degree to which an element belongs to a fuzzy set.

The data types used in fuzzy systems are linguistic in nature (linguistic variables and linguistic terms). A linguistic variable has a range of values and at least one linguistic term. It also has a name attribute. A linguistic term has a membership function, a name and an identification number. A membership function can

either be pre-defined or user-defined. Pre-defined membership functions are incorporated in the fuzzy inference system whereas user defined membership functions are created by the user by putting together a collection of operations like arithmetic, logarithmic, trigonometric etc.

The four basic components of a fuzzy system are fuzzification interface, fuzzy inference engine, fuzzy rule base and a de-fuzzification interface. Figure 26 shows these four components and the process flow through them.

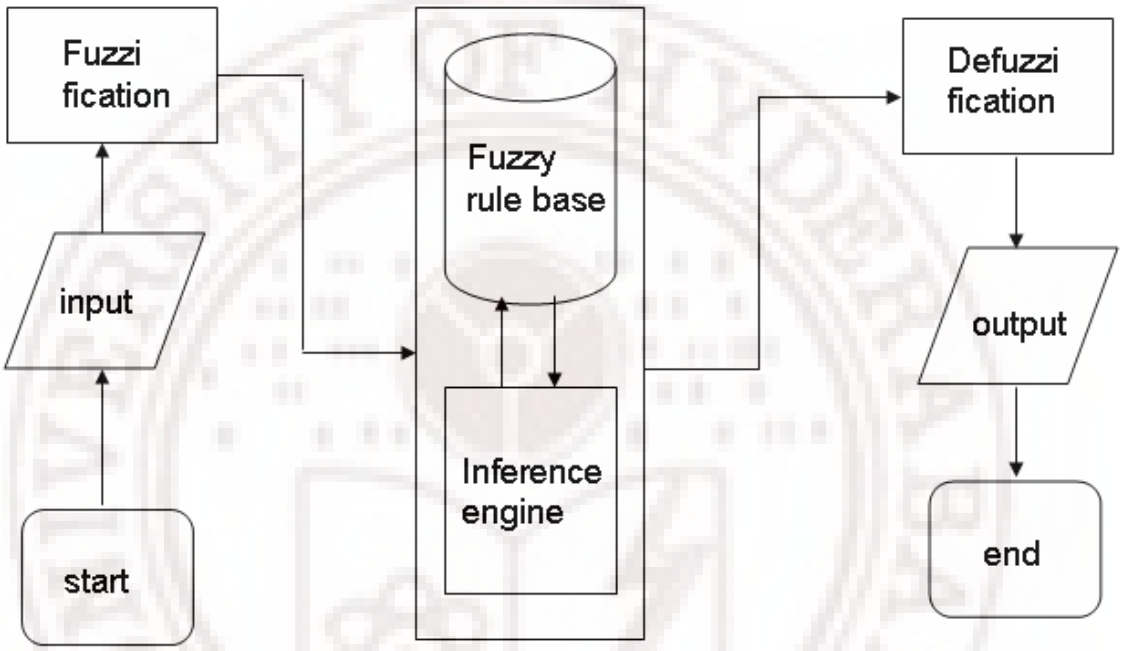


Figure 26: Fuzzy Inference System

Hilary H. Hosmer et.al [137] suggested the use of fuzzy logic approach to secure computing systems. Since many elements in security systems like in grids are imprecise [152], we need to deal with non-traditional world through fuzzy logic in order to get a more realistic and trustworthy result.

5.3 Need for Dynamic Grid Delegation

Access control policies in general follow the concept of direct authorization, where the subjects can directly authorize the requested access. In a dynamic grid computing environment, subjects and end-users may not have direct access or authorization to execution of a task or to a required resource. Direct authorization may

not be possible since the computational resources may belong to different organizations, which do not necessarily maintain the user accounts and permissions of other domains. Indirect authorization thus becomes important to tackle such situations.

Access to resources in grids can be broadly categorized as full, nil or partial implying a degree of fuzziness. Conventional security models are rigid and the tasks that need to be performed by an active grid entity, require more flexible form of access. A fuzzy logic based scheme provides more flexibility to resource access and access control in grids. The delegated rights have to be granted in a dynamic environment where the entities of a grid do not have prior information about each other. We propose a new scheme for this, which uses a two-stage fuzzy inference process based on trust relationships among the grid entities.

In the delegation approach, any entity on the grid - usually an end-user, grants rights to another entity on the grid - usually a process, to perform actions on the user's behalf. Delegation in general assumes two forms: static and dynamic. Static delegation may be used when the methods/objects to be accessed are known in advance and the entities which would provide the resources are also predefined, so that delegation can be limited to just the required actions from the predefined entities. Dynamic delegation refers to the delegation of rights to perform certain actions or access some objects that are not known at the time a task is initiated.

If a site has to submit a task or subtasks to other grid entities on behalf of an end-user, some form of delegation must be applied so that the entity running the task can verify the authenticity of the request. Various methods can be adopted to pass delegation credentials to an entity [121]. One method is creating digital signatures of task descriptions using private keys. This method is suitable for static delegation.

To implement dynamic delegation, the entities involved must establish trust relationships among themselves [188]. The most flexible form of grid delegation is *impersonation* where the *delegator* grants full access rights to the *delegate* to act on the delegator's behalf. Impersonation can have adverse impact on security as there is a chance of fully delegated rights getting compromised. Therefore delegation has to be restricted or limited which means that authorization through delegation

should not be full or unlimited.

Let us consider an example to illustrate delegation in grids. A server at a site S_i of domain D_1 wants to access a file at another site S_j of domain D_2 on behalf of an end-user. In order that S_j trusts S_i, S_i sends a message with a credential or token to S_j stating that the former (S_i) has been delegated by the end user. In dynamic grid environments, one faces two problems. First is that the end-user, after submitting the job, may get disconnected from the destination site S_j. Second issue is that the access requirements are not known beforehand (it is known dynamically [50]). Therefore the alternative is to grant access rights through dynamic delegation.

For this, site S_j has to trust S_i which means that a minimal trust relationship between the sites is mandatory. Figure 27 shows how two sites S_i and S_j can establish a trust relationship dynamically in the immediate absence of the end-user and how the trust can be used as the delegation credential or token from end-user to site S_i.

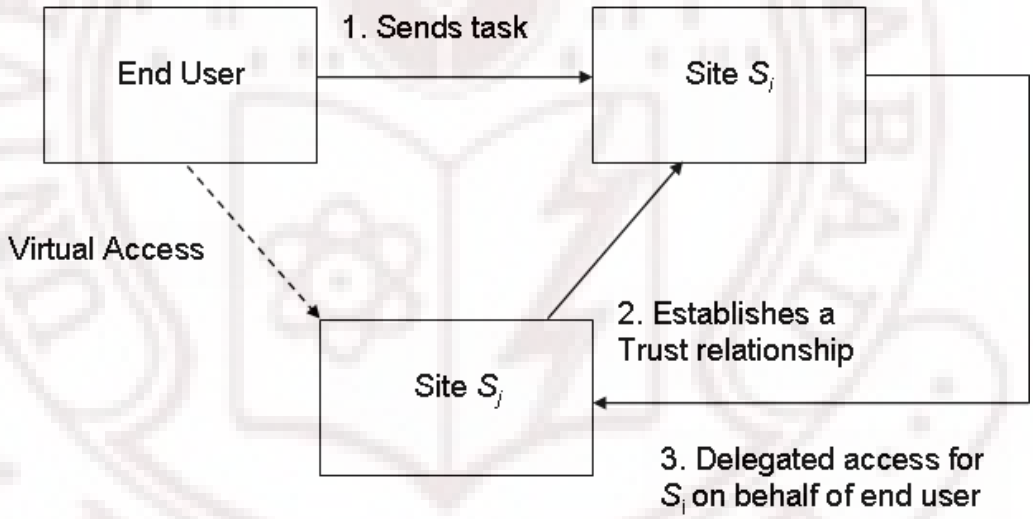


Figure 27: Dynamic Delegation Through Trust

We can secure the distributed grid applications based on a fuzzy trust model by building mutual trust relationships among multiple grid resources sites. Fuzzy trust integration approach can reinforce the security mechanisms. Fuzzy inference is capable of quantifying imprecise data or uncertainty in measuring the security index of resource sites. Different inference rules can be developed for different grid

applications using the same fuzzy logic inference engine [193].

As far as access requirements in grids are concerned, dynamic delegation is necessary to support the runtime selection of a site or a machine on which a task like global load balancing, brokering, or conditional computation (within a work flow setup) can be run. Delegated authorization can also be used to control actions performed by background processes. we analyze the existing delegation categories and schemes in grids and propose a new fuzzy trust and delegation model (FTDM) for grid access control.

5.4 Challenges of Dynamic Delegation in Grids

The two grid delegation models in use are GSI model and UNICORE model [197]. An entity claiming delegated access rights has to present some form of proof called a proxy credential or certificate within the Grid Security Infrastructure(GSI). That entity is then granted all the rights of the end-user. Proxy certificate holders can issue proxy certificates to other entities also, which implies that resources used during execution can also be dynamic. Proxies have a disadvantage that the end-user may be away or disconnected, so issuing the proxy credentials dynamically may not be possible. Also the private key bound to a proxy credential cannot be stored in an encrypted form as it has to be read by the processes to which the rights have been delegated without contact with the end-user. Another issue related to the GSI proxy credential is that it has relatively short life time. The UNICORE security model [138] though works well for static delegation, does not have the grid functions to support dynamic delegation.

The drawbacks of UNICORE and GSI delegation models reflect the need to have explicit trust delegation to enable delegated access in grids. One possible alternative is to have a trusted third party. But, a third party authorization can be done only on top of a user-role platform, which is not well defined so far in grids. This leads to the conclusion that delegation credentials have to be generated through some dynamic mechanism. Most of today's grid systems are moving towards web services, which implies that dynamic delegation mechanisms need to co-exist with the web-services infrastructures. These causes provided motivation for the development of a Fuzzy logic based dynamic grid delegation model.

5.5 *Fuzzy Trust and Delegation Model (FTDM)*

The previous section emphasizes the necessity for dynamic grid delegation and also the challenges in implementing it. Proving the trustworthiness of all resource sites is a prerequisite for secure grid computing. Trust in general can be classified as identity trust and behavior trust [158]. Identity trust deals with authenticity of a grid entity and deciding the rights of that entity. Authentication is done using cryptographic approaches like encryption and digital signatures. Behavior trust focuses more on the trustworthiness of an entity than its identity. Most of the authorization policies in grid computing systems are formulated based on the behavioral trust patterns of the grid entity. In order to define the authorization policy between two entities in a grid system, the first step is to establish a trust relationship between them.

We consider a set of factors which determines trust. The values of these parameters are obtained from the logs of previous actions of the delegator site and also from the data supplied by the delegate site. The imprecision and vagueness arising in some inputs such as defense capability and behavioral patterns is the main reason for using a fuzzy approach. The concept of trust seems subjective and of a categorical nature and hence needs to be quantified. So fuzzy approach helps to be a handy tool. Also delegation can be full (full access rights), nil (no rights-negative permissions) or partial (only a subset of the access rights are transferred) which means that decision making needs to be done in an imprecise setup.

Fuzzy logic approach is the most suitable approach to analyze such vague data. In fuzzy approach, the imprecise input is fuzzified (converting crisp data to fuzzy sets), then it is passed through a fuzzy inference process to validate the rules [179]. In the next step, this fuzzy output is defuzzified (converting fuzzy data to crisp data) to get the final crisp output. Deriving fuzzy values from the inputs is a standard process. The Fuzzy Trust and Delegation Model (FTDM) which we propose here [166], has two stages. We use the first stage fuzzy inference process to quantify the trust value and the second stage fuzzy process to decide on the possibility of delegated access from the category of delegation and trust level.

5.5.1 Stage 1 of FTDM : Computation of Fuzzy Trust Levels

The trust, which a particular site S_j places on another site S_i, depends on the previous actions of S_i. We have considered three inputs namely task success ratio(α), defense capability(β) and behavioral patterns(γ) to compute the trust level(τ). These inputs are arrived upon from the history of previous actions of the sites. Task success ratio is the ratio of the number of tasks previously executed by site S_i to the total number of tasks submitted by S_j to it and it is a numeric value. If no security breaches have been reported earlier, we assume that the site's defense capability is good. Thus the defense capability of a site can be worked out based upon the existing security mechanisms and the history of earlier access.

Typical site defensive measures are installation of encrypted tunnels, smart packet filters, stateful firewalls, traffic monitors and intrusion detection systems. If no security breaches have been reported at S_i earlier, we can assume that it has been secured properly. As trust is a measure of the defense capability and security assurance of the site, we attach more importance to these two inputs.

Behavioral patterns, the third input can be obtained if we collect data about the security intentions of the concerned site in the past, like intrusion attempts. Behavioral patterns when quantified, can work as an indicator to the future actions of the delegate site after the delegation is granted.

The last two inputs are imprecise and vague in nature. Since two out of the three inputs are imprecise and the output trust level is a non-numeric value, we use fuzzy logic approach to build the delegation model.

The three input membership functions we considered here are low, medium and high although they can have additional levels. The notations for the three membership functions for the inputs are α_l, α_m, α_h for task success ratio, β_l, β_m, β_h for defense capability and γ_l, γ_m, γ_h for behavioral patterns. (On a scale of 0-100, the support of fuzzy input levels are as follows: low is in the range 0-40, medium is in the range 38-70 and high is in the range 68-100).

Let η_{vl}, η_l, η_m, η_h, η_{vh} represent the five output trust levels *very low*, *low*, *medium*, *high* and *very high* respectively(on a 0-100 scale, the support levels are *very low* is

0-20, *low* means 18-40, *medium* 38-60, *high* means 58-80 and *very high* indicates 78-100). We consider more number of output levels for greater accuracy in trust computation. Table 9 shows the details of the fuzzy variables, their levels and ranges and also the fuzzy membership functions and their ranges. Figure 28 shows the

Table 9: Fuzzy Variables, Membership Functions and their Ranges

Levels of α, β, γ	Fuzzy Membership Function	Membership Range
l	α_l	$[{\alpha_l}^{min}, {\alpha_l}^{max}]$
m	α_m	$[{\alpha_m}^{min}, {\alpha_m}^{max}]$
h	α_h	$[{\alpha_h}^{min}, {\alpha_h}^{max}]$
l	β_l	$[{\beta_l}^{min}, {\beta_l}^{max}]$
m	β_m	$[{\beta_m}^{min}, {\beta_m}^{max}]$
h	β_h	$[{\beta_h}^{min}, {\beta_h}^{max}]$
l	γ_l	$[{\gamma_l}^{min}, {\gamma_l}^{max}]$
m	γ_m	$[{\gamma_m}^{min}, {\gamma_m}^{max}]$
h	γ_h	$[{\gamma_h}^{min}, {\gamma_h}^{max}]$
vl	η_{vl}	$[{\eta_{vl}}^{min}, {\eta_{vl}}^{max}]$
l	η_l	$[{\eta_l}^{min}, {\eta_l}^{max}]$
m	η_m	$[{\eta_m}^{min}, {\eta_m}^{max}]$
h	η_h	$[{\eta_h}^{min}, {\eta_h}^{max}]$
vh	η_{vh}	$[{\eta_{vh}}^{min}, {\eta_{vh}}^{max}]$

rationale behind the first stage fuzzy inference process. The transformation is of the form $\tau_1 : (\alpha, \beta, \gamma) \rightarrow \eta$. It shows the three inputs and the output trust level. The computed trust value may fall in any one of the five levels. Table 10 shows the fuzzy inference rules for first stage reduced from the most probable mappings. We use the composition operator AND to relate the inputs and the implication operator IMPLY to get the corresponding outputs. For the first stage fuzzy process we use three inputs, each having three levels.

Table 10: Fuzzy Inference Rules for Trust Computation

Inputs	Output (Trust Value)
(α_l, β_l,γ_l)	η_{vl}
(α_l, β_l,γ_h)	η_l
(α_h, β_l,γ_m)	η_l
(α_m, β_m,γ_m)	η_m
(α_m, β_m,γ_l)	η_m
(α_h, β_m,γ_l)	η_m
(α_l, β_h,γ_m)	η_h
(α_h, β_h,γ_l)	η_h
(α_h, β_h,γ_h)	η_{vh}

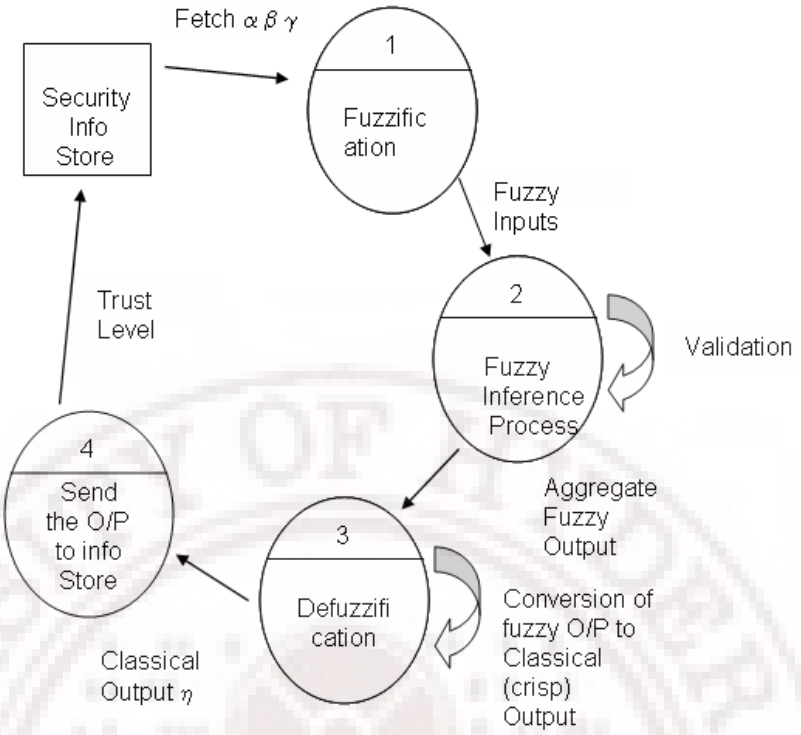


Figure 28: Stage 1 of Fuzzy Process

For the five ranges of the output Trust Level, we use 5 different possibilities which means that we have totally 27 x 5 possibilities. Out of the 135 input-output mappings, we remove the redundant rules and discard the infeasible ones which results in 27 probable mappings. We arrive at 9 rules as the most probable ones. Figure 29 illustrates how one combination of inputs in a certain range generates the output trust level.

For the example values, we have considered a case where all the inputs are in the range *high* and the fuzzy inferencing generates the output trust level which is in the range *very high*. This is an instance to show how the rules are fired and how the output is generated. These values correspond to rule no.9 i.e., AND(α_h, β_h,γ_h). The input values are mapped to the output space through a collection of fuzzy sets.

5.5.2 Stage 2 of FTDM : Computation of Fuzzy Delegation Levels

When an end-user asks an entity to perform some operation on its behalf, the user has to grant delegation to that entity. Conventionally, unlimited delegation is

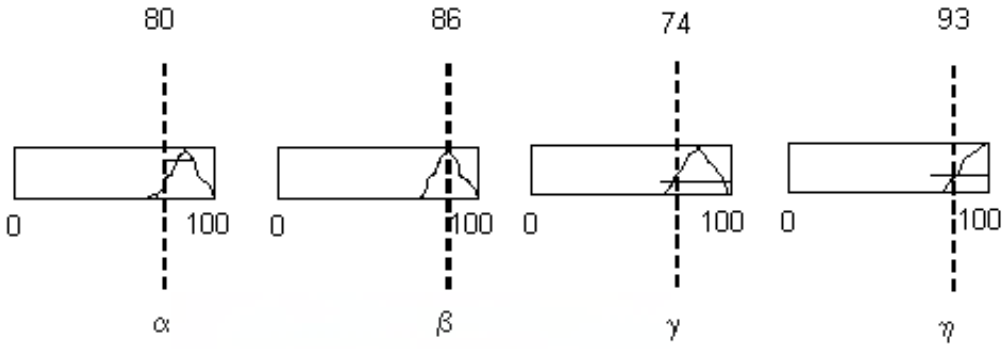


Figure 29: Generation of Trust Level

granted. Most of today's grid systems grant unrestricted delegation as it is difficult to design, implement and validate restricted delegation. Unlimited delegation can lead to security abuse. Also delegating few rights only may prevent task completion. Therefore there is the need to have a restricted and realizable grid delegation [79]. Realizable delegation can be defined as the ability of a user or principal to define and implement a policy by which it can grant a limited set of privileges to another user or principal. Restricted delegation can reduce grid security vulnerabilities to a great extent.

We use the following forms of restricted delegation:

- The delegated access rights may be used to invoke a method(m) or perform a task on behalf of the delegator
- Sometimes the delegated right may be used to access an object(o) or resource in grid

The delegated rights may be either unbounded(∞) or it may be bounded for a certain period of time(t). In all these cases the delegator must pass on some credentials to the delegate as a proof of the access rights.

5.5.2.1 Delegated Access for Method Invocation($D_{m,\infty}$) - unbounded

In this type of delegated access, a delegator grants access rights to a delegate only to invoke a method on the delegator's behalf based on how much the former trusts the latter for an unbounded period of time. A method is nothing but a function call to perform a task. This means that only a method can be called and access to

a resource is possible in this category only if that resource is the target of such a call. For method restricted delegation, the delegator lists the methods which can be called by the delegate and passes on the credentials to the delegate. This type of delegated access can be achieved by explicit method enumeration, compiler automation or by method abstractions, provided the delegated access is static. In case of dynamic delegation, the delegator needs to generate the delegation credentials based upon the trust value of the delegate.

5.5.2.2 Delegated Access for Method Invocation($D_{m,t}$) - bounded

For this type of delegated access, a delegator grants access rights to a delegate, only to invoke a method on the delegator's behalf for certain period of time. The only difference here is that the method invocation is limited by time bounds.

5.5.2.3 Delegated Access for Objects($D_{o,\infty}$) - unbounded

This category is very much similar to method restricted delegation. An object is nothing but a resource like computing power, memory or information. If the object access is not associated with method invocation which means that object or resource access is done through ways other than method invocation. This category specifies object delegation for an unbounded time. Since we arrive at the delegated access through trust values, we can have a set of trusted objects and indirectly achieve method delegation by suggesting that if the objects are trusted, then there is no need for credentials for method invocation on these objects. Another category of trusted objects for which, delegation credentials are not required are system objects(objects representing hosts, schedulers, storage units etc).

5.5.2.4 Delegated Access for Objects($D_{o,t}$) - bounded

Time dependent delegation to access objects is required in grids to prevent replay attacks. When delegated object access is restricted based on time, a pre-determined amount of time is used as the time-out for the delegation credential. The disadvantage of time-restricted delegation is that we cannot predict how long an operation or task execution may last, which means that the operation may have to continue even after the delegation credentials have expired.

The following format represents the information for restricted delegation credentials for method invocation, object access, time restricted access or for all categories.

```
Delegator Name:
Delegatee Name:
Category of Delegation (Method/Object):
Class of Delegated Access Type:
(Bounded/Unbounded)
```

Given the trust level of a grid entity and the category of delegation like method restricted(unbounded or bounded) or object-restricted(unbounded or bounded), it can be decided whether to pass on the delegation credentials to that site or not using a second-stage fuzzy process. Figure 30 gives the schematic view of the second stage fuzzy inference process we have envisaged for this task. The transformation is of the form τ_2 : (η, C)$\rightarrow$ D where C is the category of delegation and D,the delegated rights.

We identify the different contexts in which access rights are delegated. Among the four categories of restricted delegations, we associate a higher trust requirement for object restricted delegation (unbounded) as most of the grid resources come under this category. The next category is object restricted (bounded) delegation followed by method delegation (unbounded) and method delegation (bounded). Therefore, in the second stage of the fuzzy inference process, the two inputs are trust level(with five levels very low, low, medium, high and very high) and category of delegation with four possible values:object restricted delegation (unbounded and bounded) and method restricted delegation (unbounded and bounded).

Thus the outcome of the second stage is a decision on the delegated access, which could be $D_{m,t}$, $D_{m,\infty}$, $D_{o,t}$ or $D_{o,\infty}$. Table 3 shows the most probable rule set for the second stage. Figure 31 shows how a set of rules influence the output trust level. The two fuzzy inputs trust level and category of delegation are taken on a 0-100 scale. The variation of the output delegated access is also shown from 0-100.

For example, for a very high trust level (83) and object-restricted delegation category(unbounded) 78.6, the delegation access is 90.8 which is an aggregation of the output generated from the nine rules. As the value 90.8 falls in the range $\eta =$

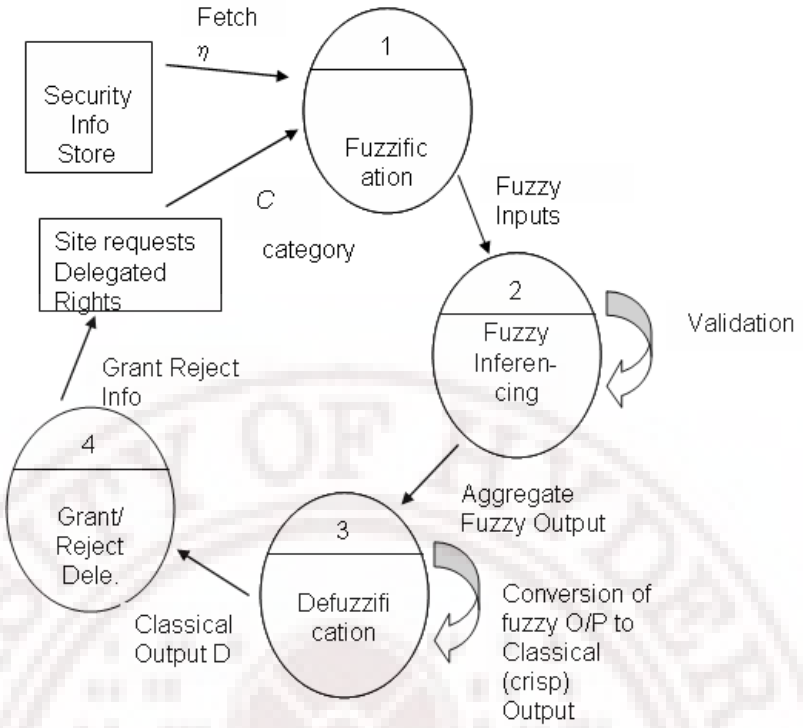


Figure 30: Stage 2 of Fuzzy Process

η_{vh} it means that the entity or site can be delegated for object access for unbounded time.

Table 11 represents the set of rules used in the second stage of the fuzzy inference process.

Table 11: Rule Set for Stage 2 Fuzzy Inference Process

Rule No.	Rule
1	if $\eta = \eta_l$ and $C = mt$ then $D = D_{m,t}$
2	if $\eta = \eta_\infty$ and $C = m\infty$ then $D = D_{m,\infty}$
3	if $\eta = \eta_h$ and $C = ot$ then $D = D_{o,t}$
4	if $\eta = \eta_{vh}$ and $C = o\infty$ then $D = D_{o,\infty}$
5	if $\eta = \eta_l$ and $C = o\infty$ then $D \neq D_{o,\infty}$
6	if $\eta = \eta_\infty$ and $C = o\infty$ then $D \neq D_{o,\infty}$
7	if $\eta = \eta_h$ and $C = o\infty$ then $D \neq D_{o,\infty}$
8	if $\eta = \eta_\infty$ and $C = mt$ then $D \neq D_{o,t}$
9	if $\eta = \eta_{vh}$ and $C = m\infty$ then $D \neq D_{m,\infty}$

In a similar way, we can compute the trust level dynamically for other categories of restricted delegation also, then relate the computed value with the type or category of delegation and arrive at a decision as to whether the delegator can

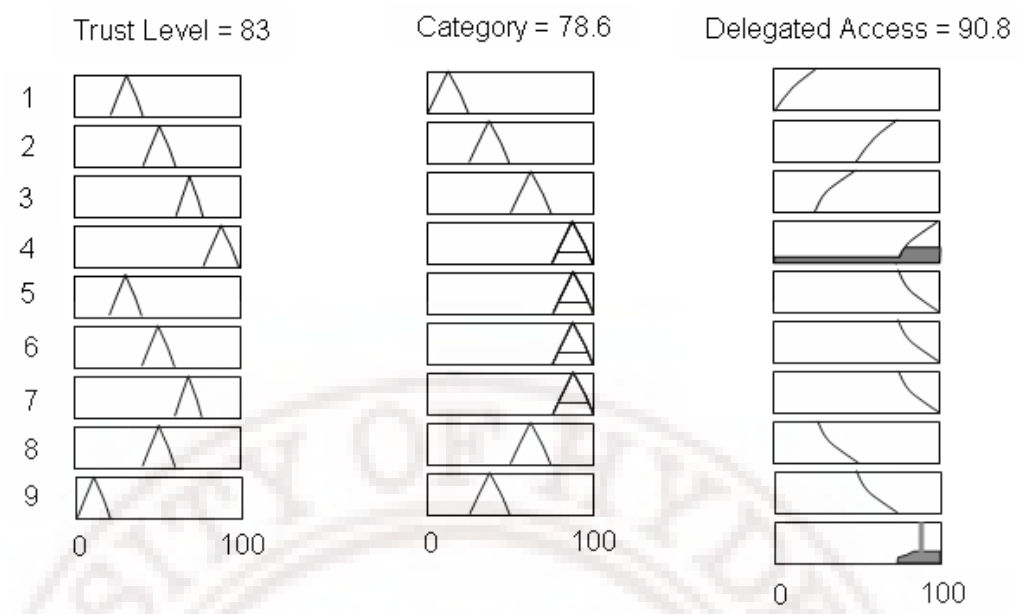


Figure 31: Illustration of Stage 2 Fuzzy Process

grant the access rights (bounded or unbounded) to the delegate. Periodic update of trust values can enhance grid security and can speed up the process of dynamic delegation.

5.6 Chapter Summary

In this chapter, we introduced the use of fuzzy logic to develop mechanisms based on trust relationships between domains, to achieve dynamic authorization of resources. Though fuzzy control has been used in many systems, the use of fuzzy logic to represent trust values is very limited and hence ours is a novel idea. The fuzzy trust and delegation model (FTDM) which we proposed, uses a fuzzy logic based scheme to determine whether dynamic delegation is possible in a grid and if so, what type of delegated access is permissible for the given trust level of a site. We considered the trust level between two sites as the deciding factor because it can be measured dynamically, reflecting the behavioral patterns and intentions of the site requesting delegation. The two-stage fuzzy inference process provides a approach solving access control issues in grids. The FIS rule design mechanism helps in formulating an efficient dynamic delegation policy in grids.

In summary, *trust relationships* form the basis for determining the degree of delegation between two sites of a distributed environment like the grids and also that this trust relationship can be extended to multiple sites making multi stage dynamic delegation feasible for automated performance improvement in grids.

CHAPTER VI

FINE GRAINED ACCESS CONTROL FRAMEWORK

6.1 Introduction

In Chapter V, we presented a trust-based mechanism for decisions based on the degree of dynamic delegated rights. In this chapter, we use the fuzzy-based control mechanism to arrive at a flexible, trust-aware fine-grained access framework for grid access control from the context of access permissions and resources allocation. The access policy (grant/deny or partial grant), is governed by the trust relationships among the grid domains. The quantification of trustworthiness of a resource site is done using fuzzy inference systems.

The rest of the chapter is organized as follows. In section 2, we discuss the fine-grained authorization requirements of a grid computing environment. Section 3, discusses the advantages of trust-aware grid sites and quantification of trust using Fuzzy Inference Process. We consider various categories of trust such as direct trust, asymmetric trust and indirect trust. In section 4, we propose a fine-grained authorization mechanism which can map the computed trust values to the corresponding access control decisions to ensure secured fine-grained resource access. We consider varying degrees of access like full access, partial access (fine grained) or nil access. And finally we summarize the work done as part of this chapter.

6.2 Fine-grained Authorization in Grids

The heterogeneous nature and independent administration of geographically dispersed resources in grid, demand the need for an access control mechanism built on fine-grained policies. We need to investigate the problem of fine-grained access control in the context of resource allocation in grids as it is the first and key step in developing access control methods specifically tailored for grid systems. Fine-grained access control system has proved to be effective for different applications

[66]. Elisa Bertino et al., [42] proposed a security component as part of a meta-scheduler service that can find the list of nodes where a user is authorized to run his/her jobs. The security component was designed in an effort to reduce the number of rules that need to be evaluated for each user request. Similarly, grid users get a higher flexibility in choosing the resources in which their jobs must execute.

A grid domain has its own local administrative policy and follows a global access policy for inter domain access. Implementation of a grid-wide access control policy, which can comply with the local policy is a real challenge. A grid would be valuable if it can contribute resources to the ultimate user and/or application across multiple domains in a secure way. Computational grid infrastructure softwares like the Globus Toolkit enable a user or application to identify and use the available resources irrespective of location and ownership. User programs/applications may contain malicious codes, which may act as a threat to the grid resources. On the other hand, grid resources, which are not trustworthy, may pose security problems to the user applications. Therefore security assurance must precede resource access. Almost all grid applications like scientific explorations, health care, government and business services have security and privacy requirements. Virtual organizations formulate and enforce policies for its members' use of community resources, not only in terms of *who can use what resource, but how it can be used*.

The resource management is a real issue in a virtual environment like a grid where heterogeneous resources are geographically distributed and are held by individual domains having their own access policies. In such an environment, both the resource and the application need to be security-aware. If a resource provider site and a resource consumer site can establish a trust relationship among themselves, then mapping a resource request as per the trust value is possible. To analyze trust and take access control decisions based on it, we need to quantify trust. Several issues need to be considered in trust computation. We consider various categories of trust to arrive at a fine-grained access control model.

The fine-grained access control and authorization requirements in grids [149], [186] can be listed as follows:

- Combining policies from different sources. While outsourcing a portion of

the policy administration to the virtual organization (VO), the policy enforcement mechanism on the resource needs to be able to combine policies from the resource owner, the local policy and the grid-wide policy.

- Fine-grain control of access rights. For the *VO* to express the differences between how its user groups are allowed to use resources, the VO needs to be able to express policies regarding a variety of aspects of resource usage, not just grant/deny access.
- *VO*-wide management of jobs and resource allocations [187]. Normally, in a virtual organization, the jobs themselves are treated as resources. This poses a challenge with the jobs being dynamic and static methods of policy management not effective. Users may also initiate jobs that are independent of the *VO* - e.g. a user may have allocations on a resource other than through the *VO* and jobs invoked under this alternate allocation should not be subject to *VO* policy.
- Fine-grained and dynamic enforcement mechanisms. To implement policies, we need supportive enforcement mechanisms. Most resources today are capable of policy enforcement at the user level. This implies that all jobs run by the same user may be governed by the same policy though finer policies for different resources is ideal. Also, the enforcement mechanisms are typically statically configured through file permissions, job quota and the like. Therefore, we require an enforcement mechanism which can handle dynamic, fine-grain policies.
- Fine-grainedness based on roles (like initiator of jobs, manager of jobs etc)
- Fine-grained access for users with resource specific account [147]
- Fine-grained access based on trust relationships

6.3 Trust Aware Grid Sites

The fundamental concept of inter domain trust relationships is that trust works as an assurance that an entity will act in exactly the way we expect. Trust, when concerned with the authentication of grid entities is called identity trust. The Grid

Security Infrastructure (GSI) of Globus provides identity trust-related services. Behavior trust of an entity implies its trustworthiness. Shared grid resources, if infected with malicious programs, may damage the user application running on the grid platform. Trust assessment of a site involves the measurement of dependability, security, reliability and performance [125].

Let us consider a grid (G) as a virtual organization of multiple domains each having its own administrative and local access policies. Let D_j be any domain in a virtual organization, which may have n number of members. Let S_{Dj} be the site representing the j^{th} domain. Then we can represent a grid as G ={D_j, where, j=1 to n }. The trust relationship between two sites S_{D1} and S_{D2} is a linear and directional value t_{12}. We consider the following categories of trust relationships namely direct trust, asymmetric trust and transitivity trust to arrive at access control decisions.

6.3.1 Direct Trust

If a domain D_i trusts another domain D_j based on their direct interactions, then it is called direct trust t_{ij} or η. It also reflects the reputation of D_j as a trusted entity. The reputation of an entity is an expectation of its behavior based on other entities' observations or information about the entity's past behavior at a given time.

6.3.2 Asymmetric Trust

Sites in a grid environment may be either resource sites or user/application sites. A bi-directional trust relationship is to be established between these sites to ensure secured access to resources and to provide security assurance to the applications. If t_{ij} is the trust from a resource site (D_i) to an application site (D_j) and t_{ji} the trust from D_j to D_i. t_{ij} is not necessarily equal to t_{ji}. The unequal trust between two sites is termed asymmetric trust. To create two-way trust awareness, we compute the asymmetric trust.

6.3.3 Transitivity Trust

Transitivity property of trust determines whether trust can be extended outside the two domains between which it was established. Transitivity property can be used to extend trust relationships with other domains and non-transitivity trust to deny

trust relationships with other domains [20]. Transitive trust is an automatic trust association between parent and child domains in a forest of domains. Transitive trust relationships flow upward through a domain tree as it is formed, creating transitive trusts between all domains in the domain tree. Here, we extend this idea for establishing trust between two domains.

Execution of jobs in a grid environment generally requires a site to interact with several trusted intermediaries before actually accessing the resource site. Let us consider sites D_i, D_j and D_k in the neighborhood. If D_i trusts D_j and D_j trusts D_k, then by using property of transitivity we can say that D_i trusts D_k. The site D_i need not directly compute the trust value on D_k. So transitivity trust is an indirect trust relationship. Transitivity trust reduces the overhead involved in computing trust for another time. It is also useful in the absence of data on past interactions. D_j acts as a trusted intermediary between D_i and D_k. Figure 32 depicts various trust paths for the above categories.

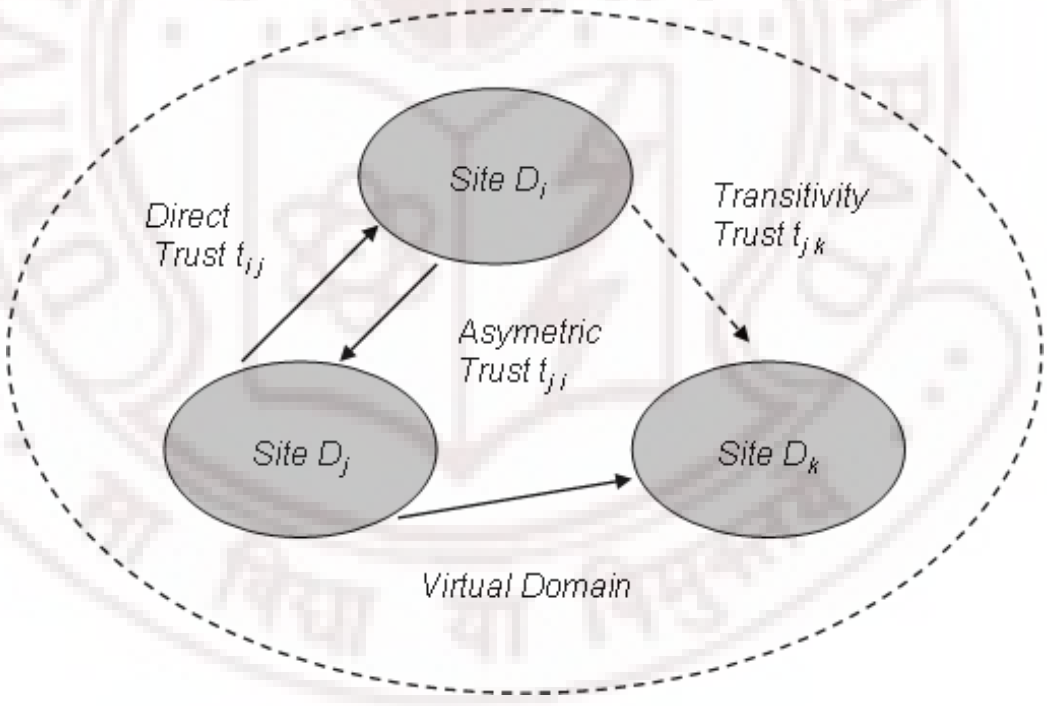


Figure 32: Trust Paths across a Virtual Organization

6.3.4 Quantification of Trust using FIS

In the previous chapter, we have seen that the linguistic evaluation of trust value is not enough to assess the security preparedness of a grid site. We need a mechanism by which we can translate this value to its corresponding numeric form. A fuzzy inference system allows us to provide input values as vague/imprecise linguistic terms, which, after fuzzification process, is converted to the corresponding fuzzy inputs.

6.3.4.1 Quantification of Direct Trust

The basic inputs which we use for trust quantification of a site are, job success ratio (α), defense mechanisms (β) and well-behavedness (γ). To fine-tune these inputs, we use one more parameter called timestamp **t**, which indicates the time of computation of trust. This timestamp **t** shows whether trust values have been updated or not.

Trust computation procedure between sites *i* and *j* starts with measurement of the trust inputs. Job success ratio α is the ratio of the number of jobs successfully executed(*m*) by *j* to the total number of jobs submitted(n) at *j* by *i* in the recent past. i.e., $a = m/n$. a of a site is an indication of its performance. This ratio is a numeric value. The Grid Resource Allocation Manager (GRAM) at site *i* maintains relevant job data as well as status of the jobs.

The second trust variable is defense mechanisms β. β represents the security preparedness of a grid site. Preferred defense mechanisms at a site are distributed firewalls, encrypted tunnels, packet filters, intrusion detection systems and traffic monitors. If site S_{Di} has to compute trust on S_{Dj}, it sends a query to site S_{Dj} asking for the availability of these mechanisms.

Let $DM^i\ (S_{Dj})$ represent the defense mechanism at a site (S_{Dj}), where i = 1 to l, and the value of $DM^i\ (S_{Dj})$ is a Boolean variable. We attach certain weights w_i to each of them, based on their effectiveness. Then β can be represented as,

$\beta = \sum w_i\ DM^i\ (S_{Dj})$, where i=1 to l and we consider l = 5, which can be increased.

The weight w_i is such that l.b $\leq w_i \leq$ u.b, where we assume that l.b= 0.3 and u.b=0.9, (the lower bound and upper bound values) depending on the defense

mechanisms. Incorporating additional attributes like *security levels* and *efficiency* adds to the accuracy of measurement of β. Various defense mechanisms possible in the grid environment include distributed firewalls, encrypted tunnels, packet filters and intrusion detection mechanisms.

The third trust input well-behavedness γ of a site during past interactions is a pointer to its access intensions. A well-behaved site never intends for unauthorized tampering of resources through its applications/task requests. Since γ is a natural term, we quantify it and analyze it through fuzzy approach. It can be measured with the help of data from intrusion detection systems at site *i*. The data of reported intrusion attempts at *i* from *j* can be analyzed using data mining techniques. These patterns indicate the well-behavedness of *j* with respect to *i*.

We consider three varying degrees of inputs namely low, medium and high (i.e. from 0 to 1 on a numeric scale). Similarly, for the output variable *direct trust*, we consider three levels low, medium and high (from 0 to 1). An inference rule consists of at least one antecedent and one consequent. For example, an inference rule can be: *if direct trust is low then access-deny*. Here, *direct trust is low* is the antecedent and *access-deny* is the consequent. We associate a weight attribute with the rule. Each antecedent has one linguistic variable, one linguistic term, and one logical operator. Similarly, each consequent also has a linguistic variable, a linguistic term and a weight attribute. We have computed and quantified the trust values using a two-stage fuzzy inference process.

For the domains D_i and D_j, the trust value may increase or decrease over a period of time based on the interactions between the sites. This means that periodic update of trust is necessary to reflect the dynamic access environments. We update trust values based on the time stamp *t* attached to it. Thus direct trust γ is a function of *t* expressed as η_t = f(α_t, β_t, γ_t).

6.3.4.2 Quantification of Asymmetric Trust

We have seen earlier that the trust values t_{ij} and t_{ji} between two sites *i* and *j* need not be the same due to the differences in the performance of sites resulting in different job success ratio values, non-uniformity in the defense mechanisms set up at the two domains and different degrees of behavioral patterns or well-behavedness

from one site to another. Thus trust is a vector with a magnitude and direction.

6.3.4.3 Quantification of Transitive Trust

In the previous two categories of trust, the trust level that a resource/application site holds on another is based on the direct relationship/interactions between them. In wide area systems such as grids, a site needs to interact with more than one site to complete a job execution. Since quantification of trust between every pair of domains is not practical, we can consider transitive trust. Scalable trust allows certain level of trust to travel to large number of nodes (or entities) and still be able to maintain maximum level of trust during its travel in a specific time frame. This propagation of trust is called transitivity trust.

A virtual organization comprises multiple grid domains. It follows a certification path. A "root" CA (RCA) issues certificates to its subordinate CAs and these CAs issue certificates to their next level subordinate CAs and so on. The "root" CA is regarded as the trustworthiest in the hierarchical certification path. Every subordinate must know the root CA's public key. Any subordinate's certificate may be verified by following the certification path till the verifier reaches the root CA (e.g. a common trusted CA or the CA who issued the subordinate's certificate). Figure 33 shows the scalability of trust in a virtual organization.

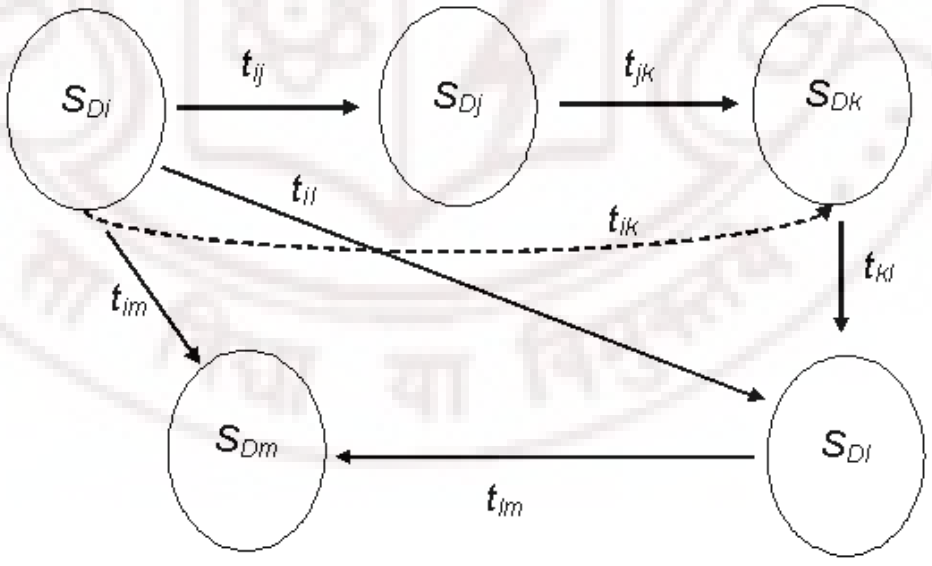


Figure 33: Scalability of Trust in a Virtual Organization

Considering the domains D_i, D_j and D_k in a virtual domain with RCA as the root CA and having their own local CAs (subordinate CAs), then "$D_j < D_i >, D_k < D_j >$ " means D_i signed the trust credential of D_j, and D_j signed the certificate of D_k. Similar to the X.509 Public Key Infrastructure (PKI), the level of trust provided by certificates "$D_j < D_i >, D_k < D_j >$ " is the same as "$D_k < D_i >$ ". In other words, possession of "$D_j < D_i >, D_k < D_j >$ " provides the same capability as "$D_k < D_i >$ " (i.e.$D_j < D_i >$, $D_k < D_j >= D_k < D_i >$). Figure 34 shows the transitivity trust path in a V.O. We quantify transitivity trust using the relation,

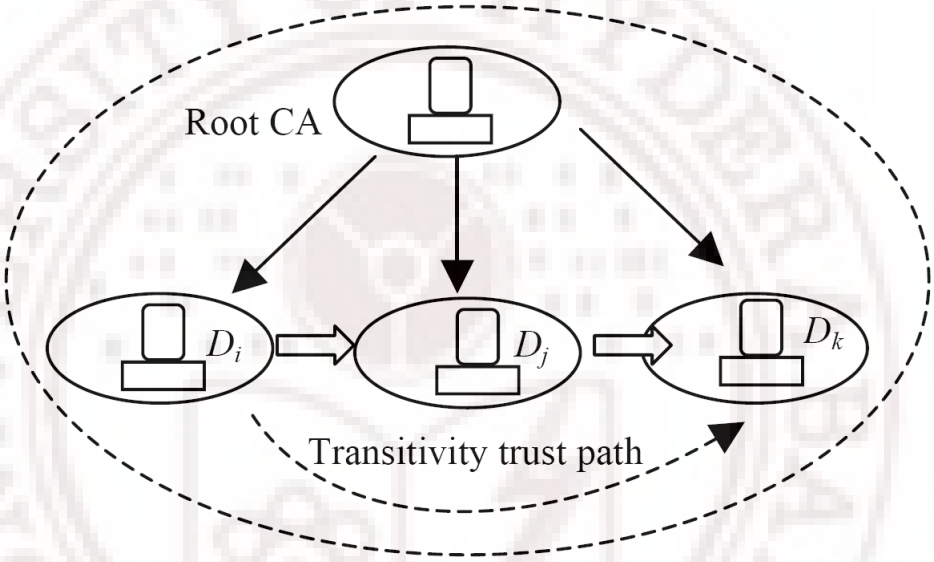


Figure 34: Transitivity of Trust

Trust (*i*, *j*)∧ Trust (*j*,*k*)⇒ Trust (*i*,*k*) provided the sites *i*, *j*, *k* have a common root CA. Here, ∧ implies the composite relation AND between the trust values Trust (*i*, *j*) and Trust (*j*,*k*). Site S_{Di} establishes a trust relationship with S_{Dk} through a trusted intermediary. As direct trust relationship is more authentic, we consider a degrade value for transitivity trust. Transitivity trust evaluation t_{ik} requires the trust values t_{ij} and t_{jk} as its inputs.

Figure 35 shows the computation of transitivity trust from the trust values trust (*i*, *j*) and trust (*j*,*k*). As transitivity trust is an indirect security assessment, we apply a reduction factor w on trust(*i*,k). i.e., t_{ik} = trust (*i*,*k*) *(1- w)

In Figure 35, t_{ij} is 0.794 and t_{jk} is 0.832. The composite operation ∧ provides

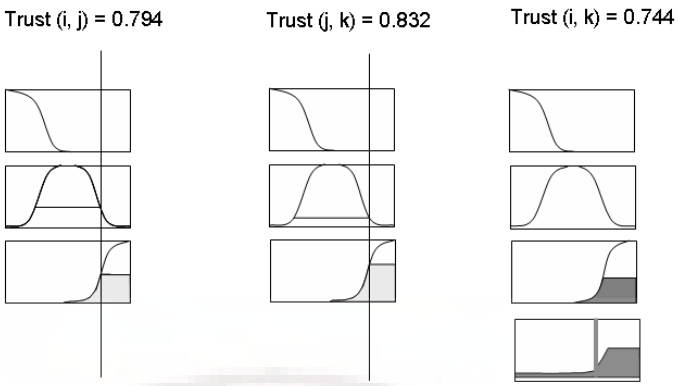


Figure 35: Quantifying Transitive Trust

the value of t_{ik} as 0.744. We assume w based on the application and the security level of the resource accessed. For security critical applications/resources a large w such as 0.9 can be adopted. For stable and relatively low security-sensitive applications/resources, a small w such as 0.3 is adopted. In general cases one can set w in the range (0.7-0.8). There is another way in which we could arrive at the transitivity trust value. Instead of site D_i trusting D_k through only Dj, we can have D_{j1},D_{j2}....D_{jp}. There can be various possibilities like t_{ik}=*Min*$\{t_{ij} \wedge t_{ik}\}$, where j=j_1,j_2.....j_p or we can also have *Max*, *Avg* relations depending on the trust requirement. This is depicted in Figure 36.

Transitivity trust values help in trust integration and propagation across multiple grid domains. Domain D_j acts as the trusted third party or intermediary between D_i and D_k.

6.4 Fine-Grained Access Control Framework using Trust Values

Fine-grained access control of grid resources is important for grid security. We need to express and enforce fine-grain policies on the usage of resources. These can no longer be expressed by simple access control as we have to specify exactly what fractions or configurations of resource may be used by a given entity. The targets of a fine-grained policy can be resource allocation and usage, job management, application services etc. The fine-grained access specifications for a resource among the *VO* participants is provided by the *VO*. The *VO* has two primary classifications of its members: one group has the role of developing, installing and

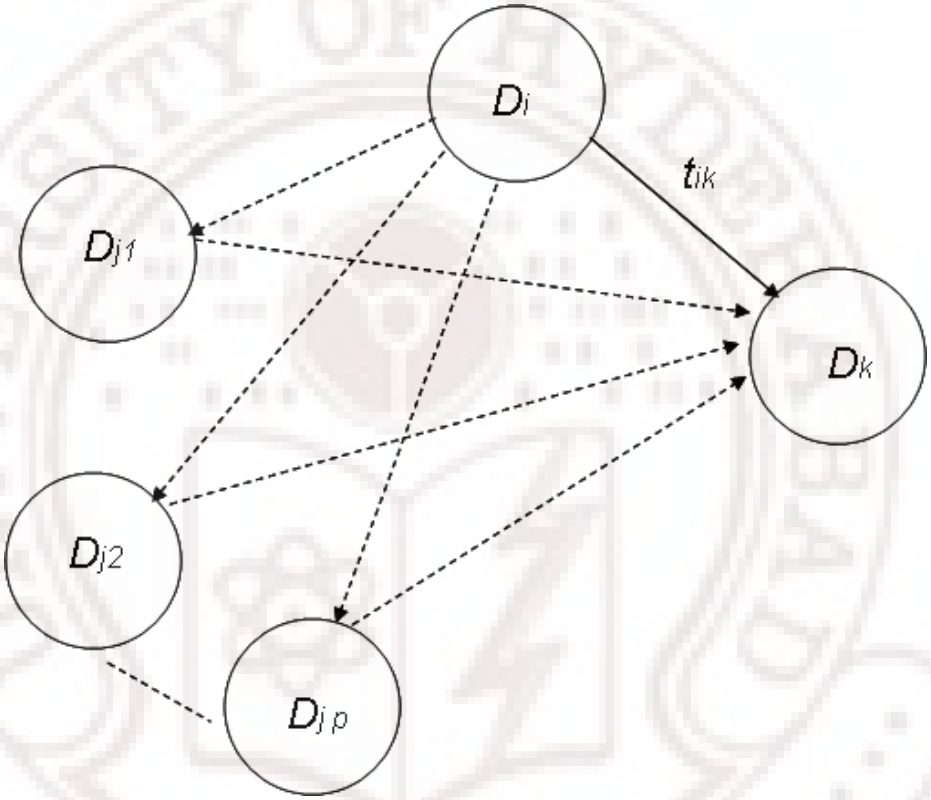


Figure 36: Constructing Transitive Trust

debugging the application services used by the virtual organization to perform their scientific computations and another group which does analysis using the application services. The first group may need a large degree of freedom for the applications they need to run (e.g. compilers, debuggers, the applications themselves) in order to debug and deploy the *VO* application services, but would only be consuming small amounts of traditional computing resources (e.g. CPU, disk and bandwidth) in doing so.

The second group may need the ability to consume large amounts of resources in order to perform scientific computations, but should only be doing so using application services approved by the *VO*. Also to specify policies such that certain users may use more/ less resources than others and that certain applications may consume more or less resources than others, we need a fine grained grid access control mechanism.

In traditional access control mechanisms, we have only the permit - deny kind of access decisions. In trust aware grid systems, we can incorporate certain degree of fine grained access based on trust values, the access control being full, nil or partial. Though the Grid Security Infrastructure of GT4 security framework supports various authorization schemes such as *self authorization*, *gridmap file*, *identity-based authorization*, *host-based authorization* etc, there is no provision made for fine-grained authorization in any of these schemes. The client identity is obtained from the credential used by the application to contact the service. The resource owner identity is obtained from the credentials configured for the resource, service, or container, depending on availability, in that order of precedence.

6.4.1 Fine-grained Authorization Engine

The *VO* may wish to specify finer-grain policies that certain users may use more or less resources than others. These policies may be dynamic and change over time. In addition to the policy on the resource utilization, the *VO* may wish to manage jobs running on *VO* resources. For example, users often have long-running computational jobs using *VO* resources and the *VO* often has short-notice high-priority jobs that require immediate access to resources. This requires suspending existing jobs to free up resources; something that normally only the user that submitted the

job has the right to do. Since going through the user who submitted the original job may not always be an option, the *VO* may give a group of its members the ability to manage any job using *VO* resources so that they can instantiate high-priority jobs on short notice.

We propose a fine grained access control system which consists of an authorization filter in the form of fine-grained access control engine. Based on the quantified trust values, this engine intercepts every access request and evaluates it against the restrictions to service/resource accessibility. Thus the resource request may be rejected; be allowed as it is or may be filtered and executed in a modified form. The filtering of a request may involve elimination of some of its attributes that the client is not allowed to specify or additionally needs to specify. Figure 37 shows the proposed fine-grained authorization mechanism.

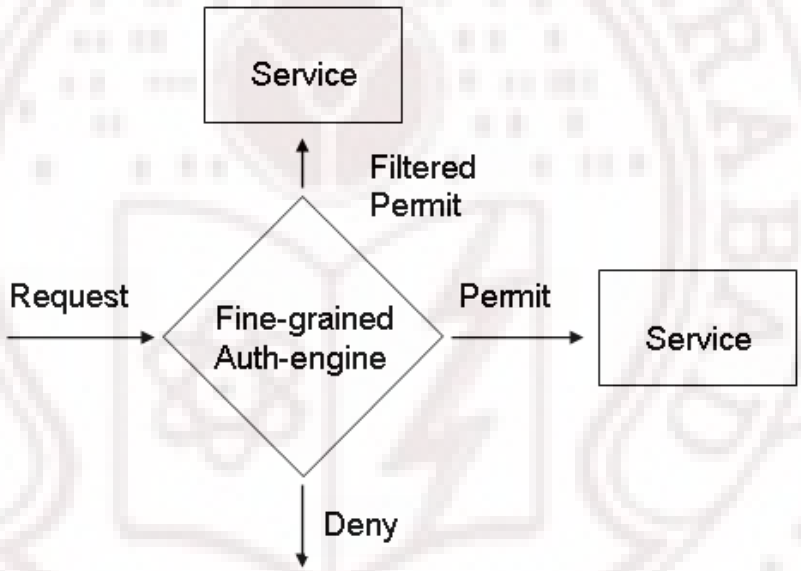


Figure 37: Fine-Grained Authorization

Any of the above authorization schemes can use the quantified trust value (for different contexts) to generate access decisions. We integrate our fine-grained authorization engine the XACML framework. The implementation of a similar integration (of Role-Based Model with XACML framework) has been explained in Chapter VII of this thesis. At the Policy Decision Point or PDP [34], the authorization or access decisions are implemented. Rules form the most important subcomponent of any access policy. Rules are essentially conditions that evaluate to

permit, deny or partially permit access.

6.4.2 Rule - Creation for Access Decisions

A fine-grained policy framework is formulated in three stages namely, creation of the rule target, definition of the effect and creation of the condition. A rule target is an entity on which the rule works. In a grid environment, a rule target is a grid resource/application. Once the rule target is created, it needs to be authenticated. In a trust aware grid environment, the trust levels are used as credentials. The effect of the rule implies the type of access decision namely permit, partial permit or deny. If the associated condition of a rule is true, then the effect of the rule is positive.

6.4.3 Trust Credential

The quantified trust value is used as an attribute for constructing the trust credential. The general format of the trust credential as per the XML syntax is as given below. Every trust update is reflected by incrementing the version number of the trust credential. The trust credential contains the policy decision which governs the fine-grained access of grid resources. The elements of this format include the granter name, the grantee, the trust type, trust level (trust value), validity period,the access type and the resource being accessed.

<**trustCredential**>
<version>1.0</version>
<granter>granterDN</granter>
<grantee>granteeDN</grantee>
<trustType>type</trustType>
<trustLevel>level</trustLevel>
<validity>
<notBefore>time</notBefore>
<notAfter>time</notAfter>
</validity>
<accessType>accessval</accessType>
<resourceName>name</resourceName>

</**trustCredential**>

6.4.4 Mapping Trust Values to Access Decisions

The trust value can be at any of the levels namely *low*, *medium* or *high*. We suggest an access control model, which maps the trust value to its corresponding access decision based on the trust values. The following are some of the sample if-then-elseif-else rules used for fuzzy inferencing. The different categories of trust are taken care of by weighted trust quantification and the trust input variables in each case.

- *if direct trust is* ***low*** *then denyAccess elseif direct trust is* ***medium*** *then partialAccess else grantAccess*
- *if asymmetric trust is* ***low*** *then denyAccess elseif asymmetric trust is* ***medium*** *then partialAccess else grantAccess*
- *if transitivity trust is* ***low*** *then denyAccess elseif transitive trust is* ***medium*** *then partialAccess else grantAccess*

Figure 38 shows the initiation process of resource access after mapping of trust values with the access decisions.

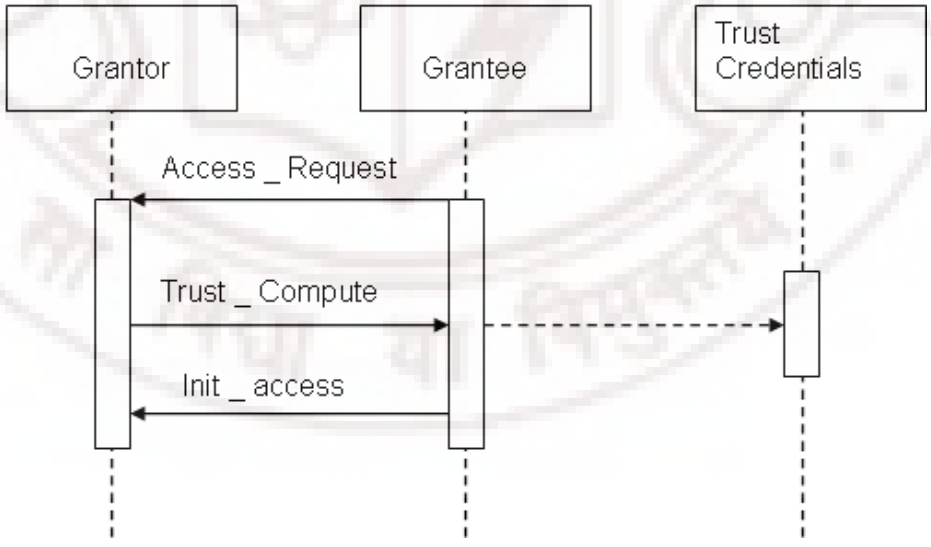


Figure 38: Initiation of Access

The overall trust aware access architecture for a virtual organization is as in Figure 39. We present the framework for any two grid domains, which could be extended to any number of domains n. Domains i and j each contain a trust management module (TMM). This module establishes trust relationships between the two domains by interacting on a set of trust inputs and then quantifies trust. After this process, the trust value is passed on to the PDP (Policy Decision Point) module, where the trust values are mapped with access privileges. PEP or the Policy Enforcement Point, acts as the access control enforcement mechanism, which implements the access decision taken at the PDP. Once the access is granted, the resource can be accessed. The working of PDP and PEP have been discussed in Chapter III, section3.3.

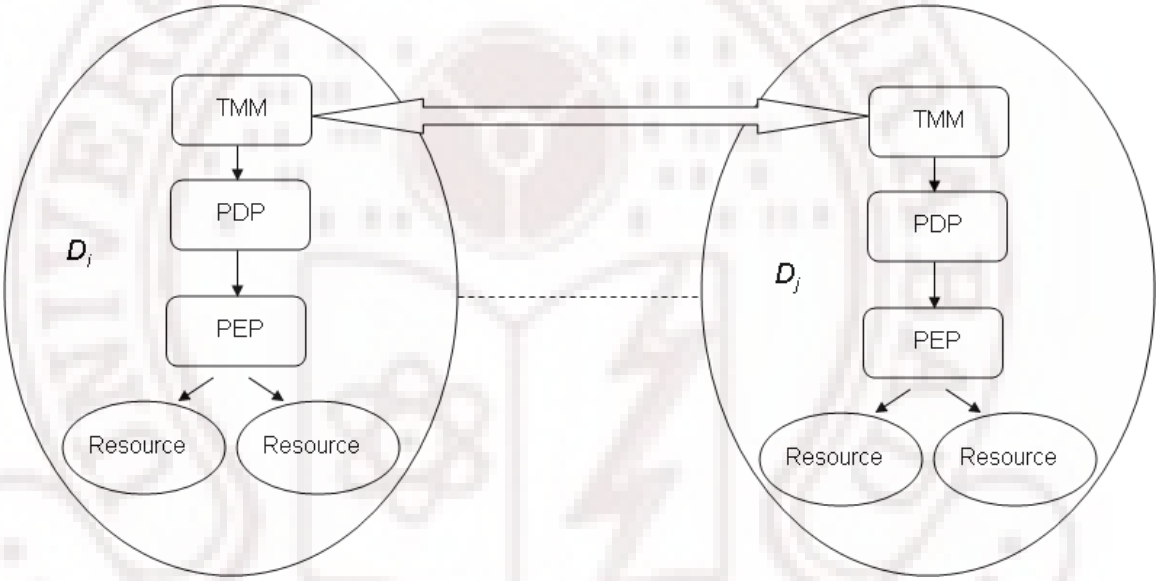


Figure 39: Trust Aware Access Control

6.5 Chapter Summary

The main contribution of this chapter is the proposed trust-aware fine-grained access control framework for grid environment. We quantify various categories of trust namely direct trust, transitive trust and asymmetric trust using fuzzy inference process and map these values to access control decisions. This framework can be incorporated with the legacy access control mechanisms. Use of transitivity trust reduces the trust computation time and is best suited for scalable virtual

environments like grids. A trust aware grid environment can deal with dynamic access requests to resources more effectively. Ascertaining that an entity with a particular identity or set of attributes has the permission to perform a particular action on a particular resource is a key issue. Trust aware grid sites can secure their resources and applications in an efficient way.

CHAPTER VII

IMPLEMENTATION DETAILS OF THE PROPOSED MODELS AND AN ANALYTICAL CASE STUDY

7.1 Introduction

In the previous chapters, we presented various architectures and models for solving grid security and access control requirements such as direct authorization, delegation, dynamic access control and fine-grained access control. But enforcement of these models through suitable mechanisms is equally important. This chapter is a logical extension of the Chapters III and IV as we elaborate the implementation details of the proposed architectures. The first step in implementation is to prepare ground for a middleware platform [190], [77] on which the implementation can be done [122]. As grid is a heterogeneous environment, we need to have a middleware to project its federated nature. We explored the possibility of using established grid middlewares like the Globus, the Gridbus etc. We found that Globus is a middleware platform with support for security-related interfaces like authentication, authorization etc where as other middlewares lacked security interfaces. Our models being solutions for security issues in grids needed security-supportive interfaces for implementation. Accordingly, we opted for the latest Globus Toolkit 4.0 as the middleware platform [72], [135]. Next, we need to decide the prerequisites in terms of the software and the technologies. The operating system used is Redhat Linux 9.0 distribution.

This chapter is organized as follows. Section 2 contains the implementation details of various interfaces like the Admin GUI and Client GUI. We also provide a GUI by which the administrator can update or modify new XACML policies [145], [96]. Appendix A describes the installation of Globus Toolkit and shows the steps to write a grid service. Appendix B gives an overview of XACML [34]. In section 3, we show as to how we can implement the architectural frameworks for cross-domain (inter-domain) authorization and delegation. Section 4 describes

the LDAP architecture which we have proposed for the grid environment with its implementation. Appendix C gives a general description of LDAP. In section 5, we give a detailed case study conducted on the security aspects of Garuda Grid and suggest ways to incorporate better security mechanisms in Garuda.

7.2 Implementation of the Interfaces and Main Modules for Single-Domain Grid Enterprise

Globus is a middleware for grid environment. It enables us to write grid applications. Installation of Globus and how to write grid services [57] is explained in Appendix A.

Technologies used

- Java - for Interfaces design, grid service implementation
- mysql - Database for storing user-role, user-delegation information.
- LDAP - server to store details about additional attributes of the users and resources in the organization.
- XACML - to express Access Control Polices [176]

To demonstrate the authorization procedure, we have written a simple grid service called the Mathservice which includes the following operations:

1. add - for adding two numbers
2. sub - subtracts two numbers
3. mul - multiplication of two numbers
4. div - division of two numbers

The whole project is put in the /home/globus/project folder. The mathservice contains two modules:

- **MathClient.java**
- **MathService.java**

Both are placed in the /home/globus/project/mathservice/src/org/globus/mathservice/ folder. We then deployed this grid service in globus container using apache *ant*.
Then run *ant deploy* (run this command where your build.xml file lies)

7.2.1 Administrator and Client GUI

- **Service Provider:**
  The service provider deploys the service in the container and specifies the constraints to access the service based on roles. The constraints are specified in plain language

  1. student role allowed to access only addition and subtraction. Time-range to access mathservice is 9am to 5pm.
  2. reader role has access to addition, subtraction and multiplication. Time-range to access mathservice is 9am to 7pm.
  3. professor role has access to all operations of the mathservice. Time-range is 9am to 11pm.

- **Administrator:**
  The administrator has a separate interface to do operations like specifying the organizational hierarchy, user-role assignments, creating and modifying the access control policies etc. The administrator creates access control policies according to the service provider specifications using XACML [116]. The administrator interfaces with the LDAP server through the LDAP browser. Figure 40 shows the graphical user interface for the administrator to create or modify RBAC policies. Figure 41 shows the interface through which the administrator can create or modify access control policies in the XACML framework.

- **Client:**
  The Client GUI as shown provides support for the clients to get details of user-role relations, log information about the user's past delegations and works as an interface for the user to delegate his role to another user. Clients can also query ldap using the LDAP browser. The client GUI for interaction to administrator as well as for delegation are depicted in Figures 42 and 43 respectively.

The attributes we have considered in formulating the access control policies are

- Role of the user

Figure 40: Administrator GUI

- Requested service
- Operation with regard to that service and
- Time-range to access the service.

XACML decision engine operates with these attributes to make the decision. The custom authorization takes the final decision based on two things.

1. First it checks the decision from XACML decision engine
2. Then checks the system dynamic attributes (free memory, cpu load)

Figure 44 shows the screenshot of user-role assignment as done in Role Based Access Control.

7.3 Main Modules

1. **MainPDP.java:**
   The custom authorization module is plugged in the globus toolkit.

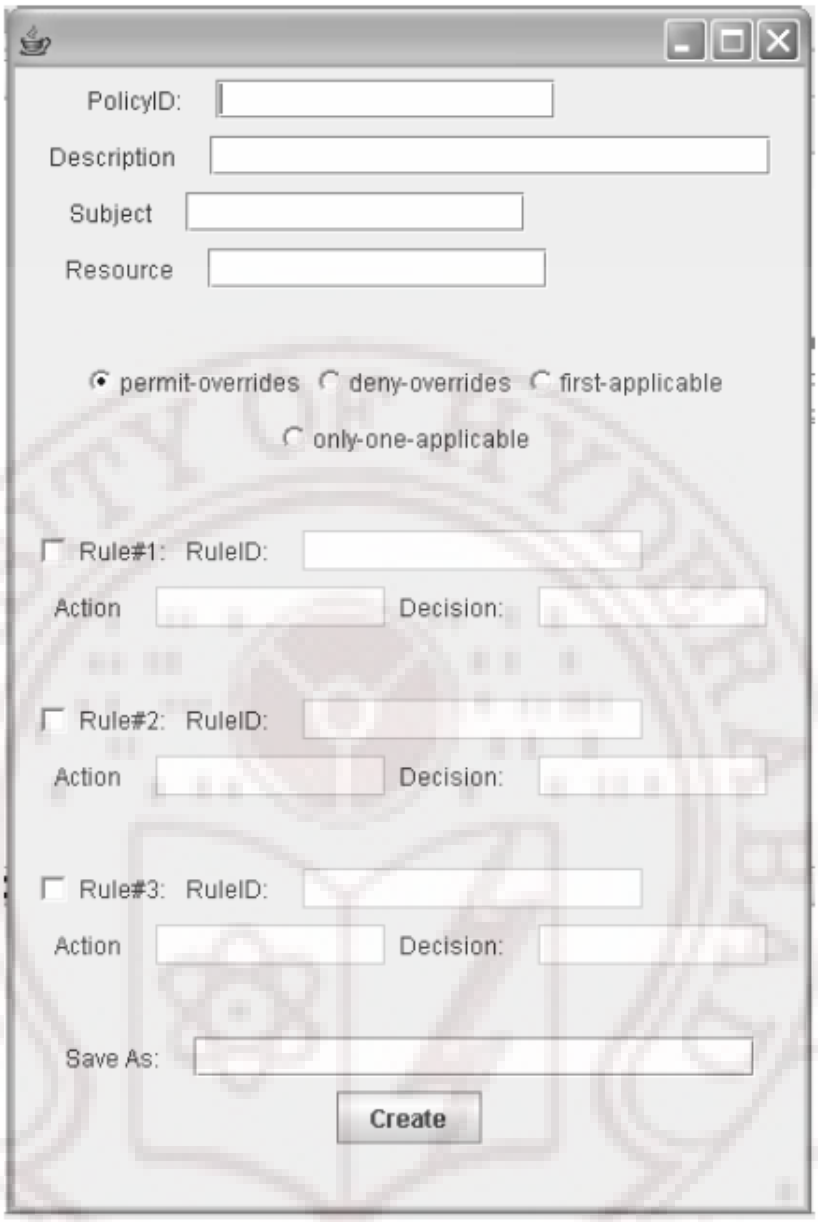


Figure 41: Administrator GUI for Creating New Policies in XACML

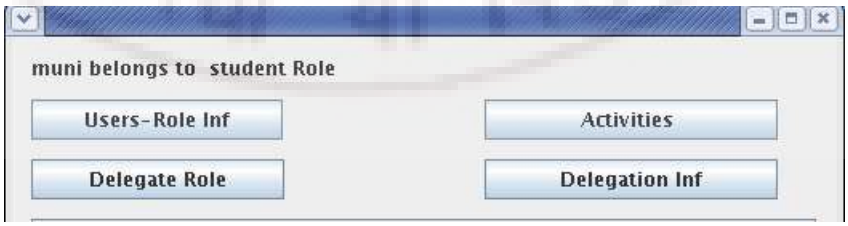


Figure 42: Client GUI

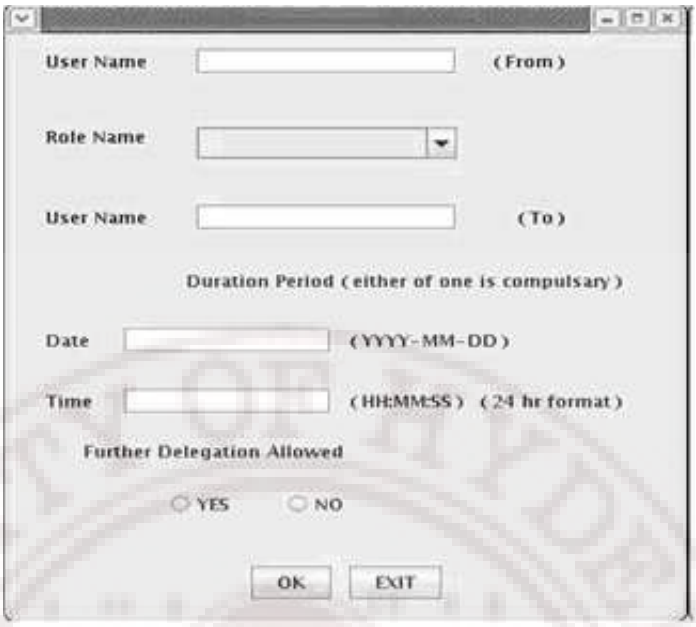


Figure 43: Client GUI for Delegation of Roles Based on Delegation Constraints

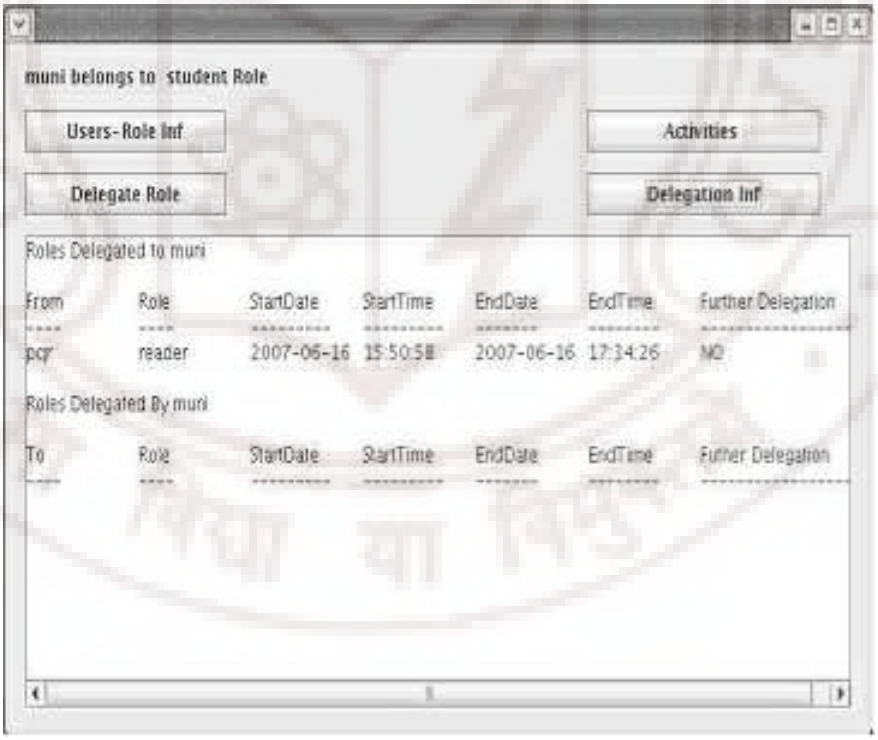


Figure 44: User-Role assignment

The custom authorization module implements the interface org.globus.wsrf.security.authorization.PDP;

This module can be found at /home/globus/project/serviceprovider/
input: user's details and request information.
Output:

- *True:* If the decision from the XACML decision engine is *permit* and the system dynamic attributes (free memory, cpu load) are below the specified limit.
- *False:* Decision from XACML decision engine is *deny*, or *permit* but the free memory and cpu load are above the specified limit.

```
//some of the imported packages
import org.globus.util.I18n;
import org.globus.wsrf.security.authorization.PDP;
import org.globus.wsrf.security.authorization.PDPConfig;
import org.w3c.dom.Node;
import org.globus.wsrf.security.authorization.PDPConstants;

//This method takes the final authorization decision
public boolean isPermitted(Subject peer, MessageContext
context, QName op) throws AuthorizationException {
  Parameters:
     peer - authenticated client subject with
            credentials and attributes
     context - holds properties of this XML
               message exchange
     operation - operation that the subject
                 wants to invoke
  Throws:
```

```
AuthorizationException
```

The MainPDP sends the details to the XACML decision engine which is running on a different system using socket function calls.

2. **PEP.java:**

The PEP receives the request, and tries to get the user's explicitly selected role or the original role.

```
//connection establishment to database
 try {
     Class.forName("com.mysql.jdbc.Driver").newInstance();
     connection = DriverManager.getConnection
     ("jdbc:mysql://localhost:3306/rbac?","root","");
 }

//get the original Role of the user
//if he not selected any role explicitly
 try {
     PreparedStatement statement1
           = connection.prepareStatement
     ("select * from usr_role where usr = ?;");
     statement1.setString(1,username);
     ResultSet rs = statement1.executeQuery();
     while(rs.next())
role = rs.getString("role");
 }

//get the Role if he explicitly
//selects one of his delegated roles
 try {
     PreparedStatement statement2
        = connection.prepareStatement
```

```
        ("select * from usr_selected where usr = ?;");
        statement2.setString(1,username);
        ResultSet rs = statement2.executeQuery();
        while(rs.next())
delrole = rs.getString("role");
 }
```

If the user selects his delegated role, the PEP tries to get other information like expiry date and time for that delegated role. Compare the expiry date and time with current date and time. If the role's time limit is not expired, the PEP formulates the RequestContext.

```
//formulating Request Context
RequestCtx request =
new RequestCtx(subjects, resourceAttrs, actionAttrs);
Parameters:
   Attributes of the subject, resource and Action
   subjetct has attributes like
   Role, name and Address etc.
```

If the access control policies are too complex and need more subject attributes than just the subject role, then the PEP contacts the LDAP server. It queries the server for more details about subject using the subject's identity and sends the RequestContext to the PDP.

```
//send the Request Context to PDP
//and get the Response Context
ResponseCtx response = pdp.evaluate(request);
```

3. **PDP.java** This is the core class for the XACML engine, providing the starting point for request evaluation. Before evaluating the RequestContext, we have to initialize the PDP with information like the location of the policies. The supported classes for PDP are

- ***PolicyFinder.java***

  This class is used by the PDP to find all policies used in evaluation. It finds a policy based on the request's context. This involve using the request data and matching this data with the Target element in the policy to decide whether the given policy applies or not.

```
//Tries to find policies applicable
//to the given RequestContext
 public PolicyFinderResult
   findPolicy(RequestCtx context)
```

  XACML policies contains rules, which may specify the conditions also (example: time-range, age > 20 etc). So, we need modules to support comparisons, string matching and boolean logic etc. Some of the classes are

- ***AddFunction.java*** This class implements all the add functions. It takes two or more operands of the appropriate type and returns the sum of the operands.
- ***ComparisionFunction.java*** This class implements all of the standard comparison functions like greater-than, less-than and greater-than-or-equal etc., on most of the data types.
- ***TargetMatch.java*** Represents the SubjectMatch, ResourceMatch, or AcitonMatch XML types in XACML, depending on the value of the type field. This is the part of the Target that actually evaluates whether the specified attribute value in the Target matches with the corresponding attribute value in the RequestContext.

```
 //Determines whether this TargetMatch
 // matches the input request
public MatchResult match(RequestCtx context)
   context - the representation of the request
   returns - the result of trying to match the
             TargetMatch and the request
```

- ***TimeRange.java*** The PDP upon receiving the request, tries to find the

appropriate timerange policy for that RequestContext. If the request to service comes within the specified range the PDP continues to identify the next applicable policies for that request otherwise it returns ResponseContext to the PEP with decision as *Deny*.

```
//Parse the timerange policy to check
//the time limits to access the resource
public boolean TimeRange(String attrs[])
```

- ***CombiningAlgorithm.java*** This class is the base type for all combining algorithms.

```
//Combines the inputs based on the
//context to produce some unified result.
public abstract Result combine(RequestCtx context,
                               List inputs)
  context - the representation of the request
  inputs - the things to combine (Policies or Rules)
 returns - a single unified result based on the
                              combining logic.
```

- ***DenyOverridesPolicyAlg.java*** This is the standard Deny Overrides policy combining algorithm. It allows a single evaluation of *Deny* to take precedence over any number of *permit*.

```
//Applies the combining rule to the set of
//policies based on the evaluation context.
public Result combine(EvaluationCtx context,
                                  List policies)
   Parameters:
     context - the context from the request
     policies - the policies to combine
   Returns:
     the result of running the combining algorithm
```

- ***PermitOverridesPolicyAlg.java*** This is the standard *PermitOverrides* policy combining algorithm. It allows a single evaluation of Permit to take

precedence over any number of deny, not applicable or indeterminate results. Note that since this implementation does an ordered evaluation, this class also supports the Ordered Permit Overrides algorithm.

```
//Applies the combining rule to the set of
//policies based on the evaluation context.
 public Result combine(EvaluationCtx context,
            List policies)
    Parameters:
       context - the context from the request
       policies - the policies to combine
    Returns:
       the result of running the combining algorithm
```

For rules also there are combining algorithms like policies

- DenyOverridesRuleAlg
- PermitOverridesRuleAlg

After evaluating the RequestContext the PDP will get a unified result, the PDP then formulates ResponseContext and send that to PEP.

4. ***ResponseCtx.java***

```
//Represents the response to a request
//made to the XACML PDP.
public void encode(OutputStream output, Indenter indenter)
    Parameters:
       output - a stream into which the XML-encoded data
       is written
       indenter - an object that creates indentation
       strings
```

When the PEP gets the ResponseContext which is encoded in XACML format, it parses the ResponseContext and gets the decision. The PEP then sends the decision to the custom Authorization module in the globus Toolkit. The PEP also sends the log information like the *role* in the Request context, time when

the request reached PEP, reasons if the decision is Deny and so on. All this communication is done through sockets over network.

The Authorization module (MainPDP.java) extracts the system specific attributes like free memory and cpu load dynamically and uses these attributes in the final decision making along with the decision from the XACML decision engine.

```
//extracts system dynamic attributes
 public double loadavg() {
// reads the load average form /proc/loadavg
public  int freemem() {
// reads the memory details from /proc/meminfo
```

If these criteria are satisfied, the Authorization module allows the client's request to the service. Otherwise, it denies the request.

7.3.1 Referencing the MainPDP from a security descriptor

The Mathservice uses the custom Authorization Module to control the access to the operations in the service. The service security descriptor file specifies the security configuration for a service.

```
In the project the Mathservice security descriptor
file is located at /opt/globus-4.0.4/etc/math_service/
math-service-security-descriptor.xml.
After modification the file  will be like this:

<securityConfig xmlns="http://www.globus.org">
  <auth-method>
      <GSITransport/>
  </auth-method>
  <authz value="ascope:org.globus.sampleauthz.MainPDP"/>
```

Our service should refer to the security descriptor file first. We have to add the following parameters to the WSDD file, where the path to the security descriptor

file is relative to GLOBUS_LOCATION. The WSDD file for Mathservice is placed at

```
/opt/globus-4.0.4/etc/math_service/server-config.wsdd
```

```
 <!-- This configuration enables the authentication
      and authorization
    options for this service found in the specified
    securityDescriptor
  -->
 <parameter name="securityDescriptor"
    value="etc/math_service/math-service-security-
    descriptor.xml"/>
```

7.4 Implementation of Cross-Domain Authorization

The modules which are required to be implemented for cross domain role mapping and authorization are as shown in Figure 23. The code shown below is a part of Authorization Server(AS)of the client domain to which the request for role mapping comes from its redirection server. Here, essentially the code tries to find the type of the message received from the Redirection server. It finds out the domain of the client whose role is being requested and then retrieves its role. Then it creates a client certificate as shown in the sequence diagram and then adds the rank of the role retrieved and sends it back to the AS.

```
if(msginstanceof ClientNotFoundRequest)
{
ClientNotFoundRequest im = (ClientNotFoundRequest)msg;
String domainC = im.getDomainClient();
System.out.println(domainC);
if(domainC.equals("nitw.ac.in"))
{
String ClientName = im.getName();
int RankC = getRank(ClientName);
```

```
ClientCertificate request = new ClientCertificate(1,RankC,1);
sendMessage(s,request);
}
}

if(msginstanceof ClientNotFoundRequest)
{
ClientNotFoundRequest im = (ClientNotFoundRequest)msg;
String domainClient = im.getDomainClient();
//get the domain of the client
String domainService = im.getDomainService();
System.out.println("the message received is" + domainClient
+ domainService);
if(domainClient.indexOf("ac.in") != -1)
//check if the domain is a sub domain
{
String serverAddress = getAddress(domainClient);
int serverPort = getPort(domainClient);
System.out.println("the server and the port are"
+ serverAddress+serverPort);
try {
sock = new Socket(serverAddress,serverPort);
}catch(IOException ioe)
{
System.out.println("ServerWindow() : mainsock : " + ioe);
}
sendMessage(sock, im);
OQMessage msg1 = receiveMessage(sock);
if(msg1instanceof ClientCertificate)
{
ClientCertificate cert = (ClientCertificate)msg1;
cert.setRank(2);
cert.setRank2(cert.getRank2()*getRank(domainClient));
```

```
cert.setRank1(cert.getRank1()*getRank(domainService));
sendMessage(s,cert);
}
}
}
```

The above code is a module from the Redirection server of the ac.in domain. It does the following:

1. gets the domain name of the client and also the name of the client
2. gets the domain for which the service is requested
3. gets the address of the AS of the client domain
4. creates a socket for the same and sends the clientNotFoundRequest message to the AS
5. multiplies the client ranking (Rank1) with the rank of the client domain
6. multiplies the service ranking (Rank2) with the rank of the Service Domain
7. changes the variable rank to 2. This is to indicate that the next AS or RS should multiply the rank of its sub domain with the Rank2
8. passes this client certificate to the server form which request has come

7.5 Implementation of Lightweight Directory Access Protocol

Lightweight Directory Access Protocol or LDAP [2] helps in introducing additional user attributes in the access control mechanism thereby helping in implementing finer access control on the grid resources. We have proposed an LDAP architecture for integration with the grid environment. The LDAP server contains the full details of users and resources in the environment and is helpful in making our access control policies more fine grained with the help of additional attributes. The Grid Resource Information Protocol (GRIP) of Globus [135], makes use of LDAP as the standard resource information protocol. LDAP is also used as a catalog access protocol. But, we mainly associate LDAP with the user attributes to achieve certain level of finer attribute representation and faster access.

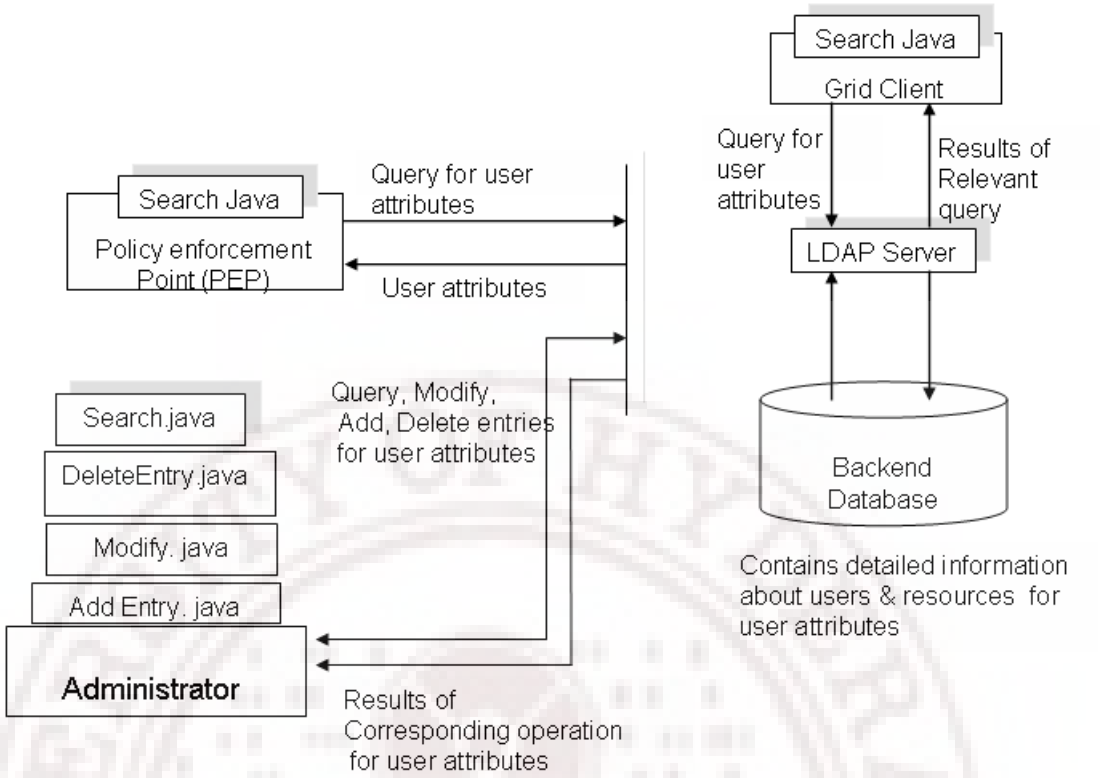


Figure 45: LDAP Architecture for Grid Environment

7.5.1 LDAP Implementation in Grids

The technology requirement for LDAP implementation in the grid access control include:

- Java - for Interfaces design and service implementation
- mysql - Backend database for OpenLDAP
- OpenLDAP - server to store details about users and resources in the organization

The steps involved in implementing LDAP are:

1. The LDAP server accepts the request from the grid Client. (information on different users and resources in the organization)
2. When a grid client requests for a service, the Policy Enforcement Point (PEP) before forming the XACML request queries the LDAP server for more user attributes

3. The LDAP server then returns the result of the query to the PEP
4. The administrator is also allowed to add new users, change attributes and modify existing users through a administrator GUI

The PEP searches the LDAP server when the access control policies are too complex and also when more user attributes (not just the user's role) are required.

7.5.2 Advantages of LDAP in Grids

The use of LDAP alongside a database improves the performance of the grid access control mechanism. Using a directory service is a performance enhancing practice. In a grid environment, *reads* must be performed very efficiently so as to service the access requests as fast as possible. As mentioned earlier, a grid mostly spans a large geographical area comprising of many domains with databases for each such domain. But the complex functions supported by such a database management system are generally not required in a grid environment. As directories are optimized for *read* accesses it becomes advisable to use them in the grid environment. Secondly, the very structure of LDAP is hierarchical and hence offers a tree-like view of the whole grid environment. This is particularly necessary in organizations so that the client can get to know the organizational structure. Combining a relational database management system with LDAP is unadvisable as the grid environment is vast and complex and also because the data models are very different. Representing directory data with a relational database requires splitting the data into multiple tables. This means that accessing data, from even one entry requires seeking/searching on different disk areas which in turn results in reduced system performance.

7.6 Case Study on the Security Aspects of Garuda Grid

As this thesis work deals with the core security challenges in grid computing environment, it is felt necessary to conduct a study on any existing grid infrastructure. We opted for the National Grid Computing Initiative of India - code named as GARUDA [4] - for this purpose. This chapter provides an overview of the GARUDA grid, which include its strategic objectives, key deliverables and major components (both physical as well as technological). Next, we discuss various

approaches adopted in GARUDA grid to address the security issues. Some of these techniques have already been deployed and initiated while some are in the design and implementation phases. The organizations and the standards which help implement GARUDA security have been briefly explained as per the available details. Then, it explores ways to incorporate some of the access control and authorization methodologies suggested in this thesis with GARUDA grid security. The chapter ends with a summary of the GARUDA grid security.

7.6.1 GARUDA - An Introduction

GARUDA is a collaboration of science researchers and experimenters on a Nation-wide grid of computational nodes, mass storage and scientific instruments in India that aims to provide the technological advances required to enable data and compute intensive applications for the 21st century. The Nation-wide computational grid GARUDA aims to connect 17 cities across the country in its Proof of Concept (PoC) phase with an aim to bring "grid" networked computing to research labs and industry. An effort under the Center for Development of Advanced Computing (C-DAC) [3], GARUDA aims to accelerate India's drive to turn its substantial research investment into tangible economic benefits by strengthening and advancing scientific and technological excellence in the area of grid and peer-to-peer technologies. The strategic objectives of GARUDA are to:

- create a test bed for the research and engineering of technologies, architectures, standards and applications in grid computing
- bring together all potential research, development and user groups to develop a national initiative on grid computing
- create the foundation for the next generation grids by addressing long term research issues in grid computing

The following are the key deliverables identified as important to achieve the GARUDA objectives:

- Grid tools and services to provide an integrated infrastructure to applications and higher-level layers

- A Pan-Indian communication fabric to provide seamless and high-speed access to resources
- Aggregation of resources including compute clusters, storage and scientific instruments
- Creation of a consortium to collaborate on grid computing and contribute towards the aggregation of resources

The major components of GARUDA as shown in Figure 46 include the computing resources, high-speed communication fabric, middleware and security mechanisms, tools to support program development, collaborative environments, data management and grid monitoring and management. Access portals and specialized problem solving environments provide a seamless user interface to the grid.

Figure 46: Core Components of GARUDA

The technology components of GARUDA include:

- Access methods and problem solving environments
- Data management

- Program development
- Collaborative environment
- Grid middleware and security
- Grid monitoring and management

The GARUDA High-Speed network is a Layer 2/3 MPLS Virtual Private Network (VPN) connecting select 45 institutions across 17 cities at 10/100 Mbps with stringent service level agreements with the service provider. This grid is a precursor to the Gigabit speed nation-wide Wide Area Network (WAN) connecting high performance computing resources and scientific instruments for seamless collaborative research and experiments. The high speed network is being established at all the Garuda partner institutes in close collaboration with ERNET which is also responsible for the operation, maintenance and management of this network.

7.6.2 Approaches to Handle Security Issues in GARUDA

The following section highlights the existing practices for implementing security in GARUDA-grid.

- evolving security practice statements and GARUDA security policies
  - evolve responsibilities for resource providers as well as users
  - carry out risk assessment and fix the weakness
  - form a group to evolve policies
- participation in International Grid Trust Federation (IGTF)
  - Asia Pacific Grid Policy Management Authority (PMA)
  - European Grid PMA
  - The Americas Grid PMA
- GARUDA - Computer Emergency Response TEAM (G-CERT)
  - To be a collaborative effort by all/many partners
  - To establish points of trusted contacts for computer security threats in GARUDA

- To provide certain mandatory services like announcements, vulnerability analysis, technical analysis and reports, education, incident tracing, intrusion detection, audits and penetration testing, security consultancy, risk analysis etc
- Create documents pertaining to best practices and policy guidelines (site security and inter-site security handbook)
- To develop capabilities to support activities such as tracking and tracing intruder activity and active detection of such activities
- Interface with CERT-IN and other CERTs
- GRIP - Guidelines and Recommendations for Security Incident Processing

7.6.3 User Registration in GARUDA

GARUDA uses a Portal Based User Registration Service - PURSE for user registration purposes. The registration authority is called RA - Registration Authority (GARUDA Contact Person) and the certificate authority is called CA - Certificate Authority (CDAC-KP). As a portal based user registration service, PURSE helps in registering users through web-based applications using the Grid Security Infrastructure, based on PKI and X.509 certificates. GARUDA uses PURSE for registration of new users. GARUDA management has identified from each partner institute, one or more GARUDA Contact Persons (GCP), who act as RA from the Partners side. A certificate is a digital document that certifies that a certain public key is owned by a particular user. This document is signed by Certificate Authority (CA). CA will send the information about the requestor to RA (GCP). CA will have no knowledge about the requestor and will wait for the GCP. A Registration Authority (RA) is an authority in a network that verifies user request for a digital certificate and tells the Certificate Authority (CA) to issue the certificate. The following figure shows the PURSE architecture.

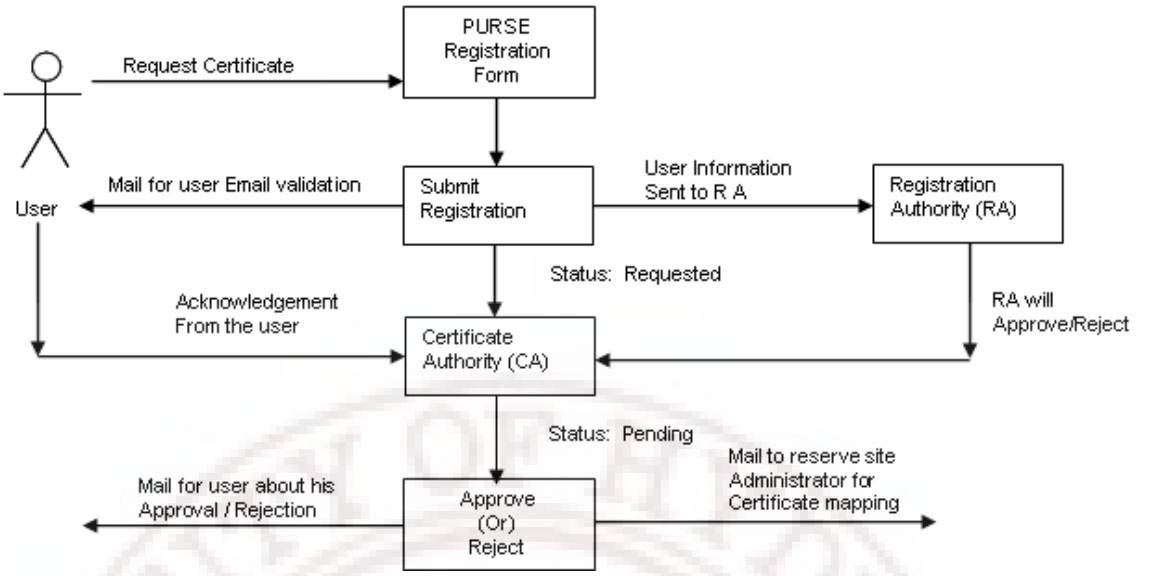


Figure 47: PURSE Architecture

7.6.4 Summary of GARUDA Grid Security

GARUDA security has been implemented in the form of two core mechanisms namely authentication and authorization. The authentication features include

- GARUDA Certificate
  - Subject Name
  - Public Key of the Subject
  - Identity of GARUDA CA
  - Digital Signature of GARUDA CA
- Adheres to GSI
  - Credential Delegation
  - Single Sign-On

The authorization features are

- User Mapping
- DN to Pool of Unix accounts

7.6.5 Suggestions for Improving Garuda Grid Security

Authentication is being used as the basic form of user identification and validation in Garuda Grid. Authorization attempts at providing restricted access rights to the users seems to be at the very initial stages in Garuda. The Moab Scheduler from Cluster Resources interfaces with Globus for user management and security. We suggest the use of an authorization-centric security mechanism in Garuda. The Role-Based Grid Authorization Model which has been proposed in Chapter III of this thesis could be incorporated in Garuda. This model, on one hand provides support for all the proven techniques of access control and authorization (as in RBAC), for a grid environment. Additionally, it facilitates *delegation* which can be a very useful indirect authorization mechanism for securing Garuda resources. Our framework is complete with features like revocation, multi-step delegation and dynamic access control.

7.7 Chapter Summary

The chapter summarizes the experimental set up and implementation details of the contributions made in Chapters III and IV of this thesis. We show as to how we can write a grid service, design an authorization mechanism for the service and then enforce the mechanism for secured resource access. We provide authorization techniques for single domain as well as cross-domain environment. We have made use of various tools and techniques for implementation of access control like the Globus middleware, eXtensible Access Control Markup Language (XACML), Lightweight Directory Access Protocol (LDAP). We used *mysql* as the back-end database for representing the RBAC policies. The entire implementation was done on the Redhat Linux platform. We supported the implementation with a case study on Garuda grid's security features.

CHAPTER VIII

CONCLUSION AND FUTURE SCOPE

In this chapter, we summarize the major contributions of the research work presented in this thesis. We also highlight the future scope and further research directions. This thesis contributes to the modelling of grid security measures along with the development of the implementation mechanisms. This work covers the fundamental aspects of grid security through access control modelling. We use algorithmic representations for the theoretical characterization of the proposed models. The applicative issues have been addressed through implementation of authorization systems. Performed on the standard Globus middleware platform, these implementations help us understand the working of the proposed models.

8.1 Summary of Contributions

Grid Computing is rapidly emerging as the dominant paradigm of wide area distributed computing. Its primary objective is to provide a service-oriented infrastructure that leverages standardized protocols and services to enable pervasive access and coordinated sharing of geographically distributed hardware, software and information sources. Grid applications are distinguished from traditional client-sever applications by their simultaneous use of large number of resources, dynamic resource requirements, use of resources from multiple administrative domains, complex communication structures and stringent performance requirements among others. While scalability, performance and heterogeneity are the desirable goals for any distributed system, the characteristics of grids lead to security problems that are not addressed by existing security technologies for distributed systems. Many of the present grid security solutions are based on user identity rather than user's access rights. In the present set up, the resource access is coarse-grained. Further more, the dynamic nature of a grid can make it impossible to establish trust relationships between sites prior to application execution. Also, the

diverse intra-domain access control solutions should interoperate with the inter-domain security technologies.

In this thesis, we have proposed new techniques that overcome many of the cited difficulties. We presented access control models that address requirements for direct authorization, indirect authorization (delegation), inter-operability with local policies (across the multiple administrative domains) and dynamically varying resource requirements. We also presented an implementation mechanism for the models with a case study on "Garuda", the National Grid Computing Initiative of India. In summary, this thesis makes four major contributions towards the advancement of grid computing security, supported by implementation and case study. The details are as below:

- **Role-Based Direct Authorization:** We provide architectural frameworks for representation of scalable access control and authorization mechanism built on the standard RBAC which uses the concept of *role* as the unit of authorization. First, we propose a single-domain authorization mechanism by treating the grid as a single logical enterprise. Then we extend this framework for cross-domain authorization for a grid envisaged as a global enterprise comprising diverse physical domains. The first step for cross-domain authorization, is to map the role of a given domain to its equivalent role in another domain. The classical Role Based Access Control, though an effective access control standard for single enterprise, does not address the issue of resolving a local role into a global role. So, we suggested a role-mapping architecture, to establish role equivalence among the domains by mapping a local domain role to its equivalent global role. We use the approach of weighted role-ranking for the same. The authorization decision is then made based on the mapped global role ranking and also the resource access policies.

- **Role-Based Grid Delegation:** We proposed the use of delegation as the primary form of authorization in grids. As credential-based delegation has its own limitations in a dynamic grid environment, a new conceptual model is required to effectively formulate the grid delegation requirements. Here, we present a framework called Role-Based Grid Delegation Model (RB-GDM) for

delegating access rights in a single grid enterprise. The basic unit of delegation in this model is *role*. Derived from the standard RBAC formalisms, this framework explores various approaches for indirect authorization through delegation. We present the RB-GDM interconnection framework for various scenarios like intra-domain and inter-domain delegation (both peer-to-peer as well as master/slave topologies). We suggest various algorithms and show how the components of these interconnection frameworks work. As the users have the choice of delegating their sub roles (junior roles), this enforces one of the basic principles of security: "The Principle of Least Privilege".

As revocation of rights is as important as delegation, we present mechanisms by which one can revoke the delegated rights. The revocation methods include grant-dependent, grant-independent as well as timeout revocation. We then extend this technique for cross-domain delegation through roles. We implemented these frameworks on the Globus platform with two academic organizations being used as illustrative domains.

- **Trust-Based Dynamic Delegation Model:** Here, we propose a Fuzzy Trust and Delegation Model (FTDM) to address the dynamic access control requirements of grids. As the grid resources are distributed in space and time, we need to have authorization mechanisms implemented in dynamic context. Also, trust relationships are central to the implementation of security in the dynamic and ever evolving communities like grids. The model comprises a two-stage fuzzy inference process wherein the first stage computes and quantifies the trust values of the sites involved in resource access based on certain criteria and the second stage determines the degree of delegation based on the trust values obtained from the first stage of fuzzy process. Use of fuzzy inference systems to make access decision is a relatively new approach in grids. This methodology suits the dynamic grid environment as fuzziness or uncertainty is typical to dynamic contexts. The rule design used in this model ensures formulation of an efficient dynamic delegation policy in grids.

- **Fine-Grained Access Control Framework for Grid Resources:** We provide a flexible, trust-aware fine-grained access framework for grid access control

from the context of access permissions and resources allocation. Presently grid security mechanisms maintain confidentiality and integrity of resource access through coarse-grained grant/deny permissions to known entities. In the proposed framework, the access policy (grant/deny or partial grant), is governed by the trust relationships among the grid domains. We quantify various categories of trust namely direct trust, transitive trust and asymmetric trust using fuzzy inference process and map these values to access control decisions. We consider varying degrees of access like full access, partial access (fine grained) or access denial (no access). The quantification of trustworthiness of a resource site is done using fuzzy inference systems. This framework can be incorporated with legacy access control mechanisms.

A framework is useful only if it is proven that it works well for that environment. We implemented the authorization models proposed through this thesis on the Globus platform. The implementation proves that these models suit the requirements of grid security. Many of the major issues of grid security have been addressed through these models. We also conducted a case study on the security requirements of "Garuda", India's grid initiative and suggested ways to incorporate some of the models to improve "Garuda" security. Though Lightweight Directory Access Protocol (LDAP) has been the basis for GRIP (Grid Resource Information Protocol) of the Globus Toolkit, its use in our work for the storage and retrieval of additional user attributes in the form of a directory structure, is an innovation. We used the eXtensible Access Control Markup Language (XACML) to specify policies which, being platform independent, adds value for inter-operability.

8.2 Conclusion

We have proposed a set of architectural solutions for grid access control and authorization. The models presented in this work support direct access control, delegation, dynamic authorization and trusted computing. The contributions made in this thesis can thus enhance the grid computing security.

8.3 Future Work

In Chapter III of this thesis, we have proposed a grid authorization model which supports inter-domain authorization. To make the design simple, we have considered two sample administrative domains of the same type (academic domain). Where as in real applications, the resource access could be across heterogeneous domains which makes role-mapping even more complex. So, our future work could incorporate heterogeneity in the access control mechanism.

The implementation details which we described in Chapter VII of this thesis advocates the use of XACML for policy representation. But the presence of diverse administrative domains makes a grid characteristically different. Services in grids may use different resources, held by different individuals or organizations. Thus, there is a lot of scope to extend this research work to study the feasibility of using XACML policy integration algorithms in grids. Another extension could be the use of XACML-RBAC, a recent development in XACML which comes integrated with RBAC elements and tags and defines a rich policy format for the generic RBAC and also for the simple Request/Response messages format used for PEP-PDP communication.

Our models provide inter-domain access mechanisms. However, a separate policy communication module between organizations and VOs, could be implemented. This is very essential in the context of ensuring inter-operability with local security solutions. As the fundamental rule of grid security is to have "local control over own resources", it is impractical to modify every local resource to accommodate inter-domain access. Though there are proposals for formulating a multi-policy framework for grids at the VO level, implementation efforts in this direction have been minimal.

Another possible improvement is regarding the use of LDAP in the grid environment. We have used LDAP to make access control policies more strict by considering additional attributes of the user/resource in the request. There are practical difficulties regarding integration of LDAP with the grid implementation due to the difference in data models. We have used *mysql* as the back-end database where as the *OPEN LDAP* implementation supports *BERKELEY* database. We have to

develop mechanisms by which we can interface LDAP with all database distributions.

In Chapter III of the thesis, we have proposed a role-mapping architecture based on weighted roles to map a role of one administrative domain to another domain. We can improvise this architecture by suggesting a complete and robust algorithm to rank the roles across domains and also to rank the domains in a virtual organization setup. This should be based on the behavioral patterns of the clients and the service providers.

We implemented the models on the Globus platform. But, we believe that security policies or models should not dictate a specific implementation technology. Rather, it should be possible to implement the models with a range of security technologies. Our proposal to implement the authorization models on Garuda Grid is a step in this direction. Implementation of the proposed models on other middleware platforms like the Gridbus, could also be considered.

Although we have incorporated the access control and authorization mechanism as part of the grid service itself, we can make this mechanism generic, if we could separate it as a service itself. This will enable us to implement a common access control interface for a set of varied services.

REFERENCES

[1] eXtensible Access Control Markup Language, Version 2.0, OASIS Standard. *http://docs.oasisopen.org/xacml/2.0/access control-xacml-2.0-core-spec-os.pdf*, 2005.

[2] Open LDAP Details. *http://openladap.org/*, 2007.

[3] Garuda India Grid Computing. *http://www.cdac.in/html/gridcomputing.asp*, 2008.

[4] Garuda: The National Grid Computing Initiative. *http://www.garudaindia.in*, 2008.

[5] Grid Computing. *http://h71028.www7.hp.com/enterprise/cache/125369-0-0-0-121.html*, 2008.

[6] Grid Computing Solutions. *http://www.sun.com/software/grid/*, 2008.

[7] Open Grid Forum. *http://www.globalgridforum.org*, 2008.

[8] Oracle Grid Computing. *http://www.oracle.com/technologies/grid/index.html*, 2008.

[9] M. Abadi and C. Fournet. Access Control Based on Execution History. In *The 10th Annual Network and Distributed System Security Symposium*, 2003.

[10] G.J. Ahn and R. Sandhu. Role-Based Authorization Constraints. *ACM Transactions on Information and System Security*, (4), 2000.

[11] M.A. Al-Kahtani and R Sandhu. A Model for Attribute-Based User-Role Assignment. In *Proceedings of the 18th Annual Computer Security Applications Conference*, Las Vegas,NV, 2002.

[12] R. Alfieri and R .Cecchini. From Gridmap-File to VOMS: Managing Authorization in a Grid Environment. *Future Generation Computer Systems (FGCS)*, 21:549–558, 2005.

[13] Farag Azzedin and Muthucumaru Maheswaran. Towards Trust-Aware Resource Management in Grid Computing systems. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[14] J.F Barkley and A.V Cincotta. Implementation of Role/Group Permission Association Using Object Access Type. *US Patent 6,202,066*, 2002.

[15] Messaoud Benantar. *Access Control Systems*. Springer Publications, 2006.

[16] V. Bertstis. Fundamentals of Grid Computing. *Red Books, IBM Corporation*, 2002.

[17] Matt Bishop. *Computer Security : Arts and Science*. Addison Wesley, 2002.

[18] Matt Bishop. *Introduction to Computer Security*. Addison Wesley, 2004.

[19] Botha,R.A. and J.H.P Eloff. A Framework for Access Control in Workflow Systems. *Information Management and Computer Security*, (3):126–133, 2001.

[20] Ed Lewis, Jan Newmarch, Lawrie Brown and Yinan Yang. A Trusted W3 Model: Transitivity Of Trust In A Heterogeneous Web Environment. In *The Fifth Australian World Wide Web Conference*, pages 17–20, Balliina, NSW, Australia, 1999.

[21] Guanying Bu and Zhiwei Xu. Access Control in Semantic Grid. *Future Generation Computer Systems Journal*, pages 113–122, 2004.

[22] Rajkumar Buyya. Delivering Grid Services as ICT Commodities:Challenges and Opportunities. *Report, GRIDS Laboratory,University of Melbourne*, 2003.

[23] Rajkumar Buyya. Grid Computing Info Centre: Grid Computing, Answers to the Enterprise Architect Magazine Query. *Enterprise Architect Magazine*, 2008.

[24] Rajkumar Buyya and Srikumar Venugopal. The Gridbus Toolkit for Service Oriented Grid and Utility Computing:An Overview and Status Report. In *IEEE International Workshop on Grid Economics and Business Models*, pages 19–36, 2004.

[25] Calvelli C. and Varadharajan V. An Analysis of Some Delegation Protocols for Distributed Systems. In *Computer Security Foundations Workshop*, 2002.

[26] CI. Catlett. The Rise of Third-Generation Grids. *Grid Connections*, (3):42–47, 2003.

[27] C.Canovas and Antonio F. Gomez. Delegation in Distributed Systems: Challenges and Open Issues. In *Workshop on Database and Expert Systems Applications*, 2003.

[28] D.W. Chadwick and A. Otenko. The PERMIS X.509 Role-Based Privilege Management Infrastructure. In *7th ACM Symposium on Access Control Models and Technologies*, pages 135–140, 2002.

[29] Anirban Chakrabarti. *Grid Computing Security*. Springer, 2007.

[30] Liang Chen and Jason Crampton. Inter-domain Role Mapping and Least Privilege. In *Symposium on Access Control Models and Technologies (SACMAT)*, 2007.

[31] Chetty,M.and Rajkumar Buyya. Weaving Computational Grids:How Analogous Are They With Electrical Grids? *IEEE Computing in Science and Engineering*, 2002.

[32] Christian Vecchiola, Xingchen Chu and Rajkumar Buyya. Aneka: A Software Platform for .NET-Based Enterprise Cloud Computing. In *International Conference on High Speed and Largescale Computing*, 2010.

[33] I. Foster, C.Kesselman and S.Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[34] OASIS Organization, Security Services Technical Committee. Security Assertion Markup Language. *http://www.oasis-open.org*, 2003.

[35] David Cordes and Jiageng Li. A Scalable Authorization Approach for the Globus Grid System. *Future Generation Computer Systems (FGCS)*, 21:291–301, 2005.

[36] IBM Corporation. Enterprise Security Architecture using IBM Tivoli Security Solutions. *Technical Report,IBM*, 2002.

[37] IBM Corporation. IBM Resource Access Control Facility. *http://www-1.ibm.com/servers/eserver/zseries/zocs/racf/*, 2002.

[38] IBM Corporation. Tivoli Security Management Design Guide. *Technical Report,IBM*, 2002.

[39] IBM Corporation. IBM and Partners Bring Grid Computing to Linux on the Mainframe. *IBM Press Release*, 2003.

[40] Oracle Corporation. Web Services Overview, Programmatic Access to Web Sites and Applications. *Oracle Magazine, http://otn.oracle.com/oraclemagazine*, 2002.

[41] J Crampton and G. Loizou. Administrative Scope and Role Hierarchy Operations. In *Seventh ACM Symposium on Access Control Model and Technologies*, pages 145–154, Monterey,CA,USA, 2002.

[42] Elisa Bertino, Pietro Mazzoleni, Bruno Crispo and Swaminathan Sivasubramanian. Towards Supporting Fine Grained Access Control for Grid Resources. In *IEEE International Workshop on Future Trends in Distributed Computing Systems (FTDCS 2004)*, pages 59–65, 2004.

[43] C. Del Prete. On Demand by IBM. *IDC Viewpoint*, 2003.

[44] Miguel L. Bote-Lorenzo, Yannis A. Dimitriadis and Eduardo Gmez-Snchez. Grid Characteristics and Uses: A Grid Definition. In *EuropeanAcross Grids Conference*, pages 291–298, Santiago de Compostela, Spain, 2003.

[45] E.Barka and R.Sandhu. A Role-Based Delegation Model and Some Extensions. In *National Information Systems Security Conference*, Baltimore, MD, 2003.

[46] E.Barka and R.Sandhu. Role-Based Delegation Model/ Hierarchical Roles (RBDM1). In *20th Annual Computer Security Applications Conference*, Tucson, Arizona, USA, 2004.

[47] James B. D. Joshi et. al. Access-Control Language for Multidomain Environments. *IEEE Internet Computing*, 2004.

[48] William Tolone et. al. Access Control in Collaborative Systems. *ACM Computing Surveys*, (1):29–41, 2005.

[49] Abadi, M. et.al. Logic in Access Control. In *18th Annual IEEE Symposium on Logic in Computer Science*, pages 228–233, Ottawa, Canada, 2003.

[50] Adam Hess et.al. An Access Control Model for Dynamic Client-Side Content. In *The 8th ACM Symposium on Access Control Models and Technologies*, pages 207–216, Como,Italy, 2003.

[51] Akshay Luther, Rajkumar Buyya et.al. Alchemi: A .NET-Based Enterprise Grid Computing System. In *International Conference on Internet Computing*, 2005.

[52] Babu Sundaram et.al. Addressing Credential Revocation in Grid Environments. In *Grid Computing Workshop 2005*, pages 323–326, 2005.

[53] Bertino,E. et.al. TRBAC-Temporal Role-Based Access Control Model. *ACM Transactions on Information and System Security*, (4):65–104, 2001.

[54] Bertino,E. et.al. A Logical Framework for Reasoning about Access Control Models. *ACM Transactions on Information and System Security*, (1):71–127, 2003.

[55] Bo Lang et.al. A Multipolicy Authorization Framework for Grid Security. *Technical Report, Globus Consortium*, 2005.

[56] Bonatti,P. et.al. An Algebra for Composing Access Control Policies. *ACM Transactions on Information and System Security (TISSEC)*, (1):1–35, 2003.

[57] Booth, D. et.al. Web Services Architecture. *W3C Working Draft*, 2003.

[58] Buda, G. et.al. Security Standards for the Global Information Grid. In *The Military Communications Conference (MILCOM)*, 2001.

[59] Cabrera, F. et.al. Web Services Transaction. *Microsoft Archives*, 2002.

[60] Cannon, S. et.al. Using CAS to Manage Role-Based Sub-Groups. In *CHEP03*, La Jolla, California, USA, 2003.

[61] Chadwick,D. et.al. Providing Secure Coordinated Access to Grid Services. In *MGC2006*, Melbourne, Australia, 2006.

[62] Chervenak et.al. The Data Grid:Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. *Journal of Network and Computer Applications:Special Issue on Network-Based Storage Services*, (3):187–200, 2000.

[63] Chun Ruan et.al. A Weighted Graph Approach to Authorization Delegation and Conflict Resolution. In *ACISP 2004*, pages 402–413, 2004.

[64] Cohen,E. et.al. Models for Coalition Based Access Control. In *The 7th Symposium on Access Control Models and Technologies*, 2002.

[65] Czajkoski et.al. Grid Information Services for Distributed Resource Sharing. In *10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, 2001.

[66] Damiani,E. et.al. A Fine-Grained Access Control System for XML Documents. *ACM Transactions on Information and System Security (TISSEC)*, (2):169–202, 2003.

[67] David Chadwick et.al. PERMIS: A Modular Authorization Infrastructure. *Concurrency and Computation:Practice and Experience*, (11):1341 – 1357, 2008.

[68] DeTreville, J. et.al. Binder, A Logic Based Security Language. In *The IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 2002.

[69] Earnesto Damiani et.al. New Paradigms for Access Control in Open Systems. In *SACMAT 2006*, Lake Tahoe, California, USA, 2006.

[70] Ferraiolo, Sandhu et.al. NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, 4(3), 2001.

[71] Ferraiolo,D. et.al. The Role Administration Center:Features and Case Studies. In *The 8th ACM Symposium on Access Control Model and Technologies*, 2003.

[72] Foster, I. et.al. Globus:A Toolkit-Based Grid Architecture. *The Grid:Blueprint for a New Computing Infrastructure*, pages 259–278, 1999.

[73] Foster, I. et.al. The Physiology of the Grids : An Open Grid Services Architecture for Distributed Systems Integration. *Technical Report, Argon National Labs*, 2002.

[74] Fox, G. et.al. Overview of Grid Computing Environments. *GFD-1.9*, 2003.

[75] Fruedenthal,E. et.al. dRBAC: Distributed Role-Based Access Control for Dynamic Coalition Environments. In *22nd International Conference on Distributed Computing Systems (ICCS 2002)*, 2002.

[76] Gabriel Kuper et.al. Generalized XML Security Views. In *SACMAT 2005*, pages 77–84, Stockholm, Sweden, 2005.

[77] Gabrielle Allen et.al. The Grid Application Toolkit: Toward Generic and Easy Application Programming Interfaces for the Grid. *Proceedings of the IEEE*, (3):534–550, 2005.

[78] Gallaher,M.P. et.al. The Economic Impact of Role-Based Access Control. *Planning Report01-2, National Institute of Standards and Technology*, 2001.

[79] G.Stoker et.al. Toward Realizable Restricted Delegation in Computational Grids. In *International Conference on High Performance Computing and Networking(HPCN 2001)*, Amsterdam, Netherlands, 2001.

[80] HePingHu et.al. A Scheme for Authentication and Authorization in a Grid Application. In *19th International Conference on Advanced Information Networking and Applications (AINA05)*, 2005.

[81] Herzberg,A. et.al. Access Control Meets Public Key Infrastructure,Or Assigning Roles to Strangers. In *IEEE Symposium on Security and Privacy*, pages 2–14, Oakland,CA, 2000.

[82] Jacques Wainer et.al. A Fine-grained, Controllable,User-to-User Delegation Method in RBAC. In *SACMAT 2005*, pages 59–66, Stockholm, Sweden, 2005.

[83] Jacques Wainer et.al. Fine-Grained Role-Based Delegation in Presence of the Hybrid Role Hierarchy. In *SACMAT 2006*, pages 81–90, Lake Tahoe, CA,USA, 2006.

[84] Jajodia,S. et.al. Flexible Support for Multiple Access Control Policies. *ACM Transactions on Database Systems*, (2), 2001.

[85] Jeffrey Dwoskin et.al. Scoping Security Issues for Interactive Grids. In *37th Annual Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, California,USA, 2003.

[86] Jim Longstaff et.al. The Tees Confidentiality Model: an Authorization Model for Identities and Roles. In *The 8th ACM Symposium on Access Control Models and Technologies*, pages 125–132, Como,Italy, 2003.

[87] Joerg Abendroth et.al. Partial Outsourcing: A New Model for Access Control. In *The 8th ACM Symposium on Access Control Models and Technologies*, pages 134–141, Como,Italy, 2003.

[88] Joshi,J. et.al. Digital Government Security Infrastructure Design Challenges. *IEEE Computer*, (2):66–72, 2001.

[89] Joshi,J. et.al. Security Models for Web-Based Applications. *Comunications of the ACM*, (2):38–44, 2001.

[90] Joshi,J.B.D. et.al. Temporal Hierarchies and Inheritance Semantics for GTR-BAC. In *Seventh ACM Symposium on Access Control Model and Technologies*, Monterey,CA,USA, 2002.

[91] Kang,M.H. et.al. Access Control Mechanisms for Interorganizational Workflow. In *The 6th ACM Symposium on Access Control Models and Technologies*, pages 66–74, 2001.

[92] Kevin Kane et.al. On Classifying Access Control Implementations for Distributed Systems. In *SACMAT 2006*, pages 29–38, 2006.

[93] Li, N. et.al. Delegation Logic: A Logic-Based Approach to Distributed Authorization. *ACM Transactions on Information and System Security (TISSEC)*, (1):128–171, 2003.

[94] Linda A Cornwall et.al. Authentication and Authorization Mechanisms for Multi-Domain Grid Environments. *Journal of Grid Computing*, pages 301–311, 2004.

[95] Lorch, M. et.al. The PRIMA System for Privilege Management, Authorization and Enforcement in Grid Environments. In *Fourth International Workshop on Grid Computing (GRID03)*, 2003.

[96] Ludwig Seitz et.al. Policy Administration Control and Delegation using XACML and Delegent. In *The IEEE Grid Computing Workshop 2005*, 2005.

[97] Manuel Koch et.al. A Graph-Based Formalism for RBAC. *ACM Transactions on Information and System Security (TISSEC)*, (3):332–365, 2002.

[98] Markus Lorch et.al. Authorization and Account Management in the Open Science Grid. In *Grid Computing Workshop 2005*, pages 17–24, 2005.

[99] Mary Thompson et.al. Overview of the Akenti Access Control System. In *DOE2000 Architectural Workshop*, 2000.

[100] Mazzoleni,P. et.al. XACML Policy Integration Algorithms. In *SACMAT 2006*, Lake Tahoe, California, USA, 2006.

[101] Meharan Ahsant et.al. Grid Delegation Protocol. In *UK Workshop on Grid Security Experiences*, 2004.

[102] Mohammad Kahtani et.al. Inducing Role Hierarchies with Attribute-Based RBAC. In *The 8th ACM Symposium on Access Control Models and Technologies*, pages 142–148, Como,Italy, 2003.

[103] Nathan N Vuong et.al. Managing Security Policies in a Distributed Environment Using eXtensible Markup Language (XML). In *SAC 2001*, pages 405–411, Las Vegas, NV,USA, 2001.

[104] Park,J.S. et.al. Role-Based Access Control on the Web. *ACM Transactions on Information and System Security*, (1):37–71, 2001.

[105] Patz,G. et.al. Multidimensional Security Policy Management for Dynamic Coalitions. In *Network and Distributed Systems Security Conference*, 2001.

[106] Phillips,Jr. et.al. Mobile and Cooperative Systems Information Sharing and Security in Dynamic Coalitions. *University of Connecticut, Technical Report CSE-TR-02-1*, 2002.

[107] Rafae Bhatti et.al. X-GTRBAC:An XML-Based Policy Specification Framework and Architecture for Enterprise-Wide Access Control. *ACM Transactions on Information and System Security*, (2):187–227, 2005.

[108] Sabrina De Capitani et.al. Policies, Models and Languages for Access Control. In *DNIS 2005*, pages 225–237, 2005.

[109] Salem Benferhat et.al. A Possibilistic Logic Encoding of Access Control. In *FLAIRS Conference*, pages 481–485, St.Augustine,Florida,USA, 2003.

[110] SangYeob Na et.al. Role Delegation in Role-Based Access Control. In *RBAC 2000*, Berlin, Germany, 2000.

[111] Sejong Oh et.al. An Effective Role Administration Model Using Organization Structure. *ACM Transactions on Information and System Security*, (2):113–137, 2006.

[112] Steve Barker et.al. Flexible Access Control Policy Specification with Constraint Logic Programming. *ACM Transactions on Information and System Security (TISSEC)*, (4):501–546, 2003.

[113] Tripunitara, M.V. et.al. Expressive Power of Access Control Models. In *ACM Conference on Computer and Communications Security (CCS)*, 2004.

[114] Tuecke, S. et.al. Grid Service Specification. *Global Grid Forum*, 2002.

[115] Vimercati,S.D.C. et.al. Access Control: Principles and Solutions. *Software:Practice and Experience*, (5):397–421, 2003.

[116] Vuong,N.N. et.al. Managing Security Policies in a Distributed Environment Using Extensible Markup Language(XML). In *The 16th ACM SAC2001 Symposium on Applied Computing*, Las Vegas, NV, 2001.

[117] Wang, L. et.al. A Logic Based Framework for Attribute Based Access Control. In *ACM Workshop on Formal Methods in Security Engineering*, Washington DC,USA, 2004.

[118] Weizhong Qiang et.al. A Novel VO-Based Access Control Model for Grid. In *GCC 2004*, pages 293–300, Lake Tahoe, CA,USA, 2004.

[119] Wijesekara, D. et.al. A Propositional Policy Algebra for Access Control. *ACM Transactions on Information and System Security (TISSEC)*, (2):286–325, 2003.

[120] Xinwen Zang et.al. PBDM:A Flexible Delegation Model for RBAC. In *The 8th ACM Symposium on Access Control Models and Technologies*, pages 149–157, Como,Italy, 2003.

[121] Yu,T. et.al. Supporting Structured Credentials and Sensitive Policies Through Interoperable Strategies for Automated Trust Negotiation. *ACM Transactions on Information and System Security (TISSEC)*, (1):1–42, 2003.

[122] Zhang, L.J. et.al. Developing Grid Computing Applications Part 2, Introduction to a Grid Architecture and Toolkit for Building Grid Solutions. *http://www-106.ibm.com/developerworks/views/grid/articles.jsp*, 2002.

[123] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein and Charles E. Youman. Role-Based Access Control Models. In *International Semantic Web Conference*, pages 706–721, Sanibel Island, Florida, USA, 2003.

[124] Ravi S. Sandhu, David F. Ferraiolo and D. Richard Kuhn. The NIST Model for Role-Based Access Control: Towards a Unified Standard. *ACM Workshop on Role-Based Access Control*, pages 47–63, 2000.

[125] Lalana Kagal, Tim Finin and Yun Peng. A Delegation Based Model for Distributed Trust. In *The Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents, International Joint Conferences on Artificial Intelligence*, 2001.

[126] I. Foster. The Grid:A New Infrastructure for 21st Century Science. *Physics Today*, (2):42–47, 2002.

[127] Ian Foster. What is the Grid? A Three Point Checklist. *Grid Today*, 2002.

[128] Von Welch, Frank Siebenlist, Ian T. Foster and John Bresnahan. Security for Grid Services. In *12th International Symposium on High-Performance Distributed Computing*, pages 359–368, Seattle,WA, USA, 2003.

[129] J Gail and Ravi Sandhu. Role Based Authorization Constraints Specification. *ACM Transactions on Information and System Security*, (4), 2000.

[130] P.J. Gill. Getting Down to Business with Enterprise Grid Computing. *Oracle Magazine*, 2003.

[131] Dieter Gollmann. *Computer Security*. John Wiley and Sons Limited, 2006.

[132] T. Grandison and M. Sloman. A Survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.

[133] T. Grandison and M. Sloman. Specifying and Analysing Trust for Internet Applications. In *Second IFIP Conference on e-Commerce, e-Business, e-Government*, Lisbon,Portugal, 2002.

[134] 451 Group. Specifying and Analysing Trust for Internet Applications. In *Grids 2004, Special Report*, New York, 2004.

[135] GWD-R. Globus:A Toolkit-Based Grid Architecture. *Draft-ggf-ogsi-gridservice-33*, 2003.

[136] H. He. A Role-Based Access Control Model for XML Repositories. In *First International Conference on Web Information Systems Engineering*, pages 138–145, 2000.

[137] Hilary.H.Hosmer. Security is fuzzy -Applying the Fuzzy Logic Paradigm to the Multipolicy Paradigm. In *Proceedings of the New Security Paradigms Workshop*, pages 77–86, 1993.

[138] V. Huber. UNICORE:A Grid Computing Environment for Distributed and Parallel Computing. *Lecture Notes in Computer Science*, pages 258–266, 2001.

[139] M Humphrey and M R Thompson. Security Implications of Typical Grid Computing Usage Scenarios. In *International Conference on High Performance Distributed Computing(HPDC)*, California, USA, 2001.

[140] Kai Hwang and Shanshan Song. Trusted Grid Computing with Security Assurance and Resource Optimization. In *17th International Conference on Parallel and Distributed Computing Systems (PDCS-2004)*, San Francisco, CA, USA, 2004.

[141] Marty Humphrey, Keith R Jacksonand and Mary R Thompson. Security for Grids. *Proceedings of the IEEE*, 93(3):644–652, 2005.

[142] Trent Jaeger and Jonathon Tidswell. Practical Safety in Flexible Access Control Models. *ACM Transactions on Information and System Security*, (3), 2001.

[143] James Broberg, Srikumar Venugopal and Rajkumar Buyya. Market-oriented Grids and Utility Computing: The State-of-the-art and Future Directions. *Journal of Grid Computing*, (3):255–276, 2008.

[144] J. Joseph. A Developer's Overview of OGSI and OGSI-Based Grid Computing Get an in-Depth Look at the Open Grid Service Infrastructure. *IBM Archives*, 2003.

[145] Markus Lorch, Dennis G. Kafura and Sumit Shah. An XACML-based Policy Management and Authorization Service for Globus Resources. In *4th International Workshop on Grid Computing*, pages 208–212, Phoenix, AZ, USA, 2003.

[146] Ajith Kamath and Ramiro. User-Credential Based Role Mapping in Multidomain Environment. In *International Conference on Privacy, Security, Trust (PST)*, 2006.

[147] Ali Raza Butt, Sumalatha Adabala, Nirav H Kapadia and Renato Figueiredo. Fine-Grain Access Control for Securing Shared Resources in Computational Grids. In *International Parallel and Distributed Processing Symposium*, Florida, USA, 2002.

[148] G. Karjoth. The Authorization Service for Tivoli Policy Director. In *The 17th Annual Computer Security Applications Conference*, New Orleans,Lousiana,USA, 2001.

[149] Keahey and V. Welch. Fine-Grain Authorization for Resource Management in the Grid. In *Grid 2002 Workshop*, Baltimore, MD, 2002.

[150] A. Kern. Advanced Features for Enterprisewide Role-Based Access Control. In *Proceedings of the 18th Annual Computer Security Applications Conference*, Las Vegas,NV, 2002.

[151] A. Kern and M. Kuhlmann. Observations on th Role Life-Cycle in the Context of Enterprise Security Management. In *The 7th ACM Symposium on Access Control Models and Technologies SACMAT 2002*, Monterey,California,USA, 2002.

[152] Song, K.Hwang and M.Macwan. Fuzzy Trust Integration for Security Enforcement in Grid Computing. In *IFIP International Symposium on Network and Parallel Computing(NPC 2004)*, Wuhan, China, 2004.

[153] Michiharu Kudo. PBAC:Provision-Based Access Control Model. *IJIS*, 2002.

[154] M. Lehmann. Who needs Web Services Transactions? *Oracle Magazine, http://otn.oracle.com/oraclemagazine*, 2003.

[155] 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.

[156] H Liu and J Shen. A Mission-Aware Trust Model for Grid Computing Systems. In *International Workshop on Grid and Cooperative Computing*, Sanya, China, 2002.

[157] Gasser. M. and McDermott E. An Architecture for Practical Delegation in a Distributed System. In *IEEE Symposium on Security and Privacy*, 1990.

[158] Daniel W Manchala. E-Commerce Trust Metrics and Models. *IEEE Internet Computing*, 2000.

[159] Marcos Dias De Assuncao, Rajkumar Buyya and Srikumar Venugopal. InterGrid:A Case for Internetworking Islands of Grids. *Concurrency and Computation:Practice and Experience*, (8):997–1024, 2008.

[160] Brown M.C. Grid Computing - Moving to a Standardized Platform. *IBM Archives, IBM Corporation, www.ibm.com*, 2003.

[161] M. McCommon. Letter from the Grid Computing Editor: Welcome to the New Developer Works Grid Computing Resource! *IBM Grid Computing Resource*, 2003.

[162] R. McRae. The Stanford Model for Access Control Administration. *Stanford University Journal*, 2000.

[163] E. Messmer. Role-Based Access Control on a Roll. *Network World*, 2001.

[164] Daniel Minoli. *A Networking Approach to Grid Computing*. John Wiley and Sons, 2005.

[165] T. Myer. Grid Computing:Conceptual Flyover for Developers. *IBM Corporation*, 2003.

[166] G Geethakumari, Atul Negi and V N Sastry. Dynamic Delegation Approach for Access Control in Grids. In *IEEE Conference on e-Science and Grid Computing*, Melbourne, Australia, 2001.

[167] G. Geethakumari, Atul Negi and V. N. Sastry. Indirect Authorization Topologies for Grid Access Control. In *9th International Conference on Information Technology*, pages 186–187, Bhubaneswar, India, 2006.

[168] G Geethakumari, T L Prasanna Venkatesan, Srikanth Jampala, Atul Negi and V N Sastry. A Ranking Based Cross Domain Role Mapping and Authorization Architecture for Grid Computing Systems. In *International Conference on High Performance Computing (HiPC)*, 2007.

[169] Woodas W.K.Lai, Kam-Wing Ng and Michael R Lyu. Integrating Trust in Grid Computing Systems. In *Grid and Cooperative Computing*, pages 887–890. Springer, 2004.

[170] S Oh and R. Sandhu. A Model for Role Administration using Organization Structure. In *The 7th ACM Symposium on Access Control Models and Technologies SACMAT 2002*, pages 155–168, Monterey,California,USA, 2002.

[171] S. Oh and R. Sandhu. A Model for Role Administration using Organization Structure. In *Seventh ACM Symposium on Access Control Model and Technologies*, pages 155–162, Monterey,CA,USA, 2002.

[172] Xinwen Zhang, Sejong Oh and Ravi Sandhu. PBDM: A Flexible Delegation Model in RBAC. In *ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 149–157, 2003.

[173] R. Osborn, S., Sandhu and Q. Munawer. Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *ACM Transactions on Information and System Security*, 3(2):85–106, 2000.

[174] S Osborn and Y. Gou. Modeling Users in Role-Based Access Control. In *The 5th ACM Workshop on Role-Based Access Control*, pages 31–38, Berlin,Germany, 2000.

[175] M. Otey. Grading Grid Computing. *SQL Server Magazine*, 2004.

[176] C. Papenfus and R. Botha. An XML-Based Approach to Enforcing History Based Separation of Duty Policies in Heterogeneous Workflow Environments. *SACJ/SART*, (26):60–68, 2000.

[177] Rajkumar Buyya and Srikumar Venugopal. A Gentle Introduction to Grid Computing and Technologies. *CSI Communications*, (1):9–19, 2005.

[178] Rajkumar Buyya, David Abramson and Srikumar Venugopal. The Grid Economy. *Proceedings of the IEEE, Special Issue on Grid Computing*, (3):698–714, 2008.

[179] J. Abonyi, J.A. Roubos and M. Setnes. Learning Fuzzy Classification Rules from Labeled Data. *International journal of Information Sciences*, 150(2):77–93, 2003.

[180] T. Ryutov. Representation and Evaluation of Security Policies for Distributed System Services. In *The DARPA Information Survivability Conference and Exposition*, pages 172–183, Hilton Head, South Carolina, 2000.

[181] T. Scavo and S. Cantor. Shibboleth Architecture Technical Overview. *Internet 2 Document: draft-maceshibboleth-tech-overview-02*, 2005.

[182] M.Liebrand, H.Ellis, C.Phillips, S.Demurjian and T.C.Ting. Role Delegation for a Distributed, Unified RBAC/MAC. In *Proceedings of the IFIPWG 11.3 Annual Conference*, Cambridge,U.K, 2002.

[183] Basit Shafiq and James B.D. Joshi. Secure Interoperation in a Multidomain Environment Employing RBAC Policies. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):644–652, 2005.

[184] N. Shankar and W.A. Arbaugh. On Trust for Ubiquitous Computing. *Department of Computer Science, University of Maryland College Park, M.D, USA*, 2002.

[185] W.B. Shim and S. Park. Implementing Web Access Control System for Multiple Web Servers in the Same Domain Using RBAC Concept. In *8th International Conference on Parallel and Distributed Systems (ICAPDS)*, pages 768–773, 2001.

[186] Pietro Mazzoleni, Bruno Crispo, Swaminathan Sivasubramanian and Elisa Bertino. Efficient Integration of Fine-grained Access Control in Large scale Grid Services. In *IEEE International Conference on Services Computing*, pages 77–86, Orlando, FL, USA, 2005.

[187] M.R.Thompson, A.Essiari, K.Keahey, V.Welch, S.Lang and B.Liu. Fine-Grained Authorization for Job and Resource Management Using Akenti and the Globus Toolkit. In *International Parallel and Distributed Processing Symposium (CHEP 2003)*, La Jolla, California, USA, 2003.

[188] David F Snelling and Sven van den Berghe. Explicit Trust Delegation: Security for Dynamic Grids. *FUJITSU Sci.Tech.Journal*, 40(2):282–294, 2004.

[189] DeveloperWorks Staff. Start Here to Learn about Grid Computing. *IBM Developer Works*, 2003.

[190] The Globus Security Team. Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective. *Technical Report, Globus Consortium*, 2005.

[191] J.E. Tidswell and T. Jaeger. Integrated Constraints and Inheritance in DTAC. In *The Fifth ACM Workshop on Role-Based Access Control*, pages 93–102, Berlin, Germany, 2000.

[192] J. Unger. A Visual Tour of Open Grid Services Architecture: Examine The Component Structure of OGSA. *IBM Archives*, 2003.

[193] Chris Tseng, Toan Vu and Wafa Khamisy. Universal Fuzzy System Representation with XML. *Computer Standards and Interfaces Journal*, 2005.

[194] H.F. Wedde and M. Lischka. Modular Authorization. In *The 6th ACM Symposium on Access Control Models and Technologies(SACMAT 2001*, pages 97–105, Chantilly,Virginia,USA, 2001.

[195] H.F. Wedde and M. Lischka. Composing Heterogeneous Access Policies Between Organizations. In *The IADIS International Conference e-Society 2003*, Lisbon,Portugal, 2003.

[196] Laura Pearlman, Von Welch and Ian Foster. A Community Authorization Service for Group Collaboration. In *IEEE 3rd. International Workshop on Policies for Distributed Systems and Networks*, pages 50–59, 2002.

[197] Dietmar W.Erwin and David F.Snelling. UNICORE: A Grid Computing Environment. In *The Euro-Par Conference*, pages 825–834, 2001.

[198] Bo Yuan and George J.Klir. *Fuzzy Sets and Fuzzy Logic - Theory and Applications*. Prentice Hall of India, 2003.

[199] L. Zadeh. Knowledge Representation in Fuzzy Logic. *IEEE Transactions on Knowledge Representation in Fuzzy Logic and Data Engineering*, pages 89–100, 1989.

[200] L.J, Gail-Joon Ahn Zhang and Bei-Tseng Chu. A Rule-Based Framework for Role-Based Delegation and Revocation. *ACM Transactions on Information and System Security*, 6(3):404–441, 2003.

[201] Xinwen Zhang and Masayuki Nakae. A Usage-based Authorization Framework for Collaborative Computing Systems. In *SACMAT 2006*, Lake Tahoe, California, USA, 2006.

[202] Weizhong Qiang, Hai Jin, Xuanhua Shi, Deqing Zou and Hao Zhang. RBGACA: A RBAC Based Grid Access Control Architecture. *International Journal of Grid and Utility Computing*, pages 61–70, 2005.

APPENDIX A

GLOBUS TOOLKIT

A.1 Installation of Globus Toolkit

This helps as a guide to Globus Toolkit 4.0.1 (GTK4.0.1) [135] installation. It deals with basic installation that installs core services and a few base services namely: a security infrastructure (GSI), GridFTP and RFT. After the installation and testing,we explore the possibility of using the CA of one machine on another machine. We also share the experiences and difficulties we faced during the install of Globus.

1. **Pre-requisites (Softwares/Databases)**
   - **Java**
   - **Apache Ant**
   - **Apache tomcat**
   - **Postgresql database**
2. **Optional Software**
   - **IODBC**

   Before starting the installation, first we need to **create a globus user**.
3. **Building the Toolkit**
4. **Setting up the simpleCA**
5. **Get the host and container certificates signed by simpleCA**
6. **Get valid credentials from the CA**
7. **Create grid-map file and add entries to it**
8. **Configure GridFTP**
9. **Install postgresql and configure RFT**
10. **Start the Globus container and test using the globus client**

A.2 Writing a Grid service

Writing and deploying a grid service is done by following the five simple steps. We consider a sample MathService to explain the steps.

1. **Defining the interface in WSDL:** The first step in writing a grid service is to define the service interface. We need to specify what our service is going to provide to the outer world.

2. **Implementing the service in Java:** After defining the service interface [154] the next step is implementing that interface. The implementation is ”how the service does what it says it does”.

3. **Configuring the deployment in WSDD:** Up to this point, we have written the two most important parts of our stateful Web service: the service interface (WSDL) and the service implementation (Java) [59], [40]. Next step will actually take all the loose pieces we have written up to this point and make them available through a Web services container. This step is called the deployment of the web service.

4. **Create a GAR file with Ant:** At this point we have

   (a) a service interface in WSDL

   (b) a service implementation in Java and

   (c) a deployment descriptor in WSDD

   Using these three files we will generate a Grid Archive, or GAR file. Ant (java build tool) is used to create GAR file

5. **Deploy the service into a Web Services container:** The GAR file, contains all the files and information the web server needs to deploy the web service. Deployment is done with a GT4 tool that, using Ant, unpacks the GAR file and copies the files within (WSDL, compiled stubs, compiled implementation, WSDD) into key locations in the GT4 directory tree.

The Globus Toolkit installation is a prerequisite to the implementation of models as we need to simulate a grid environment. In section A.1, we have listed the software/database requirements. In section A.2, we have shown the procedure for writing and deploying a simple Mathservice application. The implementation shown in Chapter 7 of the thesis is based on these concepts.

APPENDIX B

EXTENSIBLE ACCESS CONTROL MARKUP LANGUAGE

B.1 A Brief Introduction to XACML

eXtensible Access Control Markup Language (XACML) is an OASIS [34] initiative to standardize representation of access control policies in a flexible, extensible XML format.XACML has two basic components.

- The first is an access control policy language that lets specify the rules about who can do what and when.
- The other is a request/response language that presents requests for access and describes the answers to those queries.

XACML provides for fine-grained control of activites based on several criteria, including the following:

- Attributes of the user requesting access.
- The protocol over which the request is made.
- The authentication mechanism.

Figure B.1 shows the architectural framework for XACML.

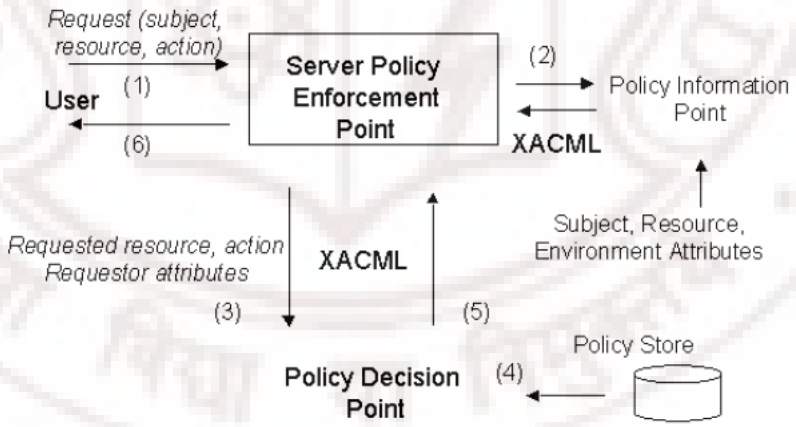


Figure 48: XACML Framework

B.2 Strengths of XACML

XACML was designed to replace existing, usually application-specific, proprietary access-control mechanisms. Advantages of XACML are:

- It is standard. As XACML becomes more widely deployed, it will be easier to interoperate with other applications using the same standard language.
- It is generic. XACML can accommodate most access control policy needs and also support new requirements as they emerge.
- It is distributed. This means that a policy can be written which in turn refers to other policies kept in arbitrary locations.
- It is powerful. While there are many ways the base language can be extended. The standard language already supports a wide variety of data types, functions, and rules about combining the results of different policies.

B.3 XACML Policy Language Model

The main components are:

- **Rule**
- **Policy** and
- **PolicySet**

The XACML Policy Language Model is depicted in Figure B.3 **Rule** A rule is the most elementary unit of policy. A rule can be evaluated on the basis of its contents. The main components of a rule are:

- Target: defines set of
  - resources
  - subjects
  - actions and
  - environment
- Effect: Two values are allowed
  - Permit
  - Deny
- Condition: represents a Boolean expression that refines the applicablity of the rule beyond the predicates implied by its target.

Policy A policy comprises four main components

- Target

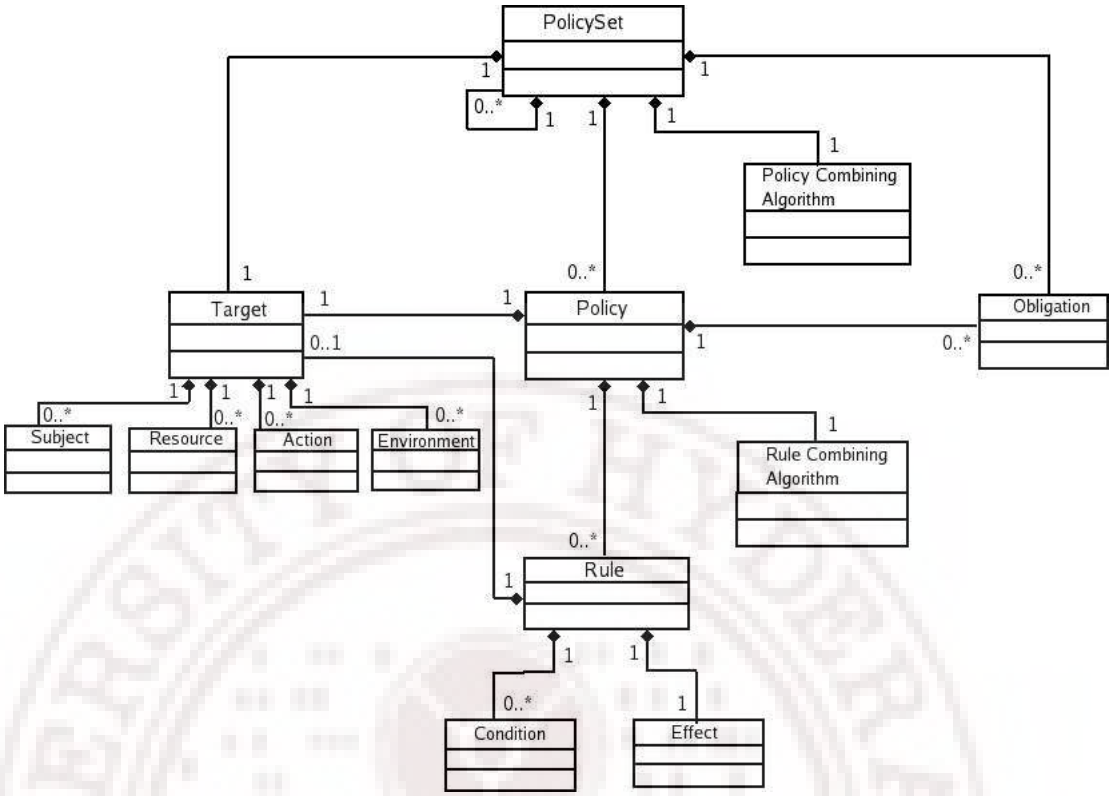


Figure 49: XACML Policy Language Model

- Rule-combining algorithm: specifies the procedure by which the results of evaluating the component rules are combined when evaluating the policy.
- set of rules and
- obligations

PolicySet A PolicySet comprises four main components

- a target
- a policy-combining algorithm identifier A policy-combining algorithm specifies the procedure by which the results of evaluating the component policies are combined when evaluting the policyset.
- a set of policies and
- obligations

This appendix gives a brief introduction to eXtensible Access Control Markup Language (XACML). We have used the XACML Policy Language Model for formulating access control policies. This is discussed in Chapter 7 of the thesis. The implementation of single-domain and multi-domain authorization has been based on the XACML framework.

APPENDIX C

LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL (LDAP)

C.1 Introduction

A directory is a specialized database specifically designed for searching and browsing, in addition to supporting basic lookup and update functions. A directory is defined by some as merely a database optimized for read access. LDAP stands for Lightweight Directory Access Protocol. LDAP is a set of open protocols used to access centrally stored information over a network. It is based on the X.500 standard for directory sharing, but is less complex and resource intensive. The LDAP information model is based on entries. An entry is a collection of attributes that has a globally-unique Distinguished Name (DN). The DN is used to refer to the entry unambiguously. Each of the entry's attributes has a type and one or more values. The types are typically mnemonic strings, like "cn" for common name, or "mail" for email address. The syntax of values depends on the attribute type. LDAP defines operations for interrogating and updating the directory. Operations are provided for adding and deleting an entry from the directory, changing an existing entry, and changing the name of an entry. But most of the time, LDAP is used to search for information in the directory. The LDAP search operation allows some portion of the directory to be searched for entries that match some criteria specified by a search filter. Information can be requested from each entry that matches the criteria. LDAP also provides a mechanism for a client to authenticate, or prove its identity to a directory server, paving the way for rich access control to protect the information the server contains.

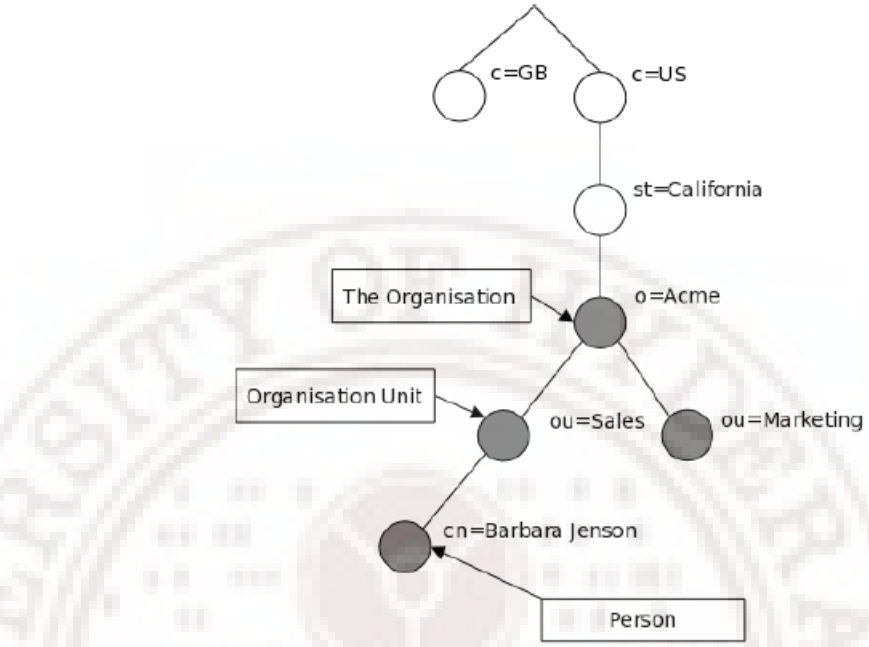
C.1.1 LDAP Directory Structure

In LDAP, directory entries are arranged in a hierarchical tree-like structure. Traditionally, this structure reflected the geographic and/or organizational boundaries. Entries representing countries appear at the top of the tree. Below them are entries representing states and national organizations. Below them might be entries representing organizational units, people etc. The LDAP directory structure is as depicted in Figure 50.

C.1.2 LDAP Installation and Configuration

The following are the steps to install and configure LDAP in grids.

```
1. tar -xvzf openldap-2.3.0.tgz
2. ./configure --prefix=/usr
--exec-prefix=/usr --bindir=/usr/bin
--sbindir=/usr/sbin \
```



The tree may also be arranged based upon Internet domain names.

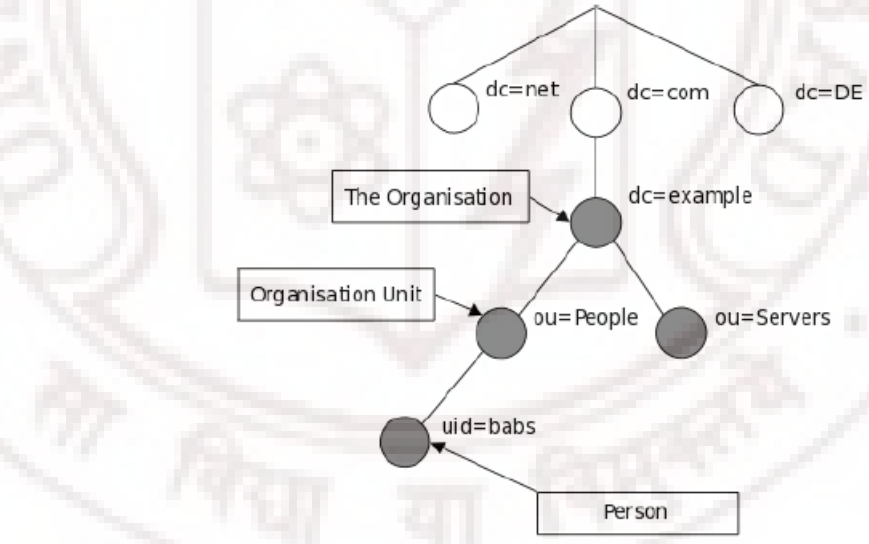


Figure 50: LDAP Directory Structure

```
--sysconfdir=/etc --datadir=/usr/share --localstatedir=/var \
--mandir=/usr/share/man --infodir=/usr/share/info --enable-sql
3. make depend   to build dependencies
4. make    compiles openLDAP
5. make install   installs openLDAP
Starting LDAP: slapd is run like this (default port number
 is 389)
$ /usr/local/etc/libexec/slapd
We can change the slapd.conf file according to our needs.
Then we populate the directory by adding the entries
which are represented in LDIF (LDAP Data Interchange Format).
```

Default OpenLDAP installs two daemons in /usr/local/libexec/. They are:

- slapd - stand-alone LDAP daemon (server)
  Slapd is an LDAP directory server that runs on many different UNIX platforms. The directory can contain pretty much anything we want to put in it. It can connect it to the global LDAP directory service, or run a service all by itself.
- slurpd - stand-alone LDAP update replication daemon
  Slurpd is a UNIX daemon that helps slapd provides replicated service. It is responsible for distributing changes made to the master slapd database out to the various slapd replicas. Slapd and slurpd communicate through a simple text file that is used to log changes.

Before using LDAP we have to make the following changes to the slapd.conf file.

1. First we include more schema files depending upon application needs. By default LDAP includes core.schema file.
2. We then specify the access control policy regarding who is authorized to read, write and modify the entries in the ldap directory.
3. We have to set the suffix and rootDN for the database which will be used at the backend. By default the root password is "secret" and may be changed.

The slapd.conf file consists of three types of configuration information: global, backend-specific, and database-specific. Global information is specified first, followed by information associated with a particular backend type, which is then followed by information associated with a particular database instance.

Schema Specifications

OpenLDAP is distributed with a set of schema specifications for one's use. Each set is defined in a file suitable for inclusion. These schema files are normally installed in the /usr/local/etc/openldap/schema directory. To use the schema specifications, we need to include the desired file in the global definition portion of slapd.conf file. The schema used by slapd may be extended to support additional syntaxes, matching rules, attribute types, and object classes. There are five steps to defining new schema:

1. Obtain Object Identifer: Each schema element is identified by a globally unique Object Identifier (OID). OIDs are also used to identify other objects. They are commonly found in protocols described by ASN.

2. Choose a name prefix : There should be atleast one textual name for each element. The name should be both descriptive and not likely to clash with names of other schema elements.

3. Create local schema file : The objectclass and attributeTypes configuration file directives can be used to define schema rules on entries in the directory. We create a file to contain definitions of the custom schema items and then include this file in the slapd.conf file.

4. Define custom attribute types: The attributetype directive is used to define a new attributetype. attributetype (RFC2252 Attribute Type Description)

5. Define custom object classes: The objectclasses directive is used to define a new object class. objectclass (RFC2252 Object Class Description).

C.1.3 LDIF

The LDAP Data Interchange Format (LDIF) is a standard data interchange format for representing LDAP directory content as well as directory update (Add, Modify, Delete, Rename) requests. Each content record is represented as a group of attributes, with records separated from one another by blank lines. The individual attributes of a record are represented as single logical lines (represented as one or more multiple physical lines via a line-folding mechanism), comprising "name: value" pairs. The OpenLDAP utility tools are used exporting data from LDAP servers to LDIF content records (ldapsearch), importing data from LDIF content records to LDAP servers (ldapadd), and applying LDIF change records to LDAP servers (ldapmodify). The main disadvantage of LDIF is that values in multi-valued attributes cannot be replaced directly. We have to delete the attribute values and then use "add:" multiple times to feed all of the required values.

Appendix C deals with the design and implementation of an LDAP (Lightweight Directory Access Protocol) and Hierarchy for a grid environment. The LDAP concept is used in Chapter 7 of the thesis dealing with implemention. We discuss the LDAP Directory Structure as well as the LDAP Data Interchange Format.

LIST OF PUBLICATIONS

1. G Geethakumari, Atul Negi and V N Sastry, "RBGDM- A Role Based Grid Delegation Model", International Journal of Computer Science and Security, CSS Journals Publishers, ISSN: 1985-1533, Vol 2, Issue 1, April 2008, pp 61 - 72.
   http://www.cscjournals.org/Journals/IJCSS/Volume2/Issue1/IJCSS-29.pdf

2. G Geethakumari, Atul Negi and V N Sastry, "A Cross-Domain Role Mapping and Authorization Framework for RBAC in Grid Systems", International Journal of Computer Science and Applications, Special Issue on "Grid and Parallel Computing" (IJCSA), ISSN 0972-9038, Vol 6, No.1, January 2009, pp 01-12.
   http://www.tmrfindia.org/ijcsa/v6i11.pdf

3. G Geethakumari, Atul Negi and V N Sastry, "Dynamic Delegation Approach for Access Control in Grids", Proceedings of the IEEE International Conference on e-Science and Grid Computing (e-Science2005), Australia, Dec 2005, pp 387-394.
   http://portal.acm.org/citation.cfm?id=1107836.1107896
   http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/10501/33262/01572249.pdf?temp=x

4. G Geethakumari, Atul Negi and V N Sastry, "A Fine-Grained Access Framework for Grids using Fuzzy Inference Systems", Proceedings of the Mexican Conference on Information Security, MCIS 2006, Mexico (CD Format), November 13-17, 2006.

5. G Geethakumari, Atul Negi and V N Sastry, "Grid Security Through Delegation of Roles", Proceedings of the IEEE TENCON 2006, Hong-Kong, November 2006, pp 1 - 4.
   http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4142293

6. G Geethakumari, Atul Negi and V N Sastry, "Indirect Authorization Topologies for Grid Access Control", Proceedings of the International Conference on Information Technology (ICIT) 2006, Bhubaneswar, India, December 2006, pp 186 - 187.
   http://portal.acm.org/citation.cfm?id=1193210.1193684
   http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4273187

7. G Geethakumari, Prasanna Venkatesan, Srikanth Jampala, Atul Negi and V N Sastry, "Role-Based Access control Architecture for Delegation in Multi Domain Grid Environments", IEEE International Conference on e-Science and Grid Computing (e-Science 2007 - Poster Presentation), Bangalore, India, December 2007.

8. G Geethakumari, Prasanna Venkatesan, Srikanth Jampala, Atul Negi and V N Sastry, “A Ranking Based Cross Domain Role Mapping and Authorization Architecture for Grid Computing Systems”, IEEE High-Performance Computing Conference (HiPC 2007 - Poster Presentation),Goa, India, December 2007.

9. G Geethakumari, Srikanth Jampala, T.L.Prasanna Venkatesan, Atul Negi and V.N.Sastry, “Multi-Domain Delegation and Revocation Model for Grid Systems”, Proceedings of the IEEE TENCON 2008 Conference (CD Format), November 18-21, Hyderabad, India, pp 1 - 6.
http://ieeexplore.ieee.org/xlps/abs_all.jsp?isnumber=4766377

VITA

Name:
G Geethakumari
Educational Qualifications:
B Tech - Reginal Engineering College, Calicut, India
M Tech - Jawaharlal Nehru Technological University, Hyderabad, India
PhD (Computer Science) Under Progress - University of Hyderabad, India
Professional Experience:
July 2008 - Till date : BITS Pilani, Hyderabad Campus, India
Nov 2006 - June 2008 : NIT, Warangal, India
Sept 2004 - Nov 2006 : IDRBT, Hyderabad, India
June 2002 - Sept 2004 : BRECW, Hyderabad, India
Areas of Interest:
Information Security, Access Control Modelling, Grid Computing, Computer Networks, Operating Systems
Awards and Honors:
1.Co-Chief Investigator, ISEAP (Sponsored by MHRD - in the year 2007), NIT, Warangal, India.
2.Program Committee Member, IEEE e-Science and Grid Computing Conference, 2006.
3.Program Committee Member, ICCS-2007
4.Program Committee Member, ICCS-2008
5.Program Committee Member, ICCS-2009
Membership in Professional Bodies:
Member, IEEE
Seminars/Workshops Organized:
(1) IBM Technology Day workshop at NIT, Warangal, India during August 17-18th, 2007 as part of the IBM Technology day celebrations. The two day workshop covered a wide range of topics like Linux Kernel Programming and Autonomic Computing. (2) Workshop on Foundations of Information Security at NIT, Warangal, India during October 1-5th, 2007.

e-mail ID: geethamaruvada@gmail.com